UNIVERSITY OF BIRMINGHAM

DOCTORAL THESIS

The Hierarchical Organisation and Dynamics of Complex Networks

---

Author:

David MCDONALD

Supervisor:

Dr. Shan HE

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

in the

Department of Computer Science

UNIVERSITY OF BIRMINGHAM

Abstract

School of Computer Science

Department of Computer Science

Doctor of Philosophy

The Hierarchical Organisation and Dynamics of Complex Networks

by David MCDONALD

Complex networks offer flexible representations of complex heterogeneous real-world systems. They are often *weighted*, *attributed*, *directed* and/or *dynamic*. As such, gaining an overall understanding of information flow through these systems remains a challenging problem in the machine learning community. This thesis provides a comprehensive examination of the hierarchy inherent to many complex networks, with the following contributions:

- The first algorithm to learn low dimensional non-Euclidean representations of attributed nodes in a weighted complex network.
- The first algorithm to learn low dimensional non-Euclidean representations of attributed nodes in a directed complex network.
- A framework to explore the multi-scale organization of meso-scopic architectures in signalling networks, allowing for the identification of statistically significant drug-able targets.

Through these contributions, the work proposed in this thesis contributes towards a greater understanding of the hierarchy in the organization and dynamics of complex real-world systems.

Acknowledgements

I would like to take this opportunity to thank my supervisor Dr. Shan He. His advice and expertise has been invaluable over the course of the PhD. I have found his knowledge and enthusiasm infectious. Without his constant guidance and support, I would not have been able to produce this thesis. I would also like to thank him for all of his help outside of the PhD.

I would like to thank my thesis group members Dr. Peter Tino and Dr. David Parker for their constructive comments and continuous support during my PhD study.

I wish to thank Dr. Tim Evans from Imperial College London and Dr. Hyung Jin Chang for reading this thesis and being the examiners of my PhD viva. I found both of their comments and feedback to be extremely useful and insightful and they are responsible for helping to get this thesis into its best form.

Next, I wish to extend my thanks to Dr. Manfred Kerber, Alexandros Evangelidis and Dr. Uday Reddy for providing great mentorship in my academic career outside of research. Through them, I have learnt a great many skills related to small-group teaching.

I would now like to thank my friends and colleagues Han Zhang and Sami Cass Darweish. Their support, friendship, and ability to challenge have helped to grow me into a better researcher. As well, they have always been around to provide a much needed distraction.

Next, I would like to thank both my family and my fiancée's family for their constant encouragement and support over this challenging time. I could not have done this without them and am so grateful for their time and patience.

Finally, I would like to thank my fiancée Dora Emese Pop. Her support over this period is the reason that I was able to do this PhD at all. This thesis is dedicated to her.

Contents

List of Figures

List of Tables

Dedicated to Dora Emese Pop

Chapter 1

Introduction

1.1 Complex Networks

Throughout our world, we observe complex systems – groups of *elements* that connect to each other through *relations* in a non-uniform way. Through these relations, these elements are able to work together and function as a coherent whole that is greater than the sum of its parts. We see this in the simple relationships amongst people that form an entire society; in the iterations between genes, proteins and metabolites that form a living organism; and in the links between pages that make up the internet. Within these systems, interactions are not controlled globally, but emerge locally based on some local organisation that gives rise to new levels of organisation. In this way, we see that the organisation of complex systems is *hierarchical*: elements belong to many different systems on many different scales, with all the levels affecting each other (Barabási and Albert, 1999).

Many complex systems observed in nature can be represented as a *complex network*. A complex network is a graph that is comprised of non-trivial and non-uniform features (Barabási and Albert, 1999). Despite being characterised as complex objects, we have observed that many real world networks – ranging from protein interaction networks to the internet – possess shared features that unite such seemingly disparate subject and give rise to the prevailing popularity of their study. For example, we have observed that these complex networks follow scale-free distribution of node degree (Barabási and Albert, 1999; Barabási, 2009). Connections are preferentially made between nodes with a probability proportional to their existing degree, giving rise to the so-called preferential attachment model, a model

that adheres to the old adage 'popularity is attractive'. Furthermore, many complex networks are characterised by the 'small world' phenomenon – where one would expect a small average shortest path length and a high degree of clustering (Watts and Strogatz, 1998); and are typically very sparse with the number of edges in the same order as the number of nodes Barabási and Albert, 1999.

Already, we have seen the application of complex network models to the problems of identification of biologically relevant modules according to genetic expression profiles (Ideker et al., 2002); the multi scale detection of the function and structure of brain networks (Ashourvan et al., 2019); and testable protein functional predictions (Palla et al., 2005; Zhang et al., 2016b). And with the growing availability of data and drive of the network science community, models are becoming more and more complex, with the integration of multiple types of data and more sophisticated analysis techniques.

Network science is the interdisciplinary endeavour of making sense of the complex networks that we observe in nature. It unites scientists from such varied fields as mathematics, computer science, medicine, biology and sociology. Researchers from all these fields and more have contributed their expertise and perspective. This section aims to give a brief overview of some of the interesting areas of research happening in this field. For the purpose of brevity, the review presented here will only focus on the fundamental concepts that will prove to be relevant for this thesis. For a more general overview of network science, the reader is pointed towards Barabási, 2002.

To fully understand a network's function, we must consider both its structure and the dynamics that take place upon it (Lambiotte et al., 2011). Its structure is characterised by its topology: the sets of features defined by purely by the vertices and edges of the network, discounting any additional prior knowledge that we may have about what system the network is modelling. For example, we may observe a high degree of clustering, short paths between any two nodes in the network, and a heterogeneous degree distribution. Such features are ubiquitous across real-world systems (Barabási and Albert, 1999). On the other hand, dynamics represent the systems at work, and are specific to that system. For example, how the gene expression levels in patients with breast cancer change over time.

1.1.1 Complex Network Modelling

A complex network, in its simplest form, is typically modelled as a graph:

$$\mathbb{G} = (V, E) \tag{1.1}$$

V denotes the set of *entities* – called nodes – in the network. $E \subseteq V \times V$ is the set of *relations* – or edges – between them. The number of nodes in a network is given by:

$$N = |V| \tag{1.2}$$

and the number of edges m is:

$$m = |E| \tag{1.3}$$

Directed Networks

Complex networks may be *undirected*, meaning that edges do not carry direction, or *directed*, meaning the opposite. Concretely, if a network is undirected then for two nodes $u, v \in V$, $(u, v) \in E \implies (v, u) \in E$, whereas $(u, v) \in E \not\Longrightarrow (v, u) \in E$ for the directed case. Directed networks are sometimes called *di-graphs*.

Self-loops

Self-loops are edges that connect nodes to themselves. The set of self-loops is accordingly defined as:

$$E^{\text{self}} = \{(u, v) \in E \mid u = v\} \tag{1.4}$$

Adjacency Matrix

An alternate complex network representation is the *adjacency matrix*. An adjacency matrix $\mathbf{A}$ is an $N \times N$ square matrix, where the elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph:

$$\mathbf{A}_{uv} \neq 0 \iff (u, v) \in E \tag{1.5}$$

For a simple undirected network, $\mathbf{A}$ is a symmetric $\{0,1\}^{N\times N}$ matrix. The relationship between a graph and the eigen-values and eigen-vectors of its adjacency matrix is studied in spectral graph theory (Von Luxburg, 2007).

1.1.2 Attributed Networks

Often the nodes and edges of complex networks are annotated with additional information, called *attributes*. The propose of the node attributes is to provide further information about the elements within a system. Likewise, edge attributes can better characterise the relations between nodes. Interpretation of attributes is entirely dependent of the system that the network is modelling. For example, attributes may be used to describe node document contents or labels in citation network (Bojchevski and Günnemann, 2018); location and relationship type in social networks (Kipf and Welling, 2016); or node type in heterogeneous networks (Chang et al., 2015). The extra information provided by attributes can provide an extra insight into the system as a whole (Hou, He, and Tang, 2020).

1.1.3 Weighted Networks

A special case of edge attributes are edge *weights*. This weight can have a variety of meanings, but, when the value is positive, is typically taken to be the strength, intensity, or the capacity of the relationships between nodes (Barrat et al., 2004).

Many concepts generalise to weighted graphs. For example, the adjacency matrix becomes a *weighted adjacency matrix*, $\mathbf{W}$, where each element in $\mathbf{W}$ contains the weight of the edge between two nodes.

1.1.4 Signed Networks

Another special case of edge-attributed networks are *signed* networks. Signed network are networks where each edge is assigned a positive or negative sign. Signed networks are particularly useful for encoding the topology of regulatory and signalling networks – where edges representing an activatory relationship are assigned a positive sign, and inhibitory relationships are labelled with a negative sign. Signed

networks can also be used to represent the topology of boolean networks (see section 1.5.6).

1.1.5 Bi-Partite Networks

A bi-partite network is a class of attributed network, where nodes are assigned to one of two groups, V_1 and V_2, where $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \phi$. Every edge in the network connects a node from group V_1 to a node in V_2. An example bi-partite network is a metabolic network, where V_1 is the set of metabolite and V_2 is the set of reactions. An edge between metabolite $m \in V_1$ and reaction $r \in V_2$ indicates that m takes part in reaction r (Christensen and Nielsen, 1999).

1.1.6 General Complex Networks

Note that, in general, all complex networks can be considered attributed, weighted networks with no loss of information. In the case of unweighted networks, we can simply set $\mathbf{W} = \mathbf{A}$, that is: set the weight of all edges in the network to 1. Nodes in unattributed networks can be assigned one-hot feature vectors to trivially make them attributed. Furthermore, node features may be composed from micro-scale topological features, such as degree, or betweenness centrality (see section 1.2).

1.1.7 Basic Network Concepts

This subsection introduces some basic concepts and definitions related to networks that will be referred to throughout the thesis.

Neighbours

The *neighbours* of a node u are all the nodes $v \in V$ that are adjacent to u. That is: v is a neighbour of $u \iff (u, v) \in E$. An *isolated* node is a node with no neighbours.

Paths

A path between two nodes u and v is a node sequence of some arbitrary length l $n_1, ..., n_l$ beginning with node $n_1 = u$ and ending with node $n_l = v$. The sequence is built using the edges in E such that: $\forall i \in [0, l-1], (n_i, n_{i+1}) \in E$.

The shortest path, $\mathrm{SP}(u, v)$, between a pair of nodes $u, v \in V$ is the minimum number of edges connecting them. In other words, it is the path connecting nodes u and v with minimum length. This is sometimes called a *geodesic*, and indicates the "closeness" of arbitrary pairs of nodes in the network – according the the manifold described by that network (Tenenbaum, De Silva, and Langford, 2000).

For unweighted networks, *shortest path length* is the number of nodes in the shortest path between two nodes. In weighted networks, shortest path lengths are determined by the total sum of edge weights in a path, rather than the number of elements.

Reachability

Reachability refers to the ability to traverse from one vertex to another within a network. A vertex s can reach a vertex t if there exists a path which starts at s and ends at t.

Cycle / Feedback Loop

A *cycle* $C = (u, ..., u)$ (that we refer to as a feedback loop in this thesis) is a non-empty path where the start and end nodes are the same, and the only repeated node in the path is the start node. For a signed network, we further define a *coherent feedback loop* as a cycle where every edge has the same sign.

Random Walks

A random walk across the nodes of a network is a sampling method for network-based data. It is highly related to the concept of a 'path'. The key difference is that random walks do not have a specific destination in mind.

A random walk starting from node u is a stochastic process based on the idea of a walker walking randomly over of the nodes of the network. It is used to build a sequence of nodes $n_1, ..., n_l$ of length l where $n_1 = u$. The random walk process is iterative and for iteration i with $n_i = v$ being the current position of the random walker, the position of the walker for iteration $i + 1$ is a random choice based on

any number of factors. The factors may include: first order connection (Perozzi, Al-Rfou, and Skiena, 2014), higher order connection (Grover and Leskovec, 2016), or node or edge attributes (Hou, He, and Tang, 2020), for example. Random walking is typically a Markovian process, but that is not a requirement. Random walking can be used to explain the movement of molecules according to Brownian motion.

Subnetworks

A subnetwork $\mathbb{S} = (V_S, E_S)$ of a network $\mathbb{G} = (V, E)$ is a network whose vertex set is a subset of the vertex set of $\mathbb{G}$ and whose edge set is a subset of the edge set of $\mathbb{G}$. In this case, we write:

$$\mathbb{S} \subseteq \mathbb{G} \tag{1.6}$$

and

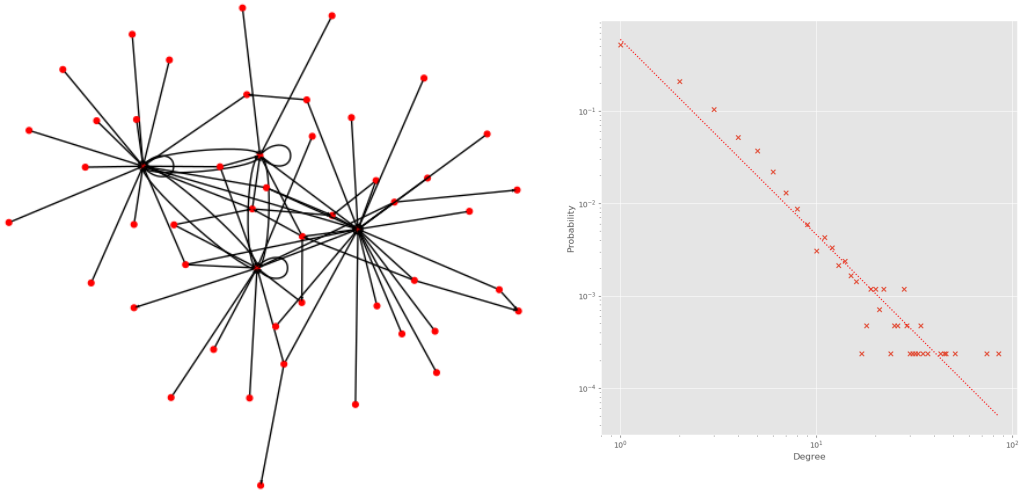
$$\mathbb{S} \subseteq \mathbb{G} \iff V_S \subseteq V \wedge E_S \subseteq E \tag{1.7}$$

holds.

Connectedness

An undirected network is said to be *connected* if a path exists between all nodes in the network. If a network is not connected, then it can be broken into *connected components*, disjoint subnetworks of the original network that are connected.

These concepts naturally generalise to directed networks. If a path exists between any node pair in a directed network, then that network is *strongly connected*. A directed network is *weakly connected* if the undirected equivalent of the network is connected. Of course, all strongly connected networks are also weakly connected. Furthermore, directed networks that are not weakly or strongly connected can be broken up into weakly or strongly connected components.



(A) Example scale-free directed network with $N = 50$ nodes.

(B) Example of a scale-free degree distribution.

FIGURE 1.1: Examples of both a scale-free network and a log-log plot of degree against frequency in the network. (A) shows a small scale-free network with $N = 50$ nodes. This network contains 4 'hubs' of high degree. (B) plots degree against frequency for a larger scale-free network. Note that both axes are on a logarithmic scale. The red line plots a straight line of best fit. The close adherence to this line of best fit reflects how well the underlying network fits a scale-free distribution.

1.1.8 Characteristics of Real-World Complex Networks

Complex networks built from real-world complex systems often display a number of common characteristics, despite coming from diverse areas of study. Such characteristics include a heavy tail in the degree distribution, a high clustering coefficient, assortativity or disassortativity among vertices, community structure, and hierarchical structure (Barabási, 2002).

Scale Free Degree Distribution

In addition to connecting to similar nodes, nodes in complex networks often connect to 'popular' nodes – that is nodes of high degree. In fact, the variation between high assortativity and high dis-assortivity across complex networks from different areas implies some implicit trade off between connections made from popularity vs connections made from similarity (Papadopoulos et al., 2012).

The preference for nodes to connect to nodes of high degree gives rise to *scale-free* networks. A network is said to be scale-free if its degree distribution – the probability that a node selected uniformly at random has a certain number of links – follows

a mathematical function called a power law. Characteristically, scale free networks contain many nodes with small degree and a small number of nodes with degree that is orders of magnitude larger than the average - these vertices are often called "hubs" (Barabási and Albert, 1999). See figure 1.1 for examples of a scale-free network and a scale-free degree distribution.

It is worth noting that, by definition, there is no inherent threshold above which a node can be viewed as a hub. If there were a hard threshold to determine a hub, the network would not be scale-free. This is because, as the name implies, there is no inherent scale to scale free networks, meaning that "zooming in" or "zooming out" on the network would not change the degree distribution (Barabási, 2002).

Some networks with a power-law degree distribution (and specific other types of structure) can be highly resistant to the random deletion of vertices (Cohen et al., 2000).

Important examples of scale-free networks include the internet (Adamic et al., 2000), social networks (Leskovec and Mcauley, 2012), and the hierarchy of nouns (Nickel and Kiela, 2017).

Small World Networks

A network is said to be a *small-world* network by analogy with the small-world hypothesis (popularly known as six degrees of separation). The small world hypothesis is the idea that two arbitrary people in the social network of the world are connected by only six degrees of separation. In other words, that the diameter of this social network is not much larger than six. This property was, famously, tested experimentally by Stanley Milgram in 1967 (Milgram, 1967).

Furthermore, it has been demonstrated that, with the addition of only a small number of long-range links, a regular graph, in which the diameter is proportional to the size of the network, can be transformed into a "small world" in which the average number of edges between any two vertices is very small (mathematically, it should grow as the logarithm of the size of the network), while the clustering co-efficient stays large. It is known that a wide variety of abstract graphs exhibit the small-world property, for example, random graphs and scale-free networks (Barabási and Albert, 1999). Furthermore, real world networks such as the World

Wide Web and the metabolic network also exhibit this property (Krioukov et al., 2009).

Assortativity

Assortativity is the property that nodes like to connect to similar nodes. The definition of similarity often varies, but a common mixing measure is node degree. Nodes tend to connect with nodes of similar degree.

Interestingly, technological and biological networks typically show dis-assortative mixing, or disassortativity, as high degree nodes tend to attach to low degree nodes (Newman, 2003).

1.2 Microscopic Network Features

When characterising complex networks, we commonly examine the micro-scale – considering each individual node and edge separately. Features such as node degree, betweenness centrality and clustering co-efficient would be characterised as micro-scale complex network features, for example.

1.2.1 Node Degree

The *degree* k_n of node n is the number of edges incident to it:

$$k_n = |\{(u, v) \in E \mid u = n \vee v = n\}| = \sum_{i=1}^{N} A_{ni} \tag{1.8}$$

The *self-degree* of a node n is the number of self-loops incident to it:

$$k_n^{\text{self}} = |\{(u, u) \in E^{\text{self}} \mid u = n\}| = A_{nn} \tag{1.9}$$

For directed networks, degree may be further divided into *in-degree*, k_n^{in}, the number of incoming edges to node n:

$$k_n^{\text{in}} = |\{(u, v) \in E \mid v = n\}| = \sum_{i=1}^{N} A_{in} \tag{1.10}$$

and *out-degree*, k_n^{out}, the number of outgoing edges from node n:

$$k_n^{\text{out}} = |\{(u,v) \in E \mid u = n\}| = \sum_{i=1}^{N} A_{ni} \tag{1.11}$$

Naturally,

$$k_n = k_n^{\text{self}} + k_n^{\text{in}} + k_n^{\text{out}} \tag{1.12}$$

For weighted networks, the sum of the weights of the edges incident to a node is called node *strength*:

$$s_n = \sum_{i=1}^{N} W_{ni} \tag{1.13}$$

1.2.2 Betweenness Centrality

Betweenness centrality bc_n of node n is a measure of node centrality based on shortest paths. It is defined as the fraction of all shortest paths in the network that pass through n:

$$bc_n = \sum_{u \neq n \neq v} \frac{\sigma_{uv}(n)}{\sigma_{uv}} \tag{1.14}$$

where σ_{uv} is the total number of shortest paths from node u to v, and $\sigma_{uv}(n)$ is the number of those paths that pass through node n.

1.2.3 Clustering Coefficient

The clustering co-efficient is a measure of the degree to which nodes in a network tend to cluster together. Two measures exist. The local measure quantifies how close a node's neighbours are to a clique – a fully connected subnetwork (Watts and Strogatz, 1998). The global clustering co-efficient is called *transitivity* (see section 1.3.2).

1.3 Macroscopic Network Features

Macro scale features are features of the complex network as a whole. Many micro-scale features, such as node degree k and betweenness centrality bc, can be used to

characters networks at the macro level by averaging across all nodes in the network. In this thesis, we denote mean micro measures using $\langle \cdot \rangle$ – for example $\langle k \rangle$ is the mean node degree across all nodes in the network.

1.3.1 Network Density

Network *density*, ρ, is the ratio of the number of edges, m, in a network to the total possible number of edges. That is:

$$\rho = \frac{2m}{N(N-1)} \tag{1.15}$$

for undirected networks, and:

$$\rho = \frac{m}{N(N-1)} \tag{1.16}$$

for directed networks, where $N = |V|$ is the number of the nodes in the network.

Networks with very low density – such as most real world networks – are called *sparse*.

1.3.2 Transitivity

The global clustering co-efficient T (sometimes called transitivity) is based on the ratio of closed triplets to all triplets in the network and is computed as:

$$T = \frac{\text{number of closed triplets}}{\text{number of all triplets}} \tag{1.17}$$

A 'closed triplet' refers to a fully connected three node subnetwork (sometimes called a triangle) (Wasserman, Faust, et al., 1994).

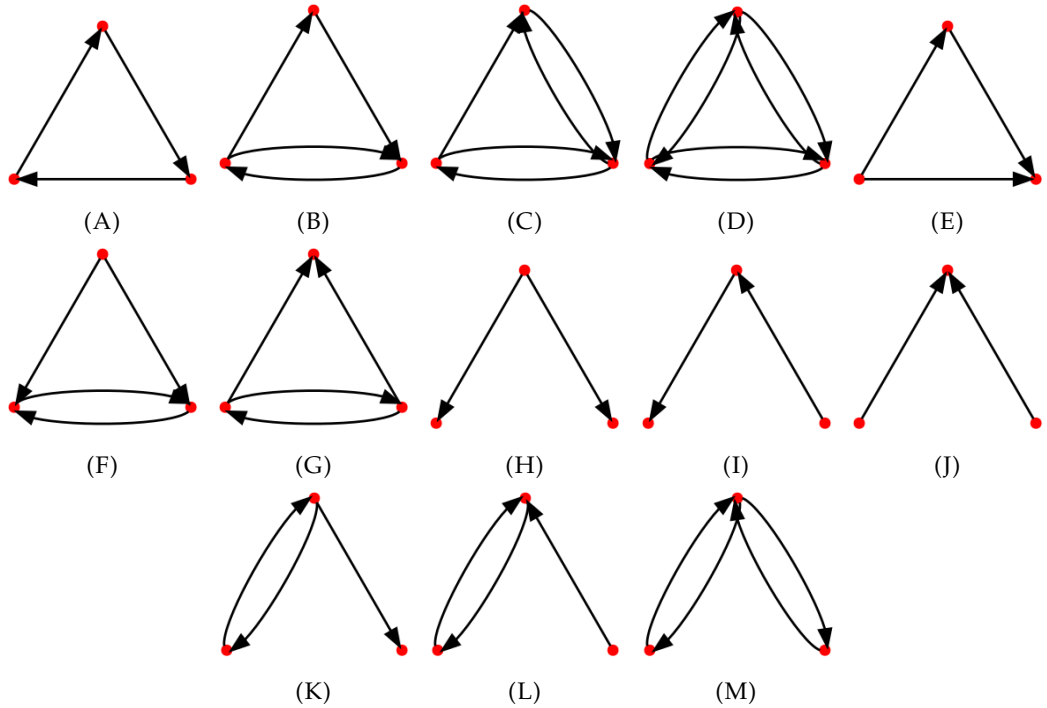


FIGURE 1.2: An enumeration of all possible non-isomorphic 3-node directed motifs.

1.3.3 Network Diameter

The diameter D of a connected network is a global, or *macro*-scale measure and is defined as the greatest shortest path distance between any pair of nodes in the network:

$$D = \max_{u,v \in V} |\mathrm{SP}_{uv}| \tag{1.18}$$

1.4 Meso-scopic Complex Network Features

Between the micro- and macro-scales, we have the meso-scale – characterised by higher-order *architectural* features comprised of sets of entities in the network. Meso- (or intermediate-) level analyses are occasionally specifically designed to reveal connections between micro- and macro-levels (Rombach et al., 2014). Understanding higher order network organisation is a necessary starting point for higher-resolution modelling of complex biologic processes (Csete and Doyle, 2004).

1.4.1 Network Motifs

One of the most enriched meso-scopic features of real-world complex networks are network *motifs*. Network motifs are patterns of interconnections occurring in complex networks at numbers that are significantly higher than those in randomised Erdös-Renyi networks (Milo et al., 2002). Each motif may reflect a framework in which particular functions are achieved efficiently. They have been studied to uncover structural design principles of complex networks (Milo et al., 2002; Alon, 2007). Considering only undirected networks, motifs can be divided into a-cyclic (linear or tree-like) and cyclic, i.e., those with loops. For directed networks, the loops can be further divided into feed-forward and feedback loops (Milo et al., 2002). Figure 1.2 provides an enumeration of all possible three node motifs in a directed network.

1.4.2 Hierarchical Organisation

Hierarchical organisation refers to nodes in a complex network dividing into groups that "further subdivide into groups of groups, and so forth over multiple scales" (Clauset, Moore, and Newman, 2008).

1.4.3 Modular Structure

Complex networks often display a *modular* structure: subsets of nodes that contain a higher degree of inter-connectivity than the rest of the network (Girvan and Newman, 2002; Palla et al., 2005; Lancichinetti, Fortunato, and Kertész, 2009). Finding the modular structure of a network based on its topology is commonly referred to as the problem of 'community detection' (Girvan and Newman, 2002). While a rigorous definition for a 'community' within a network still seems to elude the scientific community (Lancichinetti, Fortunato, and Kertész, 2009), the most popular definition is the planted l-partition model. This was popularised thanks to Girvan and Newman in their seminal work (Girvan and Newman, 2002), and states that as long the probability of a node being connected to its group is greater than the probability of it being connected to the rest of the graph, then the resulting groups are the communities in the network.

This is characterised by the *modularity* measure $Q(\mathbb{S})$ of a subnetwork $\mathbb{S}$, given by:

$$Q(\mathbb{S}) = \frac{1}{2m} \sum_{u,v \in \mathbb{S}} \left[A_{uv} - \frac{k_u k_v}{2m} \right] \tag{1.19}$$

Modules often have an interpret-able meaning – for example, they correspond to friendship groups in a social network; functional modules in Protein-Protein Interaction (PPI) networks; and scientific disciplines in co-authorship networks. Biological networks, including animal brains, exhibit a high degree of modularity.

A convincing case has been made that 'community detection' is not a well-posed problem, but an 'umbrella term' with many facets that emerge from what it is about the network that we are interested in (Schaub et al., 2017). For example, searching for closely knit nodes requires a discrete clustering algorithm, whereas searching for structurally similar nodes leads to an approach like the stochastic block model (SBM) (Decelle et al., 2011).

Appendix A.1 in the appendices provides a review of literature related to identification of modular structure.

1.4.4 Core-Periphery Structure

A core-periphery structure is defined as a modular structure comprised of two parts: the core and the periphery. The core is a central to the structure and is a densely connected subnetwork, while the periphery is a set of sparsely connected, and usually non-central set of nodes, which are linked to the core. We note that all nodes in the core are central but not all central nodes form a network core. While the definition of a core and a community is similar, a core will connect densely with nodes in the periphery, whereas a community does not (Kojaku and Masuda, 2017).

There may be one core-periphery structure in a network, or multiple. Furthermore, some nodes in the notwork may not belong to either a core or a periphery and can be considered noise or *residual* nodes (Kojaku and Masuda, 2017). In the case of multiple cores, two situations are possible. We first define a "global core" to be the core of the entire structure at study. In the first situation, a local core will be the global core, if it is the only local core that occupies a central position in the network,

and all other local cores are peripheral. Such a situation occurs if the network has a hierarchical modular structure with one single module, i.e., the global core, being at the top of the hierarchy. The second situation is that the "global core" does not exist, i.e., no single local core occupies a central position in the network.

1.4.5 Bow-tie Architecture

A bow-tie or hourglass structure is a core-periphery structure found in directed networks, which consists of three layers: input, output and middle layers (Supper et al., 2009). The input and output layers (often referred to as in- and out-components) are peripheral layers are nodes connected with incoming and outgoing edges respectively. The mid-layer is the core, which is also called the "waist", or "knot", or the "selector". The core is the strongly connected component in the network – meaning that every node in the core can be reached by any other node by respecting edge direction (see section 1.1.7) (Timár et al., 2017). Furthermore, the core contains both feed-forward and feedback loops. Feedback loops are of particular relevance in signalling networks, for example, where the signal obtained from receptors in the input layer can be de-noised, filtered and integrated (Brandman et al., 2005). Speaking in terms of motifs, we would expect that the number of feedback loop motifs to be enriched in the core, and for the in- and out-components to be composed of many acyclic motifs and feed-forward loops. Multiple bow-tie architectures may be found within any biological network (Friedlander et al., 2015). Figure 1.3 provides an example of a small network displaying the bow-tie architecture.

Although bow-tie architectures are conceived as a special case of the core-periphery structures in directed networks, they are distinct network architectures. By definition, in a core-periphery structure, the core nodes should be both central and densely connected. However, for a bow-tie architecture, the only requirement for core nodes is that they are strongly connected, which means that they are not central and they might not be densely connected. Therefore, we should regard the bow-ties as a distinct network architecture rather than a special case of the core-periphery structure.

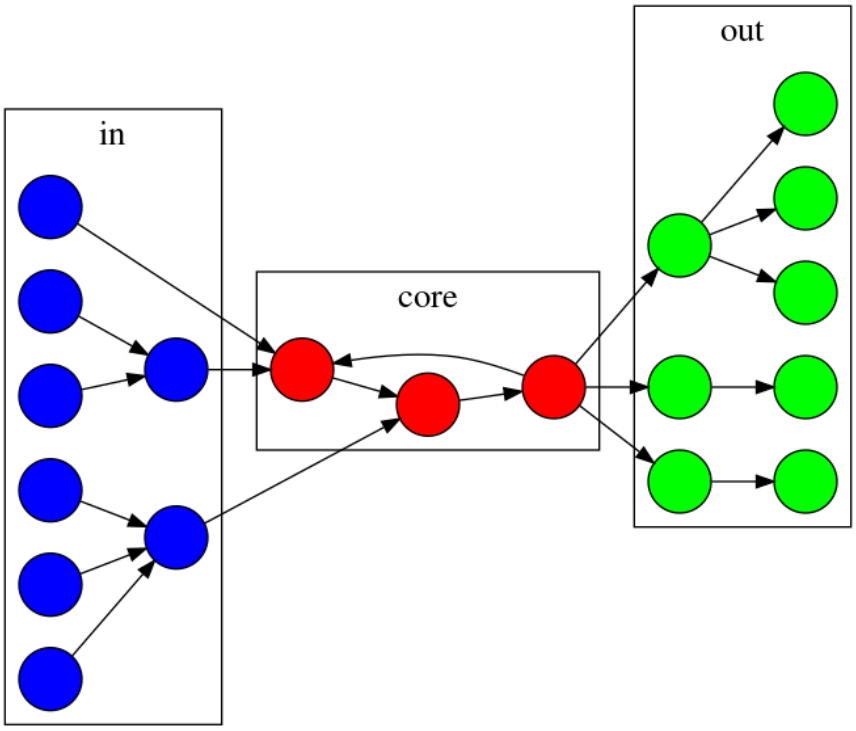


FIGURE 1.3: An example bow-tie network. The three components are identifiable by colour. *blue*: in-component, *red*: core, and *green*: out-component. The core is the only component that is strongly connected: each node in the core is reachable from every other node in the core. Nodes in the in-component connect sparsely to other nodes in the in-component and nodes in the core. Likewise, nodes in the out-component are sparsely connected to core nodes and other nodes in the out-component.

1.4.6 Why Study Meso-Scale Network Architectural Features?

As previously mentioned, the study of meso-scopic features is of particular importance for complex networks as they provide an understanding of the linking between the micro and the macro, as well as explain emergent dynamical behaviour that the system exhibits.

For example: consider modular structure in social networks. Modular structure in social networks could correspond to friendship circles. Friendship circles explain the micro – edges are more likely to form between people in the sample social circle; they explain the macro – the number of social circles would have an effect on the transitivity of the network; and information will flow through more easily between members of the same social circle than between members of different social circles.

1.5 Dynamics on Complex Networks

As mentioned in section 1.4, many complex networks are enriched in the presence of meso-scopic architectural features. However, network architectures include not only topological structure but also interaction functions. By including interaction functions, network architectures can represent both the functional and logical organisation of entities. Network *dynamics* is the study of networks whose status changes over time (Nagler, Levina, and Timme, 2011; Majdandzic et al., 2014). Status may refer to a network's connections, a network's internal state, or both. A network that is not dynamic, we henceforth refer to as *static*.

1.5.1 Dynamic Network Topology

The changing of a network's connections over time refers to the addition and deletion of nodes and edges in a network over time. This is typically modelled discretely and will give rise to time-series network snapshots described by a series of T graphs indexed by t: $\{\mathbb{G}(t) = (V(t), E(t)) \mid t \in [1, T]\}$. The theory that the underlying dynamical processes guiding network formation can be inferred by using network snapshots to build a dynamical model drives the study of topologically dynamic networks. Commonly, inductive models aim to predict future links emerging in a system with more accuracy than a single static network snapshot can provide (Hamilton, Ying, and Leskovec, 2017; Bojchevski and Günnemann, 2018).

1.5.2 Dynamic Network State

This thesis is concerned primarily with the changing of a network's internal state – characterised by the changing of node values, with changes being driven by update rules based on a node's connections. Modelling of network dynamics can be broadly broken down into two categories: boolean networks (Kim, Park, and Cho, 2013) and ordinary differential equation (ODE) dynamic networks (Cheng et al., 2012; Wu et al., 2014; Zhang et al., 2018). We henceforth focus on boolean networks. As a primer for boolean networks, the next section introduces boolean logic.

TABLE 1.1: A truth table of three atomic boolean functions: logical negation: $\neg$, AND $\wedge$, and OR $\vee$. Logical negation in an operation of one boolean variable, whereas AND and OR are functions of two variables. Rows in the table correspond to different combinations of values for the boolean variables p and q. Many further atomic operations exist, but are omitted for brevity. The reader is pointed towards Whitesitt, 2012 for more details.

Boolean Variables		Logical Negation		Boolean Connectives	
p	q	$\neg p$	$\neg q$	$p \wedge q$	$p \vee q$
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

1.5.3 Boolean Logic

A boolean *variable* is a variable that can have only two possible values: true (1), or false (0). A boolean *literal* is either a boolean variable p or its *negation* $\neg p$. The negation of a boolean variable p is given by $\neg p := 1 - p$. Logical negation is often called the NOT operator.

A boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is a function of n boolean variables that evaluates to either 0 or 1. The negation operation can be considered an atomic boolean function from $\{0,1\} \rightarrow \{0,1\}$. Further atomic boolean functions include the AND (or conjunctive) operation: $\wedge$, which accepts an arbitrary number of boolean literal inputs and returns true if all the literals are true, and the OR (or disjunctive) operation: $\vee$, which accepts an arbitrary number of literal inputs and returns true if at least one literal is true. Table 1.1 provides a truth table of these three atomic boolean operations.

1.5.4 Boolean Clauses

A boolean *clause* is a finite collection of boolean literals. The most common clause is the *disjunctive clause*, in which all literals are connected by the OR logical operator. A disjunctive clause is true (evaluates to 1) if at least one of its literals is true.

On the other hand, a conjunctive clause is a set of all literals connected by the AND logical operator, and evaluates to true when all of its component literals evaluate to true.

$$f = (x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_5 \vee \neg x_6) \wedge x_7 \wedge (x_8 \vee x_9 \vee x_{10}) \wedge \neg x_{11}$$

FIGURE 1.4: An example of a boolean function f that is in CNF. f is a function of 11 boolean variables: $x_1, ..., x_{11}$.

1.5.5 Conjunctive Normal Form

A boolean function is in conjunctive normal form (CNF) if it is the conjunction of one or more disjunctive clauses. That is: it is 'an AND of ORs'. The only propositional connectives a formula in CNF can contain AND, OR, and NOT. Furthermore, the NOT operator can only be used as part of a literal. A general boolean function can be arbitrarily complicated, however all boolean functions can be transformed into CNF, and so only the AND, OR and NOT operators are, in general, necessary. Figure 1.4 provides an example of a boolean function in CNF.

1.5.6 Boolean Networks

A boolean network consists of a discrete set of N boolean variables each of which has a boolean function (possibly different for each variable) assigned to it. Each function takes input from a subset of those variables and its output is what determines the state of the variable it is assigned to (Kauffman, 1969). This set of functions, in effect, determines a topology on the set of variables – which are then considered nodes in a network.

We define an update to a network state to be the transition of one network state to another according to the rules of the system. We differentiate between the "synchronous" update scheme (where all nodes in the network update their values at the same time), and the "asynchronous" update scheme, where a node is selected at random (according to some prior distribution) to have its value updated. The seemingly simple synchronous model was only fully understood in the mid 2000s (Drossel, 2008). For the remainder of this thesis we will be only considering the synchronous update scheme.

Boolean networks have been used in biology to model regulatory networks. Although Boolean networks are a crude simplification of genetic reality where genes

are not simple binary switches, there are several cases where they correctly capture the correct pattern of expressed and suppressed genes (Albert and Othmer, 2003).

Formal Definition of a Boolean Network

The dynamics of a synchronous boolean network is taken as a discrete time series where the state of the entire network at time $t+1$ is determined by evaluating each variable's function on the state of the network at time t.

Formally, we define $x_i(t) \in \{0,1\}$ to be the boolean value of node i at time t. Furthermore, we define $\bar{x_i}(t)$ to be the logical negation of boolean variable x_i: $\bar{x_i}(t) = \neg x_i(t)$. Concretely, the value for node i at time $t+1$ is determined by the boolean function f_i, a function over the input nodes of i as:

$$x_i(t+1) = f_i(x_1(t), ..., x_N(t)) \tag{1.20}$$

Note that, the general form for f_i can be written as a function for all nodes in the network $j \in [1, N]$, where only the input nodes contribute to the output of the function.

The state $\mathbf{x} \in \{0,1\}^N$ of a boolean network is defined as the set of values in $\{0,1\}$ for every node in the boolean network. It follows that, for a network of size N, there are 2^N possible states for that network to be in.

1.5.7 Network Attractors

We define the state transition graph (STG) or state "landscape" of a boolean network to be the graph formed by all the possible states that a network can be in, with connections between states if an update will take the network from the first state to the second.

We define an "attractor" to be a set of states that the network will reach that it will never leave through any normal updates. Attractors may be a set of size 1, in which case they are called "steady-state" or point attractors, or they may be cycles of arbitrary period n. The set of states that lead to an attractor is called the *basin* of the attractor. The time it takes to reach an attractor is called *transient time* (Drossel, 2008). In the case of regulatory and signalling networks, attractors typically correspond directly to phenotypes (Huang et al., 2005; Kim, Park, and Cho, 2013).

1.5.8 Hamming Distance

For two boolean network states $\mathbf{x}, \mathbf{y} \in \{0,1\}^N$, the distance between them is given by *Hamming distance*:

$$D_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} |x_i - y_i| \tag{1.21}$$

Hamming distance can be interpreted as the number of modifications required to transform boolean state $\mathbf{x}$ into $\mathbf{y}$ (or vice-versa). Sometimes mean hamming distance is reported instead:

$$mD_H(\mathbf{x}, \mathbf{y}) = \frac{1}{N} D_H(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} |x_i - y_i| \tag{1.22}$$

which is the expected hamming distance between states $\mathbf{x}$ and $\mathbf{y}$ for a particular boolean variable (node) in the network.

1.5.9 Criticality

Dynamic systems that exhibit complex behaviour are often said to be poised right between order and disorder (Roli et al., 2018). This property is called *criticality* and its presence is a characteristic feature of real world dynamical complex networks. In fact, criticality is hypothesised to drive both the robustness and evolve-ability of living processes (Daniels et al., 2018).

Informally, criticality can be thought of as the general robustness that biological systems exhibit – up until a breaking point where the system enters a disorder state that it cannot return from. It has been shown that the average criticality in biological networks is not predictable solely from macro-scale properties such as mean degree $\langle k \rangle$ and mean bias[1] in the logic functions (Daniels et al., 2018).

Average Sensitivity

Average sensitivity $\langle s \rangle$ is a popular measure of criticality, largely due to its scale-ability – making it suitable for even very large networks (Daniels et al., 2018). It is

[1]The mean activity bias p of a boolean function f is the probability that that function takes value 1 (Shmulevich and Kauffman, 2004).

defined to be an indicator of the critical transition in boolean network from an ordered to a chaotic phase (Shmulevich and Kauffman, 2004). It is a measure of the expected change to the network at time $t+1$ after flipping a single bit at time t. Accordingly, this puts the critical value at $\langle s \rangle = 1$. A value of $\langle s \rangle$ much less than one would indicate that the network is operating in the ordered regime, and, similarly, a value much greater indicates disorder. As such, we expect boolean models of biological models to have a value of $\langle s \rangle$ very close to 1, indicating that the network is operating right on the limit of order and disorder. This was shown to be the case in Daniels et al., 2018.

Defining the discrete boolean network dynamics as $\mathbf{x}(t+1) = f(\mathbf{x}(t))$, then $\frac{\partial f_i(\mathbf{x})}{\partial x_j}$ measures the sensitivity of node i to bit-flipping the value of node j for state $\mathbf{x}$:

$$\frac{\partial f_i(\mathbf{x})}{\partial x_j} = f_i(x_1, ..., x_j, ..., x_N) \oplus f_i(x_1, ..., \bar{x_j}, ..., x_N) \tag{1.23}$$

where $\oplus$ is the exclusive OR operation[2]. Simply put, $\frac{\partial f_i(\mathbf{x})}{\partial x_j}$ is the number of nodes that have different values at time $t+1$ cause by flipping the bit of boolean variable x_j at time t.

Using this definition, the average sensitivity of a boolean network of size N is:

$$\langle s \rangle = \frac{1}{N} \sum_{i=1}^{N} \left\langle \sum_{j=1}^{N} \frac{\partial f_i(\mathbf{x})}{\partial x_j} \right\rangle_{\mathbf{x}} \tag{1.24}$$

where $\langle \cdot \rangle_{\mathbf{x}}$ here means the expected value with respect to state $\mathbf{x}$.

Other Measures of Criticality

Another measures of criticality include the *Derrida curve*[3] (Balleza et al., 2008) and *perturbation avalanches* (Serra et al., 2007).

[2]Recall $\bar{x}$ is the logical negation of boolean variable x.

[3]The Derrida curve consists of plotting the mean hamming distance at time $t+1$, $md_H(t+1)$, versus the mean hamming distance at time t, $md_H(t)$, over sampled pairs of states. If the network is operating in the chaotic regime, then small Hamming distances tend to diverge and the Derrida curve lies above the main diagonal for small initial Hamming distances (Balleza et al., 2008).

1.5.10 Control Kernel

The *control kernel* of a given boolean network model of a bio-molecular regulatory network is defined as the minimal set of network nodes needing to be regulated to drive the network state to converge to any desired attractor regardless of the system's initial state (Kim, Park, and Cho, 2013). Here, 'regulated' refers to pinning the values of those nodes to their values in the desired attractor.

Previously, we saw that boolean networks can potentially converge to a number of different attractors, corresponding to a number of different phenotypes for a cell. However, by pinning the values of a set of nodes in the network, the state landscape can be sufficiently changed such that all states converge to a desired attractor. The choice of value to pin the nodes in this set is governed by their value in the desired attractor (Kim, Park, and Cho, 2013).

It has been shown that the size of the control kernel was related to both the topological and logical characteristics of a network (Kim, Park, and Cho, 2013). In addition, the control kernel of the human signalling network included many drug targets and chemical-binding interactions, suggesting a therapeutic application of the control kernel (Kim, Park, and Cho, 2013). This second property makes the identification of a control kernel an attractive area of research.

We henceforth refer to a node belonging to at least one control kernel to be a *control node*.

1.6 Network Embedding

Traditionally, networks are represented as graphs $\mathbb{G} = (V, E)$, where V is the set of nodes in the network and E is the set of edges describing the relationships between nodes. However, when dealing with large networks – containing billions of nodes – traditional approaches suffer some drawbacks (Cui et al., 2018).

First and foremost of these drawbacks is computational complexity. Most network processing algorithms must be iterative or combinatorial in nature. Shortest path length, for example, can be used as an abstraction of node similarity, and is determined by a combinatorial process in $O(N^3)$. Further, we may employ a stochastic

node-traversal approach to estimate node similarity and this process involves iteratively sampling neighbours in the network until convergence.

Next, is applicability to downstream machine learning tasks. Machine learning has recently exploded in popularity, largely due to the successes of deep learning on a variety of disparate tasks. However, most machine learning models assume that data is represented as independent vectors. This means that most methods cannot be applied directly to network data described in the traditional way. One naive approach is to model entities using their row on the adjacency matrix. This explicitly preserves the local information described by E, but is typically prohibitively high-dimensional.

Finally, traditional representations are difficult to make parallel. Distributed computing is the de-facto gold standard for computing in the age of dig data, however, traditional network representation methods are not well suited for this. This is because nodes linked in E would have to be split up across servers, creating a performance bottleneck (Cui et al., 2018). While limited progress has been made to address these issues (for example: see "NetworKit: A tool suite for large-scale network analysis"), the performance depends largely on the topological properties of the underlying graph.

Due to these aforementioned drawbacks, *network embedding* has achieved enormous popularity and success. Network embedding can be seen as a form of unsupervised representation learning and involves learning low-dimensional representations of the entities in a network such that relationships between the entities is preserved. Typically, the entities that are learned are the nodes in the graph (Perozzi, Al-Rfou, and Skiena, 2014), however, it may be the graph in its entirety (Defferrard, Bresson, and Vandergheynst, 2016). Further, the embedding process may consider more than the first order similarity described by E.

Network embedding overcomes the computational complexity limitation of traditional representation techniques by mapping many iterative and combinatorial operations to constant operations in the embedding space. In the example mentioned above, node similarity between any pair of nodes can be determined directly using the metric function of the space, no sampling required.

By learning dense representations of entities in the network, network embedding

is able to de-noise the network while preserving the intrinsic structural information (Cui et al., 2018), further improving performance over the raw network representation. In general, network embedding is capable of supporting a diverse array of downstream tasks, such as node classification, (Grover and Leskovec, 2016; Kipf and Welling, 2016), graph reconstruction (Nickel and Kiela, 2017), node clustering (Wang et al., 2017c), network visualisation (Herman, Melançon, and Marshall, 2000) and link prediction (Wang, Cui, and Zhu, 2016).

The diversity of downstream goals is what separates network embedding from the related field of *graph embedding* (Fu and Ma, 2012). The latter focuses primarily on the graph reconstruction task: that is learning low dimensional representations of a graph from which the original information in the graph can be recovered. In addition to reconstruction, network embedding is typically concerned with network inference tasks, in which we are free to incorporate any additional information that we feel is relevant the inference task that we are interested in.

1.6.1 Structure Preserving Network Embedding

Network structure is a crucial factor in network inference. Many inference tasks could be performed in the graph domain, but, as previously mentioned, the graph domain suffers from a number of problems – chief of which is the problem of high dimensionality. Because of this, much work has been done in preserving rich structural information. High-order measures of similarities, for example shared neighbours (Tang et al., 2015), community membership (Wang et al., 2017c), or random walk proximity (Perozzi, Al-Rfou, and Skiena, 2014) can be included to further improve the quality of the resulting embedding with respect to the performance measures of choice.

1.6.2 Network Embedding with Attributes

In addition to structural information, networks may be richly annotated with side information or *attributes*. Normally, the attributes describe information from a different source to the source of the interactions in the network and so incorporating

them in the network embedding process is often helpful for a wide number of downstream tasks (Li et al., 2017c). However, the actual act of attribute incorporation can be challenging and often great care must be taken to balance the weight of topological and attribute information in the embedding process.

1.6.3 Directed Network Embedding

As discussed in section 1.1.1, many networks are directed, meaning that the relationships between nodes are asymmetric. This proves to be a challenge for network embedding as many approaches learn a single node representation in a symmetric metric space. However, it is possible to overcome these challenges in a number of ways. One solution is to learn multiple representations of a single node (Sun et al., 2019). This allows for an symmetric measure of distance in the following way: Suppose two nodes in the network u and v have two different learned representations each: $\mathbf{x}_u \neq \mathbf{y}_u \in \mathbb{R}^n$ and $\mathbf{x}_v \neq \mathbf{y}_v \in \mathbb{R}^n$. If $D(u, v) := ||\mathbf{x}_u - \mathbf{y}_v||$ and $D(v, u) := ||\mathbf{x}_v - \mathbf{y}_u||$, then, in general, $D(u, v) \neq D(v, u)$.

Another solution is to remove the relax the strict metric requirements and use an asymmetric measure of similarity. For example: use Kullback-Leibler divergence (Bojchevski and Günnemann, 2018).

1.6.4 Supervised Network Embedding

So far, we have only discussed network embedding in the unsupervised setting. That is, we do not consider any labels in the embedding process, only structural and side information. However, if we have a specific goal in mind and have the target information, we can leverage network embedding as representation in and end-to-end process (Li et al., 2017a). We can use the network embedding as a latent hidden layer and augment the supervised objective with an additional objective on the hidden layer to preserve the structural information in the network. This has been shown to perform well even with partial information, that is in the semi-supervised setting when some label information is missing (Kipf and Welling, 2016).

1.6.5 Hyperbolic Network Embedding

A recent trend in network embedding research is to suppose that the hidden metric space underpinning many complex networks is, in fact, hyperbolic (Papadopoulos et al., 2010). A hyperbolic metric space has been shown to explain the scale-free degree distribution observed in real-world networks (Krioukov et al., 2009). Moreover, it can explain the *small-world* effect observed in complex networks (Papadopoulos, Psomas, and Krioukov, 2015), help with routing of information packets around the network (Papadopoulos et al., 2010) and explains the implicit trade-off between popularity and similarity that controls a node's connections (Papadopoulos et al., 2012). In fact, it has been shown that hyperbolic network geometry emerges spontaneously from models of growing simplicial complexes that are purely combinatorial (Bianconi and Rahmede, 2017; Bianconi and Ziff, 2018), and can explain link percolation transition (Kryven, Ziff, and Bianconi, 2019). Further details on hyperbolic spaces are provided in section 1.7.2.

1.6.6 General Form of Network Embedding

Network embedding considers a network $\mathbb{G} = (V, E)$ of N nodes with $|V| = N$, and E is the set of edges. For the general case, the matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ is used to encode the weights of these edges, where $\mathbf{W}_{uv}$ is the weight of the interaction between node u and node v, where $\mathbf{W}_{uv} \neq 0 \iff (u, v) \in E$. If the network is unweighted, then $\mathbf{W}_{uv} = 1$ for all $(u, v) \in E$. Furthermore, the matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$ describes the attributes of each node in the network. These attributes may be discrete or continuous. If the network does not have attributes, then we can set $d = N$ and $\mathbf{X} = \mathbf{I}_N$ to the N-dimensional identity matrix. This allows us to formulate the problem of network embedding as follows:

Problem 1. *Given a network, described by $\mathbb{G} = (V, \boldsymbol{W}, \boldsymbol{X})$, find low dimensional representations of the nodes in the network such that any desired properties of the network are preserved. The described problem may or may not be supervised.*

Problem 1 is intentionally vague: the form the learned representations as well as the measure of the quality of a learned embedding depends entirely on what is meant by 'desired properties'. For example: if the purpose of the embedding is

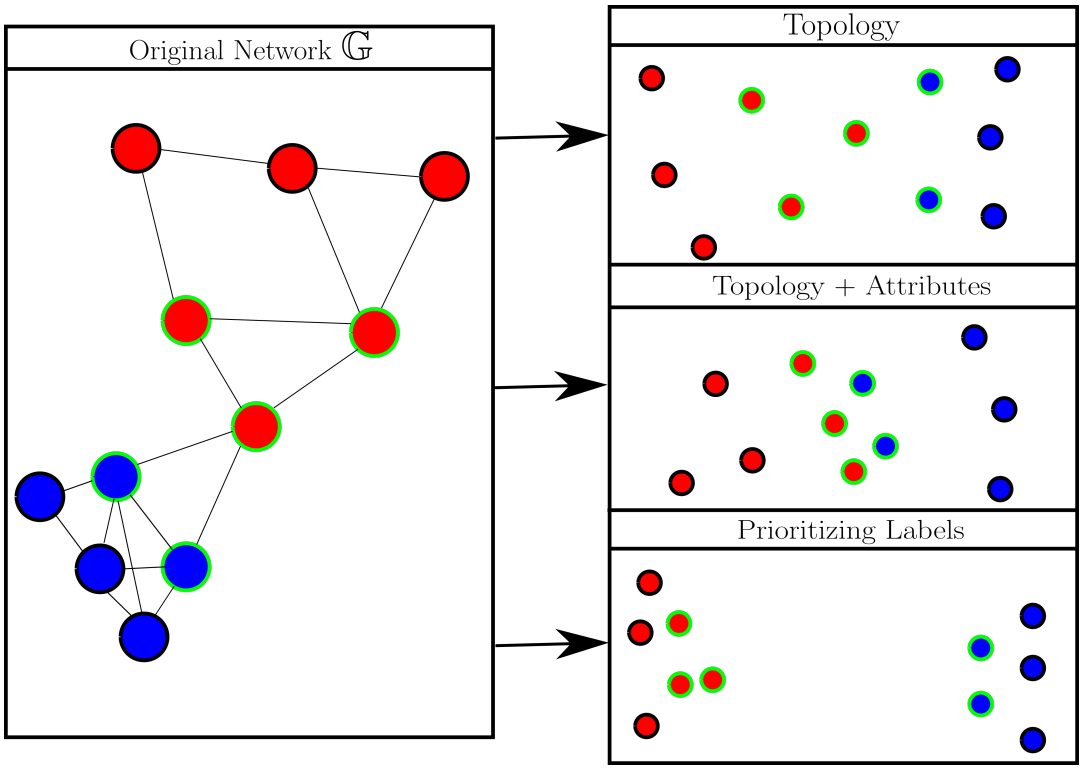


FIGURE 1.5: The effect of embedding objective in the outcome. The left shows an example network $\mathbb{G}$. The nodes in $\mathbb{G}$ belong to one of two classes, marked in blue and red respectively. Furthermore, the nodes in $\mathbb{G}$ are annotated with attributes. For clarity, we have only highlighted one group of nodes possessing similar attributes, which have been drawn with a green circle. On the right, we have plotted three example embeddings resulting from consideration of three different embedding objectives. Top-right shows an embedding that preserves only topological information. Middle-right shows an embedding preserving topological and attribute information. Finally bottom-right shows an embedding that prioritises class separation.

to preserve local structure, then the preservation of neighbourhoods in the learned space is important, however if the goal is clear class separation, with respect to known node labels, then neighbourhood preservation may be de-prioritised in favour of large margins between classes. Figure 1.5 provides an example of this, where three different embedding objectives are considered for the same network, producing three different embeddings.

1.6.7 Relationship to Manifold Learning

Network embedding is closely related to the field of manifold learning. Indeed, many classical non-linear manifold learning techniques, such as Isomap (Tenenbaum, De Silva, and Langford, 2000) and Laplacian Eigenmaps (Belkin and Niyogi,

2002), must first construct nearest neighbour graphs based on dissimilarities between samples before dimensionality reduction takes place. Many of these techniques are directly applicable to embedding of (single-layer, unweighted) complex networks by simply omitting the graph construction step. In practice, however, these methods are not suitable. Isomap, for example, is inapplicable to real-world networks, since it requires the expensive computation of all shortest paths in the network.

1.7 Hyperbolic Geometry

This section introduces the concept of non-Euclidean geometry before going into detail regarding hyperbolic geometry – a geometry that will be used in chapter 3 and chapter 4.

1.7.1 Introduction to Non-Euclidean Geometry

Everyone is familiar with Euclidean geometry. This is the geometry of the world in which we live. It lives in a space that is *connected* (the space cannot be broken up as the union of open subsets), *flat* and *isotropic* (the Gaussian curvature[4] of the space is zero for all points in the space) (Reynolds, 1993).

Euclidean Geometry obeys all of the Euclid's postulates that seem "obviously true" in the world that we live in. These consist of axioms such as the existence of a straight line segment between two points, a circle being uniquely described by a centre and a radius, and the *parallel postulate*: that given a line l and a point P not on l, there exists exactly one line through P that is parallel to (does not intersect) l.

Moving beyond familiar Euclidean geometry, there are three types of connected, isotropic spaces:

- Euclidean, with Gaussian curvature equal to 0,
- Spherical, with strictly positive Gaussian curvature, and

[4]Gaussian curvature $\mathcal{K}$ of a surface S at a point P, we find the normal vector at P, and then define a normal plane as a plane that intersects the surface and contains the normal vector. The intersection of a normal plane and the surface will form a curve called a normal section and the curvature of this curve is the normal curvature. For most points on most surfaces, different normal sections will have different curvatures; the maximum and minimum values of these are called the principal curvatures, call these K_1 and K_2. The Gaussian curvature is the product of the two principal curvatures $\mathcal{K} = K_1 K_2$.

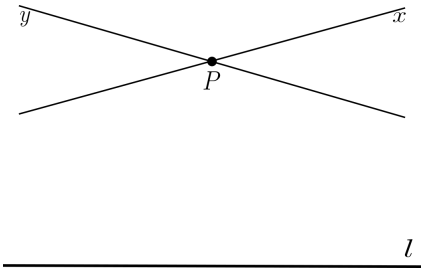


FIGURE 1.6: Parallel postulate for hyperbolic space. Both lines x and y pass through point P and are parallel to line l.

- Hyperbolic, with strictly negative Gaussian curvature.

We shall not focus on Spherical geometry here (however, it will become a useful comparison later). Instead the hyperbolic geometry shall be the focus of this two chapters of this thesis.

1.7.2 Hyperbolic Geometry

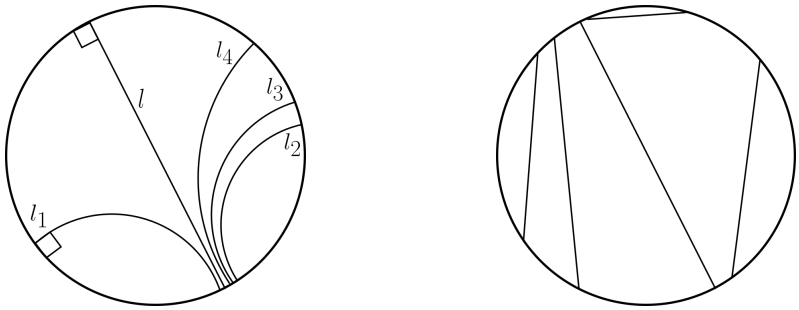
Hyperbolic geometry is what one obtains by negating Euclid's fifth postulate, the *parallel postulate*. This means that, in hyperbolic geometry, there exists a line l and a point P not on l such that at least two distinct lines parallel to l pass through P (Krioukov et al., 2010) (see figure 1.6).

Hyperbolic spaces and trees are very similar (Krioukov et al., 2010). Informally, trees can be thought of as "discrete hyperbolic spaces" and can be embedded into a two dimensional hyperbolic plane without distortion (Sarkar, 2011; De Sa et al., 2018).

Furthermore, hyperbolic space cannot be embedded into Euclidean space without distortion. Informally, hyperbolic spaces are "larger" and have more "space" than Euclidean spaces (Krioukov et al., 2010). For example: the area A of a circle of radius r, does not grow quadratically with r, as we are used to in Euclidean space[5], but grows exponentially with r as $A = 2\pi \cosh(\zeta r - 1)$[6].

Because of the distortion caused by representing hyperbolic spaces in Euclidean spaces, there are not one but many equivalent models of hyperbolic spaces (Cannon et al., 1997). Each model emphasises different aspects of hyperbolic geometry, and

[5]Recall that the area of a circle is given by $A = \pi r^2$ in Euclidean geometry.
[6]$\zeta = \sqrt{-K}$ and cosh is Hyperbolic cosine function: $\cosh(x) = (e^x + e^{-x})/2$.



(A) Poincaré disk (B) Klein disk

FIGURE 1.7: Lines on the Poincaré and Klein disks. (A) plots five parallel lines on the Poincaré disk. Every one of l_1, l_2, l_4 and l_4 is the arc of a circle, is parallel to l and meet the boundary at right angles. (B) plots lines on the Klein disk. Each straight Euclidean line on the Klein disk maps to a straight line hyperbolic geometry.

can be freely mapped to each other with an *isometry*, but no model simultaneously represents all of its properties.

1.7.3 Poincaré Disk

Traditionally, the most popular model of hyperbolic geometry is the *Poincaré disk* $\mathbb{P}^2$ (called the *Poincaré ball* $\mathbb{P}^n$ for $n > 2$ dimensions). Here, the entire hyperbolic plane is represented as the interior of a Euclidean unit ball, sitting in a Euclidean ambient space, where the centre of the circle represents the origin of the space and the boundary of the ball represents infinity (Krioukov et al., 2010; Nickel and Kiela, 2017). Euclidean and hyperbolic distances (r_e and r_h respectively) from the disk centre are related exponentially by $r_e = \tanh(r_h/2)$. Shortest paths between points on the Poincaré disk (called *geodesics*) are given by the diameters of the disk (if both points lie on a diameter) or Euclidean circle arcs that contain both points and intersect the boundary of the ball perpendicularly. Figure 1.7A shows a number of parallel lines on the Poincaré disk.

For machine learning practitioners, this model has the advantage that it is *conformal*: the Euclidean angles in the model are equal to the hyperbolic angles in the underlying hyperbolic geometry.

1.7.4 Klein Disk

Similar to the Poincaré disk, we have also the much less popular *Klein* disk model of hyperbolic geometry $\mathbb{K}^n$. This model also represents hyperbolic geometry as a

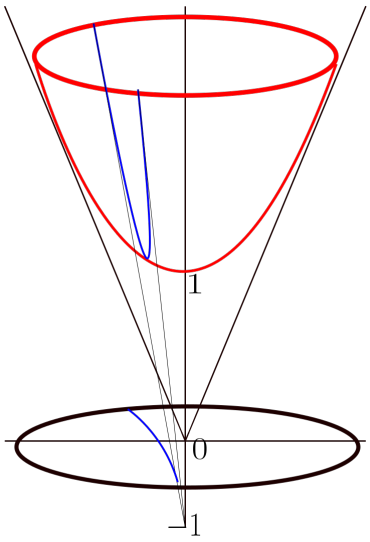


FIGURE 1.8: The relationship between the two-dimensional hyperboloid model $\mathbb{H}^2$ (shown in red) and the Poincarè disk $\mathbb{P}^2$ (drawn in black). The blue lines map isometrically to each other.

unit disk (or ball) in an ambient Euclidean space. This model preserves straight lines: straight Euclidean lines map to straight hyperbolic lines. This has the natural advantage that linear separability on the disk reflects separability in the underlying geometry that it is representing. Figure 1.7B provides an example of a number of straight lines on the Klein disk. However, unlike the *Poincaré ball*, the Klein disk is not conformal, and the Euclidean angles in the model are not equal to hyperbolic angles.

1.7.5 The Hyperboloid Model

Unlike disk models, that sit in an ambient Euclidean space of dimension n, the hyperboloid model of n-dimensional hyperbolic geometry, $\mathbb{H}^n$, is an n-dimensional hyperboloid manifold that sits in $n+1$-dimensional Minkowski space-time. Both the *Poincaré* and *Klein* models can be viewed as one-to-one projections from points on the hyperboloid to points on disks orthogonal to the main axis of the hyperboloid (Krioukov et al., 2010). Informally, we can see these relationships as analogous to the relationship between a projected map and a globe (Reynolds, 1993). Figure 1.8 demonstrates the relationship between the two dimensional hyperboloid and Poincaré disk.

Since the hyperboloid model is relevant to two upcoming chapters, we shall now introduce it in more detail.

Minkowski Space-time

$n+1$-dimensional Minkowski space-time (denoted a $\mathbb{R}^{n:1}$) is defined as the combination of n-dimensional Euclidean space with a time co-ordinate t. A point $\mathbf{u} \in \mathbb{R}^{n:1}$ has spacial coordinates $\mathbf{u}^i$ for $i = 1, 2, ..., n$ and time co-ordinate $\mathbf{u}^{n+1}$.

Minkowski Bilinear Form

The *Minkowski bilinear form* is defined as:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{R}^{n:1}} = \sum_{k=1}^{n} \mathbf{u}^k \mathbf{v}^k - \psi^2 \mathbf{u}^{n+1} \mathbf{v}^{n+1} \tag{1.25}$$

where ψ is the speed of information flow in the system (normally set to 1 for simplified calculations). Further details have been omitted for brevity, however the reader is directed to Clough and Evans, 2017 for more details. The bilinear form acts as an inner product and allows (among other things) the computation of vector norms in the familiar way:

$$||\mathbf{u}||_{\mathbb{R}^{n:1}} := \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}_{\mathbb{R}^{n:1}} \tag{1.26}$$

It is worth noting at this point that $n+1$-dimensional Minkowski space-time contains the entire n-dimensional Euclidean space: $\mathbb{R}^n \subset \mathbb{R}^{n:1}$. $\mathbb{R}^n$ is the set of all $\mathbf{x} \in \mathbb{R}^{n:1}$ such that its time co-ordinate $\mathbf{x}^{n+1} = 0$. Furthermore, the Minkowski bilinear form (equation (1.25)) is a generalisation of the Euclidean inner product and is equivalent for all $\mathbf{x} \in \mathbb{R}^n = \{\mathbf{x} \in \mathbb{R}^{n:1} \mid \mathbf{x}^{n+1} = 0\}$. However, it is possible for $\langle \mathbf{u}, \mathbf{u} \rangle_{\mathbb{R}^{n:1}} < 0$ and so norms may be imaginary.

Hyperboloid Definition

The points $\mathbf{u} \in \mathbb{R}^{n:1}$ satisfying:

$$\mathbb{H}^n = \{\mathbf{u} \in \mathbb{R}^{n:1} \mid \langle \mathbf{u}, \mathbf{u} \rangle_{\mathbb{R}^{n:1}} = -1 \wedge \mathbf{u}^{n+1} > 0\} \tag{1.27}$$

define the hyperboloid model (Wilson and Leimeister, 2018). The first condition ($\langle \mathbf{u}, \mathbf{u} \rangle_{\mathbb{R}^{n:1}} = -1$) defines a hyperbola of two sheets, and the second $\left(\mathbf{u}^{n+1} > 0\right)$ selects the top sheet.

Distance on the Hyperboloid

Geodesics between points on the model are given by the hyperbola formed by the intersection of $\mathbb{H}^n$ and the two dimensional plane containing the origin and both of the points (Reynolds, 1993). The distance along the geodesic between two points $\mathbf{u}, \mathbf{v} \in \mathbb{H}^n$ is given by:

$$D_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v}) = \operatorname{arccosh}(-\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{R}^{n:1}}) \tag{1.28}$$

and can be considered analogous to the length of the great circle connecting two points in spherical geometry (Reynolds, 1993).

Tangent Space of a Point on the Hyperboloid

The tangent space of a point $\mathbf{u} \in \mathbb{H}^n$ is defined as:

$$T_{\mathbf{u}}\mathbb{H}^n = \{\mathbf{x} \in \mathbb{R}^{n:1} \mid \langle \mathbf{u}, \mathbf{x} \rangle_{\mathbb{R}^{n:1}} = 0\} \tag{1.29}$$

$T_{\mathbf{u}}\mathbb{H}^n$ is the collection of all points in $\mathbb{R}^{n:1}$ that are orthogonal to point $\mathbf{u} \in \mathbb{H}^n$, with respect to the Minkowski bilinear form. It can be shown that $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathbb{R}^{n:1}} > 0$ $\forall \mathbf{x} \in T_{\mathbf{u}}\mathbb{H}^n$ $\forall \mathbf{u} \in \mathbb{H}^n$ (Reynolds, 1993). In other words, the tangent space of the hyperboloid is positive definite (with respect to the Minkowski bilinear form) for all points $\mathbf{u} \in \mathbb{H}^n$ on the hyperboloid. This property defines $\mathbb{H}^n$ (equipped with the Minkowski bilinear form) as a Riemannian manifold (Reynolds, 1993). Furthermore, $||\mathbf{x}||_{\mathbb{R}^{n:1}} \geq 0$ for all vectors $\mathbf{x} \in T_{\mathbf{u}}\mathbb{H}^n$, for any point $\mathbf{u} \in \mathbb{H}^n$ allowing for full Riemannian gradient descent (Wilson and Leimeister, 2018; Nickel and Kiela, 2018).

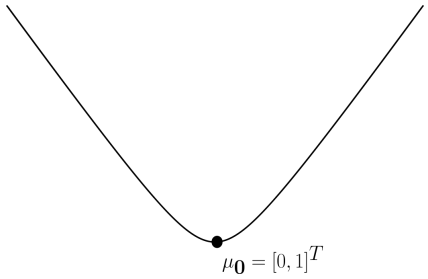


FIGURE 1.9: $\mu_{\mathbf{0}}$ on the one-dimensional hyperboloid $\mathbb{H}^1 = \{(x,y) \mid x^2 - y^2 = -1\}$.

Bottom Tip of the Hyperboloid

We now define the "bottom tip" of the hyperboloid as:

$$\mu_{\mathbf{0}} := [0,...,0,1]^T \in \mathbb{H}^n \tag{1.30}$$

$\mu_{\mathbf{0}}$ is the point on the upper hyperboloid with zeros for all of its spacial co-ordinates and 1 as its time co-ordinate. Figure 1.9 highlights $\mu_{\mathbf{0}}$ on the one-dimensional hyperboloid.

Note the following property of the tangent space $\mu_{\mathbf{0}}$:

$$T_{\mu_{\mathbf{0}}}\mathbb{H}^n = \{\mathbf{x} \in \mathbb{R}^{n:1} \mid \langle \mathbf{x}, \mu_{\mathbf{0}} \rangle_{\mathbb{R}^{n:1}} = 0\} \equiv \mathbb{R}^n \tag{1.31}$$

In other words, $T_{\mu_{\mathbf{0}}}\mathbb{H}^n$ is the entire n-dimensional Euclidean space (every point in $n+1$-dimensional Minkowski space with a 0 time co-ordinate). That is, the set of all $\mathbf{x} \in \mathbb{R}^{n:1}$ such that $\mathbf{x}^{n+1} = 0$. Furthermore, in this space, Euclidean and Minkowski norms coincide[7]: $\forall \mathbf{x} \in T_{\mu_{\mathbf{0}}}\mathbb{H}^n, ||\mathbf{x}|| \equiv ||\mathbf{x}||_{\mathbb{R}^{n:1}}$.

Exponential Map

The *exponential map*, $\text{Exp}_{\mathbf{u}}(\mathbf{x}) : T_{\mathbf{u}}\mathbb{H}^n \to \mathbb{H}^n$, is a Riemannian operation to transport vectors from the tangent space of a point $\mathbf{u} \in \mathbb{H}^n$ on the hyperboloid to the hyperboloid itself. It is defined as:

$$\text{Exp}_{\mathbf{u}}(\mathbf{x}) = \cosh(r) \cdot \mathbf{u} + \sinh(r) \cdot \frac{\mathbf{x}}{r} \tag{1.32}$$

[7]The Euclidean norm $||\mathbf{x}||$ of $\mathbf{x} \in \mathbb{R}^{n:1}$ is the usual norm of Euclidean geometry considering only the spacial co-ordinates of $\mathbf{x}$.

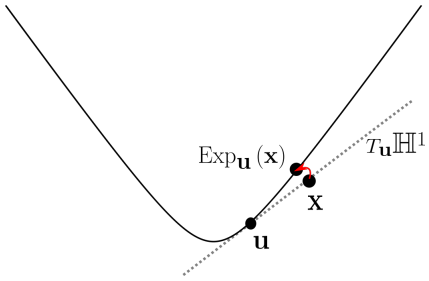


FIGURE 1.10: Example of the exponential map on the hyperboloid. We show a points on the hyperboloid $\mathbf{u} \in \mathbb{H}^n$, as well its corresponding tangent space $T_{\mathbf{u}}\mathbb{H}^n$ given by the dotted line. The red line shows how Exp$\mathbf{u}$ transports vector $\mathbf{X}$ from the the tangent space of vector $\mathbf{u}$ to the hyperboloid. The hyperbolic distance $D_{\mathbb{H}^1}(\mathbf{v}, \text{Exp}_{\mathbf{u}}(\mathbf{x}))$ of vector $\mathbf{u}$ and $\text{Exp}_{\mathbf{u}}(\mathbf{x})$ is equal to the Minkowski norm of $\mathbf{x}$.

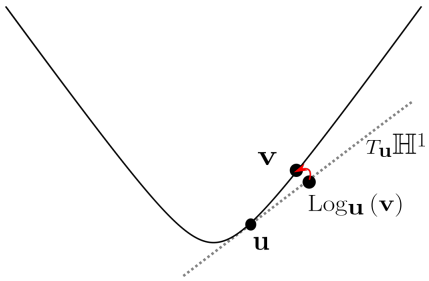


FIGURE 1.11: Example of the logarithmic map on the hyperboloid. We show two points on the hyperboloid $\mathbf{u}, \mathbf{v} \in \mathbb{H}^n$, as well the tangent space $T_{\mathbf{u}}\mathbb{H}^n$ of point $\mathbf{u}$ given by the dotted line. The red line shows how $\text{Log}_{\mathbf{u}}$ transports vector $\mathbf{v}$ from the hyperboloid to the tangent space of vector $\mathbf{u}$.

where $r = ||\mathbf{x}||_{\mathbb{R}^{n:1}} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_{\mathbb{R}^{n:1}}}$ denotes the Minkowski norm of $\mathbf{x}$. This is analogous to the exponential map in spherical geometry which maps points from the tangent space of a point on the sphere back to the sphere itself[8]. Figure 1.10 provides an example of the exponential map transporting a vector from the tangent space of a vector on the hyperboloid to the hyperboloid itself.

Logarithmic Map

The inverse of the exponential map, the *logarithmic map*, $\text{Log}_{\mathbf{u}}(\mathbf{v}) : \mathbb{H}^n \to T_{\mathbf{u}}\mathbb{H}^n$, transports a vector $\mathbf{v} \in \mathbb{H}^n$ from the hyperboloid to the tangent space of an arbitrary point $\mathbf{u} \in \mathbb{H}^n$. We set $\alpha = -\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{R}^{n:1}}$ and compute $\mathbf{x} = \text{Log}_{\mathbf{u}}(\mathbf{v}) : \mathbb{H}^n \to T_{\mathbf{u}}\mathbb{H}^n$ as:

$$\mathbf{x} = \text{Log}_{\mathbf{u}}(\mathbf{v}) = \frac{\text{arccosh}(\alpha)}{\sqrt{\alpha^2 - 1}}(\mathbf{v} - \alpha\mathbf{u}) \tag{1.33}$$

[8]The spherical exponential map is given by $\text{Exp}_{\mathbf{p}}(\mathbf{v}) := \cos(||\mathbf{v}||) \cdot \mathbf{p} + \sin(||\mathbf{v}||) \cdot \mathbf{v}/||\mathbf{v}||$. Compare this to the hyperboloid case. For a concrete example: imagine that $\mathbf{u}$ was a point on the globe. A plane flies (seemingly in a straight line to the passengers of the plane) parallel to the surface of the globe in a straight direction of $\mathbf{x}$ for a distance of $||\mathbf{x}||$. Then $\mathbf{u}' = \text{Exp}_{\mathbf{u}}(\mathbf{x})$ is the point on the globe that the plane will land at.

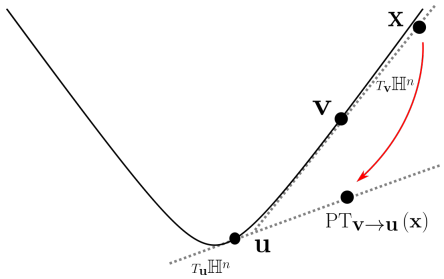


FIGURE 1.12: Example of the parallel transport operation on the hyperboloid. We show two points on the hyperboloid $\mathbf{u}, \mathbf{v} \in \mathbb{H}^n$, as well their corresponding tangent spaces $T_{\mathbf{u}}\mathbb{H}^n$ and $T_{\mathbf{v}}\mathbb{H}^n$ given by the dotted lines. The point $\mathbf{x} \in T_{\mathbf{v}}\mathbb{H}^n$ lies on the tangent space of point $\mathbf{v}$. The red arrow shows how $\mathbf{x}$ is mapped from $T_{\mathbf{v}}\mathbb{H}^n$ to $T_{\mathbf{u}}\mathbb{H}^n$ via $\mathrm{PT}_{\mathbf{v}\to\mathbf{u}}$. Since $\mathrm{PT}_{\mathbf{v}\to\mathbf{u}}$ is norm-preserving, then $||\mathbf{x}||_{\mathbb{R}^{n:1}} \equiv ||\mathrm{PT}_{\mathbf{v}\to\mathbf{u}}(\mathbf{x})||_{\mathbb{R}^{n:1}}$.

For $\mathbf{u}, \mathbf{v} \in \mathbb{H}^n$, the logarithmic map computes the vector $\mathbf{x} = \mathrm{Log}_{\mathbf{u}}(\mathbf{v}) \in \mathbb{R}^{n:1}$ with the following properties. First, $\mathbf{x}$ is orthogonal to $\mathbf{u}$ with respect to the Minkowski bilinear form: $\langle \mathbf{u}, \mathbf{x} \rangle_{\mathbb{R}^{n:1}} = 0$. Next, the Minkowski norm of $\mathbf{x}$ is equal to the hyperbolic distance between $\mathbf{u}$ and $\mathbf{v}$:

$$||\mathrm{Log}_{\mathbf{u}}(\mathbf{v})||_{\mathbb{R}^{n:1}} = D_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v}) \tag{1.34}$$

A consequence of this is that, when $\mathbf{v} = \mathbf{u}$, $\mathrm{Log}_{\mathbf{u}}(\mathbf{v}) = \mathbf{0}$. Finally, the direction of $\mathbf{x}$ from the origin, is the same are the direction of the geodesic from $\mathbf{u}$ to $\mathbf{v}$ (Nagano et al., 2019). Another noteworthy property of $\mathrm{Log}_{\mathbf{u}}$ is that for $\mathbf{v}, \mathbf{w} \in \mathbb{H}^n$ is that for $\mathbf{x} = \mathrm{Log}_{\mathbf{u}}(\mathbf{v})$ and $\mathbf{y} = \mathrm{Log}_{\mathbf{u}}(\mathbf{w})$:

$$||\mathbf{x} - \mathbf{y}||_{\mathbb{R}^{n:1}} \neq D_{\mathbb{H}^n}(\mathbf{v}, \mathbf{w}) \tag{1.35}$$

That is, $\mathrm{Log}_{\mathbf{u}}$ does not preserve distances between arbitrary vectors, only distances to $\mathbf{u}$. Figure 1.11 provides an example of the logarithmic map operation on the one-dimensional hyperboloid.

Parallel Transport

The hyperboloid *parallel transport* operation, $\mathrm{PT}_{\mathbf{u}\to\mathbf{u}'}(\mathbf{x}) : T_{\mathbf{u}}\mathbb{H}^n \to T_{\mathbf{u}'}\mathbb{H}^n$, provides a smooth norm-preserving mapping between tangent spaces on a Riemannian manifold (Nagano et al., 2019). It transports a vector $\mathbf{x} \in T_{\mathbf{u}}\mathbb{H}^n$ to $\mathbf{y} \in T_{\mathbf{u}'}\mathbb{H}^n$. It is

computed as:

$$\mathbf{y} = \mathrm{PT}_{\mathbf{u}\to\mathbf{u}'}(\mathbf{x}) = \mathbf{x} + \frac{\langle \mathbf{u}' - \alpha\mathbf{u}, \mathbf{x}\rangle_{\mathbb{R}^{n:1}}}{\alpha + 1}(\mathbf{u} + \mathbf{u}') \tag{1.36}$$

where $\alpha = -\langle \mathbf{u}, \mathbf{u}'\rangle_{\mathbb{R}^{n:1}}$. We provide an example of the parallel transport operation in figure 1.12.

