



DEEP NEURAL NETWORKS ON GENETIC MOTIF DISCOVERY

the Interpretability and Identifiability Issues

by

YU ZHANG

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
December 2021

UNIVERSITY OF
BIRMINGHAM

University of Birmingham Research Archive

e-theses repository

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.

ABSTRACT

Deep neural networks have made great success in a wide range of research fields and real-world applications. However, as a black-box model, the drastic advances in the performance come at the cost of model interpretability. This becomes a big concern especially for domains that are safety-critical or have ethical and legal requirements (e.g., avoiding algorithmic discrimination). In other situations, interpretability might be able to help scientists gain new “knowledge” that is learnt by the neural networks (e.g., computational genomics), and neural network based genetic motif discovery is such a field. It naturally leads us to another question: Can current neural network based motif discovery methods identify the underlying motifs from the data? How robust and reliable is it? In other words, we are interested in the motif identifiability problem.

In this thesis, we first conduct a comprehensive review of the current neural network interpretability research, and propose a novel unified taxonomy which, to the best of our knowledge, provides the most comprehensive and clear categorisation of the existing approaches. Then we formally study the motif identifiability problem in the context of neural network based motif discovery (i.e., if we only have access to the predictive performance of a neural network, which is a black-box, how well can we recover the underlying “true” motifs by interpreting the learnt model). Systematic controlled experiments show that although accurate models tend to recover the underlying motifs better, the motif identifiability (a measure of the similarity between true motifs and learnt motifs) still varies in a large range. Also, the over-complexity (without overfitting) of a high-accuracy model (e.g., using 128 kernels while 16 kernels are already good enough) may be harmful to the motif identifiability. We thus propose a robust neural network based motif discovery workflow addressing above issues, which is verified on both synthetic and real-world datasets. Finally, we propose probabilistic kernels in place of conventional convolutional kernels and study whether it would

be better to directly learn probabilistic motifs in the neural networks rather than post hoc interpretation. Experiments show that although probabilistic kernels have some merits (e.g., stable output), their performance is not comparable to classic convolutional kernels under the same network setting (the number of kernels).

ACKNOWLEDGMENTS

As a student of the Joint PhD Training Program offered by the University of Birmingham (UoB) and Southern University of Science and Technology (SUSTech), I would like to thank my supervisors, Prof. Peter Tiño and Prof. Aleš Leonardis in UoB and Prof. Ke Tang in SUSTech, for their supervision as well as UoB and SUSTech for their support for my studies in both university.

Many thanks to Dr. Ildem Akerman for the biological datasets and helpful discussions on my second work. I would like to thank my thesis group members, Prof. Mohan Sridharan, Dr. Hyung Jin Chang, and also administrative staffs, Ms. Kate Sterne, Ms. Sarah Brookes, Ms. Kate Campbell. I would also like to thank Dr. Shan He and Dr. Ke Chen for agreeing to be the examiners of this thesis.

During my study in UoB, I am lucky to have the company of many good friends, Dr. Chengbin Hou (soon to be), Rui He, Jinxing Sun, Boping Guo, Yayun Pu, Mingyang Feng, Cheng Li. I am also grateful to many friends in SUSTech: Dr. Shengcai Liu, Dr. Guiying Li, Dr. Xiaofei Lu, Dr. Liangpeng Zhang, Dr. Wenjing Hong, Dr. Peng Yang, Zhiyuan Wang, Xuanfeng Li, Zeyu Dai, Fu Peng, Ning Lu, Qi Yang, Lan Tang, many from Prof. Xin Yao's lab: Dr. Bo Yuan, Dr. Liyan Song, Guoji Fu, Shuyi Zhang, Dr. Weijie Zheng, Hao Tong, and many from Prof. Qi Wang's lab: Weihuang Wen, Yi Liu.

I have also met many old friends in different cities or countries in the world: Xiaopan Cui (Germany), Hao Zheng (Edinburgh), Xueming Hui (France), Hongxiang Chen and Yingqi Hou (London), and more in Shenzhen: Leiming Xu, Yihang Wang, Bowen Bai, Jie Dong, Yuda Wang, Jintao Tuo, Jiaming Sun, Zebin Ou, Lingqi Zhang, Zhiyuan Fang.

Finally, I would like to thank my parents, grandparents, uncles, aunts, and cousins, my big loving family. Although I only go back home twice a year, I can always feel your love and support.

Contents

1	Introduction	1
1.1	Background	2
1.2	Scope of the Thesis	4
1.3	Research Questions and Contributions of the Thesis	4
1.4	Publications Resulting from the Thesis	7
2	Background	8
2.1	Deep Neural Networks	8
2.1.1	Feedforward Neural Networks	8
2.1.2	Optimisation	10
2.1.3	Convolution and Convolutional Neural Networks	14
2.2	Importance of Interpretability	16
2.2.1	High Reliability Requirement	16
2.2.2	Ethical and Legal Requirement	17
2.2.3	Scientific Usage	17
2.3	Motif Discovery	18
2.3.1	Motif Representation	18
2.3.2	Classic Motif Discovery Approaches	21
2.3.3	Neural Network Based Motif Discovery	22
2.4	Identifiability Issue	23
2.5	Chapter Summary	24

3	A Novel 3D Taxonomy of Neural Network Interpretability	25
3.1	An (Extended) Definition of Interpretability	26
3.2	Previous Work	28
3.3	A 3D Taxonomy	30
3.4	Passive Interpretation of Trained Networks	36
3.4.1	Passive, <i>Rule</i> as Explanation	36
3.4.2	Passive, <i>Hidden Semantics</i> as Explanation	43
3.4.3	Passive, <i>Attribution</i> as Explanation	45
3.4.4	Passive, Explanation <i>by Example</i>	53
3.5	Active Interpretability Intervention During Training	54
3.5.1	Active, <i>Rule</i> as Explanation (semi-local or global)	54
3.5.2	Active, <i>Hidden semantics</i> as Explanation (global)	55
3.5.3	Active, <i>Attribution</i> as Explanation	55
3.5.4	Active, Explanations <i>by Prototypes</i> (global)	56
3.6	Evaluation of Interpretability	56
3.7	Discussion	58
3.8	Chapter Summary	59
4	Model Performance and Motif Identifiability	60
4.1	Background and Motivation	61
4.2	The Proposed Methods	64
4.2.1	Probabilistic Motifs and Motif Similarity	64
4.2.2	Synthetic Data Generation	67
4.3	Experimental Studies	70
4.3.1	Experiment Setup	70
4.3.2	The Motif Identifiability Problem	73
4.3.3	Quantitatively Evaluate Model identifiability with AU-CDF	74
4.3.4	Accurate Models Tend to Have Better Motif Identifiability	76