Note that, by setting $\mathbf{u}' = \mu_{\mathbf{0}} = [0, ..., 0, 1]^T$, we are able to transport any vector $\mathbf{x}$ in the tangent space of an arbitrary point $\mathbf{u} \in \mathbb{H}^n$ to the n-dimensional Euclidean space that is the tangent space of bottom tip of the hyperboloid, $\mu_{\mathbf{0}}$. On other words, we can define the *parallel transport to* $\mathbb{R}^n$ operation as $\mathrm{PT}_{\mathbf{u}\to\mu_{\mathbf{0}}}(\mathbf{x}) : T_{\mathbf{u}}\mathbb{H}^n \to \mathbb{R}^n$, since $T_{\mu_{\mathbf{0}}}\mathbb{H}^n = \mathbb{R}^n$.

1.8 Research Questions

Due to the ubiquity of complex real-world systems and flexibility of complex networks to model those systems, this thesis focuses on feature extraction techniques for complex networks. This thesis provides frameworks for learning representations of both micro- and meso-scopic features, preserving their inherent hierarchy, whilst transforming them into suitable forms for further analyses.

The next subsections introduce the major research questions that this thesis intends to answer in detail.

1.8.1 Hyperbolic Embedding of Weighted and Attributed Undirected Networks

As discussed in section 1.1.6, all (undirected) complex networks can be considered attributed and weighted networks. Furthermore, as mentioned in section 1.6, network embedding can overcome the numerous drawbacks of graph-based inference approaches by learning low-dimensional representations where node attributes and weights can be incorporated in the learning process. Section 1.6.5 shows that low-dimensional hyperbolic node representations can better explain phenomenon that complex networks exhibit – phenomenon such as a high clustering co-efficient, a scale-free degree distribution and the small world principle.

Existing hyperbolic approaches, however, cannot currently handle the embedding of attributed networks. In addition, many attributed network embedding approaches do not allow for the explicit control of the trade-off between topology and attributes in the learned embedding (see section 1.6.2). Based on this, the first research question of this thesis is:

- Can existing hyperbolic network embedding approaches be improved by generalising to attributed and weighted networks, whilst allowing for explicit control of the trade-off between topology and attributes in the embedding procedure?

A flexible framework for low-dimensional hyperbolic representation learning on the nodes in an undirected complex network with attributed nodes and weighted edges is introduced in chapter 3. The details of the approach are described in section 3.2.

1.8.2 Hyperbolic Embedding of Attributed and Directed Networks

All complex networks can be considered directed networks. Following on from the previous question, it is natural to ask if hyperbolic embedding can be applied to attributed and directed networks. As highlighted in section 1.6.3, directed network embedding is a challenging task as metric spaces naturally provide a symmetric similarity measure over the nodes in the learned space.

Inspired by previous works (Bojchevski and Günnemann, 2018), we investigate whether an inductive approach for learning low-dimensional node representations of an attributed and directed network is suitable for hyperbolic space. Here, 'inductive' means mapping directly from node attributes which has the advantage of handling the embedding of unseen nodes – making any solution suitable for time-series (see section 1.5.1) or incomplete complex networks exhibiting properties explained by hyperbolic geometry – such as a scale free degree distribution. This gives rise to the second research question:

- Can existing inductive attributed and directed network embedding approaches be extended to hyperbolic space?

Chapter 4 proposes a solution to this question, that is described in section 4.2 and validated in section 4.3.

1.8.3 Hierarchical Organisation of Network Architectures

To understand the overall behaviour of dynamic networks, understanding of the micro scale is not enough. Meso-scopic network properties – the complex, high-order interplay of nodes – are what drive the network to perform a function (Supper et al., 2009). Furthermore, meso-scale features are arranged hierarchically, just as nodes are arranged hierarchically in the network. Small feedback loops merge to form tight, modular, strongly connected structure, which gives way to the larger architectures within the network, such as the bow-tie architecture. But how do the dynamics of the network flow through this hierarchy? In light of this, the third research question is:

- Does the hidden hierarchy of small, local architectures in signalling networks drive global network behaviour?

The purpose of this question is to understand the relationships between local and global dynamics in intra-cellular networks, as well as to identify targets for control. A general framework for decomposing the components of bow-tie networks into a hierarchy of loops is provided in section 5.2 and the impact of deep core feedback loops on the global behaviour of a network is investigated and discussed in section 5.4.

1.9 Thesis Contributions

The main purpose of the work proposed in this thesis is to provide frameworks that can better make sense of all the structured data that is produced in the age of big data and of high-throughput biological experiments. It will generate knowledge from that data through the means of extracting useful and easy-to-interpret features from complex network models – knowledge that will allow for the automatic generation of predictions or further study by domain experts. This will be achieved through novel network embedding and feature extraction techniques and contributes to the field in the following three aspects:

- We propose a novel approach to embed attributed and weighted networks to hyperbolic space, achieving state-of-the-art results on a number of downstream tasks through the use of random walk network sampling and a novel hyperboloid learning algorithm (chapter 3).
- We develop a novel inductive embedder model to embed attributed directed network to distributions in hyperbolic space, as well as a novel method for measuring asymmetric similarity between those distributions (chapter 4).
- Using a simple iterative merging approach, we show that highly modular coherent feedback loops found within strongly connected components of signalling networks with missing information significantly affect network behaviour and so make promising candidates for future drug targets (chapter 5).
- Based on the algorithms and methods proposed in this thesis, we have developed the following open-source software:
  - HEAT (`https://github.com/DavidMcDonald1993/heat`);
  - HEADNet (`https://github.com/DavidMcDonald1993/HEADNET`);
  - IMPLISig (`https://github.com/DavidMcDonald1993/IMPLISig`).

1.10 List of Publications

This thesis summarises the contributions of the following publications:

- David McDonald and Shan He (2020b). "HEAT: Hyperbolic Embedding of Attributed Networks". In: *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, pp. 28–40. *Winner: best paper award*.
- David McDonald and Shan He (2020a). "HEADNet: Hyperbolic Embedding of Attributed Directed Networks". In: *IEEE Transactions on Neural Networks and Learning Systems. Submitted*.
- David McDonald, Sami Cass Darweish, and Shan He (2020). "Multi-scale Hierarchical Decomposition of Bow-tie Architectures in Intra-cellular Networks". In: *Bioinformatics. Preparing for submission*.

1.11 Thesis Organisation

This thesis consists of six chapters, focusing on uncovering the hidden hierarchy of real-world complex networks. The first chapter introduces the background and the importance of this topic. We state the research questions to be studied and summarise the contributions.

In chapter 2, we review the important literature in related fields. Section 2.1 introduces widely used undirected network embedding techniques. Then section 2.2 focuses on embedding techniques specifically for directed networks. Section 2.3 reviews related works in uncovering higher-order meso-scopic architectural features in complex networks, as well as their relationships to overall network dynamics.

In chapter 3, we study network embedding of attributed networks to hyperbolic space. Section 3.1 describes the motivation and defines the problem. Section 3.2 proposes a novel two-step network embedding approach that incorporates node attributes to generate random walks that provide the basis for a novel learning objective to learn low-dimensional hyperbolic representations of nodes in a network. Our approach is validated on a number of real-world benchmark complex network datasets in section 3.3.

In chapter 4, we expand upon the work proposed in chapter 3 by looking at hyperbolic embedding of attributed directed networks. Section 4.1 introduces the subject and defines the formal problem. We propose a simple objective function and optimisation strategy for the problem in section 4.2. In section 4.3, we validate our approach on a number of downstream machine learning tasks.

In chapter 5, we propose a general framework for uncovering the hierarchical modular structure of loops in components of bow-tie architectures found within regulatory and signalling networks. Section 5.1 provides the motivation for this work. Section 5.2 describes the general framework. Finally, section 5.4 validates the proposed method.

Chapter 6, the final chapter, concludes the thesis and proposes several promising future directions for the work presented here, based on literature.

Chapter 2

Literature Review

This chapter reviews all literature relevant to the concepts identified in chapter 1, highlighting existing research on complex network representation learning and higher-order network architectural features. Section 2.1 reviews literature related to undirected network embedding. Section 2.2 examines directed network embedding techniques. Section 2.3 looks at complex network architectural features, as well as their relationship to global dynamics and network control. Finally, section 2.4 summarises the chapter.

2.1 Undirected Network Embedding

Problem 1 identified in section 1.6 describes the problem of learning low dimensional representations of nodes in a (possibly weighted and attributed) network. To address this problem, approaches have been proposed from many disciplines and angles.

2.1.1 Matrix Factorisation Based Approaches

Matrix factorisation methods, with the goal of learning low-rank representation for the original matrix, can naturally be applied to solve the problem of network embedding. In many matrix factorisation models, Singular Value Decomposition (SVD) is commonly used in network embedding due to its success at finding a low-rank approximation (Zhan et al., 2018). Non-negative matrix factorisation is often used because of its advantages as an additive model (Li et al., 2019). Vicus (Wang et al., 2017a) is able to better captures local behaviour of networks by constructing a novel

matrix based upon the graph Laplacian that incorporates neighbourhood information explicitly.

2.1.2 Random Walk Based Approaches

An emerging network embedding paradigm comes from natural language processing (NLP). In particular, the skip-gram model and the Word2Vec algorithm that aims to vectorise words and phrases in a Euclidean 'semantic' space such that similar words are mapped close together (Mikolov et al., 2013a; Mikolov et al., 2013b). The principle idea is, given a corpus of words and a particular sentence, generate a 'context' for each input word with the aim of maximising the likelihood of observing context words in the embedding space, given the input word. Similarities are measured by dot products and accordingly, observation probabilities are computed using a multi-layer perception with a linear hidden layer and softmax output. Through the use of sub-sampling and negative sampling (replacing softmax with sigmoid), training can be made very efficient and the resulting embeddings can be obtained from the activation of the hidden units. This idea naturally extends to networks, where sentences are replaced by 'neighbourhood graphs' generated from random walks. Furthermore, the shallow architecture of the skip-gram model has been replaced with multiple non-linear layers to learn the highly non-linear relationships between nodes by adopting a deep learning framework. Examples include Deepwalk (Perozzi, Al-Rfou, and Skiena, 2014) and LINE (Tang et al., 2015).

By introducing additional parameters into the random walk to control a breadth vs. depth first neighbourhood search, node2vec (Grover and Leskovec, 2016) is able to identify neighbourhoods of nodes with high *homophily* and high structural similarity. The use of these parameters to control the random walk, therefore controlled the definition of community and offered great flexibility to the practitioner to customised the search based on exactly what they are looking for. In detail, for a given a source node u, node2vec defines a random walk of length l as a sequence

$n_1 = u, ..., n_l$ defined by the following probability distribution:

$$P(c_i = x | n_{i-1} = v \wedge n_{i-2} = t) = \begin{cases} \alpha_{tx} w_{xv} / Z & \text{if } (x, v) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

where

$$\alpha_{tx} = \begin{cases} \frac{1}{p} & \text{if } |\text{SP}(t, x)| = 0, \\ 1 & \text{if } |\text{SP}(t, x)| = 1, \\ \frac{1}{q} & \text{if } |\text{SP}(t, x)| = 2. \end{cases} \tag{2.2}$$

and Z is a normalising constant and $|\text{SP}(t, x)|$ is the shortest path length between nodes t and x (see section 1.1.7).

In this approach, two parameters are introduced to the random walk: p and q. p is the so-called *return parameter* that controls how far away from the source node u that the walk explores. When it is set to a high value, the likelihood of backtracking is low – ensuring that the walk explores more of the network. On the other hand, when it is high, the walk does not venture far from u. q is the *in-out* parameter: when $q > 1$, the walk is biased to nodes close to t, whereas, $q < 1$ biases towards nodes that are further away. In combination, p and q offer a flexible trade-off between depth-first and breadth-first search.

Random walk based approaches use a sliding window of size c – called the *window* or *context* size – to build a set of pairs of nodes that should be embedded close together. The setting of c has been shown to be robust and depends largely on the sampling budget (Grover and Leskovec, 2016).

2.1.3 Deep Neural Networks

The fundamental problem of network embedding is learning a mapping from one vector space to another. Matrix factorisation based approaches assume that the mapping is linear, however, this may not be the case. Deep neural networks provide a complex non-linear mapping between vector spaces that have seen huge successes in other fields, and so is a natural and appealing choice for network embedding.

SDNE (Wang, Cui, and Zhu, 2016) propose a semi-supervised deep model with a custom loss function based on first- and second-order proximity to avoid biasing the embeddings to maximised the margins between classes and preserve local information. SDAE (Cao, Lu, and Xu, 2016) employ a novel 'surfing' random walk approach sample the network and then train a series of stacked auto-encoders. Furthermore, SiNE (Wang et al., 2017b) use deep learning to learn representations of signed networks. End-to-end solutions, such as PALE (Man et al., 2016), employ network embedding techniques as part of achieving a larger end-goal (in the case of PALE, that is network alignment for predicting anchor links).

2.1.4 Embedding with Attributes and Labels

Embedding of networks with side attributes is challenging as one must always be aware of the trade off between topological and attribute similarity. Some approaches embed based on statistics of attributes, and pairs of attributes (Gibert, Valveny, and Bunke, 2012); and known community structure (Wang et al., 2017c); while others draw from the well known fields of manifold learning and multi-view learning to align the projections based on topology and attributes (Li et al., 2017c); and deep learning has also been used (Liao et al., 2018). Text-assisted Deepwalk (TADW) (Yang et al., 2015) generalizes Deepwalk (Perozzi, Al-Rfou, and Skiena, 2014) to nodes with text attributes. By generalising from regular pixel lattices to arbitrary graphs, it is possible embed and classify entire small graphs (Niepert, Ahmed, and Kutzkov, 2016; Defferrard, Bresson, and Vandergheynst, 2016). Additionally, even multi-layer networks have been embedded (Liu et al., 2017).

Since node labels are distinct from topology and attributes and contain additional information, it can be informative to incorporate them in the embedding process when they are wholly or partially known. The popular Graph Convolutional Network (GCN) (Kipf and Welling, 2016) extend and simplify existing graph convolution approaches to embed nodes in a semi-supervised setting. Furthermore, GraphSAGE (Hamilton, Ying, and Leskovec, 2017) introduces an inductive framework for online learning of node embeddings capable of generalising to unseen nodes. LANE (Huang, Li, and Hu, 2017b) can deal with weighted/unweighted networks and binary/real valued attributes and labels. It employs spectral methods to

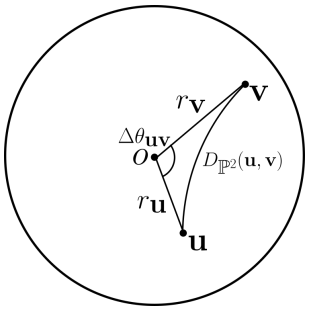


FIGURE 2.1: Hyperbolic law of cosines. Two points $\mathbf{u} = (r_{\mathbf{u}}, \theta_{\mathbf{u}})$ and $\mathbf{v} = (r_{\mathbf{v}}, \theta_{\mathbf{v}})$ are a distance of $r_{\mathbf{u}}$ and $r_{\mathbf{v}}$ from the origin of the Poincaré disk respectively. The angular distance between $\mathbf{u}$ and $\mathbf{v}$ is given by $\Delta\theta_{\mathbf{uv}} = \pi - |\pi - |\theta_{\mathbf{u}} - \theta_{\mathbf{v}}||$. Then the length of the geodesic between $\mathbf{u}$ and $\mathbf{v}$ is given by: $D_{\mathbb{P}^2}(\mathbf{u}, \mathbf{v}) = \operatorname{arccosh}(\cosh(r_{\mathbf{u}})\cosh(r_{\mathbf{v}}) - \sinh(r_{\mathbf{u}})\sinh(r_{\mathbf{v}})\cos(\Delta\theta_{\mathbf{uv}}))$.

find the separate latent embeddings to jointly model and unify topology, attributes and labels. RoSANE Hou, He, and Tang, 2020 introduces a scalable framework based on the Skip-Gram model for embedding sparse networks.

It is worth noting that, by transforming an unweighted graph into a so-called 'flow graph' (Lambiotte et al., 2011) – by weighting links by node expression or attribute similarity, for example – many embedding techniques that are applicable to weighted graphs can be applied to unweighted graphs with node attributes. Additionally, dynamic networks can be transformed into attributed networks by annotating nodes and edges, and then embedding techniques can be applied to preserve dynamics (Iacobelli and Figueiredo, 2016). Furthermore, GloDyNE Hou et al., 2020 adapts Skip-Gram for network embedding in a dynamic setting.

2.1.5 Hyperbolic Network Embedding

As mentioned in section 1.6.5, an emerging popular belief in the literature is that the metric space that underpins complex networks is not Euclidean, but it hyperbolic. One of the most popular hyperbolic embedding models is the Popularity-Similarity (or PS) model (Papadopoulos et al., 2012). This model extends the "popularity is attractive" aphorism of preferential attachment (Barabási and Albert, 1999) to include node similarity as a further dimension of attachment. Informally, nodes 'like to connect to popular' nodes but also nodes that 'so the same thing' (see section 1.1.8). The PS model sustains that the clustering and hierarchy observed in real world networks is the result of this simple principle (Papadopoulos et al., 2012). This model has a

clear interpretation in two dimensional hyperbolic space, where nodes are assigned radial co-ordinates and are placed on a hyperbolic disk, with radial coordinates representing popularity and angular coordinates representing similarity. Then the hyperbolic distance $d_{\mathbb{P}^2}(\mathbf{u}, \mathbf{v})$ between two nodes with polar co-ordinates $\mathbf{u} = (r_{\mathbf{u}}, \theta_{\mathbf{u}})$ and $\mathbf{v} = (r_{\mathbf{v}}, \theta_{\mathbf{v}})$, given by the hyperbolic law of cosines (see figure 2.1) as:

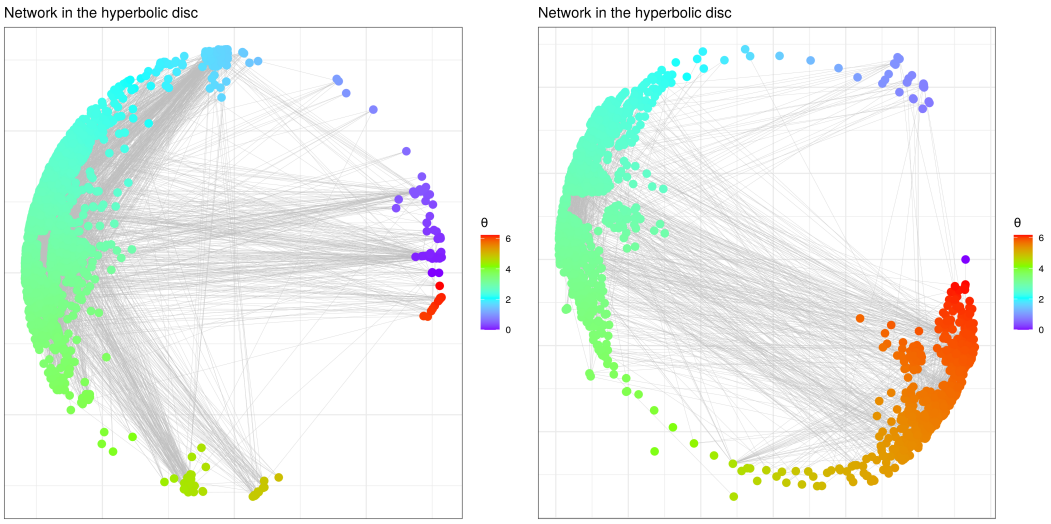
$$D_{\mathbb{P}^2}(\mathbf{u}, \mathbf{v}) = \operatorname{arccosh}\left[\cosh(r_{\mathbf{u}})\cosh(r_{\mathbf{v}}) - \sinh(r_{\mathbf{u}})\sinh(r_{\mathbf{v}})\cos(\Delta\theta_{\mathbf{uv}})\right] \tag{2.3}$$

$$\Delta\theta_{\mathbf{uv}} = \pi - |\pi - |\theta_{\mathbf{u}} - \theta_{\mathbf{v}}|| \tag{2.4}$$

controls their connection probabilities. Nodes with short hyperbolic distances show a higher probability of being connected. HyperMap (HM) (Papadopoulos, Psomas, and Krioukov, 2015), a pioneering PS model embedding method, employed maximum likelihood (ML) to search the space of all PS models with similar structural properties as the observed network, to find the one that fit the original network best. This was later extended by Papadopoulos, Aldecoa, and Krioukov, 2015.

Due to the computationally demanding task of maximum likelihood estimation, heuristic methods were developed to improve upon existing PS model embedding techniques. For example, LABNE (Alanis-Lobato, Mier, and Andrade-Navarro, 2016a) uses Laplacian Eigenmaps was to efficiently estimate the angular coordinates of nodes in the PS model. The same authors later extended LABNE with LABNE+HM, which combined both exact and heuristic approaches to leverage the performance of ML estimation against the efficiency of heuristic search, with the trade-off defined with a user controlled parameter (Alanis-Lobato, Mier, and Andrade-Navarro, 2016b). See figure 2.2 for a pair of example embeddings produced with LABNE+HM. Additionally, many classical manifold learning techniques can be applied in the PS model setting with a framework called *coalescent embedding* (Thomas et al., 2016). Mercator (García-Pérez et al., 2019) also employs both machine learning and maximum likelihood methods to learn PS network embeddings.

The n-dimensional Poincaré ball can provide more degrees of freedom in the embedding process and capture further dimensions of attractiveness than just "popularity" and "similarity". The algorithm proposed in Nickel and Kiela, 2017 embeds networks to the Poincaré ball using retraction updates to optimise an objective that



(A) Cora_ML (B) Citeseer

FIGURE 2.2: Example PS model embedding of the (A) Cora_ML and (B) Citeseer citation networks produced using LABNE+HM (Alanis-Lobato, Mier, and Andrade-Navarro, 2016b). Radial positions are determined based on node degree. Angular co-ordinates are initially provided using Laplacian Eigenmaps, and then optimised using HM within a window size of $\frac{2\pi}{100}$ radians. Node colour corresponds to angular co-ordinate. Network temperature was set to the default 0.1. Network scaling exponents were set to the auto-computed 3.000 (Cora_ML) and 2.866 (Citeseer).

aims to maximize the likelihood of observing true node pairs versus arbitrary pairs of nodes in the network. The authors were able to embed hierarchical text corpora, such that the semantics of word hierarchy was preserved – words with more general meaning – 'mammal', for example – were represented as hyperbolic vectors close to the origin, whereas word with specific meaning – like 'tiger' – could be found close to the boundary of the ball. This approach was later applied by the same authors on the hyperboloid model (Nickel and Kiela, 2018), and extended to embed any arbitrary parameterised object (Dhingra et al., 2018). Furthermore, hyperbolic skip-gram models have been proposed (Chamberlain, Clough, and Deisenroth, 2017; Leimeister and Wilson, 2018), and heterogeneous networks embedded to hyperbolic space (Wang, Zhang, and Shi, 2019).

Interestingly, Wilson et al., 2014 were able to determine the appropriate curvature to embed any symmetric dissimilarity data – selecting any of Spherical, Euclidean or Hyperbolic that is the most appropriate for the data and results in the least optimisation.

It is worth mentioning that trees can be embedded in hyperbolic space without distortion (Sarkar, 2011) and so, some works by embed general graphs to trees and

then compute an exact distortion-free embedding of the resulting tree (De Sa et al., 2018).

For attributed hyperbolic network embedding, graph convolution has recently been generalized to hyperbolic space to allow for non-Euclidean representation learning on attributed networks (Chami et al., 2019).

2.1.6 Scalability of Network Embedding Algorithms

A common hurdle in network embedding is scale-ability. Even an algorithm with $\mathcal{O}(N)$ complexity can perform poorly when the number of nodes in the network N is very large, as it typically is for real-world complex systems. Naturally, this leads to network embedding practitioners to apply simplifications and sampling techniques in order to provide tractable solutions to real-world problems.

Hierarchical Softmax

Softmax (or the normalised exponential function) for network embedding is defined as:

$$p(u \mid v) = \frac{\exp(\mathbf{x}_u^T \mathbf{x}_v)}{\sum_{v' \in V} \exp(\mathbf{x}_u^T \mathbf{x}_{v'})} \tag{2.5}$$

where $u, v \in V$ are nodes in a network with positions $\mathbf{x}_u$ and $\mathbf{x}_v$ respectively. Softmax has been a popular activation function in the machine learning community for decades, but its reliance on computing the the summation over denominator for a single parameter update makes it ultimately unsuitable for network embedding. Because of this, *hierarchical softmax* was proposed (Mikolov et al., 2013b). It is built upon a binary tree where each leaf represents an output, and accordingly transforms the $\mathcal{O}(N)$ complexity of softmax into $\mathcal{O}\left(\log_2(N)\right)$. Probabilities are computed by first building a path from the root of the binary tree to the leaf of interest, and then computing a product of probabilities given by sigmoid functions on all of the internal nodes of the tree.

Negative Sampling

An alternative (and increasingly popular) solution to the challenge of network embedding scale-ability it *negative sampling*. Negative sampling is based on the concept of noise contrastive estimation (NCE), which essentially states that a good model should differentiate fake signal from the real one.

Negative sampling objectives typically take the following form:

$$\log p(u \mid v) = \log \sigma(\mathbf{x}_u^T \mathbf{x}_v) + \sum_{i=1}^{K} \mathop{\mathbb{E}}_{v' \sim P_n(u)} \left[\log \sigma(-\mathbf{x}_u^T \mathbf{x}_{v'}) \right] \tag{2.6}$$

where $\sigma(\cdot)$ is the sigmoid function[1].

$P_n(\cdot)$ is the *noise distribution* that controls the frequency by which a node is selected as a negative sample, which is normally set to be proportional to the *frequency* of that node in the network. For example, a common setting of $P_n(\cdot)$ is the uni-gram distribution to the power of 3/4 (Grover and Leskovec, 2016). In this setting, if a node v has frequency f_v, then its probability for selection is computed as:

$$p_{P_n(\cdot)}(v) := \frac{f_v^{3/4}}{\sum_{v' \in V} f_{v'}^{3/4}} \tag{2.7}$$

'Frequency in the network' can mean many things, from degree to number of occurrences in a corpus of node pairs produce by random walk network sampling (Grover and Leskovec, 2016). Even what constitutes a negative sample is up for debate. Most works, however, will exclude nodes that have a relationship with node u from the set of possible negative samples for node u (Grover and Leskovec, 2016).

K is a hyper-parameter in equation (2.7) that controls the number of negative samples seen for each positive sample. Obviously, the setting of K involves a trade-off between training speed and overall final quality. It has been shown that, for small datasets, $K = 10$ performs well, and for larger ones, K can be set as low as 3 (Grover and Leskovec, 2016).

As we can see from the formulation of negative sampling given in equation (2.7), the original intention was to differentiate signal with logistic regression (Grover and Leskovec, 2016), however, this is not required, so long as the spirit of differentiating

[1]The sigmoid function is defined as $\sigma(x) := \frac{1}{1+\exp(-x)}$.

real signal from fake is preserved. For example, Nickel and Kiela, 2017 use negative sampling as part of an objective function that used softmax.

2.1.7 Evaluation Criteria

Since the problem of network embedding is so general, the measure of a successful embedding is, naturally, largely problem dependent. For our purposes, we follow most common practices and evaluate embedding performance in three ways, each measuring a different and important function of the embedding.

Network Reconstruction

High-quality embeddings should have high-capacity to reflect the original data (Nickel and Kiela, 2017). As the name implies, *network reconstruction* is a measure of the learned embedding to reconstruct the original network. It is an unsupervised measure that relies only on the edges of the network to act as ground truth. For a given embedding, nodes are ranked using the distance/similarity measure of the metric space. The essence of the task is intuitive in that nodes that are adjacent in the original network should be close together in the learned space, whereas nodes that are far apart in the network are embedded far apart in the space.

Since E and $\hat{E}$ provide ground truth labels and nodes can be ranked based on distance – where a smaller distance indicates a higher belief that two nodes should share a link, global metrics, such as Area Under Receiver Operating Characteristic (AUROC) and Average Precision (AP), that evaluate rank are preferred. These measures avoid setting an explicit threshold of distance to say nodes should share a link, but can be used to determine that threshold later down the line (Fawcett, 2006). In this setting, every possible threshold is testing – referring to every unique distance between pairs of nodes of interest. For each setting of threshold, any value greater than the threshold to set to true and the confusion matrix is determined, along with the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

AUROC is the area under the ROC curve that is generated from plotting false positive rate (FPR) on the x-axis against true positive rate (TPR) on the y-axis for

TABLE 2.1: A confusion matrix reporting measures commonly required to evaluate a classifier. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class.

		Ground Truth	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

every possible setting of a threshold. FPR is the ratio of FP to all negatives (N = FP + TN):

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.8}$$

TPR (also called Recall) is the ratio of TP to all positives (P = TP + FN), and is the quality of a predictor to detect as many positives as possible:

$$\text{TPR} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.9}$$

AUROC ranges between 0 and 1, where a value close to 1 indicates that the predictor is high-quality: true positives are ranked before true negatives.

AP is the area under the curve generated from plotting recall on the x-axis against precision on the y-axis, for all possible thresholds. Precision is a measure of a predictors ability to specifically predict true positives while not predicting too many false positives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.10}$$

AP ranges between 0 and 1, where a value of 1 indicates that a predictor has a high capacity to be specific, while detecting as many true positives as possible.

Broadly, two categories of network reconstruction measure exist: global network reconstruction and local network reconstruction. Global network reconstruction looks at node pairs and ranks the pairs of nodes in the edge set E, against a ground truth negative set $\hat{E}$ of node pairs not in E (often sampled from $V \times V \setminus E$ such that $|E| = |\hat{E}|$) (Bojchevski and Günnemann, 2018). Often global network reconstruction would be evaluated using AUROC and AP.

Local network reconstruction looks at each node in the network individually and rank all other nodes by distance to that node (Nickel and Kiela, 2017). The idea is the same: true neighbours should like close and non-neighbours further apart. In this setting, measures are computed for each node and averaged over all nodes in the network. Local network reconstruction is evaluating the embedding as a recommender system – for a query node u, what nodes are similar? Typical local network reconstruction measures are mean Average Precision (mAP), which is the AP for each node in the network, averaged over all nodes in the network:

$$\text{mAP} = \frac{1}{N} \sum_{u \in V} \text{Precision}_u \tag{2.11}$$

where Precision_u refers to the Precision with respect to query node u. Other local network reconstruction measures are precision at k (p@k), which looks at the k closest nodes to a node in the embedding space and reports the precision:

$$\text{p@}k = \frac{\text{number of true recommendations in } k \text{ closest elements}}{\text{number of recommendations in } k \text{ closest elements}} \tag{2.12}$$

Link Prediction

In addition to reconstructing the original data, a common downstream task for embeddings is to predict new links. *Link prediction* is the measure by which embeddings are able to achieve this.

For static networks, link prediction is evaluated by holding out a small random sampling of edges from the network. Depending on the algorithm, care may be taken to ensure that this does not break up connected components or leave any isolated nodes. In this case, as with network reconstruction, a (potentially equal) number of ground truth non-edges are selected. For structurally dynamic networks, link prediction is formulated inductively: using $\mathbb{G}(1), ..., \mathbb{G}(t)$, can we predict $\mathbb{G}(t+1)$?

In any case, once an embedding is learned, the held-out edges are ranked against the non-edges and link prediction is evaluated in the same way as network reconstruction.

Node Classification

Often nodes in complex networks are labelled. *Node classification* is the measure to which node classes can be well separated in the learned space.

Typically the embedding process for node classification is performed in an unsupervised manner – using only topological and, perhaps, node and edge attributes, to generate a low dimensional representation. However, it may be performed in a semi-supervised manner – sometimes partial label information may be used in learning an embedding, in which case node classification is the task of predicting those labels held out from training (Kipf and Welling, 2016). Once an embedding is learned, it is used as input to a classifier (a logistic regressor or support vector classifier, for example) that uses small subsets of node labels to assign labels to every node in the network. The intuition is that high quality embeddings will be able to clearly separate node classes in the low dimensional space and, therefore, the classifier will achieve high performance with respect to the held-out labels.

Common measures of classifier performance include: F_1 score, precision, recall and AUROC. F_1 is the harmonic mean of precision and recall:

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.13}$$

F_1 has been criticised since it gives equal importance to precision and recall. In practice, different types of mis-classifications incur different costs. In other words, the relative importance of precision and recall is an aspect of the problem (Hand and Christen, 2018). Other measures are the Matthew's Correlation Coefficient (MCC) (Chicco and Jurman, 2020), and the G-measure (the geometric mean of Precision and Recall) (Tharwat, 2018).

Since, in general, there are more than two classes in most complex networks, the aforementioned classification measures are reported in the multi-class setting. In this setting, each class is evaluated separately and the final score is computed as an unbiased mean (macro-average) or biased mean based on class frequency (micro-average).

2.2 Directed Network Embedding

When the weight matrix **W** describing the connections of a network is asymmetric, problem 1 becomes the problem of *directed network embedding*.

2.2.1 Euclidean Directed Network Embedding

Many random walk based approaches can handle both directed and undirected networks and are capable of an asymmetric distance measure when both the "source" and "context" embeddings are retained (Chen, Yang, Tang, et al., 2007; Perozzi, Al-Rfou, and Skiena, 2014; Grover and Leskovec, 2016). However, care must be taken to avoid the random walker becoming stuck in a small strongly connected component in the network – skewing the distribution of node samples. The occurrence probability of a node in a walk is some implicit measure of node popularity – perhaps the node has high in-degree or a high probability that a random surfer will teleport to it (Page et al., 1999). However, for directed networks, a pair of nodes forming a feedback loop will trap the random walker – inflating the count of those nodes and forcing the embedding to try "too-hard" to push them apart from the other nodes in the network. Some random walk based algorithms are tailor made to handle this drawback using restart (Zhou et al., 2017).

Furthermore, LINE (with second-order proximity) (Tang et al., 2015) is able to handle directed graphs by assuming that "vertices sharing many connections to other vertices are similar to each other". Asymmetric Transitivity Preservation (ATP) (Sun et al., 2019) aims to directly preserve the inherently asymmetric nature of the transitive relationships in Community Question Answering graphs, again through the use of learning separate source, context representations. Knowledge graphs with heterogeneous (symmetric / asymmetric / one-to-many / many-to-many) relationships have been embedded by Feng et al., 2016.

Embedding of attributed directed networks so far limited to the Euclidean domain. Graph2Gauss (G2G) (Bojchevski and Günnemann, 2018) provides an inductive framework to learn Gaussian node representations, based on learning a Kullback-Leibler divergence based objective. While requiring $2n$ parameters to learn an n-dimension representation of a node, this has the advantage of learning a single

(Gaussian) representation of every node in the network. This is useful for downstream tasks beyond link prediction by providing a single vector representation of a node (the mean of the distribution) upon which to train the downstream model, like a node classifier. Further, the inductive property of G2G allows it to handle the embedding of nodes that were not seen during the training process, by learning a mapping directly from attributes. Despite these desirable features, G2G embeds to Euclidean space and so the hidden hierarchy of elements may not be well preserved in low dimensions.

2.2.2 Hyperbolic Directed Network Embedding

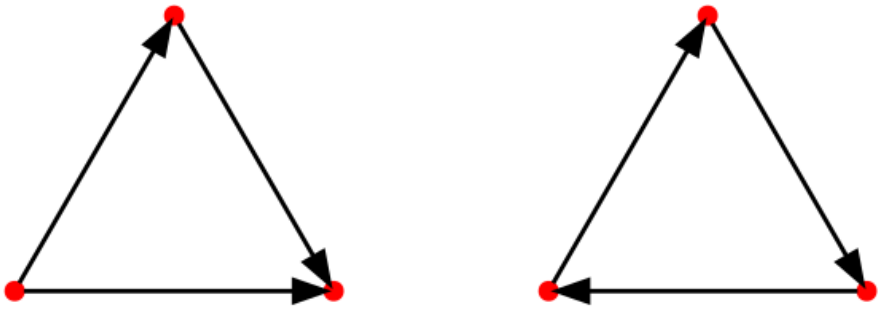
Almost all directed network embedding algorithms embed to Euclidean space. The examples of a non-Euclidean embedding algorithms for directed networks are few. One approach is to construct a bi-partite network[2] from a given directed graph, where each node is transformed into two nodes in the newly constructed network – one for incoming edges and one for the outgoing and then embed to the PS model[3] (Wu, Di, and Fan, 2019). This model has the drawback that there is only free dimension to optimise, constraining the overall representation power of model. Since the radial co-ordinate of the PS model is a fixed function of node degree, only the angular *similarity* co-ordinate is free to optimise.

Directed acyclic graphs (DAGs) can be embedded to the Poincaré disk (Ganea, Becigneul, and Hofmann, 2018; Suzuki, Takahama, and Onoda, 2019). Of course, transforming general complex networks into DAGs naturally results in a loss of information.

2.2.3 Evaluation Criteria

Since directed network embedding is a slight reformulation of problem 1, all of the evaluation measures identified in section 2.1.7 are applicable to evaluate directed network approaches.

[2]See section 1.1.5 for a definition of a bi-partite network.
[3]The PS model is described in section 2.1.5.



(A) (B)

FIGURE 2.3: (A) provides and example of a three node motif containing a three node positive feed-forward loop. The left and top nodes are the inputs, and the right node is the target. Notice that the left input node modulates the top input node. (B) provides an example of a three node motif containing a positive feedback loop.

2.3 Network Architectures and Dynamics

2.3.1 Feed-forward Loops

Intricate dynamic properties have been traditionally associated with cascades of cycles (of proteins) (Pomerening, Sontag, and Ferrell, 2003). Cascades of such cycles form the backbone of most signalling pathways that propagate external stimuli from the membrane to the nucleus or other distant targets (Kholodenko, 2006).

In addition to cascades, another highly enriched directed acyclic motif in molecular networks are *feed-forward loops*. Feed-forward loops (FFLs) are small, three-node motifs containing two input node, one of which modulates the other, and one target node. Figure 2.3A provides an example of one such motif. Note that the terminology is a little confusing: despite being commonly referred to as loops in the literature, FFLs do not contain a cycle in the graph theoretic sense.

FFLs have been well studied in the context of signalling (Mangan and Alon, 2003). They have been found to be responsible for the signal processing functions that perform information transmission:

- **Persistent signal detection**: coherent FFLs have been shown to filter out 'spurious pulses' (Mangan and Alon, 2003; Chou, 2018).
- **Fold-change detection**: incoherent FFLs detect input stimulus fold-change and differentiate this from the noise of the background (Goentoro et al., 2009).

- **Oscillatory signal detection**: incoherent FFLs have also been shown to detect oscillating signals and even count the number of periodic pulses (Zhang et al., 2016a).

- **Pulse shaping**: incoherent FFLs can change the waveform of transmitted pulses. In molecular networks, especially in gene regulatory networks, pulse shaping is a mechanism for speeding response times (Mangan and Alon, 2003).

2.3.2 Feedback Loops

Within molecular and signalling networks, the notion of feedback is one of the most fundamental concepts in biological control. Positive feedback amplifies the signal, whereas negative feedback attenuates it (Kholodenko, 2006). Feedback loops can be used to sense the duration of signals (Kolch, Calder, and Gilbert, 2005). In addition, attractor landscape analysis has revealed that feedback loops control cellular response (Choi et al., 2012). Developmental transcription networks often use motifs comprised positive-feedback loops that are made up of two transcription factors that regulate each other (Alon, 2007), and human gene regulatory networks (GRNs) are enriched in positive-positive two-node loops (Kim et al., 2011b). Furthermore, it has been shown that cell-cycle networks contains a giant backbone motif spanning the entire network nodes that provides the main functional response, and the remaining edges form smaller motifs (Wang et al., 2010).

Positive Feedback Loops

Positive feedback loops are defined as cyclic motifs without any negative feedback loop. Figure 2.3B provides an example of a three-node positive feedback loop. The dynamics generated by positive cyclic motifs are *ultra-sensitive* response and *bi-stability* (or *multi-stability*):

- **Ultra-sensitivity** generates a sigmoid input-output relationship that is steeper than a *Michaelis* (hyperbolic) response. Although ultra-sensitivity can be achieved by cascades and feed-forward loops, robust ultra-sensitivity is mainly realised by positive feedback loops (Mitrophanov and Groisman, 2008).

Ultra-sensitivity amplifies signals, which is an important information processing function for nonlinear information transmission. Usually, positive feedback loops amplify the magnitudes of relative changes, i.e., fold changes, detect by feed-forward loops (Goentoro et al., 2009) or allosteric proteins (Olsman and Goentoro, 2016), rather than absolute changes.

- **Bi-stability** is a special case of switching dynamics generated by positive feedback loops. It has two stable steady-states for a single value of the input. This characteristic indicates that stable steady-states depend on history, i.e., the direction of the dynamics. Such dependence of the state on its history is also called *hysteresis*.

These dynamics enable molecular networks to amplify signals, made binary decisions and store state information. Through artificially evolving dynamical networks that display specific dynamical characteristics of real-world GRNs related to the cell differentiation process (namely, *hysteresis*, and *multi-stationarity*), it has been shown that multi-stationarity, in particular, is driven by a high ratio of PFLs to negative ones (Kim et al., 2008). Furthermore, PFLs have been shown to facilitate switch-like behaviour (Shin et al., 2010).

Negative Feedback Loops

Negative feedback provides adaption to noise and perturbations, which is important for maintaining homeostasis of the whole system. Strong negative feedback can also result in dampened or sustained oscillations for both the input and output nodes:

- **Oscillation** in molecular networks is used to regulates the timing and speed of biological processes (Ferrell Jr, Tsai, and Yang, 2011).

- **Adaptation** is a system's ability to return to basal or near-basal output level after responding to a change in input stimulus even when the change is persistent. Adaptation not only allows signalling networks to more accurately detect changes in a wide range of input, but also plays an important role in maintaining homeostasis against perturbations (Ma et al., 2009).

2.3.3 Higher-Order Network Architectures

Small-scale local network architectures can only give rise to simple small-scale dynamics, like bi-stability and oscillations. However, complex network architectures are necessary to generate complex global dynamics to strike a balance between robustness and flexibility.

General core-periphery structure is ubiquitous in complex systems. The concept was introduced by Borgatti and Everett, 2000 and entails single dense, cohesive core and a sparse, unconnected periphery. Identification required a combinatorial approach. Contemporary methods relax the strict initial definition to search for multiple cores (Rombach et al., 2014; Zhang, Martin, and Newman, 2015), and introducing *residual nodes* that do not belong to any core-periphery structure in the network (Kojaku and Masuda, 2017).

The bow-tie architecture has been well studied for biological networks. Research suggests that the bow-tie architecture has evolved over time to facilitate robust biologic function (Kitano, 2004), and, based on their design, also have inherent but predictable fragilities (Csete and Doyle, 2004). This was studied in detail by Friedlander et al., 2015. Furthermore, the strongly-connected processing cores of bow-tie architectures contain many feedback loops.

In general, the definition of what is meant by 'bow-tie architecture' that was given in section 1.4.5 is one of many. Bow-tie builder (Supper et al., 2009) does not constrain the core of a bow-tie to be strongly connected, merely that it must be minimal and central to a desired set of inputs and outputs. Accordingly, they adopt a greedy heuristic search to identify a core for a set of proteins in order to reconstruct pathways from a protein-protein interaction (PPI) network.

2.3.4 Boolean Network Control

Controlling boolean networks is the act of identifying the set of driver nodes with time-dependent control that can guide the system's entire dynamics. Investigation into many real-world networks has revealed a number of interesting properties (Nepusz and Vicsek, 2012):

- they are much more controllable than their randomised counterparts;

- transcriptional regulatory networks are particularly easy to control;
- scale-free degree distributions have better control-ability properties than uncorrelated networks;
- and, positively correlated in- and out-degrees enhance the control-ability of the proposed dynamics.

A driver node cannot be determined by simple micro-scale topological measures – for example, driver nodes tend not be high-degree nodes (Liu, Slotine, and Barabási, 2011). Despite this, it has been shown that there is a link between a node's topological position in the underlying hierarchical structure of the network to its ability to control a directed weighted network (Liu, Slotine, and Barabási, 2012).

Since the task of driver node identification is always subject to the constraint that the set of driver nodes is as small as possible, often heuristic searches are employed. For example, a genetic algorithm (GA) was employed to identify the minimal set of 'control nodes' for a number of GRNs (Kim, Park, and Cho, 2013). As mentioned in section 1.5.10, control nodes refer to the set of nodes in the network that must have their value pinned to their value in a desired attractor state, such that the entire state landscape of the resulting network transforms into the basin of attraction for that attractor state. In other words, every initial network state is guaranteed to converge to a desired final state. This opens to the door to promising novel drug target discovery.

2.3.5 Reducing Complexity of Boolean Networks

Since enumeration of the entire state landscape of a general GRN is intractable[4], it is necessary to reduce its complexity. By reducing a GRN to a kernel, by preserving important genes, Kim et al., 2011a were able to reduce the complexity of the network dramatically, while preserving many known drug targets. Furthermore, decomposition of large boolean networks into directed acyclic graphs (DAGs), where each node represents a strongly connected component (SCC) in the original network, can reduce the complexity of finding attractors (Zhao, Kim, and Filippone, 2013; Su, Pang, and Paul, 2019).

[4]Recall, a boolean network of N nodes will have a state landscape containing 2^N unique states.

2.3.6 Evaluation Criteria

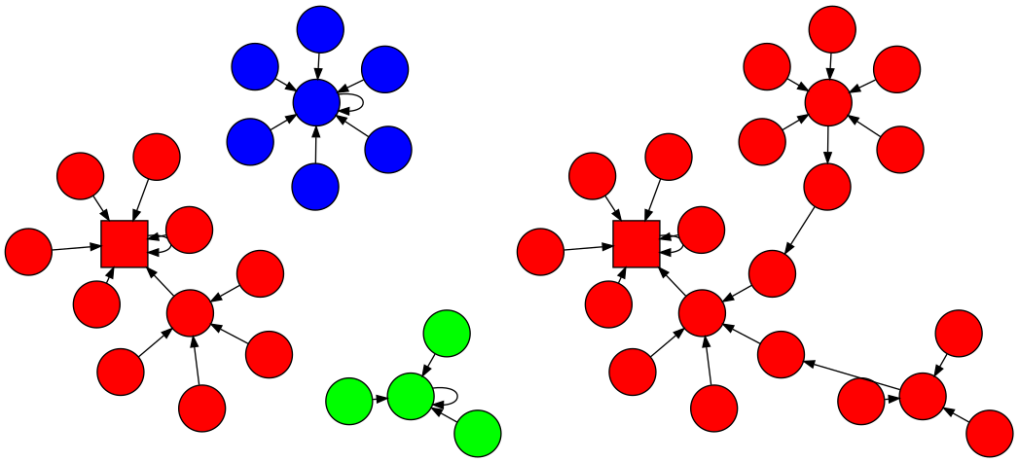
The third research question of this thesis is concerned with "driving global network behaviour". Driving network behaviour here refers to controlling the expression of subsets of nodes in a boolean network to guide overall network behaviour.

Since attractors govern cell fate (Huang et al., 2005), we define the measure of the impact made to the behaviour of a boolean network to be the impact made in terms of its attractors (Kim, Park, and Cho, 2013). To determine the attractors of a boolean network, its state landscape must be computed. If the network is too big, then this is intractable and so the state landscape is commonly sampled (Helikar et al., 2008). Sampling typically involves randomly selecting a number of initial states – say, 10000 – and computing the resulting attractors for those sampled states using the update scheme of interest (Helikar et al., 2008; Kim, Park, and Cho, 2013). Once attractors for a boolean network have been determined, the impact of controlling a set nodes can be measured. This is achieved by pinning that value of the nodes and determining the resulting attractors for the controlled network. We are concerned with one primary measure of control: control kernels.

Control Kernel Identification

As discussed in section 1.5.10, a control kernel is a minimal set of nodes required to drive a network to converge to a desired target state (or set of states) (Kim, Park, and Cho, 2013). Figure 2.4 provides an example of the effect that a control kernel has upon the state transition graph of a small boolean network. To evaluate the success of a potential control kernel, a desired target attractor is selected (commonly this is attractor with the largest basin – corresponding to 'normal cell function' – but that is not required), and the set of nodes in the potential kernel set have their values pinned to their corresponding values in the target attractor. In the case that the desired attractor is cyclic, then a randomly selected state within that attractor is selected (Kim, Park, and Cho, 2013).

A successful control kernel would force all initial conditions to converge to the desired attractor, after an arbitrary number of updates. As such, the measure of kernel success is binary: either this condition is satisfied or it is not. Two kernels



(A) STG of the original boolean network (B) STG of the controlled boolean network

FIGURE 2.4: Example of the effect of a control kernel on the state transition graph (STG). (A) provides an example of (a subset of) the STG of a small boolean network. Each node represents a boolean network state and an edge between states signifies that a state transitions into the other after a network update. It shows three steady state attractors, identified as the states with self loops. For each attractor, all states in the corresponding basin are coloured in the same colour. The red square denotes the primary attractor (the attractor with the largest basin) which has been set as the target attractor for control. (B) shows the STG of the same network after control. Now all states belong to the basin of the target attractor, meaning that any initial network condition is guaranteed to eventually converge to the desired attractor.

may be compared based on size, where a smaller kernel is preferred over a larger one.

Naturally, when the state space is large, and enumerating all attractors for the network is computationally infeasible, the sampling strategy mentioned above is applied to select a number of random initial states, and full basin sizes are estimated based on the proportion of states in the partially reconstructed STG that fall in the basin of an attractor.

2.4 Chapter Summary

This chapter has reviewed relevant works in the fields of directed and undirected network embedding, network architectures and network control.

For network embedding, we had previously defined the general problem of network embedding in problem 1. We saw that the problem is general and ultimately depends on the downstream tasks of interest. Accordingly, approaches to the task of network embedding are varied and diverse. Solutions in the literature broadly fall into two categories: shallow (section 2.1.1) and deep (section 2.1.3). For both

approaches, random walks are commonly used to sample the network, allowing for efficient sampling of the network, while allowing for the easy incorporation of node and/or edge attributes. In addition, the literature has shown that a hyperbolic metric space can better explain all of the key characteristics of complex networks, compared to Euclidean spaces. However, few hyperbolic network embedding approaches are able to generalise to attributed, or even weighted, networks.

Following this, in section 2.2, we review literature related to directed network embedding, a more challenging variation of problem 1, since most embedding approaches impose a symmetric measure of similarity over nodes in the target space. We have seen that random walks can sample directed networks efficiently, with the caveat that walks must be designed to avoid the walker becoming stuck. Furthermore, the output of any directed network embedding algorithm must accommodate for an asymmetric measure of similarity between nodes. This may include learning two representations of each node in the network, or embedding to probability distributions and using a pseudo-metrics, like Kullback-Leibler divergence[5].

In section 2.3, we review known properties of network architectural features. We saw the importance of both feed-forward and feedback loops in signal processing. In particular, we note that feed-forward loops are responsible for the detection of persistent signals, positive feedback loops amplify signal, and negative feedback loops are responsible for robustness to perturbations. We then examined techniques to uncover larger scale architectural features in dynamic networks and note their role in the facilitation of signal flow as well emphasising that the cores of bow-tie architectures defined in section 1.4.5 are highly enriched with feedback loops, whereas feed-forward loops are found the in- and out-components. Finally, in section 2.3.4, we reviewed the topic of network control, its relationship to potential drug target candidates, as well as a common evaluation measure for network control.

[5]We refer to Kullback-Leibler divergence as 'pseudo-metric' since it does not satisfy all the necessary conditions for a function to be considered a metric, but it does generate a topology on the space of probability distributions.

Chapter 3

Hyperbolic Embedding of Weighted and Attributed Networks

Previously, we have introduced the problem of attributed network embedding. The literature review reported a wide variety of solutions, but none that could handle general attributed and weighted networks. Furthermore, some approaches could not directly control the trade-off between topology and attributes. This chapter introduces our approach to address this gap Section 3.1 states the importance of the problem and the motivation behind our solution. Section 3.2 introduces our solution in detail. Section 3.3 validates our approach on a number of benchmark datasets against all of the metrics introduced in section 2.1.7. Finally, section 3.4 summarizes the chapter.

3.1 Introduction

As introduced previously, the success of machine learning algorithms often depends upon data representation (Cui et al., 2018). Unsupervised representation learning – learning alternative (low dimensional) representations of data – has become common for processing information on non-Euclidean domains, such as complex networks. Prediction over nodes and edges requires careful feature engineering (Grover and Leskovec, 2016) and representation learning leads to the extraction of features from a graph that are most useful for downstream tasks, without careful design or a priori knowledge.

Our literature review has suggested that an emerging representation learning approach for complex networks is hyperbolic embedding (see section 2.1.5). This approach is based on compelling evidence that the underlying metric space of many complex networks is hyperbolic (Krioukov et al., 2010). A hyperbolic space can be interpreted as a continuous representation of a discrete tree structure that captures the hierarchical organisation of elements within a complex system (Krioukov et al., 2010). Furthermore, hyperbolic metric spaces have been shown to explain other characteristics typical to complex networks, characteristics such as clustering (Krioukov et al., 2010) and the small world phenomenon (Bianconi and Rahmede, 2017). Hyperbolic spaces therefore offer a natural continuous representations of hierarchical complex networks (Krioukov et al., 2010).

However, existing hyperbolic embedding approaches cannot deal with attributed networks, of which nodes (entities) are richly annotated with attributes (Hamilton, Ying, and Leskovec, 2017). For example, a paper within a citation network may be annotated with the presence of keywords, and the people in a social network might have additional information such as interests, hobbies, and place of work. These attributes might provide additional proximity information to constrain the representations of the nodes. Therefore, incorporating node attributes can improve the quality of the final embedding, with respect to many different downstream tasks (Hamilton, Ying, and Leskovec, 2017).

This chapter proposes the first hyperbolic embedding method for attributed networks called HEAT. The intuition behind HEAT is to extract training samples from the original graph, which can capture both topological and attribute similarities, and then learn a hyperbolic embedding based on these samples. To extract training samples, a novel random walk algorithm with a teleport procedure is developed. The purpose of this walk is to capture phantom links between nodes that do not necessarily share a topological link, but have highly similar attributes. To learn the embeddings from these extracted samples, HEAT employs a novel learning objective that is optimised using full Riemannian stochastic gradient descent (RSGD) in hyperbolic space.

Thorough experimentation shows that HEAT can achieve better performance on

several downstream tasks compared with several state-of-the-art embedding algorithms. As a general framework, HEAT can embed both unattributed and attributed networks with continuous and discrete attributes, which opens the door to hyperbolic manifold learning for a wide range of complex networks.

3.2 HEAT: Hyperbolic Embedding of Attributed Networks

3.2.1 Hyperbolic Attributed Network Embedding: Problem Definition

We reformulate problem 1 for hyperbolic attributed network embedding as follows: We consider a network of N nodes given by the set V with $|V| = N$. We use E to denote the set of all interactions between the nodes in our network. $E = \{(u, v)\} \subseteq V \times V$. We use the matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ to encode the weights of these interactions, where $\mathbf{W}_{uv}$ is the weight of the interaction between node u and node v. We have that $\mathbf{W}_{uv} \neq 0 \iff (u, v) \in E$. If the network is unweighted then $\mathbf{W}_{uv} = 1$ for all $(u, v) \in E$. Furthermore, the matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$ describes the attributes of each node in the network. These attributes may be discrete or continuous. Edge attributes could be handled by transforming them into node attributes shared by both nodes connected by the edge. If the network does not have attributes, then we set $d = N$ and $\mathbf{X} = \mathbf{I}_N$ to the N-dimensional identity matrix.

Problem 2. *Given a network, described by $\mathbb{G} = (V, \boldsymbol{W}, \boldsymbol{X})$, find a set of low-dimensional vectors on the n-dimensional hyperboloid $\{\boldsymbol{x}_v \in \mathbb{H}^n \mid v \in V\}$, with $n \ll N$. Hyperbolic distance, given by geodesics on the hyperboloid (see section 1.7.5), reflects node similarity in the embedding space. The described problem is unsupervised.*

3.2.2 HEAT Overview

In order to solve problem 2, we introduce our algorithm HEAT. Our proposed approach consists of two main components:

1. A novel network sampling algorithm based on random walks to extract samples than can capture both topological and attribute similarity.

2. A novel learning algorithm that can learn hyperbolic embeddings from training samples using Riemannian stochastic gradient descent in hyperbolic space.

3.2.3 Sample the Network using Random Walks with Jump

We propose a novel random-walk procedure to obtain training samples that capture both topological and attribute similarity. Random walks have been proposed in the past as a robust sampling method of elements from structured data, such as graphs, since they provide an efficient, flexible and parallelise-able sampling method (Perozzi, Al-Rfou, and Skiena, 2014). For every node in the network, several walks with a fixed length l are performed (Grover and Leskovec, 2016). We note that random walks traditionally take into account only first-order topological similarity, that is: nodes are similar if they are connected in the network. However, additional topological similarity can also be considered. For example, second-order similarity between nodes (that is: the similarity of neighbourhoods) could be incorporated into the topological similarity matrix using a weighted sum (Wang, Cui, and Zhu, 2016). We leave this as future work.

We propose that, in addition to standard random walks which capture topological similarity, we use attribute similarity to 'jump' the random walker to the nodes with similar attributes. To this end, we define the attribute similarity $\mathbf{Y}$ as cosine similarity of the attribute vectors of the nodes. We assign a value of 0 similarity to any negative similarity values. That is:

$$Y_{uv} = \max\left[\frac{\mathbf{X}_u^T \mathbf{X}_v}{||\mathbf{X}_u|| ||\mathbf{X}_v||}, 0\right] \tag{3.1}$$

where $||\cdot||$ is the Euclidean norm. We select cosine similarity as it can readily handle high dimensional data well without making a strong assumption about the data. We propose HEAT as a general framework and, so, can change the cosine similarity to a more sophisticated and problem-dependant measure of pairwise node attribute similarity.

To define the probability of moving from a node to another based on both topological and attribute similarity, we then additionally define $\bar{\mathbf{W}}$ and $\bar{\mathbf{Y}}$ to be the row-normalises versions of the weight matrix $\mathbf{W}$ and attribute similarity matrix $\mathbf{Y}$ respectively. Each row in $\bar{\mathbf{W}}$ and $\bar{\mathbf{Y}}$ describes a discrete probability distribution corresponding to the likelihood of jumping from one node to the next according to either topological similarity or attribute similarity. In detail, the entry $\bar{\mathbf{W}}_{uv}$ encodes

the probability of moving from node u to v based on the strength of the topological link between u and v, and $\bar{\mathbf{Y}}_{uv}$ likewise encodes the teleport probability based on attribute similarity.

Controlling the Trade-Off Between Topology and Attributes

To control the trade-off between topology and attributes, we introduce the hyper-parameter $0 \leq \alpha \leq 1$. Formally, we use i to denote the ith node in the walk ($x_0 = s$), and for each step $i = 1, 2, ..., l$ in the walk, we sample $\pi_i \sim U(0, 1)$ and determine the ith node as follows:

$$P(x_i = v \mid x_{i-1} = u) = \begin{cases} \hat{\mathbf{W}}_{uv} & \text{if } \pi_i < \alpha, \\ \hat{\mathbf{Y}}_{uv} & \text{otherwise.} \end{cases} \tag{3.2}$$

We follow previous works and consider nodes that appear within a maximum distance of each other in the same walk to be context pairs (Grover and Leskovec, 2016; Perozzi, Al-Rfou, and Skiena, 2014). We call this maximum distance the "context-size" and it is a hyper-parameter that controls the size of a local neighbourhood of a node. Previous works show that increasing context size typically improves performance, at some computational cost (Grover and Leskovec, 2016). All of the source-context pairs are added into a set $\mathcal{D}$.

3.2.4 Setting α

For practical applications, to set the value of the hyper-parameter α that controls the trade-off between topology and similarity in the sampling process, we suggest the following approach: First, randomly sample a proportion of edges from the network and remove them. Next, select an equal number of 'non-edge' node pairs $(u, v) \in V \times V \setminus E$. These two edge sets will form a validation set. Then perform HEAT to generate hyperboloid embeddings for a range of values of α. Removed edges can be ranked against the sampled non-edges and a global ranking measure, such as AUROC or AP, could be used to select a value for α. This procedure is performed in

section 3.3.4, where we evaluate link prediction. As we show in section 3.3.6, HEAT is robust in general to the setting of α for three common downstream tasks.

3.2.5 Hyperboloid Embedding Learning

For the hyperboloid embedding learning procedure of HEAT, we aim to maximize the probability of observing all of the pairs in $(u, v) \in \mathcal{D}$ in the low-dimensional hyperbolic space.

We define the probability of two nodes sharing a connection to be a function of their distance in the embedding space. This is motivated by the intuition of network embedding that nodes separated by a small distances share a high degree of similarity and should, therefore share a high probability of connection, and nodes very far apart in the embedding space should share a low probability of connection. This principle forms the basis of an objective function that is optimised by HEAT to learn node embeddings.

We make the common simplifying assumption that a source node and neighbourhood node have a symmetric effect over each other in feature space (ie: $P((u, v)) = P((v, u))$ for all $u, v \in V$) (Grover and Leskovec, 2016). To this end, we define the symmetric function:

$$\hat{P}((u, v)) := -D^2_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v}) \tag{3.3}$$

to be the un-normalised probability of observing a link between source node u and context node v, where $\mathbf{u}$ and $\mathbf{v}$ are their respective hyperbolic positions. We square the distance because this leads to stable gradients[1]. We normalise the probability using:

$$P((u, v)) := \frac{1}{Z(u)} \exp\left[\hat{P}((u, v))\right] \tag{3.4}$$

$$Z(u) := \sum_{v' \in V} \exp\left[\hat{P}((u, v'))\right] \tag{3.5}$$

where $Z(u)$ is a normalising partition function.

[1] For $\mathbf{x} \in \mathbb{H}^n$, and $\mathbf{x}' \in \mathbb{H}^n$, $\lim_{\mathbf{x}' \to \mathbf{x}} \langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^{n:1}} \to -1$ and $\lim_{x \to -1} \partial_x \operatorname{arccosh}^2(-x) \to 2$. Contrast this with $\lim_{x \to -1} \partial_x \operatorname{arccosh}(-x) \to \infty$ (De Sa et al., 2018).

Negative Sampling

Computing the gradient of the partition function $Z(u)$ involves a summation over all nodes $v \in V$, which for large networks, is prohibitively computationally expensive (Grover and Leskovec, 2016). Following literature (see section 2.1.6), we overcome this limitation through *negative sampling*. We define the set of negative samples for u as the set of $v \in V$ for which we observe no relation with u:

$$\text{Neg}(u) := \{v \in V \mid (u, v) \notin \mathcal{D}\} \tag{3.6}$$

We further define:

$$\text{Neg}_K(u, v) := \{x_i \sim^{P_n} \text{Neg}(u) \mid i = 1, 2, ..., K\} \cup \{v\} \tag{3.7}$$

to be a random sample with replacement of size K from the set of negative samples of u, according to a noise distribution P_n including v. Following Grover and Leskovec, 2016, we set $P_n = U^{\frac{3}{4}}$, the uni-gram distribution raised to the $\frac{3}{4}$ power[2].

We aim to represent the obtained distribution of pairs in a low-dimensional hyperbolic space. To this end, we formulate a loss function L that encourages maximising the probability of observing all positive sample pairs $P((u, v))$ for all $(u, v) \in \mathcal{D}$ and minimising the probability of observing all other pairs. To this end, we define the loss function L for an embedding $\Theta = \{\mathbf{u} \in \mathbb{H}^n \mid u \in V\}$ to be the the mean of negative log-likelihood of observing all the source-context pairs in $\mathcal{D}$, against the negative sample noise:

$$L(\Theta) = -\frac{1}{|\mathcal{D}|} \sum_{(u,v)\in\mathcal{D}} \log \left[\frac{\exp\left(-D^2_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v})\right)}{\sum_{v' \in \text{Neg}_K(u,v)} \exp\left(-D^2_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v}')\right)} \right] \tag{3.8}$$

The numerator of equation (3.8), $\exp\left(-D^2_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v})\right)$, is concerned with the hyperbolic distance between nodes u and v in the positive sample set $\mathcal{D}$. Minimising L involves minimising the distance between $\mathbf{u}$ and $\mathbf{v}$ in the embedding space. The denominator $\sum_{v' \in \text{Neg}_K(u,v)} \exp\left(-D^2_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v}')\right)$ is a sum over all v' in a given sample

[2]The probability that a node is selected as a negative sample is proportional to its occurrence probability, that we define to be the number of times that it appeared over all the random walks.

of size K of the negative samples for node u. Minimising L involves maximising this term, thereby pushing $\mathbf{u}$ and $\mathbf{v}'$ far apart in the embedding space. Overall, we observe that minimising L involves maximising $P((u,v))$ for all $(u,v) \in \mathcal{D}$ as required. This encourages source-context pairs to be close together in the embedding space, and u to be embedded far from the noise nodes v' (Nickel and Kiela, 2017).

3.2.6 Optimisation

Since we use hyperboloid model, unlike some previous works (Nickel and Kiela, 2017; De Sa et al., 2018) that use the Poincaré ball model and approximate gradients with retraction updates, we are able to use full Riemannian optimisation and so our gradient computation is exact and possesses a simple form (Wilson and Leimeister, 2018; Nickel and Kiela, 2018). We follow a three step procedure in to compute gradients and then update hyperbolic coordinates (Nickel and Kiela, 2018). The procedure is based on the fact that the loss function L is defined over the whole ambient Minkowski space $\mathbb{R}^{n:1}$, which is further defined over $\mathbb{H}^n \subset \mathbb{R}^{n:1}$. Therefore, for a given point on the hyperboloid $\mathbf{u} \in \mathbb{H}^n$, we can compute the gradient of L with respect to $\mathbf{u}$, denoted $\nabla_{\mathbf{u}}^{\mathbb{H}^n} L \in T_{\mathbf{u}}\mathbb{H}^n$. Then to perform gradient descent optimisation, we move $\mathbf{u}$ along $-\nabla_{\mathbf{u}}^{\mathbb{H}^n} L$ by a small amount η to $\mathbf{x} = -\eta \nabla_{\mathbf{u}}^{\mathbb{H}^n} L \in T_{\mathbf{u}}\mathbb{H}^n$. Finally we map $\mathbf{x}$ back to $\mathbb{H}^n$ using an exponential mapping.

To compute the gradient of L for vector $\mathbf{u} \in \mathbb{H}^n$ with respect to the hyperboloid $\nabla_{\mathbf{u}}^{\mathbb{H}^n} L$, we first compute the gradient with respect to the Minkowski ambient space $\mathbb{R}^{n:1}$ as:

$$\nabla_{\mathbf{u}}^{\mathbb{R}^{n:1}} L = \left(\frac{\partial L}{\partial u^1}\bigg|_{\mathbf{u}}, \ldots, \frac{\partial L}{\partial u^n}\bigg|_{\mathbf{u}}, -\frac{\partial L}{\partial u^{n+1}}\bigg|_{\mathbf{u}} \right) \tag{3.9}$$

It follows from equation (3.9) that:

$$\nabla_{\mathbf{u}}^{\mathbb{R}^{n:1}} \langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{R}^{n:1}} = \mathbf{v} \tag{3.10}$$

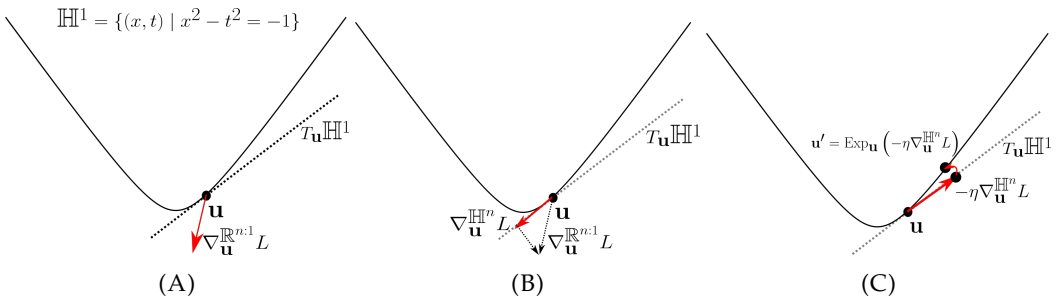


FIGURE 3.1: Three step optimisation on $\mathbb{H}^n$. (A) provides a representation of $\mathbb{H}^1$ a one dimensional manifold in two dimensional Minkowski space $\mathbb{R}^{1:1}$. One point on $\mathbb{H}^1$ is highlighted: $\mathbf{u}$. The red arrow is an example $\nabla^{\mathbb{R}^{n:1}}_{\mathbf{x}_u} L$ vector. Finally, $T_{\mathbf{u}}\mathbb{H}^n$ is given by the dotted black line. (B) highlights the component of $\nabla^{\mathbb{R}^{n:1}}_{\mathbf{x}_u} L$ lying on $T_{\mathbf{u}}\mathbb{H}^n$. Finally, (C) presents the mapping from $-\eta \nabla^{\mathbb{H}^n}_{\mathbf{u}} L$ back to the Hyperboloid using the exponential map $\mathrm{Exp}_{\mathbf{u}}$.

Let $o_{uv} := -D^2_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v})$ and $\mathrm{Neg}_K(u) := \bigcup_{\{v|(u,v)\in\mathcal{D}\}} \mathrm{Neg}_K(u, v)$. Then:

$$\nabla^{\mathbb{R}^{n:1}}_{\mathbf{u}} L = \frac{1}{|\mathcal{D}|} \sum_{v \in \mathrm{Neg}_K(u)} (\delta_{vv'} - P((u,v))) \cdot \nabla^{\mathbb{R}^{n:1}}_{\mathbf{u}} o_{uv'} \tag{3.11}$$

and

$$\nabla^{\mathbb{R}^{n:1}}_{\mathbf{u}} o_{uv} = 2 \cdot \frac{D_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v})}{\sqrt{\langle \mathbf{u}, \mathbf{v} \rangle^2_{\mathbb{R}^{n:1}} - 1}} \cdot \mathbf{v} \tag{3.12}$$

where $\delta_{vv'}$ is the Kronecker delta function[3].

We then use the vector projection formula to compute the projection of the ambient gradient to its component in the tangent space:

$$\nabla^{\mathbb{H}^n}_{\mathbf{u}} L = \nabla^{\mathbb{R}^{n:1}}_{\mathbf{u}} L + \langle \mathbf{u}, \nabla^{\mathbb{R}^{n:1}}_{\mathbf{u}} L \rangle_{\mathbb{R}^{n:1}} \cdot \mathbf{u} \tag{3.13}$$

Having computed the gradient component in the tangent space of $\mathbf{u}$, we apply the hyperboloid exponential map (equation (1.32)) to take a vector $\mathbf{x} \in T_{\mathbf{u}}\mathbb{H}^n$ to its corresponding point on the hyperboloid.

Altogether, we have that the three-step procedure for computing the new position of $\mathbf{u}$, with learning rate η is:

1. Calculate ambient gradient $\nabla^{\mathbb{R}^{n:1}}_{\mathbf{u}} L$ (equations (3.11) and (3.12)),

[3] $\delta_{vv'} = \begin{cases} 1, \text{ if } v = v', \\ 0, \text{ otherwise.} \end{cases}$

TABLE 3.1: Network statistics. *Key:* N is the number of nodes, $|E|$ is the number of edges, d is the dimension of node features, y is the number of classes.

Network	N	$\|E\|$	d	y
Cora_ML (Bojchevski and Günnemann, 2018)	2995	8416	2879	7
Citeseer (Bojchevski and Günnemann, 2018)	4230	5358	2701	6
Pubmed (Bojchevski and Günnemann, 2018)	18230	79612	500	3
PPI (LCC) (Hamilton, Ying, and Leskovec, 2017)	3480	54806	50	121
MIT (Hou, He, and Tang, 2020)	6402	251230	2804	32

2. Project $\nabla_{\mathbf{u}}^{\mathbb{R}^{n:1}} L$ to tangent $\nabla_{\mathbf{u}}^{\mathbb{H}^n} L$ (equation (3.13)),

3. Set $\mathbf{u} = \text{Exp}_{\mathbf{u}} \left(-\eta \nabla_{\mathbf{u}}^{\mathbb{H}^n} L\right)$ (equation (1.32)).

Figure 3.1 provides an example of this procedure operating on $\mathbb{H}^1$.

3.2.7 Complexity Analysis

Building the attribute similarity matrix has complexity $O(N^2)$. Performing random walks has time complexity $O(sNl)$, where s is the number of walks starting from each node. This returns sN walks. The sliding window with context-size c will give at $sN(l-c+1)(c-1)$ training pairs. To learn node embeddings, we optimise equation (3.8) by feeding in all training pairs. The complexity for each pair is $O(1+K)$ where K is the number of negative samples, therefore the overall complexity of learning is $O((1+K)|\mathcal{D}|)$.

3.3 Experimental Validation

3.3.1 Datasets

We evaluate HEAT on three citation networks (Bojchevski and Günnemann, 2018), one PPI network (Hamilton, Ying, and Leskovec, 2017), and one social network for MIT university (Hou, He, and Tang, 2020). We select these citation networks because they are a common set of networks used to evaluation embedding algorithms (Kipf and Welling, 2016; Bojchevski and Günnemann, 2018). We select the PPI network to evaluate performance on a different network type than citation. For the PPI network, (Hamilton, Ying, and Leskovec, 2017) use positional gene sets, motif gene sets and immunological signatures as features and gene ontology sets as labels, collected

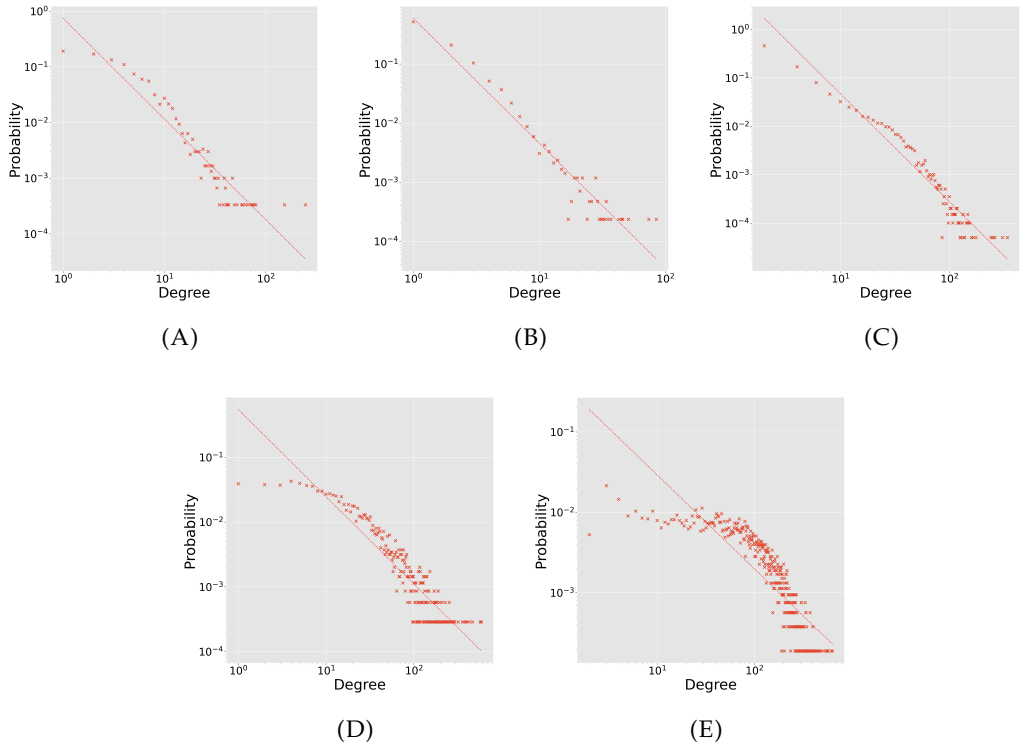


FIGURE 3.2: Degree distributions and statistics of the five network datasets. Both axes are on a log scale. *Left to right:* (A) Cora_ML, (B) Citeseer, (C) Pubmed, (D) PPI, and (E) MIT.

from the Molecular Signatures Database (Subramanian et al., 2005). Each connected component represents a different tissue sample. For our experiments, we select the largest connected component. Features for all networks were scaled to have a mean of 0 and a standard deviation of 1. We also use a social network for MIT university composed from Facebook friendships. Following Hou, He, and Tang, 2020, we use the "year" attribute as a class label and take the remaining six attributes as input attributes using a one-hot encoding. Table 3.1 shows the statistics of these five networks. Figure 3.2 shows log-log plots of the degree distributions of the five networks. We observe that all five follow a broadly scale-free degree distribution, characterised by a straight line on a log-log plot, which suggest that a hyperbolic embedding space is a suitable choice for these networks.

3.3.2 Benchmark Algorithms and Settings

Table 3.2 details all of the benchmark algorithms. For all benchmark methods we adopt the original source code. We train all methods in an unsupervised manner. For hyper-parameter settings we follow the suggestions of the original papers. For

TABLE 3.2: Description of benchmark algorithms

Algorithm	Description
DEEPWALK (Perozzi, Al-Rfou, and Skiena, 2014)	One of the most successful representation learning approaches based on random walks. Considers only structural information.
ATTRPURE (Hou, He, and Tang, 2020)	Performs SVD on a devised attribute similarity matrix.
TADW (Yang et al., 2015)	Uses matrix factorisation to jointly model attribute and structural information.
AANE (Huang, Li, and Hu, 2017a)	A distributed alternative to TADW.
SAGEGCN (Kipf and Welling, 2016)	Graph convolutional network (GCN) originally designed for semi-supervised learning but adapted in (Hamilton, Ying, and Leskovec, 2017) for unsupervised learning.
N&K (Nickel and Kiela, 2017)	Hyperbolic embedding approach based only on structural information.

DEEPWALK, we set walks per node to 10, walk length to 80, context size to 10 and top-k value to 30. For TADW, and AANE, we set the balancing factors to 0.2, 0.05, and 0.8 respectively. For SAGEGCN, we set learning rate to 0.001, dropout rate to 0.5, batch size to 512, normalisation to true, weight decay rate to $1e-4$ and epochs 100. For N&K, we set learning rate to 1.0, epochs to 1500, number of negative samples to 10, batch size to 512, and burn-in to 20 epochs. For HEAT, we set learning rate 1.0, epochs to 5, number of negative samples to 10, batch size to 512, context-size to 10, walks per node to 10, and walk length to 80. We fix $\alpha = 0.2$ for all experiments, which we determined using the procedure described in section 3.2.4 on the Cora_ML network. As shown in section 3.3.6, the performance of HEAT is robust to the setting of α. For each experiment, we perform 30 independent runs of all algorithms and report the mean result.

3.3.3 Network Reconstruction

As identified in section 2.1.7, we following common practice and use network reconstruction to evaluate the capacity of the learned embeddings to reflect the original

data (Nickel and Kiela, 2017). After training our model to convergence upon the complete information, we compute distances in the embedding space between all pairs of nodes according to both models.

Table 3.3 provides a summary of the network reconstruction results. From this we see that HEAT has a high capacity for learning network structure, even at low dimensions. Furthermore, by incorporating attributes, performance increased further on two out of the five networks studied. Table 3.4 provides statistics from a one-sided t-test for network reconstruction. We compare against the best bechmark aglroithm for each network based on AP.

3.3.4 Link Prediction

Following literature (see section 2.1.7), to evaluate the link prediction ability of the learned embeddings, we randomly select 10% of the edges in the network and remove them (Nickel and Kiela, 2017). We then randomly select also an equal number of true non-edges in the network. An embedding is learned for each incomplete network and pairs of nodes are ranked by distance. Table 3.5 provides a summary of the link prediction results. T-test statistics of a one-sided t-test between $\text{HEAT}_{\alpha=0.2}$ and the best ranked benchmark algorithm according to AP are provided in table 3.7. From this, we see that HEAT is capable of highly competitive link prediction ability with and without attributes. We see that the inclusion of attributes improves performance on three out of five networks and suggest that this is because of the high level of *homophily*[4] in citation networks.

3.3.5 Node Classification

To evaluate node classification, we learn an embedding, using complete topological and attribute information with no knowledge of class labels. We then use an out-of-the-box Support Vector Classifier (SVC) to evaluate the separation of classes in the embedding space For hyperbolic embeddings (HEAT and N&K), we first project to the n-dimensional Klein model of hyperbolic space $\mathbb{K}^n$ (see section 1.7.4) , which preserves straight lines (Cannon et al., 1997). For the PPI network, each protein has

[4]Here, 'homophily' refers to the preference for nodes to connect to similar nodes (Kim and Altmann, 2017).

TABLE 3.3: Summary of network reconstruction for embedding dimension 10. We present AUROC and AP scores, mean average precision (mAP) and precision at k (p@k) for $k \in \{1, 3, 5, 10\}$ to 3 decimal places and rank to 1 decimal place. Rank is the average position that a true edge appears in the list of false edges ranked by distance. For mAP, we rank distances with respect to each node in the network and compute a separate precision score for each node, then report the mean of these precision. For p@k, we report the number of the k closest nodes that are true neighbours. All scores are averaged over 30 random starting seeds. Standard deviation is given in brackets.

	Cora_ML							
	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
N&K	209.8(5.7)	0.987(0.000)	0.986(0.000)	0.674(0.002)	0.791(0.005)	0.736(0.005)	0.723(0.004)	0.713(0.004)
AANE	3945.7(29.5)	0.758(0.002)	0.775(0.003)	0.145(0.001)	0.182(0.003)	0.192(0.003)	0.206(0.002)	0.238(0.002)
TADW	539.9(13.4)	0.967(0.001)	0.959(0.001)	0.401(0.002)	0.458(0.004)	0.425(0.004)	0.438(0.003)	0.444(0.005)
ATTRPURE	4768.0(33.3)	0.708(0.002)	0.735(0.003)	0.124(0.001)	0.156(0.004)	0.164(0.003)	0.174(0.003)	0.203(0.002)
DEEPWALK	185.6(7.6)	0.989(0.000)	0.986(0.001)	0.712(0.004)	0.757(0.006)	0.723(0.004)	0.722(0.005)	0.720(0.005)
SAGEGCN	913.9(62.8)	0.944(0.004)	0.935(0.005)	0.289(0.018)	0.304(0.022)	0.341(0.022)	0.363(0.022)	0.392(0.023)
$\text{HEAT}_{\alpha=0.0}$	**40.9(3.1)**	**0.998(0.000)**	**0.997(0.000)**	**0.853(0.003)**	**0.874(0.005)**	**0.869(0.004)**	**0.858(0.003)**	**0.850(0.003)**
$\text{HEAT}_{\alpha=0.2}$	58.6(3.5)	0.996(0.000)	0.995(0.000)	0.838(0.003)	0.895(0.005)	0.838(0.004)	0.823(0.004)	0.813(0.006)
$\text{HEAT}_{\alpha=1.0}$	3512.3(57.6)	0.785(0.004)	0.814(0.004)	0.137(0.003)	0.164(0.008)	0.178(0.007)	0.198(0.007)	0.226(0.008)
	Citeseer							
N&K	67.5(5.1)	0.994(0.000)	0.994(0.001)	0.799(0.003)	0.774(0.005)	0.772(0.006)	0.800(0.006)	0.837(0.005)
AANE	3741.9(26.4)	0.650(0.002)	0.645(0.003)	0.097(0.001)	0.088(0.002)	0.112(0.003)	0.132(0.004)	0.196(0.005)
TADW	297.1(11.6)	0.972(0.001)	0.964(0.002)	0.376(0.002)	0.314(0.004)	0.352(0.005)	0.399(0.005)	0.469(0.009)
ATTRPURE	3922.2(26.6)	0.633(0.002)	0.630(0.002)	0.089(0.001)	0.083(0.002)	0.102(0.002)	0.126(0.003)	0.189(0.004)
DEEPWALK	23.5(2.8)	0.998(0.000)	0.997(0.001)	0.798(0.004)	0.721(0.005)	0.805(0.006)	0.853(0.005)	0.899(0.007)
SAGEGCN	618.9(163.8)	0.942(0.015)	0.939(0.014)	0.216(0.025)	0.160(0.018)	0.291(0.036)	0.364(0.043)	0.470(0.050)
$\text{HEAT}_{\alpha=0.0}$	9.6(2.2)	0.999(0.000)	0.999(0.000)	0.830(0.003)	0.740(0.005)	**0.898(0.003)**	**0.932(0.005)**	**0.967(0.004)**
$\text{HEAT}_{\alpha=0.2}$	**7.6(1.8)**	**0.999(0.000)**	**0.999(0.000)**	**0.926(0.002)**	**0.899(0.004)**	0.894(0.005)	0.908(0.005)	0.933(0.005)
$\text{HEAT}_{\alpha=1.0}$	1567.9(24.1)	0.853(0.002)	0.865(0.002)	0.154(0.003)	0.122(0.004)	0.176(0.005)	0.227(0.005)	0.294(0.011)
	Pubmed							
N&K	451.5(13.0)	0.995(0.000)	0.994(0.000)	0.737(0.002)	0.743(0.003)	0.803(0.002)	0.835(0.002)	0.835(0.002)
AANE	19746.8(60.0)	0.777(0.001)	0.784(0.001)	0.104(0.000)	0.106(0.001)	0.187(0.001)	0.219(0.002)	0.253(0.001)
TADW	2155.6(41.1)	0.976(0.000)	0.973(0.001)	0.514(0.002)	0.502(0.003)	0.550(0.002)	0.600(0.002)	0.617(0.002)
ATTRPURE	25982.8(67.9)	0.707(0.001)	0.707(0.001)	0.097(0.000)	0.098(0.001)	0.175(0.001)	0.207(0.001)	0.241(0.001)
DEEPWALK	565.1(17.0)	0.994(0.000)	0.993(0.000)	0.816(0.002)	0.794(0.003)	0.808(0.002)	0.845(0.002)	0.856(0.002)
SAGEGCN	4118.9(312.0)	0.954(0.004)	0.945(0.004)	0.261(0.018)	0.215(0.016)	0.341(0.017)	0.405(0.019)	0.467(0.019)
$\text{HEAT}_{\alpha=0.0}$	**95.8(5.8)**	**0.999(0.000)**	0.998(0.000)	0.902(0.001)	0.875(0.002)	**0.916(0.002)**	**0.923(0.001)**	**0.923(0.001)**
$\text{HEAT}_{\alpha=0.2}$	166.4(7.3)	0.998(0.000)	**0.998(0.000)**	**0.903(0.001)**	**0.907(0.002)**	0.884(0.002)	0.901(0.002)	0.902(0.002)
$\text{HEAT}_{\alpha=1.0}$	12548.0(112.8)	0.858(0.001)	0.856(0.001)	0.096(0.001)	0.085(0.002)	0.161(0.003)	0.197(0.003)	0.242(0.003)
	PPI							
N&K	6757.7(43.5)	0.938(0.000)	0.940(0.000)	0.421(0.002)	**0.801(0.004)**	**0.699(0.003)**	0.664(0.003)	0.640(0.003)
AANE	50719.7(84.4)	0.537(0.001)	0.588(0.000)	0.089(0.000)	0.470(0.002)	0.251(0.001)	0.201(0.001)	0.168(0.001)
TADW	23408.8(95.7)	0.786(0.001)	0.767(0.001)	0.142(0.001)	0.502(0.002)	0.298(0.002)	0.258(0.002)	0.242(0.001)
ATTRPURE	51304.6(107.4)	0.511(0.001)	0.512(0.001)	0.038(0.000)	0.164(0.004)	0.084(0.002)	0.065(0.001)	0.051(0.001)
DEEPWALK	9962.9(85.3)	0.909(0.001)	0.903(0.001)	0.388(0.002)	0.721(0.005)	0.582(0.003)	0.543(0.003)	0.523(0.002)
SAGEGCN	44688.5(1161.1)	0.592(0.011)	0.607(0.010)	0.095(0.004)	0.455(0.005)	0.218(0.007)	0.169(0.007)	0.137(0.007)
$\text{HEAT}_{\alpha=0.0}$	**4926.0(42.9)**	**0.955(0.000)**	**0.952(0.000)**	**0.468(0.002)**	0.782(0.005)	0.696(0.004)	**0.664(0.003)**	**0.643(0.003)**
$\text{HEAT}_{\alpha=0.2}$	5628.5(52.6)	0.949(0.000)	0.946(0.001)	0.457(0.002)	0.786(0.006)	0.667(0.005)	0.627(0.005)	0.604(0.003)
$\text{HEAT}_{\alpha=1.0}$	47878.4(179.4)	0.563(0.002)	0.568(0.001)	0.068(0.000)	0.424(0.002)	0.183(0.001)	0.135(0.002)	0.108(0.002)
	MIT							
N&K	32506.7(128.6)	0.927(0.000)	0.930(0.000)	0.565(0.001)	**1.000(0.000)**	0.884(0.002)	0.855(0.002)	0.827(0.002)
AANE	157174.0(135.6)	0.647(0.000)	0.639(0.000)	0.189(0.000)	**1.000(0.000)**	0.528(0.002)	0.414(0.001)	0.321(0.001)
TADW	79981.9(134.4)	0.820(0.000)	0.814(0.000)	0.397(0.000)	**1.000(0.000)**	0.760(0.002)	0.700(0.001)	0.639(0.001)
ATTRPURE	171241.0(133.6)	0.615(0.000)	0.617(0.000)	0.184(0.000)	0.993(0.001)	0.543(0.002)	0.428(0.001)	0.323(0.001)
DEEPWALK	33037.6(168.4)	0.926(0.000)	0.919(0.000)	0.578(0.001)	**1.000(0.000)**	0.857(0.003)	0.819(0.002)	0.782(0.002)
SAGEGCN	89637.4(1885.4)	0.798(0.004)	0.797(0.003)	0.380(0.006)	**1.000(0.000)**	0.670(0.007)	0.599(0.009)	0.545(0.010)
$\text{HEAT}_{\alpha=0.0}$	**24776.7(97.4)**	**0.944(0.000)**	**0.940(0.000)**	**0.639(0.001)**	1.000(0.000)	**0.902(0.002)**	**0.877(0.002)**	**0.850(0.002)**
$\text{HEAT}_{\alpha=0.2}$	28621.0(126.8)	0.936(0.000)	0.934(0.000)	0.633(0.001)	**1.000(0.000)**	0.901(0.002)	0.870(0.002)	0.839(0.002)
$\text{HEAT}_{\alpha=1.0}$	162505.9(1248.6)	0.635(0.003)	0.650(0.002)	0.222(0.002)	**1.000(0.000)**	0.588(0.003)	0.490(0.004)	0.402(0.005)

TABLE 3.4: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the network reconstruction task, for an embedding dimension of 10. For each network, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML							
	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
$\text{HEAT}_{\alpha=0.2}$	**58.6(3.5)**	**0.996(0.000)**	**0.995(0.000)**	**0.838(0.003)**	**0.895(0.005)**	**0.838(0.004)**	**0.823(0.004)**	**0.813(0.006)**
N&K	209.8(5.7)	0.987(0.000)	0.986(0.000)	0.674(0.002)	0.791(0.005)	0.736(0.005)	0.723(0.004)	0.713(0.004)
t-statistic	1.24E+02	1.24E+02	8.65E+01	2.45E+02	8.03E+01	8.61E+01	1.01E+02	8.15E+01
p-value	2.40E-62	2.40E-62	1.55E-60	1.94E-86	2.83E-60	6.84E-60	5.05E-67	1.01E-54
$p < 0.05$	1	1	1	1	1	1	1	1
	Citeseer							
$\text{HEAT}_{\alpha=0.2}$	**7.6(1.8)**	**0.999(0.000)**	**0.999(0.000)**	**0.926(0.002)**	**0.899(0.004)**	**0.894(0.005)**	**0.908(0.005)**	**0.933(0.005)**
DEEPWALK	23.5(2.8)	0.998(0.000)	0.997(0.001)	0.798(0.004)	0.721(0.005)	0.805(0.006)	0.853(0.005)	0.899(0.007)
t-statistic	2.65E+01	2.65E+01	1.98E+01	1.76E+02	1.61E+02	6.59E+01	4.40E+01	2.18E+01
p-value	6.38E-31	6.38E-31	2.87E-23	9.26E-66	4.90E-74	5.42E-56	1.50E-46	3.03E-29
$p < 0.05$	1	1	1	1	1	1	1	1
	Pubmed							
$\text{HEAT}_{\alpha=0.2}$	**166.4(7.3)**	**0.998(0.000)**	**0.998(0.000)**	**0.903(0.001)**	**0.907(0.002)**	**0.884(0.002)**	**0.901(0.002)**	**0.902(0.002)**
N&K	451.5(13.0)	0.995(0.000)	0.994(0.000)	0.737(0.002)	0.743(0.003)	0.803(0.002)	0.835(0.002)	0.835(0.002)
t-statistic	1.05E+02	1.05E+02	7.51E+01	3.60E+02	2.46E+02	1.62E+02	1.44E+02	1.40E+02
p-value	1.50E-56	1.50E-56	1.98E-50	1.43E-71	1.00E-78	2.06E-76	8.45E-73	1.53E-73
$p < 0.05$	1	1	1	1	1	1	1	1
	PPI							
$\text{HEAT}_{\alpha=0.2}$	**5628.5(52.6)**	**0.949(0.000)**	**0.946(0.001)**	**0.457(0.002)**	0.786(0.006)	0.667(0.005)	0.627(0.005)	0.604(0.003)
N&K	6757.7(43.5)	0.938(0.000)	0.940(0.000)	0.421(0.002)	0.801(0.004)	0.699(0.003)	0.664(0.003)	0.640(0.003)
t-statistic	9.06E+01	9.06E+01	5.14E+01	6.81E+01	-1.12E+01	-3.02E+01	-3.73E+01	-4.70E+01
p-value	9.11E-63	9.11E-63	5.80E-49	1.80E-52	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	1	1	1	1	0	0	0	0
	MIT							
$\text{HEAT}_{\alpha=0.2}$	**28621.0(126.8)**	**0.936(0.000)**	**0.934(0.000)**	**0.633(0.001)**	1.000(0.000)	**0.901(0.002)**	**0.870(0.002)**	**0.839(0.002)**
N&K	32506.7(128.6)	0.927(0.000)	0.930(0.000)	0.565(0.001)	1.000(0.000)	0.884(0.002)	0.855(0.002)	0.827(0.002)
t-statistic	1.18E+02	1.18E+02	6.44E+01	2.34E+02	N/A	3.56E+01	2.95E+01	2.43E+01
p-value	4.80E-71	4.80E-71	1.28E-55	3.63E-88	N/A	2.54E-41	7.48E-36	2.28E-32
$p < 0.05$	1	1	1	1	0	1	1	1

multiple labels from the Gene Ontology (Hamilton, Ying, and Leskovec, 2017). To evaluate our model in this multi-label case, we adopt a one-vs-all setting, where we train a separate classifier for each class. Table 3.6 proves a summary of the node classification results for embedding dimension 5. T-test statistics of a one-sided t-test between $\text{HEAT}_{\alpha=0.2}$ and the best ranked benchmark algorithm according to F1 are provided in table 3.8. While HEAT never ranked first for any network, it consistently ranked highly on all networks unlike all other benchmark algorithms – with $\text{HEAT}_{\alpha=0.2}$ the best overall rank averaged across all the datasets. Further, we observe that when considering node attributes, i.e., $\alpha = 0.2$, HEAT outperformed the other hyperbolic embedding algorithm N&K on all networks. We also note that, even without node attributes, HEAT obtained better results than N&K on all networks. Comparing with the Euclidean benchmark algorithms, we observe competitive results – especially when incorporating attributes.

Figure 3.3 plots AUROC scores achieved by the SVC model after training on 2%

TABLE 3.5: Summary of link prediction for embedding dimension 10. We present AUROC, AP, and mean average precision (mAP) to 3 decimal places and rank to 1 decimal place. Rank is the average position that a true edge appears in the list of false edges ranked by distance. For mAP, we rank distances with respect to each node in the network and compute a separate precision score for each node, then report the mean. All scores are averaged over 30 random starting seeds. Standard deviation is given in brackets. Mean Ranks is the average position of an algorithm in a ranked list of performance.

	Cora_ML				PPI			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	96.6(7.9)	0.922(0.006)	0.935(0.005)	0.248(0.008)	722.1(18.2)	0.912(0.002)	0.918(0.002)	0.185(0.004)
AANE	315.3(14.5)	0.743(0.012)	0.761(0.012)	0.098(0.007)	3822.2(38.1)	0.535(0.005)	0.582(0.005)	0.070(0.004)
TADW	72.8(4.9)	0.941(0.004)	0.933(0.005)	0.180(0.008)	2237.5(55.8)	0.728(0.007)	0.722(0.006)	0.080(0.004)
ATTRPURE	356.5(12.4)	0.710(0.010)	0.736(0.011)	0.093(0.005)	3854.5(41.2)	0.511(0.005)	0.511(0.004)	0.018(0.002)
DEEPWALK	178.9(10.7)	0.855(0.009)	0.896(0.006)	0.224(0.010)	1066.4(23.8)	0.870(0.003)	0.873(0.003)	0.144(0.005)
SAGEGCN	160.5(14.9)	0.870(0.012)	0.873(0.010)	0.114(0.008)	3622.2(68.4)	0.560(0.008)	0.574(0.007)	0.067(0.004)
$\text{HEAT}_{\alpha=0.00}$	131.8(23.9)	0.893(0.020)	0.928(0.012)	0.276(0.069)	**431.6(91.9)**	**0.948(0.011)**	**0.947(0.009)**	**0.255(0.022)**
$\text{HEAT}_{\alpha=0.20}$	**49.0(4.3)**	**0.961(0.004)**	**0.963(0.004)**	**0.288(0.010)**	493.0(98.8)	0.940(0.012)	0.940(0.010)	0.241(0.022)
$\text{HEAT}_{\alpha=1.00}$	267.5(11.1)	0.782(0.009)	0.811(0.010)	0.096(0.008)	3646.2(40.0)	0.557(0.005)	0.563(0.004)	0.064(0.004)
	Citeseer				MIT			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	145.5(8.9)	0.820(0.011)	0.853(0.008)	0.204(0.012)	2727.1(35.3)	0.918(0.001)	0.922(0.001)	0.256(0.003)
AANE	284.8(12.7)	0.646(0.016)	0.643(0.016)	0.086(0.007)	11816.8(77.5)	0.646(0.002)	0.638(0.003)	0.098(0.003)
TADW	55.9(5.2)	0.931(0.006)	0.917(0.008)	0.149(0.011)	6133.4(38.8)	0.816(0.001)	0.815(0.001)	0.180(0.003)
ATTRPURE	297.1(12.8)	0.630(0.016)	0.630(0.016)	0.083(0.006)	12823.6(64.3)	0.616(0.002)	0.617(0.002)	0.098(0.003)
DEEPWALK	259.9(9.1)	0.677(0.011)	0.775(0.009)	0.213(0.014)	2834.4(44.7)	0.915(0.001)	0.909(0.002)	0.241(0.003)
SAGEGCN	138.1(13.2)	0.829(0.016)	0.848(0.015)	0.132(0.012)	7142.0(117.9)	0.786(0.004)	0.784(0.003)	0.153(0.004)
$\text{HEAT}_{\alpha=0.00}$	16.5(2.5)	0.981(0.003)	0.979(0.005)	0.791(0.014)	**2246.9(28.9)**	**0.933(0.001)**	**0.930(0.001)**	**0.279(0.003)**
$\text{HEAT}_{\alpha=0.20}$	**6.3(1.3)**	**0.993(0.002)**	**0.994(0.001)**	**0.810(0.013)**	2510.4(38.6)	0.925(0.001)	0.925(0.001)	0.273(0.004)
$\text{HEAT}_{\alpha=1.00}$	120.4(6.4)	0.851(0.008)	0.864(0.009)	0.144(0.011)	12442.3(114.6)	0.627(0.003)	0.644(0.003)	0.112(0.003)
	Pubmed				Mean Ranks			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	801.7(20.4)	0.880(0.003)	0.900(0.003)	0.210(0.004)	3.8	3.8	3.4	3.2
AANE	1503.8(33.5)	0.774(0.005)	0.782(0.005)	0.088(0.002)	7.4	7.4	7.6	7.2
TADW	465.5(18.7)	0.930(0.003)	0.923(0.004)	0.157(0.005)	3.4	3.4	3.6	5
ATTRPURE	1954.5(29.4)	0.706(0.004)	0.706(0.005)	0.082(0.003)	9	9	9	8.8
DEEPWALK	1742.8(28.6)	0.738(0.004)	0.833(0.003)	0.203(0.005)	5.8	5.8	5.4	3.8
SAGEGCN	635.5(41.1)	0.905(0.006)	0.901(0.005)	0.114(0.006)	5	5	5.6	6.4
$\text{HEAT}_{\alpha=0.00}$	1531.0(30.1)	0.770(0.005)	0.858(0.003)	0.245(0.005)	3	3	2.6	1.6
$\text{HEAT}_{\alpha=0.20}$	**274.7(9.9)**	**0.959(0.001)**	**0.960(0.002)**	**0.284(0.004)**	**1.4**	**1.4**	**1.4**	**1.4**
$\text{HEAT}_{\alpha=1.00}$	951.7(25.2)	0.857(0.004)	0.856(0.004)	0.077(0.002)	6.2	6.2	6.4	7.6

to 10% training example from each class. From this, we see that HEAT is capable of achieving highly competitive class separation – often performing on par with the state-of-the-art TADW – even when only a very small percentage of node labels are known. Furthermore, we see that HEAT is largely robust to the setting of training sample proportion.

3.3.6 Parameter Sensitivity

Here we evaluate HEAT's robustness to the setting of the control parameters. Our results indicated that the most sensitive parameter is α, which controls the trade off between topology and attributes. We therefore run HEAT over a range of values $\alpha \in [0, 1]$ in steps of 0.05. Figure 3.4 plots various metric scores for network re-

TABLE 3.6: Summary of node classification results for embedding dimension 5. Bold indicates best performance. Standard deviation is given in brackets. Mean Ranks is the average position of an algorithm in a ranked list of performance.

	Cora_ML				PPI			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.732(0.016)	0.806(0.015)	0.671(0.018)	0.943(0.004)	0.388(0.001)	**0.695(0.002)**	0.269(0.002)	0.704(0.000)
AANE	0.617(0.001)	0.786(0.001)	0.509(0.001)	0.918(0.000)	**0.398(0.002)**	0.686(0.002)	**0.280(0.002)**	0.704(0.000)
TADW	0.764(0.014)	0.823(0.014)	0.713(0.015)	0.965(0.004)	0.393(0.001)	0.689(0.001)	0.275(0.001)	0.704(0.000)
ATTRPURE	0.619(0.001)	0.790(0.001)	0.509(0.001)	0.916(0.000)	0.397(0.001)	0.689(0.002)	0.279(0.002)	**0.705(0.000)**
DEEPWALK	0.833(0.004)	0.866(0.004)	**0.803(0.005)**	0.971(0.001)	0.388(0.001)	0.693(0.002)	0.269(0.002)	0.704(0.000)
SAGEGCN	0.578(0.065)	0.696(0.035)	0.499(0.079)	0.903(0.014)	0.388(0.002)	0.693(0.002)	0.270(0.002)	0.704(0.000)
$HEAT_{\alpha=0.00}$	0.790(0.008)	0.848(0.008)	0.740(0.009)	0.960(0.002)	0.389(0.002)	0.694(0.002)	0.270(0.002)	0.704(0.000)
$HEAT_{\alpha=0.20}$	**0.834(0.006)**	**0.874(0.005)**	0.798(0.007)	**0.976(0.001)**	0.389(0.002)	0.693(0.002)	0.271(0.002)	0.704(0.000)
$HEAT_{\alpha=1.00}$	0.538(0.025)	0.700(0.016)	0.438(0.028)	0.889(0.006)	0.390(0.001)	0.691(0.002)	0.272(0.002)	0.703(0.000)
	Citeseer				MIT			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.468(0.037)	0.839(0.025)	0.326(0.036)	0.844(0.007)	0.501(0.019)	0.859(0.017)	0.354(0.018)	0.932(0.005)
AANE	0.523(0.000)	0.836(0.001)	0.381(0.000)	0.861(0.000)	0.022(0.002)	0.482(0.039)	0.011(0.001)	0.863(0.000)
TADW	0.845(0.002)	0.874(0.003)	0.819(0.002)	0.969(0.001)	**0.560(0.002)**	0.723(0.003)	**0.457(0.003)**	0.948(0.000)
ATTRPURE	0.564(0.001)	0.813(0.002)	0.432(0.001)	0.882(0.000)	0.023(0.004)	0.422(0.037)	0.012(0.002)	0.866(0.000)
DEEPWALK	0.858(0.006)	0.881(0.005)	0.836(0.008)	0.973(0.002)	0.537(0.005)	0.831(0.006)	0.397(0.005)	0.949(0.001)
SAGEGCN	0.359(0.074)	0.687(0.046)	0.248(0.069)	0.802(0.023)	0.424(0.022)	0.815(0.017)	0.287(0.018)	0.916(0.006)
$HEAT_{\alpha=0.00}$	0.561(0.027)	0.837(0.017)	0.423(0.031)	0.873(0.004)	0.559(0.009)	**0.860(0.007)**	0.415(0.010)	**0.952(0.001)**
$HEAT_{\alpha=0.20}$	**0.873(0.005)**	**0.905(0.005)**	**0.844(0.007)**	**0.977(0.001)**	0.538(0.010)	0.817(0.008)	0.401(0.012)	0.951(0.001)
$HEAT_{\alpha=1.00}$	0.727(0.003)	0.810(0.005)	0.659(0.004)	0.929(0.001)	0.008(0.006)	0.438(0.156)	0.004(0.003)	0.835(0.004)
	Pubmed				Mean Ranks			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.744(0.008)	0.769(0.007)	0.720(0.011)	0.893(0.005)	6.6	3.4	6.6	6
AANE	0.735(0.000)	0.755(0.000)	0.715(0.000)	0.899(0.000)	6.2	7.4	6	5.8
TADW	0.788(0.001)	0.807(0.001)	0.769(0.001)	0.921(0.001)	3	4.8	3	3.2
ATTRPURE	0.700(0.000)	0.721(0.000)	0.679(0.000)	0.879(0.000)	5.8	7.8	5.8	5.8
DEEPWALK	0.804(0.003)	0.813(0.002)	0.795(0.003)	0.923(0.001)	3.6	2.4	3.4	2.8
SAGEGCN	0.745(0.029)	0.760(0.024)	0.730(0.034)	0.891(0.014)	7	6.8	7	7.6
$HEAT_{\alpha=0.00}$	0.793(0.003)	0.812(0.003)	0.774(0.003)	0.924(0.002)	4	2.8	4	3.8
$HEAT_{\alpha=0.20}$	**0.819(0.003)**	**0.833(0.003)**	**0.804(0.003)**	**0.942(0.001)**	**2.2**	**2.2**	**2.4**	**2.6**
$HEAT_{\alpha=1.00}$	0.735(0.015)	0.757(0.013)	0.715(0.016)	0.895(0.007)	6.6	7.4	6.8	7.4

construction, link prediction, and node classification obtained from a 5 dimensional embedding. From these plots, we can see the performance of HEAT on the three tasks on all of the networks is robust to a wide range of values of α. In particular, we find that performance is most consistent for $\alpha \in [0, 0.5]$.

We further investigate the robustness of HEAT to the setting of embedding dimension on all three downstream tasks. Figure 3.5 plots AUROC scores for embedding dimensions 5, 10, 25 and 50. From this, we see that the performance of HEAT is constant over a range of settings of dimension, especially in the case of node classification – as indicated by the largely flat curves.

TABLE 3.7: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the link prediction task, for embedding dimension 10. For each network, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold.

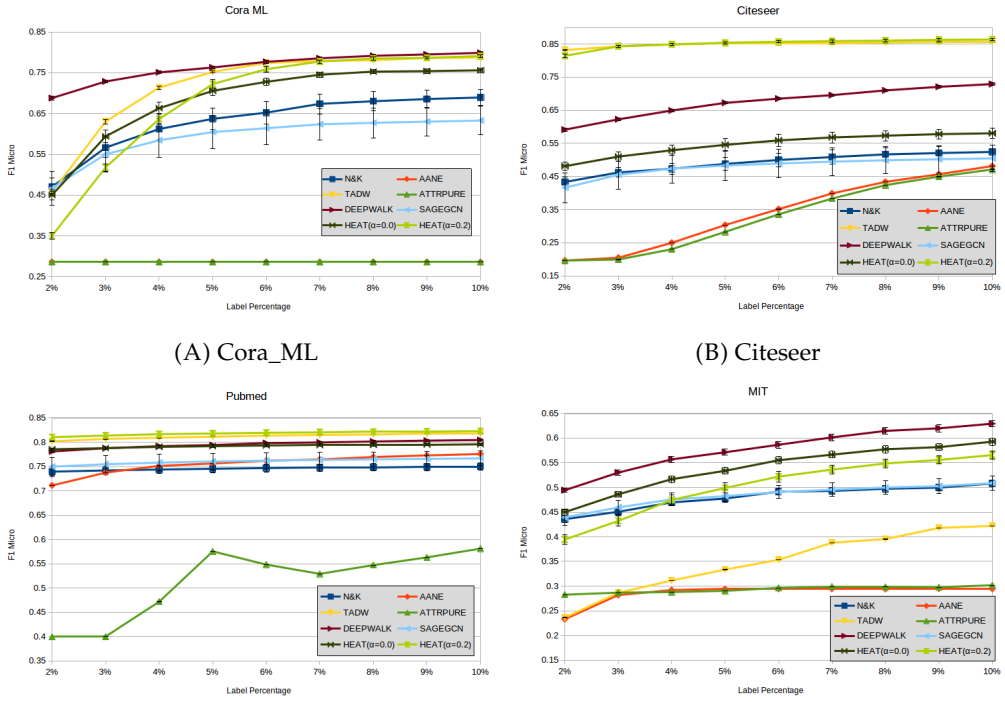
	Cora_ML			
	Mean Rank	AUROC	AP	mAP
$\text{HEAT}_{\alpha=0.20}$	**49.0(4.3)**	**0.961(0.004)**	**0.963(0.004)**	**0.288(0.010)**
N&K	96.6(7.9)	0.922(0.006)	0.935(0.005)	0.248(0.008)
t-statistic	2.91E+01	2.91E+01	2.41E+01	1.72E+01
p-value	5.31E-31	5.31E-31	2.94E-30	1.67E-24
$p < 0.05$	1	1	1	1
	Citeseer			
$\text{HEAT}_{\alpha=0.20}$	**6.3(1.3)**	**0.993(0.002)**	**0.994(0.001)**	**0.810(0.013)**
TADW	55.9(5.2)	0.931(0.006)	0.917(0.008)	0.149(0.011)
t-statistic	5.09E+01	5.09E+01	5.40E+01	2.11E+02
p-value	7.51E-33	7.50E-33	2.25E-32	1.31E-82
$p < 0.05$	1	1	1	1
	Pubmed			
$\text{HEAT}_{\alpha=0.20}$	**274.7(9.9)**	**0.959(0.001)**	**0.960(0.002)**	**0.284(0.004)**
TADW	465.5(18.7)	0.930(0.003)	0.923(0.004)	0.157(0.005)
t-statistic	4.94E+01	4.94E+01	5.13E+01	1.08E+02
p-value	1.53E-40	1.53E-40	2.23E-38	2.19E-68
$p < 0.05$	1	1	1	1
	PPI			
$\text{HEAT}_{\alpha=0.20}$	**493.0(98.8)**	**0.940(0.012)**	**0.940(0.010)**	**0.241(0.022)**
N&K	722.1(18.2)	0.912(0.002)	0.918(0.002)	0.185(0.004)
t-statistic	1.25E+01	1.25E+01	1.14E+01	1.36E+01
p-value	6.29E-14	6.29E-14	5.15E-13	6.73E-15
$p < 0.05$	1	1	1	1
	MIT			
$\text{HEAT}_{\alpha=0.20}$	**2510.4(38.6)**	**0.925(0.001)**	**0.925(0.001)**	**0.273(0.004)**
N&K	2727.1(35.3)	0.918(0.001)	0.922(0.001)	0.256(0.003)
t-statistic	2.27E+01	2.27E+01	9.45E+00	1.92E+01
p-value	1.12E-30	1.12E-30	1.58E-13	6.33E-27
$p < 0.05$	1	1	1	1

TABLE 3.8: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the node classification task, for embedding dimension 5. For each network, we select the benchmark algorithm according to F1. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.834(0.006)	**0.874(0.005)**	0.798(0.007)	**0.976(0.001)**
DEEPWALK	0.833(0.004)	0.866(0.004)	0.803(0.005)	0.971(0.001)
t-statistic	8.75E-01	6.17E+00	-2.91E+00	1.85E+01
p-value	1.93E-01	4.08E-08	9.97E-01	5.94E-24
$p < 0.05$	0	1	0	1
	Citeseer			
$\text{HEAT}_{\alpha=0.20}$	**0.873(0.005)**	**0.905(0.005)**	**0.844(0.007)**	**0.977(0.001)**
DEEPWALK	0.858(0.006)	0.881(0.005)	0.836(0.008)	0.973(0.002)
t-statistic	1.08E+01	1.78E+01	4.36E+00	1.20E+01
p-value	1.68E-15	2.29E-25	2.80E-05	1.20E-15
$p < 0.05$	1	1	1	1
	Pubmed			
$\text{HEAT}_{\alpha=0.20}$	**0.819(0.003)**	**0.833(0.003)**	**0.804(0.003)**	**0.942(0.001)**
DEEPWALK	0.804(0.003)	0.813(0.002)	0.795(0.003)	0.923(0.001)
t-statistic	2.07E+01	3.04E+01	1.20E+01	7.15E+01
p-value	2.88E-28	1.28E-35	1.37E-17	1.03E-51
$p < 0.05$	1	1	1	1
	PPI			
$\text{HEAT}_{\alpha=0.20}$	0.389(0.002)	**0.693(0.002)**	0.271(0.002)	0.704(0.000)
AANE	0.398(0.002)	0.686(0.002)	0.280(0.002)	0.704(0.000)
t-statistic	-1.67E+01	1.27E+01	-1.64E+01	-6.55E+00
p-value	1.00E+00	1.24E-18	1.00E+00	1.00E+00
$p < 0.05$	0	1	0	0
	MIT			
$\text{HEAT}_{\alpha=0.20}$	0.538(0.010)	**0.817(0.008)**	0.401(0.012)	**0.951(0.001)**
TADW	0.560(0.002)	0.723(0.003)	0.457(0.003)	0.948(0.000)
t-statistic	-1.16E+01	5.80E+01	-2.46E+01	1.09E+01
p-value	1.00E+00	6.61E-36	1.00E+00	2.76E-12
$p < 0.05$	0	1	0	1

3.4 Chapter Summary

This chapter presents HEAT to fill the gap of embedding attributed networks in hyperbolic space. We have designed a random walk algorithm to obtain the training samples that capture both network topological and attribute similarity. We have also derived an algorithm that learns hyperboloid embeddings from the training samples. Our results on three networks show that, by including attributes, HEAT can improve the quality of a learned hyperbolic embedding in a number of downstream machine learning tasks. We find that, in general, HEAT does not perform as well in the node classification task compared with the Euclidean benchmark algorithms (table 3.6) than in the network reconstruction and link prediction tasks (table 3.3): while it is the most consistent across all datasets, it does not rank first on any particular dataset as it does for reconstruction and link prediction. This could be attributed to the SVC learning sub-optimal decision boundaries, since it is using



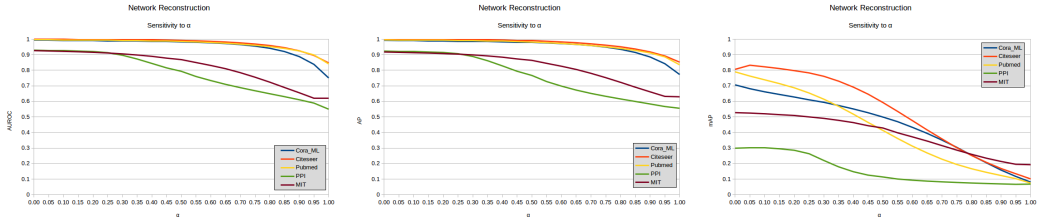
(A) Cora_ML (B) Citeseer

(C) Pubmed (D) MIT

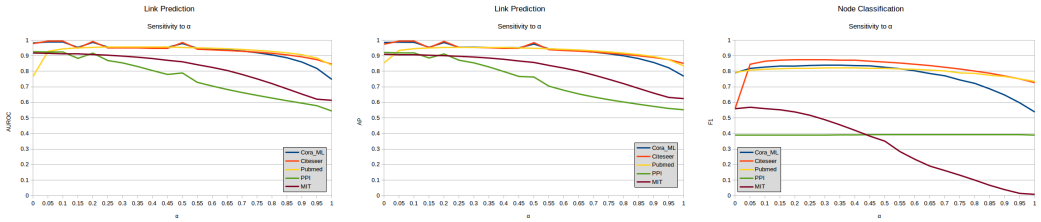
FIGURE 3.3: Plots of micro-averaged F1 scores against labelled percentage for the (A) Cora_ML, (B) Citeseer, (C) Pubmed, and (D) MIT networks. Each score is averaged over 30 random starting seeds. $\text{HEAT}_{\alpha=0.2}$ considers node attributes, while $\text{HEAT}_{\alpha=0.0}$ indicates the embedding only considers network topology. We present only the results for an embedding dimension of 10 for clarity, but obtain similar results for all dimensions.

a Euclidean optimisation procedure. Many neural network operations, in particular: logistic regression, have been generalized to the Poincaré ball (Ganea, Bécigneul, and Hofmann, 2018), and this may provide superior results. However, we leave this as future work and this is discussed in detail in section 6.2.4.

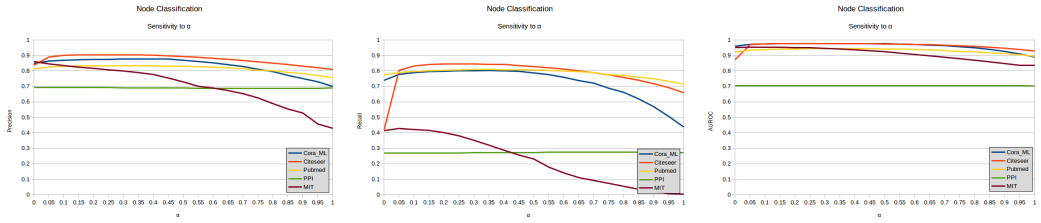
HEAT provides a general hyperbolic embedding method for both unattributed and attributed networks, which opens the door to hyperbolic manifold learning on a wide ranges of networks.



(A) Network reconstruction: AUROC (B) Network reconstruction: AP (C) Network reconstruction: mAP

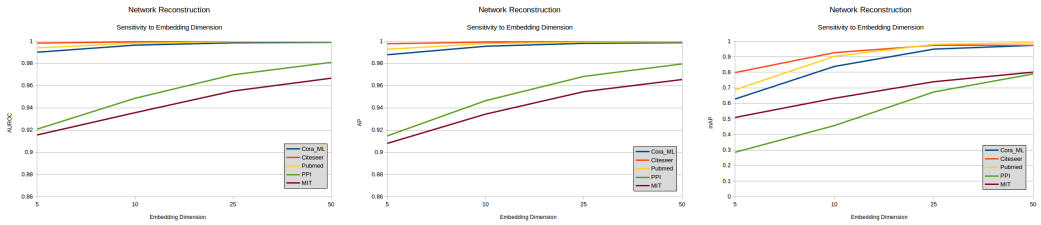


(D) Link prediction: AUROC (E) Link prediction: AP (F) Node classification: F1



(G) Node classification: Precision (H) Node classification: Recall (I) Node classification: AUROC

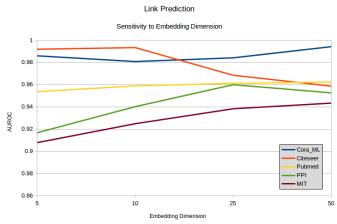
FIGURE 3.4: Plots to show the effect of α on downstream machine learning tasks. We fix the embedding dimension to 5 for these experiments.



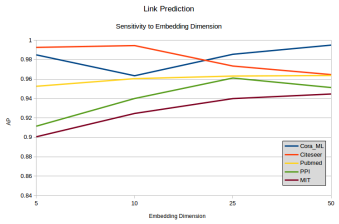
(A) Network reconstruction: AUROC

(B) Network reconstruction: AP

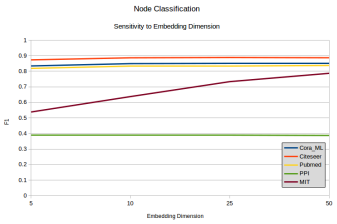
(C) Network reconstruction: mAP



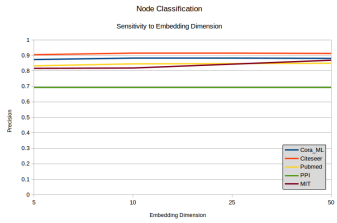
(D) Link prediction: AUROC



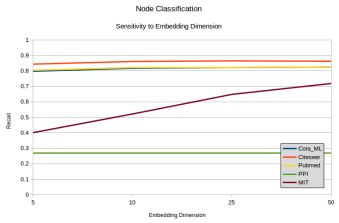
(E) Link prediction: AP



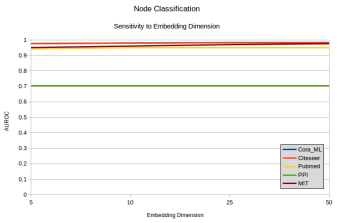
(F) Node classification: F1



(G) Node classification: Precision



(H) Node classification: Recall



(I) Node classification: AUROC

FIGURE 3.5: Plots to show the effect of embedding dimension on downstream machine learning tasks. We fix $\alpha = 0.2$ for these experiments.

Chapter 4

Hyperbolic Embedding of Attributed and Directed Networks

This chapter aims to answer our second research question (given in section 1.8.2) and focuses on hyperbolic attributed directed network embedding. Literature review revealed that directed networks comprise a large subset of all complex networks. Most directed network embedding algorithms are limited to learning two representations for each node in the network; few are able to handle attributes and even fewer embed to hyperbolic space. This is the gap that this chapter aims to fill. Section 4.1 provides the rationale behind our approach. Section 4.2 introduces our algorithm. Section 4.3 validates our approach on a number of benchmark directed network datasets on common downstream machine learning tasks. Finally, section 4.4 summarises the chapter.

4.1 Introduction

Complex networks are models of diverse real-world systems (Papadopoulos et al., 2012). The ubiquity of complex networks in such fields as the study of protein-protein interactions, the world wide web and social interaction modelling, has lead to the study of complex networks emerging as a popular research topic. They are described as a set of entities (nodes) and the relations between them (edges). In many complex networks, edges are *directed* and this is essential information that characterises the network as a whole. In addition to the structural information described by edges, often nodes within a complex network are richly annotated with *attributes*.

These attributes provide additional information about the elements within the network and this information is necessary to understand the role that these elements play within the system (Hamilton, Ying, and Leskovec, 2017).

However, when dealing with large networks containing billions of nodes, traditional network-based approaches suffer some drawbacks, such as computational complexity, parallelizability and applicability to downstream machine learning tasks (Cui et al., 2018). To overcome these drawbacks, *network embedding* is often used to learn low dimensional representations of the nodes in a large network. By learning dense representations of entities in the network, network embedding is able to de-noise the network while preserving the intrinsic structural information (Cui et al., 2018), further improving performance on downstream tasks over using the raw network representation.

A recent trend in network embedding research is to suppose that the hidden metric space underpinning many complex networks is, in fact, hyperbolic (Papadopoulos et al., 2010). A hyperbolic metric space has been shown to explain the scale-free degree distribution observed in real-world networks (Krioukov et al., 2009). Moreover, it can explain the *small-world* effect observed in complex networks (Bianconi and Rahmede, 2017), help with routing of information packets around the network (Papadopoulos et al., 2010) and explains the implicit trade-off between popularity and similarity that controls a node's connections (Papadopoulos et al., 2012).

However, there is no approach that can embed attributed directed networks into hyperbolic space. This is a significant gap because attributed, directed networks are the most general form of complex networks modelling real world systems. To fill this gap, we propose HEADNet, an algorithm that can learn hyperbolic representations of arbitrary low dimensions of the nodes in an attributed and directed network. Moreover, we aim to capture the uncertainty in the learned embeddings using Gaussian distribution, similar to Bojchevski and Günnemann, 2018, but in hyperbolic space. This allows us to characterise node neighbourhood diversity and use an established asymmetric similarity measure with interpretations in both probability and information theory.

To this end, in this chapter we propose:

1. an embedding model to map attributed nodes to Gaussian distributions in hyperbolic space;

2. the use of a node similarity measure in hyperbolic space based on a novel mapping procedure to map hyperbolic Gaussian distributions to Euclidean ones, preserving hyperbolic distance and direction.

HEADNet not only learns low-dimensional representations, but also preserves

1. the network's latent structural hierarchy;

2. the interplay of attributes and structural information;

3. the direction of relations between nodes in a network

for downstream tasks – such as link prediction – allowing for greater insight into complex systems than ever before.

4.2 Hyperbolic Embedding of Attributed and Directed Networks

4.2.1 Intuition

HEADNet is built upon the principle that nodes in an attributed directed network can be mapped to Gaussian distributions in hyperbolic space. The intuition behind this approach is that the distributions capture the uncertainty in the learned representations of nodes. Uncertainty here means that nodes with highly diverse neighbourhoods – in terms of attribute presence or class label, for example – tend to be embedded as distributions with greater variance (Bojchevski and Günnemann, 2018). Hyperbolic space is used to reflect the inherent hierarchy of the elements within many real-world systems. Kullback-Leibler divergence, which is a measure of the penalty of encoding one distribution as another, is used as an asymmetric measure of similarity between nodes. This forms the basis of an objective function that aims to maximize the similarity between true node pairs (the directed edges in the network), while minimising the similarity between all other node pairs.

4.2.2 Problem Definition

We reformulate problem 1 for hyperbolic attributed directed network as follows:

We consider a network of N nodes given by the set V with $|V| = N$. We use E to denote the set of all interactions between the nodes in our network. $E = \{(u, v)\} \subseteq V \times V$. We consider the case that the network is *directed*. That is $(u, v) \in E \implies (v, u) \in E$. Further, the nodes in V may be annotated with d-dimensional attributes described by matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$. In the special case that nodes do not have attributes, we set $\mathbf{X} = \mathbf{I}_N$ which is the N-dimensional identity matrix.

Problem 3. *Given a directed network, described by $\mathbb{G} = (V, E, \boldsymbol{X})$, find low dimensional representations of the nodes in the network in hyperbolic space such that the structural information in network is preserved. The described problem is unsupervised.*

4.2.3 HEADNet Overview

To address problem 3, we propose HEADNet, which maps nodes in an attributed and directed network to n-dimensional Gaussian distributions in hyperbolic space. The distributions are described by $n + 1$-dimensional hyperboloid mean vectors, along with n-dimensional Euclidean vectors corresponding to diagonal covariance matrices.

HEADNet is comprised of three main steps.

1. The network embedding step. This step addresses the problem described in problem 3 by using a novel node embedder to transform the nodes in $\mathbb{G}$ into a set of N n-dimensional Gaussian distributions in hyperbolic space. Edges in E are not required to perform an embedding, however they are used to train the embedder (see step 3). It is this property that allows HEADNet to embed unseen nodes after training.

2. The node similarity measurement step. For a pair of nodes, this step takes their hyperbolic distributions provided by step 1 as input and provides a asymmetric measure of similarity between then.

3. Node representation learning step. This step uses the output of step 2 as part of a learning objective explicitly based on known connectivity information to

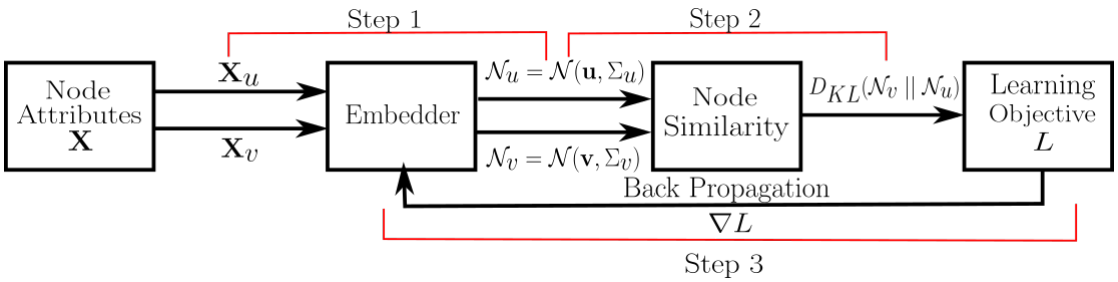


FIGURE 4.1: Schematic representation of the main steps of the learning procedure for HEAD-Net. The first step is the embedder which accepts node attributes and outputs Gaussian distributions in hyperbolic space. Next is the node similarity step that takes two hyperbolic Gaussian distributions and computes node similarity using Kullback-Leibler divergence. Finally, the node similarity is passed to an objective function that aims to maximize the similarity of true node pairs. The objective function is used to drive the learning of embedder parameters.

train the embedder model used in step 1. The learning objective is designed to maximize the similarity between all of the node pairs $(u, v) \in E$ and minimize the similarity of all other pairs of nodes in $V \times V \setminus E$.

Figure 4.1 provides an overview of these three steps.

Once the embedder has been trained using step 3, it can then be used by itself to map nodes to distributions in hyperbolic space (using step 1 only), or it can be used in conjunction with the node similarity step to query the similarly between any arbitrary pair of nodes (combining steps 1 and 2). For both applications, the nodes do not necessarily have to be previously seen during model training.

4.2.4 Network Embedding Step

This section describes how node attributes $\mathbf{X}_u$ are mapped to a pair of parameters $(\mathbf{u}, \Sigma_u) \in \mathbb{H}^n \times \mathbb{R}^n$ describing a Gaussian distribution in hyperbolic space. $\mathbf{u}$ is an $n+1$-dimensional vector on the hyperboloid model representing the mean of the distribution describing node u. Σ_u is an n-dimensional Euclidean vector that represents the diagonal covariance matrix of the distribution for node u.

We compute $\mathbf{u}$ and Σ_u as functions of node attributes $\mathbf{X}_u$. Mapping directly from attributes has the advantage of readily handling previously unseen nodes (Bojchevski and Günnemann, 2018). To this end we propose a novel embedder model. The embedder is a two-layer feed-forward neural network with two parallel output layers. We select this architecture as it provides a flexible non-linear mapping between attributes and distributions, while reducing the overall number of parameters

compared to two separate models for computing means and variances respectively. The embedder has three components:

1. a map from attributes $\mathbf{X}_u$ to $\mathbf{h}_u$, a $d' \ll d$ dimensional hidden layer representation;

2. a map from the hidden representation $\mathbf{h}_u$ to Euclidean vectors representing diagonal covariance matrices $\Sigma_u \in \mathbb{R}^n$;

3. a map from the hidden representation $\mathbf{h}_u$ to hyperbolic means $\mathbf{u} \in \mathbb{H}^n$.

See figure 4.3 for an overview of the model architecture.

Mapping to Hidden Representation $\mathbf{h}_u$

We first map non-linearly[1] from node attributes to a latent hidden representation $\mathbf{h}_u \in \mathbb{R}^{d'}$ ($d' \ll N$) with:

$$\mathbf{h}_u = \text{relu}\left(\mathbf{W}_h \mathbf{X}_u + \mathbf{b}_h\right) \tag{4.1}$$

where $\mathbf{W}_h \in \mathbb{R}^{d' \times d}$ and $\mathbf{b}_h \in \mathbb{R}^{d'}$ are a weight matrix and bias to learn. We share this intermediary representation $\mathbf{h}_u$ for computing both $\mathbf{u}$ and Σ_u, which has the advantage of regularising the model by reducing the overall number of parameters (Bojchevski and Günnemann, 2018). We select the relu function as a non-linearity to overcome the vanishing gradients problem. Furthermore, performance of relu was superior to tanh and sigmoid in some small preliminary benchmark tasks.

Mapping from Hidden Representation $\mathbf{h}_u$ to Σ_u

From the hidden representation of node u, $\mathbf{h}_u$, we compute its covariance vector Σ_u using:

$$\Sigma_u = \text{elu}_{\alpha=1}\left(\mathbf{W}_\Sigma \mathbf{h}_u + \mathbf{b}_\Sigma\right) + 1 \tag{4.2}$$

[1] Recall $\text{relu}(x) := \max(x, 0)$.

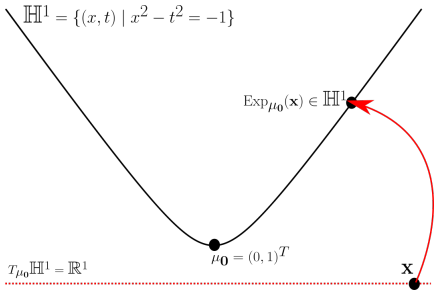


FIGURE 4.2: Example of Exp_{μ_0} in operation. μ_0 is highlighted at the base of the hyperboloid. $\mathbb{R}^1$, the tangent space $\mu_0$is highlighted in red with a point $\mathbf{x} \in \mathbb{R}^1$ in black. The red arrow shows how Exp_{μ_0} takes $\mathbf{x}$ to a unique point on the hyperboloid. This new point has the property that its hyperbolic distance from μ_0 is equal to $||\mathbf{x}||$.

We use the elu function[2] to ensure the required positive definiteness in Σ (Bojchevski and Günnemann, 2018).

Mapping from Hidden Representation $\mathbf{h}_u$ to $\mathbf{u}$

To map from hidden representation $\mathbf{h}_u \in \mathbb{R}^{d'}$ to hyperbolic means $\mathbf{u} \in \mathbb{H}^n$, we apply some fundamental operations of Riemannian geometry to map from Euclidean space to hyperbolic space. Specifically, we use the combination of the exponential map for the hyperboloid and a property about the tangent space of the "bottom tip" of the hyperboloid.

Often it is useful to transport vectors to and from the hyperboloid and the tangent space of a point on the hyperboloid. In order to achieve this, we apply the hyperboloid exponential map operation Exp. (see equation (1.32).

In equation (1.30), we defined μ_0, the bottom-tip of the hyperboloid. Furthermore, we noted that the tangent space of μ_0 is the entire n-dimensional Euclidean plane (see equation (1.31) for details). We now combine both Exp. and the property that $T_{\mu_0}\mathbb{H}^n = \mathbb{R}^n$ to define a point-wise non-linearity for the output of a dense neural network layer. $\text{Exp}_{\mu_0} : \mathbb{R}^n \to \mathbb{H}^n$ is computed as:

$$\text{Exp}_{\mu_0}(\mathbf{x}) = \left[\sinh\left(||\mathbf{x}||\right) \frac{\mathbf{x}}{||\mathbf{x}||}, \cosh\left(||\mathbf{x}||\right)\right] \tag{4.3}$$

for $\mathbf{x} \in \mathbb{R}^n$, where $||\cdot||$ is the Euclidean norm operation.

[2] Recall $\text{elu}_\alpha := \begin{cases} x, \text{if } x > 0 \\ \alpha\left(\exp(x) + 1\right) \text{otherwise.} \end{cases}$

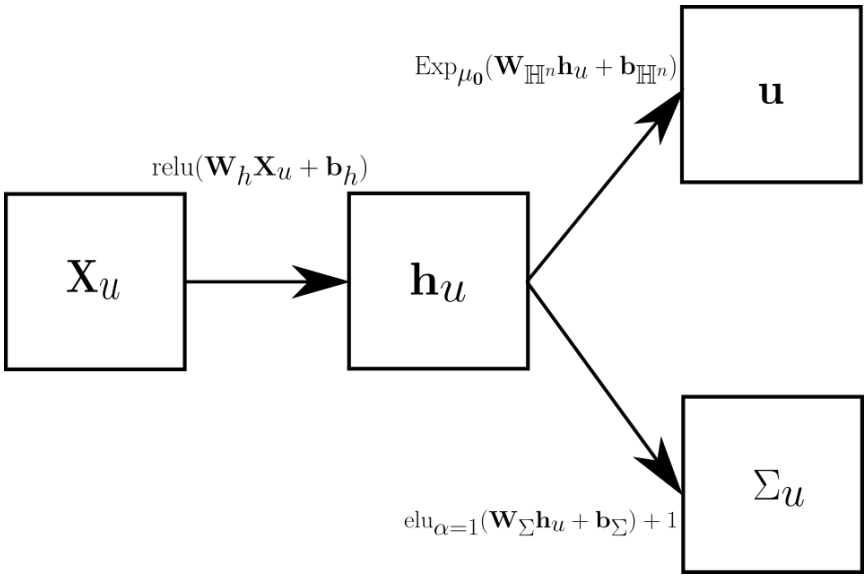


FIGURE 4.3: Schematic view of the embedder model. The model is a two-layer feed-forward neural network that is fed node attributes and outputs Gaussian distributions in hyperbolic space.

Returning to our embedder model, to compute $\mathbf{u} \in \mathbb{H}^n$, we propose the use of Exp_{μ_0}:

$$\mathbf{u} = \mathrm{Exp}_{\mu_0}\left(\mathbf{W}_{\mathbb{H}^n}\mathbf{h}_u + \mathbf{b}_{\mathbb{H}^n}\right) \tag{4.4}$$

where $\mathbf{W}_{\mathbb{H}^n} \in \mathbb{R}^{n \times d'}$ and $\mathbf{b}_{\mathbb{H}^n} \in \mathbb{R}^n$ are a weight matrix and bias to learn. The use of Exp_{μ_0} as a non-linearity provides a principled mapping from n-dimensional Euclidean space ($\mathbb{R}^n = T_{\mu_0}\mathbb{H}^n$) to $\mathbb{H}^n$ as required. Figure 4.2 gives an example of Exp_{μ_0} transporting a vector from $\mathbb{R}$ to $\mathbb{H}^1$.

Complete Embedder Model

We combine equation (4.1), equation (4.2) and equation (4.4) to construct our embedder. Figure 4.3 provides a schematic overview of the complete model.

Number of Parameters of Embedder Model

In total, three weight matrices ($\mathbf{W}_h \in \mathbb{R}^{d' \times d}$, $\mathbf{W}_\Sigma \in \mathbb{R}^{n \times d'}$, and $\mathbf{W}_{\mathbb{H}^n} \in \mathbb{R}^{n \times d'}$) and biases ($\mathbf{b}_h \in \mathbb{R}^{d'}$, $\mathbf{b}_\Sigma \in \mathbb{R}^n$, and $\mathbf{b}_{\mathbb{H}^n} \in \mathbb{R}^n$) are to be learned. That is a total of $d \times d' + d' + 2(d' \times n + n)$ parameters.

4.2.5 Node Similarity Measurement Step

This section describes the procedure for computing the similarity measure for any pair of nodes, using the Gaussian distributions provided by the embedder. We use Kullback-Leibler divergence as the measure of similarity between node distributions. For continuous distributions P and Q, Kullback-Leibler divergence $D_{KL}(P \mid\mid Q)$ is given by:

$$D_{KL}(P \mid\mid Q) := \int_x P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{4.5}$$

We see immediately that equation (4.5) is asymmetric in P and Q. When P and Q are normal distributions, equation (4.5) simplifies to:

$$D_{KL}(P \mid\mid Q) = \frac{1}{2}\left[tr\left(\Sigma_u^{-1}\Sigma_v\right) + \left(\mu_u - \mu_v\right)^T \Sigma_u^{-1}\left(\mu_u - \mu_v\right) - n - \log\left(\frac{det(\Sigma_v)}{det(\Sigma_u)}\right)\right] \tag{4.6}$$

where $tr(.)$ denotes the trace of a matrix and $det(.)$ denotes the determinant of a matrix.

Mapping Hyperbolic Co-ordinates to Vectors in Euclidean Space

Since our embedder outputs Euclidean covariance matrices, we only require a modification to the term $\left(\mu_u - \mu_v\right)^T \Sigma_u^{-1}\left(\mu_u - \mu_v\right)$ in order to apply equation (4.6). To this end, we propose a mapping from the hyperbolic mean vectors output by the embedder to Euclidean vectors, such that hyperbolic relationships are preserved. This mapping preserves the hyperbolic distance between a given node pair, while allowing for a simple form of the training objective (see section 4.2.6). For the remainder of this section, we will introduce a mapping $\Psi_{\mathbf{u}}$ that will replace the term $\left(\mu_u - \mu_v\right)^T \Sigma_u^{-1}\left(\mu_u - \mu_v\right)$ to reflect a hyperbolic distribution, rather than a Euclidean one. $\Psi_{\mathbf{u}}$ will behave exactly like the original term in that it will be a weighted sum over a distance vector. Furthermore, when all variances are identity matrices, then our derived equivalent to equation (4.6) will become equivalent to hyperbolic distance squared as expected[3].

[3]Note that this means that we can fix $\Sigma = \mathbf{I}_N$ to embed undirected networks. This form of section 4.2.6 is equivalent to the loss of HEAT (equation (3.8)).

For a node pair (u, v), our mapping, Ψ, is a two step procedure that relies upon $\mathbf{u}$ – the hyperbolic mean of node u – to serve as an "anchor point". The two steps are

1. transport vectors from $\mathbb{H}^n$ to the tangent space of the anchor point $T_{\mathbf{u}}\mathbb{H}^n$;
2. then, transport vectors from $T_{\mathbf{u}}\mathbb{H}^n$ to the tangent space of the bottom tip of the hyperboloid $T_{\mu_\mathbf{0}}\mathbb{H}^n$, which is the n-dimensional Euclidean space[4].

Transporting Vectors from $\mathbb{H}^n$ to the Tangent Space of an Anchor Point

To transport the hyperbolic mean of node v – the vector $\mathbf{v} \in \mathbb{H}^n$ – from a point on the hyperboloid to a point on the tangent space of the anchor point $T_{\mathbf{u}}\mathbb{H}^n$, we take the inverse of the exponential map, that we call the *logarithmic map* (Nagano et al., 2019).

As previously mentioned, the anchor point used in the mapping is the hyperbolic position of the node in the network that we are interested in measuring similarity with. For example, suppose that we were interested in computing the similarity of node v to node u. Then we would select the hyperbolic position of node u, given by $\mathbf{u}$, as the anchor point. The purpose of the first step (transporting to the tangent space $T_{\mathbf{u}}\mathbb{H}^n$ of the anchor node u) is to preserve the relationship between node v and the anchor node u. Skipping this step and mapping each point directly from $\mathbb{H}^n$ to Euclidean space (using $\text{Log}_{\mu_\mathbf{0}} : \mathbb{H}^n \to \mathbb{R}^n$, for example), would distort the distance between nodes u and v. This is because $\text{Log}_{\mu_\mathbf{0}}$ would preserve the distances to $\mu_\mathbf{0}$ (see equation (1.34)) and not distances between nodes u and v (see equation (1.35)).

Transporting from Anchor Node Tangent Space to $T_{\mu_\mathbf{0}}\mathbb{H}^n$

After transporting from $\mathbf{v}$ to $\mathbf{x} \in T_{\mathbf{u}}\mathbb{H}^n$, we require a second step to map to Euclidean space. This step must preserve the Minkowski norm of $\mathbf{x}$ – since this is the hyperbolic distance between the nodes u and v – while transforming $\mathbf{x}$ into a vector that is orthogonal to the bottom tip of the hyperboloid (since we know that all of the vectors orthogonal to the bottom tip are Euclidean vectors). The purpose of mapping to Euclidean vectors is that they allow for a simple form of the measurement of node similarity (Bojchevski and Günnemann, 2018).

[4]Recall $T_{\mu_\mathbf{0}}\mathbb{H}^n = \mathbb{R}^n$. See equation (1.30).

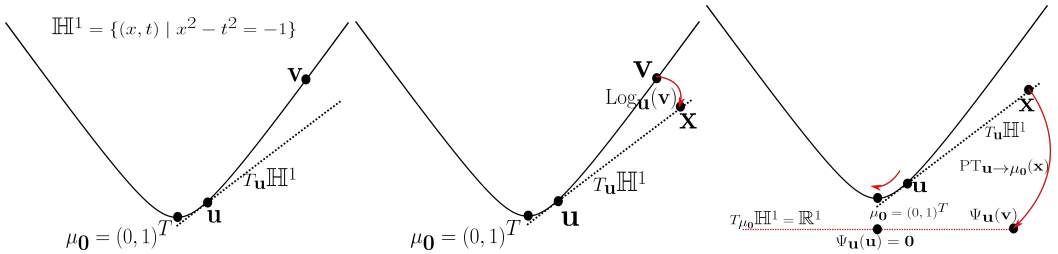


FIGURE 4.4: Example of $\Psi_{\mathbf{u}}$ on the hyperboloid. (A) shows a representation of $\mathbb{H}^1$ a one dimensional manifold in two dimensional Minkowski space $\mathbb{R}^{1:1}$. Three points in $\mathbb{H}^1$ are highlighted: the bottom tip of the hyperboloid $\mu_{\mathbf{0}}$, the anchor point $\mathbf{u}$ and $\mathbf{v}$. (B) shows how the logarithmic map takes $\mathbf{v}$ to a vector $\mathbf{x} \in T_{\mathbf{v}}\mathbb{H}^1$. The Minkowski norm of $\mathbf{x}$ is equal to the hyperbolic distance between $\mathbf{u}$ and $\mathbf{v}$. $||\mathbf{x}||_{\mathbb{R}^{1:1}} = D_{\mathbb{H}^1}(\mathbf{u}, \mathbf{v})$. (C) shows how $\mathrm{PT}_{\mathbf{u}\to\mu_{\mathbf{0}}}$ simultaneously transports $\mathbf{u}$ to $\mu_{\mathbf{0}}$ and $\mathbf{x}$ to $\Psi_{\mathbf{u}}(\mathbf{v}) \in \mathbb{R}^1$. It also shows that $\Psi_{\mathbf{u}}(\mathbf{u}) = \mathbf{0}$.

Combining the Two Steps

We now combine the above two steps to map from $\mathbb{H}^n$ to $\mathbb{R}^n$. For two points $\mathbf{u}, \mathbf{v} \in \mathbb{H}^n$, we can map to the Euclidean space $\left(T_{\mu_{\mathbf{0}}}\mathbb{H}^n\right)$ by combining the logarithmic map operation (equation (1.33)) and the parallel transport[5] to $\mu_{\mathbf{0}}$ operation $\mathrm{PT}_{\mathbf{u}\to\mu_{\mathbf{0}}}(\mathbf{x})$ as:

$$\mathbf{y} = \mathrm{PT}_{\mathbf{u}\to\mu_{\mathbf{0}}}(\mathbf{x}) = \mathbf{x} + \frac{\langle \mu_{\mathbf{0}} - \beta\mathbf{u}, \mathbf{x}\rangle_{\mathbb{R}^{n:1}}}{\beta + 1}(\mathbf{u} + \mu_{\mathbf{0}}) \tag{4.7}$$

where $\beta = -\langle \mathbf{u}, \mu_{\mathbf{0}}\rangle_{\mathbb{R}^{n:1}}$.

We now introduce $\Psi_{\mathbf{u}}$ as the composition of equation (1.33) and equation (4.7):

$$\Psi_{\mathbf{u}} : \mathbb{H}^n \to \mathbb{R}^n := \mathrm{PT}_{\mathbf{u}\to\mu_{\mathbf{0}}} \circ \mathrm{Log}_{\mathbf{u}} \tag{4.8}$$

Figure 4.4 gives an example of $\Psi_{\mathbf{u}}$ operating on the one-dimensional hyperboloid.

We apply our mapping function $\Psi_{\mathbf{u}}$ to map hyperbolic co-ordinates to vectors in Euclidean space. For a given node pair (u, v), each mapping from their hyperbolic co-ordinates $\mathbf{u}$ and $\mathbf{v}$ to their Euclidean positions $\Psi_{\mathbf{u}}(\mathbf{u})$ and $\Psi_{\mathbf{u}}(\mathbf{v})$ is performed "relative" to the anchor node u.

Here we note some properties of $\Psi_{\mathbf{u}}$. In general, for $\mathbf{x}, \mathbf{y}, \mathbf{v} \in \mathbb{H}^n$:

$$\mathbf{x} \neq \mathbf{y} \implies \Psi_{\mathbf{x}}(\mathbf{v}) \neq \Psi_{\mathbf{y}}(\mathbf{v}) \tag{4.9}$$

[5] Recall the parallel transport operation $\mathrm{PT}_{\mathbf{u}\to\mathbf{v}}$ given in equation (1.36).

and so the Euclidean position of node v (relative to anchor node u), $\Psi_{\mathbf{u}}(\mathbf{v})$, is wholly dependent on hyperbolic coordinate $\mathbf{u}$ of the anchor node u that is used in the mapping. This follows from equation (1.35). In other words, changing $\mathbf{u}$ would result in a different position for $\Psi_{\mathbf{u}}(\mathbf{v})$.

The Euclidean norm of $\Psi_{\mathbf{u}}(\mathbf{v})$ is equal to the hyperbolic distance between the mean vectors of nodes u and v on the hyperboloid (Nagano et al., 2019). That is:

$$||\Psi_{\mathbf{u}}(\mathbf{v})|| = D_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v}) = \text{arccosh}(-\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{R}^{n,1}}) \tag{4.10}$$

This follows from equation (1.34). Further, the direction of the vector $\Psi_{\mathbf{u}}(\mathbf{v})$ is the same as the direction of the geodesic from $\mathbf{u}$ to $\mathbf{v}$ (Nagano et al., 2019).

We can interpret $\Psi_{\mathbf{u}}$ as a distance preserving map for all nodes relative to node u. For all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{H}^n$, $\Psi_{\mathbf{u}}$ preserves distances $D_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v})$ and $D_{\mathbb{H}^n}(\mathbf{u}, \mathbf{w})$, but not $D_{\mathbb{H}^n}(\mathbf{v}, \mathbf{w})$ (Nagano et al., 2019). Further, we observe that for all $\mathbf{u} \in \mathbb{H}^n$:

$$\Psi_{\mathbf{u}}(\mathbf{u}) = \mathbf{0} \tag{4.11}$$

Figure 4.5 demonstrates an example mapping from $\mathbb{H}^2$ to $\mathbb{R}^2$, that illustrates how $\Psi_{\mathbf{u}}$ preserves hyperbolic distances to node u but distorts all distances between other nodes.

Node Similarity

Putting everything together, we now propose the following modified form for equation (4.6) as an asymmetric measure of node similarity in hyperbolic space:

$$D_{KL}(\mathcal{N}_v \,||\, \mathcal{N}_u) = \frac{1}{2} \sum_{i=1}^{n} \left[\frac{\Sigma_v^i}{\Sigma_u^i} + \frac{\left[\Psi_{\mathbf{u}}(\mathbf{v})^i\right]^2}{\Sigma_u^i} - 1 - \log\left(\frac{\Sigma_v^i}{\Sigma_u^i}\right) \right] \tag{4.12}$$

where $\mathbf{x}^i$ is the i-th component of vector $\mathbf{x}$. Note that we omit $\Psi_{\mathbf{u}}(\mathbf{u})$ since $\Psi_{\mathbf{u}}(\mathbf{u}) = \mathbf{0}$. In addition, the terms involving Σ in equation (4.6) simplify to a simple sum over vector elements since Σ is diagonal (Bojchevski and Günnemann, 2018).

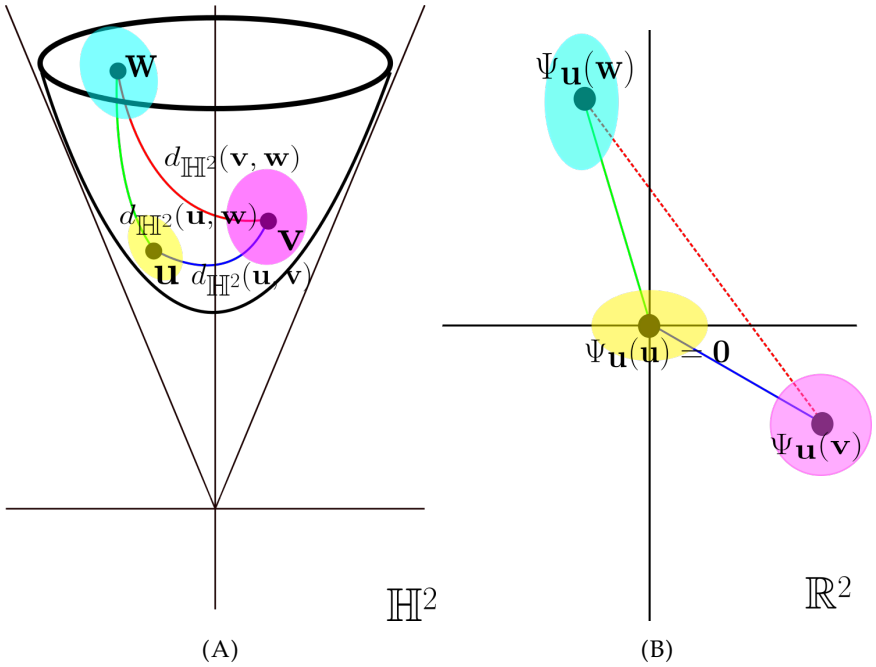


FIGURE 4.5: Here we show how the Ψ operator preserves distances relative to the vector used in the mapping. (A) shows three points $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{H}^2$ on the two-dimensional hyperboloid. The red, green and blue lines show the geodesics between the points with lengths of $D_{\mathbb{H}^2}(\mathbf{v}, \mathbf{w})$, $D_{\mathbb{H}^2}(\mathbf{u}, \mathbf{w})$, and $D_{\mathbb{H}^2}(\mathbf{u}, \mathbf{v})$ respectively. (B) shows their resulting positions in $\mathbb{R}^2$ after applying $\Psi_{\mathbf{u}}$. Distances $D_{\mathbb{H}^2}(\mathbf{u}, \mathbf{v})$ and $D_{\mathbb{H}^2}(\mathbf{u}, \mathbf{w})$ are preserved but $D_{\mathbb{H}^2}(\mathbf{v}, \mathbf{w})$ is not.

4.2.6 Node Representation Learning

The section describes the learning objective of HEADNet, using the node similarity computed in the previous section. The learning procedure is based on the idea that we aim to maximize the similarity between ground truth node pairs – the directed edges in the network – while simultaneously minimising the similarity between all other node pairs. Since node distributions are computed by the embedder model, here 'learning' refers to training the embedder model.

Learning Objective

To learn node representations, we select to optimise an energy-based objective (LeCun et al., 2006). Following previous works, we define the energy $\mathcal{E}_{uv}$ between two nodes u and v (which are described by normal distributions $\mathcal{N}_{\mathbf{u}}$ and $\mathcal{N}_{\mathbf{v}}$ to be:

$$\mathcal{E}_{uv} = D_{KL}(\mathcal{N}_v \,||\, \mathcal{N}_u) \tag{4.13}$$

and observe that, in general, $\mathcal{E}_{uv} \neq \mathcal{E}_{vu}$ as required (Bojchevski and Günnemann, 2018). In addition to readily providing a simple form for an asymmetric measure of similarity between learned node representations given by D_{KL}, embedding nodes to normal distributions rather than points can offer greater insight into role of the node in network. For example, distributions with greater uncertainty, measured by the determinant of the covariance matrix may correspond to nodes from highly diverse neighbourhoods in the original network.

In accordance with other energy-based methods, we aim to minimize the energy (ie: reduce the Kullback-Leibler divergence) for desirable node pairings and, likewise, maximize the energy of undesirable pairings. We define the directed edge set E and the non-edge set $V \times V \setminus E$ to be the sets of desirable and undesirable pairs respectively. We base our objective on negative log-likelihood:

$$L = \mathop{\mathbb{E}}_{\substack{(u,v)\sim E \\ S\sim \mathbb{S}^k_{uv}}} \left[\mathcal{E}_{uv} + \log \sum_{(u',v')\in S} \exp\left(-\mathcal{E}_{u'v'}\right) \right] \tag{4.14}$$

where $\mathbb{S}^k_{uv} = \{S \subset V \times V : |S| = k \wedge (u,v) \in S\}$ is the set of all subsets of node pairs containing exactly k elements and the pair (u,v). The first term ($\mathcal{E}_{uv}$) will cause the average energy of all pairs $(u,v) \in E$ to be reduced. The second, contrastive, term $\log \sum_{(u',v')\in S} \exp\left(-\mathcal{E}_{u'v'}\right)$ causes the energy of all undesirable pairs to be "pulled-up" a little. The energy of the correct answer is also pulled up, but not as hard as it is pushed down by the first term (LeCun et al., 2006). The parameter k is a hyper-parameter used to control the number of incorrect pairs seen for every correct pair.

4.2.7 Optimisation

All parameters in the embedder model are Euclidean and so updated using Adam optimiser (Kingma and Ba, 2014) with a learning rate of 0.001.

4.2.8 Connection to Variational Autoencoder

A natural comparison can be made between HEADNet and a Variational Autoencoder (VAE) (Kingma and Welling, 2013).

An autoencoder is a neural network designed for dimensionality reduction. The overall concept is simple and consists of two components: an encoder $f(\cdot)$ and a decoder $g(\cdot)$. The objective is to learn the best encoding-decoding scheme using an iterative optimisation process: for example, gradient descent (Rumelhart, Hinton, and Williams, 1985). At each iteration, we feed the autoencoder architecture (the encoder followed by the decoder) with some data $\mathbf{x}$, we compare the encoded-decoded output, $\hat{\mathbf{x}} = g(f(\mathbf{x}))$, with the initial data and back-propagate the error, $||\hat{\mathbf{x}} - \mathbf{x}||$, through the architecture to update the weights of the two networks. Thus, intuitively, the overall autoencoder architecture (encoder and decoder) creates a bottleneck for data that ensures only the main structured part of the information can go through and be reconstructed. The output of the encoder $\mathbf{z} = f(\mathbf{x})$ (the bottleneck) is often referred to as the 'latent space' underpinning the observed data and the act of encoding to a latent space can be naturally viewed as data embedding.

VAEs are autoencoders that address two drawbacks to traditional autoencoders. Namely, the lack of interpretability and preserved structure in the latent space. In theory, with a suitable deep encoder/decoder architecture, the latent space could be a single dimension, where each data point is represented as a real number, and the decoder exhibits no reconstruction loss. However, this latent representation would likely not be informative and provide no insight into the observed data – there would be no guarantee that similar data points would be represented (embedded) close together in the latent space, and no guaranteed that sampling the latent space would produce meaningful data.

VAEs make the natural link between autoencoders and the generation of new, meaningful data by regularising the latent space representation of the data such that both the aforementioned issues are addressed. The regularisation is achieved by encoding not to points in the latent space, but to distributions and applying a penalty

to learned distribution that encourage them to be as close to the chosen prior distribution, $p(\mathbf{z})$, as possible[6]. In this way, both a local and global regularisation of the latent space is ensured (local because of the variance control and global because of the mean control). It is this regularity that is required in order to make generative process possible, and can be expressed through two main properties: continuity (two close points in the latent space should not give two completely different contents once decoded) and completeness (for a chosen distribution, a point sampled from the latent space should give "meaningful" content once decoded). This penalty on the latent space representations, in addition to the standard reconstruction component of the loss, enacts a trade-off between a high-quality reconstruction of the data and regularity in the latent space which naturally regularises both models.

The connection between VAEs and HEADNet comes from the commonly selected choice of prior: $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbb{I})$, a standard Gaussian distribution; as well as the latent space penalty coming the form of Kullback Leibler divergence $D_{KL}(f(\mathbf{x}), \mathcal{N}(\mathbf{0}, \mathbb{I}))$. Like HEADNet, the encoder makes the common simplifying assumption of feature independence to encode data points to d-dimensional means and d-dimensional vectors corresponding to diagonal covariance matrices (i.e., $f(\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$). Furthermore, the architecture of the encoder neural network is similar to the architecture of the embedder for HEADNet (Kingma and Welling, 2013).