4.3.5	The Sweet Spot of the Number of Convolutional Kernels	76
4.3.6	Experiments on Synthetic Dataset 2 and 3	78
4.4	Chapter Summary	81
5	A Robust Neural Network Based Motif Discovery Workflow	84
5.1	The Proposed Robust Motif Discovery Workflow	85
5.1.1	Step 1: Finding an Appropriate Number of Kernels	85
5.1.2	Step 2: Training Multiple Models with Re-sampled Data	85
5.1.3	Step 3: Clustering and Representative Motifs	86
5.2	Experimental Studies	88
5.2.1	Experiment Setup	88
5.2.2	Motif Recovery on Synthetic Datasets	89
5.2.3	Recovered Transcription Factors	94
5.2.4	Shuffled Background Sequences and “Negative” Phantom Motifs . . .	95
5.3	Chapter Summary	98
6	Direct Learning of Probabilistic Motifs in Neural Networks	100
6.1	Motivation	101
6.2	The Proposed Approach	102
6.3	Experimental Studies	104
6.3.1	Experiment Setup	104
6.3.2	Probabilistic Kernels Are More Stable	107
6.3.3	Accuracy	108
6.3.4	LikelihoodNet Needs More Kernels	109
6.3.5	Temperature Parameter and Learnt Motifs	111
6.4	Chapter Summary	111
7	Conclusions	114
7.1	Thesis Summary	114

7.2	Limitations	116
7.3	Future Work	117
A	Supplementary Tables	120
B	Supplementary Figures	122
	References	125

List of Figures

2.1	A PWM for -10 region of <i>E.coli</i> promoters (Stormo et al., 1982). The score for the consensus sequence TATAAT is the sum of the shaded elements, 85. Other sequences that are different from it will have lower scores.	20
2.2	The sequence logo of the LexA-binding motif of Gram-positive bacteria. (source: Wikipedia)	20
3.1	The 3 dimensions of our taxonomy.	31
3.2	The distribution of the interpretability papers in the 3D space of our taxonomy. We can rotate and observe the density of work in certain areas/planes and find the missing parts of interpretability research. (Please see https://yzhang-gh.github.io/tmp-data/index.html for an online interactive version.)	35
4.1	From convolutional kernel to PPM.	63
4.2	An illustration of the proposed (asymmetric, from T to P) similarity measurement.	67
4.3	The 8 ground-truth motifs used in the synthetic datasets. The y -axis is the accumulated probability.	68
4.4	Motif identifiability vs. model performance (synthetic dataset 1). The data points are from neural networks with 4 to 9 kernels (50 runs each). For each ground-truth motif, we identify a most similar kernel from every run (i.e., a network) among its learnt convolutional kernels and make the scatter plot. .	74

4.5	Visualisation of kernels with different similarities to motif 0. The motifs are shown in sequence logos (Stormo et al., 1982 ; Schneider and Stephens, 1990) for clear comparison. The y -axis is the accumulated probability multiplied with the information content, $2 - \text{entropy}$, at each position.	75
4.6	Marginal identifiability kernel density estimation for data points of high/low accuracy. The coloured numbers are the AU-CDF values (less is better). . .	77
4.7	Identifiability-performance distributions under different numbers of kernels. .	78
4.8	The AU-CDF per number of kernels (synthetic dataset 1). Less is better. The empty circle indicates the minimal value in the line.	79
4.9	Motif identifiability vs. model performance (synthetic dataset 2). Please see Figure 4.4 for details.	80
4.10	The AU-CDF per number of kernels (synthetic dataset 2). Less is better. The empty circle indicates the minimal value in the line.	81
4.11	The AU-CDF per number of kernels (synthetic dataset 3). Less is better. The empty circle indicates the minimal value in the line.	82
5.1	The alignment of candidate motifs within a cluster. The box with solid border means a probability distribution P over the nucleotide index set \mathcal{N} and the box with dotted line border means a wildcard nucleotide distribution for padding. . .	87
5.2	The (validation set) model performance vs. number of kernels curve for all three synthetic datasets. The y axis is the average accuracy over all classes. Each model is trained 5 times (independently) and every dot corresponds to a single run. The shadow means one standard derivation (plus or minus). . .	90
5.3	The hierarchical clustering result of 50×5 learnt kernels (synthetic dataset 1). They are kernels from 50 5-kernel neural networks, with the best hyper-parameters on the validation set. Please see Figure 5.4 for their comparison to “ground truth” motifs.	91

5.4	Ground-truth motifs (motif 0 to 3) and learnt representative kernels. The first row shows the four motifs in synthetic dataset 1. The second row shows the found motifs by our method. They are annotated in the clustering plot (Figure 5.3). The y -axis is the accumulated probability.	91
5.5	The hierarchical clustering result of learned kernels (synthetic dataset 3). . .	92
5.6	Ground-truth motifs (motif 4 to 7) and learnt representative kernels. The first row shows the other four motifs in synthetic dataset 3 in addition to dataset 1. The second row shows the found motifs by our method. They are annotated in the clustering plot (Figure 5.5). The y -axis is the accumulated probability.	93
5.7	Visualisations of the unmatched kernels in Figure 5.6. Note the empty position can be viewed as uninformative wildcard distribution.	93
5.8	The (validation set) model performance vs. number of kernels curve (real dataset 1, FOXA2). The shadow means one standard derivation (plus or minus).	94
5.9	The hierarchical clustering result of learned kernels (real dataset 1, FOXA2).	96
5.10	Motifs found by the neural network (left) and Homer (right)	96
5.11	The sensitivity of the predicted probability of positive class to each kernel (real dataset 1). It is the derivative of the output positive class neuron (in the last layer) with respect to the output of the first-layer kernel convolution (after global average/maximum pooling).	98

6.1	The average response of learnt kernels on the sequences in validation set (synthetic dataset 1). The first row shows the results of convolutional kernels and the second row is for the likelihood kernels. The response has been standardised (zero mean, unit standard deviation) respectively for each kernel. The vertical dashed lines are the related ground-truth motif locations. There might be a slight shift between the peaks of response and the motif locations as the kernel is usually longer than the motif, or the kernel may only contain a part of a motif. Please see Supplementary Figure B.4 for the results of other kernels.	107
6.2	The effect of temperature parameter on learned motifs (synthetic dataset 1). The y -axis is the accumulated probability. The learnt kernels are trimmed in both ends if there are uninformative wildcard positions (entropy less than 1.95, see Equation 4.9). All above neural networks (under different T) have comparable performance.	112
B.1	The (validation set) model performance vs. number of kernels curve (real dataset 2, PDX1). The shadow means one standard derivation (plus or minus).	122
B.2	The hierarchical clustering result of learned kernels (real dataset 2, PDX1). .	123
B.3	Motifs found by the neural network (left) and Homer (right) on real dataset 2 (PDX1). The confidence level of Homer is much lower than that of dataset 1.	123
B.4	The complete results for Figure 6.1. The probabilistic kernels (second row) in (d) and (e) jointly represent the motif 0 (overlapping on one nucleotide), which are $\begin{smallmatrix} \text{G} & \text{A} & \text{A} & \text{T} \\ \text{A} & \text{A} & \text{A} & \text{A} \end{smallmatrix}$, $\begin{smallmatrix} \text{T} & \text{C} & \text{C} & \text{A} & \text{G} & \text{A} \\ \text{A} & \text{A} & \text{T} & \text{A} & \text{G} & \text{A} \end{smallmatrix}$ and $\begin{smallmatrix} \text{G} & \text{A} & \text{A} & \text{T} & \text{C} & \text{T} & \text{A} & \text{G} & \text{A} \\ \text{A} & \text{A} & \text{A} & \text{A} & \text{C} & \text{A} & \text{G} & \text{A} & \text{A} \end{smallmatrix}$ respectively.	124