Through this lens, we can view HEADNet as an encoder that learns uses not one, but N priors (corresponding to the N nodes in the network) for regularisation, and that the priors are not fixed (only the shape of them). We leverage the structure in the network to select priors that we use to regularise each distribution, based on the distributions of the neighbours in the network. Another difference is that regularisation involves not only minimising D_{KL} between learned distributions and priors, but also maximising distances between selected priors (based on negative

[6]Note that the choice to embed to a distribution in the latent space necessitates a sampling from that learned distribution which is fed into the decoder. In other words, $\hat{\mathbf{x}} = g(\mathbf{z})$ where $\mathbf{z} \sim f(\mathbf{x})$. In order to train the two models, the *reparameterisation* trick is applied in order to back-propagate error across the sampling process. This essentially reformulates the sampling as a differentiable function of the distribution parameters. For example, when dealing with a Gaussian prior ($p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbb{I})$) and simplifying the encoded distribution with an assumption of independent variables, one can reformulate $\mathbf{z} \sim \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$ to the differentiable form $\mathbf{z} = \zeta \cdot \sigma(\mathbf{x}) + \mu(\mathbf{x})$, where $\zeta \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$ (Kingma and Welling, 2013).

sampling).

Clearly, a key difference between a VAE and HEADNet is the omission of the decoder in HEADNet. This leads to a natural extension: By simultaneously learning a decoder along with our proposed embedder, we would be able to generate new, meaningful entities in our system. For example: consider a social network where nodes are annotated with characteristics that describe a person. Each link describes a friendship, and we find that the network displays characteristics that would make us believe that a hyperbolic metric space may underpin it. After learning a hyperbolic encoder (HEADNet) and a decoder, we would be able to, given a person, "generate people" that might be friends with that person by sampling from the learned distribution for that person and decoding to obtain the attributes that that theoretical person may exhibit. We leave this potential extension as future work.

4.3 Experimental Validation

4.3.1 Datasets

Synthetic Networks

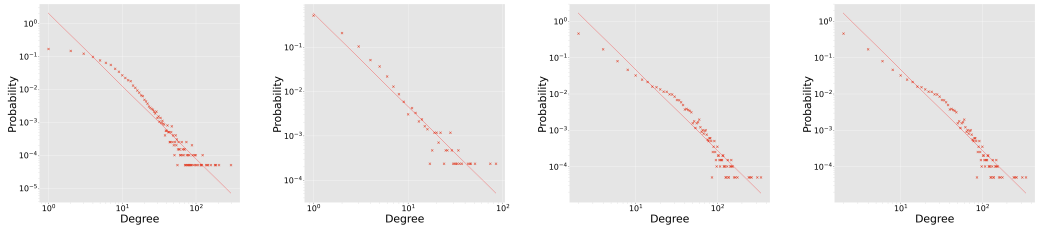
We generate 30 synthetic scale free directed networks with $N = 1000$ nodes. We set α, the probability for adding a new node connected to an existing node chosen randomly according to the in-degree distribution, to 0.41; we set β, the probability for adding an edge between two existing nodes, to 0.54; and we set γ, the probability for adding a new node connected to an existing node chosen randomly according to the out-degree distribution, to 0.05 (Bollobás et al., 2003).

Real World Networks

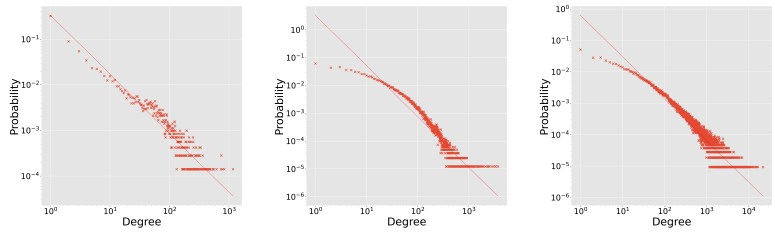
Table 4.1 details all of the real world networks used and figure 4.6 plots their degree distributions. We see that all real world networks in this study follow a broadly power-law distribution, suggesting that a hyperbolic embedding is a suitable approach. For the two social networks, Twitter and Google+, we use the 10000 features with the greatest variance.

TABLE 4.1: Real World Network statistics. *Key:* N is the number of nodes, $|E|$ is the number of edges, d is attribute dimension. - indicates no attributes.

Network	N	$\|E\|$	d
Cora_ML (Bojchevski and Günnemann, 2018)	2995	8416	2879
Citeseer (Bojchevski and Günnemann, 2018)	4230	5358	602
Pubmed (Bojchevski and Günnemann, 2018)	18230	79612	500
Cora (Bojchevski and Günnemann, 2018)	19793	65311	8710
Wiki_vote (Leskovec and Sosič, 2016)	7115	103689	-
Twitter (Leskovec and Mcauley, 2012)	81306	1768149	10000
Google+ (Leskovec and Mcauley, 2012)	107614	13673453	10000



(A) Cora_ML (B) Citeseer (C) Pubmed (D) Cora



(E) Wiki Vote (F) Twitter (G) Google+

FIGURE 4.6: Degree distributions of the network datasets. Both axes are on a log scale. (A) Cora_ML, (B) Citeseer, (C) Pubmed, (D) Cora, (E) Wiki Vote, (F) Twitter, and (G) Google+.

4.3.2 Benchmark Algorithms

Since there is no algorithm that can embed attributed directed networks into hyperbolic space, we select Euclidean embedding algorithms that can handle directed networks as benchmark algorithms to compare against (see table 4.2). For all benchmark algorithms, we adopt the default hyper-parameter settings. For LINE, we set the order to second only as the authors claim that this is suitable for directed graphs. We set the negative ratio to 10 and the number of epochs to 1000. For ATP, we break cycles using the hierarchical grouping method that the authors demonstrated obtained good results (Sun et al., 2017). We select the log and harmonic transforms as they are the best performing (Sun et al., 2019). For $\text{G2G}_{*,K=\{1,3\}}$, we set K to 1 and 3 respectively $\text{G2G}_{\text{NA},*}$ denotes G2G without attributes. For all G2G models, we

TABLE 4.2: Description of benchmark algorithms.

Algorithm	Variant	Description
ATP (Sun et al., 2019)	log	ATP (log transform)
	harmonic	ATP (harmonic transform)
LINE (Tang et al., 2015)	2nd order	LINE (second order proximity)
G2G (Bojchevski and Günnemann, 2018)	$\text{G2G}_{\text{NA},K=1}$	G2G w/o attributes, $K = 1$
	$\text{G2G}_{\text{NA},K=3}$	G2G w/o attributes, $K = 3$
	$\text{G2G}_{K=1}$	G2G w/ attributes, $K = 1$
	$\text{G2G}_{K=3}$	G2G w/ attributes, $K = 3$
N&K (Nickel and Kiela, 2017)	-	Undirected Poincaré ball
HEADNet	$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	HEADNet w/o attributes, identity variance
	$\text{HEADNet}_{\text{NA}}$	HEADNet w/o attributes
	$\text{HEADNet}_{\Sigma=\mathbb{I}}$	HEADNet w/ attributes, identity variance
	HEADNet	HEADNet w/ attributes

fix the hidden dimension to 128 and train for the default maximum of 2000 epochs. For N&K, we set learning rate to 1.0, epochs to 1500, number of negative samples to 10, batch size to 512, and burn-in epochs to 20. For HEADNet, we train for 1000 epochs, stopping early if there is no improvement for 100 epochs, and set the ratio of negative to positive pairs to $k = 10$. We set the the dimension of the latent hidden dimension to $d' = 128$. We set a maximum memory limit of 50GB for each algorithm.

4.3.3 Network Reconstruction

We use network reconstruction to evaluate the capacity of the learned embeddings to reflect the original data (Nickel and Kiela, 2017). After training our model to convergence using complete information, we compute distances in the embedding space between all pairs of nodes according to both models. We assign the true edges in the network positive labels and all other pairs as negatives. We then rank node pairs by their distance in increasing order and, with a sliding threshold, compute both the average precision (AP) and the area under the receiver operating characteristic (AUROC) curve, as well as recommender metrics: mAP, and p@k.

Table 4.3 and table 4.4 provides a summary of the network reconstruction results for embedding dimension 25+25. Results for further dimensions can be found in appendix C.1. ATP, LINE, $\text{G2G}_{\text{NA},K=1}$, $\text{G2G}_{\text{NA},K=3}$, and $\text{HEADNet}_{\text{NA}}$ are algorithms that do not use attributes to perform an embedding. $\text{G2G}_{K=1}$, $\text{G2G}_{K=3}$, and HEADNet use attributes. Comparing between like algorithms with and without attributes, for example: $\text{G2G}_{\text{NA},K=1}$ and $\text{G2G}_{K=1}$, we see that algorithms that do not

TABLE 4.3: Summary of the network reconstruction task on synthetic, Cora_ML and Citeseer networks for an embedding dimension of 25+25. For clarity, we only report the results for an embedding dimension of 25+25, but obtain similar results for all dimensions (see appendix C.1). $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. A dash (–) indicates that a network did not have attributes. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the t-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Synthetic								
ATP (log)	617.7(59.5)	0.642(0.034)	0.663(0.029)	0.018(0.011)	0.006(0.018)	0.011(0.014)	0.017(0.015)	0.030(0.020)
ATP (harmonic)	611.4(53.7)	0.646(0.030)	0.663(0.030)	0.018(0.011)	0.006(0.018)	0.011(0.014)	0.017(0.015)	0.028(0.020)
LINE	1208.6(50.2)	0.301(0.017)	0.379(0.005)	0.010(0.002)	0.014(0.004)	0.048(0.012)	0.084(0.027)	0.119(0.035)
$\text{G2G}_{\text{NA},K=1}$	83.9(9.7)	0.952(0.007)	0.896(0.015)	0.216(0.020)	0.139(0.018)	0.121(0.025)	0.073(0.026)	0.048(0.017)
$\text{G2G}_{\text{NA},K=3}$	95.0(15.0)	0.946(0.009)	0.875(0.023)	0.129(0.018)	0.080(0.016)	0.105(0.025)	0.142(0.031)	0.288(0.072)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	7.4(1.9)	0.996(0.001)	0.994(0.002)	0.661(0.059)	0.547(0.087)	0.337(0.018)	0.236(0.011)	0.142(0.007)
$\text{G2G}_{K=1}$	–	–	–	–	–	–	–	–
$\text{G2G}_{K=3}$	–	–	–	–	–	–	–	–
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	–	–	–	–	–	–	–	–
Significance Test								
NK	52.5(8.7)	0.970(0.006)	0.951(0.008)	0.270(0.017)	0.250(0.015)	0.161(0.010)	0.124(0.008)	0.085(0.006)
$\text{HEADNet}_{\text{NA}}$	**3.5(1.1)**	**0.999(0.001)**	**0.998(0.001)**	**0.832(0.048)**	**0.752(0.071)**	**0.763(0.063)**	**0.769(0.068)**	**0.793(0.072)**
t-statistic	3.06E+01	2.71E+01	3.02E+01	6.02E+01	3.80E+01	5.14E+01	5.12E+01	5.40E+01
p-value	2.18E-24	8.61E-23	2.13E-24	2.60E-38	2.39E-28	1.54E-31	6.05E-31	3.11E-31
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	–	–	–	–	–	–	–	–
t-statistic	–	–	–	–	–	–	–	–
p-value	–	–	–	–	–	–	–	–
$p < 0.05$	–	–	–	–	–	–	–	–
Cora_ML								
ATP (log)	4112.2(26.7)	0.511(0.003)	0.526(0.003)	0.035(0.001)	0.019(0.002)	0.032(0.002)	0.044(0.002)	0.072(0.004)
ATP (harmonic)	4069.8(26.4)	0.516(0.003)	0.534(0.003)	0.038(0.001)	0.026(0.002)	0.036(0.002)	0.045(0.002)	0.070(0.004)
LINE	3972.1(35.1)	0.528(0.004)	0.480(0.003)	0.061(0.002)	0.100(0.004)	0.122(0.004)	0.136(0.005)	0.137(0.008)
$\text{G2G}_{\text{NA},K=1}$	55.9(5.0)	0.993(0.001)	0.991(0.001)	0.598(0.007)	0.538(0.012)	0.514(0.012)	0.503(0.012)	0.481(0.017)
NK	103.8(3.4)	0.988(0.000)	0.986(0.001)	0.581(0.003)	0.604(0.007)	0.428(0.003)	0.337(0.002)	0.220(0.001)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	11.4(3.3)	0.999(0.000)	0.998(0.001)	0.817(0.011)	0.766(0.012)	0.574(0.009)	0.447(0.007)	0.276(0.003)
$\text{G2G}_{K=1}$	64.7(6.9)	0.992(0.001)	0.989(0.001)	0.585(0.008)	0.535(0.013)	0.492(0.010)	0.478(0.012)	0.456(0.018)
$\text{G2G}_{K=3}$	54.4(5.6)	0.994(0.001)	0.991(0.001)	0.609(0.013)	0.559(0.012)	0.538(0.015)	0.527(0.016)	0.504(0.014)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	17.3(2.8)	0.998(0.000)	0.997(0.001)	0.752(0.003)	0.697(0.006)	0.527(0.002)	0.413(0.001)	0.260(0.001)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	45.5(5.3)	0.995(0.001)	0.992(0.001)	0.643(0.009)	0.586(0.012)	0.583(0.012)	0.571(0.012)	0.538(0.016)
$\text{HEADNet}_{\text{NA}}$	**3.1(1.3)**	**1.000(0.000)**	**1.000(0.000)**	**0.965(0.012)**	**0.960(0.014)**	**0.954(0.016)**	**0.949(0.018)**	**0.946(0.018)**
t-statistic	4.26E+01	4.26E+01	2.80E+01	1.16E+02	1.13E+02	1.02E+02	9.87E+01	9.18E+01
p-value	1.87E-30	1.87E-30	9.20E-25	1.26E-66	3.27E-68	1.51E-64	2.28E-59	5.82E-64
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**4.4(1.4)**	**1.000(0.000)**	**0.999(0.000)**	**0.939(0.016)**	**0.932(0.017)**	**0.912(0.021)**	**0.902(0.021)**	**0.894(0.024)**
t-statistic	4.12E+01	4.12E+01	2.72E+01	8.75E+01	9.20E+01	7.48E+01	7.48E+01	6.78E+01
p-value	3.69E-30	3.69E-30	2.44E-24	5.71E-53	1.26E-58	8.78E-51	8.13E-49	8.21E-52
$p < 0.05$	1	1	1	1	1	1	1	1
Citeseer								
ATP (log)	2253.6(33.5)	0.580(0.006)	0.588(0.009)	0.024(0.001)	0.008(0.001)	0.027(0.003)	0.034(0.005)	0.068(0.016)
ATP (harmonic)	2238.1(31.6)	0.582(0.006)	0.588(0.009)	0.023(0.001)	0.008(0.001)	0.025(0.003)	0.032(0.005)	0.064(0.017)
LINE	3163.9(29.4)	0.410(0.005)	0.417(0.002)	0.034(0.001)	0.036(0.002)	0.059(0.004)	0.052(0.005)	0.029(0.019)
$\text{G2G}_{\text{NA},K=1}$	6.8(1.6)	0.999(0.000)	0.998(0.000)	0.801(0.006)	0.714(0.009)	0.591(0.012)	0.563(0.015)	0.551(0.070)
NK	27.4(2.4)	0.995(0.000)	0.995(0.001)	0.722(0.004)	0.645(0.008)	0.364(0.002)	0.252(0.001)	0.141(0.000)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	3.8(1.2)	0.999(0.000)	0.999(0.000)	0.883(0.002)	0.827(0.005)	0.442(0.001)	0.297(0.001)	0.158(0.000)
$\text{G2G}_{K=1}$	16.8(2.9)	0.997(0.001)	0.993(0.002)	0.635(0.012)	0.532(0.016)	0.400(0.021)	0.351(0.023)	0.417(0.069)
$\text{G2G}_{K=3}$	17.4(2.7)	0.997(0.001)	0.993(0.002)	0.620(0.009)	0.511(0.011)	0.406(0.016)	0.372(0.020)	0.462(0.080)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	15.0(1.9)	0.997(0.000)	0.994(0.002)	0.643(0.003)	0.540(0.006)	0.343(0.001)	0.246(0.001)	0.143(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	6.2(1.7)	0.999(0.000)	0.999(0.001)	0.780(0.007)	0.670(0.013)	0.622(0.017)	0.592(0.026)	0.593(0.069)
$\text{HEADNet}_{\text{NA}}$	**2.3(0.8)**	**1.000(0.000)**	**1.000(0.000)**	**0.947(0.011)**	**0.921(0.016)**	**0.892(0.026)**	**0.895(0.028)**	**0.929(0.041)**
t-statistic	1.15E+01	1.15E+01	8.74E+00	7.05E+01	6.68E+01	4.67E+01	4.32E+01	2.30E+01
p-value	6.59E-15	6.59E-15	8.19E-12	6.45E-53	3.30E-55	2.36E-43	5.15E-46	1.25E-27
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	7.8(2.1)	0.999(0.000)	0.996(0.002)	0.773(0.022)	**0.711(0.030)**	**0.702(0.022)**	**0.684(0.026)**	**0.699(0.088)**
t-statistic	-3.14E+00	-3.17E+00	-8.05E+00	-1.51E+00	6.78E+00	1.55E+01	1.35E+01	5.19E+00
p-value	9.99E-01	9.99E-01	1.00E+00	9.30E-01	2.07E-08	5.46E-22	7.92E-20	1.57E-06
$p < 0.05$	0	0	0	0	1	1	1	1

TABLE 4.4: Summary of the network reconstruction task on the Pubmed, Cora and Wiki Vote networks for an embedding dimension of 25+25. For clarity, we only report the results for an embedding dimension of 25+25, but obtain similar results for all dimensions (see appendix C.1). $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. $\text{HEADNet}_{\Sigma=\mathbb{I}}$ denoted HEADNet with an identity variance. A dash (–) indicates that a network did not have attributes. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Pubmed								
ATP (log)	64006.9(226.7)	0.278(0.003)	0.381(0.001)	0.063(0.003)	0.060(0.004)	0.118(0.008)	0.130(0.009)	0.134(0.010)
ATP (harmonic)	63820.6(225.7)	0.280(0.003)	0.387(0.001)	0.064(0.003)	0.064(0.004)	0.121(0.008)	0.132(0.009)	0.134(0.010)
LINE	59709.1(278.7)	0.326(0.003)	0.384(0.001)	0.024(0.000)	0.057(0.001)	0.103(0.002)	0.114(0.003)	0.109(0.004)
$\text{G2G}_{\text{NA},K=1}$	186.9(14.5)	0.998(0.000)	0.997(0.000)	0.846(0.006)	0.800(0.007)	0.832(0.011)	0.851(0.009)	0.861(0.007)
NK	409.6(12.8)	0.995(0.000)	0.994(0.000)	0.750(0.002)	0.754(0.003)	0.520(0.001)	0.408(0.001)	0.279(0.000)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	14.6(3.0)	1.000(0.000)	1.000(0.000)	0.970(0.001)	0.948(0.002)	0.630(0.000)	0.484(0.000)	0.325(0.000)
$\text{G2G}_{K=1}$	1856.6(140.4)	0.979(0.002)	0.969(0.002)	0.263(0.016)	0.166(0.011)	0.270(0.013)	0.327(0.015)	0.387(0.015)
$\text{G2G}_{K=3}$	1473.9(68.0)	0.983(0.001)	0.975(0.001)	0.281(0.008)	0.204(0.007)	0.319(0.007)	0.379(0.007)	0.439(0.008)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	262.4(10.8)	0.997(0.000)	0.996(0.000)	0.629(0.001)	0.532(0.002)	0.435(0.001)	0.370(0.001)	0.274(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	47.7(5.5)	0.999(0.000)	0.999(0.000)	0.915(0.002)	0.876(0.003)	0.915(0.005)	0.919(0.005)	0.922(0.004)
$\text{HEADNet}_{\text{NA}}$	**18.4(2.8)**	**1.000(0.000)**	**1.000(0.000)**	**0.970(0.001)**	**0.950(0.002)**	**0.983(0.001)**	**0.988(0.001)**	**0.991(0.001)**
t-statistic	2.60E+01	2.60E+01	1.62E+01	1.17E+02	1.15E+02	6.73E+01	7.79E+01	9.45E+01
p-value	1.89E-28	1.89E-28	4.00E-20	4.56E-52	1.06E-67	1.46E-37	2.10E-39	2.18E-42
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	272.2(12.6)	0.997(0.000)	0.995(0.000)	0.616(0.009)	0.514(0.009)	0.681(0.007)	0.732(0.006)	0.771(0.006)
t-statistic	-8.96E+01	-8.96E+01	-6.94E+01	-1.70E+02	-2.04E+02	-1.47E+02	-1.29E+02	-1.17E+02
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0	0	0	0	0
Cora								
ATP (log)	33041.2(69.9)	0.494(0.001)	0.508(0.001)	0.049(0.000)	0.043(0.001)	0.053(0.001)	0.061(0.001)	0.077(0.002)
ATP (harmonic)	32769.0(69.1)	0.498(0.001)	0.514(0.001)	0.053(0.000)	0.053(0.001)	0.058(0.001)	0.064(0.001)	0.078(0.001)
LINE	32541.2(108.1)	0.502(0.002)	0.469(0.001)	0.087(0.001)	0.151(0.002)	0.157(0.002)	0.162(0.002)	0.166(0.003)
$\text{G2G}_{\text{NA},K=1}$	135.6(11.0)	0.998(0.000)	0.997(0.000)	0.808(0.010)	0.795(0.011)	0.742(0.014)	0.724(0.013)	0.719(0.011)
NK	401.7(11.3)	0.994(0.000)	0.993(0.000)	0.740(0.001)	0.787(0.002)	0.574(0.001)	0.449(0.001)	0.286(0.000)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	13.9(2.8)	1.000(0.000)	1.000(0.000)	0.958(0.001)	0.941(0.002)	0.718(0.001)	0.561(0.000)	0.344(0.000)
$\text{G2G}_{K=1}$	181.7(20.9)	0.997(0.000)	0.996(0.000)	0.763(0.015)	0.746(0.017)	0.684(0.019)	0.672(0.018)	0.675(0.015)
$\text{G2G}_{K=3}$	122.4(14.2)	0.998(0.000)	0.998(0.000)	0.813(0.013)	0.800(0.014)	0.756(0.016)	0.739(0.015)	0.735(0.012)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	25.7(2.7)	1.000(0.000)	0.999(0.000)	0.929(0.001)	0.919(0.002)	0.694(0.001)	0.544(0.001)	0.337(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	81.3(10.1)	0.999(0.000)	0.998(0.000)	0.865(0.008)	0.854(0.008)	0.824(0.011)	0.802(0.011)	0.783(0.009)
$\text{HEADNet}_{\text{NA}}$	**6.8(1.5)**	**1.000(0.000)**	**1.000(0.000)**	**0.988(0.002)**	**0.988(0.001)**	**0.983(0.003)**	**0.980(0.004)**	**0.975(0.005)**
t-statistic	4.00E+01	4.00E+01	3.23E+01	8.70E+01	9.59E+01	7.76E+01	8.59E+01	1.01E+02
p-value	4.53E-28	4.53E-28	1.40E-25	3.36E-40	5.45E-40	1.58E-38	6.33E-43	1.17E-52
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**11.9(2.0)**	**1.000(0.000)**	**1.000(0.000)**	**0.973(0.003)**	**0.972(0.003)**	**0.966(0.004)**	**0.961(0.004)**	**0.956(0.005)**
t-statistic	3.69E+01	3.69E+01	2.97E+01	7.17E+01	7.77E+01	6.75E+01	7.64E+01	9.01E+01
p-value	1.06E-27	1.06E-27	2.46E-25	1.39E-44	3.91E-46	4.39E-39	3.36E-42	2.33E-52
$p < 0.05$	1	1	1	1	1	1	1	1
Wiki Vote								
ATP (log)	53732.0(186.4)	0.482(0.002)	0.503(0.001)	0.030(0.000)	0.013(0.001)	0.022(0.001)	0.027(0.001)	0.041(0.001)
ATP (harmonic)	53742.5(191.0)	0.482(0.002)	0.503(0.001)	0.030(0.000)	0.013(0.001)	0.022(0.001)	0.027(0.001)	0.041(0.001)
LINE	72072.5(208.1)	0.305(0.002)	0.375(0.001)	0.034(0.001)	0.070(0.002)	0.127(0.003)	0.152(0.004)	0.176(0.005)
$\text{G2G}_{\text{NA},K=1}$	3933.7(259.6)	0.962(0.003)	0.950(0.003)	0.550(0.010)	0.502(0.011)	0.410(0.014)	0.392(0.012)	0.410(0.012)
$\text{G2G}_{\text{NA},K=3}$	4676.5(167.6)	0.955(0.002)	0.932(0.003)	0.215(0.014)	0.174(0.013)	0.226(0.011)	0.259(0.010)	0.309(0.011)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	893.8(23.5)	0.991(0.000)	0.988(0.000)	0.838(0.004)	0.839(0.008)	0.580(0.002)	0.475(0.001)	0.360(0.001)
$\text{G2G}_{K=1}$	–	–	–	–	–	–	–	–
$\text{G2G}_{K=3}$	–	–	–	–	–	–	–	–
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	–	–	–	–	–	–	–	–
Significance Test								
NK	4269.5(26.1)	0.959(0.000)	0.963(0.000)	0.205(0.001)	0.279(0.003)	0.230(0.002)	0.210(0.001)	0.188(0.001)
$\text{HEADNet}_{\text{NA}}$	**156.6(12.5)**	**0.998(0.000)**	**0.998(0.000)**	**0.936(0.012)**	**0.923(0.024)**	**0.943(0.006)**	**0.932(0.006)**	**0.917(0.007)**
t-statistic	7.80E+02	7.80E+02	4.31E+02	3.36E+02	1.49E+02	6.35E+02	6.14E+02	5.84E+02
p-value	1.36E-88	1.36E-88	5.35E-84	1.47E-54	1.31E-44	2.29E-70	1.23E-65	6.26E-62
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	–	–	–	–	–	–	–	–
t-statistic	–	–	–	–	–	–	–	–
p-value	–	–	–	–	–	–	–	–
$p < 0.05$	–	–	–	–	–	–	–	–

use attributes outperform their equivalents with attributes. This suggests that attributes act as a natural form of regularisation or constraint on the learning process, preventing over-fitting to the given training data. Furthermore, we observe that our algorithm without attributes outperforms all other algorithms according all metrics. Additionally, by comparing HEADNet to HEADNet$_{\Sigma=\mathbb{I}}$, we find that training the variance matrix results in superior performance across all networks, compared with fixing it to the identity matrix.

4.3.4 Link Prediction

To evaluate link prediction ability, we randomly select 10% of the edges in the network and remove them (ensuring that every node has at least one connecting edge) (Bojchevski and Günnemann, 2018). We randomly select also an equal number of non-edges in the network. We then train each model on the incomplete network and rank the pairs of nodes based on similarity.

Table 4.5 provides a summary of the link prediction results. We see that incorporating attributes improves the performance on all three algorithms that are capable of doing so on three out of the four networks: G2G$_{K=1}$ vs. G2G$_{\text{NA},K=1}$, G2G$_{K=3}$ vs. G2G$_{\text{NA},K=3}$, and HEADNet vs. HEADNet$_{\text{NA}}$. The poorer performance on the Pubmed network perhaps suggests that the *homophily* property (that is, nodes with like attributes are more likely to connect) does not hold as strongly for this network and using purely topological measures can better predict links. In addition, we see that the two methods that embed to a hyperbolic space (HEADNet$_{\text{NA}}$ and HEADNet) obtain superior performance compared to all of the Euclidean alternatives.

Table 4.6 provides a comparison of mAP scores achieved in the link prediction task by HEADNet with and without fixing the variance matrix to the identity matrix. We find that in three out of the four networks, performance is improved when both the mean and variance matrices are learned simultaneously.

4.3.5 Link Prediction: Unseen Nodes

Mapping directly from attributes allows our model to handle previously unseen nodes. To evaluate the capacity for HEADNet to predict edges on unseen nodes, we

TABLE 4.5: Summary of the link prediction task on synthetic, Cora_ML, Citeseer, Pubmed, Cora and Wiki Vote networks. For clarity, we only report the results for an embedding dimension of 25+25, but obtain similar results for all dimensions (see appendix C.2). $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. A dash (–) indicates that a network did not have attributes. All measures are reported to 3 decimal places. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

Synthetic (∼173 edges removed)					Pubmed (8865 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	88.7(8.1)	0.493(0.041)	0.574(0.040)	0.017(0.020)	ATP (log)	7692.6(24.3)	0.132(0.003)	0.327(0.001)	0.083(0.005)
ATP (harmonic)	88.7(8.1)	0.493(0.041)	0.570(0.040)	0.017(0.020)	ATP (harmonic)	7682.0(24.9)	0.134(0.003)	0.330(0.001)	0.085(0.004)
LINE	122.0(8.2)	0.301(0.037)	0.383(0.012)	0.018(0.009)	LINE	4859.8(52.3)	0.452(0.006)	0.433(0.003)	0.045(0.002)
$\text{G2G}_{\text{NA},K=1}$	47.0(6.7)	0.733(0.045)	0.651(0.051)	0.010(0.005)	$\text{G2G}_{\text{NA},K=1}$	66.0(7.5)	0.993(0.001)	0.991(0.001)	0.503(0.012)
$\text{G2G}_{\text{NA},K=3}$	35.2(6.3)	0.801(0.040)	0.738(0.049)	0.023(0.010)	NK	107.6(6.2)	0.988(0.001)	0.987(0.001)	0.475(0.005)
$\text{G2G}_{K=1}$	–	–	–	–	$\text{G2G}_{K=1}$	284.1(19.9)	0.968(0.002)	0.958(0.003)	0.146(0.006)
$\text{G2G}_{K=3}$	–	–	–	–	$\text{G2G}_{K=3}$	229.6(10.4)	0.974(0.001)	0.965(0.002)	0.170(0.004)
Significance Test					Significance Test				
NK	34.6(9.8)	0.804(0.063)	0.738(0.072)	0.030(0.008)	$\text{G2G}_{\text{NA},K=3}$	39.2(4.9)	0.996(0.001)	0.995(0.001)	0.612(0.011)
$\text{HEADNet}_{\text{NA}}$	**22.3(4.0)**	**0.876(0.026)**	**0.866(0.032)**	**0.051(0.009)**	$\text{HEADNet}_{\text{NA}}$	**36.0(4.3)**	**0.996(0.000)**	**0.996(0.001)**	**0.781(0.009)**
t-statistic	6.40E+00	5.84E+00	8.95E+00	9.43E+00	*t*-statistic	2.69E+00	2.69E+00	3.74E+00	6.49E+01
p-value	7.68E-08	4.53E-07	1.95E-11	1.84E-13	*p*-value	4.68E-03	4.68E-03	2.10E-04	1.06E-53
$p < 0.05$	1	1	1	1	$p < 0.05$	1	1	1	1
HEADNet	–	–	–	–	HEADNet	73.4(6.8)	0.992(0.001)	0.989(0.001)	0.421(0.007)
t-statistic	–	–	–	–	*t*-statistic	-2.23E+01	-2.23E+01	-2.23E+01	-7.97E+01
p-value	–	–	–	–	*p*-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	–	–	–	–	$p < 0.05$	0	0	0	0
Cora_ML (842 edges removed)					Cora (6532 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	500.4(11.9)	0.407(0.014)	0.446(0.009)	0.031(0.004)	ATP (log)	3945.8(35.1)	0.396(0.005)	0.435(0.004)	0.038(0.001)
ATP (harmonic)	496.0(12.2)	0.412(0.014)	0.454(0.010)	0.032(0.004)	ATP (harmonic)	3920.1(35.2)	0.400(0.005)	0.440(0.004)	0.040(0.002)
LINE	405.2(11.8)	0.520(0.014)	0.475(0.008)	0.063(0.008)	LINE	3159.6(36.1)	0.516(0.006)	0.473(0.003)	0.084(0.004)
$\text{G2G}_{\text{NA},K=1}$	27.1(3.6)	0.969(0.004)	0.965(0.005)	0.240(0.013)	$\text{G2G}_{\text{NA},K=1}$	74.8(6.6)	0.989(0.001)	0.989(0.001)	0.456(0.012)
$\text{G2G}_{\text{NA},K=3}$	26.6(3.6)	0.970(0.004)	0.966(0.006)	0.244(0.014)	$\text{G2G}_{\text{NA},K=3}$	70.9(7.5)	0.989(0.001)	0.990(0.001)	0.485(0.012)
NK	29.6(3.7)	0.966(0.004)	0.965(0.005)	0.272(0.015)	NK	109.4(7.5)	0.983(0.001)	0.984(0.001)	0.479(0.005)
$\text{G2G}_{K=3}$	24.2(3.0)	0.972(0.004)	0.967(0.006)	0.244(0.014)	$\text{G2G}_{K=1}$	66.5(5.3)	0.990(0.001)	0.989(0.001)	0.426(0.012)
Significance Test					Significance Test				
$\text{G2G}_{K=1}$	23.8(2.9)	0.973(0.003)	0.967(0.006)	0.241(0.012)	$\text{G2G}_{K=3}$	64.1(5.2)	0.990(0.001)	0.990(0.001)	0.445(0.012)
$\text{HEADNet}_{\text{NA}}$	24.4(3.9)	0.972(0.005)	**0.974(0.004)**	**0.394(0.016)**	$\text{HEADNet}_{\text{NA}}$	86.6(5.9)	0.987(0.001)	0.989(0.001)	**0.623(0.008)**
t-statistic	-5.92E-01	-5.92E-01	5.08E+00	4.30E+01	*t*-statistic	-1.57E+01	-1.57E+01	-3.54E+00	6.90E+01
p-value	7.22E-01	7.22E-01	2.69E-06	3.16E-43	*p*-value	1.00E+00	1.00E+00	1.00E+00	2.48E-51
$p < 0.05$	0	0	1	1	$p < 0.05$	0	0	0	1
HEADNet	**16.9(2.6)**	**0.981(0.003)**	**0.980(0.004)**	**0.415(0.017)**	HEADNet	**46.1(4.3)**	**0.993(0.001)**	**0.993(0.001)**	**0.661(0.006)**
t-statistic	9.69E+00	9.69E+00	1.01E+01	4.60E+01	*t*-statistic	1.48E+01	1.48E+01	1.56E+01	8.96E+01
p-value	5.41E-14	5.41E-14	2.59E-14	2.98E-43	*p*-value	3.01E-21	3.01E-21	2.35E-21	3.92E-50
$p < 0.05$	1	1	1	1	$p < 0.05$	1	1	1	1
Citeseer (536 edges removed)					Wiki Vote (10369 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	303.9(11.2)	0.435(0.021)	0.494(0.021)	0.029(0.004)	ATP (log)	5768.5(37.5)	0.444(0.004)	0.471(0.003)	0.013(0.001)
ATP (harmonic)	302.2(11.3)	0.438(0.021)	0.494(0.021)	0.028(0.004)	ATP (harmonic)	5768.1(36.1)	0.444(0.003)	0.471(0.003)	0.013(0.001)
LINE	299.9(7.3)	0.442(0.014)	0.433(0.007)	0.040(0.005)	LINE	7171.1(48.1)	0.309(0.005)	0.376(0.002)	0.042(0.002)
$\text{G2G}_{\text{NA},K=1}$	28.8(4.6)	0.948(0.008)	0.962(0.007)	0.279(0.015)	$\text{G2G}_{\text{NA},K=1}$	478.7(32.7)	0.954(0.003)	0.941(0.004)	0.109(0.005)
$\text{G2G}_{\text{NA},K=3}$	28.5(4.6)	0.949(0.009)	0.962(0.007)	0.264(0.013)	$\text{G2G}_{\text{NA},K=3}$	476.3(33.7)	0.954(0.003)	0.932(0.006)	0.114(0.004)
NK	39.1(6.2)	0.929(0.011)	0.945(0.007)	0.265(0.012)	$\text{G2G}_{K=1}$	–	–	–	–
$\text{G2G}_{K=1}$	15.0(3.0)	0.974(0.006)	0.972(0.009)	0.259(0.012)	$\text{G2G}_{K=3}$	–	–	–	–
Significance Test					Significance Test				
$\text{G2G}_{K=3}$	14.9(2.8)	0.974(0.005)	0.972(0.008)	0.247(0.012)	NK	199.1(7.8)	0.981(0.001)	0.979(0.001)	0.179(0.004)
$\text{HEADNet}_{\text{NA}}$	35.0(5.2)	0.937(0.010)	0.951(0.007)	**0.370(0.026)**	$\text{HEADNet}_{\text{NA}}$	**121.6(8.6)**	**0.988(0.001)**	**0.984(0.001)**	**0.218(0.004)**
t-statistic	-1.87E+01	-1.87E+01	-1.09E+01	2.32E+01	*t*-statistic	3.66E+01	3.66E+01	1.65E+01	3.52E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.18E-25	*p*-value	7.88E-42	7.88E-42	3.25E-23	9.48E-41
$p < 0.05$	0	0	0	1	$p < 0.05$	1	1	1	1
HEADNet	**12.8(2.1)**	**0.978(0.004)**	0.975(0.007)	**0.414(0.020)**	HEADNet	–	–	–	–
t-statistic	3.19E+00	3.19E+00	1.30E+00	3.92E+01	*t*-statistic	–	–	–	–
p-value	1.21E-03	1.21E-03	9.95E-02	6.18E-39	*p*-value	–	–	–	–
$p < 0.05$	1	1	0	1	$p < 0.05$	–	–	–	–

TABLE 4.6: Comparison of HEADNet as a recommender system with and without learning the variance matrix. We present mAP achieved on the link prediction task. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

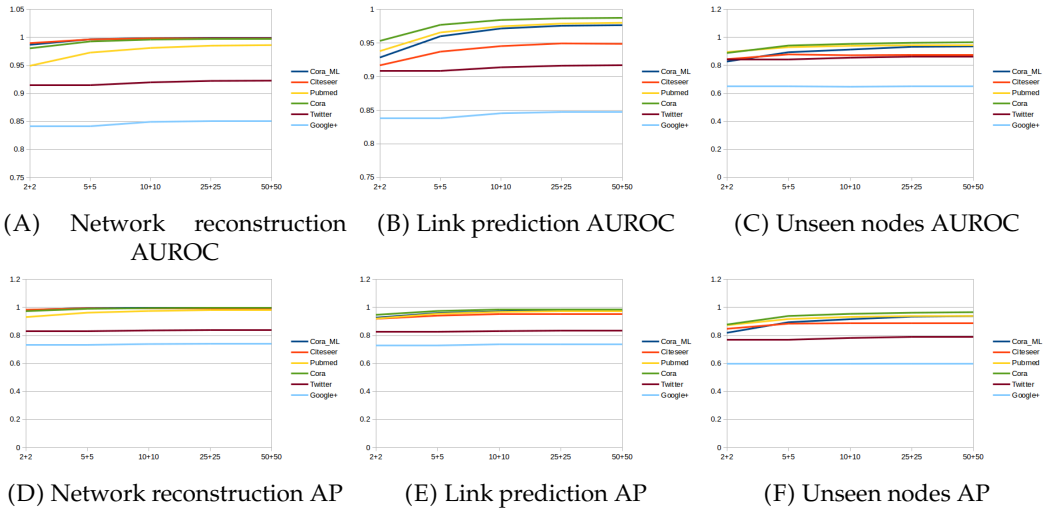
		Embedding Dimension			
		5	10	25	50
Cora_ML	HEADNet$_{\Sigma=\mathbb{I}}$	0.277(0.019)	0.297(0.016)	0.302(0.015)	0.306(0.017)
	HEADNet	**0.341(0.022)**	**0.395(0.020)**	**0.415(0.017)**	**0.428(0.023)**
	t-statistic	1.21E+01	2.08E+01	2.68E+01	2.35E+01
	p-value	1.41E-17	2.84E-28	2.36E-34	2.37E-30
	$p < 0.05$	1	1	1	1
Citeseer	HEADNet$_{\Sigma=\mathbb{I}}$	0.274(0.019)	0.291(0.016)	0.301(0.014)	0.311(0.017)
	HEADNet	**0.332(0.023)**	**0.392(0.026)**	**0.414(0.020)**	**0.424(0.017)**
	t-statistic	1.09E+01	1.77E+01	2.56E+01	2.56E+01
	p-value	1.13E-15	4.23E-23	1.80E-31	1.55E-33
	$p < 0.05$	1	1	1	1
Pubmed	HEADNet$_{\Sigma=\mathbb{I}}$	0.317(0.007)	0.389(0.007)	0.436(0.008)	0.441(0.007)
	HEADNet	0.302(0.007)	0.374(0.008)	0.421(0.007)	0.436(0.005)
	t-statistic	-8.07E+00	-7.40E+00	-8.41E+00	-3.57E+00
	p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
	$p < 0.05$	0	0	0	0
Cora	HEADNet$_{\Sigma=\mathbb{I}}$	0.484(0.005)	0.558(0.006)	0.575(0.006)	0.578(0.008)
	HEADNet	**0.540(0.010)**	**0.625(0.006)**	**0.661(0.006)**	**0.664(0.007)**
	t-statistic	2.80E+01	4.43E+01	4.46E+01	4.46E+01
	p-value	1.58E-29	1.58E-46	2.31E-46	2.31E-46
	$p < 0.05$	1	1	1	1

devise the following experiment. We randomly sample 10% nodes from the network and remove all in- and out- going edges from each of these selected nodes (this may result in the removal of additional nodes if we remove all of a nodes neighbours). For each removed edge, we randomly select a pair of nodes as a negative sample. We then train and evaluate in the same way as in the link prediction experiment. We compare only against G2G as only G2G can handle unseen nodes.

Table 4.7 shows a summary of the unseen nodes experiment for embedding dimensions 5+5 and 10+10. From this, we see that HEADNet can achieve substantially better results than G2G – outperforming it on all networks by all metrics – in relatively small embedding dimensions. These results suggest that embedding to a hyperbolic metric space using attributes can provide an effective method for predicting the links of previously unseen nodes.

TABLE 4.7: Summary of link prediction on unseen nodes. For clarity, we only report the results for an embedding dimension of 5+5 and 10+10, but obtain similar results for all dimensions (see appendix C.3). All measures are reported to 3 decimal places with the exception of mean rank which is to 1 decimal place. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the t-statistic, we select the benchmark algorithm according to AP. The selected benchmark algorithm is identified with an asterisk (*). Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Cora_ML (300 nodes removed)				Pubmed (1972 nodes removed)			
		Mean Rank	AUROC	AP		Mean Rank	AUROC	AP
5+5	*G2G$_{K=1}$	427.9(67.1)	0.739(0.027)	0.683(0.028)	*G2G$_{K=1}$	2772.4(245.4)	0.836(0.011)	0.814(0.012)
	G2G$_{K=3}$	618.5(78.9)	0.621(0.040)	0.572(0.037)	G2G$_{K=3}$	3914.7(364.3)	0.768(0.019)	0.753(0.016)
	HEADNet	**255.5(44.4)**	**0.845(0.018)**	**0.856(0.016)**	HEADNet	**2043.8(144.9)**	**0.879(0.007)**	**0.873(0.007)**
	t-statistic	11.74	18.04	29.44	t-statistic	14	18.17	23.77
	p-value	2.497E-16	4.429E-24	5.682E-32	p-value	1.069E-18	1.987E-23	4.54E-28
	$p < 0.05$	1	1	1	$p < 0.05$	1	1	1
	Citeseer (423 nodes removed)				Cora (1980 nodes removed)			
		Mean Rank	AUROC	AP		Mean Rank	AUROC	AP
	*G2G$_{K=1}$	227.3(24.3)	0.774(0.022)	0.792(0.020)	*G2G$_{K=1}$	2465.9(262.6)	0.801(0.019)	0.766(0.023)
	G2G$_{K=3}$	232.9(25.9)	0.768(0.024)	0.786(0.021)	G2G$_{K=3}$	3545.1(376.8)	0.714(0.030)	0.669(0.033)
	HEADNet	**166.7(23.1)**	**0.835(0.016)**	**0.852(0.015)**	HEADNet	**1088.8(74.6)**	**0.912(0.005)**	**0.920(0.004)**
	t-statistic	9.903	12.18	13.24	t-statistic	27.63	30.41	36.42
	p-value	2.285E-14	3.059E-17	7.183E-19	p-value	5.196E-25	5.138E-26	2.839E-27
	$p < 0.05$	1	1	1	$p < 0.05$	1	1	1
	Cora_ML (300 nodes removed)				Pubmed (1972 nodes removed)			
		Mean Rank	AUROC	AP		Mean Rank	AUROC	AP
10+10	*G2G$_{K=1}$	428.5(74.4)	0.739(0.032)	0.682(0.033)	*G2G$_{K=1}$	2766.2(215.8)	0.836(0.011)	0.811(0.011)
	G2G$_{K=3}$	636.2(74.2)	0.610(0.037)	0.561(0.033)	G2G$_{K=3}$	3878.4(375.2)	0.770(0.018)	0.754(0.017)
	HEADNet	**207.6(39.9)**	**0.874(0.016)**	**0.884(0.015)**	HEADNet	**1681.7(100.1)**	**0.900(0.005)**	**0.897(0.004)**
	t-statistic	14.34	20.66	30.38	t-statistic	24.97	29.98	39.99
	p-value	1.431E-18	1.958E-24	2.442E-29	p-value	1.102E-26	2.446E-29	5.557E-33
	$p < 0.05$	1	1	1	$p < 0.05$	1	1	1
	Citeseer (423 nodes removed)				Cora (1980 nodes removed)			
		Mean Rank	AUROC	AP		Mean Rank	AUROC	AP
	*G2G$_{K=1}$	224.8(26.7)	0.776(0.026)	0.793(0.024)	*G2G$_{K=1}$	2464.4(226.4)	0.801(0.017)	0.764(0.022)
	G2G$_{K=3}$	232.9(29.2)	0.768(0.026)	0.786(0.024)	G2G$_{K=3}$	3559.7(348.6)	0.713(0.025)	0.665(0.027)
	HEADNet	**162.0(19.7)**	**0.840(0.015)**	**0.857(0.013)**	HEADNet	**693.3(70.5)**	**0.944(0.005)**	**0.949(0.004)**
	t-statistic	10.36	11.79	13.24	t-statistic	40.9	43.38	46.05
	p-value	1.084E-14	8.412E-16	2.365E-17	p-value	3.584E-31	4.407E-31	4.548E-30
	$p < 0.05$	1	1	1	$p < 0.05$	1	1	1



(A) Network reconstruction AUROC (B) Link prediction AUROC (C) Unseen nodes AUROC

(D) Network reconstruction AP (E) Link prediction AP (F) Unseen nodes AP

FIGURE 4.7: Summary of AUROC scores over embedding dimensions 2+2, 5+5, 10+10, 25+25, and 50+50 for three downstream machine learning tasks. Network reconstruction (A) AUROC and (D) AP; link prediction (B) AUROC and (E) AP; unseen nodes (C) AUROC and (F) AP. All measures are averaged over 30 random seeds.

4.3.6 Parameter Sensitivity

Here we evaluate the robustness of HEADNet to the setting of embedding dimension. Figure 4.7 shows AUROC scores on three downstream machine learning tasks: network reconstruction, link prediction, and predicting links on unseen nodes over a range of embedding dimensions: 2+2, 5+5, 10+10, 25+25, and 50+50. We see from the almost flat curves that HEADNet is robust to the setting of embedding dimension, and even achieves good performance the 2+2 dimensional setting.

4.4 Chapter Summary

This chapter presents HEADNet to fill the gap of embedding directed networks in hyperbolic space. We propose an embedding model to map attributed nodes to Gaussian distributions in hyperbolic space. We further propose the use of a mapping procedure to map hyperbolic Gaussian distributions to Euclidean ones, preserving hyperbolic distance and direction, based on previous works (Nagano et al., 2019). We achieve state-of-the art performance on a number of downstream machine learning tasks, including predicting neighbours of previously unseen nodes. Our results show that HEADNet provides a general hyperbolic embedding method for directed

networks with and without node attributes, which opens the door to hyperbolic manifold learning on a wider range of network than previously possible.

Chapter 5

Hierarchical Organisation of Network Architectures

Literature has revealed that dynamic network behaviour is governed by meso-scopic architectural features (see section 1.4). Meso-scopic features are arranged hierarchically (Kim et al., 2012; García-Pérez, Boguñá, and Serrano, 2018). As discussed in section 2.3, local network architectures generate local behaviour, for example: feedback loops generate bi-stability and oscillation. However, it is the multi-scale, hierarchical organisation of these local architectures that give rise to larger network architectures, such as the bow-tie architecture, and that generates the complex network dynamics that allow for critical network behaviour, necessary for a cell's survival and reproduction (Roli et al., 2018).

The purpose of this chapter is to investigate this multi-scale arrangement of architectures in intra-cellular networks – focusing specifically on the organisation of coherent feedback loops within the cores of bow-tie architectures and feed-forward loops found within in- and out-components. Furthermore, we investigate the relationship of local dynamical behaviour to global network behaviour. Section 5.1 introduces our hypotheses that serve as the motivation for our approach. Section 5.2 describes an overview of our algorithm and provides a simple method for identifying the components in a bow-tie architecture, based on the definition of such an architecture given in section 1.4.5. Next, section 5.2.2 provides a framework to identify the position of nodes within a hierarchy of coherent feedback loops found within the cores of bow-tie architectures based on an iterative subnetwork merging algorithm. By adapting the algorithm introduced in section 5.2.2, section 5.2.4 can

decompose in- and out-components of bow-tie architectures into a hierarchy of feed-forward loops. In section 5.3 we validate the decomposition of the core produced by the algorithm described in section 5.2.2. We describe our experimental setup and benchmark datasets, and, in section 5.4, we evaluate our approach on a number of synthetic and real-world dynamic network models derived from literature. Finally, section 5.5 summarises the chapter.

5.1 Introduction

Complex networks are models of real-world systems. Real-world systems are always changing and so complex networks must be dynamic. This may refer to the changing of a network's structure, its internal state, or both. An understanding of the dynamical behaviour of a complex system is essential to understand the system as a whole, and, as such, the study of the dynamics of complex networks have attracted much attention in a variety of disciplines, such as social network analyses (Sarkar and Moore, 2006; Braha and Bar-Yam, 2009), multi-agent systems (Jiang and Wang, 2010), and biological system modelling (Albert and Othmer, 2003; Kim, Park, and Cho, 2013).

To understand dynamics over the nodes of a complex network, both topological and interaction (collectively called architectural) information is required (Liu, Slotine, and Barabási, 2011). Small-scale network architectures generate rich local network dynamics, such as bi-stability, and oscillation (Timár et al., 2017). Positive *coherent* feedback loops[1], in particular, amplify signal, whereas negative coherent feedback loops attenuate it (Kholodenko, 2006). Feedback loops, in general, form a subset of network motifs that have been found to be enriched in a diverse array of organisms, including *Escherichia coli* (E. coli) (Shen-Orr et al., 2002), *Saccharomyces cerevisiae* (budding yeast) (Milo et al., 2002), and human (Odom et al., 2004).

However, localized dynamics are not enough to explain the complex, global behaviours of real-world systems. To survive and reproduce in the uncertain and dynamic environment, intra-cellular networks must evolve such that can process noisy and variable information, and this requires the following functional characteristics:

[1]Recall the definition of coherent feedback loops given in section 1.1.7: a coherent feedback loop is a cycle in a directed and signed network where each edge has the same sign.

robustness and *flexibility*. Striking the balance between these two seemingly conflicting functional properties is the key for a cell to successfully process uncertain and variable information. An optimal balance enables molecular networks to attain the highest level of information processing capabilities (Roli et al., 2018). The balance cannot be realised by simple network architectures with simple dynamics such as bi-stability or oscillations. Instead, more complex network architectures are required to generate the necessary global networks dynamics to strike a balance between robustness and flexibility.

A ubiquitous higher-order network architecture is the bow-tie architecture – comprised of weakly connected in- and out-components connected to a strongly-connected core. These cores have been shown to compress signals into a small set of universal, modular building blocks (Timár et al., 2017), as well as gives rise to the necessary critical behaviour of the network as a whole (Mitarai, Jensen, and Semsey, 2015). By definition, any two nodes in the core are strongly connected, which means they belong to the same feedback loop. This definition, therefore, suggests that the core contains at least one feedback loop or multiple nested feedback loops. As discussed previously, these feedback loops generate diverse local network dynamics. When multiple positive and negative feedback loops are coupled, the core generates complex dynamics, which enables the molecular networks to make robust yet flexible decisions, producing increasingly diverse and complex cellular behaviour (Kim, Yoon, and Cho, 2008; Mitarai, Jensen, and Semsey, 2015). This is why nested feedback loops have been found in many biological networks (Kim et al., 2012).

In addition to the strongly connected core, the in- and out-components meso-scale architectural components that are responsible for signal de-noising and filtering respectively (Timár et al., 2017). These components are typically acyclic networks that are, accordingly, enriched by acyclic network motifs such as cascades and feed-forward loops. Feed-forward loops, particular, are local architectures that have been shown to produce small-scale dynamical behaviour, such as: persistent signal detection (Mangan and Alon, 2003; Chou, 2018), fold-change detection (Goentoro et al., 2009) and pulse shaping (Mangan and Alon, 2003).

However, the link between the scales of network architectures – from small, localized coherent feedback loops to larger, higher-order processing cores to overall

global network dynamical behaviour – remains largely unexplored. We are left with the following question: can we predict and control global complex network dynamics based on incomplete information about local network architecture? As loops are small-scale, local architectures that are found within the components of larger bow-tie architectures, we accordingly break down our question into the following two parts:

1. how do local dynamics organize as global dynamics?
2. can we use known information about the relationship between local architectures and local dynamics to predict and control global dynamics? This requires understanding of both:
   (a) how small-scale architectures are organized as larger architectures,
   (b) and how this organisation correlated to the organisation of local dynamics in global dynamics.

Since completely capturing and modelling interactions requires intimate knowledge of the parameters of the system (for example: reaction rates), which are often unknown or infeasible to determine, we consider the case when only minimal architectural information is known – that is, only the direction and signs of interactions are known.

5.1.1 Hypotheses

It has been shown that coherent feedback loops form the information processing building blocks of complex biological networks (Milo et al., 2002). For example, positive feedback loops amplify and negative feedback loops attenuate signal, which are two of the most fundamental information processing steps. Based on these observations, we further hypothesise that coherent feedback loops are arranged hierarchically within the cores of bow-tie architectures and that it is this multi-scale organisation of architecture that, ultimately, determine how information is processed and governs overall global network dynamics (Supper et al., 2009). Furthermore, we hypothesise that it is feed-forward loops, not feedback loops, that are arranged within the in- and out-components.

Our claim is that nodes at the deepest levels of the uncovered loop hierarchy govern the dynamics of the network as a whole, that we divide into two parts:

1. The deeper that a node appears in the coherent feedback loop hierarchy, the more likely it is to have a significant impact on the output of the network,

2. and, the better candidate that it is to be a therapeutic target.

In the core, we focus on coherent loops because:

- positive feedback loops amplify signal (Kholodenko, 2006), and so, we hypothesise that any change will be accordingly amplified through the hierarchy;

- and negative feedback loops are responsible for reducing signal (Kholodenko, 2006) as well as for network adaptation (Ma et al., 2009), and so, control the robustness of the global dynamics through the feedback loop hierarchy.

Likewise for the in- and out-components, we consider feed-forward loops since they are responsible for noise filtering, fold-change detection and the identification of persistent signals (Mangan and Alon, 2003; Goentoro et al., 2009; Chou, 2018). Additionally, (incoherent) feed-forward loops have been shown the be responsible for pulse shaping (Mangan and Alon, 2003).

To test our hypotheses, we propose IMPLISig (for Iterative Merging of Positive/negative Feedback Loops in Signalling networks), a general framework that breaks down the components of the bow-tie architectures with incomplete interaction information into a hierarchy of coherent feedback loops. This enables us to validate our intuition that the hierarchy of coherent feedback loops in the core determines how information is processed and governs overall global network dynamics.

5.2 Decomposition of Bow Tie Architectures

The following three sections detail the entire decomposition procedure. It can be broadly divided into the following three stages:

1. Identification of the three components (the core, in- and out-components) that form the bow-tie architecture (described in section 5.2.1)

2. Decomposition of the core into a hierarchy of coherent feedback loops (described in section 5.2.2)

3. Decomposition of the in- and out-components into a hierarchy of feed-forward loops (described in section 5.2.4).

5.2.1 Identification of the Bow-Tie Components

Identification of a bow-tie architecture within a signalling network $\mathbb{G} = (V, E)$ is based on the definition of a bow-tie architecture given in section 1.4.5.

Identifying the Core

We begin with the identification of the core. Recall that the core of the bow-tie architecture is a large strongly connected component that forms the processing centre of the system (Timár et al., 2017). Identification of the core requires the simple application of Tarjan's algorithm (Tarjan, 1972). We proceed based on the assumption that a signalling network under study has exactly one large strongly connected component. Note that a general directed network $\mathbb{G}$ containing n strongly connected components can be broken down into n disjoint strongly connected subnetworks: $\mathbb{G} = (\mathcal{C}_1, ..., \mathcal{C}_n)$ where $\bigcup_i \mathcal{C}_i = \mathbb{G}$ and $V_{\mathcal{C}_i} \cap V_{\mathcal{C}_j} = \emptyset$ for all $i \neq j$. Our approach can be applied to each of those subnetworks $\mathcal{C}_i$ in parallel.

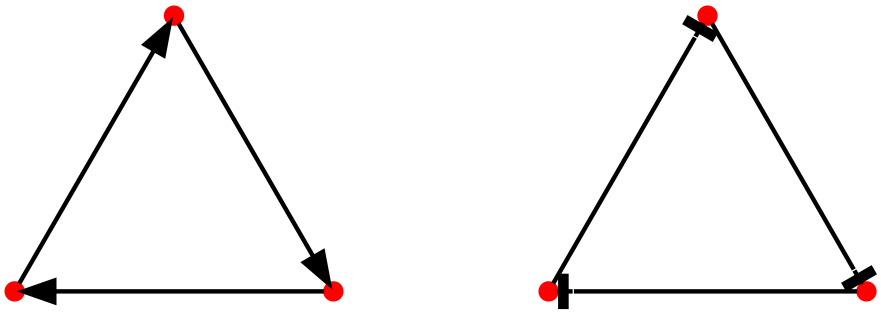
Identifying the In- and Out-Components

For a given core, $\mathcal{C} = (V_{\mathcal{C}}, E_{\mathcal{C}}) \subseteq \mathbb{G}$, identification of the in- and out-components is a trivial task. The in and out components are defined based on reachability[2] to all of the nodes in the core. Since the core is strongly connected, it suffices to check reachability to (and from) just one node $c \in V_{\mathcal{C}}$ in the core. Based on this, we can identify the two peripheral components (the in-component $\mathcal{I}$ and the out-component $\mathcal{O}$) as follows:

$$\mathcal{I} := \{v \in V \setminus \mathcal{C} \mid \textsc{hasPath}(\mathbb{G}, v, c)\} \tag{5.1}$$

$$\mathcal{O} := \{v \in V \setminus \mathcal{C} \mid \textsc{hasPath}(\mathbb{G}, c, v)\} \tag{5.2}$$

[2]Recall the definition of reachability given in section 1.1.7.



(A) A three-node positive coherent feedback loop. (B) A three-node negative coherent feedback loop.

FIGURE 5.1: Two examples of coherent feedback loops. (A) shows a positive feedback loop, which has been shown to amplify a signal, whereas (B) shows a negative feedback loop, which reduces signal (Kholodenko, 2006). For nodes in the core of bow-tie architecture, we suggest that both make for promising targets for network control.

where HASPATH$(\mathbb{G}, u, v)$ is a boolean function that returns true if network $\mathbb{G}$ has a path from node u to node v. It may be that one or both of the sets $\mathcal{I}$ and $\mathcal{O}$ is empty.

5.2.2 Decomposition of the Core

Intuition

Literature has shown that feedback loops form the building blocks of complex biological systems (Milo et al., 2002). As we highlighted in section 2.3.2, coherent feedback loops are responsible for a number of fundamental signalling operations in the core of bow-tie architectures. Accordingly, the decomposition of the core is built upon the identification of highly modular coherent feedback loops. Figure 5.1 provides some examples of small coherent feedback loops, the occurrence of both of which are enriched in many biological networks (Kholodenko, 2006).

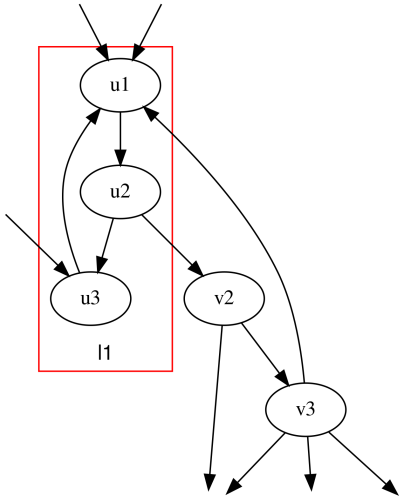
Our algorithm, IMPLISig, decomposes the core of bow-tie architectures based upon two principles:

1. all directed feedback loops are – trivially – strongly connected,

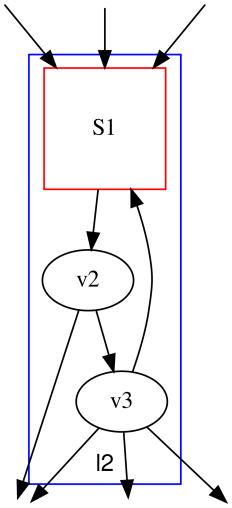
2. feedback loops of feedback loops are also strongly connected.

The second principle can be understood in the following way: Consider a strongly connected network $\mathbb{S}_0$ of N nodes. A single loop $l_1 = (u_1, ..., u_n)$[3] in that network, containing $n > 1$ nodes, may, instead, be represented as a single supernode S_1. Here 'represented' means that edges incident to the members of l_1 are

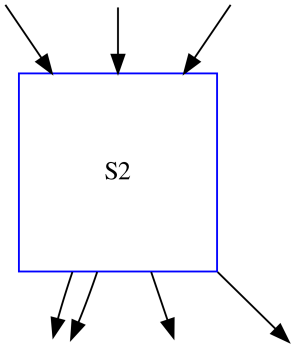
[3]Here we use brackets $(\cdot)$ to denote an ordered set of nodes that represents a loop in the network.



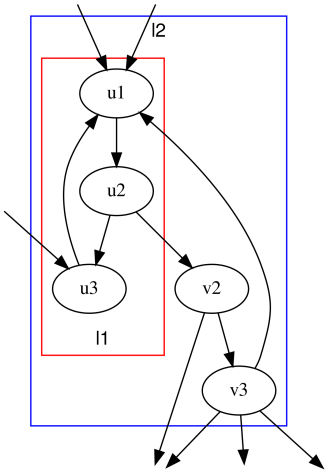
(A) Coherent feedback loop in $\mathbb{G}_0$.



(B) Coherent feedback loop in $\mathbb{G}_1$.



(C) A super-node in $\mathbb{S}_2$.



(D) An expanded super-node.

FIGURE 5.2: The intuition behind IMPLISig. (A) shows a five node strongly connected component within a larger strongly connected network $\mathbb{S}_0$ containing N nodes. A coherent feedback loop, $\mathrm{l1} = (\mathrm{u1}, \mathrm{u2}, \mathrm{u3})$, is highlighted in red. In (B), the nodes in l1 have been replaced with a single super-node, producing a condensed network $\mathbb{S}_1$ containing $N - 3 + 1 = N - 2$ nodes. A new coherent feedback loop, $\mathrm{l2} = (\mathrm{S1}, \mathrm{v2}, \mathrm{v3})$, is highlighted in blue. (C) shows the nodes in l2 condensed into a super-node, S2, forming the condensed network $\mathbb{S}_2$. (D) provides an expanded view of S2, where both l2 and l1 are expanded to show the strongly connected component in $\mathbb{S}_0$ that the node S2 in $\mathbb{S}_2$ represents. This demonstrates the hierarchy of loops that are uncovered by IMPLISig.

made appropriately incident to S_1: edges incoming to members of l_1 become edges incoming to S_1, and likewise out-going edges are made out-going. All of the nodes in l_1 are then deleted from the network. We can interpret this new version of the network as a smaller, 'condensed' network $\mathbb{S}_1$, that contains $N - n + 1 < N$ nodes. Principle 1 tells us that the original loop was strongly connected and so the new super-node represents a strongly connected subnetwork in the original network $\mathbb{S}_0$.

If a new loop $l_2 = (v_1 = S_1, v_2, ..., v_m)$ of size $m > 1$ – containing that super-node S_1 – is identified in the condensed network $\mathbb{S}_1$, then that loop l_2 may be replaced with a new super-node S_2 – forming an even more condensed network $\mathbb{S}_2$ that contains even fewer nodes than $\mathbb{S}_1$. This new super-node S_2 represents a larger underlying strongly connected component in the original network $\mathbb{S}_0$, containing all of $\{u_1, ..., u_n, v_2, ..., v_m\}$.

The strongly connected property is guaranteed. The nodes in l_1 represent a loop in the original network and so all of $\{u_1, ..., u_n\}$ are strongly connected. Since l_2 is a loop in the condensed network $\mathbb{S}_1$, then a cycle must exist in the original network of this form: $(u_i, v_2, ..., v_m, u_j)$, where $u_i, u_j \in l_1$ and all of $v_2, ..., v_m \in l_2$. Now $u_i, u_j \in l_1$ and so a path exists from u_j to u_i making a cycle that contains all of $\{v_2, ..., v_m\}$. As such, we conclude that S_2 represents a strongly connected component in $\mathbb{S}_0$ containing all of $\{u_1, ..., u_n, v_2, ..., v_m\}$.

Figure 5.2 provides an illustrative example of this, where we show a strongly connected subnetwork of a larger network containing five nodes.

Naturally, this process can be iterated repeatedly, producing a series of T more-and-more condensed networks $\mathbb{S}_0, \mathbb{S}_1, ..., \mathbb{S}_T$ where $|\mathbb{S}_T| = 1$. That is, $\mathbb{S}_T$ contains a single super node that represents the entire original strongly connected network $\mathbb{S}_0$ in a hierarchy of feedback loops. This is principle that IMPLISig is built upon.

IMPLISig Overview

As alluded to in the previous section, the decomposition of the core consists of iterating over the following three steps until a given core SCC $\mathcal{C} = (V_\mathcal{C}, E_\mathcal{C})$ contains only one node:

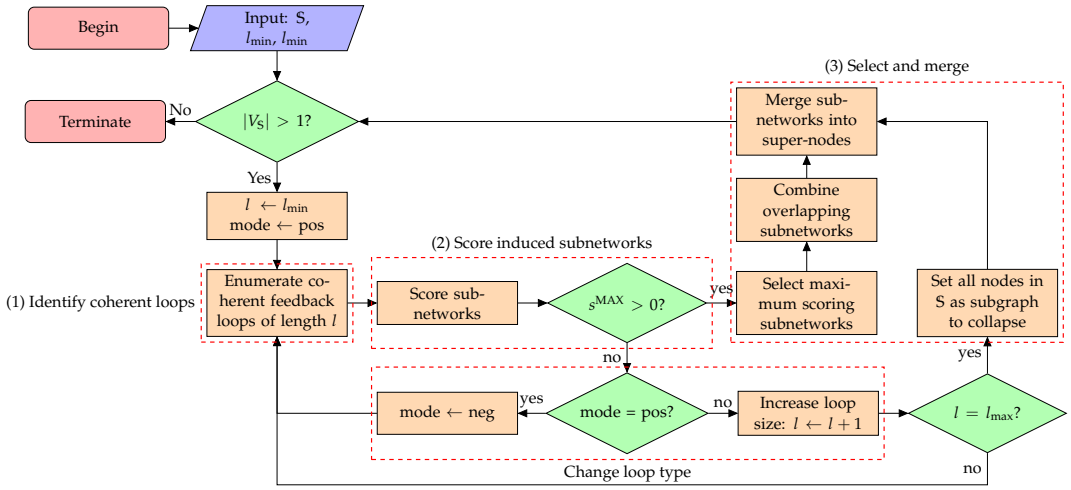


FIGURE 5.3: High-level view of IMPLISig algorithm. IMPLISig takes as input a strongly connected component S, as well as a two integer parameters, l_{min} and l_{max}, that set the minimum and maximum loop sizes to consider. IMPLISig terminates when the entire SCC has been collapsed into a single node. It can be broadly broken down into three parts: (1) identify coherent feedback loops, (2) score subnetworks induced by those feedback loops, and (3) select and merge subnetwork(s).

1. Identify all coherent feedback loops of length n. Our procedure for loop identification is a depth first search guided by the signs of edges. It is described in detail in section 5.2.2.

2. Score the subnetwork induced by each feedback loop. The score function is based on the modularity of the induced subnetwork – rewarding subnetwork with a high ratio of internal edges to total edges. We informally think of the internal edges as "pulling" the subnetwork into a super-node where the external edges pull the nodes apart. See section 5.2.2 for details on the score function.

3. Select the feedback loop(s) for merging. Feedback loops are selected based on the score computed in the previous step. In the case of multiple best-scoring subnetworks, we do not arbitrarily break ties and select all of them. In the case that a node appears in multiple best-scoring subnetworks, we merge those loops into a single subnetwork. The procedures for selecting loops for collapse and handling overlaps are detailed in section 5.2.2.

The entire IMPLISig algorithm is described in algorithm 1. A high level overview is provided in figure figure 5.3.

Algorithm 1 takes as input a strongly connected component $\mathcal{C}$ as well as two additional integer control parameters l_{min} and l_{max}, the smallest and largest size loop

Algorithm 1 IMPLISig algorithm: Iteratively collapse a strongly connected component $\mathcal{C} = (V_{\mathcal{C}}, E_{\mathcal{C}})$ into super-nodes.

function IMPLISIG($\mathcal{C}$, $l_{\min}$, $l_{\max}$)
  $\mathbb{H} \leftarrow$ Graph()
  while $|\mathcal{C}| > 1$ **do**
    foundLoop $\leftarrow$ False
    for l in $[l_{\min}, l_{\max}]$ **do**
      for mode $\in$ { pos, neg } **do**
        for $u \in V_{\mathcal{C}}$ **do**
          for $\mathbb{S}'$ in FINDCYCLES($\mathcal{C}, u, l, \emptyset, u$, mode) **do**
            scores[$\mathbb{S}'$] $\leftarrow$ SCORE($\mathbb{S}'$)
          end for
        end for
        $s^{\text{MAX}} \leftarrow$ MAX(scores) ▷ Determine maximum score
        if $s^{\text{MAX}} > 0$ **then**
          foundLoop $\leftarrow$ True
          $\mathcal{S}^{\text{MAX}} \leftarrow \{\mathbb{S}' \mid \text{scores}[\mathbb{S}'] = s^{\text{MAX}}\}$
          break
        end if
      end for
      if foundLoop **then**
        break
      end if
    end for
    if $\neg$ foundLoop **then**
      ▷ No coherent feedback loop of size $\leq l_{\max}$
      $\mathcal{S}^{\text{MAX}} \leftarrow \{\mathcal{C}\}$ ▷ Combine $\mathbb{S}$ into single node
    end if
    COMBINEOVERLAPS($\mathcal{S}^{\text{MAX}}$)
    for $\mathbb{S}^{\text{MAX}} \in \mathcal{S}^{\text{MAX}}$ **do**
      COLLAPSEINTOSUPERNODE($\mathcal{C}$, $\mathbb{H}$, $\mathbb{S}^{\text{MAX}}$)
    end for
  end while
  return $\mathbb{H}$ ▷ Node hierarchy is returned
end function

to search for before giving up respectively. If $l_{\max}$ has been reached and still no positive scoring subnetwork has been found, then there are no induced strongly connected components of size less than or equal to $l_{\max}$ within what remains of $\mathcal{C}$. In this situation, we collapse all of $\mathcal{C}$ into a single super-node and terminate.

Algorithm 1 returns $\mathbb{H} = (V_{\mathbb{H}}, E_{\mathbb{H}})$, a directed acyclic graph (DAG), that shows the hierarchy of the nodes and super-nodes in $\mathbb{S}$. The leaves of $\mathbb{H}$ are the nodes in $\mathcal{C}$. The root node of $\mathbb{H}$ is a single super-node that represents the entirety of $\mathcal{C}$. Edges in $\mathbb{H}$ show the formation of super-nodes. An edge $(u, v) \in E_{\mathbb{H}}$ shows that node u forms part of super-node v. Of course, u may, itself, be a super-node. Figure 5.4

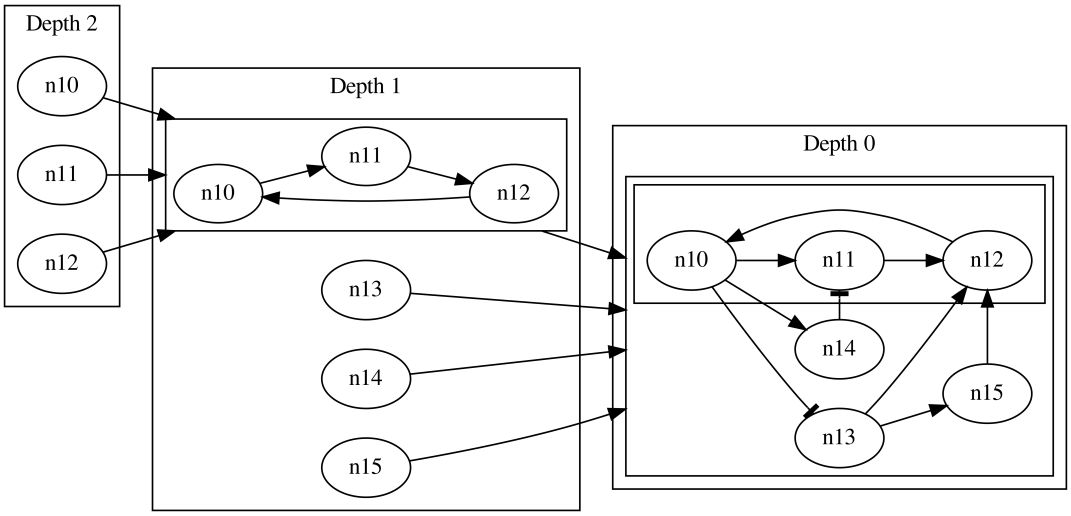


FIGURE 5.4: Example SCC hierarchy $\mathbb{H}$. The leaves of $\mathbb{H}$: n10, n11, n12, n13, n14 and n15 represent the six nodes in the original SCC. The root node (drawn at the right of the figure) represents the entire SCC merged together. The intermediary node represents the positive feedback loop comprised of nodes n10, n11 and n12. The directed edges show node membership in super-nodes. Nodes in the hierarchy are grouped by their depth.

provides an example of node hierarchy $\mathbb{H}$ returned by IMPLISig.

Identifying Coherent Feedback Loops

We defined a coherent feedback loop in section 1.1.7 as a cycle of a signed and directed network comprised only of edges with the same sign. In order to identify all coherent feedback loops of length n in SCC $\mathcal{C}$, we apply algorithm 2 to identify all coherent cycles of length n starting from each node $u \in V_{\mathcal{C}}$.

Algorithm 2 performs a recursive depth first search on the neighbours of a node. The functions positiveNeighbours(u) (and negativeNeighbours(u)) return all nodes $v \in V_{\mathcal{C}}$ such that $(u, v) \in E_{\mathcal{C}}$ and (u, v) is a positively (or negatively) signed edges. To ensure that nodes are not repeated in the cycle, the variable visited holds the list of previously visited nodes. This list is removed from the list of possible neighbours to explore. When the depth of the search is one less than the desired length of the cycle, the possible set of neighbours is equal to the intersection between the set of neighbours and the original, starting, node (algorithm 2, line 14). Naturally, this is non-empty when the set of neighbours contains the start node and is empty otherwise.

Algorithm 2 Enumerate coherent cycles of length n starting from node u.

```
function FindCycles(𝔾, u, n, visited, start, mode)
    if n = 0 then                                              ▷ Base case
        return [[u]]                                           ▷ Search depth is 0, u = start
    else
        visited ← visited ∪ {u}
        if mode = pos then
            neighbours ← 𝔾.positiveNeighbours(u)               ▷ Consider positive edges
        else if mode = neg then
            neighbours ← 𝔾.negativeNeighbours(u)               ▷ Consider negative edges
        else
            neighbours ← 𝔾.neighbours(u)
        end if
        if n > 1 then
            neighbours ← neighbours \ visited                  ▷ Remove visited nodes
        else
            neighbours ← neighbours ∩ start                    ▷ Continue if start ∈ neighbours
        end if
        for v ∈ neighbours do
            for cycle ∈ FindCycles(𝔾, v, n − 1, visited, start, mode) do
                yield [u] + cycle
            end for
        end for
    end if
end function
```

Subnetwork Score Function

After enumerating a set of coherent cycles, we score the induced subnetwork $\mathbb{S}' = (V_{\mathbb{S}'}, E_{\mathbb{S}'})$ formed by the nodes in the cycle in the following manner:

$$\textsc{Score}(\mathbb{S}') = \frac{\sum_{u \in \mathbb{S}'} k_u^{in}}{\sum_{u \in \mathbb{S}'} k_u^{in} + k_u^{out}} \tag{5.3}$$

where k_u^{in} is the number of edges from node u that connect to nodes in subnetwork $\mathbb{S}'$, and k_u^{out} the number of edges from node u that connect to nodes in $\mathbb{S} \setminus \mathbb{S}'$.

Since the subnetworks are selected based on dynamical information – the signs of the relations between nodes – we apply a purely topological subnetwork scoring function that looks for modular structure defined by a high ratio of internal edges to total possible edges (Lancichinetti, Fortunato, and Kertész, 2009). This is motivated by our intuition that, for two feedback loops containing the same number of nodes, the one with a greater ratio of internal to external edges should be selected as that

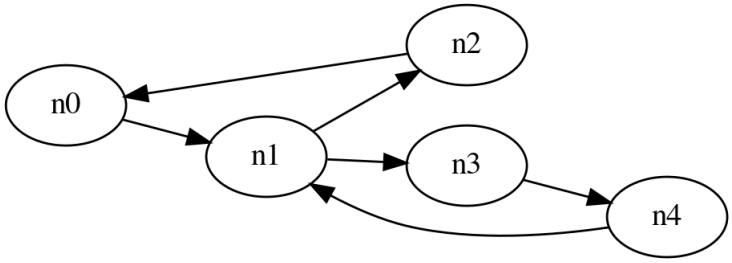


FIGURE 5.5: In this example, both the subnetworks induced by the loops (n0, n1, n2) and (n1, n3, n4) are assigned same score and so would be selected to be collapsed in the same iteration of IMPLISig. Since these two subnetworks contain an overlapping node (node n1), then the subnetworks are combined into a single subnetwork containing five nodes before being collapsed into a super-node.

one forms a more cohesive, functional group[4].

Selecting Subnetworks for Collapse

At each iteration of the algorithm, we determine the set of all maximum scoring subnetworks $\mathcal{S}^{\text{MAX}}$. If $\left|\mathcal{S}^{\text{MAX}}\right| = 1$ (only one subnetwork is maximal) then it is collapsed using algorithm 4. If no positive scoring subnetwork can be found, that is $\left|\mathcal{S}^{\text{MAX}}\right| = 0$, then either the mode is changed from positive to negative or the cycle size l is incremented.