List of Tables

3.1	Some interpretable “terms” used in practice.	27
3.2	Example explanations of networks. Please see Section 3.3 (understanding the latter two dimensions) for details. Due to lack of space, we do not provide examples for semi-local interpretability here.	33
3.3	An overview of the interpretability papers. The colours correspond to the three dimensions in Figure 3.1.	37
3.4	Formulation of gradient-related attribution methods. S_c is the output for class c (and it can be any neuron of interest), σ is the nonlinearity in the network and g is a replacement of σ' (the derivative of σ) in $\frac{\partial S_c(\mathbf{x})}{\partial \mathbf{x}}$ in order to rewrite DeepLIFT and LRP with gradient formulation (see Ancona, Ceolini, et al., 2018 for more details). \mathbf{x}_i is the i -th feature (pixel) of \mathbf{x}	48
4.1	Classification rules of synthetic datasets. The numbers are the indices of motifs (0-based). Symbol \wedge means logical AND operation, while \vee means logical OR	70
4.2	The hyperparameters of neural networks in the experiments.	72
4.3	The AU-CDF of marginal interpretability distribution for high-accuracy and (relatively) low-accuracy models (synthetic dataset 1).	77
4.4	The AU-CDF of marginal interpretability distribution (synthetic dataset 2). Please see Table 4.3 for details.	79

4.5	The AU-CDF of marginal interpretability distribution for high-accuracy and (relatively) low-accuracy models (synthetic dataset 3). Please see Table 4.3 for details.	82
6.1	The hyperparameters of neural networks in the experiments.	105
6.2	The performance of neural networks with different kernels. The performance is the average accuracy (in percentage) of all classes. For each given number of kernels, we find the best hyperparameter combination using the validation set and report the 5-run average of the test-set accuracy, along with its standard derivation. The “diff” column is the relative difference comparing to that of convolutional neural networks. Also see Table 6.2 (continued) for results on real datasets.	109
6.2	(continued) The performance of different neural networks on real datasets. .	110
6.3	The performance of LikelihoodNet under different temperature T on real dataset 1 (FOXA2). The results for $T = 1$ (default setting) are copied from Table 6.2 (continued) for easy comparison.	112
A.1	Additional results of LikelihoodNet on real dataset FOXA2.	120
A.2	The one-sided Wilcoxon signed-rank test results for Table 6.2 (extended to 15 runs for each number of kernels). It is a non-parametric version of the paired Student’s t -test. The null hypothesis is that the performance of LikelihoodNet is not greater than that of ConvNet. When the number of kernels is larger than the number of true motifs (4 for synthetic dataset 1, and 8 for synthetic dataset 3), the performance difference is significant (the p -values are almost always less than 0.05).	121

List of Algorithms

1	Generate synthetic motifs	69
2	Generate synthetic sequences from motifs and a classification rule	71

Acronyms

CDF cumulative density function.

CNN convolutional neural network.

DNN deep neural network.

EM expectation-maximisation.

ERM empirical risk minimisation.

GAN generative adversarial networks.

GRU gated recurrent unit.

LSTM long short-term memory.

MLP multilayer perceptron.

MSE mean squared error.

NLP natural language processing.

PCA principal component analysis.

PDF probability density function.

PFM position frequency matrix.

PPM position probability matrix.

PSSM position-specific scoring matrix.

PWM position weight matrix.

ReLU rectified linear unit.

RNN recurrent neural network.

SGD stochastic gradient descent.

SRM structural risk minimisation.

SVM support vector machine.

TF transcription factor.

CHAPTER 1

Introduction

Deep neural networks (DNNs) have recently shown great capability in various research fields and applications (LeCun, Bengio, et al., 2015) such as computer vision (Krizhevsky et al., 2012; He, Zhang, et al., 2016), speech recognition (Hinton et al., 2012), natural language processing (NLP) (Sutskever et al., 2014), game playing (Silver et al., 2017), and other fields such as bioinformatics (Xiong et al., 2015; Zhang, Hu, et al., 2017) which is the main focus of this thesis. However, “there is no such thing as a free lunch”. Comparing to classic machine learning models (e.g., decision trees), deep neural networks are able to get better performance but have a disadvantage of being much more uninterpretable. The lack of interpretability directly influences the usage of deep neural networks in some domains, for example, financial systems and criminal justice. There is also other situation where interpretability is needed, which is to help gain new knowledge in scientific research, such as astronomy and bioinformatics (Parks et al., 2018; Xiong et al., 2015; Zhang, Hu, et al., 2017). Neural network based motif discovery (He, Shen, et al., 2021) is such an example, while in this thesis we systematically investigate it as a motif identifiability problem.

In this chapter, the general background, research questions and contributions of this thesis are introduced. The rest of this chapter is organised as follows: Section 1.1 gives a brief introduction of deep neural networks, and the motif discovery problem in bioinformatics.

In Section 1.2, we clarify a possible confusion about neural network interpretability. And Section 1.3 summarises the main content and major contributions of the thesis. Finally, Section 1.4 lists the published and submitted papers resulting from the thesis.

1.1 Background

Over the last few years, deep neural networks have achieved tremendous success, while the latest applications can be found in these surveys ([Dong et al., 2021](#); [Dixit and Silakari, 2021](#); [Bouwman et al., 2019](#)). They have not only beaten many previous machine learning techniques (e.g., decision trees, support vector machines (SVMs)), but also achieved the state-of-the-art performance on certain real-world tasks ([Wu, Schuster, et al., 2016](#); [Hinton et al., 2012](#)). Products powered by DNNs are now used by billions of people¹, e.g., in facial and voice recognition. DNNs have also become powerful tools for many scientific fields, such as medicine ([Wainberg et al., 2018](#)), bioinformatics ([Xiong et al., 2015](#); [Zhang, Hu, et al., 2017](#)) and astronomy ([Parks et al., 2018](#)), which usually involve massive data volumes.

However, deep learning still has some significant disadvantages. As a really complicated model with millions of free parameters (e.g., AlexNet ([Krizhevsky et al., 2012](#)), 62 million), DNNs are often found to exhibit unexpected behaviours. For instance, even though a network could get the state-of-the-art performance and seemed to generalise well on the object recognition task, [Szegedy et al. \(2014\)](#) found a way that could arbitrarily change the network’s prediction by applying a certain imperceptible change to the input image. This kind of modified input is called “adversarial example”. [Nguyen, Yosinski, et al. \(2015\)](#) showed another way to produce completely unrecognisable images (e.g., look like white noise), which are, however, recognised as certain objects by DNNs with 99.99% confidence. These observations suggest that even though DNNs can achieve superior performance on many tasks, their underlying mechanisms may be very different from those of humans’ and have not yet been well-understood.

¹<https://www.acm.org/media-center/2019/march/turing-award-2018>

The interpretability issue not only affects people’s trust on deep learning systems, but also is related to many ethical problems, e.g., algorithmic discrimination. Moreover, interpretability is a desired property for deep networks to become powerful tools in other research fields, e.g., drug discovery and genomics. Take computational genomics as an example, deep neural networks have been used to predict sequence specificity of DNA- and RNA-binding proteins (Alipanahi et al., 2015). Besides their high predictive performance, the interpretation of the learnt models can suggest potential motifs (i.e., motif discovery). Ideally, if a neural network has higher (out-of-sample) performance, it must have learnt better “knowledge” about the task. However, the “knowledge” is hidden behind the numerous real-valued weights and cannot be easily revealed.