In the case of multiple best-scoring subnetworks $\left(\left|\mathcal{S}^{\text{MAX}}\right| > 1\right)$, we avoid giving arbitrary priority to one by selecting all $\mathbb{S}^{\text{MAX}} \in \mathcal{S}^{\text{MAX}}$ for simultaneous collapse. Additionally, we merge best scoring subnetworks that overlap (using algorithm 3). 'Overlapping' here refers to containing one or more of the same nodes. We see an

Algorithm 3 Combine any maximum-scoring subnetworks that overlap.

```
function CombineOverlaps( S^MAX )
    N_{S^MAX} ← |S^MAX|                          ▷ Number of maximum-scoring subnetworks
    overlaps ← zeros(N_{S^MAX}, N_{S^MAX})       ▷ Determine pairwise overlaps
    for i ∈ [1, N_{S^MAX}] do
        for j ∈ [1, N_{S^MAX}] do
            overlaps[i, j] ← |S_i^MAX ∩ S_j^MAX|
        end for
    end for
    S^MAX ← ConnectedComponents(overlaps)        ▷ Group by overlap
    return S^MAX
end function
```

[4]Note that IMPLISig is presented as a general framework, and domain/problem-specific score functions could be applied.

example network that would give rise to this situation in figure figure 5.5. Algorithm 3 is a two-step procedure that:

1. first determines the pairwise overlaps between all maximum-scoring subnetworks,

2. then, groups subnetworks using the connected components formed from the overlap matrix.

Collapse a Selected Subnetwork into Super-node

After identifying a subnetwork $\mathbb{S}^{\text{MAX}} \subseteq \mathcal{C}$ for collapse, we merge it into a single super-node that represents it. We achieve this by inserting a new node S into $\mathcal{C}$, and systematically removing all the nodes $u \in \mathbb{S}^{\text{MAX}}$ from $\mathcal{C}$, adding any incoming/outgoing connections to the new super-node S. We detail this is algorithm algorithm 4.

Algorithm 4 Collapse a subnetwork $\mathbb{S}^{\text{MAX}} \subseteq \mathcal{C}$ into a super-node S.

```
1: function CollapseIntoSuperNode(𝒞, ℍ, 𝕊^MAX)
2:     𝒞.addNode(S)
3:     ℍ.addNode(𝕊^MAX)
4:     for n ∈ 𝕊^MAX do
5:         ℍ.addEdge(n, 𝕊^MAX)
6:         for (u, v, w) in 𝒞.outEdges(n) do                ▷ u = n
7:             𝒞.addEdge(S, v, w)
8:         end for
9:         for (u, v, w) in 𝒞.inEdges(n) do                 ▷ v = n
10:            𝒞.addEdge(u,S, w)
11:        end for
12:        𝒞.removeNode(n)
13:    end for
14: end function
```

Termination

IMPLISig terminates when $\mathcal{C}$ has been merged into a single node. That is, when $|V_{\mathcal{C}}| = 1$.

5.2.3 Computational Complexity of Algorithm 1

For determining all SCCs in a given network we apply Tarjan's algorithm. The Tarjan procedure is called once for each node, and considers each edge at most once (Tarjan, 1972). The algorithm's running time is, therefore, linear in the number of edges and nodes: achieving $\mathcal{O}(|V| + |E|)$.

For each SCC, we enumerate all directed loops of length l in an SCC containing $|V_{\mathcal{C}}|$ nodes requires a depth first search of the neighbours of a node of depth l. Therefore the computational complexity of FINDCYCLES$(\cdot)$ (algorithm 2) is $\mathcal{O}\left(|V_{\mathcal{C}}| \cdot \langle k_{\text{out}} \rangle^{l}\right)$, where $\langle k_{\text{out}} \rangle$ is the expected out-degree of a node. Since all loops of length $l \leq l_{\max}$ can be discovered as part of a depth-first search of depth $l_{\max}$ by monitoring depth as part of the search, we can say that the total complexity of enumerating all cycles of length $l \in [l_{\min}, l_{\max}]$ is $\mathcal{O}\left(|V_{\mathcal{C}}| \cdot \langle k_{\text{out}} \rangle^{l_{\max}}\right)$.

Scoring each subnetwork is linear to the sum of the degrees of the nodes, which, in the worst case is $\mathcal{O}(l_{\max} \cdot \langle k \rangle)$ since each subnetwork will contain a maximum of $l_{\max}$ nodes with expected degree $\langle k \rangle$. This must be repeated for all subnetworks. Identification of the best scoring subnetwork can be made a constant operation if subnetworks are indexed by score and the best score is maintained as scores are being computed. This makes the total complexity of scoring $\mathcal{K}$ enumerated subnetworks, each containing at most $l_{\max}$ nodes, and selecting the best scoring subnetworks from that set is $\mathcal{O}(\mathcal{K} \cdot l_{\max} \cdot \langle k \rangle)$

From a set of $n = \left|\mathcal{S}^{\text{MAX}}\right|$ best scoring subnetworks, determining overlaps requires identification of common vertices between all n subnetworks. Set intersection between two subnetworks (node sets s and t) is $\mathcal{O}(\min(|s|, |t|))$ and this is repeated for all $n \cdot (n-1)$ unique pairs of best scoring subnetworks. Since all subnetworks must contain, at most, $l_{\max}$ nodes, computation of the existence of a common element in all n best scoring subnetworks is $\mathcal{O}(n \cdot (n-1) \cdot l_{\max})$, in the worst case. From this, subnetworks can be grouped according to common elements by determining connected components in a undirected network containing n nodes – each representing a best scoring subnetwork, where an edge exists between nodes u and v if subnetworks u and v contain at least one common node. Supposing that the total number of pairs of best scoring subnetworks containing a common node is m, then

the complexity of determining unique groups (algorithm 3) is $\mathcal{O}\left(n+m\right)$[5].

Merging a subnetwork, containing at most $l_{\max}$ nodes, into a super-node involves iterating over each node in the subnetwork and adding an edge for each edge incident to that node, and then removing that node. This has complexity $\mathcal{O}\left(l_{\max} \cdot \langle k \rangle\right)$. At most, there will be $n = \left|\mathcal{S}^{\text{MAX}}\right|$ best scoring subnetworks (in the case that there were no overlaps), and so the total complexity of merging is $\mathcal{O}\left(n \cdot l_{\max} \cdot \langle k \rangle\right)$.

In summary, the total complexity of one iteration of IMPLISig is:

$$\mathcal{O}\left(|V_{\mathcal{C}}| \cdot \langle k_{\text{out}} \rangle^{l_{\max}} + \mathcal{K} \cdot l_{\max} \cdot \langle k \rangle + n \cdot (n-1) \cdot l_{\max} + n + m + n \cdot l_{\max} \cdot \langle k \rangle\right)$$

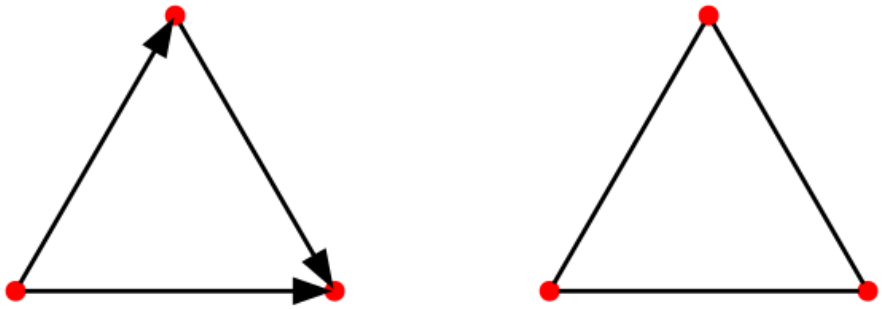
where $|V_{\mathcal{C}}|$ is the number of nodes in the SCC; $l_{\min}$, and $l_{\max}$ are the lower- and upper-bounds of the loop size to consider; $\mathcal{K}$ is the number of subnetworks induced from loops containing within $l_{\min}$, and $l_{\max}$ nodes; $\langle k \rangle$ is the expected degree of a node in the network; n is the number of best scoring subnetworks; and m is the number of overlaps.

At each iteration, the number of nodes, $|V_{\mathcal{C}}|$, decreases and, accordingly, so does the number of subnetworks $\mathcal{K}$.

5.2.4 Decomposition of the In- and Out-Components

The algorithm described in section 5.2.2 cannot be applied directly to the decomposition of the in- and out-components. This is because the in- and out-components are not typically strongly connected and so are not enriched with feedback loops. As discussed in section 1.4.5, we, instead, find an abundance of acyclic and feed-forward loops in these two components. Section 2.3.1 highlighted the importance of feed-forward loops, in particular, for noise filtering, fold-change detection and the identification of persistent signals (Mangan and Alon, 2003; Goentoro et al., 2009; Chou, 2018). In addition, (incoherent) feed-forward loops have been shown the be responsible for pulse shaping (Mangan and Alon, 2003). As such, we modify the algorithm described in the previous section to decompose in- and out-components into a hierarchy of feed-forward loops.

[5]This follows from the complexity of determining connected components in an undirected network being $\mathcal{O}\left(|V|+|E|\right)$, since a depth-first search is performed.



(A) Directed (B) Undirected

FIGURE 5.6: Transforming a directed feed-forward loop into an undirected loop. (A) shows the feed-forward loop in the original, directed network. This feed-forward loop does not contain any cycles. (B) shows the same feed-forward loop in an undirected network. This now does form a cycle and so will be discovered by algorithm 2.

Modifying the Coherent Feedback Loop Core Decomposition

The overall algorithm described by algorithm 1 can be applied to the in- and out-components with a number of minor modifications to the identification of cycles:

- **Identifying feed-forward loops**: as previously discussed, in- (and out-) components are characterised by their 'tree-like' appearance – that is: they connect into (or out from) the nodes in the core and rarely form directed cycles. However, by ignoring edge direction, nodes in feed-forward loops will form cycles where they did not before (see figure 5.6). As such, we transform each of the in- and out-components into their undirected equivalent before applying algorithm 1. This, in effect, broadens the search to all loops in the network – both feed-forward and feedback – while still selecting loops based on the ratio of internal to total edges (using equation (5.3)). Note that $l_{\text{min}} \geq 3$ as $l_{\text{min}} = 2$ would return every edge in the network.

- **Allowing for incoherent loops**: due the importance of incoherent feed-forward loops, we relax the requirement for any identified loop to be coherent. Consequently, we ignore edge sign in the search for cycles in line 8 of algorithm 1.

With the above modifications, our algorithm can be applied to decompose in- and out-components of bow-tie architectures.

5.3 Experimental Validation of Core Decomposition

The purpose of this section is to describe the experimental setup of the validation of the hypotheses introduced in section 5.1.1. Since the core of the bow-tie architecture is akin to the CPU of the computer, we validate these hypotheses through validation of the decomposition of the core of the bow-tie architectures found in a variety of signalling networks. Accordingly, we design the following two experiments to serve as case studies to validate our approach:

1. **Deep Feedback Loops in Bow-Tie Cores Control Global Dynamics** The first experiment focuses on bow-tie architectures and is concerned with measuring the significance of inhibiting a node in the strongly connected core with respect to the global network dynamics as a whole. We characterise the effect on the global network dynamics by measuring the change in expression of a subset of downstream output nodes. The purpose of this experiment is to validate hypothesis 1: namely to show that inhibiting nodes found deeply within the uncovered coherent feedback loop hierarchy will have the greatest effect on global network dynamics.

2. **Control Kernel Recovery & Identification** Our second set of experiments is concerned with validating our second hypothesis: that those same nodes make better therapeutic target candidates. To this end, our experiment involves recovering a literature-curated set of validated control kernels for a variety of benchmark gene regulatory networks.

The following subsections introduce these experiments as well as the network datasets in detail.

5.3.1 Deep Feedback Loops in Bow-Tie Cores Control Global Dynamics

To investigate hypothesis 1, we suppose the following:

- that a network $\mathbb{G}$ has at least one bow-tie architecture $\mathcal{B} = (\mathcal{I}, \mathcal{C}, \mathcal{O})$ within.
- that a subset of N_{out} nodes in the out-component $\mathcal{O}$ correspond to cell phenotype.

The purpose of the first experiment is to show that core nodes found deep within the uncovered coherent feedback loop hierarchy have a significant effect on the phenotype of the network. Concretely, we measure whether inhibiting nodes in a core of a bow-tie network $\mathbb{G}$ can have a significant effect on those N_{out} output nodes of interest.

Details of Experiment

Here we provide details on the experimental setup for the evaluation of hypothesis 1.

We suppose a bow-tie network $\mathbb{G} = (V, E)$ that contains a core, $\mathcal{C} \subseteq V$, with in- and out-components $\mathcal{I} \subseteq V$ and $\mathcal{O} \subseteq V$ respectively. We measure whether inhibiting nodes in the core $\mathcal{C}$ of $\mathbb{G}$ can have a significant effect on N_{out} (where $N_{\text{out}} \leq |\mathcal{O}|$) output nodes of interest $\{o_i \mid i \in [1, N_{\text{out}}]\} \subseteq \mathcal{O}$. In the case that the state space for network $\mathbb{G}$ is large[6], we sample 10000 randomly selected initial states $\mathbf{x}_i$ compute their corresponding attractors by partially reconstructing the stage transition diagram (STG)[7].

Let $N_s = \min\left(10000, 2^N\right)$ denote the number of sampled unique initial states. We use $\mathbb{I}$ to denote the set of all initial states: $\mathbb{I} = \{\mathbf{x}_i \mid i \in [1, N_s]\}$, with $|\mathbb{I}| = N_s$. We will further use $\mathbb{A}^{\mathbb{G}} = \{A^{\mathbb{G}}_{\mathbf{x}_i} \mid \mathbf{x}_i \in \mathbb{I}\}$ to denote the corresponding set of attractors for those initial states $\mathbf{x}_i \in \mathbb{I}$ for network $\mathbb{G}$. Note that the elements in $\mathbb{A}^{\mathbb{G}}$ may not be unique – different initial states $\mathbf{x}_i \in \mathbb{I}$ may belong to the same basin of attraction and therefore converge to the same attractor. Note also that the attractor for initial state $\mathbf{x}_i$ $A^{\mathbb{G}}_{\mathbf{x}_i} \in \mathbb{A}^{\mathbb{G}}$ is a set of states: $A^{\mathbb{G}}_{\mathbf{x}_i} = \{\mathbf{y}_1, ..., \mathbf{y}_k\}$. $|A^{\mathbb{G}}_{\mathbf{x}_i}| = k$ means that the attractor determined from initial condition $\mathbf{x}_i$ for network $\mathbb{G}$ has period length k. Naturally $|A^{\mathbb{G}}_{\mathbf{x}_i}| = 1 \implies A^{\mathbb{G}}_{\mathbf{x}_i}$ is a steady state attractor.

Suppose that we are interested in measuring the significance of change for output node o. Then, for each identified attractor $A^{\mathbb{G}}_{\mathbf{x}_i} \in \mathbb{A}^{\mathbb{G}}$, we record the mean activation of node o. Mean activation $\langle A^{\mathbb{G}}_{\mathbf{x}_i} \rangle_o$ for node o for attractor $A^{\mathbb{G}}_{\mathbf{x}_i}$ with $|A^{\mathbb{G}}_{\mathbf{x}_i}| = k$ is

[6] Our definition of "large" is network size $N > 13$, since $N = 14$ would require consideration of $2^{14} = 16384 > 10000$ states.

[7] Reconstructing the STG refers to iteratively building the STG, starting from each initial state $\mathbf{x}_i$. For each state $\mathbf{x}_i$, its neighbour in the STG is computed by performing a network update. Neighbours are repeated computed until a cycle is found. The found cycle represents the corresponding attractor for initial condition $\mathbf{x}_i$. The period of the cycle may be 1, indicating that the attractor is a steady state attractor (see section 1.5.7). We focus on the synchronous update scheme only.

computed as the number of times that node o takes a value of 1 divided by the period length k of the attractor:

$$\langle A^{\mathbb{G}}_{\mathbf{x}_i} \rangle_o = \frac{1}{k} \sum_{\mathbf{y} \in A^{\mathbb{G}}_{\mathbf{x}_i}} \mathbf{y}_o \tag{5.4}$$

where $\mathbf{y}_o \in \{0, 1\}$ is the boolean value that node o takes in state $\mathbf{y}$.

To examine the significance of turning off specific nodes in the core, we perform the following: For each node, $c \in \mathcal{C}$, in the core of the bow-tie architecture of the network, we simulate the *inhibition* of node c. We achieve this by copying and modifying $\mathbb{G}$ by pinning the value of c to 0 to construct $\mathbb{G}_c$, the network with node c inhibited. We then compute the resulting set of attractors for network $\mathbb{G}_c$, $\mathbb{A}^{\mathbb{G}_c} = \{A^{\mathbb{G}_c}_{\mathbf{x}_i} \mid \mathbf{x}_i \in \mathbb{I}\}$, for each previously sampled initial state $\mathbf{x}_i \in \mathbb{I}$. We again record the mean activation of output node o of interest.

To evaluate the significance in output activation between the original network $\mathbb{G}$ and modified network $\mathbb{G}_c$, we perform a paired t-test. A paired t-test measures the significance of the change in the mean difference of output activation for the resulting attractors of the same initial state $\mathbf{x}_i$, In other words, we are measuring the significance of:

$$t(c, o; \mathbb{G}, \mathbb{I}) := \mathop{\mathbb{E}}_{\mathbf{x}_i \sim \mathbb{I}} \left[\langle A^{\mathbb{G}}_{\mathbf{x}_i} \rangle_o - \langle A^{\mathbb{G}_c}_{\mathbf{x}_i} \rangle_o \right] \tag{5.5}$$

$$= \frac{1}{N_s} \sum_{\mathbf{x}_i \in \mathbb{I}} \langle A^{\mathbb{G}}_{\mathbf{x}_i} \rangle_o - \langle A^{\mathbb{G}_c}_{\mathbf{x}_i} \rangle_o \tag{5.6}$$

We then record which core nodes c significantly changed the activation of each output node o, using a significance level of 0.05. Note that a equation (5.5) is a signed statistic and both one- and two-tailed t-tests can be performed, depending on whether or not direction of change of expression is important.

5.3.2 Network Datasets

We validate hypothesis 1 on synthetic random boolean networks displaying the bow-tie architecture, as well as on three case study boolean network models of the signalling networks of cancerous cells that we have obtained from literature:

1. the AGS gastric cancer cell line (Flobak et al., 2015),
2. the EGFR-Erbb1 signalling network with an over-expressed Erbb11 receptor (Samaga et al., 2009),
3. and, a boolean model tumour cell invasion and metastasis (Cohen et al., 2015).

Further experimental details are provided in section 5.3.1.

Synthetic Bow Tie Networks

As a proof of concept, we randomly generate synthetic bow tie networks with three different core sizes to validate our hypothesis coherent feedback loops are promising targets. Each of our generated networks contain the three components characteristic of the bow-tie architecture: the in-, out- and core components.

We aim to validate our intuition that nodes in a coherent feedback loop have a more significant effect on the expression of output nodes compared to the other nodes in a strongly connected component that do not belong to a coherent feedback loop. To this end, we plant a three-node coherent feedback loop in the core of every synthetic network. The sign of the remaining edges in the network is otherwise randomly selected, taking care to ensure that exactly one coherent feedback loop can be found the core.

Construction of the networks is a two step process:

1. generate topology (described by algorithm 5 in the appendices),
2. generate boolean rules based on topology (described by algorithm 9 in the appendices).

Complete details of synthetic bow-tie generation can be found in appendix D.1 in the appendices.

For these synthetic bow-tie networks, we consider every node in the out-component to be an output node of interest. Since the networks are synthetic and outputs are not associated with phenotype, we do not consider the direction of the mean change in expression, and, accordingly, we perform two-tailed t-tests to measure significance.

TABLE 5.1: Description of the output transcription factors of interest, and their mean activation across the attractors of 10000 sampled states in the AGS Gastric cancer cell line network (Flobak et al., 2015).

Role	Output Node o	Mean Activation $\mathbb{E}_{\mathbf{x}_i \sim \mathbb{I}} \langle A^{\mathbb{G}}_{\mathbf{x}_i} \rangle_o$
Pro-Cancer	RSK	1.0000
	TCF	1.0000
	cMYC	1.0000
Anti-Cancer	Caspase8	0.0000
	Caspase9	0.0000
	FOXO	0.0000

Cancerous Case Study Networks

To concretely validate hypothesis 1, we use three boolean network models of cancerous networks from literature:

1. a cell fate decision network in the AGS gastric cancer cell line (Flobak et al., 2015);
2. the EGFR-Erbb1 signalling network (Samaga et al., 2009);
3. and, molecular pathways enabling tumour cell invasion and migration (Cohen et al., 2015).

Details of these networks – in particular, the output nodes of interest for each of them – follow.

For these real-world case study networks, direction of change is important. Since we are dealing with cancerous networks, we are concerned specifically with statistically significant decreases in expression of pro-cancer output nodes relevant and, likewise, statistically significant increases in expression of anti-cancer outputs. Accordingly, we use a one sided paired t-test to determine if the expression of each output gene was significantly difference from the original control network $\mathbb{G}$ for each modified network $\mathbb{G}_c$ in the appropriate direction.

AGS Gastric Cancer Cell Line

The AGS Gastric Cancer Cell Line network is a "dynamical model representing a cell fate decision network in the AGS gastric cancer cell line, relying on background knowledge extracted from literature and databases" (Flobak et al., 2015). Based on

TABLE 5.2: Description of the output transcription factors of interest, and their mean activation across the attractors of 10000 sampled states in the EGFR/ErbB signalling network (Samaga et al., 2009). We set Erbb11 to always on to simulate a cancerous phenotype.

Role	Output Node o	Mean Activation $\mathbb{E}_{\mathbf{x}_i \sim \mathbb{I}} \langle A^{\mathbb{G}}_{\mathbf{x}_i} \rangle_o$
Pro-Cancer	elk1	0.0987
	creb	0.3005
	ap1	0.1241
	cmyc	0.0806
	p70S6_2	0.0299
	hsp27	0.2478
Anti-Cancer	pro_apoptotic	0.7522

the "growth score" measure introduced for this network (Flobak et al., 2015), we identify the six transcription factors (TFs) described in table 5.1 as the output nodes of interest for our study. We group them into two groups of three depending on whether they contribute positively or negatively to growth score. We label these groups pro-cancer and anti-cancer respectively and maintain this group labelling scheme for the remaining two case study networks. As table 5.1 shows, each pro-cancer TF o take a value of $\langle A^{\mathbb{G}}_{\mathbf{x}_i} \rangle_o = 1$ for each resulting attractor $A^{\mathbb{G}}_{\mathbf{x}_i}$ for each sampled initial state $\mathbf{x}_i \in \mathbb{I}$.

EGFR-Erbb1 Signalling Network

For our next case study, we select the Epidermal growth factor receptor (EGFR/ErbB1) signalling network (Samaga et al., 2009). To simulate a cancerous mutation on the network, we mutate the Erbb11[8] receptor to be always on (simulating over-expression). We follow previous works and consider the activations of seven output TFs in the network as representative of the cell phenotype. The output nodes and mean activations are described in table 5.2.

Tumour Cell Invasion and Migration

Our final case study network is Tumour Cell Invasion and Migration (TCIM) network, a logical model of metastasis that was constructed to "recapitulate published

[8]We determined that over-expressing Erbb11 had the most cancerous effect on the EGFR-Erbb1 network with respect to the output TFs identified in table 5.2. That is, overall, over-expressing Erbb11 caused the greatest increase in the expression of cell growth TFs and greatest decrease in the expression of the cell death TF.

TABLE 5.3: Description of the output nodes of interest, and their mean activation across the attractors of 10000 sampled states in the Tumour Cell Invasion and Migration (TCIM) network (Cohen et al., 2015).

Role	Output Node o	Mean Activation $\mathbb{E}_{\mathbf{x}_i \sim \mathbb{I}} \langle A^{\mathrm{G}}_{\mathbf{x}_i} \rangle_o$
Pro-Cancer	Metastasis	0.2769
Anti-Cancer	CellCycleArrest	0.66905
	Apoptosis	0.1035

experimental results of known gene perturbations on local invasion and migration processes" (Cohen et al., 2015). This network was validated by the authors on lung cancer. We select the output nodes detailed in table 5.3 since they are the three "terminal" output nodes – nodes with no outgoing edges – corresponding clearly to phenotype.

5.3.3 Control Kernel Recovery

For literature validation of hypothesis 2, we see first how many control nodes appear in core coherent feedback loops, and then – as a global measure of performance – use the depth of a node in the hierarchy of coherent loops as a predictor for control kernel membership.

For this, we use five boolean models of gene regulatory networks (GRNs): Saccharomyces cerevisae cell cycle (SCCC) (Li et al., 2004), Schizosaccharomyces pombe (SP) (Wang et al., 2010), Mammalian cortical development (MCD) (Giacomantonio and Goodhill, 2010), Arabidopsis thaliana development (ATD) (Espinosa-Soto, Padilla-Longoria, and Alvarez-Buylla, 2004; Alvarez-Buylla et al., 2007), and Mouse myeloid development (MMD) (Krumsiek et al., 2011). Details of the control kernels identified for those five GRNs by Kim, Park, and Cho, 2013 are given in table 5.4.

5.3.4 Network Statistics

The statistics of all of the networks used in this study are given in table 5.5.

TABLE 5.4: Control kernels for small GRNS identified in Kim, Park, and Cho, 2013. Each row within the same network represents a separate control kernel.

Network	Control Kernels	All in Core (Largest SCC)?
SCCC	{Cdh1, MBF, SBF, Sic1}	Yes
SP	{Cdc25, Rum1, Ste9, Wee1}	Yes
MCD	{Fgf8_g}	Yes
	{Emx2_g}	Yes
	{Sp8_g}	Yes
	{Fgf8_p}	Yes
	{Emx2_p}	Yes
	{Sp8_p}	Yes
ATD	{AP1}	Yes
	{LFY}	Yes
	{SEP}	Yes
MMD	{GATA_1, EKLF, Fli_1}	Yes
	{EKLF, Fli_1, PU1}	Yes

5.4 Experimental Results

5.4.1 Measuring Significance of Turning off Members of Coherent Feedback Loops

As mentioned in section 5.3.2, we used small synthetic bow tie networks, with a single planted coherent feedback loop in the core, to validate our intuition that members of coherent feedback loops in the cores of bow-tie architectures make for significant targets. To this end, we generated 1000 random synthetic bow tie networks with different core sizes and types planted coherent feedback loop.

Figure 5.7 provides a box-plot of the results of this experiment for $N_{\text{core}} = 6$, showing the expected change in an output node caused by inhibiting a core node. From left to right: figure 5.7 shows a planted positive feedback loop (achieving a one-sided p-value of 2.2901E-16), a planted negative feedback loop (achieving a one-sided p-value of 3.7529E-33), and a mix of a planted positive or planted negative feedback loop (achieving a one-sided p-value of 2.7695E-27). We see that for both a planted positive and planted negative feedback loop there was a statistically significant increase in the proportion of significantly changed outputs between the two groups of 'Planted coherent feedback loop' and 'Other core nodes'. This suggests

TABLE 5.5: Network statistics. For synthetic bow-tie networks, we report the mean and standard deviation for 1000 samples. *Key:* N number of nodes; $|E|$ number of edges; $\langle k \rangle$ mean degree; $\langle \text{bc} \rangle$ mean betweenness centrality; ρ network density; N_{in} number of nodes in the in-component; N_{core} number of nodes in the largest SCC; N_{out} number of nodes in the out-component; $\langle k_{\text{core}} \rangle$ mean degree of the core; $\langle s \rangle$ average sensitivity (Daniels et al., 2018). Note that $\langle s \rangle$ is missing for SCCC and SP since the logic of those networks is described using threshold networks. Likewise, ATD is described by a multi-valued network (Espinosa-Soto, Padilla-Longoria, and Alvarez-Buylla, 2004) and so $\langle s \rangle$ is missing.

	N	$\lvert E \rvert$	$\langle k \rangle$	$\langle bc \rangle$	ρ	N_{in}	N_{core}	N_{out}	$\langle k_{\text{core}} \rangle$	$\langle bc_{\text{core}} \rangle$	ρ_{core}	$\langle s \rangle$
Effect of Inhibition of Core Nodes on Output Nodes												
$\text{Synthetic}_{N_{\text{core}}=6}$	19	$32.297_{\pm 2.660}$	$3.400_{\pm 0.280}$	$0.033_{\pm 0.004}$	$0.094_{\pm 0.008}$	3	6	10	$6.883_{\pm 0.443}$	$0.103_{\pm 0.011}$	0.300	$0.999_{\pm 0.030}$
$\text{Synthetic}_{N_{\text{core}}=10}$	23	$41.343_{\pm 2.828}$	$3.595_{\pm 0.246}$	$0.043_{\pm 0.004}$	$0.082_{\pm 0.006}$	3	10	10	$5.834_{\pm 0.283}$	$0.100_{\pm 0.009}$	0.189	$0.999_{\pm 0.029}$
$\text{Synthetic}_{N_{\text{core}}=15}$	28	$51.803_{\pm 2.859}$	$3.700_{\pm 0.204}$	$0.050_{\pm 0.004}$	$0.069_{\pm 0.004}$	3	15	10	$5.254_{\pm 0.191}$	$0.094_{\pm 0.008}$	0.129	$0.999_{\pm 0.028}$
EGFR	104	254	4.885	0.023	0.024	33	34	35	6.088	0.061	0.055	0.954
Gastric	73	141	3.863	0.062	0.027	0	65	8	4.077	0.068	0.031	0.974
TCIM	32	159	9.938	0.024	0.160	2	20	10	12.600	0.035	0.279	0.729
Control Kernel Recovery												
SCCC	11	34	5.667	0.086	0.258	2	9	1	6.333	0.114	0.347	–
SP	9	26	5.778	0.125	0.361	1	8	0	6.000	0.141	0.411	–
MCD	10	19	3.800	0.214	0.211	0	10	0	3.800	0.214	0.211	0.900
ATD	15	41	5.467	0.063	0.195	2	10	2	6.200	0.095	0.289	–
MMD	11	30	5.000	0.088	0.227	0	8	4	6.125	0.111	0.411	1.023

that our hypothesis that targeting members of coherent feedback loops will significantly affect output expression versus targeting incoherent feedback loops. Box plots for $N_{\text{core}} = 10$ and $N_{\text{core}} = 15$ can be found in figure D.4 of supplementary materials.

5.4.2 Cancerous Case Study Networks

Table 5.6 shows the AUROC and AP scores achieved by IMPLISig for predicting the impact of turning off a core node with respect to the output nodes of interest. We compare against a number of topological microscopic network features. From this, we find that a node's position in the hierarchy of coherent feedback loops can provide a high quality prediction of the significance of inhibiting it. Strikingly, best performance is achieved on the Gastric cancer network, a network that exhibits the most cancerous behaviour in its base state (see table 5.1), beating out all other measures by a significant margin for both AUROC and AP.

5.4.3 Control Kernel Recovery

In table 5.7, we evaluate the success of a the uncovered position of nodes in the coherent feedback loop hierarchy at predicting whether that node belongs to a control kernel. We see that merge depth provides a reliable indication of a node's control node status – achieving AUROC and AP scores greater than 0.5 in four out of the

TABLE 5.6: Summary of results on case study networks We present AUROC and AP scores for predicting a significant effect on an output node. The AUROC and AP scores are averaged across all output nodes. All measures are reported to 3 decimal places. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient, and cn: core number. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of significance. Full AUROC and AP scores are given in table D.1 (Gastric Cancer), table D.3 (EGFR), and table D.5 (TCIM) in the appendices.

	Gastric		EGFR		TCIM	
	AUROC	AP	AUROC	AP	AUROC	AP
k	0.551	0.175	0.638	0.378	0.488	0.513
k_{in}	0.568	0.148	0.566	0.358	0.473	0.491
k_{out}	0.530	0.160	0.685	0.439	0.526	0.514
bc	0.640	0.244	**0.728**	0.525	0.491	0.541
cc	0.422	0.134	0.415	0.288	0.555	0.595
cn	0.668	0.160	0.454	0.300	0.471	0.473
IMPLISig	**0.831**	**0.310**	0.668	**0.556**	**0.666**	**0.633**

TABLE 5.7: Feedback loop hierarchy position as predictor of control node. We present AUROC and AP scores for predicting a significant effect on an output node. The AUROC and AP scores are averaged across all output nodes. All measures are reported to 3 decimal places. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient and cn: core number,. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of control kernel membership.

AUROC	SCCC	SP	MCD	ATD	MMD	Mean AUROC
k	0.300	0.250	0.875	0.810	0.688	0.584
k_{in}	**0.575**	0.625	0.542	0.619	0.563	0.585
k_{out}	0.150	0.000	0.688	**0.929**	0.688	0.491
bc	0.550	0.125	**1.000**	0.762	0.719	0.631
cc	0.325	0.500	0.500	0.381	0.406	0.422
cn	0.350	0.500	0.625	0.810	0.625	0.582
IMPLISig	0.300	**0.688**	0.833	0.881	**0.938**	**0.728**
AP	SCCC	SP	MCD	ATD	MMD	Mean AP
k	0.385	0.500	0.889	0.589	0.775	0.628
k_{in}	**0.542**	**0.667**	0.678	0.433	0.536	0.571
k_{out}	0.383	0.417	0.800	**0.867**	0.750	0.643
bc	0.518	0.450	**1.000**	0.556	0.813	0.667
cc	0.402	0.500	0.600	0.304	0.600	0.481
cn	0.389	0.500	0.667	0.533	0.583	0.534
IMPLISig	0.389	0.650	0.867	0.639	**0.900**	**0.689**

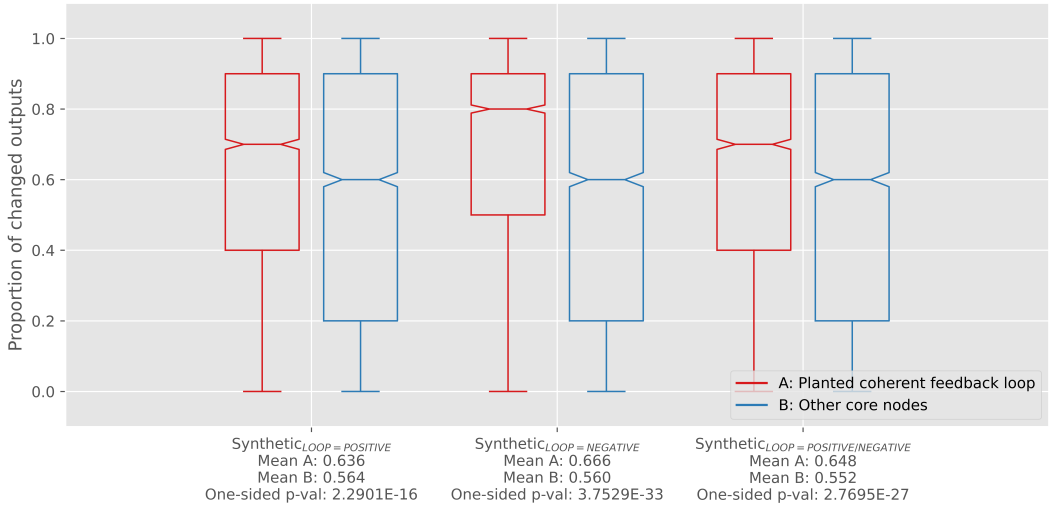
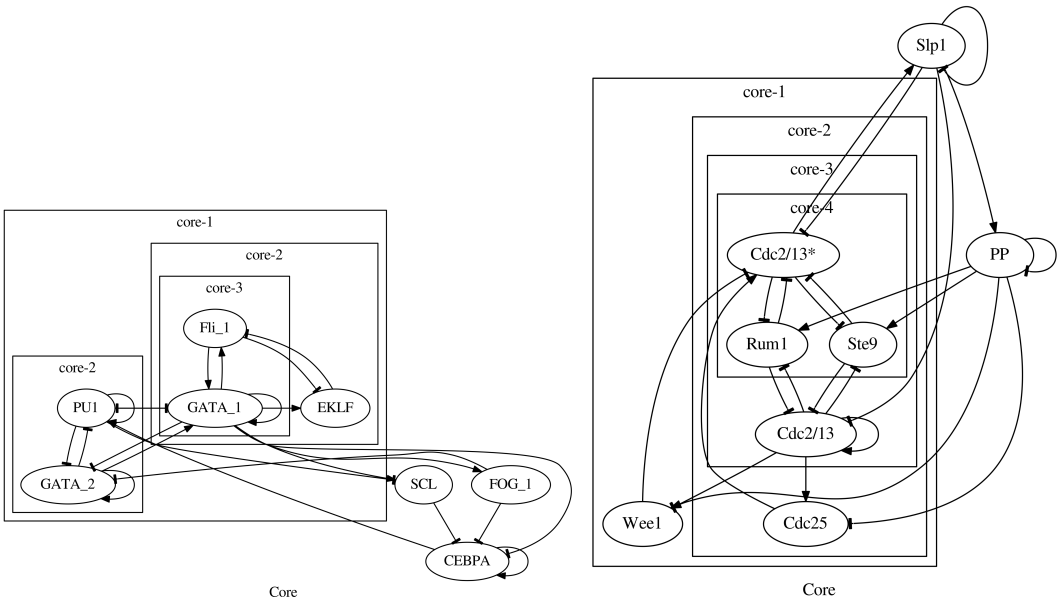


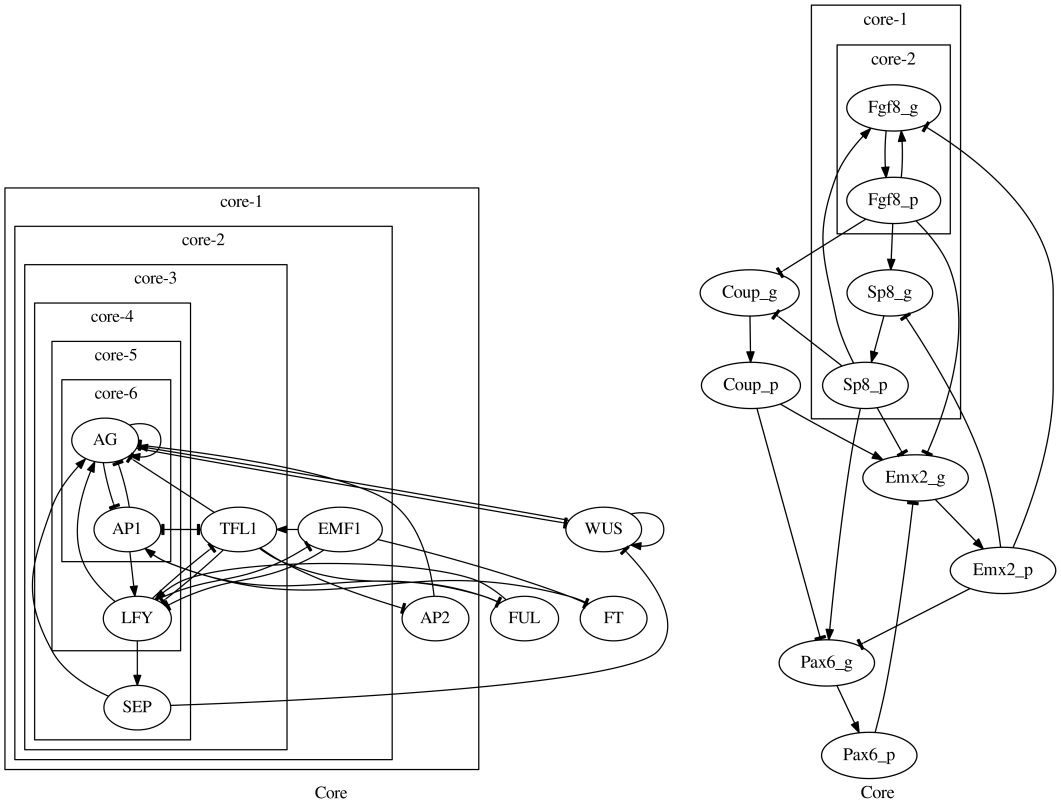
FIGURE 5.7: Comparison of expected significant change in synthetic bow tie networks. Box plots to show the difference in expected change for three types of synthetic bow tie networks. Each network has $N_{\text{in}} = 3$ nodes in the in-component, $N_{\text{core}} = 6$ nodes in the core, and $N_{\text{out}} = 10$ nodes in the out-component.. Each core was built with a single planted coherent feedback loop l_c of three nodes that the rest of the core was built upon (see algorithm 5). From left to right: the core feedback loop is positive, negative and either positive or negative. For each network, the planted coherent loop is the only loop of that type in the entire core of the network. For each network architecture, 1000 networks $\mathbb{G}$ were randomly generated. Then, for each node c in the core, a new network $\mathbb{G}_c$ was created as a copy of $\mathbb{G}$ with c forced off. A two-tailed paired t-test was performed to determine which, if any, of the 10 output nodes displayed a statistically significant mean difference in expression. For each core node c, the mean number of changed outputs was recorded in one of two groups: if $c \in l_c$, then it was recorded in the group 'Planted coherent feedback loop', otherwise it was recorded in group 'Other core nodes'. Since there was always three nodes in the core positive feedback loop, and 1000 networks of each type were generated, then the group size for 'Planted coherent feedback loop' was 3000. The group size of 'Other core nodes' was also 3000. p-values from a one tailed paired t-test performed between the two groups are reported on the x-axis. The y-axis reported the proportion of significantly changes outputs. Similar results were obtained for $N_{\text{core}} = 10$ and $N_{\text{core}} = 15$ (see figure D.4).

five networks. As well, we see that IMPLISig provides the most reliable measure across all of the networks, achieving the greatest mean AUROC measure of 0.728 and the greatest overall mean AP measure of 0.689. IMPLISig also achieves a mean rank of 2.4 for both AUROC and AP, the lowest mean rank across all measures (see table D.14 in the appendices for full rankings). Figure 5.8 provides a representation of the uncovered decomposition of the cores of four of the GRNs.



(A) Core of the MMD network

(B) Core of the SP network



(C) Core of the ATD network

(D) Core of the MCD network

FIGURE 5.8: Decomposition of the cores of four GRNs. (A): decomposition of the core of the MMD network. The outer black box contains all of the nodes in the largest SSC in the network (the core of the bow-tie architecture). Within this, the boxes labelled core-1, core-2 and core-3 show the three layers of the uncovered coherent feedback loop hierarchy. Note that all of the members of the right core-2, that is Fli_1, GATA_1 and EKLF form a control kernel (see table 5.4). (B): decomposition of the core of the SP network. Two out of the three members of core-4 belong to the control kernel: Rum1 and Ste9. (C): decomposition of the core of the ATD network. The three control kernels of {AP1}, {LFY}, and {SEP} can be found in the thee deepest layers: core-6, core-5 and core-4 respectively. (D): decomposition of the core of the MCD network. The entirety of this network forms an SCC and so belongs to the core. Both members of core-2, Fgf8_g and Fgf8_p, are control kernels.

5.5 Chapter Summary

In this chapter we have presented IMPLISig, a fast and flexible framework to uncover the hidden hierarchy of modular, coherent feedback loops in the cores of signalling networks, as well as the position of peripheral nodes in a hierarchy of modular feed-forward loops. For core nodes, we have shown that the position of a node within this hierarchy provides an indication of the significance of the effect that inhibiting that node will have upon the output of the network. As well, we have shown that the same measure is a reliable indicator of a control kernel membership, and is successful at recovering kernels obtained from literature.This leads us to believe that IMPLISig could be a useful tool to speed up the identification of novel therapeutic targets in the future. Overall, we conclude that IMPLISig can help to bridge the gap between network architectural scales and overall dynamic network behaviour.

5.5.1 A Note on Decomposing General Network Architecture

While the framework introduced in this chapter has been proposed within the context of bow-tie architectures, it can be readily applied to any signed and directed network. In the case that the network contains multiple weakly connected components, they can be readily identified by treating each edge as an undirected edge and performing a standard search for connected components. Application of Tarjan's algorithm can be used to identify any arbitrary number of strongly connected components $\{\mathcal{C}_1, ..., \mathcal{C}_n\}$ within a weakly connected directed network. As discussed in section 5.2.1, our approach can be applied to each of those subnetworks $\mathcal{C}_i$, $i \in [1, n]$, in parallel. The output would be a directed, acyclic graph, corresponding to a *condensation*[9] of the original network. For each SCC, a loop hierarchy would be produced by algorithm 1. Based on the promising validation of our approach for bow-tie core decomposition in section 5.4, we suggest that this general network decomposition may prove useful in uncovering the flow of information through general signalling networks, comprised a variety of architectures. The DAG produced by the application of Tarjan's algorithm illustrates the high-level flow of information through the

[9] A condensation of a directed network is the network produced by replacing all SCCs within the network with single nodes.

network from inputs from network inputs to outputs, through multiple SCCs. Additionally, IMPLISig could potentially explain the flow through each SCC. In combination, we are able to build a more complete picture of the flow of information through multiple architectural scales. We leave validation of this intuition as future work.

Chapter 6

Discussion

This thesis provides a number of frameworks for low-dimensional modelling of complex networks, that uncover the hidden hierarchical organisation and dynamics of such networks. After a through literature review of the subject in chapter 2, we study the problem of low- dimensional representation learning of attributed and weighted complex networks to a non-Euclidean space in chapter 3, and then examined the same problem for directed networks in chapter 4. The main challenge stems from flexibility: general complex networks are ubiquitous yet highly heterogeneous – and representation learning techniques must, themselves, be flexible enough to handle a wide range of network types, while learning a low dimensional representation that can simultaneously capture many meso- and macro-scopic features common to complex networks – features such as the small-world property, clustering, a scale-free degree distribution, and node hierarchy. The theme of hierarchy continues into chapter 5, where we introduce a general framework for uncovering a hidden hierarchy of modular loops in dynamic systems described by boolean networks.This present chapter now summarises the contributions of this thesis and highlights potential future directions for all of the work proposed here.

6.1 Contributions & Conclusions

This thesis is centrally concerned with a general understanding of the hidden hierarchy of complex networks. From static, attributed and undirected, to static, attributed and directed, to fully dynamic and directed networks. The problems discussed in the previous chapters are arranged, themselves, hierarchically, with each

problem becoming progressively more general and more challenging. Since the primary contributions of this thesis are solutions to those problems, we will state those contributions broadly in the order in which they appear.

6.1.1 Uncovering the Hierarchical Organisation of Attributed and Weighted Undirected Networks through Random Walk Network Sampling and Hyperbolic Embedding

The first major contribution of this thesis is provided in chapter 3, where we introduce HEAT, a framework for learning low-dimensional hyperbolic vector representations of nodes in an attributed complex network. Our framework is flexible enough to handle both discrete and continuous attributes, as well as networks without attributes. In addition, it controls the trade-off between consideration of attributes and topology with an easy-to-interpret hyper-parameter.

As a framework, HEAT is easily extensible. For example, the proposed pairwise node similarity measure based on attributes (equation (3.1)) can be replaced with any domain- or problem-specific measure of pairwise similarity.

By embedding to hyperbolic space, we find that our method can better preserve the hierarchy implicit to attributed complex networks. We conclude this from our experiments, that show that HEAT can outperform a number of popular attributed network benchmark algorithms on a number of downstream machine learning tasks. The key difference between HEAT and these attributed embedding benchmarks is the choice of metric space, and so, our results show that attributed complex networks have a hidden hierarchy, just as unattributed ones have been shown to have (Papadopoulos et al., 2012; Alanis-Lobato, Mier, and Andrade-Navarro, 2016a; Alanis-Lobato, Mier, and Andrade-Navarro, 2016b; Nickel and Kiela, 2017). Furthermore, we show that incorporation of attributes can improve performance on two popular downstream machine learning tasks – namely, link prediction and node classification – justifying the usefulness of embedding as a tool for the simultaneous representation of topology as well as attributes.

6.1.2 Uncovering the Hierarchical Organisation of Attributed and Directed Networks through Inductive Hyperbolic Embedding

We then expanded upon the ideas presented in chapter 3 in the following chapter. In chapter 4, we extend previous works in the Euclidean domain to first introduce the problem of representing nodes in an attributed and directed complex network as low-dimensional Gaussian representations in hyperbolic space, the first formulation of such a problem.

We then present HEADNet, an inductive framework for tackling this problem. HEADNet is the first method for attributed and directed hyperbolic complex network embedding to a single, unified node representation (rather than two representations of each node in the network (Wu, Di, and Fan, 2019)).

Our results show that our approach is well founded: as with undirected networks, by supposing that the metric space underpinning directed networks is hyperbolic – and, therefore, that the elements of directed networks exhibit some hidden hierarchy – we are able to achieve state-of-the-art results. As an inductive method, HEADNet is capable of generalising about a network, ultimately learning the relationships between attributes and hyperbolic co-ordinates, rather than just learning a 'lookup table' of node positions. The advantage of this is that the embedding of unseen nodes (and prediction of their position in the hierarchy of the network) can be readily handled, so long as the attributes of those nodes fall in line with the distribution of attributes already seen during training. We are not the first to inductively embed nodes in a complex network (for example: both Hamilton, Ying, and Leskovec, 2017 and Bojchevski and Günnemann, 2018 do this), however, to the best of our knowledge, we are the first to propose this for embedding attributed and directed networks to hyperbolic space.

6.1.3 Uncovering the Hierarchical Organisation of Network Architectures

Chapter 5 is concerned with information flow through meso-scopic architectures found within dynamic networks. We follow on from our previous chapters by not only looking at more sophisticated forms of complex network (we can consider a state-changing dynamic network as an attributed directed network, whose attributes

change over time), but also by considering hierarchy not at the micro-scale – the hierarchy of nodes in the network – but at the meso-scale – the multi-scale emergence of loops within medium- to large-sized network architectures – namely, the bow-tie architecture. This lead to an understanding of the relationship between architectures and network behaviour as a whole. To this end, we introduce IMPLISig, a flexible framework for uncovering a hidden hierarchy of modular loops within the three components of bow-tie architectures in large dynamic networks. As with HEAT, IMPLISig is customise and extensible. For example: changing the subnetwork score function (equation (5.3)) naturally results in changes to the discovered hierarchy, and selection of a score function can be made based on the network and problem at hand.

We found that the deeper a node is in this uncovered feedback loop hierarchy of core nodes, the more significant its impact on the output of the network (section 5.4.2). We show that members of core coherent feedback loops can have a significant effect on the output of the overall network – implying that information flows through the loop hierarchy, being modulated, amplified and attenuated as it does so. We find that inhibition of members of these core loops have an increased significance of expression change over a range of output nodes, compared with the expected change of inhibiting a node in a large strongly connected component. Furthermore, members of these core loops are prime candidates for network control: many nodes in control kernels can be found at deep levels of this loop hierarchy (section 5.4.3). This leads us to conclude that IMPLISig can be used in the future to reduce the search space for novel drug targets in large intra-cellular networks.

Work still needs to be done, however, to characterise the meaningfulness of the uncovered hierarchies of the in- and out-component.

6.2 Future Directions

As our literature review has revealed, there are a number of related works that can be considered for extending the work proposed in this thesis in the future.

6.2.1 Incorporate Edge Features into Hyperbolic Embedding

In chapter 3 and chapter 4, we leverage additional node attributes into a hyperbolic embedding procedure. HEAT is even capable of handling weighted edges through the use of random-walk network sampling. However, as mentioned in section 1.1.3, weighted edges are a single, special case of general edge attributes. Literature has shown that explicit incorporation of edge attributes can improve embedding quality (Goyal et al., 2018a; Goyal et al., 2018b), but, so far, none of these approaches have been applied to hyperbolic space. HEAT, in particular, would be a prime candidate for extension in this way: random-walks provides an overall very flexible sampling method, and extending it to include general edge attributes has already been proposed (Goyal et al., 2018a). As mentioned in section 3.2, the second-order similarity between nodes could be incorporated into the topological similarity matrix using a weighted sum (Wang, Cui, and Zhu, 2016).

6.2.2 Incorporate Structural Features into Hyperbolic Embedding

In addition to microscopic edge features, meso-scopic features – such as community structure or feedback loop – can also be incorporated into network embedding (Wang et al., 2017c). Community structure, in particular, is linked with information flow: gossip flows more easily amongst friends than between friendship groups, for example. Both HEAT and HEADNet could achieve this through the addition of another term to their respective objective functions. Community memberships could be considered node labels and the additional term in the objective would transform the learning into a supervised (or semi-supervised, if only partial memberships are known) embedding problem, where the weighting of the new term would control the trade-off between representing the topological and attribute similarity exactly, or increasing the margins between different communities in the hyperbolic embedding space.

6.2.3 Hyperbolic Embedding of Multi-Layer Networks

Whole networks themselves may form just part of a system as a whole. For example, *multi-layer networks* have recently become popular for jointly modelling multiple *layers* of a system simultaneously (Mucha et al., 2010; Mucha and Porter, 2010). Embedding multi-layer networks is a challenging problem since the links between layers need to be considered differently to intra-layer links, however, consideration of this macro-level network labelling in the embedding process has been shown to produce high quality results (Liu et al., 2017). Generalising either HEAT or HEADNet to simultaneously map nodes withing a multi-layer networks to a single, unified hyperbolic representation is left as future work.

6.2.4 Hyperbolic Neural Networks

Recently, hyperbolic neural networks have been proposed to perform classical neural network operations in (the Poincaré ball model of) hyperbolic space (Ganea, Bécigneul, and Hofmann, 2018). HEAT and HEADNet provide methods for transforming structured data into a form suitable for these operations, as well as for incorporation of data from multiple sources (topology and attributes). This has the immediate consequence that node classification can be performed in native hyperbolic space, using a hyperbolic classifier, rather than by using a Euclidean classifier trained on Klein node positions (as we did in chapter 3). This would, no doubt, result in better quality node classifications, and would likely improve on the node classification performance of HEAT reported in section 3.3.5.

In addition, these operations could be used to augment the loss function for semi-supervised hyperbolic embedding learning, as discussed in section 6.2.2. The generalisation of graph convolution (Defferrard, Bresson, and Vandergheynst, 2016; Kipf and Welling, 2016) to hyperbolic space (Chami et al., 2019) opens the door to many interesting extensions of the works proposed here with respect to supervised and semi-supervised embedding learning.

Finally, we note that HEADNet could be transformed from an embedder of Euclidean attributes to hyperbolic distributions – itself comprised of Euclidean parameters – to a wholly hyperbolic model. It would accept hyperbolic parameters as input,

and output hyperbolic distributions using only operations in hyperbolic space, and optimised using full RSGD. In the case of Euclidean node attributes, perhaps they could be first transformed to hyperbolic attributes using Exp_{μ_0}[1], for example.

6.2.5 Hyperbolic Variational Autoencoder

Previously, Variational Autoencoder (VAE) (Kingma and Welling, 2013) has been derived on the Poincaré disk (Chen, Chen, and Zhang, 2018; Mathieu et al., 2019). As discussed in section 4.2.8, HEADNet has a number of connections to VAEs and the addition of an encoder would allow use to generate meaningful new members of a system, based on the learned distributions of existing members.

6.2.6 Embedding Space Analysis

Another interesting direction for future research is analyses of learned hyperbolic embedding. Given that hyperbolic spaces are continuous approximations of trees (Krioukov et al., 2010), examining the position of elements within the space can provide an insight into the position of that element within a hierarchy. For example, in Nickel and Kiela, 2017, the authors use hyperbolic distance from the origin of the Poincaré disk to infer the hierarchy of nouns. Since we have shown that incorporation of attributes can improve downstream performance on a number of common tasks, it would be interesting to investigate how attributes would affect an entities perceived position in a hierarchy.

Inspired by community detection in hyperbolic space (Bruno et al., 2019; Wang et al., 2019), investigation of hierarchy position can be extended further. As mentioned in section 6.2.2, community structure can be considered explicitly in the hyperbolic embedding process. Explicit consideration of communities in hyperbolic embedding would actually have an interesting extra advantage over embedding to a Euclidean space, namely that community hierarchy is preserved. Communities could be ranked by hyperbolic distance from the origin to give an indication of how central or important a particular module is. Consider a highly modular PPI network: communities formed of highly inter-linked proteins that are embedded close to the origin of the hyperbolic space would, in some sense, 'be more popular'

[1] Recall $\text{Exp}_{\mu_0} : \mathbb{R}^n \to \mathbb{H}^n$. See figure 4.2.

than those further away. Perhaps those modules are more central to the information flow through the network. Moreover, the elements within a particular module can be ranked by position in hierarchy. The ranking of entire communities could be achieved in a number of ways, but the simplest seems to be to compute the *Frećhet mean* (Wilson and Leimeister, 2018) of all of the vectors belonging to that community and then determine the hyperbolic distance of that mean point from the origin. This is left as an interesting future direction.

Naturally, even in the case where community membership or node labels are not known, investigation into the hierarchical position of groups could still be informative. Clustering algorithms (for example: the recently proposed HDBSCAN (McInnes, Healy, and Astels, 2017)) can be applied to cluster the hyperbolic vectors and then these clusters could be ranked by hierarchy in the same way.

6.2.7 Reducing Complexity of Logical Systems

Literature has shown that it is possible to reduce the complexity of boolean networks (Kim et al., 2011a). This is closely linked to chapter 5, where we uncover the hierarchy of information flow. Following on from what is proposed in that chapter, we suggest investigation into constructing boolean functions for super-nodes such that the overall dynamical behaviour of the network does not change. Other natural extensions of IMPLISig are application to continuous dynamical models, as well as developing a library of score functions to uncover a variety of useful hierarchies.

6.2.8 Flow of Information in Hyperbolic Space

This thesis has claimed that there is a natural connection between the hierarchy of elements, the hierarchy of groups of elements within a system, and information flow through that system. Hyperbolic space can capture the hierarchy of elements within a system, and it has been shown to reflect the spread of information between those elements. For example: hyperbolic space can be used to efficiently route packets of information between entities of large networks using only local information (Papadopoulos et al., 2010). A natural question, then, is can a hyperbolic capture the flow of information between groups of nodes? IMPLISig returns a DAG, $\mathbb{H}$, where

each element is a node or super-node and each relationship is a 'participates in' relationship (see section 5.2 for details). A DAG is a perfect candidate for hyperbolic embedding (Sarkar, 2011), and if $\mathbb{H}$ were to be embedded to a hyperbolic space, the principle of hyperbolic packet routing (Papadopoulos et al., 2010) could be applied to potentially reveal the full hierarchical flow of information through a dynamic network.

Appendix A

Supplementary for Chapter 2

A.1 Identifying Modular Structure

A.1.1 Cut- and Spectral-Based Approaches

The flagship Kernighan-Lin algorithm (Kernighan and Lin, 1970) focused on 'cutting' the network into modules, in such a way that the number of edges cut was minimised. However, this often favoured cuts of small, peripheral sub-graphs, so it was adapted into ratio cut (Wei and Cheng, 1991), normalised cut (Shi and Malik, 2000) and min-max cut (Ding et al., 2001) that took the number of nodes in each resulting sub-graph into account, and thus resulted in a partition that was more balanced. Contemporary cut-based approaches are concerned more with edges, rather than vertices and gave rise to a new measure for a good cut, called conductance. Conductance is still prolific in the literature: it has been used to detect communities in bipartite networks (Barber, 2007), combined with PageRank (Page et al., 1999) and used as the basis for a greedy optimisation algorithm (Lancichinetti and Fortunato, 2009) capable of finding overlapping communities at different scales. Spectral clustering dates back to the work of Donath and Hoffman in 1973 (Donath and Hoffman, 1973). However, it was popularised in the early 2000s (Shi and Malik, 2000; Ng, Jordan, Weiss, et al., 2002; Ding, 2004). Spectral methods rely upon constructing Laplacian matrices from the raw network data and eigen-decomposing them. Clustering the resulting eigen-vectors results in clusters of the original data points. Spectral approaches have many advantages over other techniques and, as a result, they have become popular in the machine learning community for clustering on non-linear manifolds. According to Von Luxburg, 2007, 'these methods do not make

assumptions about the form of the clusters' and are capable of correctly identifying typically challenging clusters, such as the famous two spirals example. For community detection, they have the additional benefit of efficiency, especially if the graph adjacency matrix is sparse.

A.1.2 Modularity

Girvan and Newman, 2002 marked a significant advance in the field by providing the first quantitative measure of a community: modularity. The modularity of a partition of a network (defined in equation (1.19)) scores a network partition by comparing the number of links inside a given module with the expected number that would be found in a random graph of the same size and degree sequence. Here, m is the number of modules in the partition, l_s is the number of links in module s, L is the total number of links in the network and d_s is the total degree of the nodes in s. Girvan and Newman propose a hierarchical divisive algorithm that removes edges based on their 'betweenness' (the number of shortest paths from two nodes in the network that go through them) until the modularity quality function is maximised. The early work of Girvan and Newman has since been expanded upon. For example, edge clustering in favour of edge-betweenness (Radicchi et al., 2004), iteratively adding links to a module based on their expected increase in modularity (Clauset, Newman, and Moore, 2004), and multi-stage local optimisation in the popular Louvain algorithm (Blondel et al., 2008).

However, it has been shown that modularity-based approaches have their limitations. Fortunato and Barthelemy, 2007 show what they referred to as the 'resolution limit' - that modularity-based approaches can fail to identify communities that are smaller in size than a scale that depends on the size of the network, and this results in incorrect community division in the cases when even small communities must be considered. Some methods use a 'resolution parameter' to control the scale of the communities detected, however, setting this parameter is difficult as the scale of communities is not known a-priori and the weak definition of a community in general. Additionally, most modularity-based methods rely to some extent on greedy optimisation which has poor accuracy compared to other techniques (Fortunato, 2010).

A.1.3 Finding Modules with Flow

Another definition of community comes from finding subnetworks where flow congregates. The Markov Clustering algorithm (MCL) simulates a diffusion process on a graph by repeatedly performing stages of expansion and inflation and only keeping the k largest elements for efficiency (Van Dongen, 2001). Another significant flow-based approach is the the work of Rosvall and Bergstrom with their famous Infomap algorithm (Rosvall and Bergstrom, 2007) that translated the problem of community detection into the problem of optimally compressing the information in a graph such that the most information can be uncovered when the compression is decoded. They used simulated annealing to minimize a function that represented both compression and data loss resulting in a map that "best captures the community structure with respect to the dynamics on the network" (De Domenico et al., 2015). This approach was also shown to work well with dynamic processes in their later work (Rosvall and Bergstrom, 2008). Like modularity optimisation, computing an exact solution to this problem is NP-hard, which facilitates the need for heuristic algorithms which are computationally expensive (Fortunato, 2010).

A.1.4 Deep Learning for Module Detection

Deep learning has recently taken the computer science community by storm by allowing for computational models that extract features at different levels of abstraction (LeCun, Bengio, and Hinton, 2015). Many complex networks contain highly non-linear features (Yang et al., 2016), making the use of deep non-linear models very appealing. The deep learning community has began to explore the possibilities of using neural networks for clustering in the graph domain – a key advancement for identifying modules of close inter-connectivity. Convolutional neural networks (CNNs), powerful machine learning tools that have proven very successful for challenging classification tasks that have recently been generalized to take a graph input (Defferrard, Bresson, and Vandergheynst, 2016). CNNs have also been used for semi-supervised learning on graphs, where they are capable of learning both graph structure and node features (Kipf and Welling, 2016). However, existing deep learning models suffer from a high number of parameters to tune, when exposing them

to unseen networks.

A.1.5 Hierarchical and Multi-scale Module Detection

In many networks representing complex real-world phenomena, finding a single partition – where each node is assigned to exactly one community – does not accurately reflect the underlying community structure of the data being represented. Sometimes, nodes can belong to more than one community. Methods that uncover the hierarchically organized modular structure of complex networks are of special importance to computational biology as biological systems display nested hierarchy, with clusters embedded in larger clusters (Ashourvan et al., 2019). Lancichinetti, Fortunato, and Kertész, 2009 used greedy optimisation to maximize a fitness function based on conductance with a parameter that controlled the scale of communities detected. While this algorithm did not use modularity in its fitness function and so did not suffer from a resolution limit, as pointed out earlier, greedy optimisation often returns poor results compared to other heuristic searches (Fortunato, 2010). Further work by the same authors expanded upon the framework provided in Lancichinetti, Fortunato, and Kertész, 2009, by replacing the simple fitness function with a function that assessed the statistical significance of communities discovered (Lancichinetti, Radicchi, and Ramasco, 2010). They also propose overcoming the shortcomings of a greedy search using a consensus approach (Lancichinetti and Fortunato, 2012). These later advancements make their work very appealing as they offer a very simple framework that allows users to input their own quality functions based on what kind of modules they are hoping to extract, and overcome the weakness of the greedy search with a consensus approach. The consensus approach also makes the choice of parameter settings more robust (Lancichinetti and Fortunato, 2012).

A.1.6 Multi-layer Module Detection

Community detection within multi-layer network follow from the same principles as the single layer case. Community detection within multi-layer network follow from the same principles as the single layer case. In fact, many classical community

detection algorithms have been generalized for application to multiplayer networks (Mucha et al., 2010). For example, Ashourvan et al., 2019 used a multi scale variation of the Louvain algorithm to detect hierarchical communities in functional brain networks, and De Domenico et al., 2015 generalizes random walking and the map equation (Rosvall and Bergstrom, 2008) to move across multiple layers.

These algorithms can successfully identify closely related sets of nodes, of different types, which, in the context of computational biology can be useful for predictions such as function and interactions. Special cases of multi-layer networks are the multi-slice networks where the nodes are the same for each layer but the topology of the network may differ. Examples include: temporal networks where each layer represents a different time point and networks representing many different types of interactions amongst nodes (Mucha and Porter, 2010).

A.1.7 Finding Modules Considering Attributes

Community detection traditionally only considers only the structure – or topology – of the network at hand. However, often nodes in network may be enriched with additional attributes that are not solely based upon the observed topology of the network. For example, people in a social network may be annotated with preferences such as hobbies and interests and we would expect two people with the same interests to still somehow be similar, even if we do not observe a direct link between them in the network. Within the context of computational biology, this is perhaps even more relevant, due to the vast quantity and variety of data now available, and the successes of integrative models in the past. Integrative models get their name from the principle of integrating observed data (say, gene expression) with prior knowledge (often in the form of a known protein interaction network and/or previously curated functional annotations). Mitra et al., 2013 offers a summary of many the integrative approaches popular in the literature.

One of the most successful integrative approach is the identification of so-called 'active modules'. It is a relatively recent trend within the interdisciplinary fields of network science and translational medicine and aims to augment known physical interactions with observed expression levels to identify connected sub-networks that are maximally differently expressed. While the problem is related to the topology of

the network – the sub-graphs must be connected, for example – biologically relevant modules often do not align with communities based solely on network topology (He et al., 2016). Computing an exact solution is NP-hard (Ideker et al., 2002), so the authors employ a heuristic search based on simulated annealing to search for the maximally scoring sub-graph in the network. Genetic algorithms (GAs) (Klammer et al., 2010), greedy methods (Nacu et al., 2007) and propagation of flow from cancer genes have since been used (Vandin, Upfal, and Raphael, 2011). More recent work has employed a memetic algorithm to ensure connected-ness (Li et al., 2017b); a multi-objective optimisation process to control the trade off between biological activity and functional enrichment of the detected modules (Chen, Liu, and He, 2017); and a cooperative co-evolutionary approach (He et al., 2016). Interestingly, despite the NP-hardness of the problem, it has been shown that, by transforming the above problem into the well known Prize Collecting Stein Tree (PCST) problem, exact solutions can be obtained in reasonable computational time with integer programming (Dittrich et al., 2008).

Appendix B

Supplementary for Chapter 3

B.1 Network Reconstruction

Here we present additional results for network reconstruction for HEAT. We present results for embedding dimension 5 (table B.1), 25 (table B.3) and 50 (table B.5). T-statistics are described in table B.2, table B.4, table B.6 respectively.

B.2 Link Prediction

Here we present additional results for link prediction for HEAT. We present results for embedding dimension 5 (table B.7), 25 (table B.9) and 50 (table B.11). T-statistics are described in table B.8, table B.10, table B.12 respectively.

B.3 Node Classification

Here we present additional results for node classification task for HEAT. We present results for embedding dimension 10 (table B.13), 25 (table B.15) and 50 (table B.17).

TABLE B.1: Summary of network reconstruction for embedding dimension 5. We present AUROC and AP scores, mean average precision (mAP) and precision at k (p@k) for $k \in \{1, 3, 5, 10\}$ to 3 decimal places and rank to 1 decimal place. Rank is the average position that a true edge appears in the list of false edges ranked by distance. For mAP, we rank distances with respect to each node in the network and compute a separate precision score for each node, then report the mean of these precision. For p@k, we report the number of the k closest nodes that are true neighbours. All scores are averaged over 30 random starting seeds. Standard deviation is given in brackets.

	Cora_ML							
	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
N&K	253.6(11.1)	0.985(0.001)	0.984(0.001)	0.644(0.003)	**0.764(0.005)**	0.711(0.005)	0.701(0.005)	0.693(0.006)
AANE	4087.2(34.4)	0.750(0.002)	0.750(0.003)	0.089(0.001)	0.099(0.002)	0.111(0.002)	0.128(0.002)	0.157(0.002)
TADW	984.4(35.8)	0.940(0.002)	0.926(0.003)	0.250(0.005)	0.279(0.006)	0.281(0.004)	0.301(0.005)	0.334(0.008)
ATTRPURE	4691.9(32.6)	0.712(0.002)	0.724(0.003)	0.082(0.001)	0.089(0.002)	0.102(0.002)	0.117(0.002)	0.140(0.002)
DEEPWALK	328.7(13.4)	0.980(0.001)	0.975(0.001)	0.536(0.004)	0.566(0.008)	0.532(0.004)	0.539(0.006)	0.548(0.006)
SAGEGCN	1395.7(130.4)	0.915(0.008)	0.891(0.011)	0.156(0.014)	0.144(0.017)	0.175(0.019)	0.199(0.023)	0.237(0.027)
$HEAT_{\alpha=0.00}$	**89.3(3.7)**	**0.995(0.000)**	**0.993(0.000)**	**0.706(0.004)**	0.731(0.007)	**0.717(0.005)**	**0.723(0.004)**	**0.734(0.004)**
$HEAT_{\alpha=0.20}$	162.7(6.7)	0.990(0.000)	0.988(0.001)	0.628(0.004)	0.666(0.008)	0.640(0.006)	0.648(0.006)	0.661(0.007)
$HEAT_{\alpha=1.00}$	4071.0(73.5)	0.751(0.005)	0.773(0.005)	0.081(0.003)	0.076(0.005)	0.095(0.006)	0.114(0.005)	0.144(0.007)
	Citeseer							
N&K	100.6(7.9)	0.991(0.001)	0.992(0.001)	0.791(0.004)	0.766(0.006)	0.765(0.005)	0.792(0.006)	0.832(0.007)
AANE	3782.1(23.8)	0.646(0.002)	0.640(0.003)	0.078(0.001)	0.069(0.002)	0.089(0.002)	0.114(0.002)	0.164(0.004)
TADW	485.3(11.5)	0.955(0.001)	0.944(0.002)	0.259(0.002)	0.208(0.003)	0.257(0.003)	0.296(0.004)	0.363(0.007)
ATTRPURE	3758.1(24.1)	0.648(0.002)	0.643(0.003)	0.075(0.001)	0.066(0.002)	0.081(0.002)	0.101(0.003)	0.149(0.005)
DEEPWALK	39.4(3.5)	0.996(0.000)	0.995(0.001)	0.729(0.004)	0.646(0.006)	0.706(0.007)	0.756(0.009)	0.805(0.009)
SAGEGCN	985.1(126.5)	0.908(0.012)	0.893(0.013)	0.121(0.011)	0.083(0.009)	0.145(0.018)	0.193(0.024)	0.274(0.040)
$HEAT_{\alpha=0.00}$	**10.2(2.4)**	**0.999(0.000)**	**0.998(0.001)**	**0.806(0.003)**	0.713(0.004)	**0.844(0.004)**	**0.883(0.006)**	**0.931(0.006)**
$HEAT_{\alpha=0.20}$	19.3(2.6)	0.998(0.000)	0.998(0.000)	0.798(0.003)	**0.741(0.006)**	0.758(0.005)	0.790(0.007)	0.830(0.009)
$HEAT_{\alpha=1.00}$	1631.3(24.4)	0.847(0.002)	0.852(0.003)	0.101(0.002)	0.071(0.003)	0.114(0.006)	0.153(0.007)	0.220(0.009)
	Pubmed							
N&K	586.3(20.4)	0.993(0.000)	0.992(0.000)	0.704(0.004)	0.708(0.004)	**0.773(0.002)**	0.810(0.002)	0.816(0.003)
AANE	18340.9(63.1)	0.793(0.001)	0.774(0.001)	0.074(0.000)	0.065(0.001)	0.125(0.001)	0.153(0.001)	0.190(0.001)
TADW	4568.0(35.8)	0.948(0.000)	0.938(0.001)	0.344(0.001)	0.310(0.002)	0.369(0.002)	0.420(0.002)	0.455(0.002)
ATTRPURE	26314.0(72.6)	0.703(0.001)	0.685(0.001)	0.076(0.000)	0.070(0.001)	0.133(0.001)	0.161(0.001)	0.197(0.001)
DEEPWALK	1270.3(31.7)	0.986(0.000)	0.984(0.000)	0.606(0.003)	0.548(0.004)	0.592(0.003)	0.657(0.003)	0.694(0.003)
SAGEGCN	6405.0(311.9)	0.928(0.004)	0.903(0.006)	0.140(0.009)	0.106(0.008)	0.198(0.013)	0.247(0.015)	0.307(0.017)
$HEAT_{\alpha=0.00}$	**262.6(9.1)**	**0.997(0.000)**	**0.996(0.000)**	**0.790(0.003)**	**0.749(0.004)**	0.770(0.003)	**0.818(0.002)**	**0.836(0.002)**
$HEAT_{\alpha=0.20}$	533.4(14.5)	0.994(0.000)	0.993(0.000)	0.688(0.003)	0.624(0.004)	0.697(0.003)	0.761(0.002)	0.788(0.002)
$HEAT_{\alpha=1.00}$	14115.4(405.1)	0.841(0.005)	0.835(0.004)	0.074(0.002)	0.059(0.003)	0.119(0.004)	0.151(0.005)	0.194(0.006)
	PPI							
N&K	**7217.9(48.5)**	**0.934(0.000)**	**0.936(0.001)**	**0.382(0.002)**	**0.752(0.006)**	**0.642(0.004)**	**0.612(0.003)**	**0.601(0.003)**
AANE	50816.3(87.4)	0.536(0.001)	0.588(0.001)	0.085(0.000)	0.463(0.002)	0.242(0.001)	0.195(0.001)	0.163(0.001)
TADW	29118.4(92.1)	0.734(0.001)	0.717(0.001)	0.106(0.001)	0.461(0.002)	0.236(0.002)	0.193(0.001)	0.172(0.001)
ATTRPURE	51410.8(98.1)	0.510(0.001)	0.511(0.001)	0.038(0.000)	0.163(0.004)	0.084(0.001)	0.065(0.001)	0.051(0.001)
DEEPWALK	16651.9(144.2)	0.848(0.001)	0.832(0.001)	0.231(0.001)	0.562(0.005)	0.385(0.004)	0.352(0.004)	0.345(0.004)
SAGEGCN	46554.1(956.1)	0.575(0.009)	0.587(0.008)	0.080(0.002)	0.435(0.003)	0.194(0.003)	0.146(0.004)	0.115(0.004)
$HEAT_{\alpha=0.00}$	7768.1(62.3)	0.929(0.001)	0.923(0.001)	0.298(0.002)	0.603(0.005)	0.472(0.004)	0.450(0.003)	0.455(0.003)
$HEAT_{\alpha=0.20}$	8681.2(76.5)	0.921(0.001)	0.915(0.001)	0.285(0.002)	0.599(0.006)	0.445(0.005)	0.415(0.005)	0.412(0.004)
$HEAT_{\alpha=1.00}$	49309.2(136.1)	0.550(0.001)	0.556(0.001)	0.068(0.000)	0.425(0.002)	0.182(0.002)	0.135(0.002)	0.107(0.001)
	MIT							
N&K	33502.6(342.8)	0.925(0.001)	**0.927(0.001)**	**0.551(0.003)**	**1.000(0.000)**	**0.863(0.003)**	**0.832(0.003)**	**0.807(0.003)**
AANE	163472.8(144.0)	0.633(0.000)	0.627(0.000)	0.178(0.000)	**1.000(0.000)**	0.507(0.001)	0.386(0.001)	0.286(0.001)
TADW	99294.0(352.7)	0.777(0.001)	0.762(0.001)	0.306(0.002)	**1.000(0.000)**	0.651(0.003)	0.571(0.003)	0.497(0.003)
ATTRPURE	174186.9(191.0)	0.608(0.000)	0.611(0.000)	0.176(0.000)	0.990(0.001)	0.530(0.002)	0.413(0.002)	0.306(0.002)
DEEPWALK	50054.3(384.8)	0.887(0.001)	0.873(0.001)	0.448(0.002)	**1.000(0.000)**	0.722(0.004)	0.664(0.004)	0.619(0.004)
SAGEGCN	98421.4(3672.5)	0.779(0.008)	0.769(0.007)	0.321(0.007)	**1.000(0.000)**	0.587(0.010)	0.505(0.011)	0.445(0.011)
$HEAT_{\alpha=0.00}$	**32696.9(169.3)**	**0.926(0.000)**	0.918(0.000)	0.528(0.001)	**1.000(0.000)**	0.792(0.003)	0.748(0.002)	0.716(0.002)
$HEAT_{\alpha=0.20}$	37561.2(395.4)	0.916(0.001)	0.908(0.001)	0.509(0.002)	**1.000(0.000)**	0.773(0.003)	0.725(0.003)	0.689(0.003)
$HEAT_{\alpha=1.00}$	168980.5(1478.5)	0.620(0.003)	0.630(0.003)	0.193(0.002)	**1.000(0.000)**	0.502(0.004)	0.397(0.004)	0.314(0.004)

TABLE B.2: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the network reconstruction task, for embedding dimension 5. For each network, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML							
	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
$\text{HEAT}_{\alpha=0.20}$	**162.7(6.7)**	**0.990(0.000)**	**0.988(0.001)**	0.628(0.004)	0.666(0.008)	0.640(0.006)	0.648(0.006)	0.661(0.007)
N&K	253.6(11.1)	0.985(0.001)	0.984(0.001)	0.644(0.003)	0.764(0.005)	0.711(0.005)	0.701(0.005)	0.693(0.006)
t-statistic	3.84E+01	3.84E+01	2.35E+01	-1.75E+01	-5.70E+01	-4.96E+01	-3.82E+01	-1.96E+01
p-value	5.78E-38	5.78E-38	1.43E-31	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	1	1	1	0	0	0	0	0
	Citeseer							
$\text{HEAT}_{\alpha=0.20}$	**19.3(2.6)**	**0.998(0.000)**	**0.998(0.000)**	**0.798(0.003)**	**0.741(0.006)**	**0.758(0.005)**	**0.790(0.007)**	**0.830(0.009)**
DEEPWALK	39.4(3.5)	0.996(0.000)	0.995(0.001)	0.729(0.004)	0.646(0.006)	0.706(0.007)	0.756(0.009)	0.805(0.009)
t-statistic	2.55E+01	2.55E+01	1.95E+01	7.02E+01	6.37E+01	3.26E+01	1.66E+01	1.05E+01
p-value	4.83E-32	4.83E-32	2.93E-25	1.29E-55	1.46E-55	2.60E-36	1.92E-23	2.99E-15
$p < 0.05$	1	1	1	1	1	1	1	1
	Pubmed							
$\text{HEAT}_{\alpha=0.20}$	**533.4(14.5)**	**0.994(0.000)**	**0.993(0.000)**	0.688(0.003)	0.624(0.004)	0.697(0.003)	0.761(0.002)	0.788(0.002)
N&K	586.3(20.4)	0.993(0.000)	0.992(0.000)	0.704(0.004)	0.708(0.004)	0.773(0.002)	0.810(0.002)	0.816(0.003)
t-statistic	1.16E+01	1.16E+01	5.12E+00	-2.02E+01	-8.65E+01	-1.26E+02	-8.91E+01	-4.90E+01
p-value	2.48E-16	2.48E-16	1.90E-06	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	1	1	1	0	0	0	0	0
	PPI							
$\text{HEAT}_{\alpha=0.20}$	8681.2(76.5)	0.921(0.001)	0.915(0.001)	0.285(0.002)	0.599(0.006)	0.445(0.005)	0.415(0.005)	0.412(0.004)
N&K	7217.9(48.5)	0.934(0.000)	0.936(0.001)	0.382(0.002)	0.752(0.006)	0.642(0.004)	0.612(0.003)	0.601(0.003)
t-statistic	-8.85E+01	-8.85E+01	-1.25E+02	-1.72E+02	-9.88E+01	-1.66E+02	-1.85E+02	-1.98E+02
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0	0	0	0	0
	MIT							
$\text{HEAT}_{\alpha=0.20}$	37561.2(395.4)	0.916(0.001)	0.908(0.001)	0.509(0.002)	1.000(0.000)	0.773(0.003)	0.725(0.003)	0.689(0.003)
N&K	33502.6(342.8)	0.925(0.001)	0.927(0.001)	0.551(0.003)	1.000(0.000)	0.863(0.003)	0.832(0.003)	0.807(0.003)
t-statistic	-4.25E+01	-4.25E+01	-1.11E+02	-6.13E+01	N/A	-1.12E+02	-1.34E+02	-1.54E+02
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00	N/A	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0	0	0	0	0

TABLE B.3: Summary of network reconstruction for embedding dimension 25. We present AUROC and AP scores, mean average precision (mAP) and precision at k (p@k) for $k \in \{1, 3, 5, 10\}$ to 3 decimal places and rank to 1 decimal place. Rank is the average position that a true edge appears in the list of false edges ranked by distance. For mAP, we rank distances with respect to each node in the network and compute a separate precision score for each node, then report the mean of these precision. For p@k, we report the number of the k closest nodes that are true neighbours. All scores are averaged over 30 random starting seeds. Standard deviation is given in brackets.