The lack of interpretability of the overall model can hamper the efforts by scientists to extract biologically useful information from the trained network, i.e., relevant sequential motifs. Due to the limitation of the neural network interpretation methods, the current common practice is to extract motifs from the trained first-layer convolutional kernels. Then there is a natural question: *How stable and reliable is the neural network based motif discovery?* In other words, if we only have access to the (out-of-sample) performance of a neural network, which is a black-box, how well will the underlying “true” motifs be identified?

Without taking into account the motif identifiability, current methods usually optimise the neural networks for performance on a given machine learning task, which can lead to potentially large and deep models. This not only increases the difficulty of the model interpretation, but can also bring into question above interpretation method (viewing the first-layer convolutional kernels as potential biologically relevant motifs), as the motifs may be represented in a distributed way and hierarchically.

Motivated by aforementioned issues, this thesis focuses on the neural network interpretability, and the motif identifiability problem in neural network based motif discovery, specifically, what is the relationship between model performance and the motif identifiability, how to robustly identify the underlying motifs using neural networks, and also an alternative form of convolutional kernels for motif discovery.

1.2 Scope of the Thesis

Our definition about the neural network interpretability will be given in Section 3.1, which is mainly about how to explain the inner workings of a *trained* neural network. However, the terminology interpretability is also commonly used in a much broader field of understanding the general neural network methodology. For example, the empirical success of DNNs raises many unsolved questions to theoreticians (Vidal et al., 2017). What are the merits (or inductive bias) of DNN architectures (Bruna and Mallat, 2013; Eldan and Shamir, 2016)? What are the properties of DNNs’ loss surface/critical points (Nouiehed and Razaviyayn, 2018; Yun et al., 2018; Choromanska et al., 2015; Haeffele and Vidal, 2017)? Why DNNs generalises so well with just simple regularisation (Srivastava et al., 2014; Mianjy et al., 2018; Salehinejad and Valaee, 2019)? What about DNNs’ robustness/stability (Sengupta and Friston, 2018; Zheng, Song, et al., 2016; Haber and Ruthotto, 2017; Chang et al., 2018; Thekumparampil et al., 2018; Creswell and Bharath, 2018)? There are also studies about how to generate adversarial examples (Konda Reddy Mopuri and Radhakrishnan, 2017; Mopuri et al., 2018) and detect adversarial inputs (Zheng and Hong, 2018). These are the interpretability of deep learning methodology (or deep learning theory) and are not covered by this thesis.

1.3 Research Questions and Contributions of the Thesis

There are two main research topics in this thesis:

Neural network interpretability. The interpretability of neural networks has drawn more and more attention of the deep learning research community in recent years. Many methods have been proposed intending to open the black-box of neural networks. However, there is still no clear and systematic overview of all these different interpretability studies, i.e., a taxonomy. In the context of this thesis, interpretability is the important first step of

using deep neural networks for scientific research, which is about how to extract “knowledge” from a trained accurate deep neural network. This topic will be covered in Section 3.

Motif identifiability in neural network based motif discovery. By interpreting a trained neural network (e.g., trained to predict DNA or RNA sequence specificities), researchers are able to collect some potential motifs, as a common practice, from the first-layer convolutional kernels. However, how do we know whether the interpreted motifs uniquely recover the true motifs underlying the data? It is possible for two neural networks to have the same test performance, yet have different inner weights (including the convolutional kernels ones). The identifiability issue cares about whether it is theoretical possible to identify the true values of the parameters (motifs in our case) of a model. Specifically, we ask three research questions under this topic:

- Does this motif identifiability issue exists? As the model performance is the only criterion we have, is it a good indicator of the motif identifiability? (Chapter 4)
- How can we deal with the motif identifiability issue, i.e., robustly recover the motifs underlying the data? (Chapter 5)
- Can we improve the motif representation/interpretation of the current neural network based motif discovery method? (Chapter 6)

The main contributions of the thesis are summarised as follows:

- A comprehensive review of the neural network interpretability research is conducted. We provide an extended definition of interpretability and summarise the importance of interpretability into three groups. More importantly, a novel taxonomy of neural network interpretability is proposed, which is organised along three dimensions: type of engagement (passive vs. active interpretation approaches), the type of explanation, and the focus (from local to global interpretability). This taxonomy provides a meaningful

3D view of distribution of papers from the relevant literature as two of the dimensions are not simply categorical but allow ordinal subcategories. This work, while being first time available online for exact one year, has been showing its impact in the research field².

- We propose a sequence data generator (Section 4.2.2), which has flexible and controllable “ground truth” motifs and class structures (classification rules). This is important as normally we do not have full knowledge about the underlying motifs in real-world sequence datasets, so there may not be a fair way to compare the motif identifiability.
- The motif identifiability problem in neural network based motif discovery is systematically investigated (in other words, can the neural network tell us the true motifs underlying the data?). We propose a novel similarity measure between probabilistic motifs in the context of convolutional kernels, which takes into account the possible shift between two motifs and explicitly handles the “wildcard” distribution. Together with above synthetic datasets, we find that although high levels of motif identifiability usually imply high levels of model performance, the reverse is not necessarily true, i.e., high levels of performance do not necessarily mean high levels of motif identifiability. But in common practice, model performance is often taken as the only indicator of suitability of the discovered motifs.
- We thus propose a robust (semi-automatic) motif discovery workflow to enhance reliability of extracted motifs from trained neural networks, which includes three steps: (1) the determination of number of convolutional kernels, (2) training multiple independent neural networks on re-sampled data, and (3) hierarchical clustering of the learnt kernels, and generation of representative motifs. Experiments on both synthetic and real-world datasets show that it can help find potential motifs reliably.
- We explore an alternative way (i.e., probabilistic kernels) to conventional convolutional

²By 23 December 2021, it has been cited 40 times according to the Google Scholar.

kernels in order to learn the motifs directly in the neural network training (instead of post hoc interpretation). Although experiments show that convolutional kernels have their irreplaceable merits, the advantages and disadvantages of probabilistic kernels are analysed.

1.4 Publications Resulting from the Thesis

The work resulting from this thesis has been presented in the following published or submitted papers.

Refereed or Submitted Journal Papers

- [1] **Yu Zhang**, Peter Tiño, Aleš Leonardis, and Ke Tang. “A Survey on Neural Network Interpretability”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 5, pp. 726–742, 2021. ([Zhang, Tiño, et al., 2021](#))
- [2] **Yu Zhang**, Ildem Akerman, Peter Tiño. “Neural Network Based Motif Discovery as an Identifiability Problem”. To submit to *PLOS Computational Biology*.

Paper in Preparation

- [3] **Yu Zhang**, Peter Tiño, Aleš Leonardis. “Direct Learning of Motifs in Neural Networks with Probabilistic Kernels”. Prepared as a conference paper.

The first work (neural network interpretability) is presented in Chapter 3. Chapter 4 and Chapter 5 contain more details of the two topics of the Work 2, relationship between model performance and motif identifiability, and a proposed robust motif discovery workflow. Chapter 6 contains the main content of the Work 3 (probabilistic kernels).

CHAPTER 2

Background

This chapter introduces the general background of the thesis. Section 2.1 briefly describe the techniques used in deep learning, including network architectures, optimisation etc. Section 2.2 summarises the importance of neural network interpretability into three groups. And in Section 2.3, the problem of genetic motif discovery is introduced, including motif representations, classic motif discovery approaches and neural network based approaches. Then the model (motif) identifiability issue of neural networks is presented in Section 2.4. Finally, Section 2.5 summarises this chapter.

2.1 Deep Neural Networks

2.1.1 Feedforward Neural Networks

Feedforward neural networks, also known as multilayer perceptrons (MLPs), are the basic forms of deep learning models. **Neural networks** are computational models which were loosely inspired by the biological neural networks in the human brains ([McCulloch and Pitts, 1943](#)). A neural network consists of a collection of “neurons” (usually connected in a

hierarchical structure), where each neuron operates as in this formula

$$y = h(\mathbf{w}^\top \mathbf{x} + b), \quad (2.1)$$

where y is a scalar output, \mathbf{x} is the inputs from other neurons represented as a vector, h is a nonlinear activation function (e.g., a logistic sigmoid function or hyperbolic tangent function), \mathbf{w} collects the weights of all incoming connections and b is a bias term which can be interpreted as a negated threshold. (Note that this “bias” has nothing to do with the statistical bias, which is the expected derivation between an estimator’s estimation and the true value.) The term **feedforward** requires the neural network topology to have no closed directed loops or cycles. The other kind of neural networks, recurrent neural networks (RNNs, [Rumelhart et al., 1986](#)), contains connections that feed the outputs back to the inputs and thus is able to model temporal dynamic behaviour. Long short-term memory (LSTM) networks and gated recurrent unit (GRU) networks are two of the most common recurrent neural networks (RNNs). In this thesis, we mainly focus on feedforward neural networks.

A neural network is typically composed of a series of layers. The first and last layers are called input layer and output layer respectively, while the layers in between are called hidden layers. Except for the input layer which is only used for taking input, each layer performs an affine transformation (i.e., a linear transformation \mathbf{W} and a translation/shift \mathbf{b}) followed by a nonlinear activation function as mentioned above (the output layer may use a different function, such as softmax, in order to get a probability-like output). Formally, a layer computes such a function

$$f^{(i)}(\mathbf{x}) = h^{(i)}(\mathbf{W}^{(i)\top} \mathbf{x} + \mathbf{b}^{(i)}), \quad (2.2)$$

where the superscript i denotes the layer index. The whole neural network (d layers) can be represented as a composite function

$$y = f(\mathbf{x}) = f^{(d)} \dots (f^{(2)}(f^{(1)}(\mathbf{x}))), \quad (2.3)$$

where output y is a scalar for regression tasks, and it can be a vector \mathbf{y} for multi-class classification tasks. The number of layers is the **depth** of the network, and the **width** of a network is determined by the dimensionality of the output vectors of hidden layers. Previous studies (universal approximation theorems) have shown that a one-hidden-layer neural network (with non-polynomial activation function) can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy, as long as it has a sufficiently large number of hidden neurons, i.e., wide enough (Cybenko, 1989; Hornik et al., 1989; Funahashi, 1989; Leshno et al., 1993; Barron, 1994). Latest work has also found that depth can be exponentially more valuable than width for standard feedforward neural networks in terms of the expressive power (Eldan and Shamir, 2016) (specifically, they found there are two-hidden-layer networks of width polynomial in w which cannot be arbitrarily well approximated by one-hidden-layer networks, unless their width is exponential in w). This shows the importance of having depth in deep learning models. Above universal approximation theorems imply there exist a large network that can be arbitrarily accurate, however, they do not tell us how to get the model parameters.

2.1.2 Optimisation

Neural network optimisation (i.e., training) is to find a set of parameters $\boldsymbol{\theta}$ of a neural network, which can fit the training data well and is also expected to have good predictive performance on unseen data (measured by a loss function, see below). It is one of the most important problems in the neural network research and applications. As neural networks become more and more complex, their training process may require days to months of time and huge computing resources. To date, one of the largest neural network models is GPT-3 (Generative Pre-trained Transformer, Brown et al., 2020). It has 175 billion parameters and requires 3×10^{23} FLOPS (floating-point operation) during pre-training, which would take more than 300 years on an NVIDIA V100 GPU. For the work presented in this thesis, most of the computations (neural network training) were performed using the University of

Birmingham’s BlueBEAR High Performance Computing service (thanks!).

Loss function in supervised learning measures the difference between the predicted values and the true values, which is to be minimised in the network training process. In regression tasks (predicting a numerical value), the most common loss function is the mean squared error (MSE). For a set of n data points, each of which has the true value $y^{(i)}$ and the prediction $\hat{y}^{(i)}$, the MSE of \hat{y} on this sample is

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2. \quad (2.4)$$

For classification tasks, we can measure the model’s **accuracy** or its opposite, the **error rate**. The error rate can be represented by the 0-1 loss function,

$$\text{0-1 Loss} = \frac{1}{n} \sum_{i=1}^n \mathbb{I}[y^{(i)} \neq \hat{y}^{(i)}], \quad (2.5)$$

where $\mathbb{I}[\text{condition}]$ is an indicator function which outputs 1 if condition (i.e., $y^{(i)} \neq \hat{y}^{(i)}$) is true and 0 otherwise. Another commonly used loss function is cross-entropy,

$$H(P \parallel Q) = -\mathbb{E}_{x \sim P} \log Q(x). \quad (2.6)$$

When used as loss function, $P(y \mid \mathbf{x})$ is the target probability distribution of the class label (given an input vector \mathbf{x}), which is 1 for the true class and 0 for other classes, $Q(\hat{y} \mid \mathbf{x})$ is the predicted class probability distribution. Its discrete form on a set of n data points is as follows,

$$\text{CrossEntropy} = - \sum_i^n P(y \mid \mathbf{x}^{(i)}) \log Q(\hat{y} \mid \mathbf{x}^{(i)}). \quad (2.7)$$

Using above loss functions, the goal of a machine learning algorithm, which is to minimise the generalisation error, can be written as

$$E(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[L(f(\mathbf{x}; \boldsymbol{\theta}), y) \right], \quad (2.8)$$

where \mathcal{D} is the underlying joint distribution of the data, L is the loss function. However,

in practice we almost never know the true data distribution \mathcal{D} but only have access to a training set D where each data point is assumed to be sampled from \mathcal{D} independently (i.i.d. assumption). As a reasonable approximation, a common way is to minimise the expected loss on the training set

$$\hat{E}(\boldsymbol{\theta}) = \frac{1}{n} \sum_i^n L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}). \quad (2.9)$$

This is called **empirical risk minimisation** (ERM). However, as the model is optimised on a finite training set, it is likely (if it has high capacity¹) to learn the particularities of the training set and generalise poorly to unseen data (i.e., overfitting). So in practice we need to balance the model's complexity and its fitness on training data. This inductive principle is called **structural risk minimisation** (SRM, [Vapnik, 1999](#)). There are various ways to implement the SRM principle, which are also commonly referred as regularisation. One of the most typical approaches is adding a parameter norm penalty $\Omega(\boldsymbol{\theta})$ to the optimisation objective function,

$$\tilde{E}(\boldsymbol{\theta}) = \frac{1}{n} \sum_i^n L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}) + \lambda \Omega(\boldsymbol{\theta}), \quad (2.10)$$

where $\lambda \in [0, \infty)$ is a hyperparameter that controls the relative importance of the regularisation term. A common choice for the Ω is the L^2 norm, $\Omega(\boldsymbol{\theta}) = \frac{1}{2} \|\boldsymbol{\theta}\|_2^2$, which is also known as **weight decay**. Another option is the L^1 norm, $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 = \sum_i |\theta_i|$. Compared with L^2 regularisation, L^1 regularisation tends to push more θ_i to 0, in other words, make the $\boldsymbol{\theta}$ more sparse. This sparsity property is also widely used for feature selection.