	Cora_ML							
	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
N&K	196.6(4.8)	0.988(0.000)	0.987(0.000)	0.684(0.003)	0.799(0.006)	0.744(0.003)	0.730(0.003)	0.720(0.004)
AANE	3807.3(29.5)	0.767(0.002)	0.789(0.003)	0.204(0.001)	0.278(0.003)	0.269(0.003)	0.281(0.003)	0.308(0.002)
TADW	333.9(9.8)	0.980(0.001)	0.975(0.001)	0.555(0.002)	0.630(0.006)	0.581(0.004)	0.578(0.004)	0.581(0.004)
ATTRPURE	4580.0(37.7)	0.719(0.002)	0.747(0.003)	0.179(0.001)	0.248(0.005)	0.237(0.002)	0.247(0.003)	0.272(0.003)
DEEPWALK	106.2(6.4)	0.994(0.000)	0.992(0.001)	0.824(0.003)	0.856(0.006)	0.837(0.003)	0.832(0.003)	0.826(0.004)
SAGEGCN	536.7(37.5)	0.967(0.002)	0.967(0.002)	0.505(0.021)	0.544(0.024)	0.574(0.015)	0.586(0.014)	0.596(0.013)
$HEAT_{\alpha=0.00}$	30.2(3.3)	0.998(0.000)	0.997(0.000)	0.888(0.002)	0.884(0.004)	0.926(0.003)	**0.934(0.002)**	**0.933(0.002)**
$HEAT_{\alpha=0.20}$	**24.8(2.3)**	**0.999(0.000)**	**0.998(0.000)**	**0.949(0.001)**	**0.982(0.002)**	**0.944(0.002)**	0.926(0.002)	0.912(0.003)
$HEAT_{\alpha=1.00}$	3212.2(63.4)	0.803(0.004)	0.832(0.004)	0.197(0.003)	0.267(0.007)	0.266(0.006)	0.282(0.007)	0.296(0.007)
	Citeseer							
N&K	53.6(3.1)	0.995(0.000)	0.995(0.000)	0.804(0.003)	0.779(0.005)	0.776(0.005)	0.805(0.004)	0.840(0.005)
AANE	3651.8(24.4)	0.658(0.002)	0.653(0.003)	0.112(0.001)	0.101(0.002)	0.127(0.003)	0.156(0.003)	0.200(0.004)
TADW	130.7(6.0)	0.988(0.001)	0.984(0.001)	0.531(0.003)	0.453(0.005)	0.488(0.004)	0.526(0.006)	0.597(0.007)
ATTRPURE	3727.6(28.9)	0.651(0.003)	0.645(0.003)	0.105(0.001)	0.098(0.003)	0.122(0.002)	0.147(0.003)	0.200(0.006)
DEEPWALK	13.8(2.0)	0.999(0.000)	0.998(0.001)	0.834(0.003)	0.761(0.004)	0.855(0.003)	0.899(0.005)	0.942(0.005)
SAGEGCN	204.2(64.5)	0.981(0.006)	0.980(0.005)	0.439(0.037)	0.330(0.028)	0.543(0.034)	0.637(0.033)	0.735(0.030)
$HEAT_{\alpha=0.00}$	10.4(2.3)	0.999(0.000)	0.999(0.000)	0.833(0.002)	0.746(0.004)	0.907(0.004)	0.943(0.004)	0.976(0.003)
$HEAT_{\alpha=0.20}$	**4.2(1.1)**	**1.000(0.000)**	**1.000(0.000)**	**0.974(0.001)**	**0.960(0.003)**	**0.957(0.003)**	**0.963(0.003)**	**0.981(0.004)**
$HEAT_{\alpha=1.00}$	1626.8(23.6)	0.848(0.002)	0.867(0.002)	0.191(0.002)	0.163(0.003)	0.223(0.005)	0.273(0.005)	0.338(0.009)
	Pubmed							
N&K	407.9(11.1)	0.995(0.000)	0.994(0.000)	0.750(0.002)	0.754(0.003)	0.814(0.002)	0.843(0.002)	0.841(0.001)
AANE	18364.4(51.7)	0.793(0.001)	0.810(0.001)	0.142(0.001)	0.156(0.002)	0.251(0.002)	0.284(0.001)	0.313(0.001)
TADW	1564.6(21.7)	0.982(0.000)	0.981(0.000)	0.617(0.002)	0.615(0.002)	0.651(0.002)	0.692(0.002)	0.702(0.002)
ATTRPURE	26544.5(129.1)	0.701(0.001)	0.721(0.001)	0.136(0.001)	0.148(0.002)	0.242(0.002)	0.275(0.002)	0.304(0.002)
DEEPWALK	341.1(9.4)	0.996(0.000)	0.996(0.000)	0.891(0.001)	0.866(0.002)	0.911(0.002)	0.925(0.001)	0.927(0.001)
SAGEGCN	2363.7(336.7)	0.973(0.004)	0.972(0.003)	0.471(0.031)	0.421(0.027)	0.551(0.021)	0.618(0.020)	0.664(0.017)
$HEAT_{\alpha=0.00}$	63.3(4.7)	0.999(0.000)	0.998(0.000)	0.900(0.001)	0.836(0.002)	**0.967(0.001)**	**0.976(0.001)**	**0.975(0.001)**
$HEAT_{\alpha=0.20}$	**54.0(3.5)**	**0.999(0.000)**	**0.999(0.000)**	**0.979(0.000)**	**0.988(0.001)**	0.966(0.001)	0.966(0.001)	0.963(0.001)
$HEAT_{\alpha=1.00}$	11864.9(238.1)	0.866(0.003)	0.865(0.002)	0.111(0.003)	0.105(0.004)	0.190(0.005)	0.228(0.006)	0.272(0.007)
	PPI							
N&K	6625.6(37.9)	0.940(0.000)	0.941(0.000)	0.430(0.001)	0.807(0.005)	0.709(0.003)	0.673(0.002)	0.648(0.002)
AANE	48893.8(81.1)	0.554(0.001)	0.600(0.001)	0.097(0.000)	0.487(0.002)	0.272(0.001)	0.216(0.001)	0.178(0.001)
TADW	20748.6(82.5)	0.811(0.001)	0.797(0.001)	0.201(0.001)	0.574(0.003)	0.392(0.003)	0.354(0.002)	0.338(0.001)
ATTRPURE	51271.6(94.5)	0.512(0.001)	0.512(0.001)	0.037(0.000)	0.165(0.004)	0.085(0.002)	0.065(0.001)	0.050(0.001)
DEEPWALK	4888.4(48.4)	0.955(0.000)	0.952(0.000)	0.630(0.002)	0.919(0.004)	0.831(0.003)	0.784(0.003)	0.735(0.002)
SAGEGCN	41412.4(1209.8)	0.622(0.011)	0.643(0.011)	0.131(0.007)	0.510(0.010)	0.289(0.013)	0.232(0.012)	0.189(0.012)
$HEAT_{\alpha=0.00}$	**2703.6(25.4)**	**0.975(0.000)**	**0.973(0.000)**	0.667(0.002)	0.936(0.004)	**0.895(0.002)**	**0.856(0.002)**	**0.809(0.002)**
$HEAT_{\alpha=0.20}$	3314.2(38.9)	0.970(0.000)	0.968(0.000)	**0.673(0.002)**	**0.959(0.002)**	0.885(0.003)	0.839(0.003)	0.786(0.002)
$HEAT_{\alpha=1.00}$	46377.0(127.0)	0.577(0.001)	0.582(0.001)	0.070(0.000)	0.426(0.002)	0.186(0.002)	0.139(0.001)	0.113(0.001)
	MIT							
N&K	32300.6(82.1)	0.927(0.000)	0.930(0.000)	0.567(0.001)	**1.000(0.000)**	0.889(0.001)	0.861(0.001)	0.832(0.001)
AANE	152088.1(155.3)	0.658(0.000)	0.652(0.000)	0.198(0.000)	**1.000(0.000)**	0.550(0.002)	0.440(0.001)	0.343(0.001)
TADW	68191.8(200.5)	0.847(0.000)	0.844(0.001)	0.464(0.001)	**1.000(0.000)**	0.816(0.002)	0.765(0.001)	0.713(0.001)
ATTRPURE	173328.8(150.9)	0.610(0.000)	0.616(0.000)	0.190(0.000)	0.994(0.001)	0.552(0.002)	0.437(0.001)	0.332(0.001)
DEEPWALK	20138.5(123.9)	0.955(0.000)	0.953(0.000)	0.719(0.001)	**1.000(0.000)**	0.949(0.001)	0.927(0.001)	0.898(0.001)
SAGEGCN	81151.9(2192.3)	0.818(0.005)	0.829(0.005)	0.463(0.009)	**1.000(0.000)**	0.783(0.009)	0.726(0.010)	0.674(0.011)
$HEAT_{\alpha=0.00}$	**16898.0(72.7)**	**0.962(0.000)**	**0.959(0.000)**	**0.740(0.001)**	1.000(0.000)	0.956(0.001)	0.943(0.001)	**0.924(0.001)**
$HEAT_{\alpha=0.20}$	19954.6(94.6)	0.955(0.000)	0.955(0.000)	0.739(0.001)	**1.000(0.000)**	**0.966(0.001)**	**0.946(0.001)**	0.920(0.001)
$HEAT_{\alpha=1.00}$	155622.6(816.3)	0.650(0.002)	0.674(0.002)	0.258(0.001)	**1.000(0.000)**	0.655(0.002)	0.568(0.003)	0.485(0.002)

TABLE B.4: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the network reconstruction task, for embedding dimension 25. For each network, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML							
	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
$\text{HEAT}_{\alpha=0.20}$	**24.8(2.3)**	**0.999(0.000)**	**0.998(0.000)**	**0.949(0.001)**	**0.982(0.002)**	**0.944(0.002)**	**0.926(0.002)**	**0.912(0.003)**
DEEPWALK	106.2(6.4)	0.994(0.000)	0.992(0.001)	0.824(0.003)	0.856(0.006)	0.837(0.003)	0.832(0.003)	0.826(0.004)
t-statistic	6.61E+01	6.61E+01	4.56E+01	1.95E+02	1.14E+02	1.42E+02	1.32E+02	9.88E+01
p-value	9.29E-40	9.29E-40	2.20E-34	3.81E-58	7.81E-48	1.89E-70	5.91E-68	2.33E-65
$p < 0.05$	1	1	1	1	1	1	1	1
	Citeseer							
$\text{HEAT}_{\alpha=0.20}$	**4.2(1.1)**	**1.000(0.000)**	**1.000(0.000)**	**0.974(0.001)**	**0.960(0.003)**	**0.957(0.003)**	**0.963(0.003)**	**0.981(0.004)**
DEEPWALK	13.8(2.0)	0.999(0.000)	0.998(0.001)	0.834(0.003)	0.761(0.004)	0.855(0.003)	0.899(0.005)	0.942(0.005)
t-statistic	2.31E+01	2.31E+01	1.73E+01	2.72E+02	2.31E+02	1.23E+02	6.21E+01	3.64E+01
p-value	5.52E-27	5.52E-27	2.31E-19	3.54E-69	7.85E-78	3.91E-72	3.98E-50	1.05E-40
$p < 0.05$	1	1	1	1	1	1	1	1
	Pubmed							
$\text{HEAT}_{\alpha=0.20}$	**54.0(3.5)**	**0.999(0.000)**	**0.999(0.000)**	**0.979(0.000)**	**0.988(0.001)**	**0.966(0.001)**	**0.966(0.001)**	**0.963(0.001)**
DEEPWALK	341.1(9.4)	0.996(0.000)	0.996(0.000)	0.891(0.001)	0.866(0.002)	0.911(0.002)	0.925(0.001)	0.927(0.001)
t-statistic	1.57E+02	1.57E+02	7.14E+01	3.50E+02	2.67E+02	1.59E+02	1.43E+02	1.29E+02
p-value	4.65E-54	4.65E-54	1.21E-37	1.75E-64	2.93E-62	2.06E-65	4.32E-75	1.88E-71
$p < 0.05$	1	1	1	1	1	1	1	1
	PPI							
$\text{HEAT}_{\alpha=0.20}$	**3314.2(38.9)**	**0.970(0.000)**	**0.968(0.000)**	**0.673(0.002)**	**0.959(0.002)**	**0.885(0.003)**	**0.839(0.003)**	**0.786(0.002)**
DEEPWALK	4888.4(48.4)	0.955(0.000)	0.952(0.000)	0.630(0.002)	0.919(0.004)	0.831(0.003)	0.784(0.003)	0.735(0.002)
t-statistic	1.39E+02	1.39E+02	1.47E+02	1.03E+02	5.00E+01	6.98E+01	7.57E+01	1.00E+02
p-value	1.87E-72	1.87E-72	3.79E-72	1.04E-67	3.01E-43	6.03E-58	3.38E-59	3.20E-66
$p < 0.05$	1	1	1	1	1	1	1	1
	MIT							
$\text{HEAT}_{\alpha=0.20}$	**19954.6(94.6)**	**0.955(0.000)**	**0.955(0.000)**	**0.739(0.001)**	1.000(0.000)	**0.966(0.001)**	**0.946(0.001)**	**0.920(0.001)**
DEEPWALK	20138.5(123.9)	0.955(0.000)	0.953(0.000)	0.719(0.001)	1.000(0.000)	0.949(0.001)	0.927(0.001)	0.898(0.001)
t-statistic	6.46E+00	6.46E+00	2.90E+01	9.70E+01	N/A	4.52E+01	5.57E+01	7.41E+01
p-value	1.51E-08	1.51E-08	9.90E-34	3.41E-65	N/A	9.31E-47	1.81E-51	2.26E-57
$p < 0.05$	1	1	1	1	0	1	1	1

TABLE B.5: Summary of network reconstruction for embedding dimension 50. We present AUROC and AP scores, mean average precision (mAP) and precision at k (p@k) for $k \in \{1, 3, 5, 10\}$ to 3 decimal places and rank to 1 decimal place. Rank is the average position that a true edge appears in the list of false edges ranked by distance. For mAP, we rank distances with respect to each node in the network and compute a separate precision score for each node, then report the mean of these precision. For p@k, we report the number of the k closest nodes that are true neighbours. All scores are averaged over 30 random starting seeds. Standard deviation is given in brackets.

	Cora_ML							
	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
N&K	192.4(5.1)	0.988(0.000)	0.987(0.000)	0.689(0.002)	0.803(0.004)	0.748(0.004)	0.733(0.003)	0.722(0.004)
AANE	3779.5(28.0)	0.768(0.002)	0.788(0.002)	0.227(0.001)	0.320(0.004)	0.302(0.003)	0.308(0.002)	0.321(0.003)
TADW	227.7(7.4)	0.986(0.000)	0.984(0.001)	0.668(0.002)	0.738(0.006)	0.692(0.004)	0.684(0.003)	0.684(0.004)
ATTRPURE	4706.9(36.8)	0.712(0.002)	0.738(0.003)	0.198(0.001)	0.286(0.005)	0.264(0.003)	0.268(0.003)	0.283(0.003)
DEEPWALK	67.1(3.9)	0.996(0.000)	0.995(0.001)	0.876(0.002)	0.896(0.004)	0.885(0.003)	0.879(0.003)	0.874(0.002)
SAGEGCN	427.5(37.4)	0.974(0.002)	0.974(0.002)	0.574(0.025)	0.614(0.030)	0.638(0.016)	0.646(0.012)	0.652(0.013)
$HEAT_{\alpha=0.00}$	31.3(3.5)	0.998(0.000)	0.997(0.000)	0.888(0.002)	0.882(0.003)	0.927(0.002)	0.942(0.002)	0.946(0.002)
$HEAT_{\alpha=0.20}$	**17.4(1.9)**	**0.999(0.000)**	**0.999(0.000)**	**0.973(0.001)**	**0.992(0.001)**	**0.971(0.002)**	**0.961(0.002)**	**0.949(0.002)**
$HEAT_{\alpha=1.00}$	3087.8(63.6)	0.811(0.004)	0.839(0.004)	0.228(0.002)	0.316(0.005)	0.310(0.004)	0.322(0.004)	0.327(0.005)
	Citeseer							
N&K	49.8(2.8)	0.995(0.000)	0.996(0.000)	0.804(0.003)	0.779(0.006)	0.777(0.004)	0.807(0.005)	0.841(0.006)
AANE	3476.1(25.2)	0.674(0.002)	0.671(0.003)	0.128(0.001)	0.118(0.002)	0.154(0.003)	0.188(0.004)	0.238(0.005)
TADW	59.2(4.6)	0.995(0.000)	0.992(0.001)	0.636(0.002)	0.545(0.004)	0.576(0.006)	0.613(0.005)	0.687(0.008)
ATTRPURE	3585.6(27.8)	0.664(0.003)	0.662(0.003)	0.120(0.001)	0.114(0.003)	0.147(0.003)	0.179(0.004)	0.229(0.005)
DEEPWALK	9.3(1.7)	0.999(0.000)	0.999(0.000)	0.859(0.002)	0.792(0.004)	0.884(0.003)	0.920(0.004)	0.960(0.003)
SAGEGCN	122.2(35.6)	0.989(0.003)	0.988(0.003)	0.532(0.040)	0.406(0.038)	0.617(0.032)	0.704(0.025)	0.790(0.018)
$HEAT_{\alpha=0.00}$	10.3(2.3)	0.999(0.000)	0.999(0.000)	0.838(0.002)	0.753(0.004)	0.910(0.004)	0.945(0.004)	0.977(0.003)
$HEAT_{\alpha=0.20}$	**4.0(1.1)**	**1.000(0.000)**	**1.000(0.000)**	**0.973(0.001)**	**0.956(0.002)**	**0.967(0.003)**	**0.977(0.002)**	**0.989(0.002)**
$HEAT_{\alpha=1.00}$	1654.4(23.3)	0.845(0.002)	0.866(0.002)	0.198(0.002)	0.171(0.003)	0.230(0.004)	0.277(0.004)	0.334(0.006)
	Pubmed							
N&K	398.4(10.8)	0.996(0.000)	0.994(0.000)	0.754(0.002)	0.759(0.003)	0.817(0.002)	0.845(0.002)	0.843(0.002)
AANE	19395.7(50.7)	0.781(0.001)	0.805(0.001)	0.157(0.001)	0.178(0.002)	0.280(0.002)	0.310(0.002)	0.334(0.001)
TADW	1161.6(14.4)	0.987(0.000)	0.987(0.000)	0.712(0.001)	0.708(0.002)	0.729(0.002)	0.767(0.002)	0.778(0.002)
ATTRPURE	27769.6(96.6)	0.687(0.001)	0.709(0.001)	0.151(0.001)	0.172(0.002)	0.271(0.001)	0.303(0.001)	0.327(0.002)
DEEPWALK	256.0(7.9)	0.997(0.000)	0.997(0.000)	0.913(0.001)	0.881(0.002)	0.944(0.001)	0.951(0.001)	0.952(0.001)
SAGEGCN	1784.6(58.5)	0.980(0.001)	0.979(0.001)	0.564(0.012)	0.516(0.013)	0.628(0.008)	0.694(0.006)	0.729(0.005)
$HEAT_{\alpha=0.00}$	74.6(5.1)	0.999(0.000)	0.998(0.000)	0.872(0.001)	0.787(0.002)	0.962(0.001)	0.981(0.001)	0.985(0.001)
$HEAT_{\alpha=0.20}$	**34.4(3.3)**	**1.000(0.000)**	**1.000(0.000)**	**0.990(0.000)**	**0.994(0.001)**	**0.985(0.001)**	**0.984(0.001)**	**0.982(0.001)**
$HEAT_{\alpha=1.00}$	11976.6(64.5)	0.865(0.001)	0.865(0.001)	0.112(0.001)	0.106(0.002)	0.193(0.003)	0.231(0.003)	0.275(0.002)
	PPI							
N&K	6629.2(38.9)	0.940(0.000)	0.941(0.000)	0.430(0.001)	0.804(0.004)	0.707(0.002)	0.673(0.002)	0.648(0.002)
AANE	46248.1(71.3)	0.578(0.001)	0.616(0.001)	0.106(0.000)	0.497(0.001)	0.282(0.001)	0.226(0.001)	0.190(0.001)
TADW	20453.7(64.3)	0.813(0.001)	0.798(0.001)	0.229(0.001)	0.597(0.003)	0.423(0.001)	0.388(0.002)	0.377(0.001)
ATTRPURE	51040.1(85.3)	0.508(0.001)	0.510(0.001)	0.037(0.000)	0.164(0.004)	0.084(0.002)	0.065(0.001)	0.049(0.001)
DEEPWALK	3182.6(32.5)	0.971(0.000)	0.968(0.000)	0.787(0.001)	0.976(0.002)	0.941(0.002)	0.909(0.002)	0.859(0.002)
SAGEGCN	39875.7(1152.2)	0.636(0.011)	0.660(0.011)	0.153(0.008)	0.547(0.012)	0.333(0.015)	0.272(0.014)	0.222(0.014)
$HEAT_{\alpha=0.00}$	**1542.2(27.1)**	**0.986(0.000)**	**0.984(0.000)**	**0.783(0.002)**	**0.960(0.002)**	**0.953(0.002)**	**0.930(0.002)**	**0.895(0.002)**
$HEAT_{\alpha=0.20}$	2089.2(30.2)	0.981(0.000)	0.980(0.000)	0.789(0.002)	0.988(0.002)	0.952(0.002)	0.921(0.002)	0.876(0.002)
$HEAT_{\alpha=1.00}$	45799.9(112.5)	0.582(0.001)	0.588(0.001)	0.071(0.000)	0.427(0.002)	0.189(0.002)	0.143(0.002)	0.118(0.001)
	MIT							
N&K	32270.3(80.7)	0.927(0.000)	0.930(0.000)	0.566(0.001)	**1.000(0.000)**	0.889(0.001)	0.862(0.001)	0.833(0.001)
AANE	143678.5(158.9)	0.677(0.000)	0.673(0.000)	0.211(0.000)	**1.000(0.000)**	0.577(0.001)	0.468(0.001)	0.367(0.001)
TADW	52131.6(115.9)	0.883(0.000)	0.876(0.000)	0.508(0.000)	**1.000(0.000)**	0.831(0.002)	0.782(0.001)	0.731(0.001)
ATTRPURE	175814.3(153.0)	0.605(0.000)	0.614(0.000)	0.194(0.000)	0.994(0.001)	0.553(0.002)	0.440(0.001)	0.334(0.001)
DEEPWALK	15413.2(79.0)	0.965(0.000)	0.964(0.000)	0.795(0.001)	**1.000(0.000)**	0.977(0.001)	0.962(0.001)	0.939(0.001)
SAGEGCN	78197.7(1836.1)	0.824(0.004)	0.839(0.004)	0.495(0.009)	**1.000(0.000)**	0.818(0.005)	0.764(0.006)	0.712(0.007)
$HEAT_{\alpha=0.00}$	12818.5(87.0)	**0.971(0.000)**	**0.968(0.000)**	0.791(0.001)	**1.000(0.000)**	0.967(0.001)	0.959(0.001)	0.946(0.001)
$HEAT_{\alpha=0.20}$	14822.5(83.5)	0.967(0.000)	0.965(0.000)	**0.801(0.001)**	**1.000(0.000)**	**0.983(0.001)**	**0.970(0.001)**	**0.949(0.001)**
$HEAT_{\alpha=1.00}$	153389.2(640.4)	0.655(0.001)	0.682(0.001)	0.271(0.001)	**1.000(0.000)**	0.674(0.002)	0.586(0.002)	0.503(0.002)

TABLE B.6: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the network reconstruction task, for embedding dimension 50. For each network, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML							
	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
$\text{HEAT}_{\alpha=0.20}$	**17.4(1.9)**	**0.999(0.000)**	**0.999(0.000)**	**0.973(0.001)**	**0.992(0.001)**	**0.971(0.002)**	**0.961(0.002)**	**0.949(0.002)**
DEEPWALK	67.1(3.9)	0.996(0.000)	0.995(0.001)	0.876(0.002)	0.896(0.004)	0.885(0.003)	0.879(0.003)	0.874(0.002)
t-statistic	6.33E+01	6.33E+01	4.06E+01	1.99E+02	1.37E+02	1.36E+02	1.26E+02	1.21E+02
p-value	1.53E-43	1.53E-43	6.64E-34	6.62E-66	5.24E-52	2.40E-67	2.04E-68	1.53E-71
$p < 0.05$	1	1	1	1	1	1	1	1
	Citeseer							
$\text{HEAT}_{\alpha=0.20}$	**4.0(1.1)**	**1.000(0.000)**	**1.000(0.000)**	**0.973(0.001)**	**0.956(0.002)**	**0.967(0.003)**	**0.977(0.002)**	**0.989(0.002)**
DEEPWALK	9.3(1.7)	0.999(0.000)	0.999(0.000)	0.859(0.002)	0.792(0.004)	0.884(0.003)	0.920(0.004)	0.960(0.003)
t-statistic	1.42E+01	1.42E+01	1.19E+01	2.63E+02	1.97E+02	1.05E+02	7.27E+01	4.00E+01
p-value	3.53E-19	3.53E-19	4.65E-15	5.68E-69	1.18E-66	9.26E-68	3.02E-51	1.09E-40
$p < 0.05$	1	1	1	1	1	1	1	1
	Pubmed							
$\text{HEAT}_{\alpha=0.20}$	**34.4(3.3)**	**1.000(0.000)**	**1.000(0.000)**	**0.990(0.000)**	**0.994(0.001)**	**0.985(0.001)**	**0.984(0.001)**	**0.982(0.001)**
DEEPWALK	256.0(7.9)	0.997(0.000)	0.997(0.000)	0.913(0.001)	0.881(0.002)	0.944(0.001)	0.951(0.001)	0.952(0.001)
t-statistic	1.41E+02	1.41E+02	6.00E+01	3.91E+02	2.64E+02	1.70E+02	1.51E+02	1.25E+02
p-value	1.90E-54	1.90E-54	2.01E-35	3.93E-66	1.16E-58	3.24E-75	2.55E-77	2.20E-68
$p < 0.05$	1	1	1	1	1	1	1	1
	PPI							
$\text{HEAT}_{\alpha=0.20}$	**2089.2(30.2)**	**0.981(0.000)**	**0.980(0.000)**	**0.789(0.002)**	**0.988(0.002)**	**0.952(0.002)**	**0.921(0.002)**	**0.876(0.002)**
DEEPWALK	3182.6(32.5)	0.971(0.000)	0.968(0.000)	0.787(0.001)	0.976(0.002)	0.941(0.002)	0.909(0.002)	0.859(0.002)
t-statistic	1.35E+02	1.35E+02	1.20E+02	6.29E+00	2.24E+01	1.91E+01	2.30E+01	3.41E+01
p-value	3.63E-74	3.63E-74	9.33E-71	2.54E-08	4.15E-28	2.32E-26	6.19E-31	2.58E-40
$p < 0.05$	1	1	1	1	1	1	1	1
	MIT							
$\text{HEAT}_{\alpha=0.20}$	**14822.5(83.5)**	**0.967(0.000)**	**0.965(0.000)**	**0.801(0.001)**	1.000(0.000)	**0.983(0.001)**	**0.970(0.001)**	**0.949(0.001)**
DEEPWALK	15413.2(79.0)	0.965(0.000)	0.964(0.000)	0.795(0.001)	1.000(0.000)	0.977(0.001)	0.962(0.001)	0.939(0.001)
t-statistic	2.81E+01	2.81E+01	2.13E+01	3.06E+01	N/A	2.76E+01	2.44E+01	3.98E+01
p-value	9.86E-36	9.86E-36	2.26E-29	1.88E-37	N/A	6.82E-30	2.79E-31	9.63E-43
$p < 0.05$	1	1	1	1	0	1	1	1

TABLE B.7: Summary of link prediction for embedding dimension 5. We present AUROC, AP, and mean average precision (mAP) to 3 decimal places and rank to 1 decimal place. Rank is the average position that a true edge appears in the list of false edges ranked by distance. For mAP, we rank distances with respect to each node in the network and compute a separate precision score for each node, then report the mean. All scores are averaged over 30 random starting seeds. Standard deviation is given in brackets.

	Cora_ML				PPI			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	106.7(9.2)	0.914(0.008)	0.928(0.007)	0.226(0.010)	762.6(21.6)	0.907(0.003)	0.914(0.002)	0.168(0.005)
AANE	316.5(13.0)	0.742(0.011)	0.743(0.012)	0.062(0.005)	3841.6(38.3)	0.533(0.005)	0.577(0.005)	0.069(0.004)
TADW	118.9(9.2)	0.904(0.007)	0.890(0.010)	0.114(0.006)	2500.8(47.6)	0.696(0.006)	0.687(0.006)	0.072(0.004)
ATTRPURE	350.0(11.4)	0.715(0.009)	0.727(0.011)	0.060(0.005)	3864.5(38.9)	0.509(0.005)	0.511(0.004)	0.018(0.002)
DEEPWALK	180.0(11.0)	0.854(0.009)	0.888(0.008)	0.170(0.008)	1403.9(25.6)	0.829(0.003)	0.820(0.003)	0.110(0.004)
SAGEGCN	202.9(29.3)	0.835(0.024)	0.821(0.025)	0.061(0.008)	3692.3(87.1)	0.551(0.011)	0.564(0.009)	0.065(0.004)
$HEAT_{\alpha=0.00}$	23.1(3.6)	0.982(0.003)	0.985(0.002)	**0.486(0.009)**	**610.0(15.4)**	**0.926(0.002)**	**0.921(0.002)**	**0.171(0.006)**
$HEAT_{\alpha=0.20}$	**18.2(2.1)**	**0.986(0.002)**	**0.985(0.002)**	0.421(0.010)	686.1(16.1)	0.917(0.002)	0.911(0.002)	0.158(0.006)
$HEAT_{\alpha=1.00}$	309.7(13.5)	0.748(0.011)	0.769(0.012)	0.056(0.005)	3745.2(37.5)	0.545(0.005)	0.553(0.005)	0.064(0.004)
	Citeseer				MIT			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	158.5(8.5)	0.803(0.011)	0.837(0.008)	0.187(0.011)	2806.7(43.5)	0.916(0.001)	0.919(0.001)	0.247(0.004)
AANE	288.4(13.2)	0.641(0.016)	0.637(0.017)	0.071(0.007)	12319.9(77.6)	0.631(0.002)	0.625(0.003)	0.094(0.003)
TADW	86.8(6.3)	0.893(0.008)	0.874(0.010)	0.105(0.009)	7206.2(86.8)	0.784(0.003)	0.771(0.003)	0.140(0.003)
ATTRPURE	283.3(11.4)	0.648(0.014)	0.644(0.015)	0.070(0.006)	13046.4(65.1)	0.609(0.002)	0.611(0.002)	0.095(0.003)
DEEPWALK	236.7(10.2)	0.706(0.013)	0.792(0.009)	0.189(0.010)	4017.5(51.3)	0.880(0.002)	0.865(0.002)	0.178(0.003)
SAGEGCN	170.6(17.5)	0.788(0.022)	0.793(0.023)	0.069(0.010)	7882.2(415.1)	0.764(0.012)	0.755(0.009)	0.129(0.004)
$HEAT_{\alpha=0.00}$	18.5(2.7)	0.978(0.003)	0.974(0.006)	**0.766(0.015)**	**2755.1(33.4)**	**0.917(0.001)**	**0.909(0.001)**	**0.216(0.003)**
$HEAT_{\alpha=0.20}$	**7.5(1.6)**	**0.992(0.002)**	**0.993(0.002)**	0.712(0.015)	3078.7(40.0)	0.908(0.001)	0.901(0.001)	0.205(0.003)
$HEAT_{\alpha=1.00}$	124.7(6.7)	0.846(0.008)	0.851(0.011)	0.094(0.008)	12903.6(119.7)	0.613(0.004)	0.624(0.003)	0.098(0.003)
	Pubmed				Mean Ranks			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	916.4(34.4)	0.862(0.005)	0.885(0.004)	0.180(0.005)	3.4	3.4	2.6	2.6
AANE	1376.7(27.8)	0.793(0.004)	0.775(0.004)	0.061(0.002)	7.6	7.6	7.6	7.2
TADW	751.2(41.2)	0.887(0.006)	0.870(0.007)	0.096(0.005)	3.8	3.8	4	5
ATTRPURE	1977.3(32.0)	0.703(0.005)	0.685(0.005)	0.063(0.002)	8.8	8.8	8.8	8
DEEPWALK	1725.2(32.9)	0.741(0.005)	0.828(0.004)	0.140(0.004)	5.6	5.6	5.4	3.8
SAGEGCN	799.9(56.1)	0.880(0.008)	0.860(0.007)	0.066(0.004)	5.4	5.4	5.8	7
$HEAT_{\alpha=0.00}$	1546.1(30.7)	0.768(0.005)	0.854(0.003)	0.205(0.004)	2.6	2.6	2.2	**1.4**
$HEAT_{\alpha=0.20}$	**309.4(11.9)**	**0.954(0.002)**	**0.953(0.002)**	**0.209(0.004)**	**1.6**	**1.6**	**2**	2.2
$HEAT_{\alpha=1.00}$	1070.1(38.2)	0.839(0.006)	0.835(0.006)	0.059(0.003)	6.2	6.2	6.6	7.8

TABLE B.8: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the link prediction task, for embedding dimension 5. For each network, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML			
	Mean Rank	AUROC	AP	mAP
$\text{HEAT}_{\alpha=0.20}$	**18.2(2.1)**	**0.986(0.002)**	**0.985(0.002)**	**0.421(0.010)**
N&K	106.7(9.2)	0.914(0.008)	0.928(0.007)	0.226(0.010)
t-statistic	5.15E+01	5.15E+01	4.60E+01	7.46E+01
p-value	1.44E-32	1.44E-32	1.29E-32	2.24E-59
$p < 0.05$	1	1	1	1
	Citeseer			
$\text{HEAT}_{\alpha=0.20}$	**7.5(1.6)**	**0.992(0.002**)	**0.993(0.002)**	**0.712(0.015)**
TADW	86.8(6.3)	0.893(0.008)	0.874(0.010)	0.105(0.009)
t-statistic	6.70E+01	6.70E+01	6.65E+01	1.95E+02
p-value	9.46E-37	9.41E-37	6.87E-36	4.18E-71
$p < 0.05$	1	1	1	1
	Pubmed			
$\text{HEAT}_{\alpha=0.20}$	**309.4(11.9)**	**0.954(0.002)**	**0.953(0.002)**	**0.209(0.004)**
N&K	916.4(34.4)	0.862(0.005)	0.885(0.004)	0.180(0.005)
t-statistic	9.14E+01	9.14E+01	8.03E+01	2.58E+01
p-value	2.17E-44	2.17E-44	8.74E-48	9.54E-33
$p < 0.05$	1	1	1	1
	PPI			
$\text{HEAT}_{\alpha=0.20}$	**686.1(16.1)**	**0.917(0.002)**	0.911(0.002)	0.158(0.006)
N&K	762.6(21.6)	0.907(0.003)	0.914(0.002)	0.168(0.005)
t-statistic	1.56E+01	1.56E+01	-3.86E+00	-7.27E+00
p-value	7.09E-22	7.09E-22	1.00E+00	1.00E+00
$p < 0.05$	1	1	0	0
	MIT			
$\text{HEAT}_{\alpha=0.20}$	3078.7(40.0)	0.908(0.001)	0.901(0.001)	0.205(0.003)
N&K	2806.7(43.5)	0.916(0.001)	0.919(0.001)	0.247(0.004)
t-statistic	-2.52E+01	-2.52E+01	-5.65E+01	-4.61E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0

TABLE B.9: Summary of link prediction for embedding dimension 25. We present AUROC, AP, and mean average precision (mAP) to 3 decimal places and rank to 1 decimal place. Rank is the average position that a true edge appears in the list of false edges ranked by distance. For mAP, we rank distances with respect to each node in the network and compute a separate precision score for each node, then report the mean. All scores are averaged over 30 random starting seeds. Standard deviation is given in brackets.

	Cora_ML				PPI			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	93.0(8.0)	0.925(0.007)	0.937(0.005)	0.258(0.010)	713.0(17.6)	0.913(0.002)	0.919(0.002)	0.188(0.004)
AANE	303.3(13.6)	0.753(0.011)	0.774(0.011)	0.145(0.006)	3705.9(42.2)	0.549(0.005)	0.592(0.005)	0.072(0.004)
TADW	59.5(5.1)	0.952(0.004)	0.949(0.005)	0.238(0.010)	2222.8(53.5)	0.730(0.007)	0.730(0.006)	0.091(0.004)
ATTRPURE	345.8(13.9)	0.718(0.011)	0.744(0.011)	0.135(0.006)	3854.9(42.0)	0.511(0.005)	0.511(0.004)	0.018(0.002)
DEEPWALK	181.9(11.2)	0.852(0.009)	0.896(0.006)	0.254(0.009)	890.1(19.2)	0.892(0.002)	0.901(0.002)	0.183(0.005)
SAGEGCN	131.5(10.0)	0.893(0.008)	0.907(0.007)	0.203(0.011)	3511.9(76.8)	0.573(0.009)	0.592(0.010)	0.073(0.004)
$HEAT_{\alpha=0.00}$	67.3(59.6)	0.946(0.049)	0.962(0.032)	0.504(0.212)	**262.2(8.3)**	**0.968(0.001)**	**0.968(0.001)**	**0.375(0.007)**
$HEAT_{\alpha=0.20}$	**20.4(17.7)**	**0.984(0.014)**	**0.986(0.013)**	**0.579(0.185)**	330.5(91.1)	0.960(0.011)	0.961(0.009)	0.357(0.032)
$HEAT_{\alpha=1.00}$	250.3(13.4)	0.796(0.011)	0.825(0.011)	0.140(0.007)	3540.3(38.8)	0.569(0.005)	0.577(0.005)	0.064(0.004)
	Citeseer				MIT			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	128.9(6.7)	0.840(0.008)	0.869(0.007)	0.213(0.012)	2709.5(34.5)	0.919(0.001)	0.922(0.001)	0.258(0.003)
AANE	280.4(10.7)	0.651(0.013)	0.647(0.015)	0.100(0.009)	11569.6(68.5)	0.653(0.002)	0.648(0.002)	0.101(0.003)
TADW	38.4(3.8)	0.953(0.005)	0.947(0.007)	0.209(0.013)	5489.1(67.1)	0.836(0.002)	0.840(0.002)	0.216(0.003)
ATTRPURE	282.6(10.3)	0.648(0.013)	0.645(0.014)	0.098(0.008)	12979.6(63.8)	0.611(0.002)	0.616(0.002)	0.099(0.003)
DEEPWALK	275.9(10.5)	0.657(0.013)	0.767(0.009)	0.230(0.011)	2056.4(31.2)	0.938(0.001)	0.939(0.001)	0.307(0.003)
SAGEGCN	111.8(12.0)	0.862(0.015)	0.888(0.013)	0.250(0.018)	6677.5(172.8)	0.800(0.005)	0.812(0.005)	0.192(0.004)
$HEAT_{\alpha=0.00}$	119.5(70.1)	0.852(0.088)	0.884(0.066)	0.394(0.269)	**1827.0(31.5)**	**0.945(0.001)**	**0.944(0.001)**	**0.321(0.003)**
$HEAT_{\alpha=0.20}$	**26.3(14.0)**	**0.968(0.017)**	**0.973(0.015)**	**0.513(0.228)**	2060.6(32.0)	0.938(0.001)	0.940(0.001)	0.316(0.003)
$HEAT_{\alpha=1.00}$	124.9(7.0)	0.845(0.009)	0.865(0.009)	0.179(0.010)	12026.2(94.7)	0.640(0.003)	0.666(0.003)	0.131(0.004)
	Pubmed				Mean Ranks			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	916.4(34.4)	0.862(0.005)	0.885(0.004)	0.180(0.005)	4.2	4.2	4	3.8
AANE	1376.7(27.8)	0.793(0.004)	0.775(0.004)	0.061(0.002)	7.4	7.4	7.8	7.4
TADW	751.2(41.2)	0.887(0.006)	0.870(0.007)	0.096(0.005)	3.2	3.2	3.4	5.2
ATTRPURE	1977.3(32.0)	0.703(0.005)	0.685(0.005)	0.063(0.002)	9	9	9	8.8
DEEPWALK	1725.2(32.9)	0.741(0.005)	0.828(0.004)	0.140(0.004)	5.4	5.4	5.4	3.4
SAGEGCN	799.9(56.1)	0.880(0.008)	0.860(0.007)	0.066(0.004)	4.6	4.6	4.6	5.4
$HEAT_{\alpha=0.00}$	1546.1(30.7)	0.768(0.005)	0.854(0.003)	0.205(0.004)	3.2	3.2	2.8	1.8
$HEAT_{\alpha=0.20}$	**309.4(11.9)**	**0.954(0.002)**	**0.953(0.002)**	**0.209(0.004)**	**1.6**	**1.6**	**1.4**	**1.4**
$HEAT_{\alpha=1.00}$	1070.1(38.2)	0.839(0.006)	0.835(0.006)	0.059(0.003)	6.4	6.4	6.6	7.8

TABLE B.10: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the link prediction task, for embedding dimension 25. For each network, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML			
	Mean Rank	AUROC	AP	mAP
$\text{HEAT}_{\alpha=0.20}$	**20.4(17.7)**	**0.984(0.014)**	**0.986(0.013)**	**0.579(0.185)**
TADW	59.5(5.1)	0.952(0.004)	0.949(0.005)	0.238(0.010)
t-statistic	1.16E+01	1.16E+01	1.39E+01	1.01E+01
p-value	1.26E-13	1.26E-13	1.15E-16	2.51E-11
$p < 0.05$	1	1	1	1
	Citeseer			
$\text{HEAT}_{\alpha=0.20}$	**26.3(14.0)**	**0.968(0.017)**	**0.973(0.015)**	**0.513(0.228)**
TADW	38.4(3.8)	0.953(0.005)	0.947(0.007)	0.209(0.013)
t-statistic	4.57E+00	4.57E+00	8.79E+00	7.29E+00
p-value	3.24E-05	3.20E-05	3.27E-11	2.40E-08
$p < 0.05$	1	1	1	1
	Pubmed			
$\text{HEAT}_{\alpha=0.20}$	**259.3(9.0)**	**0.961(0.001)**	**0.963(0.001)**	**0.313(0.005)**
TADW	407.9(13.5)	0.939(0.002)	0.938(0.003)	0.201(0.005)
t-statistic	5.02E+01	5.02E+01	4.80E+01	8.49E+01
p-value	4.30E-45	4.30E-45	2.80E-41	7.52E-63
$p < 0.05$	1	1	1	1
	PPI			
$\text{HEAT}_{\alpha=0.20}$	**330.5(91.1)**	**0.960(0.011)**	**0.961(0.009)**	**0.357(0.032)**
N&K	713.0(17.6)	0.913(0.002)	0.919(0.002)	0.188(0.004)
t-statistic	2.26E+01	2.26E+01	2.37E+01	2.87E+01
p-value	3.58E-21	3.58E-21	5.78E-22	1.21E-23
$p < 0.05$	1	1	1	1
	MIT			
$\text{HEAT}_{\alpha=0.20}$	2060.6(32.0)	0.938(0.001)	**0.940(0.001)**	**0.316(0.003)**
DEEPWALK	2056.4(31.2)	0.938(0.001)	0.939(0.001)	0.307(0.003)
t-statistic	-5.16E-01	-5.16E-01	3.15E+00	1.09E+01
p-value	6.96E-01	6.96E-01	1.30E-03	9.68E-16
$p < 0.05$	0	0	1	1

TABLE B.11: Summary of link prediction for embedding dimension 50. We present AUROC, AP, and mean average precision (mAP) to 3 decimal places and rank to 1 decimal place. Rank is the average position that a true edge appears in the list of false edges ranked by distance. For mAP, we rank distances with respect to each node in the network and compute a separate precision score for each node, then report the mean. All scores are averaged over 30 random starting seeds. Standard deviation is given in brackets.

	Cora_ML				PPI			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	93.3(7.0)	0.925(0.006)	0.937(0.005)	0.262(0.009)	715.0(17.4)	0.913(0.002)	0.919(0.002)	0.188(0.004)
AANE	305.8(12.2)	0.751(0.010)	0.770(0.009)	0.161(0.007)	3509.4(44.4)	0.573(0.005)	0.609(0.005)	0.074(0.004)
TADW	63.9(4.4)	0.949(0.004)	0.951(0.004)	0.280(0.008)	2313.1(35.3)	0.719(0.004)	0.718(0.005)	0.094(0.004)
ATTRPURE	356.7(12.6)	0.709(0.010)	0.735(0.010)	0.150(0.007)	3855.9(42.5)	0.507(0.005)	0.507(0.004)	0.017(0.002)
DEEPWALK	181.8(11.4)	0.852(0.009)	0.897(0.007)	0.262(0.009)	864.5(17.5)	0.895(0.002)	0.906(0.002)	0.195(0.005)
SAGEGCN	119.8(8.9)	0.903(0.007)	0.919(0.007)	0.235(0.009)	3466.7(85.9)	0.578(0.010)	0.599(0.011)	0.075(0.004)
$HEAT_{\alpha=0.00}$	20.2(3.3)	0.984(0.003)	0.987(0.002)	0.680(0.009)	**750.1(17.8)**	**0.909(0.002)**	**0.915(0.002)**	**0.191(0.004)**
$HEAT_{\alpha=0.20}$	**8.1(1.4)**	**0.994(0.001)**	**0.995(0.001)**	**0.732(0.009)**	866.7(23.4)	0.895(0.003)	0.904(0.003)	0.182(0.004)
$HEAT_{\alpha=1.00}$	241.3(13.2)	0.804(0.011)	0.831(0.011)	0.163(0.007)	3496.2(38.8)	0.575(0.005)	0.583(0.005)	0.064(0.004)
	Citeseer				MIT			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	122.6(7.6)	0.848(0.009)	0.875(0.007)	0.217(0.012)	2706.9(34.3)	0.919(0.001)	0.922(0.001)	0.258(0.003)
AANE	265.6(10.1)	0.670(0.013)	0.667(0.012)	0.117(0.009)	11055.9(98.2)	0.669(0.003)	0.666(0.003)	0.105(0.003)
TADW	34.0(3.2)	0.959(0.004)	0.956(0.006)	0.255(0.013)	4748.4(52.0)	0.858(0.002)	0.858(0.002)	0.227(0.003)
ATTRPURE	270.2(10.0)	0.664(0.013)	0.665(0.013)	0.115(0.009)	13173.5(57.3)	0.605(0.002)	0.614(0.002)	0.100(0.003)
DEEPWALK	286.6(9.5)	0.643(0.012)	0.764(0.009)	0.240(0.010)	1900.2(28.8)	0.943(0.001)	0.946(0.001)	0.333(0.003)
SAGEGCN	91.6(9.1)	0.887(0.011)	0.911(0.009)	0.307(0.020)	6480.1(128.7)	0.806(0.004)	0.822(0.004)	0.209(0.005)
$HEAT_{\alpha=0.00}$	166.7(11.5)	0.793(0.014)	0.840(0.010)	0.219(0.011)	**1701.8(31.7)**	**0.949(0.001)**	**0.947(0.001)**	**0.325(0.003)**
$HEAT_{\alpha=0.20}$	**34.1(3.6)**	**0.959(0.005)**	**0.965(0.004)**	**0.370(0.017)**	1891.3(29.1)	0.943(0.001)	0.945(0.001)	0.323(0.003)
$HEAT_{\alpha=1.00}$	128.3(6.8)	0.841(0.008)	0.862(0.009)	0.183(0.011)	11940.5(87.3)	0.642(0.003)	0.672(0.002)	0.138(0.003)
	Pubmed				Mean Ranks			
	Mean Rank	AUROC	AP	mAP	Mean Rank	AUROC	AP	mAP
N&K	739.5(18.8)	0.889(0.003)	0.906(0.003)	0.227(0.004)	3.4	3.4	3.4	4.6
AANE	1466.6(20.0)	0.780(0.003)	0.803(0.003)	0.132(0.003)	7.2	7.2	7.6	7.6
TADW	434.1(13.8)	0.935(0.002)	0.937(0.002)	0.235(0.004)	3.2	3.2	3.6	3.8
ATTRPURE	2082.5(27.8)	0.687(0.004)	0.709(0.005)	0.128(0.004)	8.8	8.8	9	8.8
DEEPWALK	1763.3(29.2)	0.735(0.004)	0.825(0.003)	0.257(0.004)	5.8	5.8	5	2.6
SAGEGCN	448.4(30.1)	0.933(0.005)	0.939(0.003)	0.233(0.008)	4.6	4.6	4.6	4.8
$HEAT_{\alpha=0.00}$	1468.9(29.9)	0.779(0.004)	0.858(0.003)	0.228(0.004)	3.6	3.6	3.4	3.2
$HEAT_{\alpha=0.20}$	**251.4(8.5)**	**0.962(0.001)**	**0.964(0.001)**	**0.313(0.004)**	**2**	**2**	**2**	**2**
$HEAT_{\alpha=1.00}$	898.9(22.5)	0.865(0.003)	0.866(0.003)	0.093(0.003)	6.4	6.4	6.4	7.6

TABLE B.12: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the link prediction task, for embedding dimension 50. For each network, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML			
	Mean Rank	AUROC	AP	mAP
$\text{HEAT}_{\alpha=0.20}$	**8.1(1.4)**	**0.994(0.001)**	**0.995(0.001)**	**0.732(0.009)**
TADW	63.9(4.4)	0.949(0.004)	0.951(0.004)	0.280(0.008)
t-statistic	6.61E+01	6.61E+01	5.89E+01	2.09E+02
p-value	9.31E-39	9.31E-39	1.62E-35	2.40E-85
$p < 0.05$	1	1	1	1
	Citeseer			
$\text{HEAT}_{\alpha=0.20}$	34.1(3.6)	0.959(0.005)	**0.965(0.004)**	**0.370(0.017)**
TADW	34.0(3.2)	0.959(0.004)	0.956(0.006)	0.255(0.013)
t-statistic	-1.04E-01	-9.01E-02	6.35E+00	2.92E+01
p-value	5.41E-01	5.36E-01	2.40E-08	6.52E-35
$p < 0.05$	0	0	1	1
	Pubmed			
$\text{HEAT}_{\alpha=0.20}$	**251.4(8.5)**	**0.962(0.001)**	**0.964(0.001)**	**0.313(0.004)**
SAGEGCN	448.4(30.1)	0.933(0.005)	0.939(0.003)	0.233(0.008)
t-statistic	3.45E+01	3.45E+01	3.62E+01	4.75E+01
p-value	4.19E-28	4.19E-28	6.96E-32	1.25E-38
$p < 0.05$	1	1	1	1
	PPI			
$\text{HEAT}_{\alpha=0.20}$	866.7(23.4)	0.895(0.003)	0.904(0.003)	0.182(0.004)
N&K	715.0(17.4)	0.913(0.002)	0.919(0.002)	0.188(0.004)
t-statistic	-2.84E+01	-2.84E+01	-2.50E+01	-5.10E+00
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0
	MIT			
$\text{HEAT}_{\alpha=0.20}$	1891.3(29.1)	0.943(0.001)	0.945(0.001)	0.323(0.003)
DEEPWALK	1900.2(28.8)	0.943(0.001)	0.946(0.001)	0.333(0.003)
t-statistic	1.19E+00	1.19E+00	-5.84E+00	-1.17E+01
p-value	1.19E-01	1.19E-01	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0

TABLE B.13: Summary of node classification results for embedding dimension 10. Bold indicates best performance. Mean Ranks is the average position of an algorithm in a ranked list of performance.

	Cora_ML				PPI			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.761(0.012)	0.827(0.010)	0.704(0.015)	0.953(0.003)	0.387(0.002)	**0.695(0.002)**	0.268(0.002)	0.704(0.000)
AANE	0.706(0.001)	0.814(0.001)	0.624(0.001)	0.944(0.000)	0.395(0.001)	0.688(0.002)	0.277(0.001)	0.705(0.000)
TADW	0.837(0.001)	0.869(0.001)	0.808(0.001)	0.983(0.000)	0.393(0.001)	0.688(0.001)	0.275(0.001)	0.705(0.000)
ATTRPURE	0.675(0.002)	0.782(0.001)	0.594(0.002)	0.937(0.000)	**0.398(0.001)**	0.688(0.001)	**0.281(0.001)**	**0.705(0.000)**
DEEPWALK	**0.855(0.003)**	**0.886(0.003)**	**0.827(0.003)**	0.977(0.001)	0.387(0.001)	0.694(0.001)	0.269(0.001)	0.704(0.000)
SAGEGCN	0.679(0.033)	0.758(0.027)	0.616(0.041)	0.933(0.008)	0.388(0.002)	0.694(0.002)	0.269(0.002)	0.704(0.000)
$HEAT_{\alpha=0.00}$	0.804(0.008)	0.858(0.006)	0.756(0.011)	0.965(0.002)	0.388(0.002)	0.694(0.002)	0.269(0.002)	0.704(0.000)
$HEAT_{\alpha=0.20}$	0.849(0.005)	0.884(0.005)	0.817(0.006)	**0.980(0.001)**	0.388(0.002)	0.694(0.003)	0.269(0.003)	0.704(0.000)
$HEAT_{\alpha=1.00}$	0.655(0.012)	0.757(0.009)	0.578(0.015)	0.927(0.004)	0.390(0.001)	0.690(0.002)	0.272(0.002)	0.702(0.000)
	Citeseer				MIT			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.516(0.029)	0.856(0.022)	0.370(0.029)	0.861(0.005)	0.508(0.007)	**0.858(0.007)**	0.361(0.007)	0.937(0.001)
AANE	0.609(0.001)	0.829(0.001)	0.482(0.001)	0.897(0.000)	0.036(0.003)	0.414(0.012)	0.019(0.002)	0.866(0.000)
TADW	0.878(0.003)	0.899(0.003)	0.857(0.003)	0.980(0.001)	0.544(0.001)	0.739(0.002)	0.430(0.001)	0.949(0.000)
ATTRPURE	0.597(0.002)	0.837(0.003)	0.465(0.003)	0.892(0.001)	0.056(0.004)	0.409(0.012)	0.030(0.002)	0.869(0.000)
DEEPWALK	**0.927(0.003)**	**0.939(0.003)**	**0.916(0.004)**	**0.987(0.001)**	**0.661(0.008)**	0.841(0.004)	**0.544(0.011)**	**0.965(0.001)**
SAGEGCN	0.466(0.069)	0.727(0.031)	0.347(0.069)	0.843(0.023)	0.480(0.018)	0.832(0.013)	0.337(0.017)	0.929(0.003)
$HEAT_{\alpha=0.00}$	0.594(0.019)	0.857(0.012)	0.455(0.023)	0.879(0.003)	0.634(0.007)	0.849(0.006)	0.507(0.010)	0.959(0.001)
$HEAT_{\alpha=0.20}$	0.887(0.003)	0.914(0.004)	0.861(0.004)	0.981(0.001)	0.637(0.007)	0.819(0.008)	0.522(0.011)	0.960(0.001)
$HEAT_{\alpha=1.00}$	0.743(0.003)	0.826(0.004)	0.675(0.004)	0.938(0.001)	0.029(0.007)	0.545(0.050)	0.015(0.004)	0.855(0.002)
	Pubmed				Mean Ranks			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.763(0.006)	0.786(0.006)	0.742(0.007)	0.909(0.003)	6.8	4	7.2	7
AANE	0.800(0.000)	0.817(0.000)	0.784(0.000)	0.931(0.000)	5.2	7	5	5
TADW	0.833(0.001)	0.846(0.001)	**0.822(0.001)**	**0.951(0.000)**	2.8	4.2	2.8	**2.4**
ATTRPURE	0.774(0.000)	0.791(0.000)	0.757(0.000)	0.916(0.000)	5.8	7.4	5.8	5.4
DEEPWALK	0.823(0.002)	0.834(0.001)	0.813(0.002)	0.937(0.001)	3	2.4	2.8	2.4
SAGEGCN	0.778(0.013)	0.797(0.011)	0.761(0.016)	0.914(0.007)	7	6.4	7	7.4
$HEAT_{\alpha=0.00}$	0.801(0.002)	0.822(0.002)	0.782(0.003)	0.929(0.001)	4.8	3.4	5	4.8
$HEAT_{\alpha=0.20}$	**0.833(0.002)**	**0.846(0.002)**	0.821(0.002)	0.949(0.000)	**2.6**	**2.4**	**2.6**	2.8
$HEAT_{\alpha=1.00}$	0.761(0.003)	0.780(0.002)	0.744(0.003)	0.912(0.002)	7	7.8	6.8	7.8

TABLE B.14: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the node classification task, for embedding dimension 10. For each network, we select the benchmark algorithm according to F1. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.849(0.005)	0.884(0.005)	0.817(0.006)	**0.980(0.001)**
DEEPWALK	0.855(0.003)	0.886(0.003)	0.827(0.003)	0.977(0.001)
t-statistic	-5.80E+00	-1.99E+00	-7.24E+00	1.84E+01
p-value	1.00E+00	9.74E-01	1.00E+00	2.23E-24
$p < 0.05$	0	0	0	1
	Citeseer			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.887(0.003)	0.914(0.004)	0.861(0.004)	0.981(0.001)
DEEPWALK	0.927(0.003)	0.939(0.003)	0.916(0.004)	0.987(0.001)
t-statistic	-4.98E+01	-2.89E+01	-5.24E+01	-3.01E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0
	Pubmed			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.833(0.002)	0.846(0.002)	0.821(0.002)	0.949(0.000)
TADW	0.833(0.001)	0.846(0.001)	0.822(0.001)	0.951(0.000)
t-statistic	-9.39E-01	3.46E-01	-1.98E+00	-1.85E+01
p-value	8.23E-01	3.66E-01	9.72E-01	1.00E+00
$p < 0.05$	0	0	0	0
	PPI			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.388(0.002)	**0.694(0.003)**	0.269(0.003)	0.704(0.000)
ATTRPURE	0.398(0.001)	0.688(0.001)	0.281(0.001)	0.705(0.000)
t-statistic	-2.23E+01	1.14E+01	-2.12E+01	-3.66E+01
p-value	1.00E+00	3.60E-14	1.00E+00	1.00E+00
$p < 0.05$	0	1	0	0
	MIT			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.637(0.007)	0.819(0.008)	0.522(0.011)	0.960(0.001)
DEEPWALK	0.661(0.008)	0.841(0.004)	0.544(0.011)	0.965(0.001)
t-statistic	-1.20E+01	-1.34E+01	-8.13E+00	-2.79E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0

TABLE B.15: Summary of node classification results for embedding dimension 25. Bold indicates best performance. Mean Ranks is the average position of an algorithm in a ranked list of performance.

	Cora_ML				PPI			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.776(0.009)	0.828(0.009)	0.729(0.011)	0.958(0.002)	0.385(0.001)	**0.698(0.001)**	0.266(0.001)	0.704(0.000)
AANE	0.742(0.001)	0.813(0.001)	0.683(0.001)	0.953(0.000)	0.398(0.002)	0.684(0.002)	0.281(0.002)	0.704(0.000)
TADW	**0.861(0.001)**	**0.890(0.001)**	**0.834(0.001)**	**0.986(0.000)**	0.394(0.001)	0.689(0.001)	0.276(0.001)	0.705(0.000)
ATTRPURE	0.739(0.002)	0.812(0.002)	0.678(0.002)	0.951(0.000)	**0.400(0.001)**	0.684(0.001)	**0.283(0.001)**	**0.705(0.000)**
DEEPWALK	0.859(0.003)	0.888(0.004)	0.833(0.003)	0.979(0.001)	0.387(0.001)	0.694(0.002)	0.268(0.002)	0.704(0.000)
SAGEGCN	0.760(0.017)	0.821(0.012)	0.708(0.022)	0.954(0.004)	0.389(0.002)	0.693(0.002)	0.270(0.002)	0.704(0.000)
$HEAT_{\alpha=0.00}$	0.809(0.006)	0.859(0.006)	0.764(0.008)	0.967(0.001)	0.389(0.002)	0.692(0.002)	0.271(0.002)	0.704(0.000)
$HEAT_{\alpha=0.20}$	0.852(0.003)	0.884(0.003)	0.822(0.004)	0.982(0.001)	0.389(0.002)	0.693(0.002)	0.270(0.002)	0.704(0.000)
$HEAT_{\alpha=1.00}$	0.697(0.005)	0.783(0.005)	0.628(0.006)	0.940(0.002)	0.394(0.002)	0.687(0.002)	0.276(0.002)	0.704(0.000)
	Citeseer				MIT			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.606(0.023)	0.867(0.011)	0.467(0.027)	0.880(0.006)	0.495(0.005)	0.862(0.007)	0.347(0.005)	0.937(0.001)
AANE	0.676(0.001)	0.857(0.001)	0.558(0.001)	0.927(0.000)	0.039(0.004)	0.448(0.018)	0.021(0.002)	0.872(0.000)
TADW	0.881(0.001)	0.912(0.001)	0.853(0.001)	0.982(0.000)	0.512(0.003)	0.715(0.002)	0.399(0.003)	0.947(0.000)
ATTRPURE	0.681(0.004)	0.861(0.005)	0.564(0.005)	0.928(0.001)	0.065(0.003)	0.430(0.007)	0.035(0.002)	0.875(0.000)
DEEPWALK	**0.917(0.005)**	**0.933(0.004)**	**0.901(0.006)**	**0.987(0.001)**	**0.769(0.005)**	0.867(0.004)	**0.692(0.006)**	**0.974(0.001)**
SAGEGCN	0.646(0.046)	0.800(0.018)	0.544(0.056)	0.904(0.016)	0.515(0.014)	0.823(0.010)	0.375(0.015)	0.935(0.002)
$HEAT_{\alpha=0.00}$	0.641(0.012)	0.860(0.012)	0.512(0.016)	0.890(0.005)	0.749(0.006)	**0.876(0.005)**	0.654(0.009)	0.970(0.001)
$HEAT_{\alpha=0.20}$	0.889(0.003)	0.914(0.003)	0.865(0.004)	0.984(0.001)	0.733(0.008)	0.844(0.006)	0.649(0.010)	0.970(0.001)
$HEAT_{\alpha=1.00}$	0.755(0.003)	0.834(0.003)	0.690(0.004)	0.944(0.001)	0.058(0.009)	0.509(0.024)	0.031(0.005)	0.866(0.001)
	Pubmed				Mean Ranks			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.770(0.005)	0.792(0.006)	0.749(0.006)	0.913(0.003)	7.4	4.2	7.6	7.4
AANE	0.816(0.000)	0.830(0.000)	0.802(0.000)	0.939(0.000)	5.6	7	5.6	6
TADW	**0.845(0.001)**	**0.856(0.000)**	**0.835(0.001)**	**0.956(0.000)**	**2.8**	3.4	**2.6**	**2.2**
ATTRPURE	0.797(0.001)	0.814(0.001)	0.780(0.001)	0.927(0.000)	5.6	7.4	5.4	5.4
DEEPWALK	0.831(0.001)	0.842(0.001)	0.821(0.001)	0.941(0.001)	3	**2**	3	2.2
SAGEGCN	0.798(0.011)	0.817(0.010)	0.781(0.013)	0.928(0.006)	6	5.8	6	6.2
$HEAT_{\alpha=0.00}$	0.799(0.002)	0.819(0.002)	0.780(0.003)	0.926(0.001)	4.8	4.2	5.2	6
$HEAT_{\alpha=0.20}$	0.835(0.002)	0.847(0.002)	0.822(0.002)	0.950(0.001)	3.2	3	3.2	2.8
$HEAT_{\alpha=1.00}$	0.769(0.002)	0.787(0.002)	0.752(0.003)	0.918(0.001)	6.6	8	6.4	6.8

TABLE B.16: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the node classification task, for embedding dimension 25. For each network, we select the benchmark algorithm according to F1. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.852(0.003)	0.884(0.003)	0.822(0.004)	0.982(0.001)
TADW	0.861(0.001)	0.890(0.001)	0.834(0.001)	0.986(0.000)
t-statistic	-1.50E+01	-8.25E+00	-1.62E+01	-3.67E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0
	Citeseer			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.889(0.003)	0.914(0.003)	0.865(0.004)	0.984(0.001)
DEEPWALK	0.917(0.005)	0.933(0.004)	0.901(0.006)	0.987(0.001)
t-statistic	-2.70E+01	-2.20E+01	-2.72E+01	-1.62E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0
	Pubmed			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.835(0.002)	0.847(0.002)	0.822(0.002)	0.950(0.001)
TADW	0.845(0.001)	0.856(0.000)	0.835(0.001)	0.956(0.000)
t-statistic	-2.62E+01	-2.23E+01	-2.78E+01	-3.94E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0
	PPI			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.389(0.002)	**0.693(0.002)**	0.270(0.002)	0.704(0.000)
ATTRPURE	0.400(0.001)	0.684(0.001)	0.283(0.001)	0.705(0.000)
t-statistic	-2.86E+01	2.05E+01	-2.84E+01	-3.45E+01
p-value	1.00E+00	2.54E-28	1.00E+00	1.00E+00
$p < 0.05$	0	1	0	0
	MIT			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.733(0.008)	0.844(0.006)	0.649(0.010)	0.970(0.001)
DEEPWALK	0.769(0.005)	0.867(0.004)	0.692(0.006)	0.974(0.001)
t-statistic	-2.17E+01	-1.82E+01	-2.07E+01	-2.42E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0

TABLE B.17: Summary of node classification results for embedding dimension 50. $HEAT_{\alpha=0.2}$ considers node attributes, whereas $HEAT_{\alpha=0.0}$ does not. Bold indicates best performance. Standard deviation is given in brackets. Mean Ranks is the average position of an algorithm in a ranked list of performance.

	Cora_ML				PPI			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.775(0.007)	0.824(0.008)	0.732(0.008)	0.958(0.001)	0.384(0.001)	**0.699(0.001)**	0.264(0.001)	0.704(0.000)
AANE	0.753(0.002)	0.830(0.002)	0.690(0.003)	0.952(0.000)	0.403(0.002)	0.681(0.002)	0.287(0.002)	0.702(0.000)
TADW	**0.863(0.001)**	**0.889(0.001)**	**0.838(0.001)**	**0.983(0.000)**	0.397(0.002)	0.686(0.002)	0.279(0.002)	0.705(0.000)
ATTRPURE	0.744(0.002)	0.825(0.002)	0.678(0.003)	0.950(0.000)	**0.403(0.001)**	0.682(0.002)	**0.286(0.002)**	**0.706(0.000)**
DEEPWALK	0.857(0.003)	0.883(0.003)	0.833(0.003)	0.979(0.000)	0.388(0.001)	0.693(0.001)	0.269(0.001)	0.704(0.000)
SAGEGCN	0.785(0.013)	0.839(0.010)	0.737(0.017)	0.959(0.003)	0.389(0.002)	0.693(0.003)	0.270(0.003)	0.704(0.000)
$HEAT_{\alpha=0.00}$	0.806(0.005)	0.853(0.004)	0.764(0.007)	0.968(0.001)	0.387(0.002)	0.695(0.002)	0.268(0.002)	0.704(0.000)
$HEAT_{\alpha=0.20}$	0.852(0.003)	0.881(0.003)	0.825(0.004)	0.982(0.000)	0.387(0.002)	0.694(0.002)	0.269(0.002)	0.704(0.000)
$HEAT_{\alpha=1.00}$	0.701(0.006)	0.785(0.006)	0.634(0.008)	0.941(0.001)	0.395(0.002)	0.686(0.002)	0.277(0.002)	0.703(0.001)
	Citeseer				MIT			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.646(0.015)	0.885(0.011)	0.509(0.019)	0.898(0.005)	0.492(0.003)	0.862(0.006)	0.344(0.003)	0.937(0.001)
AANE	0.716(0.001)	0.879(0.002)	0.605(0.002)	0.937(0.000)	0.044(0.007)	0.449(0.026)	0.024(0.004)	0.875(0.000)
TADW	**0.889(0.001)**	**0.920(0.001)**	0.860(0.001)	0.984(0.000)	0.515(0.003)	0.719(0.004)	0.403(0.004)	0.946(0.000)
ATTRPURE	0.718(0.002)	0.875(0.002)	0.609(0.003)	0.938(0.001)	0.077(0.002)	0.462(0.007)	0.042(0.001)	0.878(0.000)
DEEPWALK	0.883(0.005)	0.909(0.004)	0.858(0.006)	0.980(0.001)	**0.816(0.003)**	0.893(0.002)	**0.751(0.004)**	**0.978(0.000)**
SAGEGCN	0.715(0.031)	0.831(0.014)	0.628(0.041)	0.931(0.011)	0.526(0.013)	0.818(0.011)	0.388(0.013)	0.937(0.002)
$HEAT_{\alpha=0.00}$	0.659(0.012)	0.874(0.009)	0.529(0.015)	0.900(0.004)	0.795(0.003)	**0.895(0.003)**	0.715(0.004)	0.975(0.000)
$HEAT_{\alpha=0.20}$	0.887(0.003)	0.913(0.002)	**0.863(0.003)**	**0.984(0.000)**	0.787(0.004)	0.869(0.004)	0.719(0.005)	0.976(0.000)
$HEAT_{\alpha=1.00}$	0.758(0.002)	0.834(0.002)	0.695(0.003)	0.946(0.001)	0.078(0.006)	0.493(0.017)	0.042(0.003)	0.869(0.001)
	Pubmed				Mean Ranks			
	F1	Precision	Recall	AUROC	F1	Precision	Recall	AUROC
N&K	0.774(0.003)	0.795(0.003)	0.755(0.004)	0.916(0.002)	7.6	5	7.8	7
AANE	0.825(0.000)	0.837(0.000)	0.813(0.000)	0.944(0.000)	5.4	6.6	5.6	6.6
TADW	**0.858(0.000)**	**0.868(0.000)**	**0.849(0.000)**	**0.960(0.000)**	**2.2**	3.2	**2.2**	**1.8**
ATTRPURE	0.801(0.002)	0.816(0.001)	0.786(0.002)	0.929(0.001)	5.8	7.2	5.8	5.4
DEEPWALK	0.833(0.001)	0.844(0.001)	0.823(0.001)	0.942(0.001)	3	3	3	2.8
SAGEGCN	0.811(0.004)	0.827(0.004)	0.796(0.004)	0.935(0.001)	5.2	5.6	5	5.6
$HEAT_{\alpha=0.00}$	0.800(0.002)	0.819(0.002)	0.781(0.002)	0.926(0.001)	5.8	4	6	5.2
$HEAT_{\alpha=0.20}$	0.838(0.002)	0.850(0.002)	0.826(0.002)	0.952(0.001)	3.4	**2.6**	3	3
$HEAT_{\alpha=1.00}$	0.774(0.003)	0.791(0.002)	0.757(0.003)	0.921(0.001)	6.6	7.8	6.6	7.6

TABLE B.18: T-test statistics achieved by $\text{HEAT}_{\alpha=0.2}$ on the node classification task, for embedding dimension 50. For each network, we select the benchmark algorithm according to F1. Significant results at a significance level of 0.05 are highlighted in bold.

	Cora_ML			
	F1	Precision	Recall	AUROC
$\text{HEAT}_{\alpha=0.20}$	0.852(0.003)	0.881(0.003)	0.825(0.004)	0.982(0.000)
TADW	0.863(0.001)	0.889(0.001)	0.838(0.001)	0.983(0.000)
t-statistic	-1.80E+01	-1.60E+01	-1.53E+01	-1.44E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0
	Citeseer			
$\text{HEAT}_{\alpha=0.20}$	0.887(0.003)	0.913(0.002)	**0.863(0.003)**	0.984(0.000)
TADW	0.889(0.001)	0.920(0.001)	0.860(0.001)	0.984(0.000)
t-statistic	-3.17E+00	-1.33E+01	4.46E+00	-1.01E+00
p-value	9.99E-01	1.00E+00	3.95E-05	8.40E-01
$p < 0.05$	0	0	1	0
	Pubmed			
$\text{HEAT}_{\alpha=0.20}$	0.838(0.002)	0.850(0.002)	0.826(0.002)	0.952(0.001)
TADW	0.858(0.000)	0.868(0.000)	0.849(0.000)	0.960(0.000)
t-statistic	-5.94E+01	-5.63E+01	-5.89E+01	-7.04E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0
	PPI			
$\text{HEAT}_{\alpha=0.20}$	0.387(0.002)	**0.694(0.002)**	0.269(0.002)	**0.704(0.000)**
AANE	0.403(0.002)	0.681(0.002)	0.287(0.002)	0.702(0.000)
t-statistic	-3.74E+01	2.53E+01	-3.62E+01	2.68E+01
p-value	1.00E+00	8.48E-33	1.00E+00	2.16E-33
$p < 0.05$	0	1	0	1
	MIT			
$\text{HEAT}_{\alpha=0.20}$	0.787(0.004)	0.869(0.004)	0.719(0.005)	0.976(0.000)
DEEPWALK	0.816(0.003)	0.893(0.002)	0.751(0.004)	0.978(0.000)
t-statistic	-3.30E+01	-2.95E+01	-2.59E+01	-1.90E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0

Appendix C

Supplementary for Chapter 4

C.1 Network Reconstruction

Here we present additional results for the network reconstruction task for HEADNet. We present results for embedding dimension 5+5 (table C.1 and table C.2), 10+10 (table C.3 and table C.4) and 50+50 (table C.5 and table C.6).

C.2 Link Prediction

Here we present additional results for the link prediction task for HEADNet. We present results for embedding dimension 5+5 (table C.7), 10+10 (table C.8) and 50+50 (table C.9).

C.3 Link Prediction: Unseen Nodes

Table C.10 shows the results for the unseen nodes experiment on all networks for embedding dimensions 25+25 and 50+50. Like the link prediction case, we see stronger performance on all networks over all dimensions than G2G, including the 5+5 case. Following our findings in table 4.7, we are lead to believe that supposing nodes exist on a hidden hyperbolic space can better predict the links on nodes that are not seen during the training process.