Gradient descent ([Cauchy et al., 1847](#)) is the most common technique for minimizing above loss functions. It is an iterative optimisation method, while in each step the parameters are updated with

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \cdot \nabla E(\boldsymbol{\theta}_t), \quad (2.11)$$

at time step t , where η is the **learning rate** and $\nabla E(\boldsymbol{\theta})$ is the gradient, collecting all the partial derivatives $\frac{\partial E}{\partial \theta_i}$. As the $-\nabla E(\boldsymbol{\theta})$ represents the direction in which $E(\boldsymbol{\theta})$ decreases

¹It can be viewed as the ability to fit a wide variety of functions

fastest, gradient descent is also often called steepest descent. Take ERM as an example, in each update we need to calculate the gradients $\nabla \hat{E}(\boldsymbol{\theta}) = \frac{1}{n} \sum_i^n \nabla L(\boldsymbol{\theta})|_{(\mathbf{x}^{(i)}, y^{(i)})}$ on the whole training set. This is called **batch** gradient descent. The drawback is obvious, it can be very slow for big dataset and requires a huge amount of memory during training, which can easily run out of computing resources (especially on GPU). In practice, the gradients are usually calculated with a small number of randomly sampled training examples as an approximation. This method is called **minibatch stochastic** gradient descent (SGD) and the number of training examples in a minibatch is called batch size. It is common to set the batch size to a power of 2 (e.g., 32) considering the potential efficiency for the hardware. Training a neural network usually requires many times of going through the training data. In this context, an **epoch** means one complete pass of the training set.

A remaining challenge is how to choose the learning rate η . If the learning rate is too large, the training error may keep oscillate or even inadvertently increase. While the learning rate is too small, the training process is very slow and may be stuck with a local minimum. Learning rate can be changed by a manually configured scheduler, for example, reducing the learning rate by a rate if the validation performance stops improving for a certain number of epochs. Many approaches have been proposed to change the learning rate adaptively. AdaGrad (Adaptive Gradient, [Duchi et al., 2011](#)) changes the learning rate η for every parameter θ_i based on its history, performing smaller updates for frequently updated parameters and larger updates for less updated parameters. Instead of including all the gradient history, Adadelta ([Zeiler, 2012](#)) calculates an exponential decaying average of the past gradients. Adam ([Kingma and Ba, 2015](#)), which has been very popular, not only updates the learning rate like Adadelta, but also do the same thing for momentum. Momentum ([Qian, 1999](#); [Nesterov, 1983](#)) is a technique to accelerate the network training, which increases the gradient updates that align with the past updates and reduces those that change directions. However, there is no one best choice. While Adam is preferred for some models (e.g., Transformer-based networks ([Vaswani et al., 2017](#)) in NLP; generative adversarial networks), vanilla SGD (or with momentum) usually performs well in many

computer vision tasks ([Wilson et al., 2017](#)).

2.1.3 Convolution and Convolutional Neural Networks

In general, convolution is a mathematical operation on two functions g and h ,

$$(g * h)(t) = \int_{-\infty}^{\infty} g(\tau) \cdot h(t - \tau) d\tau. \quad (2.12)$$

Convolution $g * h$ characterises the similarity of g and h (flipped and offset by t). The former is usually viewed as **input** and the latter as the **kernel** in convolutional neural networks. The convolution output is sometimes called **feature map**. The discrete form of convolution is

$$(g * h)(t) = \sum_{\tau=-\infty}^{\infty} g(\tau) \cdot h(t - \tau). \quad (2.13)$$

In practice, most of the neural networks implement another operation **cross-correlation**

$$\begin{aligned} (g * h)(t) &= \sum_{\tau=-\infty}^{\infty} g(\tau) \cdot h(\tau - t) \\ &= \sum_{\tau=-\infty}^{\infty} g(t + \tau) \cdot h(\tau). \end{aligned} \quad (2.14)$$

which has the same effect of convolution but without flipping h ([Goodfellow, Bengio, et al., 2016](#)). In the context of deep learning, the kernel h will be learned from the training data, so flipping the kernel or not does not matter. Unless otherwise stated, we refer to Equation (2.14) when talking about convolution throughout the thesis. Considering a one-dimensional input sequence S of length L_S and a convolutional kernel K of length L_K (indexed by j , 1-based), the convolution output of S and K is as below, indexed by i (1-based):

$$(S * K)(i) = \sum_{j=1}^{L_K} S(i + j) \cdot K(j), \quad i \in \{1, 2, \dots, L_S - L_K + 1\}. \quad (2.15)$$

If the input sequence S is represented with one-hot encoding (with C channels), above convolution becomes

$$(S * K)(i) = \sum_{j=1}^{L_K} \sum_{c=1}^C S(i+j, c) \cdot K(j, c), \quad i \in \{1, 2, \dots, L_S - L_K + 1\}, \quad (2.16)$$

while $C = 4$ for DNA or RNA sequences. One-hot encoding means at each position of S , there is only one channel is 1 and all the other channels are 0. Kernel K is represented as an $L_K \times 4$ matrix of free real-valued weights (parameters).

Convolutional neural networks (CNNs) (LeCun, Boser, et al., 1989) are a special kind of neural networks that have at least one convolutional layer (replacing the linear transformation \mathbf{W} with convolution operation). The employment of convolutional kernels (aka filters) brings several advantages: First, it largely reduces the number of parameters comparing to traditional fully-connected layers. It does not learn different parameters for every position, but learn a set of kernels applied among all positions. More importantly, with the help of convolutional kernels, convolutional networks are able to capture local patterns that are invariant to position (equivariance to translation).

Another typical layer in convolutional neural networks is **pooling** layer. A pooling function summarises a certain range or neighbourhood of the inputs with, for example, the maximum value (max pooling, Zhou and Chellappa, 1988) or the average (average pooling). By using pooling, the network discards some information from the inputs but instead gets a certain level of invariance of small translations to the inputs. (We will see in later chapters that different choices of the pooling operation represents different assumptions in neural network based motif discovery.) A typical convolutional neural network architecture consists of convolutional layers, followed by pooling layers and then fully-connected layers. The convolutional layers are sometimes viewed as feature extractors or detectors, which is why they are also used for motif discovery in bioinformatics.

2.2 Importance of Interpretability

DNNs are often criticised for the black-box property as they hide their decision logic behind numerous real-valued weights and layer-by-layer transformations. The need for interpretability² has already been stressed by many papers (Lipton, 2018; Doshi-Velez and Kim, 2017; Guidotti et al., 2018), emphasizing cases where lack of interpretability may be harmful. However, a clearly organised exposition of such argumentation is missing. We summarise the arguments for the importance of interpretability into three groups.