TABLE C.1: Summary of the network reconstruction task on synthetic, Cora_ML and Citeseer networks for an embedding dimension of 5+5. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. $\text{HEADNet}_{\Sigma=\mathbb{I}}$ denoted HEADNet with an identity variance. A dash (–) indicates that a network did not have attributes. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Synthetic								
ATP (log)	632.4(54.6)	0.634(0.025)	0.645(0.024)	0.016(0.011)	0.006(0.018)	0.011(0.015)	0.014(0.014)	0.023(0.016)
ATP (harmonic)	610.5(52.6)	0.647(0.024)	0.652(0.023)	0.016(0.011)	0.006(0.018)	0.009(0.013)	0.013(0.013)	0.023(0.015)
LINE	1209.7(49.7)	0.300(0.017)	0.379(0.005)	0.007(0.001)	0.008(0.002)	0.026(0.008)	0.053(0.016)	0.132(0.041)
$\text{G2G}_{\text{NA},K=1}$	89.5(10.9)	0.949(0.007)	0.892(0.014)	0.170(0.019)	0.089(0.015)	0.061(0.018)	0.042(0.019)	0.044(0.013)
$\text{G2G}_{\text{NA},K=3}$	89.0(10.5)	0.949(0.007)	0.901(0.017)	0.126(0.014)	0.073(0.012)	0.081(0.014)	0.118(0.019)	0.276(0.067)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	18.4(3.5)	0.990(0.002)	0.985(0.004)	0.325(0.032)	0.182(0.021)	0.160(0.019)	0.151(0.021)	0.112(0.012)
$\text{G2G}_{K=1}$	–	–	–	–	–	–	–	–
$\text{G2G}_{K=3}$	–	–	–	–	–	–	–	–
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	–	–	–	–	–	–	–	–
Significance Test								
NK	77.7(9.5)	0.955(0.006)	0.937(0.009)	0.225(0.013)	0.224(0.012)	0.135(0.007)	0.102(0.006)	0.068(0.004)
$\text{HEADNet}_{\text{NA}}$	**6.3(1.6)**	**0.997(0.001)**	**0.995(0.002)**	**0.595(0.051)**	**0.458(0.054)**	**0.531(0.049)**	**0.536(0.049)**	**0.619(0.063)**
t-statistic	4.04E+01	3.50E+01	3.27E+01	3.83E+01	2.29E+01	4.35E+01	4.83E+01	4.75E+01
p-value	1.79E-28	2.58E-26	2.53E-26	5.95E-29	9.56E-22	4.09E-29	3.50E-30	1.42E-29
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	–	–	–	–	–	–	–	–
t-statistic	–	–	–	–	–	–	–	–
p-value	–	–	–	–	–	–	–	–
$p < 0.05$	–	–	–	–	–	–	–	–
Cora_ML								
ATP (log)	4178.6(26.8)	0.504(0.003)	0.520(0.003)	0.021(0.000)	0.009(0.001)	0.014(0.001)	0.019(0.001)	0.038(0.003)
ATP (harmonic)	4136.5(26.5)	0.509(0.003)	0.528(0.003)	0.026(0.001)	0.017(0.001)	0.020(0.002)	0.025(0.002)	0.042(0.003)
LINE	3842.6(46.4)	0.544(0.006)	0.491(0.004)	0.086(0.002)	0.109(0.004)	0.154(0.005)	0.185(0.008)	0.232(0.010)
$\text{G2G}_{\text{NA},K=1}$	67.8(8.6)	0.992(0.001)	0.988(0.002)	0.537(0.024)	0.464(0.028)	0.440(0.029)	0.434(0.028)	0.425(0.030)
NK	133.4(5.6)	0.984(0.001)	0.983(0.001)	0.540(0.004)	0.568(0.007)	0.400(0.003)	0.315(0.002)	0.209(0.001)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	27.9(4.2)	0.997(0.001)	0.995(0.001)	0.673(0.009)	0.614(0.011)	0.460(0.006)	0.365(0.005)	0.238(0.003)
$\text{G2G}_{K=1}$	75.8(8.6)	0.991(0.001)	0.987(0.002)	0.523(0.020)	0.456(0.023)	0.423(0.022)	0.417(0.021)	0.409(0.019)
$\text{G2G}_{K=3}$	65.6(7.2)	0.992(0.001)	0.989(0.002)	0.552(0.019)	0.491(0.020)	0.472(0.020)	0.469(0.018)	0.451(0.016)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	31.0(7.6)	0.996(0.001)	0.995(0.001)	0.654(0.030)	0.594(0.034)	0.448(0.022)	0.359(0.015)	0.235(0.007)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	57.3(8.6)	0.993(0.001)	0.990(0.002)	0.583(0.024)	0.519(0.024)	0.504(0.028)	0.495(0.028)	0.465(0.026)
$\text{HEADNet}_{\text{NA}}$	**8.5(2.8)**	**0.999(0.000)**	**0.999(0.001)**	**0.862(0.018)**	**0.840(0.023)**	**0.807(0.023)**	**0.795(0.023)**	**0.795(0.023)**
t-statistic	2.97E+01	2.97E+01	2.46E+01	5.02E+01	5.28E+01	4.61E+01	4.49E+01	5.15E+01
p-value	7.84E-27	7.84E-27	1.70E-26	2.10E-47	5.06E-51	1.38E-46	1.09E-45	6.70E-50
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**9.5(2.7)**	**0.999(0.000)**	**0.999(0.001)**	**0.833(0.018)**	**0.796(0.023)**	**0.778(0.021)**	**0.768(0.018)**	**0.765(0.019)**
t-statistic	2.92E+01	2.92E+01	2.52E+01	4.52E+01	4.61E+01	4.35E+01	4.42E+01	5.05E+01
p-value	2.62E-26	2.62E-26	2.84E-25	1.13E-44	1.30E-47	1.02E-43	1.13E-41	6.71E-47
$p < 0.05$	1	1	1	1	1	1	1	1
Citeseer								
ATP (log)	2415.1(63.1)	0.549(0.012)	0.547(0.016)	0.014(0.001)	0.004(0.001)	0.011(0.001)	0.017(0.002)	0.004(0.019)
ATP (harmonic)	2418.8(62.2)	0.549(0.012)	0.545(0.013)	0.015(0.001)	0.005(0.001)	0.012(0.001)	0.016(0.003)	0.003(0.013)
LINE	3489.6(31.4)	0.349(0.006)	0.392(0.002)	0.031(0.001)	0.036(0.002)	0.090(0.005)	0.082(0.006)	0.158(0.061)
$\text{G2G}_{\text{NA},K=1}$	8.1(2.1)	0.999(0.000)	0.998(0.001)	0.748(0.013)	0.641(0.016)	0.498(0.028)	0.469(0.035)	0.513(0.063)
NK	50.4(4.7)	0.991(0.001)	0.992(0.001)	0.709(0.004)	0.633(0.006)	0.359(0.002)	0.248(0.001)	0.139(0.001)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	5.1(1.2)	0.999(0.000)	0.999(0.000)	0.832(0.003)	0.756(0.005)	0.419(0.001)	0.285(0.001)	0.155(0.000)
$\text{G2G}_{K=1}$	19.1(3.5)	0.997(0.001)	0.993(0.002)	0.611(0.018)	0.500(0.022)	0.365(0.025)	0.329(0.030)	0.409(0.085)
$\text{G2G}_{K=3}$	19.5(3.6)	0.997(0.001)	0.992(0.002)	0.593(0.020)	0.475(0.023)	0.376(0.025)	0.349(0.025)	0.443(0.076)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	24.0(3.2)	0.996(0.001)	0.992(0.002)	0.548(0.002)	0.419(0.005)	0.296(0.002)	0.220(0.001)	0.133(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	7.3(1.8)	0.999(0.000)	0.998(0.001)	0.744(0.010)	0.623(0.014)	0.549(0.030)	0.514(0.040)	0.546(0.074)
$\text{HEADNet}_{\text{NA}}$	**3.3(1.1)**	**1.000(0.000)**	**0.999(0.000)**	**0.881(0.021)**	**0.821(0.030)**	**0.776(0.039)**	**0.775(0.036)**	**0.819(0.069)**
t-statistic	1.02E+01	1.02E+01	7.93E+00	3.23E+01	3.29E+01	2.54E+01	2.68E+01	1.48E+01
p-value	7.23E-14	7.23E-14	1.39E-10	1.25E-31	4.15E-31	3.03E-32	2.15E-34	1.25E-21
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	11.0(2.6)	0.998(0.000)	0.995(0.002)	0.675(0.021)	0.570(0.028)	**0.576(0.023)**	**0.556(0.019)**	**0.589(0.095)**
t-statistic	-6.34E+00	-6.36E+00	-8.53E+00	-1.59E+01	-9.24E+00	3.85E+00	5.22E+00	1.97E+00
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.55E-04	2.65E-06	2.68E-02
$p < 0.05$	0	0	0	0	0	1	1	1

TABLE C.2: Summary of the network reconstruction task on the Pubmed, Cora and Wiki Vote networks for an embedding dimension of 5+5. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. $\text{HEADNet}_{\Sigma=\mathbb{I}}$ denoted HEADNet with an identity variance. A dash (–) indicates that a network did not have attributes. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Pubmed								
ATP (log)	64106.7(205.6)	0.277(0.002)	0.380(0.001)	0.031(0.001)	0.016(0.002)	0.029(0.003)	0.034(0.005)	0.047(0.006)
ATP (harmonic)	63914.7(204.2)	0.279(0.002)	0.386(0.001)	0.035(0.001)	0.028(0.002)	0.037(0.004)	0.036(0.005)	0.048(0.006)
LINE	58883.2(222.3)	0.336(0.003)	0.388(0.001)	0.040(0.001)	0.071(0.002)	0.159(0.003)	0.201(0.004)	0.241(0.005)
$\text{G2G}_{\text{NA},K=1}$	279.2(39.1)	0.997(0.000)	0.995(0.001)	0.800(0.020)	0.745(0.024)	0.760(0.028)	0.795(0.022)	0.811(0.019)
NK	591.0(23.3)	0.993(0.000)	0.992(0.000)	0.704(0.002)	0.708(0.003)	0.491(0.001)	0.389(0.001)	0.267(0.001)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	104.9(6.6)	0.999(0.000)	0.998(0.000)	0.853(0.001)	0.825(0.002)	0.531(0.001)	0.416(0.001)	0.289(0.000)
$\text{G2G}_{K=1}$	2045.6(154.6)	0.977(0.002)	0.966(0.002)	0.242(0.014)	0.153(0.008)	0.255(0.010)	0.309(0.013)	0.368(0.014)
$\text{G2G}_{K=3}$	1688.5(145.1)	0.981(0.002)	0.971(0.002)	0.247(0.010)	0.172(0.006)	0.287(0.007)	0.346(0.009)	0.408(0.009)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	645.6(14.3)	0.993(0.000)	0.990(0.000)	0.487(0.002)	0.388(0.003)	0.335(0.001)	0.295(0.001)	0.230(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	90.8(6.3)	0.999(0.000)	0.998(0.000)	0.858(0.002)	0.803(0.003)	0.816(0.003)	0.840(0.003)	0.858(0.003)
$\text{HEADNet}_{\text{NA}}$	**60.7(5.4)**	**0.999(0.000)**	**0.999(0.000)**	**0.889(0.002)**	**0.851(0.003)**	**0.873(0.003)**	**0.896(0.003)**	**0.910(0.002)**
t-statistic	1.98E+01	1.98E+01	1.47E+01	6.97E+01	6.48E+01	6.71E+01	7.83E+01	7.63E+01
p-value	1.79E-27	1.79E-27	3.40E-21	1.04E-57	2.18E-55	7.28E-57	1.08E-60	9.84E-59
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	615.0(41.8)	0.993(0.000)	0.990(0.001)	0.449(0.014)	0.350(0.012)	0.532(0.009)	0.596(0.009)	0.651(0.008)
t-statistic	-6.80E+01	-6.80E+01	-6.27E+01	-1.64E+02	-2.05E+02	-1.62E+02	-1.49E+02	-1.29E+02
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0	0	0	0	0
Cora								
ATP (log)	33205.9(76.0)	0.492(0.001)	0.506(0.001)	0.028(0.001)	0.015(0.001)	0.019(0.001)	0.022(0.001)	0.034(0.003)
ATP (harmonic)	32914.7(74.4)	0.496(0.001)	0.512(0.001)	0.036(0.000)	0.031(0.001)	0.028(0.001)	0.029(0.001)	0.039(0.003)
LINE	29550.4(182.8)	0.548(0.003)	0.502(0.002)	0.120(0.002)	0.170(0.002)	0.211(0.003)	0.241(0.003)	0.286(0.005)
$\text{G2G}_{\text{NA},K=1}$	162.5(20.6)	0.998(0.000)	0.997(0.000)	0.780(0.017)	0.761(0.019)	0.704(0.022)	0.688(0.021)	0.686(0.018)
NK	635.8(28.4)	0.990(0.000)	0.990(0.000)	0.695(0.002)	0.748(0.002)	0.543(0.002)	0.427(0.001)	0.275(0.001)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	94.5(6.6)	0.999(0.000)	0.998(0.000)	0.802(0.001)	0.785(0.002)	0.585(0.001)	0.466(0.001)	0.304(0.000)
$\text{G2G}_{K=1}$	237.5(33.6)	0.996(0.001)	0.995(0.001)	0.708(0.029)	0.680(0.034)	0.618(0.032)	0.612(0.029)	0.625(0.025)
$\text{G2G}_{K=3}$	155.9(18.5)	0.998(0.000)	0.997(0.000)	0.772(0.018)	0.752(0.021)	0.708(0.021)	0.694(0.018)	0.696(0.014)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	108.2(5.8)	0.998(0.000)	0.998(0.000)	0.785(0.001)	0.763(0.002)	0.577(0.001)	0.463(0.001)	0.303(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	111.9(12.0)	0.998(0.000)	0.998(0.000)	0.824(0.013)	0.809(0.014)	0.766(0.018)	0.745(0.017)	0.733(0.015)
$\text{HEADNet}_{\text{NA}}$	**28.2(3.0)**	**1.000(0.000)**	**0.999(0.000)**	**0.923(0.004)**	**0.919(0.004)**	**0.889(0.005)**	**0.881(0.005)**	**0.881(0.005)**
t-statistic	3.70E+01	3.70E+01	3.27E+01	3.88E+01	4.02E+01	3.58E+01	4.12E+01	5.05E+01
p-value	1.85E-28	1.85E-28	9.45E-28	3.67E-30	1.29E-30	1.35E-28	7.78E-31	5.60E-34
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**39.0(4.4)**	**0.999(0.000)**	**0.999(0.000)**	**0.890(0.007)**	**0.874(0.008)**	**0.864(0.007)**	**0.861(0.007)**	**0.863(0.007)**
t-statistic	3.11E+01	3.11E+01	2.74E+01	2.40E+01	2.14E+01	2.75E+01	3.35E+01	4.17E+01
p-value	2.52E-28	2.52E-28	3.19E-27	6.06E-27	5.18E-26	4.19E-27	7.74E-31	5.89E-36
$p < 0.05$	1	1	1	1	1	1	1	1
Wiki Vote								
ATP (log)	53362.7(266.8)	0.485(0.003)	0.505(0.002)	0.030(0.000)	0.013(0.001)	0.022(0.001)	0.027(0.001)	0.041(0.001)
ATP (harmonic)	53393.9(285.4)	0.485(0.003)	0.505(0.002)	0.030(0.000)	0.013(0.001)	0.022(0.001)	0.027(0.001)	0.041(0.001)
LINE	70632.8(593.1)	0.319(0.006)	0.380(0.002)	0.043(0.001)	0.076(0.002)	0.147(0.003)	0.183(0.003)	0.228(0.003)
$\text{G2G}_{\text{NA},K=1}$	4888.7(498.0)	0.953(0.005)	0.937(0.006)	0.407(0.046)	0.315(0.050)	0.289(0.028)	0.310(0.024)	0.352(0.020)
$\text{G2G}_{\text{NA},K=3}$	4554.3(94.8)	0.956(0.001)	0.937(0.002)	0.222(0.006)	0.159(0.005)	0.203(0.004)	0.238(0.003)	0.291(0.003)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	2764.4(42.8)	0.973(0.000)	0.963(0.001)	0.514(0.002)	0.468(0.005)	0.356(0.002)	0.302(0.002)	0.241(0.001)
$\text{G2G}_{K=1}$	–	–	–	–	–	–	–	–
$\text{G2G}_{K=3}$	–	–	–	–	–	–	–	–
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	–	–	–	–	–	–	–	–
Significance Test								
NK	4603.0(43.5)	0.956(0.000)	0.960(0.000)	0.172(0.001)	0.224(0.003)	0.199(0.002)	0.188(0.001)	0.174(0.001)
$\text{HEADNet}_{\text{NA}}$	**925.1(42.9)**	**0.991(0.000)**	**0.987(0.001)**	**0.703(0.022)**	**0.691(0.038)**	**0.571(0.017)**	**0.593(0.013)**	**0.636(0.009)**
t-statistic	3.30E+02	3.30E+02	1.65E+02	1.34E+02	6.68E+01	1.22E+02	1.68E+02	2.81E+02
p-value	6.74E-97	6.74E-97	2.76E-69	1.69E-42	6.64E-34	5.99E-42	8.20E-46	1.53E-52
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	–	–	–	–	–	–	–	–
t-statistic	–	–	–	–	–	–	–	–
p-value	–	–	–	–	–	–	–	–
$p < 0.05$	–	–	–	–	–	–	–	–

TABLE C.3: Summary of the network reconstruction task on synthetic, Cora_ML and Citeseer networks for an embedding dimension of 10+10. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. A dash (–) indicates that a network did not have attributes. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Synthetic								
ATP (log)	614.0(52.0)	0.645(0.024)	0.660(0.025)	0.016(0.011)	0.006(0.018)	0.010(0.014)	0.014(0.013)	0.025(0.018)
ATP (harmonic)	598.5(50.1)	0.654(0.025)	0.664(0.025)	0.017(0.011)	0.006(0.018)	0.010(0.013)	0.015(0.014)	0.025(0.018)
LINE	1227.1(48.9)	0.290(0.017)	0.375(0.005)	0.013(0.003)	0.017(0.004)	0.066(0.012)	0.119(0.021)	0.205(0.034)
$\text{G2G}_{\text{NA},K=1}$	85.3(9.7)	0.951(0.007)	0.895(0.014)	0.197(0.018)	0.118(0.015)	0.091(0.019)	0.054(0.018)	0.046(0.019)
$\text{G2G}_{\text{NA},K=3}$	73.9(10.6)	0.958(0.006)	0.910(0.019)	0.137(0.017)	0.085(0.015)	0.103(0.022)	0.136(0.032)	0.292(0.071)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	10.3(2.5)	0.995(0.001)	0.992(0.003)	0.545(0.091)	0.389(0.103)	0.297(0.045)	0.216(0.022)	0.132(0.008)
$\text{G2G}_{K=1}$	–	–	–	–	–	–	–	–
$\text{G2G}_{K=3}$	–	–	–	–	–	–	–	–
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	–	–	–	–	–	–	–	–
Significance Test								
NK	56.9(9.6)	0.968(0.006)	0.949(0.009)	0.262(0.018)	0.247(0.016)	0.156(0.010)	0.120(0.007)	0.080(0.005)
$\text{HEADNet}_{\text{NA}}$	**4.1(1.6)**	**0.998(0.001)**	**0.997(0.002)**	**0.741(0.045)**	**0.626(0.061)**	**0.675(0.051)**	**0.676(0.049)**	**0.730(0.050)**
t-statistic	2.97E+01	2.60E+01	2.79E+01	5.44E+01	3.30E+01	5.50E+01	6.17E+01	7.02E+01
p-value	1.95E-24	1.46E-22	7.43E-24	9.92E-38	3.00E-27	5.90E-33	9.06E-34	8.19E-35
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	–	–	–	–	–	–	–	–
t-statistic	–	–	–	–	–	–	–	–
p-value	–	–	–	–	–	–	–	–
$p < 0.05$	–	–	–	–	–	–	–	–
Cora_ML								
ATP (log)	4157.0(26.7)	0.506(0.003)	0.521(0.003)	0.025(0.001)	0.012(0.001)	0.022(0.001)	0.031(0.002)	0.058(0.004)
ATP (harmonic)	4115.2(26.6)	0.511(0.003)	0.528(0.003)	0.029(0.001)	0.020(0.002)	0.026(0.001)	0.034(0.002)	0.058(0.004)
LINE	3864.7(35.2)	0.541(0.004)	0.487(0.003)	0.078(0.002)	0.115(0.004)	0.153(0.003)	0.175(0.006)	0.190(0.011)
$\text{G2G}_{\text{NA},K=1}$	59.3(6.0)	0.993(0.001)	0.990(0.001)	0.578(0.010)	0.510(0.013)	0.488(0.014)	0.479(0.012)	0.461(0.015)
NK	45.9(5.0)	0.995(0.001)	0.992(0.001)	0.637(0.007)	0.577(0.010)	0.574(0.011)	0.561(0.012)	0.530(0.015)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	110.6(4.8)	0.987(0.001)	0.986(0.001)	0.569(0.003)	0.593(0.007)	0.421(0.003)	0.331(0.002)	0.217(0.001)
$\text{G2G}_{K=1}$	14.7(4.3)	0.998(0.001)	0.998(0.001)	0.772(0.020)	0.721(0.021)	0.536(0.016)	0.419(0.012)	0.264(0.005)
$\text{G2G}_{K=3}$	66.2(8.3)	0.992(0.001)	0.989(0.002)	0.566(0.017)	0.507(0.020)	0.468(0.021)	0.459(0.021)	0.446(0.020)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	57.5(6.1)	0.993(0.001)	0.990(0.001)	0.594(0.016)	0.540(0.017)	0.519(0.021)	0.510(0.021)	0.488(0.021)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	**45.9(5.0)**	**0.995(0.001)**	**0.992(0.001)**	**0.637(0.007)**	**0.577(0.010)**	**0.574(0.011)**	**0.561(0.012)**	**0.530(0.015)**
$\text{HEADNet}_{\text{NA}}$	4.1(1.4)	1.000(0.000)	0.999(0.000)	0.950(0.012)	0.943(0.013)	0.932(0.017)	0.927(0.018)	0.925(0.018)
t-statistic	4.43E+01	4.43E+01	3.19E+01	1.25E+02	1.26E+02	9.83E+01	9.50E+01	9.15E+01
p-value	9.51E-32	9.51E-32	1.88E-26	7.85E-62	1.36E-68	1.09E-59	7.43E-59	2.31E-63
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**5.4(1.6)**	**0.999(0.000)**	**0.999(0.000)**	**0.926(0.019)**	**0.916(0.022)**	**0.895(0.025)**	**0.887(0.024)**	**0.880(0.027)**
t-statistic	4.23E+01	4.23E+01	3.01E+01	7.90E+01	7.71E+01	6.49E+01	6.78E+01	6.13E+01
p-value	3.90E-32	3.90E-32	1.50E-27	2.70E-43	6.13E-45	1.29E-42	3.82E-45	1.03E-45
$p < 0.05$	1	1	1	1	1	1	1	1
Citeseer								
ATP (log)	2344.6(56.3)	0.563(0.011)	0.565(0.014)	0.017(0.001)	0.005(0.001)	0.014(0.002)	0.019(0.003)	0.042(0.040)
ATP (harmonic)	2361.2(40.7)	0.559(0.008)	0.558(0.010)	0.017(0.000)	0.006(0.001)	0.014(0.002)	0.018(0.003)	0.027(0.038)
LINE	3258.9(25.1)	0.392(0.005)	0.410(0.002)	0.030(0.001)	0.037(0.001)	0.079(0.004)	0.068(0.005)	0.079(0.024)
$\text{G2G}_{\text{NA},K=1}$	7.3(1.6)	0.999(0.000)	0.998(0.000)	0.777(0.006)	0.679(0.010)	0.553(0.021)	0.530(0.024)	0.553(0.083)
NK	6.8(1.8)	0.999(0.000)	0.998(0.001)	0.763(0.006)	0.648(0.009)	0.591(0.019)	0.564(0.026)	0.600(0.068)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	34.2(3.6)	0.994(0.001)	0.994(0.001)	0.718(0.003)	0.641(0.006)	0.363(0.003)	0.251(0.001)	0.141(0.000)
$\text{G2G}_{K=1}$	4.7(1.5)	0.999(0.000)	0.999(0.001)	0.844(0.003)	0.772(0.006)	0.423(0.001)	0.285(0.001)	0.155(0.000)
$\text{G2G}_{K=3}$	18.1(3.2)	0.997(0.001)	0.993(0.002)	0.624(0.011)	0.518(0.015)	0.386(0.021)	0.338(0.023)	0.384(0.082)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	17.5(3.0)	0.997(0.001)	0.993(0.002)	0.612(0.013)	0.500(0.016)	0.393(0.018)	0.362(0.026)	0.454(0.069)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	6.8(1.8)	0.999(0.000)	0.998(0.001)	0.763(0.006)	0.648(0.009)	0.591(0.019)	0.564(0.026)	0.600(0.068)
$\text{HEADNet}_{\text{NA}}$	**2.5(1.2)**	**1.000(0.000)**	**1.000(0.000)**	**0.929(0.020)**	**0.896(0.027)**	**0.853(0.042)**	**0.851(0.046)**	**0.892(0.051)**
t-statistic	1.09E+01	1.09E+01	9.03E+00	4.50E+01	4.74E+01	3.09E+01	2.99E+01	1.89E+01
p-value	4.73E-15	4.73E-15	3.33E-12	4.74E-32	3.59E-34	3.88E-30	3.37E-32	1.07E-25
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	7.9(2.0)	0.999(0.000)	0.996(0.002)	0.752(0.025)	**0.679(0.035)**	**0.677(0.031)**	**0.659(0.031)**	**0.696(0.065)**
t-statistic	-2.30E+00	-2.34E+00	-7.33E+00	-2.33E+00	4.76E+00	1.30E+01	1.27E+01	5.58E+00
p-value	9.88E-01	9.89E-01	1.00E+00	9.87E-01	1.86E-05	8.93E-18	1.65E-18	3.31E-07
$p < 0.05$	0	0	0	0	1	1	1	1

TABLE C.4: Summary of the network reconstruction task on the Pubmed, Cora and Wiki Vote networks for an embedding dimension of 10+10. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. $\text{HEADNet}_{\Sigma=\mathbb{I}}$ denoted HEADNet with an identity variance. A dash (–) indicates that a network did not have attributes. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Pubmed								
ATP (log)	64056.3(217.4)	0.277(0.002)	0.380(0.001)	0.044(0.002)	0.032(0.003)	0.063(0.008)	0.072(0.009)	0.085(0.009)
ATP (harmonic)	63868.7(216.7)	0.280(0.002)	0.387(0.001)	0.047(0.002)	0.039(0.003)	0.068(0.007)	0.074(0.009)	0.086(0.009)
LINE	58572.6(362.6)	0.339(0.004)	0.390(0.002)	0.030(0.001)	0.064(0.001)	0.131(0.003)	0.158(0.004)	0.171(0.005)
$\text{G2G}_{\text{NA},K=1}$	221.6(22.3)	0.998(0.000)	0.996(0.000)	0.828(0.011)	0.779(0.012)	0.802(0.018)	0.827(0.014)	0.839(0.012)
NK	450.6(16.0)	0.995(0.000)	0.994(0.000)	0.738(0.002)	0.743(0.003)	0.512(0.001)	0.403(0.001)	0.276(0.000)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	24.1(3.4)	1.000(0.000)	1.000(0.000)	0.967(0.001)	0.964(0.001)	0.612(0.001)	0.471(0.000)	0.318(0.000)
$\text{G2G}_{K=1}$	1951.4(147.4)	0.978(0.002)	0.967(0.002)	0.252(0.016)	0.159(0.010)	0.261(0.011)	0.317(0.013)	0.377(0.014)
$\text{G2G}_{K=3}$	1525.9(89.5)	0.983(0.001)	0.974(0.001)	0.275(0.009)	0.199(0.008)	0.315(0.007)	0.374(0.007)	0.433(0.007)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	375.7(12.1)	0.996(0.000)	0.994(0.000)	0.567(0.001)	0.466(0.002)	0.395(0.001)	0.343(0.001)	0.259(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	**53.2(6.6)**	**0.999(0.000)**	**0.999(0.000)**	**0.908(0.004)**	**0.869(0.004)**	**0.899(0.009)**	**0.905(0.007)**	**0.911(0.006)**
$\text{HEADNet}_{\text{NA}}$	23.4(3.1)	1.000(0.000)	1.000(0.000)	0.953(0.002)	0.925(0.003)	0.970(0.002)	0.976(0.002)	0.979(0.002)
t-statistic	2.23E+01	2.23E+01	1.48E+01	5.87E+01	5.93E+01	4.36E+01	5.12E+01	6.24E+01
p-value	7.36E-25	7.36E-25	3.59E-19	4.36E-40	1.33E-48	2.80E-30	2.34E-32	5.41E-38
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	373.6(26.7)	0.996(0.000)	0.994(0.000)	0.544(0.011)	0.439(0.009)	0.625(0.007)	0.681(0.007)	0.725(0.006)
t-statistic	-6.37E+01	-6.37E+01	-5.55E+01	-1.78E+02	-2.40E+02	-1.33E+02	-1.25E+02	-1.19E+02
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0	0	0	0	0
Cora								
ATP (log)	33156.7(72.0)	0.492(0.001)	0.507(0.001)	0.037(0.000)	0.026(0.001)	0.031(0.001)	0.038(0.001)	0.053(0.002)
ATP (harmonic)	32868.4(71.0)	0.497(0.001)	0.512(0.001)	0.043(0.000)	0.038(0.001)	0.039(0.001)	0.043(0.001)	0.055(0.002)
LINE	30716.1(113.5)	0.530(0.002)	0.492(0.001)	0.103(0.002)	0.164(0.003)	0.185(0.003)	0.198(0.004)	0.216(0.006)
$\text{G2G}_{\text{NA},K=1}$	145.6(13.0)	0.998(0.000)	0.997(0.000)	0.797(0.013)	0.782(0.014)	0.727(0.017)	0.710(0.017)	0.707(0.014)
NK	458.3(14.8)	0.993(0.000)	0.993(0.000)	0.728(0.002)	0.777(0.002)	0.566(0.001)	0.443(0.001)	0.283(0.000)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	26.6(3.4)	1.000(0.000)	0.999(0.000)	0.922(0.001)	0.914(0.002)	0.685(0.001)	0.536(0.000)	0.334(0.000)
$\text{G2G}_{K=1}$	201.6(19.9)	0.997(0.000)	0.996(0.000)	0.742(0.017)	0.720(0.021)	0.658(0.020)	0.647(0.019)	0.654(0.017)
$\text{G2G}_{K=3}$	129.2(14.3)	0.998(0.000)	0.997(0.000)	0.803(0.015)	0.789(0.017)	0.745(0.018)	0.729(0.016)	0.725(0.014)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	35.2(3.5)	0.999(0.000)	0.999(0.000)	0.907(0.001)	0.897(0.002)	0.676(0.001)	0.531(0.001)	0.333(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	**90.1(13.1)**	**0.999(0.000)**	**0.998(0.000)**	**0.854(0.011)**	**0.842(0.011)**	**0.807(0.015)**	**0.786(0.015)**	**0.767(0.014)**
$\text{HEADNet}_{\text{NA}}$	9.9(1.9)	1.000(0.000)	1.000(0.000)	0.980(0.002)	0.981(0.002)	0.973(0.003)	0.967(0.003)	0.959(0.004)
t-statistic	3.32E+01	3.32E+01	2.94E+01	6.00E+01	6.60E+01	5.90E+01	6.41E+01	7.27E+01
p-value	1.24E-25	1.24E-25	1.38E-24	1.07E-33	4.27E-35	5.20E-34	2.65E-35	2.06E-39
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**16.0(3.6)**	**1.000(0.000)**	**1.000(0.000)**	**0.961(0.005)**	**0.959(0.006)**	**0.951(0.006)**	**0.945(0.006)**	**0.940(0.006)**
t-statistic	2.99E+01	2.99E+01	2.68E+01	4.66E+01	4.96E+01	4.85E+01	5.40E+01	6.30E+01
p-value	6.14E-26	6.14E-26	3.37E-24	6.43E-38	1.76E-41	3.66E-36	3.07E-37	2.45E-41
$p < 0.05$	1	1	1	1	1	1	1	1
Wiki Vote								
ATP (log)	53405.2(176.9)	0.485(0.002)	0.505(0.001)	0.030(0.000)	0.013(0.001)	0.022(0.001)	0.027(0.001)	0.041(0.001)
ATP (harmonic)	53418.3(189.4)	0.485(0.002)	0.505(0.001)	0.030(0.000)	0.013(0.001)	0.022(0.001)	0.027(0.001)	0.041(0.001)
LINE	70912.2(270.8)	0.316(0.003)	0.379(0.001)	0.036(0.001)	0.078(0.003)	0.146(0.002)	0.178(0.002)	0.213(0.003)
$\text{G2G}_{\text{NA},K=1}$	4179.2(301.6)	0.960(0.003)	0.947(0.003)	0.504(0.019)	0.437(0.022)	0.361(0.019)	0.361(0.015)	0.391(0.013)
$\text{G2G}_{\text{NA},K=3}$	4522.1(145.7)	0.956(0.001)	0.935(0.003)	0.221(0.012)	0.170(0.010)	0.221(0.007)	0.255(0.006)	0.305(0.007)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	1681.3(31.2)	0.984(0.000)	0.977(0.001)	0.656(0.001)	0.637(0.004)	0.473(0.002)	0.391(0.001)	0.300(0.001)
$\text{G2G}_{K=1}$	–	–	–	–	–	–	–	–
$\text{G2G}_{K=3}$	–	–	–	–	–	–	–	–
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	–	–	–	–	–	–	–	–
Significance Test								
NK	4419.5(33.2)	0.957(0.000)	0.962(0.000)	0.185(0.001)	0.252(0.004)	0.216(0.001)	0.201(0.001)	0.182(0.001)
$\text{HEADNet}_{\text{NA}}$	**426.9(18.1)**	**0.996(0.000)**	**0.994(0.000)**	**0.858(0.006)**	**0.866(0.013)**	**0.781(0.006)**	**0.764(0.005)**	**0.766(0.004)**
t-statistic	5.78E+02	5.78E+02	3.25E+02	5.70E+02	2.41E+02	5.17E+02	6.35E+02	7.35E+02
p-value	9.94E-89	9.95E-89	5.43E-96	2.30E-63	4.87E-56	1.93E-64	2.64E-66	8.11E-67
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	–	–	–	–	–	–	–	–
t-statistic	–	–	–	–	–	–	–	–
p-value	–	–	–	–	–	–	–	–
$p < 0.05$	–	–	–	–	–	–	–	–

TABLE C.5: Summary of the network reconstruction task on synthetic, Cora_ML and Citeseer networks for an embedding dimension of 50+50. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. A dash (–) indicates that a network did not have attributes. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Synthetic								
ATP (log)	695.1(59.6)	0.597(0.037)	0.624(0.036)	0.017(0.011)	0.006(0.018)	0.012(0.014)	0.019(0.014)	0.028(0.018)
ATP (harmonic)	705.8(80.1)	0.591(0.046)	0.616(0.042)	0.016(0.011)	0.006(0.018)	0.011(0.014)	0.018(0.014)	0.028(0.018)
LINE	1229.9(52.0)	0.288(0.016)	0.376(0.005)	0.010(0.002)	0.019(0.006)	0.054(0.016)	0.072(0.025)	0.083(0.028)
$\text{G2G}_{\text{NA},K=1}$	83.7(11.4)	0.952(0.007)	0.897(0.016)	0.223(0.021)	0.147(0.019)	0.133(0.026)	0.083(0.027)	0.053(0.018)
$\text{G2G}_{\text{NA},K=3}$	105.2(15.1)	0.940(0.009)	0.864(0.026)	0.131(0.039)	0.085(0.037)	0.119(0.053)	0.153(0.047)	0.296(0.074)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	7.1(1.7)	0.996(0.001)	0.994(0.002)	0.701(0.038)	0.609(0.056)	0.348(0.013)	0.242(0.010)	0.146(0.006)
$\text{G2G}_{K=1}$	–	–	–	–	–	–	–	–
$\text{G2G}_{K=3}$	–	–	–	–	–	–	–	–
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	–	–	–	–	–	–	–	–
Significance Test								
NK	49.2(9.2)	0.972(0.006)	0.954(0.009)	0.281(0.019)	0.258(0.017)	0.166(0.011)	0.127(0.009)	0.087(0.006)
$\text{HEADNet}_{\text{NA}}$	**2.6(1.1)**	**0.999(0.001)**	**0.998(0.001)**	**0.887(0.048)**	**0.834(0.072)**	**0.839(0.074)**	**0.842(0.071)**	**0.860(0.075)**
t-statistic	2.75E+01	2.46E+01	2.80E+01	6.43E+01	4.27E+01	4.91E+01	5.43E+01	5.63E+01
p-value	4.82E-23	1.46E-21	1.87E-23	9.16E-41	3.39E-30	1.06E-30	9.00E-32	8.14E-32
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	–	–	–	–	–	–	–	–
t-statistic	–	–	–	–	–	–	–	–
p-value	–	–	–	–	–	–	–	–
$p < 0.05$	–	–	–	–	–	–	–	–
Cora_ML								
ATP (log)	4083.0(25.8)	0.515(0.003)	0.531(0.003)	0.044(0.001)	0.025(0.002)	0.042(0.002)	0.058(0.002)	0.085(0.006)
ATP (harmonic)	4041.1(25.7)	0.520(0.003)	0.538(0.003)	0.045(0.001)	0.030(0.002)	0.044(0.002)	0.058(0.002)	0.084(0.005)
LINE	4093.9(40.6)	0.514(0.005)	0.473(0.003)	0.060(0.002)	0.097(0.004)	0.110(0.004)	0.118(0.004)	0.124(0.007)
$\text{G2G}_{\text{NA},K=1}$	54.4(4.9)	0.994(0.001)	0.991(0.001)	0.609(0.007)	0.551(0.014)	0.530(0.011)	0.517(0.012)	0.495(0.017)
NK	101.7(3.7)	0.988(0.000)	0.987(0.001)	0.586(0.004)	0.607(0.007)	0.431(0.003)	0.339(0.002)	0.220(0.001)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	10.6(2.6)	0.999(0.000)	0.998(0.001)	0.823(0.003)	0.776(0.006)	0.577(0.002)	0.448(0.001)	0.277(0.000)
$\text{G2G}_{K=1}$	64.6(7.7)	0.992(0.001)	0.989(0.001)	0.586(0.008)	0.536(0.013)	0.493(0.011)	0.479(0.013)	0.459(0.018)
$\text{G2G}_{K=3}$	55.8(5.5)	0.993(0.001)	0.991(0.001)	0.609(0.011)	0.563(0.011)	0.539(0.014)	0.530(0.013)	0.509(0.013)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	13.4(2.7)	0.999(0.000)	0.998(0.001)	0.792(0.003)	0.743(0.006)	0.553(0.003)	0.432(0.001)	0.270(0.001)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	44.0(5.6)	0.995(0.001)	0.992(0.001)	0.647(0.007)	0.588(0.011)	0.587(0.009)	0.575(0.010)	0.548(0.012)
$\text{HEADNet}_{\text{NA}}$	**3.2(1.4)**	**1.000(0.000)**	**1.000(0.000)**	**0.967(0.008)**	**0.963(0.009)**	**0.956(0.012)**	**0.951(0.012)**	**0.949(0.013)**
t-statistic	3.90E+01	3.90E+01	2.78E+01	1.68E+02	1.38E+02	1.35E+02	1.36E+02	1.22E+02
p-value	2.38E-29	2.38E-29	9.02E-25	1.00E-78	4.26E-73	1.13E-71	5.47E-73	2.43E-71
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**4.0(1.4)**	**1.000(0.000)**	**0.999(0.000)**	**0.945(0.014)**	**0.937(0.016)**	**0.921(0.019)**	**0.912(0.021)**	**0.904(0.021)**
t-statistic	3.83E+01	3.83E+01	2.71E+01	1.05E+02	9.85E+01	8.74E+01	8.01E+01	8.00E+01
p-value	7.09E-29	7.09E-29	1.09E-24	4.77E-53	4.89E-62	5.58E-50	3.62E-47	3.39E-51
$p < 0.05$	1	1	1	1	1	1	1	1
Citeseer								
ATP (log)	2171.6(19.9)	0.595(0.004)	0.601(0.006)	0.026(0.001)	0.008(0.001)	0.032(0.003)	0.039(0.004)	0.034(0.006)
ATP (harmonic)	2145.5(21.2)	0.600(0.004)	0.604(0.006)	0.026(0.000)	0.008(0.001)	0.030(0.002)	0.034(0.005)	0.010(0.016)
LINE	3156.9(29.5)	0.411(0.006)	0.418(0.002)	0.052(0.002)	0.053(0.002)	0.057(0.004)	0.050(0.004)	0.010(0.016)
$\text{G2G}_{\text{NA},K=1}$	6.6(1.5)	0.999(0.000)	0.999(0.000)	0.814(0.009)	0.734(0.015)	0.611(0.012)	0.584(0.017)	0.570(0.066)
NK	26.1(1.9)	0.995(0.000)	0.995(0.001)	0.722(0.004)	0.645(0.007)	0.365(0.003)	0.252(0.001)	0.141(0.000)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	3.5(1.2)	1.000(0.000)	0.999(0.000)	0.892(0.002)	0.839(0.005)	0.447(0.001)	0.300(0.001)	0.159(0.000)
$\text{G2G}_{K=1}$	17.2(3.2)	0.997(0.001)	0.993(0.002)	0.632(0.010)	0.530(0.014)	0.395(0.014)	0.353(0.020)	0.424(0.069)
$\text{G2G}_{K=3}$	17.0(2.8)	0.997(0.001)	0.993(0.002)	0.622(0.009)	0.513(0.014)	0.411(0.016)	0.371(0.023)	0.473(0.066)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	13.6(1.8)	0.998(0.000)	0.994(0.002)	0.670(0.003)	0.571(0.005)	0.358(0.002)	0.253(0.001)	0.144(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	6.1(1.8)	0.999(0.000)	0.999(0.001)	0.790(0.010)	0.686(0.017)	0.634(0.016)	0.612(0.019)	0.634(0.054)
$\text{HEADNet}_{\text{NA}}$	**2.2(0.8)**	**1.000(0.000)**	**1.000(0.000)**	**0.947(0.016)**	**0.922(0.022)**	**0.894(0.036)**	**0.893(0.036)**	**0.934(0.042)**
t-statistic	1.09E+01	1.09E+01	8.05E+00	4.58E+01	4.64E+01	3.57E+01	3.75E+01	2.40E+01
p-value	6.12E-14	6.12E-14	3.84E-11	5.36E-43	1.01E-45	1.94E-32	1.26E-35	6.59E-31
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**7.3(2.1)**	**0.999(0.000)**	**0.996(0.002)**	**0.779(0.019)**	**0.718(0.028)**	**0.709(0.024)**	**0.685(0.023)**	**0.711(0.084)**
t-statistic	-2.41E+00	-2.45E+00	-7.60E+00	-2.66E+00	5.38E+00	1.40E+01	1.31E+01	4.19E+00
p-value	9.90E-01	9.91E-01	1.00E+00	9.95E-01	1.14E-06	2.22E-19	5.94E-19	5.64E-05
$p < 0.05$	0	0	0	0	1	1	1	1

TABLE C.6: Summary of the network reconstruction task on the Pubmed, Cora and Wiki Vote networks for an embedding dimension of 50+50. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. $\text{HEADNet}_{\Sigma=\mathbb{I}}$ denoted HEADNet with an identity variance. A dash (–) indicates that a network did not have attributes. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Pubmed								
ATP (log)	63968.3(223.6)	0.278(0.003)	0.381(0.001)	0.074(0.003)	0.076(0.005)	0.147(0.009)	0.160(0.010)	0.159(0.010)
ATP (harmonic)	63783.3(222.1)	0.280(0.003)	0.387(0.001)	0.075(0.003)	0.078(0.005)	0.149(0.009)	0.161(0.010)	0.159(0.010)
LINE	63021.9(209.2)	0.289(0.002)	0.370(0.001)	0.022(0.000)	0.058(0.001)	0.100(0.002)	0.105(0.003)	0.090(0.003)
$\text{G2G}_{\text{NA},K=1}$	179.4(10.7)	0.998(0.000)	0.997(0.000)	0.850(0.005)	0.805(0.005)	0.838(0.009)	0.856(0.008)	0.866(0.006)
NK	397.6(10.6)	0.996(0.000)	0.994(0.000)	0.754(0.001)	0.759(0.002)	0.522(0.001)	0.410(0.001)	0.280(0.000)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	14.3(2.9)	1.000(0.000)	1.000(0.000)	0.972(0.001)	0.951(0.001)	0.630(0.000)	0.484(0.000)	0.326(0.000)
$\text{G2G}_{K=1}$	1690.2(106.9)	0.981(0.001)	0.972(0.002)	0.286(0.017)	0.182(0.011)	0.286(0.011)	0.344(0.013)	0.405(0.014)
$\text{G2G}_{K=3}$	1365.3(65.8)	0.985(0.001)	0.976(0.001)	0.293(0.007)	0.210(0.005)	0.326(0.004)	0.387(0.006)	0.448(0.007)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	257.6(11.1)	0.997(0.000)	0.996(0.000)	0.636(0.001)	0.543(0.002)	0.438(0.001)	0.372(0.000)	0.275(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	46.9(5.4)	0.999(0.000)	0.999(0.000)	0.915(0.002)	0.877(0.003)	0.918(0.004)	0.922(0.003)	0.924(0.003)
$\text{HEADNet}_{\text{NA}}$	**17.9(2.8)**	**1.000(0.000)**	**1.000(0.000)**	**0.963(0.003)**	**0.938(0.006)**	**0.982(0.001)**	**0.989(0.001)**	**0.993(0.001)**
t-statistic	2.62E+01	2.62E+01	1.63E+01	7.50E+01	5.19E+01	7.63E+01	9.85E+01	1.27E+02
p-value	8.81E-29	8.81E-29	6.66E-20	1.24E-58	5.67E-44	1.44E-41	6.29E-47	1.26E-53
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	246.0(15.4)	0.997(0.000)	0.996(0.000)	0.641(0.007)	0.539(0.007)	0.699(0.005)	0.747(0.004)	0.786(0.004)
t-statistic	-6.70E+01	-6.70E+01	-4.92E+01	-2.18E+02	-2.44E+02	-1.79E+02	-1.76E+02	-1.51E+02
p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	0	0	0	0	0	0	0	0
Cora								
ATP (log)	32870.8(69.0)	0.497(0.001)	0.510(0.001)	0.061(0.000)	0.058(0.001)	0.072(0.001)	0.082(0.001)	0.098(0.001)
ATP (harmonic)	32603.2(67.5)	0.501(0.001)	0.516(0.001)	0.064(0.000)	0.065(0.001)	0.075(0.001)	0.084(0.001)	0.098(0.002)
LINE	34214.9(129.9)	0.476(0.002)	0.455(0.001)	0.088(0.001)	0.160(0.002)	0.162(0.002)	0.163(0.002)	0.155(0.002)
$\text{G2G}_{\text{NA},K=1}$	130.2(8.9)	0.998(0.000)	0.997(0.000)	0.814(0.007)	0.800(0.008)	0.750(0.009)	0.732(0.009)	0.725(0.008)
NK	385.2(9.3)	0.994(0.000)	0.994(0.000)	0.744(0.001)	0.790(0.002)	0.577(0.001)	0.451(0.001)	0.287(0.000)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	14.9(2.5)	1.000(0.000)	1.000(0.000)	0.957(0.001)	0.941(0.001)	0.716(0.001)	0.560(0.000)	0.343(0.000)
$\text{G2G}_{K=1}$	178.6(19.4)	0.997(0.000)	0.996(0.000)	0.766(0.012)	0.748(0.014)	0.688(0.015)	0.675(0.014)	0.679(0.011)
$\text{G2G}_{K=3}$	118.6(11.4)	0.998(0.000)	0.998(0.000)	0.816(0.011)	0.804(0.012)	0.761(0.014)	0.744(0.013)	0.738(0.011)
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	22.0(2.9)	1.000(0.000)	1.000(0.000)	0.940(0.001)	0.930(0.002)	0.703(0.001)	0.550(0.000)	0.340(0.000)
Significance Test								
$\text{G2G}_{\text{NA},K=3}$	79.5(6.8)	0.999(0.000)	0.998(0.000)	0.868(0.006)	0.856(0.006)	0.827(0.009)	0.806(0.009)	0.786(0.008)
$\text{HEADNet}_{\text{NA}}$	**6.1(1.8)**	**1.000(0.000)**	**1.000(0.000)**	**0.990(0.002)**	**0.990(0.002)**	**0.986(0.003)**	**0.983(0.003)**	**0.978(0.004)**
t-statistic	5.76E+01	5.76E+01	4.76E+01	1.05E+02	1.24E+02	9.16E+01	9.94E+01	1.15E+02
p-value	6.81E-35	6.81E-35	4.57E-34	2.54E-44	6.06E-46	6.72E-42	8.09E-47	3.44E-56
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	**10.9(2.8)**	**1.000(0.000)**	**1.000(0.000)**	**0.974(0.005)**	**0.973(0.005)**	**0.967(0.006)**	**0.962(0.006)**	**0.958(0.006)**
t-statistic	5.13E+01	5.13E+01	4.28E+01	7.35E+01	8.05E+01	7.08E+01	7.78E+01	9.52E+01
p-value	1.59E-37	1.59E-37	1.18E-35	1.73E-57	2.03E-61	5.65E-51	1.01E-54	2.19E-60
$p < 0.05$	1	1	1	1	1	1	1	1
Wiki Vote								
ATP (log)	53778.0(206.8)	0.481(0.002)	0.503(0.002)	0.031(0.000)	0.013(0.001)	0.022(0.001)	0.028(0.001)	0.042(0.001)
ATP (harmonic)	53758.4(200.7)	0.482(0.002)	0.503(0.001)	0.031(0.000)	0.013(0.001)	0.022(0.001)	0.027(0.001)	0.042(0.001)
LINE	72724.9(193.6)	0.299(0.002)	0.373(0.001)	0.031(0.001)	0.063(0.002)	0.115(0.003)	0.136(0.004)	0.157(0.004)
$\text{G2G}_{\text{NA},K=1}$	3853.7(244.3)	0.963(0.002)	0.951(0.003)	0.563(0.008)	0.524(0.013)	0.431(0.013)	0.408(0.011)	0.419(0.009)
$\text{G2G}_{\text{NA},K=3}$	4714.5(176.9)	0.955(0.002)	0.931(0.003)	0.220(0.013)	0.180(0.013)	0.231(0.011)	0.265(0.009)	0.314(0.009)
$\text{HEADNet}_{\text{NA},\Sigma=\mathbb{I}}$	592.7(25.9)	0.994(0.000)	0.992(0.000)	0.854(0.007)	0.818(0.011)	0.606(0.002)	0.503(0.002)	0.384(0.002)
$\text{G2G}_{K=1}$	–	–	–	–	–	–	–	–
$\text{G2G}_{K=3}$	–	–	–	–	–	–	–	–
$\text{HEADNet}_{\Sigma=\mathbb{I}}$	–	–	–	–	–	–	–	–
Significance Test								
NK	4184.2(24.7)	0.960(0.000)	0.964(0.000)	0.212(0.001)	0.286(0.003)	0.235(0.001)	0.214(0.001)	0.191(0.001)
$\text{HEADNet}_{\text{NA}}$	**59.9(9.5)**	**0.999(0.000)**	**0.999(0.000)**	**0.966(0.006)**	**0.948(0.010)**	**0.990(0.003)**	**0.988(0.003)**	**0.983(0.004)**
t-statistic	8.54E+02	8.54E+02	4.42E+02	6.36E+02	3.31E+02	1.43E+03	1.33E+03	1.03E+03
p-value	4.87E-82	4.87E-82	3.87E-90	2.94E-63	6.58E-63	1.05E-104	4.82E-84	8.98E-73
$p < 0.05$	1	1	1	1	1	1	1	1
HEADNet	–	–	–	–	–	–	–	–
t-statistic	–	–	–	–	–	–	–	–
p-value	–	–	–	–	–	–	–	–
$p < 0.05$	–	–	–	–	–	–	–	–

TABLE C.7: Summary of the link prediction task on synthetic, Cora_ML, Citeseer, Pubmed, Cora and Wiki Vote networks for an embedding dimension of 5+5. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. A dash (–) indicates that a network did not have attributes. All measures are reported to 3 decimal places. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

Synthetic (∼173 edges removed)					Pubmed (8865 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	88.6(8.7)	0.494(0.041)	0.567(0.040)	0.016(0.020)	ATP (log)	7692.5(23.8)	0.132(0.003)	0.327(0.001)	0.032(0.003)
ATP (harmonic)	85.9(8.4)	0.510(0.040)	0.574(0.040)	0.016(0.020)	ATP (harmonic)	7680.9(24.4)	0.134(0.003)	0.330(0.001)	0.035(0.003)
LINE	120.9(7.8)	0.307(0.035)	0.384(0.012)	0.020(0.008)	LINE	4880.6(62.5)	0.450(0.007)	0.433(0.003)	0.083(0.004)
$\text{G2G}_{\text{NA},K=1}$	48.9(6.4)	0.722(0.042)	0.650(0.049)	0.012(0.007)	$\text{G2G}_{\text{NA},K=1}$	98.5(13.3)	0.989(0.002)	0.987(0.002)	0.412(0.029)
NK	43.2(10.2)	0.754(0.067)	0.693(0.070)	0.017(0.007)	NK	150.5(8.8)	0.983(0.001)	0.982(0.001)	0.426(0.005)
$\text{G2G}_{K=1}$	–	–	–	–	$\text{G2G}_{K=1}$	313.0(20.3)	0.965(0.002)	0.953(0.003)	0.138(0.006)
$\text{G2G}_{K=3}$	–	–	–	–	$\text{G2G}_{K=3}$	250.1(16.1)	0.972(0.002)	0.962(0.003)	0.155(0.004)
Significance Test					Significance Test				
$\text{G2G}_{\text{NA},K=3}$	34.7(5.5)	0.805(0.036)	0.765(0.048)	0.022(0.008)	$\text{G2G}_{\text{NA},K=3}$	53.7(6.0)	0.994(0.001)	0.993(0.001)	0.494(0.010)
$\text{HEADNet}_{\text{NA}}$	**27.7(4.5)**	**0.845(0.029)**	**0.842(0.037)**	**0.046(0.010)**	$\text{HEADNet}_{\text{NA}}$	128.3(8.4)	0.986(0.001)	0.986(0.001)	0.474(0.006)
t-statistic	5.37E+00	4.87E+00	6.94E+00	1.05E+01	*t*-statistic	-3.95E+01	-3.95E+01	-2.85E+01	-8.95E+00
p-value	7.78E-07	4.88E-06	2.40E-09	5.81E-15	*p*-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	1	1	1	1	$p < 0.05$	0	0	0	0
HEADNet	–	–	–	–	HEADNet	137.5(9.7)	0.985(0.001)	0.980(0.002)	0.302(0.007)
t-statistic	–	–	–	–	*t*-statistic	-4.03E+01	-4.03E+01	-3.70E+01	-8.23E+01
p-value	–	–	–	–	*p*-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	–	–	–	–	$p < 0.05$	0	0	0	0
Cora_ML (842 edges removed)					Cora (6532 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	507.2(12.0)	0.399(0.014)	0.440(0.009)	0.017(0.002)	ATP (log)	3969.1(34.9)	0.393(0.005)	0.433(0.004)	0.018(0.001)
ATP (harmonic)	502.7(12.1)	0.404(0.014)	0.449(0.010)	0.020(0.003)	ATP (harmonic)	3942.2(35.3)	0.397(0.005)	0.438(0.004)	0.022(0.001)
LINE	400.4(14.1)	0.526(0.017)	0.479(0.010)	0.085(0.008)	LINE	2934.7(47.5)	0.551(0.007)	0.499(0.005)	0.123(0.004)
$\text{G2G}_{\text{NA},K=1}$	30.9(4.6)	0.965(0.005)	0.960(0.008)	0.210(0.018)	$\text{G2G}_{\text{NA},K=1}$	87.5(10.2)	0.987(0.002)	0.987(0.002)	0.424(0.017)
$\text{G2G}_{\text{NA},K=3}$	29.2(4.3)	0.967(0.005)	0.962(0.007)	0.219(0.014)	NK	156.8(12.5)	0.976(0.002)	0.979(0.002)	0.442(0.006)
NK	36.9(3.9)	0.957(0.005)	0.958(0.006)	0.251(0.013)	$\text{G2G}_{K=1}$	78.3(9.6)	0.988(0.001)	0.987(0.002)	0.386(0.021)
$\text{G2G}_{K=1}$	26.7(3.6)	0.969(0.004)	0.963(0.007)	0.213(0.016)	$\text{G2G}_{K=3}$	73.4(5.8)	0.989(0.001)	0.988(0.001)	0.401(0.013)
Significance Test					Significance Test				
$\text{G2G}_{K=3}$	26.2(3.6)	0.970(0.004)	0.964(0.007)	0.219(0.015)	$\text{G2G}_{\text{NA},K=3}$	80.2(8.0)	0.988(0.001)	0.988(0.001)	0.445(0.015)
$\text{HEADNet}_{\text{NA}}$	38.5(3.7)	0.956(0.004)	0.959(0.005)	**0.302(0.016)**	$\text{HEADNet}_{\text{NA}}$	156.4(7.8)	0.976(0.001)	0.980(0.001)	**0.499(0.007)**
t-statistic	-1.30E+01	-1.30E+01	-2.81E+00	2.06E+01	*t*-statistic	-3.72E+01	-3.72E+01	-2.43E+01	1.74E+01
p-value	1.00E+00	1.00E+00	9.97E-01	1.49E-28	*p*-value	1.00E+00	1.00E+00	1.00E+00	3.68E-21
$p < 0.05$	0	0	0	1	$p < 0.05$	0	0	0	1
HEADNet	25.3(3.5)	0.971(0.004)	0.971(0.005)	**0.341(0.022)**	HEADNet	90.9(5.6)	0.986(0.001)	0.987(0.001)	**0.540(0.010)**
t-statistic	9.25E-01	9.25E-01	5.07E+00	2.50E+01	*t*-statistic	-6.03E+00	-6.03E+00	-3.25E+00	2.86E+01
p-value	1.79E-01	1.79E-01	2.70E-06	1.10E-30	*p*-value	1.00E+00	1.00E+00	9.99E-01	7.71E-33
$p < 0.05$	0	0	1	1	$p < 0.05$	0	0	0	1
Citeseer (536 edges removed)					Wiki Vote (10369 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	319.1(13.6)	0.406(0.025)	0.463(0.025)	0.018(0.002)	ATP (log)	5742.5(36.3)	0.446(0.003)	0.472(0.003)	0.013(0.001)
ATP (harmonic)	318.4(12.3)	0.408(0.023)	0.462(0.020)	0.019(0.003)	ATP (harmonic)	5745.5(36.2)	0.446(0.003)	0.472(0.003)	0.013(0.001)
LINE	349.0(8.8)	0.351(0.017)	0.395(0.007)	0.045(0.006)	LINE	6932.9(74.9)	0.331(0.007)	0.384(0.003)	0.055(0.003)
$\text{G2G}_{\text{NA},K=1}$	29.7(4.6)	0.947(0.009)	0.961(0.007)	0.258(0.015)	$\text{G2G}_{\text{NA},K=1}$	576.0(51.3)	0.945(0.005)	0.929(0.006)	0.094(0.007)
$\text{G2G}_{\text{NA},K=3}$	29.1(4.8)	0.948(0.009)	0.961(0.008)	0.239(0.014)	$\text{G2G}_{\text{NA},K=3}$	468.8(20.8)	0.955(0.002)	0.937(0.003)	0.105(0.003)
NK	53.5(6.0)	0.902(0.011)	0.925(0.008)	0.255(0.011)	$\text{G2G}_{K=1}$	–	–	–	–
$\text{G2G}_{K=1}$	15.6(3.0)	0.973(0.006)	0.970(0.010)	0.248(0.017)	$\text{G2G}_{K=3}$	–	–	–	–
Significance Test					Significance Test				
$\text{G2G}_{K=3}$	15.4(2.7)	0.973(0.005)	0.971(0.009)	0.235(0.016)	NK	223.7(8.5)	0.979(0.001)	0.977(0.001)	0.169(0.003)
$\text{HEADNet}_{\text{NA}}$	38.6(4.3)	0.930(0.008)	0.941(0.008)	**0.283(0.025)**	$\text{HEADNet}_{\text{NA}}$	**184.0(9.6)**	**0.982(0.001)**	0.976(0.001)	**0.186(0.003)**
t-statistic	-2.47E+01	-2.47E+01	-1.33E+01	8.86E+00	*t*-statistic	1.70E+01	1.70E+01	-1.98E+00	2.02E+01
p-value	1.00E+00	1.00E+00	1.00E+00	4.55E-12	*p*-value	2.59E-24	2.59E-24	9.74E-01	3.89E-28
$p < 0.05$	0	0	0	1	$p < 0.05$	1	1	0	1
HEADNet	16.3(3.0)	0.971(0.006)	0.968(0.009)	**0.332(0.023)**	HEADNet	–	–	–	–
t-statistic	-1.13E+00	-1.13E+00	-9.74E-01	1.92E+01	*t*-statistic	–	–	–	–
p-value	8.69E-01	8.69E-01	8.33E-01	1.20E-25	*p*-value	–	–	–	–
$p < 0.05$	0	0	0	1	$p < 0.05$	–	–	–	–

TABLE C.8: Summary of the link prediction task on synthetic, Cora_ML, Citeseer, Pubmed, Cora and Wiki Vote networks for an embedding dimension of 10+10. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. A dash (–) indicates that a network did not have attributes. All measures are reported to 3 decimal places. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

Synthetic (∼173 edges removed)					Pubmed (8865 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	87.9(8.1)	0.498(0.039)	0.575(0.040)	0.016(0.020)	ATP (log)	7693.3(24.0)	0.132(0.003)	0.327(0.001)	0.051(0.003)
ATP (harmonic)	85.8(7.8)	0.509(0.039)	0.582(0.040)	0.016(0.020)	ATP (harmonic)	7682.3(24.6)	0.134(0.003)	0.330(0.001)	0.053(0.003)
LINE	120.6(8.0)	0.309(0.037)	0.385(0.013)	0.021(0.007)	LINE	4707.8(48.2)	0.469(0.005)	0.441(0.003)	0.060(0.003)
$\text{G2G}_{\text{NA},K=1}$	48.5(7.2)	0.725(0.048)	0.648(0.051)	0.010(0.005)	$\text{G2G}_{\text{NA},K=1}$	76.3(9.1)	0.992(0.001)	0.990(0.002)	0.467(0.018)
NK	35.5(10.2)	0.799(0.065)	0.735(0.075)	0.026(0.008)	NK	118.2(6.0)	0.987(0.001)	0.986(0.001)	0.461(0.005)
$\text{G2G}_{K=1}$	–	–	–	–	$\text{G2G}_{K=1}$	299.3(19.2)	0.966(0.002)	0.956(0.003)	0.143(0.005)
$\text{G2G}_{K=3}$	–	–	–	–	$\text{G2G}_{K=3}$	230.9(9.5)	0.974(0.001)	0.965(0.002)	0.169(0.005)
Significance Test					Significance Test				
$\text{G2G}_{\text{NA},K=3}$	33.9(5.6)	0.810(0.035)	0.760(0.040)	0.023(0.009)	$\text{G2G}_{\text{NA},K=3}$	42.5(5.0)	0.995(0.001)	0.995(0.001)	0.578(0.011)
$\text{HEADNet}_{\text{NA}}$	**24.6(3.7)**	**0.863(0.024)**	**0.859(0.029)**	**0.050(0.011)**	$\text{HEADNet}_{\text{NA}}$	52.0(6.4)	0.994(0.001)	0.994(0.001)	**0.643(0.014)**
t-statistic	7.59E+00	6.89E+00	1.10E+01	1.06E+01	*t*-statistic	-6.38E+00	-6.38E+00	-4.48E+00	2.06E+01
p-value	3.51E-10	3.67E-09	1.44E-15	2.66E-15	*p*-value	1.00E+00	1.00E+00	1.00E+00	1.17E-27
$p < 0.05$	1	1	1	1	$p < 0.05$	0	0	0	1
HEADNet	–	–	–	–	HEADNet	89.3(8.0)	0.990(0.001)	0.987(0.001)	0.374(0.008)
t-statistic	–	–	–	–	*t*-statistic	-2.71E+01	-2.71E+01	-2.60E+01	-8.41E+01
p-value	–	–	–	–	*p*-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	–	–	–	–	$p < 0.05$	0	0	0	0

Cora_ML (842 edges removed)					Cora (6532 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	505.1(12.4)	0.401(0.015)	0.441(0.010)	0.022(0.003)	ATP (log)	3962.6(34.9)	0.394(0.005)	0.433(0.004)	0.026(0.001)
ATP (harmonic)	500.8(12.7)	0.406(0.015)	0.450(0.011)	0.024(0.003)	ATP (harmonic)	3935.9(35.3)	0.398(0.005)	0.438(0.004)	0.029(0.002)
LINE	397.6(13.6)	0.529(0.016)	0.480(0.010)	0.079(0.009)	LINE	2933.3(38.0)	0.551(0.006)	0.498(0.003)	0.104(0.004)
$\text{G2G}_{\text{NA},K=1}$	28.0(3.8)	0.968(0.005)	0.964(0.006)	0.233(0.013)	$\text{G2G}_{\text{NA},K=1}$	80.2(8.5)	0.988(0.001)	0.988(0.002)	0.440(0.014)
$\text{G2G}_{\text{NA},K=3}$	27.6(3.6)	0.968(0.004)	0.965(0.006)	0.233(0.014)	NK	121.3(8.2)	0.982(0.001)	0.983(0.001)	0.469(0.005)
NK	31.6(3.7)	0.964(0.004)	0.963(0.005)	0.266(0.012)	$\text{G2G}_{K=1}$	71.4(5.6)	0.989(0.001)	0.988(0.001)	0.410(0.015)
$\text{G2G}_{K=3}$	24.8(3.2)	0.972(0.004)	0.966(0.006)	0.238(0.016)	$\text{G2G}_{K=3}$	66.4(5.0)	0.990(0.001)	0.989(0.001)	0.434(0.013)
Significance Test					Significance Test				
$\text{G2G}_{K=1}$	24.5(3.2)	0.972(0.004)	0.967(0.006)	0.231(0.016)	$\text{G2G}_{\text{NA},K=3}$	73.1(8.2)	0.989(0.001)	0.989(0.001)	0.475(0.017)
$\text{HEADNet}_{\text{NA}}$	29.3(3.2)	0.966(0.004)	0.968(0.004)	**0.352(0.014)**	$\text{HEADNet}_{\text{NA}}$	105.3(10.1)	0.984(0.002)	0.986(0.001)	**0.574(0.009)**
t-statistic	-5.82E+00	-5.82E+00	1.10E+00	3.10E+01	*t*-statistic	-1.36E+01	-1.36E+01	-8.87E+00	2.77E+01
p-value	1.00E+00	1.00E+00	1.39E-01	1.17E-37	*p*-value	1.00E+00	1.00E+00	1.00E+00	4.60E-30
$p < 0.05$	0	0	0	1	$p < 0.05$	0	0	0	1
HEADNet	**18.9(2.5)**	**0.979(0.003)**	**0.978(0.003)**	**0.395(0.020)**	HEADNet	**55.5(4.5)**	**0.992(0.001)**	**0.992(0.001)**	**0.625(0.006)**
t-statistic	7.54E+00	7.54E+00	8.82E+00	3.51E+01	*t*-statistic	1.03E+01	1.03E+01	9.15E+00	4.50E+01
p-value	2.63E-10	2.63E-10	1.00E-11	7.12E-40	*p*-value	8.82E-14	8.82E-14	1.51E-11	1.18E-33
$p < 0.05$	1	1	1	1	$p < 0.05$	1	1	1	1

Citeseer (536 edges removed)					Wiki Vote (10369 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	313.4(11.8)	0.417(0.022)	0.475(0.020)	0.022(0.003)	ATP (log)	5741.1(38.2)	0.446(0.004)	0.473(0.003)	0.013(0.001)
ATP (harmonic)	311.2(12.6)	0.421(0.023)	0.478(0.023)	0.021(0.003)	ATP (harmonic)	5741.9(38.5)	0.446(0.004)	0.473(0.003)	0.013(0.001)
LINE	315.3(7.6)	0.414(0.014)	0.422(0.007)	0.042(0.007)	LINE	6972.5(54.4)	0.328(0.005)	0.383(0.002)	0.045(0.003)
$\text{G2G}_{\text{NA},K=1}$	28.7(4.4)	0.948(0.008)	0.962(0.007)	0.268(0.014)	$\text{G2G}_{\text{NA},K=1}$	521.1(37.5)	0.950(0.004)	0.936(0.004)	0.101(0.005)
$\text{G2G}_{\text{NA},K=3}$	28.7(4.4)	0.948(0.008)	0.961(0.007)	0.251(0.013)	$\text{G2G}_{\text{NA},K=3}$	455.2(22.2)	0.956(0.002)	0.937(0.004)	0.112(0.003)
NK	44.6(5.3)	0.919(0.010)	0.938(0.007)	0.263(0.010)	$\text{G2G}_{K=1}$	–	–	–	–
$\text{G2G}_{K=1}$	15.3(2.6)	0.973(0.005)	0.971(0.008)	0.253(0.014)	$\text{G2G}_{K=3}$	–	–	–	–
Significance Test					Significance Test				
$\text{G2G}_{K=3}$	15.2(2.4)	0.974(0.005)	0.971(0.009)	0.240(0.014)	NK	208.9(8.3)	0.980(0.001)	0.978(0.001)	0.176(0.004)
$\text{HEADNet}_{\text{NA}}$	35.9(5.1)	0.935(0.009)	0.948(0.008)	**0.345(0.025)**	$\text{HEADNet}_{\text{NA}}$	**135.9(10.1)**	**0.987(0.001)**	**0.983(0.002)**	**0.210(0.004)**
t-statistic	-2.01E+01	-2.01E+01	-1.08E+01	2.01E+01	*t*-statistic	3.06E+01	3.06E+01	1.24E+01	3.50E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.66E-24	*p*-value	7.41E-37	7.41E-37	4.24E-17	5.46E-41
$p < 0.05$	0	0	0	1	$p < 0.05$	1	1	1	1
HEADNet	**13.7(2.9)**	**0.976(0.005)**	0.974(0.009)	**0.392(0.026)**	HEADNet	–	–	–	–
t-statistic	2.05E+00	2.05E+00	1.14E+00	2.78E+01	*t*-statistic	–	–	–	–
p-value	2.27E-02	2.27E-02	1.29E-01	8.24E-30	*p*-value	–	–	–	–
$p < 0.05$	1	1	0	1	$p < 0.05$	–	–	–	–

TABLE C.9: Summary of the link prediction task on synthetic, Cora_ML, Citeseer, Pubmed, Cora and Wiki Vote networks for an embedding dimension of 50+50. $\text{HEADNet}_{\text{NA}}$ denotes HEADNet without attributes. A dash (–) indicates that a network did not have attributes. All measures are reported to 3 decimal places. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

Synthetic (∼173 edges removed)					Pubmed (8865 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	93.9(10.0)	0.463(0.051)	0.546(0.047)	0.017(0.020)	ATP (log)	7692.6(24.3)	0.132(0.003)	0.327(0.001)	0.106(0.005)
ATP (harmonic)	94.9(8.8)	0.457(0.045)	0.540(0.042)	0.016(0.020)	ATP (harmonic)	7682.5(25.0)	0.134(0.003)	0.330(0.001)	0.107(0.005)
LINE	122.9(8.2)	0.296(0.037)	0.381(0.012)	0.015(0.008)	LINE	5317.1(86.9)	0.400(0.010)	0.411(0.004)	0.042(0.002)
$\text{G2G}_{\text{NA},K=1}$	47.0(7.4)	0.733(0.049)	0.648(0.054)	0.010(0.005)	$\text{G2G}_{\text{NA},K=1}$	63.0(6.2)	0.993(0.001)	0.992(0.001)	0.512(0.011)
$\text{G2G}_{\text{NA},K=3}$	37.2(7.1)	0.790(0.045)	0.733(0.053)	0.024(0.009)	NK	105.6(6.2)	0.988(0.001)	0.987(0.001)	0.479(0.005)
$\text{G2G}_{K=1}$	–	–	–	–	$\text{G2G}_{K=1}$	271.6(18.6)	0.969(0.002)	0.960(0.003)	0.152(0.007)
$\text{G2G}_{K=3}$	–	–	–	–	$\text{G2G}_{K=3}$	218.6(10.8)	0.975(0.001)	0.967(0.002)	0.174(0.004)
Significance Test					Significance Test				
NK	32.8(8.9)	0.814(0.057)	0.749(0.069)	0.033(0.009)	$\text{G2G}_{\text{NA},K=3}$	38.2(5.2)	0.996(0.001)	0.995(0.001)	0.624(0.007)
$\text{HEADNet}_{\text{NA}}$	**20.2(3.9)**	**0.889(0.024)**	**0.877(0.027)**	**0.053(0.011)**	$\text{HEADNet}_{\text{NA}}$	38.7(5.1)	0.996(0.001)	**0.996(0.001)**	**0.782(0.011)**
t-statistic	7.14E+00	6.54E+00	9.37E+00	7.69E+00	*t*-statistic	-3.70E-01	-3.70E-01	2.07E+00	6.53E+01
p-value	6.41E-09	4.60E-08	1.03E-11	1.31E-10	*p*-value	6.44E-01	6.44E-01	2.16E-02	1.32E-49
$p < 0.05$	1	1	1	1	$p < 0.05$	0	0	1	1
					HEADNet	69.7(5.5)	0.992(0.001)	0.990(0.001)	0.436(0.005)
t-statistic	–	–	–	–	*t*-statistic	-2.28E+01	-2.28E+01	-2.19E+01	-1.20E+02
p-value	–	–	–	–	*p*-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
$p < 0.05$	–	–	–	–	$p < 0.05$	0	0	0	0

Cora_ML (842 edges removed)					Cora (6532 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	495.9(12.3)	0.412(0.015)	0.451(0.010)	0.039(0.004)	ATP (log)	3925.1(34.5)	0.399(0.005)	0.437(0.004)	0.049(0.002)
ATP (harmonic)	491.5(12.5)	0.417(0.015)	0.459(0.011)	0.040(0.004)	ATP (harmonic)	3900.2(34.7)	0.403(0.005)	0.442(0.004)	0.050(0.002)
LINE	417.9(10.2)	0.505(0.012)	0.468(0.007)	0.059(0.008)	LINE	3374.6(43.1)	0.484(0.007)	0.456(0.003)	0.083(0.003)
$\text{G2G}_{\text{NA},K=1}$	26.1(3.6)	0.970(0.004)	0.966(0.005)	0.247(0.014)	$\text{G2G}_{\text{NA},K=1}$	71.6(6.2)	0.989(0.001)	0.989(0.001)	0.463(0.007)
$\text{G2G}_{\text{NA},K=3}$	26.4(3.9)	0.970(0.005)	0.966(0.007)	0.249(0.015)	NK	105.7(6.8)	0.984(0.001)	0.985(0.001)	0.482(0.005)
NK	29.2(3.4)	0.967(0.004)	0.965(0.005)	0.273(0.011)	$\text{G2G}_{K=1}$	65.3(4.7)	0.990(0.001)	0.989(0.001)	0.431(0.013)
$\text{G2G}_{K=3}$	24.0(3.1)	0.973(0.004)	0.968(0.006)	0.246(0.012)	$\text{G2G}_{K=3}$	63.6(4.9)	0.990(0.001)	0.990(0.001)	0.447(0.010)
Significance Test					Significance Test				
$\text{G2G}_{K=1}$	23.6(2.7)	0.973(0.003)	0.968(0.005)	0.239(0.014)	$\text{G2G}_{\text{NA},K=3}$	69.2(6.9)	0.990(0.001)	0.990(0.001)	0.491(0.011)
$\text{HEADNet}_{\text{NA}}$	23.0(3.2)	0.974(0.004)	**0.975(0.004)**	**0.399(0.017)**	$\text{HEADNet}_{\text{NA}}$	85.2(6.2)	0.987(0.001)	0.989(0.001)	**0.631(0.008)**
t-statistic	6.89E-01	6.89E-01	6.14E+00	4.05E+01	*t*-statistic	-9.45E+00	-9.45E+00	-3.20E+00	5.71E+01
p-value	2.47E-01	2.47E-01	5.14E-08	1.23E-43	*p*-value	1.00E+00	1.00E+00	9.99E-01	1.30E-50
$p < 0.05$	0	0	1	1	$p < 0.05$	0	0	0	1
HEADNet	**15.4(2.4)**	**0.983(0.003)**	**0.983(0.003)**	**0.428(0.023)**	HEADNet	**46.6(3.9)**	**0.993(0.001)**	**0.993(0.001)**	**0.664(0.007)**
t-statistic	1.23E+01	1.23E+01	1.26E+01	3.90E+01	*t*-statistic	1.57E+01	1.57E+01	1.27E+01	7.45E+01
p-value	5.87E-18	5.87E-18	1.27E-17	1.91E-38	*p*-value	3.02E-20	3.02E-20	1.99E-16	3.69E-53
$p < 0.05$	1	1	1	1	$p < 0.05$	1	1	1	1

Citeseer (536 edges removed)					Wiki Vote (10369 edges removed)				
	Mean Rank	AUROC	AP	mAP		Mean Rank	AUROC	AP	mAP
ATP (log)	298.2(12.2)	0.446(0.023)	0.501(0.021)	0.033(0.005)	ATP (log)	5769.3(34.9)	0.444(0.003)	0.471(0.003)	0.013(0.001)
ATP (harmonic)	294.5(11.8)	0.452(0.022)	0.505(0.020)	0.032(0.005)	ATP (harmonic)	5767.0(33.9)	0.444(0.003)	0.472(0.003)	0.013(0.001)
LINE	302.6(8.3)	0.437(0.015)	0.432(0.007)	0.039(0.007)	LINE	7276.5(45.1)	0.298(0.004)	0.373(0.001)	0.039(0.002)
$\text{G2G}_{\text{NA},K=1}$	28.7(4.5)	0.948(0.008)	0.962(0.007)	0.282(0.013)	$\text{G2G}_{\text{NA},K=1}$	472.3(27.0)	0.955(0.003)	0.942(0.003)	0.112(0.006)
$\text{G2G}_{\text{NA},K=3}$	28.3(4.3)	0.949(0.008)	0.962(0.007)	0.269(0.011)	$\text{G2G}_{\text{NA},K=3}$	474.1(21.1)	0.954(0.002)	0.933(0.004)	0.118(0.004)
NK	37.5(6.1)	0.932(0.011)	0.948(0.008)	0.266(0.013)	$\text{G2G}_{K=1}$	–	–	–	–
$\text{G2G}_{K=1}$	14.8(3.1)	0.974(0.006)	0.971(0.009)	0.259(0.014)	$\text{G2G}_{K=3}$	–	–	–	–
Significance Test					Significance Test				
$\text{G2G}_{K=3}$	14.8(2.6)	0.974(0.005)	0.972(0.009)	0.247(0.014)	NK	194.5(7.5)	0.981(0.001)	0.980(0.001)	0.180(0.004)
$\text{HEADNet}_{\text{NA}}$	33.7(4.6)	0.939(0.009)	0.953(0.007)	**0.370(0.029)**	$\text{HEADNet}_{\text{NA}}$	**117.5(9.2)**	**0.989(0.001)**	**0.985(0.001)**	**0.216(0.004)**
t-statistic	-1.95E+01	-1.95E+01	-8.98E+00	2.10E+01	*t*-statistic	3.55E+01	3.55E+01	1.58E+01	3.26E+01
p-value	1.00E+00	1.00E+00	1.00E+00	1.61E-24	*p*-value	2.30E-40	2.30E-40	6.61E-22	4.21E-39
$p < 0.05$	0	0	0	1	$p < 0.05$	1	1	1	1
HEADNet	**12.3(2.2)**	**0.979(0.004)**	**0.977(0.007)**	**0.424(0.017)**	HEADNet	–	–	–	–
t-statistic	4.05E+00	4.05E+00	2.38E+00	4.39E+01	*t*-statistic	–	–	–	–
p-value	7.85E-05	7.89E-05	1.05E-02	9.53E-46	*p*-value	–	–	–	–
$p < 0.05$	1	1	1	1	$p < 0.05$	–	–	–	–

TABLE C.10: Summary of link prediction on unseen nodes on embedding dimensions 25+25 and 50+50. All measures are reported to 3 decimal places with the exception of mean rank which is to 1 decimal place. Bold indicates best performance that is significant at a 0.05 level. For each network, for the computation of the *t*-statistic, we select the benchmark algorithm according to AP. The selected benchmark algorithm is identified with an asterisk (*). Significant results at a significance level of 0.05 are highlighted in bold. Standard deviation is given in brackets.

	Cora_ML (300 nodes removed)				Pubmed (1972 nodes removed)			
		Mean Rank	AUROC	AP		Mean Rank	AUROC	AP
25+25	*G2G$_{K=1}$	417.1(67.1)	0.746(0.028)	0.690(0.029)	*G2G$_{K=1}$	2771.0(231.7)	0.836(0.012)	0.811(0.013)
	G2G$_{K=3}$	644.3(85.2)	0.605(0.042)	0.556(0.036)	G2G$_{K=3}$	4033.5(421.9)	0.761(0.022)	0.742(0.021)
	HEADNet	**181.1(40.9)**	**0.890(0.018)**	**0.899(0.017)**	HEADNet	**1673.4(96.5)**	**0.901(0.004)**	**0.902(0.004)**
	t-statistic	16.44	23.63	33.77	*t*-statistic	24.97	29.98	39.99
	p-value	1.246E-21	6.78E-29	2.625E-34	*p*-value	1.102E-26	2.446E-29	5.557E-33
	$p < 0.05$	1	1	1	$p < 0.05$	1	1	1
	Citeseer (423 nodes removed)				Cora (1980 nodes removed)			
		Mean Rank	AUROC	AP		Mean Rank	AUROC	AP
	*G2G$_{K=1}$	229.8(25.2)	0.772(0.018)	0.789(0.019)	*G2G$_{K=1}$	2391.3(257.9)	0.807(0.018)	0.772(0.020)
	G2G$_{K=3}$	233.9(29.3)	0.768(0.023)	0.785(0.024)	G2G$_{K=3}$	3522.5(334.9)	0.716(0.023)	0.668(0.027)
	HEADNet	**154.5(21.0)**	**0.847(0.018)**	**0.864(0.016)**	HEADNet	**549.4(48.7)**	**0.956(0.003)**	**0.959(0.002)**
	t-statistic	12.59	15.93	16.48	*t*-statistic	38.44	45.21	50.29
	p-value	2.677E-18	3.798E-23	1.756E-23	*p*-value	4.621E-28	5.124E-30	1.091E-30
	$p < 0.05$	1	1	1	$p < 0.05$	1	1	1
	Cora_ML (300 nodes removed)				Pubmed (1972 nodes removed)			
		Mean Rank	AUROC	AP		Mean Rank	AUROC	AP
50+50	*G2G$_{K=1}$	424.4(71.2)	0.741(0.030)	0.684(0.032)	*G2G$_{K=1}$	2794.7(230.1)	0.835(0.012)	0.808(0.012)
	G2G$_{K=3}$	628.3(94.7)	0.615(0.048)	0.565(0.042)	G2G$_{K=3}$	4091.3(502.5)	0.758(0.026)	0.737(0.026)
	HEADNet	**170.3(39.6)**	**0.897(0.017)**	**0.904(0.017)**	HEADNet	**1659.1(126.7)**	**0.902(0.005)**	**0.903(0.004)**
	t-statistic	17.08	24.31	33.37	*t*-statistic	23.68	29.15	39.68
	p-value	1.075E-21	3.038E-28	7.244E-33	*p*-value	2.225E-27	2.771E-28	9.815E-32
	$p < 0.05$	1	1	1	$p < 0.05$	1	1	1
	Citeseer (423 nodes removed)				Cora (1980 nodes removed)			
		Mean Rank	AUROC	AP		Mean Rank	AUROC	AP
	*G2G$_{K=1}$	227.5(27.2)	0.774(0.023)	0.792(0.021)	*G2G$_{K=1}$	2353.5(237.2)	0.810(0.018)	0.776(0.022)
	G2G$_{K=3}$	233.1(27.4)	0.768(0.023)	0.786(0.022)	G2G$_{K=3}$	3710.1(243.9)	0.701(0.020)	0.652(0.023)
	HEADNet	**151.1(19.3)**	**0.850(0.016)**	**0.868(0.015)**	HEADNet	**524.0(58.1)**	**0.958(0.004)**	**0.961(0.003)**
	t-statistic	12.54	15.22	16.22	*t*-statistic	41.03	43.83	44.78
	p-value	1.112E-17	5.016E-21	2.502E-22	*p*-value	7.509E-30	3.362E-30	2.72E-29
	$p < 0.05$	1	1	1	$p < 0.05$	1	1	1

Appendix D

Supplementary for Chapter 5

D.1 Generation of Synthetic Bow-Tie Networks

D.1.1 Generate Topology

Algorithm 5 Construct synthetic bow tie networks with core positive feedback loop.

function CONSTRUCTBOWTIE(N_{in}, N_{core}, N_{out}, $|E_{\text{cp}}|_{\max}$, $N_{\text{plantedLoop}}$, coreLoopType)
  $\mathbb{G} \leftarrow$ DiGraph()
  inComponent $\leftarrow \{u_i \mid i \in [1, N_{\text{in}}]\}$
  core $\leftarrow \{c_i \mid i \in [1, N_{\text{core}}]\}$
  outComponent $\leftarrow \{o_i \mid i \in [1, N_{\text{out}}]\}$
  $\mathbb{G}$.addNodesFrom(inComponent $\cup$ core $\cup$ outComponent)
  plantedLoop $\leftarrow \{c_i \mid i \in [1, N_{\text{plantedLoop}}]\}$ ▷ plantedLoop $\subseteq$ core
  BUILDCORE($\mathbb{G}$, core, plantedLoop, coreLoopType) ▷ Build strongly connected core
  for $u \in$ inComponent **do** ▷ add edges from inComponent to core
    numEdges $\leftarrow$ RANDOMCHOICE([1, $|E_{\text{cp}}|_{\max}$])
    for $i \in$ [1,numEdges] **do**
      $c \leftarrow$ RANDOMCHOICE(core)
      $w \leftarrow$ RANDOMCHOICE({-1, 1})
      $\mathbb{G}$.addEdge(u, c, w)
    end for
  end for
  for $o \in$ outComponent **do** ▷ add edges from core to outComponent
    numEdges $\leftarrow$ RANDOMCHOICE([1, $|E_{\text{cp}}|_{\max}$])
    for $i \in$ [1,numEdges] **do**
      $c \leftarrow$ RANDOMCHOICE(core)
      $w \leftarrow$ RANDOMCHOICE({-1, 1})
      $\mathbb{G}$.addEdge(c, o, w)
    end for
  end for
  return $\mathbb{G}$
end function

Generation of bow-tie topology is described in algorithm 5. It requires five input arguments: N_{in}, N_{core}, N_{out}, $|E_{\text{cp}}|_{\max}$, $N_{\text{plantedLoop}}$, and coreLoopType. N_{in}, N_{core}, and N_{out} are the number of nodes in the in-, core and out-components respectively. Our networks contain $N_{\text{in}} = 3$ nodes in the in-component, $N_{\text{out}} = 10$ nodes in the out-component, and core size $N_{\text{core}} \in \{6, 10, 15\}$. $|E_{\text{cp}}|_{\max}$ is the maximum number of connections connecting periphery nodes to core nodes. That is the maximum number of connections from nodes in the in component to the core and from nodes in the core to the nodes in the out component. $N_{\text{plantedLoop}}$ is the size of the planted positive feedback loop. We fix $N_{\text{plantedLoop}} = 3$ for all synthetic networks. coreLoopType is a flag to set the type of the core feedback loop to be positive or negative.

Algorithm 6 Add cycle to directed graph $\mathbb{G}$.

```
1: function AddCycle(𝔾, cycle, coreLoopType)
2:     cycleLength ← Length(cycle)
3:     for i ∈ [1, cycleLength] do
4:         if coreLoopType = pos then
5:             w ← 1
6:         else
7:             w ← −1
8:         end if
9:         𝔾.addEdge(cycle[i], cycle[i + 1% cycleLength], w)        ▷ % is the modulo operator
10:    end for
11: end function
```

Algorithm 7 Add chain to directed graph $\mathbb{G}$.

```
1: function AddChain(𝔾, chain)
2:     chainLength ← Length(chain)
3:     for i ∈ [1, chainLength − 1] do
4:         w ← RandomChoice({-1, 1})
5:         𝔾.addEdge(chain[i], chain[i + 1], w)
6:     end for
7: end function
```

The core of the bow-tie networks is build using algorithm 8 and is based upon adding cycles (see algorithm 6). The core is initialized as $N_{\text{plantedLoop}}$ nodes (the planted loop) forming a feedback loop. The sign of the feedback loop is determined using the flag coreLoopType. Then, for each remaining node $c \in$ core $\setminus$ plantedLoop, a chain $[s, c, s]$ is built beginning from $s \in$ plantedLoop and ending with $e \in$ plantedLoop (see algorithm 7). Then c is added to the plantedLoop set.

Algorithm 8 Build a strongly connected core around a positive feedback loop in $\mathbb{G}$.

```
1: function BUILDCORE(𝔾, core, plantedLoop, coreLoopType)
2:     ADDCYCLE(𝔾, plantedLoop, coreLoopType) ▷ create core positive feedback loop
3:     core ← core \ plantedLoop
4:     while |core| > 0 do                    ▷ Build up core from planted loop
5:         s ← RANDOMCHOICE(plantedLoop)
6:         c ← RANDOMCHOICE(core)
7:         e ←RANDOMCHOICE(plantedLoop)
8:         newChain ← [s, c, e]
9:         ADDCHAIN(𝔾, newChain)               ▷ Allow negative relations
10:        core ← core \ newChain
11:    end while                              ▷ core is guaranteed to be strongly connected
12: end function
```

Theorem 1. *Algorithm 8 produces a strongly connected core of size* $N_{core} \geq N_{plantedLoop}$.

Proof. Proof by induction on N_{core}.

Base case ($N_{\text{core}} = N_{\text{plantedLoop}}$): since the core is initialized as a positive cycle of length $N_{\text{plantedLoop}}$, which is trivially strongly connected, then the entire core is strongly connected.

Inductive hypothesis: the theorem holds for core of size $N_{\text{core}} \leq n$.

Inductive step ($N_{\text{core}} = n + 1$): A new chain containing a node c that is not already in in the core of the core is added to $\mathbb{G}$ that begins and ends with nodes $s, e \in$ plantedLoop, producing a core size of $n + 1$. To prove this core is strongly connected, we must show that the new node c can reach every node in plantedLoop. This chain forms a path from s to e. Since $s, e \in$ plantedLoop, and $|\text{plantedLoop}| = n$ then, by the inductive hypothesis, plantedLoop is strongly connected and a path exists from e to s. This forms a cycle C=[s, c, e, ..., s] and every node in C can reach every other node in C. In particular, c can reach every node in C. Furthermore, since $e \in$ plantedLoop, then e can reach every node $n \in \text{plantedLoop} \setminus C$. Now c can reach e and so c can reach all nodes in plantedLoop $\setminus$ C. Therefore c can reach all nodes in $C \cup \text{plantedLoop} \setminus C = \text{plantedLoop}$ as required. □

Theorem 1 shows that algorithm 8 produces a core that is strongly connected as required. To build the bow-tie structure around the constructed core, we sparsely connect nodes in the in-component to the core and nodes in the core to nodes in the

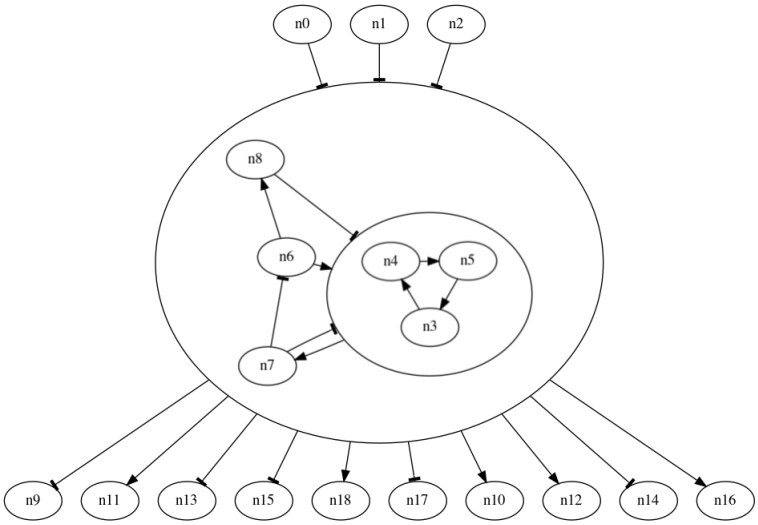


FIGURE D.1: An example synthetic bow tie network generated by algorithm 5 with $N_{\text{core}} = 6$. Nodes n0 to n2 form the in-component. Nodes n3, n4, and n5 form a (positive) coherent core feedback loop. Nodes n6, n7, and n8 comprise the remainder of the core. Nodes n3 to n8 form the largest strongly connected component in the network – the core of the bow-tie architecture. Nodes n9 to n18 form the out-component.

out-component. Each node in the in- and out-components can have at most $|E_{\text{cp}}|_{\max}$ edges incident to it.

Figure D.1 plots an example synthetic bow-tie network topology generated using algorithm 5 with $N_{\text{core}} = 6$.

D.1.2 Generate Boolean Rules from Topology

After determining a network topology, dynamics must be determined, respecting the sign of edges. This is achieved using algorithm 9. Our approach is simple, however generates dynamical behaviour operating at criticality, with an average sensitivity close to 1 (see table 5.5), and derrida curves mirroring the curves produced by real-world networks (see figure D.3) – a dynamical property of real world dynamical systems (see section 1.5.9) – suggesting that our approach is valid[1].

To build the boolean function f_u for node u, we use the set of input nodes V_u^{in} that have an edge to u as input to f_u. If $|V_u^{\text{in}}| = 0$ (that is, node u has no input nodes) then u is an input node and we determine that input nodes should keep their value. Otherwise, the rule is initialised to be empty, The set of input nodes

[1]Recall that "biological regulatory networks are more distinguished from random networks by their criticality than by other macro-scale network properties such as degree distribution, edge density, or fraction of activating conditions" (Daniels et al., 2018). It follows from this that the generated synthetic bow tie models are reasonable approximations of real-life biological networks.

Algorithm 9 Generate boolean rules from a given signed network $\mathbb{G}$.

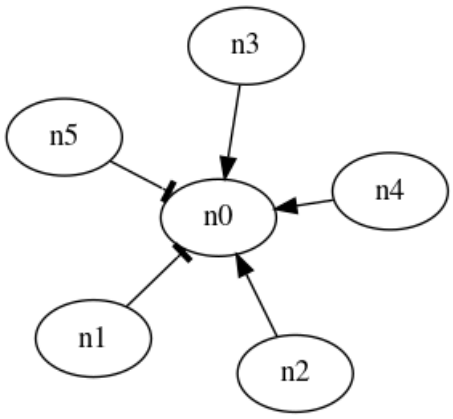
```
function CreateBooleanRules(𝔾)
    rules ← []
    for u ∈ 𝔾 do
        inEdges ← 𝔾.inEdges(u)
        if Length(inEdges) == 0 then
            f_u ← u                          ▷ nodes in in-component keep their value
        else                                 ▷ build rule from incoming edges
            f_u ← ϵ                          ▷ initialize empty string
            inEdges ← Shuffle(inEdges)       ▷ Randomize order of inputs
            for v, w ∈ inEdges do
                if w < 0 then                ▷ inhibition
                    v ← ¬v
                end if
                connective ← RandomChoice({∧, ∨})
                f_u ← f_u + connective + v
            end for
        end if
        f_u ← AddBrackets(f_u)
        rules[u] ← f_u
    end for
    return rules
end function
```

$v \in V_u^{\text{in}}$ is shuffled and added to the rule one at a time. If the rule is non-empty, then a randomly selected boolean connective (either AND ($\wedge$) or OR ($\vee$) is added to the rule, before an input node literal is added. If the sign of the edge (v, u) is positive, the literal for node v is v, otherwise it is the negation of v: $\neg v$. Once all input nodes have been processed, we surround the OR terms in f_u with brackets using the AddBrackets$(\cdot)$ function to put f_u into CNF (see section 1.5.5). Figure D.2 describes the process of transforming topology into boolean rules.

D.2 Supplementary Information on Datasets

D.2.1 Derrida Curves

Figure D.3 plots the derrida curves for the three case study networks and the three synthetic bow tie architectures. Comparison between the curves of the synthetic and real world networks show that our synthetic networks display critical behaviour that closely mirrors the expected behaviour of real world networks.



(A) The node n0 and all of the nodes n1, n2, n3, n4, and n5 that connect to it. The nodes n2, n3, and n4 are assigned an activatory effect on n0, denoted with an arrowhead. The nodes n1 and n5 are assigned an inhibitory effect on n0, denoted with the line arrowhead.

$$f_{n0} := \neg n5 \wedge n2 \vee \neg n1 \wedge n3 \vee n4$$

(B) The topology in (A) is transformed into boolean rules. Input nodes with an inhibitory relationship are negated when they added to the rule. Each input node after the first (n5) is attached to the rule using a randomly selected boolean connective.

$$f_{n0} := \neg n5 \wedge (n2 \vee \neg n1) \wedge (n3 \vee n4)$$

(C) Input nodes connected by OR are surrounded by brackets to transform the rule into CNF (see section 1.5.5).

FIGURE D.2: The procedure for transforming synthetic signed network topology into rules. (A) shows the subnetwork induced by the node n0 and all of the nodes that have a directed edge to it. (B) shows a randomly generated boolean rule from the topology described in (A). The order of the input nodes is shuffled and each input node after the first is connected by a randomly selected boolean connective. (C) shows how the rule in (B) is transformed into conjunctive normal form (CNF). Note that the rules described in (B) and (C) are not equivalent.

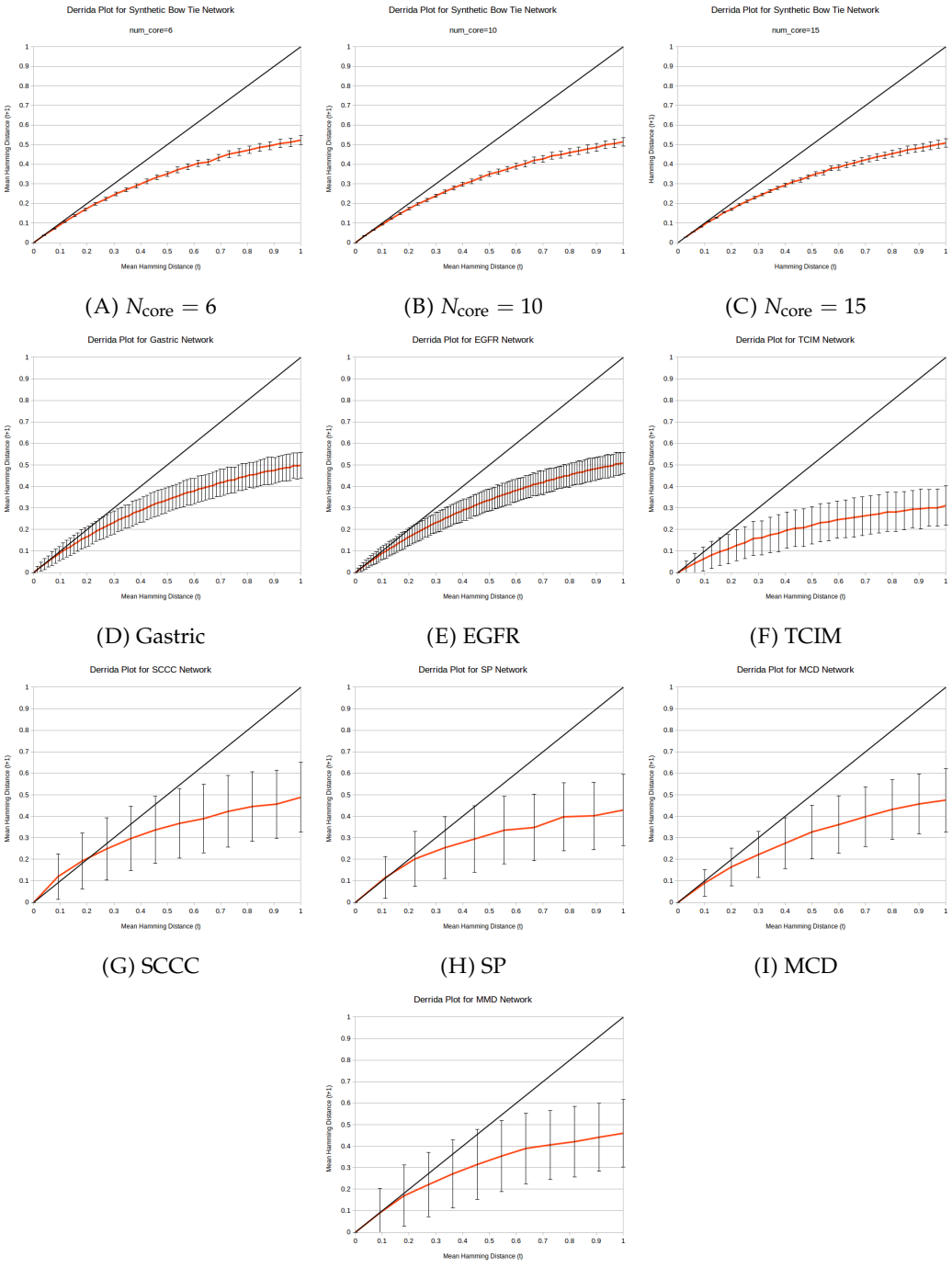
D.3 Additional Results: Synthetic Bow-Tie Networks

Figure D.4 shows the box plots of the significance of turning off members of planted coherent feedback loops for synthetic bow tie networks with $N_{\text{core}} = 10$ (figure D.4A) and $N_{\text{core}} = 15$ (figure D.4B) nodes in the core.

D.4 Additional Results: Cancerous Case Study Networks

D.4.1 AGS Gastric Cancer Cell Line

Table D.1 shows full AUROC and AP scores for each output node for the Gastric cancer network. In table D.2 we show the ranking of IMPLISig compared with a



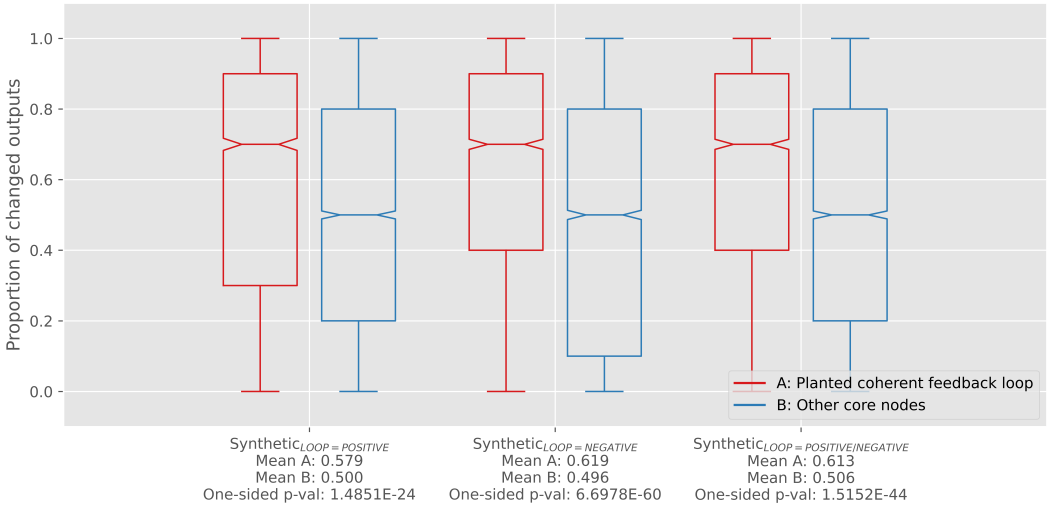
(A) $N_{\text{core}} = 6$ (B) $N_{\text{core}} = 10$ (C) $N_{\text{core}} = 15$

(D) Gastric (E) EGFR (F) TCIM

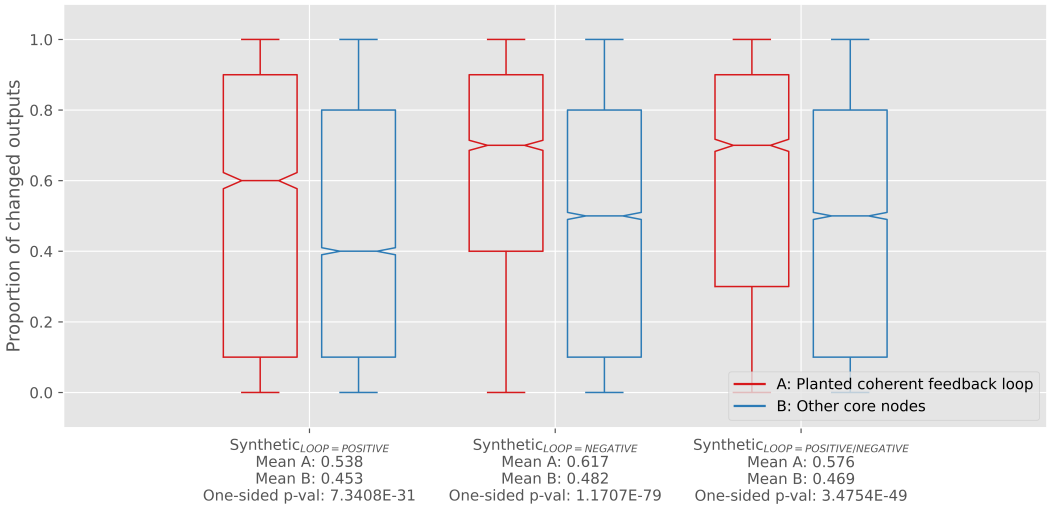
(G) SCCC (H) SP (I) MCD

(J) MMD

FIGURE D.3: Derrida curves for the synthetic ((A) (B) and (C)) and the case study networks ((D) (E) and (F)), and the GRNs ((G), (H), (I), and (J).The x-axes plot mean hamming distance at time t ($mD_H(t)$), and the y-axes plot mean hamming distance at time $t+1$ ($mD_H(t+1)$). The black lines denote criticality. The red lines plot $mD_H(t+1)$ against $mD_H(t)$ for each network. The vertical black lines are the standard deviations over 1000 sampled state pairs. We see for small perturbations at time t, the networks behave close to criticality as expected (Daniels et al., 2018). This follows from the close proximity of the red line to the black for small values of $mD_H(t)$. This property also follows from average sensitivity $\langle s \rangle \approx 1$.



(A) $N_{\text{core}} = 10$



(B) $N_{\text{core}} = 15$

FIGURE D.4: Additional synthetic bow-tie network results.

number of micro-scopic node features for AUROC and AP for each output, as well as the mean rank across all outputs For AUROC, we see that IMPLISig ranks first compared to the benchmarks for all but one output as well as ranking first for all but two outputs with respect to AP. For both metrics, considering all output nodes simultaneously, IMPLISig achieved the best performance.

D.4.2 EGFR-Erbb1 Signalling Network

Table D.3 shows full AUROC and AP scores for each output node for the EGFR-Erbb1 signalling network. Table D.4 shows the rank of IMPLISig with respect to AUROC and AP against a number of micro-scopic node features. For AUROC, we

TABLE D.1: AUROC and AP scores for Gastric cancer network. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient and cn: core number. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of significance.

AUROC	Pro-Cancer			Anti-Cancer			
	RSK	TCF	cMYC	FOXO	Caspase8	Caspase9	Mean AUROC
k	0.662	0.480	0.480	0.480	0.701	0.504	0.551
k_{in}	0.653	0.507	0.507	0.507	0.649	0.585	0.568
k_{out}	0.568	0.505	0.505	0.505	0.635	0.462	0.530
bc	0.759	0.664	0.664	0.664	**0.740**	0.347	0.640
cc	0.511	0.348	0.348	0.348	0.623	0.356	0.422
cn	0.695	0.623	0.623	0.623	0.722	0.720	0.668
IMPLISig	**0.763**	**0.896**	**0.896**	**0.896**	0.726	**0.805**	**0.831**
AP	Pro-Cancer			Anti-Cancer			
	RSK	TCF	cMYC	FOXO	Caspase8	Caspase9	Mean AP
k	0.309	0.095	0.095	0.095	0.418	0.039	0.175
k_{in}	0.223	0.105	0.105	0.105	0.291	0.056	0.148
k_{out}	0.266	0.093	0.093	0.093	0.381	0.033	0.160
bc	**0.419**	0.168	0.168	0.168	**0.507**	0.036	0.244
cc	0.177	0.082	0.082	0.082	0.351	0.033	0.134
cn	0.248	0.108	0.108	0.108	0.333	0.057	0.160
IMPLISig	0.376	**0.306**	**0.306**	**0.306**	0.422	**0.143**	**0.310**

see that IMPLISig ranks first for predicting significant targets for three output nodes. For AP, this increases to four. With respect to mean performance over all outputs, IMPLISig ranks first by both measures.

D.4.3 Tumour Cell Invasion and Migration

Table D.5 shows full AUROC and AP scores for each output node for the TCIM network. In table D.6, we show how IMPLISig ranks compared to other micro-scopic node features on the TCIM network. IMPLISig scores well for nodes associated with cell death, but poorly for inhibiting expression of Metastasis. According to both AUROC and AP, we find that IMPLISig ranks first for the output node CellCycleArrest. Furthermore, IMPLISig out-ranks all benchmarks averaging over all output nodes of interest.

TABLE D.2: Gastric network AUROC and AP ranks. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient and cn: core number. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of significance. Full AUROC and AP scores are given in table D.1 of the supplementary materials.

AUROC	Pro-Cancer			Anti-Cancer			
	RSK	TCF	cMYC	FOXO	Caspase8	Caspase9	Mean Rank
k	4	6	6	6	4	4	5.0
k_{in}	5	4	4	4	5	3	4.2
k_{out}	6	5	5	5	6	5	5.3
bc	2	2	2	2	**1**	7	2.7
cc	7	7	7	7	7	6	6.8
cn	3	3	3	3	3	2	2.8
IMPLISig	**1**	**1**	**1**	**1**	2	**1**	**1.2**
AP	Pro-Cancer			Anti-Cancer			
	RSK	TCF	cMYC	FOXO	Caspase8	Caspase9	Mean Rank
k	3	5	5	5	3	4	4.2
k_{in}	6	4	4	4	7	3	4.7
k_{out}	4	6	6	6	4	6	5.3
bc	**1**	2	2	2	**1**	5	2.2
cc	7	7	7	7	5	7	6.7
cn	5	3	3	3	6	2	3.7
IMPLISig	2	**1**	**1**	**1**	2	**1**	**1.3**

D.4.4 Core Feedback Loops

The identified core coherent feedback loops for the Gastric Cell Line, EGFR and TCIM networks are {AKT, IKKA, betacatenin, LEF, MMP, RTPK, SCH1, GRB2, GAB, PI3K, PDK1}, {pi3k, pip3, gab1}, and {SNAI1, TWIST1} respectively.

AGS Gastric Cancer Cell Line

Table D.2 shows the top 10 nodes in the largest strongly connected component in the Gastric cancer network, ranked by p-values on from a paired t-test. Bold rows indicate nodes belonging to the identified core positive feedback loop. Of the 11 nodes in the core positive feedback loop, 5 rank in the top 10. Furthermore, betacatenin is tied for first by mean rank. We note that both Ras and SOS are "one-hop" away from the identified core positive feedback loop and are merged at iteration 2 by IMPLISig.

TABLE D.3: AUROC and AP scores for EGFR-Erbb1 network. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient, and cn: core number. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of significance.

AUROC	Pro-Cancer						Anti-Cancer	
	hsp27	ap1	cmyc	p70s6_2	elk1	creb	pro_apoptotic	Mean AUROC
k	0.676	0.637	0.600	0.683	0.579	0.600	0.690	0.638
k_{in}	0.642	0.558	0.498	0.603	0.513	0.498	0.648	0.566
k_{out}	0.679	0.698	0.674	**0.745**	0.628	0.674	0.698	0.685
bc	0.710	0.676	**0.769**	0.670	**0.814**	**0.769**	0.692	**0.728**
cc	0.441	0.401	0.441	0.368	0.415	0.441	0.401	0.415
cn	0.500	0.475	0.419	0.455	0.434	0.419	0.475	0.454
IMPLISig	**0.750**	**0.714**	0.600	0.688	0.607	0.600	**0.714**	0.668
AP	Pro-Cancer						Anti-Cancer	
	hsp27	ap1	cmyc	p70s6_2	elk1	creb	pro_apoptotic	Mean AP
k	0.260	0.275	0.503	0.332	0.457	0.503	0.314	0.378
k_{in}	0.276	0.267	0.447	0.309	0.429	0.447	0.332	0.358
k_{out}	0.284	0.321	0.599	0.423	0.529	0.599	0.321	0.439
bc	0.311	0.319	**0.785**	0.342	**0.803**	**0.785**	0.327	0.525
cc	0.173	0.192	0.428	0.213	0.392	0.428	0.192	0.288
cn	0.195	0.215	0.421	0.237	0.395	0.421	0.215	0.300
IMPLISig	**0.591**	**0.550**	0.564	**0.527**	0.548	0.564	**0.550**	**0.556**

EGFR-Erbb1 Signalling Network

In table D.4, we see the top 15 nodes in the core ranked by p-value of significance. Each of the nodes in the identified core positive feedback loop, pi3k, pip3, and gab1 rank amongst them. pi3k, in particular, is ranked first for 5 out of 7 outputs of interest, including being 1 of only 3 targets that rank first for affecting the cell death associated pro_apoptotic node. We note that vav2, grb2, ras, pi34p2, and shp2 are one hop away from the the nodes in the core-positive feedback loop. This suggests that proximity to the core positive feedback loop in the original network may serve as an alternative measure of core node significance.

Tumour Cell Invasion and Migration

Table D.9 shows the top 10 nodes in the core ranked by mean rank of the significance of expression change for each of the three outputs of interest. It shows that, of the three feedback loops identified by IMPLISig, {SNAI1, TWIST1} is the most significant, with them both ranking in the top three nodes in the core. Both nodes rank

TABLE D.4: EGFR network AUROC and AP ranks. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient and cn: core number,. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of significance. Full AUROC and AP scores are given in table D.3 of the supplementary materials.

AUROC	Pro-Cancer						Anti-Cancer	
	hsp27	ap1	cmyc	p70s6_2	elk1	creb	pro_apoptotic	Mean Rank
k	4	4	3	3	4	3	4	3.6
k_{in}	5	5	5	5	5	5	5	5.0
k_{out}	3	2	2	**1**	2	2	2	**2.0**
bc	2	3	**1**	4	**1**	**1**	3	2.1
cc	7	7	6	7	7	6	7	6.7
cn	6	6	7	6	6	7	6	6.3
IMPLISig	**1**	**1**	3	2	3	3	**1**	**2.0**
AP	Pro-Cancer						Anti-Cancer	
	hsp27	ap1	cmyc	p70s6_2	elk1	creb	pro_apoptotic	Mean Rank
k	5	4	4	4	4	4	5	4.3
k_{in}	4	5	5	5	5	5	2	4.4
k_{out}	3	2	2	2	3	2	4	2.6
bc	2	3	**1**	3	**1**	**1**	3	2.0
cc	7	7	6	7	7	6	7	6.7
cn	6	6	7	6	6	7	6	6.3
IMPLISig	**1**	**1**	3	**1**	2	3	**1**	**1.7**

joint first for the significance of the increase of expression of Apoptosis, and TWIST1 also ranks joint first for Metastasis.

D.4.5 P-values

Gastric Cancer Cell Line

Table D.10 shows the p-values of the significance of the changes caused by switching off each node in the core to each output TF in the Gastric cancer cell dataset.

EGFR-Erbb11 Signalling Network

Table D.11 shows the p-values of the significance of the changes caused by switching off each node in the core to each output TF in the EGFR-Erbb1 signalling network dataset.

TABLE D.5: AUROC and AP scores for TCIM network. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient and cn: core number. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of significance.

AUROC	Pro-Cancer	Anti-Cancer		
	Metastasis	Apoptosis	CellCycleArrest	Mean AUROC
k	0.585	0.308	0.570	0.488
k_{in}	0.515	0.318	0.585	0.473
k_{out}	**0.640**	0.364	0.575	0.526
bc	0.630	0.313	0.530	0.491
cc	0.390	0.596	0.680	0.555
cn	0.510	0.394	0.510	0.471
IMPLISig	0.540	**0.667**	**0.790**	**0.666**
AP	Pro-Cancer	Anti-Cancer		
	Metastasis	Apoptosis	CellCycleArrest	Mean AP
k	0.586	0.362	0.593	0.513
k_{in}	0.521	0.370	0.583	0.491
k_{out}	0.612	0.389	0.541	0.514
bc	**0.706**	0.382	0.536	0.541
cc	0.461	**0.638**	0.684	0.595
cn	0.505	0.408	0.505	0.473
IMPLISig	0.534	0.553	**0.811**	**0.633**

TCIM Network

Table D.12 shows the p-values of the significance of the changes caused by switching off each node in the core with respect to each output TF in the TCIM network.

D.5 Additional Results: Control Kernel Recovery

Table D.13 evaluates the likelihood of nodes in core coherent feedback loops, identified by IMPLISig to belong to a control kernel. We find the core coherent feedback loops to be a highly precise measure – achieving a perfect precision score of 1.000 in two out of the five networks. In all cases, the core feedback loop contained at least one node belonging to a control kernel. Both nodes identified for the MMC network (Fgf8_g and Fgf8_p) are control kernels in their own right – targeting just one of them is sufficient to control the entire network. This is also true for AP1 identified in the ATD network. Finally, we note that the node Fli_1, identified by IMPLISig, is in

TABLE D.6: TCIM network AUROC and AP ranks. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient and cn: core number. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of significance. Full AUROC and AP scores are given in table D.5 of the supplementary materials.

AUROC	Pro-Cancer	Anti-Cancer		
	Metastasis	Apoptosis	CellCycleArrest	Mean Rank
k	3	7	5	5.0
k_{in}	5	5	3	4.3
k_{out}	**1**	4	4	3.0
bc	2	6	6	4.7
cc	7	2	2	3.7
cn	6	3	7	5.3
IMPLISig	4	**1**	**1**	**2.0**
AP	Pro-Cancer	Anti-Cancer		
	Metastasis	Apoptosis	CellCycleArrest	Mean Rank
k	3	7	3	4.3
k_{in}	5	6	4	5.0
k_{out}	2	4	5	3.7
bc	**1**	5	6	4.0
cc	7	**1**	2	3.3
cn	6	3	7	5.3
IMPLISig	4	2	**1**	**2.3**

both of the control kernels for the MMD network, identified by Kim, Park, and Cho, 2013 (see table 5.4).

Table D.14 shows AUROC and AP ranks for each output node for the control kernel recovery task.

TABLE D.7: Top 10 nodes in core of Gastric cancer network ranked by significance. Rows highlighted in bold correspond to nodes in the identified core positive feedback loop.

	Pro-Cancer			Anti-Cancer			
	RSK	TCF	cMYC	FOXO	Caspase8	Caspase9	Mean Rank
Ras	1	2	2	1	1	1	1.3
betacatenin	**1**	**1**	**1**	**1**	**1**	**3**	**1.3**
SOS	1	3	3	5	1	1	2.3
GRB2	**1**	**4**	**4**	**1**	**1**	**3**	**2.3**
SHC1	**1**	**5**	**5**	**1**	**1**	**3**	**2.7**
PI3K	**1**	**6**	**6**	**6**	**1**	**3**	**3.8**
Raf	1	6	6	6	1	3	3.8
ERK	1	6	6	6	1	3	3.8
PDK1	**1**	**6**	**6**	**6**	**1**	**3**	**3.8**
MEK	1	6	6	6	1	3	3.8

TABLE D.8: Top 15 nodes in core of EGFR network ranked by significance of change to each output. Rows highlighted in bold correspond to nodes in the identified core positive feedback loop.

	Pro-Cancer						Anti-Cancer	
	hsp27	ap1	cmyc	p70s6_2	elk1	creb	pro_apoptotic	Mean Rank
rac_cdc42	1	1	5	1	3	1	1	1.9
pi3k	**1**	**1**	**4**	**1**	**4**	**1**	**1**	**1.9**
vav2	1	4	7	5	5	1	1	3.4
mek12	7	8	1	9	1	7	8	5.9
erk12	7	8	1	9	1	7	8	5.9
mekk1	7	1	6	1	11	15	8	7.0
shp2	7	8	8	9	6	7	8	7.6
sos1	7	8	8	9	6	7	8	7.6
grb2	7	8	8	9	6	7	8	7.6
raf1	7	8	8	9	6	7	8	7.6
ras	7	8	8	9	6	7	8	7.6
pi34p2	4	6	15	8	12	4	5	7.7
pip3	**5**	**5**	**14**	**7**	**13**	**5**	**6**	**7.9**
gab1	**6**	**7**	**13**	**6**	**14**	**6**	**7**	**8.4**
p90rsk	7	8	3	9	19	7	8	8.7

TABLE D.9: Top 10 nodes in core of TCIM network ranked by significance. Rows highlighted in bold correspond to nodes in the identified core positive feedback loop.

	Pro-Cancer	Anti-Cancer		
	Metastasis	Apoptosis	CellCycleArrest	Mean Rank
CDH2	1	1	6	2.7
SNAI1	**8**	**1**	**3**	**4**
TWIST1	**1**	**1**	**13**	**5**
AKT1	9	9	1	6.3
TGFbeta	1	7	12	6.7
SNAI2	1	5	16	7.3
p73	11	6	5	7.3
AKT2	1	8	14	7.7
GF	17	1	9	9
ZEB2	1	13	16	10

TABLE D.10: P-values from one-tailed paired t-test of change in expression for the Gastric cancer dataset Flobak et al., 2015. Targets are given by rows and output nodes are given by columns. Targets identified by IMPLISig are highlighted in bold. Changes significant at a 0.05 level are highlighted in bold.

	Caspase9	TCF	cMYC	FOXO	Caspase8	RSK
DUSP1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	1.00E+00
SOS	**0.00E+00**	**7.61E-43**	**7.61E-43**	**7.61E-43**	**0.00E+00**	**0.00E+00**
MSK	1.00E+00	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	1.00E+00
MMP	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
DUSP6	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
LEF	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
MLK3	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
PDK1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	**0.00E+00**
betaTrCP	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
SFRP1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
GSK3	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
MKK3	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
DKK1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
MEK	1.00E+00	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	**0.00E+00**
GAB	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
MKK7	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
GRB2	1.00E+00	**2.84E-27**	**2.84E-27**	**0.00E+00**	**0.00E+00**	**0.00E+00**
GRAP2	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
betacatenin	1.00E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Rheb	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
TSC2	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
MDM2gene	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
S6K	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
RTPK	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
TAB1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
RTPKgene	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
IKKB	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
MDM2	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
IKKA	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
TAK1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
LRP	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
ERK	1.00E+00	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	**0.00E+00**
Dvl	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
Raf	1.00E+00	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	**0.00E+00**
IRS1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
NLK	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
NFkB	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
PTENgene	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
Rac	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
Fz	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
p38alpha	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
SHP2	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
mTORC1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
p53	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
mTORC2	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
DKK1gene	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
MKK4	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
SFRP1gene	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
AKT	1.00E+00	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	1.00E+00
pras40	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
MEKK4	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
PTEN	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
JNK1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
Egr1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
Axin	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
CK1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
Ras	**0.00E+00**	**2.94E-95**	**2.94E-95**	**0.00E+00**	**0.00E+00**	**0.00E+00**
MAP3K8	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
ASK1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
SHC1	1.00E+00	**2.75E-23**	**2.75E-23**	**0.00E+00**	**0.00E+00**	**0.00E+00**
PI3K	1.00E+00	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	**0.00E+00**

TABLE D.11: P-values from one-tailed paired t-test of change in expression over attractor states for seven output TFs of interest for the EGFR-Erbb1 signalling network. Targets are given by rows and output nodes are given by columns. Targets identified by IMPLISig are highlighted in bold. Changes significant at a 0.05 level are highlighted in bold.

	hsp27	ap1	cmyc	pro_apoptotic	elk1	creb	p70s6_2
p90rskerk12d	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
rasgap	1.00E+00	1.00E+00	1.89E-01	1.00E+00	3.76E-01	7.91E-01	1.00E+00
ccbl	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
rntre	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
erbb13	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
p90rsk	1.00E+00	1.00E+00	**1.32E-184**	1.00E+00	1.00E+00	**1.17E-211**	1.00E+00
erbb12	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
shc	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
erk12	1.00E+00	1.00E+00	**5.09E-217**	1.00E+00	**1.30E-243**	**1.17E-211**	1.00E+00
raf1	1.00E+00	1.00E+00	**1.39E-106**	1.00E+00	**1.76E-101**	**1.17E-211**	1.00E+00
endocyt_degrad	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
grb2	1.00E+00	1.00E+00	**1.39E-106**	1.00E+00	**1.76E-101**	**1.17E-211**	1.00E+00
pi3k	**0.00E+00**	**2.27E-290**	**2.02E-131**	**0.00E+00**	**7.75E-114**	**0.00E+00**	**2.82E-68**
gab1	**1.89E-258**	**1.85E-129**	**2.88E-78**	**8.79E-250**	**3.59E-55**	**1.43E-247**	**2.69E-44**
rin1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
shp1	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
aktd	1.00E+00	1.00E+00	1.28E-01	1.00E+00	5.00E-01	9.46E-01	1.00E+00
rab5a	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
ras	1.00E+00	1.00E+00	**1.39E-106**	1.00E+00	**1.76E-101**	**1.17E-211**	1.00E+00
pip3	**2.22E-289**	**2.38E-143**	**2.78E-66**	**2.22E-289**	**1.74E-56**	**1.02E-275**	**4.98E-37**
vav2	**0.00E+00**	**1.50E-250**	**1.22E-113**	**0.00E+00**	**5.93E-102**	**0.00E+00**	**3.42E-58**
rac_cdc42	**0.00E+00**	**2.27E-290**	**2.69E-130**	**0.00E+00**	**3.23E-116**	**0.00E+00**	**2.82E-68**
nck	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
sos1_eps8_e3b1	1.00E+00	1.00E+00	5.00E-01	1.00E+00	5.00E-01	5.00E-01	1.00E+00
pi34p2	**2.27E-290**	**7.15E-140**	**7.25E-66**	**2.27E-290**	**7.57E-58**	**1.21E-276**	**8.17E-33**
mekk1	1.00E+00	**2.27E-290**	**3.20E-123**	1.00E+00	**1.11E-96**	**5.09E-39**	**2.82E-68**
akt	1.00E+00	1.00E+00	2.82E-01	1.00E+00	1.59E-01	9.10E-01	**2.82E-68**
mek12	1.00E+00	1.00E+00	**5.09E-217**	1.00E+00	**1.30E-243**	**1.17E-211**	1.00E+00
pak1	1.00E+00	1.00E+00	1.00E+00	**0.00E+00**	1.00E+00	1.00E+00	1.00E+00
shp1d	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
erbb14	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
sos1	1.00E+00	1.00E+00	**1.39E-106**	1.00E+00	**1.76E-101**	**1.17E-211**	1.00E+00
shp2	1.00E+00	1.00E+00	**1.39E-106**	1.00E+00	**1.76E-101**	**1.17E-211**	1.00E+00

TABLE D.12: P-values from one-tailed paired t-test of change in expression for the TCIM network Cohen et al., 2015. Targets are given by rows and output nodes are given by columns. Targets identified by IMPLISig are highlighted in bold. Changes significant at a 0.05 level are highlighted in bold.

	Apoptosis	CellCycleArrest	Metastasis
GF	**0.00E+00**	**5.35E-05**	1.00E+00
ZEB2	9.87E-01	1.00E+00	**0.00E+00**
CDH1	1.00E+00	**4.64E-96**	1.00E+00
p73	**7.90E-17**	**9.61E-54**	8.15E-01
AKT2	**4.73E-06**	1.00E+00	**0.00E+00**
p53	1.00E+00	**0.00E+00**	1.00E+00
miR203	1.00E+00	1.00E+00	1.00E+00
TWIST1	**0.00E+00**	9.97E-01	**0.00E+00**
SNAI2	**5.75E-89**	1.00E+00	**0.00E+00**
CDH2	**0.00E+00**	**1.92E-29**	**0.00E+00**
miR200	1.00E+00	**6.06E-08**	9.58E-01
AKT1	**2.94E-02**	**0.00E+00**	**2.05E-18**
SNAI1	**0.00E+00**	**2.44E-276**	**2.88E-29**
p63	9.99E-01	**1.84E-06**	1.00E+00
DKK1	8.41E-01	5.79E-01	1.00E+00
ZEB1	1.00E+00	1.00E+00	9.93E-01
CTNNB1	4.26E-01	1.00E+00	**2.27E-02**
TGFbeta	**5.47E-16**	8.41E-01	**0.00E+00**
miR34	5.62E-01	**6.12E-04**	8.41E-01
NICD	1.00E+00	1.00E+00	**0.00E+00**

TABLE D.13: GRN control node identification evaluation metrics. The column Core Nodes in Control Kernels lists all of the nodes that appear in at least one control kernel identified in Kim, Park, and Cho, 2013, as well as the largest strongly connected component in the network. Deepest Feedback Loop(s) details the nodes in the loops at the deepest level of the hierarchy found by IMPLISig. The measures under Deepest Feedback Loop (F1, Precision and Recall) show F1, sensitivity and specificity of using the deepest coherent feedback loop as a predictor of potential control node. *Networks:* SCCC: Saccharomyces Cerevisae Cell Cycle; SP: Schizosaccharomyces Pombe; MCD: Mammalian Cortical Development; ATD: Arabidopsis Thaliana Development; and MMD: Mouse Myeloid Development.

Network	Nodes in Control Kernels	Deepest Feedback Loop(s)	Core Feedback Loop Type(s)	Metrics		
				F1	Precision	Recall
SCCC	{Cdh1, MBF, SBF, Sic1}	{{Sic1, Clb5_6}}	{Negative}	0.333	0.500	0.25
SP	{Cdc25, Rum1, Ste9, Wee1}	{{Cdc2/13*, Ste9}, {Cdc2/13*, Rum1}}	{Negative, Negative}	0.571	0.667	0.5
MCD	{Fgf8_g, Emx2_g, Sp8_g, Fgf8_p, Emx2_p, Sp8_p}	{{Fgf8_g, Fgf8_p}}	{Positive}	0.500	1.000	0.333
ATD	{AP1, LFY, SEP}	{{AG, AP1}}	{Negative}	0.400	0.500	0.333
MMD	{GATA_1, EKLF, FLI_1, PU1}	{{Fli_1, GATA_1}}	{Positive}	0.667	1.000	0.500

TABLE D.14: Control node identification ranks. *Key: k*: node degree, k_{in}: in-degree, k_{out}: out-degree, bc: betweenness centrality, cc: clustering co-efficient and cn: core number. For IMPLISig, we use depth in feedback loop hierarchy as a predictor of significance.

AUROC	SCCC	SP	MCD	ATD	MMD	Mean Rank
k	5	5	2	3	3	3.6
k_{in}	**1**	2	6	6	6	4.2
k_{out}	7	7	4	**1**	3	4.4
bc	2	6	**1**	5	2	3.2
cc	4	3	7	7	7	5.6
cn	3	3	5	3	5	3.8
IMPLISig	5	**1**	3	2	**1**	**2.4**
AP	SCCC	SP	MCD	ATD	MMD	Mean Rank
k	6	3	2	3	3	3.4
k_{in}	**1**	**1**	5	6	7	4
k_{out}	7	7	4	**1**	4	4.6
bc	2	6	**1**	4	2	3
cc	3	3	7	7	5	5
cn	4	3	6	5	6	4.8
IMPLISig	4	2	3	2	**1**	**2.4**

Bibliography

Adamic, Lada A et al. (2000). “Power-law distribution of the world wide web”. In: *science* 287.5461, pp. 2115–2115.

Alanis-Lobato, Gregorio, Pablo Mier, and Miguel A Andrade-Navarro (2016a). “Efficient embedding of complex networks to hyperbolic space via their Laplacian”. In: *Scientific Reports* 6.

— (2016b). “Manifold learning and maximum likelihood estimation for hyperbolic network embedding”. In: *Applied Network Science* 1.1, p. 10.

Albert, Réka and Hans G Othmer (2003). “The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster”. In: *Journal of theoretical biology* 223.1, pp. 1–18.

Alon, Uri (2007). “Network motifs: theory and experimental approaches”. In: *Nature Reviews Genetics* 8.6, p. 450.

Alvarez-Buylla, Elena R et al. (2007). “Gene regulatory network models for plant development”. In: *Current Opinion in Plant Biology* 10.1, pp. 83–91.

Ashourvan, Arian et al. (2019). “Multi-scale detection of hierarchical community architecture in structural and functional brain networks”. In: *Plos one* 14.5, e0215520.

Balleza, Enrique et al. (2008). “Critical dynamics in genetic regulatory networks: examples from four kingdoms”. In: *PLoS One* 3.6.

Barabási, Albert-László (2002). “Linked: How everything is connected to everything else and what it means”. In: *Plume Editors*.

— (2009). “Scale-free networks: a decade and beyond”. In: *science* 325.5939, pp. 412–413.

Barabási, Albert-László and Réka Albert (1999). “Emergence of scaling in random networks”. In: *science* 286.5439, pp. 509–512.

Barber, Michael J (2007). "Modularity and community detection in bipartite networks". In: *Physical Review E* 76.6, p. 066102.

Barrat, Alain et al. (2004). "The architecture of complex weighted networks". In: *Proceedings of the national academy of sciences* 101.11, pp. 3747–3752.

Belkin, Mikhail and Partha Niyogi (2002). "Laplacian eigenmaps and spectral techniques for embedding and clustering". In: *Advances in neural information processing systems*, pp. 585–591.

Bianconi, Ginestra and Christoph Rahmede (2017). "Emergent hyperbolic network geometry". In: *Scientific reports* 7, p. 41974.

Bianconi, Ginestra and Robert M Ziff (2018). "Topological percolation on hyperbolic simplicial complexes". In: *Physical Review E* 98.5, p. 052308.

Blondel, Vincent D et al. (2008). "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10, P10008.

Bojchevski, Aleksandar and Stephan Günnemann (2018). "Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking". In: *International Conference on Learning Representations*, pp. 1–13.

Bollobás, Béla et al. (2003). "Directed scale-free graphs". In: *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 132–139.

Borgatti, Stephen P and Martin G Everett (2000). "Models of core/periphery structures". In: *Social networks* 21.4, pp. 375–395.

Braha, Dan and Yaneer Bar-Yam (2009). "Time-dependent complex networks: Dynamic centrality, dynamic motifs, and cycles of social interactions". In: *Adaptive Networks*. Springer, pp. 39–50.

Brandman, Onn et al. (2005). "Interlinked fast and slow positive feedback loops drive reliable cell decisions". In: *Science* 310.5747, pp. 496–498.

Bruno, Matteo et al. (2019). "Community Detection in the Hyperbolic Space". In: *arXiv preprint arXiv:1906.09082*.

Cannon, James W et al. (1997). "Hyperbolic geometry". In: *Flavors of geometry* 31, pp. 59–115.

Cao, Shaosheng, Wei Lu, and Qiongkai Xu (2016). "Deep neural networks for learning graph representations". In: *Thirtieth AAAI conference on artificial intelligence*.

Chamberlain, BP, JR Clough, and MP Deisenroth (2017). "Neural Embeddings of Graphs in Hyperbolic Space". In: *CoRR*. MLG Workshop 2017.

Chami, Ines et al. (2019). "Hyperbolic graph convolutional neural networks". In: *Advances in Neural Information Processing Systems*, pp. 4869–4880.

Chang, Shiyu et al. (2015). "Heterogeneous network embedding via deep architectures". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 119–128.

Chen, Mo, Qiong Yang, Xiaoou Tang, et al. (2007). "Directed Graph Embedding." In: *IJCAI*, pp. 2707–2712.

Chen, Pengfei, Guangyong Chen, and Shengyu Zhang (2018). "Log Hyperbolic Cosine Loss Improves Variational Auto-Encoder". In:

Chen, Weiqi, Jing Liu, and Shan He (2017). "Prior knowledge guided active modules identification: an integrated multi-objective approach". In: *BMC systems biology* 11.2, p. 8.

Cheng, Tammy MK et al. (2012). "Understanding cancer mechanisms through network dynamics". In: *Briefings in functional genomics* 11.6, pp. 543–560.

Chicco, Davide and Giuseppe Jurman (2020). "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC genomics* 21.1, p. 6.

Choi, Minsoo et al. (2012). "Attractor landscape analysis reveals feedback loops in the p53 network that control the cellular response to DNA damage". In: *Science signaling* 5.251, ra83–ra83.

Chou, Chun Tung (2018). "Detection of persistent signals and its relation to coherent feed-forward loops". In: *Royal Society Open Science* 5.11, p. 181641.

Christensen, Bjarke and Jens Nielsen (1999). "Metabolic network analysis". In: *Bioanalysis and Biosensors for Bioprocess Monitoring*. Springer, pp. 209–231.

Clauset, Aaron, Cristopher Moore, and Mark EJ Newman (2008). "Hierarchical structure and the prediction of missing links in networks". In: *Nature* 453.7191, pp. 98–101.

Clauset, Aaron, Mark EJ Newman, and Cristopher Moore (2004). "Finding community structure in very large networks". In: *Physical review E* 70.6, p. 066111.

Clough, James R and Tim S Evans (2017). "Embedding graphs in Lorentzian spacetime". In: *PloS one* 12.11, e0187301.

Cohen, David PA et al. (2015). "Mathematical modelling of molecular pathways enabling tumour cell invasion and migration". In: *PLoS Comput Biol* 11.11, e1004571.

Cohen, Reuven et al. (2000). "Resilience of the internet to random breakdowns". In: *Physical review letters* 85.21, p. 4626.

Csete, Marie and John Doyle (2004). "Bow ties, metabolism and disease". In: *TRENDS in Biotechnology* 22.9, pp. 446–450.

Cui, Peng et al. (2018). "A survey on network embedding". In: *IEEE Transactions on Knowledge and Data Engineering* 31.5, pp. 833–852.

Daniels, Bryan C et al. (2018). "Criticality distinguishes the ensemble of biological regulatory networks". In: *Physical review letters* 121.13, p. 138102.

De Domenico, Manlio et al. (2015). "Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems". In: *Physical Review X* 5.1, p. 011027.

De Sa, Christopher et al. (2018). "Representation tradeoffs for hyperbolic embeddings". In: *Proceedings of machine learning research* 80, p. 4460.

Decelle, Aurelien et al. (2011). "Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications". In: *Physical Review E* 84.6, p. 066106.

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in neural information processing systems*, pp. 3844–3852.

Dhingra, Bhuwan et al. (2018). "Embedding Text in Hyperbolic Spaces". In: *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pp. 59–69.

Ding, Chris (2004). "A tutorial on spectral clustering". In: *Talk presented at ICML*.

Ding, Chris HQ et al. (2001). "A min-max cut algorithm for graph partitioning and data clustering". In: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, pp. 107–114.

Dittrich, Marcus T et al. (2008). "Identifying functional modules in protein–protein interaction networks: an integrated exact approach". In: *Bioinformatics* 24.13, pp. i223–i231.

Donath, William E and Alan J Hoffman (1973). "Lower bounds for the partitioning of graphs". In: *IBM Journal of Research and Development* 17.5, pp. 420–425.

Drossel, Barbara (2008). "Random boolean networks". In: *Reviews of nonlinear dynamics and complexity* 1, pp. 69–110.

Espinosa-Soto, Carlos, Pablo Padilla-Longoria, and Elena R Alvarez-Buylla (2004). "A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles". In: *The Plant Cell* 16.11, pp. 2923–2939.

Fawcett, Tom (2006). "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8, pp. 861–874.

Feng, Jun et al. (2016). "Knowledge graph embedding by flexible translation". In: *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Ferrell Jr, James E, Tony Yu-Chen Tsai, and Qiong Yang (2011). "Modeling the cell cycle: why do certain circuits oscillate?" In: *Cell* 144.6, pp. 874–885.

Flobak, Åsmund et al. (2015). "Discovery of drug synergies in gastric cancer cells predicted by logical modeling". In: *PLoS computational biology* 11.8.

Fortunato, Santo (2010). "Community detection in graphs". In: *Physics reports* 486.3, pp. 75–174.

Fortunato, Santo and Marc Barthelemy (2007). "Resolution limit in community detection". In: *Proceedings of the National Academy of Sciences* 104.1, pp. 36–41.

Friedlander, Tamar et al. (2015). "Evolution of bow-tie architectures in biology". In: *PLoS Comput Biol* 11.3, e1004055.

Fu, Yun and Yunqian Ma (2012). *Graph embedding for pattern analysis*. Springer Science & Business Media.

Ganea, Octavian, Gary Becigneul, and Thomas Hofmann (2018). "Hyperbolic Entailment Cones for Learning Hierarchical Embeddings". In: *International Conference on Machine Learning*, pp. 1646–1655.

Ganea, Octavian, Gary Bécigneul, and Thomas Hofmann (2018). "Hyperbolic neural networks". In: *Advances in neural information processing systems*, pp. 5345–5355.

García-Pérez, Guillermo, Marián Boguñá, and M Ángeles Serrano (2018). "Multiscale unfolding of real networks by geometric renormalization". In: *Nature Physics* 14.6, pp. 583–589.

García-Pérez, Guillermo et al. (2019). "Mercator: uncovering faithful hyperbolic embeddings of complex networks". In: *New Journal of Physics* 21.12, p. 123033.

Giacomantonio, Clare E and Geoffrey J Goodhill (2010). "A Boolean model of the gene regulatory network underlying Mammalian cortical area development". In: *PLoS Comput Biol* 6.9, e1000936.

Gibert, Jaume, Ernest Valveny, and Horst Bunke (2012). "Graph embedding in vector spaces by node attribute statistics". In: *Pattern Recognition* 45.9, pp. 3072–3083.

Girvan, Michelle and Mark EJ Newman (2002). "Community structure in social and biological networks". In: *Proceedings of the national academy of sciences* 99.12, pp. 7821–7826.

Goentoro, Lea et al. (2009). "The incoherent feedforward loop can provide fold-change detection in gene regulation". In: *Molecular cell* 36.5, pp. 894–899.

Goyal, Palash et al. (2018a). "Capturing edge attributes via network embedding". In: *IEEE Transactions on Computational Social Systems* 5.4, pp. 907–917.

— (2018b). "Embedding networks with edge attributes". In: *Proceedings of the 29th on Hypertext and Social Media*, pp. 38–42.

Grover, Aditya and Jure Leskovec (2016). "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 855–864.

Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *Advances in Neural Information Processing Systems*, pp. 1024–1034.

Hand, David and Peter Christen (2018). "A note on using the F-measure for evaluating record linkage algorithms". In: *Statistics and Computing* 28.3, pp. 539–547.

He, Shan et al. (2016). "Cooperative co-evolutionary module identification with application to cancer disease module discovery". In: *IEEE Transactions on Evolutionary Computation* 20.6, pp. 874–891.

Helikar, Tomáš et al. (2008). "Emergent decision-making in biological signal transduction networks". In: *Proceedings of the National Academy of Sciences* 105.6, pp. 1913–1918.

Herman, Ivan, Guy Melançon, and M Scott Marshall (2000). "Graph visualization and navigation in information visualization: A survey". In: *IEEE Transactions on visualization and computer graphics* 6.1, pp. 24–43.

Hou, Chengbin, Shan He, and Ke Tang (2020). "RoSANE: Robust and Scalable Attributed Network Embedding for Sparse Networks". In: *Neurocomputing*. DOI: 10.1016/j.neucom.2020.05.080. URL: https://doi.org/10.1016/j.neucom.2020.05.080.

Hou, Chengbin et al. (2020). "GloDyNE: Global Topology Preserving Dynamic Network Embedding". In: *arXiv preprint arXiv:2008.01935*.

Huang, Sui et al. (2005). "Cell fates as high-dimensional attractor states of a complex gene regulatory network". In: *Physical review letters* 94.12, p. 128701.

Huang, Xiao, Jundong Li, and Xia Hu (2017a). "Accelerated attributed network embedding". In: *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, pp. 633–641.

— (2017b). "Label informed attributed network embedding". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 731–739.

Iacobelli, Giulio and Daniel Ratton Figueiredo (2016). "Edge-attractor random walks on dynamic networks". In: *Journal of Complex Networks* 5.1, pp. 84–110.

Ideker, Trey et al. (2002). "Discovering regulatory and signalling circuits in molecular interaction networks". In: *Bioinformatics* 18.suppl_1, S233–S240.

Jiang, Fangcui and Long Wang (2010). "Consensus seeking of high-order dynamic multi-agent systems with fixed and switching topologies". In: *International Journal of Control* 83.2, pp. 404–420.

Kauffman, Stuart (1969). "Homeostasis and differentiation in random genetic control networks". In: *Nature* 224.5215, pp. 177–178.

Kernighan, Brian W and Shen Lin (1970). "An efficient heuristic procedure for partitioning graphs". In: *Bell system technical journal* 49.2, pp. 291–307.

Kholodenko, Boris N (2006). "Cell-signalling dynamics in time and space". In: *Nature reviews Molecular cell biology* 7.3, pp. 165–176.

Kim, Jeong-Rae, Yeoin Yoon, and Kwang-Hyun Cho (2008). "Coupled feedback loops form dynamic motifs of cellular networks". In: *Biophysical journal* 94.2, pp. 359–365.

Kim, Jeong-Rae et al. (2011a). "Reduction of complex signaling networks to a representative kernel". In: *Science signaling* 4.175, ra35–ra35.

Kim, Junil, Sang-Min Park, and Kwang-Hyun Cho (2013). "Discovery of a kernel for controlling biomolecular regulatory networks". In: *Scientific reports* 3, p. 2223.

Kim, Junil et al. (2008). "Evolutionary design principles of modules that control cellular differentiation: consequences for hysteresis and multistationarity". In: *Bioinformatics* 24.13, pp. 1516–1522.

Kim, Kibae and Jörn Altmann (2017). "Effect of homophily on network formation". In: *Communications in Nonlinear Science and Numerical Simulation* 44, pp. 482–494.

Kim, Man-Sun et al. (2012). "Spatiotemporal network motif reveals the biological traits of developmental gene regulatory networks in Drosophila melanogaster". In: *BMC systems biology* 6.1, p. 31.

Kim, Tae-Hwan et al. (2011b). "Evolutionary design principles and functional characteristics based on kingdom-specific network motifs". In: *Bioinformatics* 27.2, pp. 245–251.

Kingma, Diederik P and Jimmy Ba (Dec. 2014). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations*.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907*.

Kitano, Hiroaki (2004). "Biological robustness". In: *Nature Reviews Genetics* 5.11, pp. 826–837.

Klammer, Martin et al. (2010). "Identifying differentially regulated subnetworks from phosphoproteomic data". In: *BMC bioinformatics* 11.1, p. 351.

Kojaku, Sadamori and Naoki Masuda (2017). "Finding multiple core-periphery pairs in networks". In: *Physical Review E* 96.5, p. 052313.

Kolch, Walter, Muffy Calder, and David Gilbert (2005). "When kinases meet mathematics: the systems biology of MAPK signalling". In: *FEBS letters* 579.8, pp. 1891–1895.

Krioukov, Dmitri et al. (2009). "Curvature and temperature of complex networks". In: *Physical Review E* 80.3, p. 035101.

Krioukov, Dmitri et al. (2010). "Hyperbolic geometry of complex networks". In: *Physical Review E* 82.3, p. 036106.

Krumsiek, Jan et al. (2011). "Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network". In: *PloS one* 6.8, e22649.

Kryven, Ivan, Robert M Ziff, and Ginestra Bianconi (2019). "Renormalization group for link percolation on planar hyperbolic manifolds". In: *Physical Review E* 100.2, p. 022306.

Lambiotte, Renaud et al. (2011). "Flow graphs: Interweaving dynamics and structure". In: *Physical Review E* 84.1, p. 017102.

Lancichinetti, Andrea and Santo Fortunato (2009). "Community detection algorithms: a comparative analysis". In: *Physical review E* 80.5, p. 056117.

— (2012). "Consensus clustering in complex networks". In: *Scientific reports* 2, p. 336.

Lancichinetti, Andrea, Santo Fortunato, and János Kertész (2009). "Detecting the overlapping and hierarchical community structure in complex networks". In: *New Journal of Physics* 11.3, p. 033015.

Lancichinetti, Andrea, Filippo Radicchi, and José J Ramasco (2010). "Statistical significance of communities in networks". In: *Physical Review E* 81.4, p. 046110.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444.

LeCun, Yann et al. (2006). "A tutorial on energy-based learning". In: *Predicting structured data* 1.0.

Leimeister, Matthias and Benjamin J Wilson (2018). "Skip-gram word embeddings in hyperbolic space". In: *arXiv preprint arXiv:1809.01498*.

Leskovec, Jure and Julian J Mcauley (2012). "Learning to discover social circles in ego networks". In: *Advances in neural information processing systems*, pp. 539–547.

Leskovec, Jure and Rok Sosič (2016). "SNAP: A General-Purpose Network Analysis and Graph-Mining Library". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 8.1, p. 1.

Li, Chao et al. (2019). "Learner2vec-based learner community evolution analysis–a case study involving student card data". In: *IEEE Access* 7, pp. 27416–27425.

Li, Cheng et al. (2017a). "Deepcas: An end-to-end predictor of information cascades". In: *Proceedings of the 26th international conference on World Wide Web*, pp. 577–586.

Li, Dong et al. (2017b). "Active module identification in intracellular networks using a memetic algorithm with a new binary decoding scheme". In: *BMC genomics* 18.2, p. 209.

Li, Fangting et al. (2004). "The yeast cell-cycle network is robustly designed". In: *Proceedings of the National Academy of Sciences* 101.14, pp. 4781–4786.

Li, Jundong et al. (2017c). "Attributed network embedding for learning in a dynamic environment". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 387–396.

Liao, Lizi et al. (2018). "Attributed social network embedding". In: *IEEE Transactions on Knowledge and Data Engineering* 30.12, pp. 2257–2270.

Liu, Weiyi et al. (2017). "Principled multilayer network embedding". In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, pp. 134–141.

Liu, Yang-Yu, Jean-Jacques Slotine, and Albert-László Barabási (2011). "Controllability of complex networks". In: *nature* 473.7346, pp. 167–173.

— (2012). "Control centrality and hierarchical structure in complex networks". In: *Plos one* 7.9, e44459.

Ma, Wenzhe et al. (2009). "Defining network topologies that can achieve biochemical adaptation". In: *Cell* 138.4, pp. 760–773.

Majdandzic, Antonio et al. (2014). "Spontaneous recovery in dynamical networks". In: *Nature Physics* 10.1, pp. 34–38.

Man, Tong et al. (2016). "Predict anchor links across social networks via an embedding approach." In: *Ijcai*. Vol. 16, pp. 1823–1829.

Mangan, Shmoolik and Uri Alon (2003). "Structure and function of the feed-forward loop network motif". In: *Proceedings of the National Academy of Sciences* 100.21, pp. 11980–11985.

Mathieu, Emile et al. (2019). "Continuous hierarchical representations with poincaré variational auto-encoders". In: *Advances in neural information processing systems*, pp. 12565–12576.

McDonald, David, Sami Cass Darweish, and Shan He (2020). "Multi-scale Hierarchical Decomposition of Bow-tie Architectures in Intra-cellular Networks". In: *Bioinformatics*.

McDonald, David and Shan He (2020a). "HEADNet: Hyperbolic Embedding of Attributed Directed Networks". In: *IEEE Transactions on Neural Networks and Learning Systems*.

— (2020b). "HEAT: Hyperbolic Embedding of Attributed Networks". In: *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, pp. 28–40.

McInnes, Leland, John Healy, and Steve Astels (2017). "hdbscan: Hierarchical density based clustering". In: *Journal of Open Source Software* 2.11, p. 205.

Mikolov, Tomas et al. (2013a). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, Tomas et al. (2013b). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Milgram, Stanley (1967). "The small world problem". In: *Psychology today* 2.1, pp. 60–67.

Milo, Ron et al. (2002). "Network motifs: simple building blocks of complex networks". In: *Science* 298.5594, pp. 824–827.

Mitarai, Namiko, Mogens Høgh Jensen, and Szabolcs Semsey (2015). "Coupled positive and negative feedbacks produce diverse gene expression patterns in colonies". In: *Mbio* 6.2.

Mitra, Koyel et al. (2013). "Integrative approaches for finding modular structure in biological networks". In: *Nature reviews. Genetics* 14.10, p. 719.

Mitrophanov, Alexander Y and Eduardo A Groisman (2008). "Positive feedback in cellular control systems". In: *Bioessays* 30.6, pp. 542–555.

Mucha, Peter J and Mason A Porter (2010). "Communities in multislice voting networks". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20.4, p. 041108.

Mucha, Peter J et al. (2010). "Community structure in time-dependent, multiscale, and multiplex networks". In: *science* 328.5980, pp. 876–878.

Nacu, Şerban et al. (2007). "Gene expression network analysis and applications to immunology". In: *Bioinformatics* 23.7, pp. 850–858.

Nagano, Yoshihiro et al. (2019). "A Wrapped Normal Distribution on Hyperbolic Space for Gradient-Based Learning". In: *International Conference on Machine Learning*, pp. 4693–4702.

Nagler, Jan, Anna Levina, and Marc Timme (2011). "Impact of single links in competitive percolation". In: *Nature Physics* 7.3, pp. 265–270.

Nepusz, Tamás and Tamás Vicsek (2012). "Controlling edge dynamics in complex networks". In: *Nature Physics* 8.7, pp. 568–573.

Newman, Mark EJ (2003). "Mixing patterns in networks". In: *Physical review E* 67.2, p. 026126.

Ng, Andrew Y, Michael I Jordan, Yair Weiss, et al. (2002). "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems* 2, pp. 849–856.

Nickel, Maximillian and Douwe Kiela (2017). "Poincaré embeddings for learning hierarchical representations". In: *Advances in neural information processing systems*, pp. 6338–6347.

— (2018). "Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry". In: *International Conference on Machine Learning*, pp. 3776–3785.

Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov (2016). "Learning convolutional neural networks for graphs". In: *International Conference on Machine Learning*, pp. 2014–2023.

Odom, Duncan T et al. (2004). "Control of pancreas and liver gene expression by HNF transcription factors". In: *Science* 303.5662, pp. 1378–1381.

Olsman, Noah and Lea Goentoro (2016). "Allosteric proteins as logarithmic sensors". In: *Proceedings of the National Academy of Sciences* 113.30, E4423–E4430.

Page, Lawrence et al. (1999). *The pagerank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab.

Palla, Gergely et al. (2005). "Uncovering the overlapping community structure of complex networks in nature and society". In: *Nature* 435.7043, pp. 814–818.

Papadopoulos, Fragkiskos, Rodrigo Aldecoa, and Dmitri Krioukov (2015). "Network geometry inference using common neighbors". In: *Physical Review E* 92.2, p. 022807.

Papadopoulos, Fragkiskos, Constantinos Psomas, and Dmitri Krioukov (2015). "Network mapping by replaying hyperbolic growth". In: *IEEE/ACM Transactions on Networking (TON)* 23.1, pp. 198–211.

Papadopoulos, Fragkiskos et al. (2010). "Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces". In: *INFOCOM, 2010 Proceedings IEEE*. IEEE, pp. 1–9.

Papadopoulos, Fragkiskos et al. (2012). "Popularity versus similarity in growing networks". In: *Nature* 489.7417, pp. 537–540.

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena (2014). "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 701–710.

Pomerening, Joseph R, Eduardo D Sontag, and James E Ferrell (2003). "Building a cell cycle oscillator: hysteresis and bistability in the activation of Cdc2". In: *Nature cell biology* 5.4, pp. 346–351.

Radicchi, Filippo et al. (2004). "Defining and identifying communities in networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 101.9, pp. 2658–2663.

Reynolds, William F (1993). "Hyperbolic geometry on a hyperboloid". In: *The American mathematical monthly* 100.5, pp. 442–455.

Roli, Andrea et al. (2018). "Dynamical criticality: overview and open questions". In: *Journal of Systems Science and Complexity* 31.3, pp. 647–663.

Rombach, M Puck et al. (2014). "Core-periphery structure in networks". In: *SIAM Journal on Applied mathematics* 74.1, pp. 167–190.

Rosvall, Martin and Carl T Bergstrom (2007). "An information-theoretic framework for resolving community structure in complex networks". In: *Proceedings of the National Academy of Sciences* 104.18, pp. 7327–7331.

— (2008). "Maps of random walks on complex networks reveal community structure". In: *Proceedings of the National Academy of Sciences* 105.4, pp. 1118–1123.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Samaga, Regina et al. (2009). "The logic of EGFR/ErbB signaling: theoretical properties and analysis of high-throughput data". In: *PLoS computational biology* 5.8, e1000438.

Sarkar, Purnamrita and Andrew W Moore (2006). "Dynamic social network analysis using latent space models". In: *Advances in Neural Information Processing Systems*, pp. 1145–1152.

Sarkar, Rik (2011). "Low distortion delaunay embedding of trees in hyperbolic plane". In: *International Symposium on Graph Drawing*. Springer, pp. 355–366.

Schaub, Michael T et al. (2017). "The many facets of community detection in complex networks". In: *Applied Network Science* 2.1, p. 4.

Serra, Roberto et al. (2007). "Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data". In: *Journal of theoretical biology* 246.3, pp. 449–460.

Shen-Orr, Shai S et al. (2002). "Network motifs in the transcriptional regulation network of Escherichia coli". In: *Nature genetics* 31.1, pp. 64–68.

Shi, Jianbo and Jitendra Malik (2000). "Normalized cuts and image segmentation". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8, pp. 888–905.

Shin, Sung-Young et al. (2010). "Functional roles of multiple feedback loops in extracellular signal-regulated kinase and Wnt signaling pathways that regulate epithelial-mesenchymal transition". In: *Cancer research* 70.17, pp. 6715–6724.

Shmulevich, Ilya and Stuart A Kauffman (2004). "Activities and sensitivities in Boolean network models". In: *Physical review letters* 93.4, p. 048701.

Staudt, CL, A Sazonovs, and H Meyerhenke. "NetworKit: A tool suite for large-scale network analysis". In: *Network Science To appear* ().

Su, Cui, Jun Pang, and Soumya Paul (2019). "Towards optimal decomposition of Boolean networks". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Subramanian, Aravind et al. (2005). "Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles". In: *Proceedings of the National Academy of Sciences* 102.43, pp. 15545–15550.

Sun, Jiankai et al. (2017). "Breaking Cycles In Noisy Hierarchies". In: *Proceedings of the 2017 ACM on Web Science Conference*. WebSci '17. ACM, pp. 151–160. ISBN: 978-1-4503-4896-6. DOI: 10.1145/3091478.3091495. URL: http://doi.acm.org/10.1145/3091478.3091495.

Sun, Jiankai et al. (2019). "Atp: Directed graph embedding with asymmetric transitivity preservation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 265–272.

Supper, Jochen et al. (2009). "BowTieBuilder: modeling signal transduction pathways". In: *BMC systems biology* 3.1, p. 67.

Suzuki, Ryota, Ryusuke Takahama, and Shun Onoda (2019). "Hyperbolic Disk Embeddings for Directed Acyclic Graphs". In: *International Conference on Machine Learning*, pp. 6066–6075.

Tang, Jian et al. (2015). "Line: Large-scale information network embedding". In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 1067–1077.

Tarjan, Robert (1972). "Depth-first search and linear graph algorithms". In: *SIAM journal on computing* 1.2, pp. 146–160.

Tenenbaum, Joshua B, Vin De Silva, and John C Langford (2000). "A global geometric framework for nonlinear dimensionality reduction". In: *science* 290.5500, pp. 2319–2323.

Tharwat, Alaa (2018). "Classification assessment methods". In: *Applied Computing and Informatics*.

Thomas, Josephine Maria et al. (2016). "Machine learning meets network science: dimensionality reduction for fast and efficient embedding of networks in the hyperbolic space". In: *arXiv preprint arXiv:1602.06522*.

Timár, G et al. (2017). "Mapping the structure of directed networks: Beyond the bow-tie diagram". In: *Physical review letters* 118.7, p. 078301.

Van Dongen, Stijn Marinus (2001). "Graph clustering by flow simulation". PhD thesis.

Vandin, Fabio, Eli Upfal, and Benjamin J Raphael (2011). "Algorithms for detecting significantly mutated pathways in cancer". In: *Journal of Computational Biology* 18.3, pp. 507–522.

Von Luxburg, Ulrike (2007). "A tutorial on spectral clustering". In: *Statistics and computing* 17.4, pp. 395–416.

Wang, Bo et al. (2017a). "Vicus: Exploiting local structures to improve network-based analysis of biological data". In: *PLoS computational biology* 13.10.

Wang, Daixin, Peng Cui, and Wenwu Zhu (2016). "Structural deep network embedding". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234.

Wang, Guanyu et al. (2010). "Process-based network decomposition reveals backbone motif structure". In: *Proceedings of the National Academy of Sciences* 107.23, pp. 10478–10483.

Wang, Suhang et al. (2017b). "Signed network embedding in social media". In: *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, pp. 327–335.

Wang, Xiao, Yiding Zhang, and Chuan Shi (2019). "Hyperbolic heterogeneous information network embedding". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 5337–5344.

Wang, Xiao et al. (2017c). "Community preserving network embedding". In: *Thirty-first AAAI conference on artificial intelligence*.

Wang, Zuxi et al. (2019). "Fast hyperbolic mapping based on the hierarchical community structure in complex networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2019.12, p. 123401.

Wasserman, Stanley, Katherine Faust, et al. (1994). *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press.

Watts, Duncan J and Steven H Strogatz (1998). "Collective dynamics of 'small-world'networks". In: *nature* 393.6684, pp. 440–442.

Wei, Y-C and C-K Cheng (1991). "Ratio cut partitioning for hierarchical designs". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 10.7, pp. 911–921.

Whitesitt, J Eldon (2012). *Boolean algebra and its applications*. Courier Corporation.

Wilson, Benjamin and Matthias Leimeister (2018). "Gradient descent in hyperbolic space". In: *arXiv:1805.08207*.

Wilson, Richard C et al. (2014). "Spherical and hyperbolic embeddings of data". In: *IEEE transactions on pattern analysis and machine intelligence* 36.11, pp. 2255–2269.

Wu, Hulin et al. (2014). "Sparse additive ordinary differential equations for dynamic gene regulatory network modeling". In: *Journal of the American Statistical Association* 109.506, pp. 700–716.

Wu, Zongning, Zengru Di, and Ying Fan (2019). "A hyperbolic Embedding Model for Directed Networks". In: *arXiv preprint arXiv:1906.03597*.

Yang, Cheng et al. (2015). "Network representation learning with rich text information". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Yang, Liang et al. (2016). "Modularity Based Community Detection with Deep Learning." In: *IJCAI*, pp. 2252–2258.

Zhan, Kun et al. (2018). "Graph structure fusion for multiview clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 31.10, pp. 1984–1993.

Zhang, Carolyn et al. (2016a). "Processing oscillatory signals by incoherent feedforward loops". In: *PLoS computational biology* 12.9, e1005101.

Zhang, Hui et al. (2016b). "Integrated proteogenomic characterization of human high-grade serous ovarian cancer". In: *Cell* 166.3, pp. 755–765.

Zhang, Qi et al. (2018). "Using single-index ODEs to study dynamic gene regulatory network". In: *PloS one* 13.2, e0192833.

Zhang, Xiao, Travis Martin, and Mark EJ Newman (2015). "Identification of core-periphery structure in networks". In: *Physical Review E* 91.3, p. 032803.

Zhao, Yin, Jongrae Kim, and Maurizio Filippone (2013). "Aggregation algorithm towards large-scale Boolean network analysis". In: *IEEE Transactions on Automatic Control* 58.8, pp. 1976–1985.

Zhou, Chang et al. (2017). "Scalable graph embedding for asymmetric proximity". In: *Thirty-First AAAI Conference on Artificial Intelligence*.