2.2.1 High Reliability Requirement

Although deep networks have shown great performance on some relatively large test sets, the real world environment is still much more complex. As some unexpected failures are inevitable, we need some means of making sure we are still in control. Deep neural networks do not provide such an option. In practice, they have often been observed to have unexpected performance drop in certain situations, not to mention the potential attacks from the adversarial examples (Papernot et al., 2016; Moosavi-Dezfooli et al., 2017).

Interpretability is not always needed but it is important for some prediction systems that are required to be highly reliable because an error may cause catastrophic results (e.g., human lives, heavy financial loss). Interpretability can make potential failures easier to detect (with the help of domain knowledge), avoiding severe consequences. Moreover, it can help engineers pinpoint the root cause and provide a fix accordingly. Interpretability does not make a model more reliable or its performance better, but it is an important part of formulation of a highly reliable system.

²There is also a related concept *explainability*. Please see Section 3.1 for more details.

2.2.2 Ethical and Legal Requirement

A first requirement is to avoid algorithmic discrimination. Due to the nature of machine learning techniques, a trained deep neural network may inherit the bias in the training set, which is sometimes hard to notice. There is a concern of fairness when DNNs are used in our daily life, for instance, mortgage qualification, credit and insurance risk assessments.

Deep neural networks have also been used for new drug discovery and design ([Gawehn et al., 2016](#)). The computational drug design field was dominated by conventional machine learning methods such as random forests and generalised additive models, partially because of their efficient learning algorithms at that time, and also because a domain chemical interpretation is possible. Interpretability is also needed for a new drug to get approved by the regulator, such as the Food and Drug Administration (FDA). Besides the clinical test results, the biological mechanism underpinning the results is usually required. The same goes for medical devices.

Another legal requirement of interpretability is the “right to explanation” ([Goodman and Flaxman, 2017](#)). According to the EU General Data Protection Regulation (GDPR) ([European Parliament, Council of the European Union, 2016](#)), Article 22, people have the right not to be subject to an automated decision which would produce legal effects or similar significant effects concerning him or her. The data controller shall safeguard the data owner’s right to obtain human intervention, to express his or her point of view and to contest the decision. If we have no idea how the network makes a decision, there is no way to ensure these rights.

2.2.3 Scientific Usage

Deep neural networks are becoming powerful tools in scientific research fields where the data may have complex intrinsic patterns, for example, genomics ([Park and Kellis, 2015](#)), astronomy ([Parks et al., 2018](#)), physics ([Baldi et al., 2014](#)) and even social science ([Hofman et al., 2017](#)). The word “science” is derived from the Latin word “scientia”, which means

“knowledge”. When deep networks reach a better performance than the old models, they must have found some unknown “knowledge”. Interpretability is a way to reveal it.

2.3 Motif Discovery

A **motif** is a short (usually 6–15) recurring sequence pattern in biological sequences (DNA, RNA or protein). Those motifs might be related to certain regulatory functions or biological processes, for example, alternative splicing, transcription, and translation (Gerstberger et al., 2014; Xiong et al., 2015). Therefore, motif discovery (or motif mining) is of particular importance in computational biology. Over the past decades, numerous algorithms were developed for motif discovery. Those methods can be divided into different groups, for example, enumerative approaches, probabilistic approaches, etc. (Hashim et al., 2019; He, Shen, et al., 2021). In recent years, deep learning has also been applied to this field (Alipanahi et al., 2015; Zhou and Troyanskaya, 2015). In this thesis, we are interested in the latter which makes use of DNNs and thus divide the motif discovery methods into two categories: (1) the classic statistical enrichment based methods, and (2) the discriminative power based methods, typically represented by neural networks. Before introducing them in detail, we first describe how a motif is represented.

2.3.1 Motif Representation

Consensus sequence. As a straightforward method, a motif can be written as a simple sequence. For example, a common restriction enzyme *EcoRI* cuts the DNA at restriction site GAATTC. In order to allow ambiguity, the incompletely specified nucleic acids can be represented with the International Union of Pure and Applied Chemistry (IUPAC) code (Cornish-Bowden, 1985), for example, GTYRAC, where Y means C or T (pYrimidine) and R means A or G (puRine). In general, the concept of consensus sequence refers to a sequence that matches the symbols approximately, i.e., allowing a certain number of mismatches (Stormo, 2000). Day and McMorris (1992) compared 9 methods of constructing consensus sequences

and made recommendations on how to choose appropriate consensus methods.

Regular expression. A regular expression is a sequence that specifies a search pattern, which is a technique developed in theoretical computer science and formal language theory. There are different notations for describing motifs but most of them follow these conventions:

- A normal symbol (in a given alphabet set) means an exact match.
- A pair of square brackets means any of the symbol inside it, e.g., [AT] means A or T.
- A pair of curly brackets means any symbol except for the inside, e.g., {A} means any nucleotide except for A.
- The quantifier ‘?’ means zero or one occurrence of the preceding symbol.
- X or N or ‘.’ means any symbol.

The PROSITE notation ([Sigrist et al., 2002](#)) is an example of this kind. Similar to consensus sequences, using regular expressions to describe motifs still loses some information. For example, [AT] means A and T are both possible, but does not give the probability of A or T occurring in the pattern.

Position weight matrix. Position weight matrix (PWM) or position-specific scoring matrix (PSSM) is one of the most widely used ways to represent motifs ([Stormo et al., 1982](#)). A PWM is essentially a matrix of numbers containing weights (or scores) for each nucleotide at each position (e.g., an $\ell \times n$ matrix where ℓ is the length of the motif and $n = 4$ is the number of nucleotides, see Figure 2.1). It can be used to scan new sequences and produce a match score. Another type of weight matrices is position frequency matrix (PFM), which contains the position-dependent frequency of each nucleotide. PFMs can be obtained from wet lab experiments or computationally discovered by tools such as MEME ([Bailey, Boden, et al., 2009](#)). Position probability matrix (PPM) is also often used, especially for visualisation.

	A	C	G	T
−38	−38	−15	−13	17
−32	19	−38	−48	−32
−26	1	−8	−6	8
−20	12	−10	−7	−9
−14	10	−3	−10	−6
−8	−48	−32	−48	19

Figure 2.1: A PWM for -10 region of *E.coli* promoters (Stormo et al., 1982). The score for the consensus sequence TATAAT is the sum of the shaded elements, 85. Other sequences that are different from it will have lower scores.

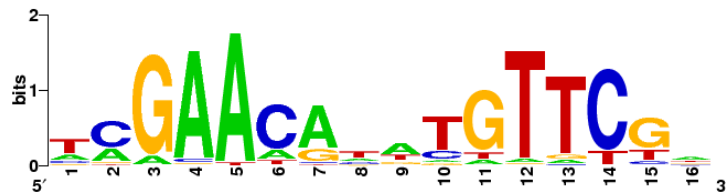


Figure 2.2: The sequence logo of the LexA-binding motif of Gram-positive bacteria. (source: Wikipedia)

Visualisation. Motifs (PPMs) can be visualised by stacking the letters on each position with their probabilities as the height (e.g., Figure 2.2). To emphasise positions that contain more “information” about the nucleotide distribution, the height is usually scaled by a position dependent factor, information content, which is

$$R_i = \log_2 4 - (H_i + e) \quad (2.17)$$

where H_i is the Shannon entropy of the nucleotide distribution at position i , and e is a correction factor only required if the motif is obtained from a few sequences. The information content R_i measures the information in bits. This representation is called sequence logos (Schneider and Stephens, 1990). In this thesis, all the motif visualisations are created with a Python module Logomaker (Tareen and Kinney, 2019).

2.3.2 Classic Motif Discovery Approaches

Traditional motif discovery approaches can be roughly divided into two groups: enumeration approaches and probabilistic approaches. Here we only introduce their basic ideas and more details can be found in the review paper ([Hashim et al., 2019](#)).

Enumeration approach. Enumeration approaches are used to search for consensus sequences or motifs in regular expression forms, which exhaustively search the whole motif space and evaluate the motifs with certain statistical metrics, for example, z-score (YMF, [Sinha and Tompa, 2003](#)) or the significance of Fisher’s exact test (DREME, [Bailey, 2011](#)). The search process requires exponential time in terms of the motif length, which can be accelerated by using special data structures (e.g., suffix trees) or parallel processing ([Yu et al., 2015](#)).

Probabilistic approach. A probabilistic approach represents the motifs as PWMs (or equivalent forms, which specify the distribution of nucleotides at each position) and constructs a generative model of the sequence data. The motifs are optimised in order to maximise the likelihood of the observed data. Many probabilistic approaches are based on the expectation-maximisation (EM) algorithm ([Lawrence and Reilly, 1990](#); [Bailey and Elkan, 1994](#)). Given a group of biological sequences where the locations of the motif are unknown, the probabilistic approaches start with an initial PWM, estimate the location of the motif on each sequence, and update the PWM with the subsequences on the estimated locations. This process is repeated until convergence or reaching the maximum number of iterations. There are also methods based on the Gibbs sampling, such as AlignACE ([Hughes et al., 2000](#)) and its improved version BioProspector ([Liu et al., 2001](#)).

2.3.3 Neural Network Based Motif Discovery

Many deep learning techniques have been applied to bioinformatics. They can learn from a large quantity of data (even millions of sequences) and compete favourably with the state of the art ([Alipanahi et al., 2015](#)) on, for example, the task of predicting sequence specificities. Different from the classic motif discovery methods introduced above, neural network based methods first train a high-accuracy classifier (e.g., between target sequences and background sequences). And then the motifs are interpreted from the trained model (e.g., from the convolutional kernels). We now introduce a few typical methods in this category and more information can be found in this survey paper ([He, Shen, et al., 2021](#)).

DeepBind ([Alipanahi et al., 2015](#)) is an early work that applies deep learning to the task of predicting sequence specificities of DNA- and RNA-binding proteins. It employs a basic CNN architecture which has a single convolutional layer, followed by ReLU activation and a global maximum and average pooling layer, and then fully-connected layers for classification. The motifs are interpreted from the trained convolutional kernels (see Figure 4.1 for an illustration). Another example of the CNN based models is DeepSEA ([Zhou and Troyanskaya, 2015](#)) which uses more convolutional layers and is trained as a multi-task model.

Researchers also tried to add recurrent structures to the network. DanQ ([Quang and Xie, 2016](#)) uses a hybrid architecture combining CNNs and bidirectional LSTM networks. BiRen ([Yang, Liu, et al., 2017](#)) has the similar architecture but uses bidirectional GRU networks instead. Experiments have shown that although adding recurrent network architectures can slightly improve the model performance, it will result in uninformative (hard-to-interpret) motifs ([Trabelsi et al., 2019](#)). This is exact the case where we cannot focus only on the performance but also need to consider the model interpretability. In this thesis we only focus on CNN based models. Note that the proposed methods in our study can be easily extended to other deep learning models, and we will discuss it in Chapter 7.

2.4 Identifiability Issue

Identifiability cares about whether it is theoretically possible to identify the true values of the parameters of a model. In general, there are two types of identifiability issues (Walter and Pronzato, 1997; Transtrum et al., 2014; Anstett-Collin et al., 2020): *structural* identifiability and *practical* identifiability. A model is (globally) structurally identifiable if all of its parameters can be identified from an *infinite* number of *noise-free* data, i.e.,

$$f(x; \boldsymbol{\theta}) = f(x; \boldsymbol{\theta}') \implies \boldsymbol{\theta} = \boldsymbol{\theta}' \quad (\boldsymbol{\theta}, \boldsymbol{\theta}' \in \boldsymbol{\Theta}), \quad (2.18)$$

where f is a model parametrised by $\boldsymbol{\theta}$, $\boldsymbol{\Theta}$ is the parameter space. A model is called locally structurally identifiable if the parameters are only required to be unique within their neighbourhood rather than the entire $\boldsymbol{\theta}$. Consider a linear model

$$f(x; \boldsymbol{\theta}) = (\theta_1 \theta_2) x. \quad (2.19)$$

It is not structurally identifiable as there are an infinite number of combinations of θ_1 and θ_2 that results in the same f . Another example is

$$f(x; \boldsymbol{\theta}) = e^{\theta_1 x} + e^{\theta_2 x}. \quad (2.20)$$

This model is also structurally unidentifiable as it is invariant to permutation of the parameters. Being structurally unidentifiable means there is no hope that we can uniquely identify a model's parameters from data. However, even if a model is structurally identifiable, it may not be practical or tractable to do that (e.g., requiring an impractical number of observations). The definition of practical identifiability is somewhat vague in literature (Raue et al., 2009; Villaverde et al., 2016), which mainly refers to whether it is possible to learn all the parameters from *finite* or *noisy* data.

Deep neural networks, which are often severely over-parametrised³ in practice, are very

³According to the classic bias-variance trade-off, a model should balance underfitting (which means it is not expressive enough and cannot capture the underlying structure of the data, imagining fitting a set

likely to be structurally unidentifiable. For example, one can permute the incoming connections of a hidden layer (just like in Equation 2.20) and do the same for its outgoing connections, while the neural network remains unchanged. This is known as weight space symmetry (Goodfellow, Bengio, et al., 2016). Besides, the use of ReLU activation function can also cause non-identifiability. This, together with the limited interpretation of neural networks, raises a concern about the commonly used neural network based motif discovery framework: Even if we get a well-performing neural network f_{θ} , how do we know whether the interpreted motifs (from θ) uniquely recover the “true” motifs underlying the data? In this thesis, we study the neural network based motif discovery as a motif identifiability problem in Chapter 4.

2.5 Chapter Summary

This chapter provides the background knowledge of this thesis: (1) an overview of the deep learning techniques, (2) the interpretability issue of neural networks, (3) motif discovery and the applications of neural networks, and (4) the motif identifiability issue in neural network based motif discovery. We will further study the interpretability and identifiability issues in the later sections of the thesis.

of data points sampled from a second order polynomial (U-shape) with a first order, i.e., linear model) and overfitting (which means it is too expressive and is prone to memorise the specific characteristic of a certain training set). The number of trainable parameters of a model can be viewed as a rough indicator of its complexity/expressive power. For deep neural networks, the number of parameters is usually huge and sometimes much larger than the number of the training data points. It is still an open question why deep neural networks exhibit remarkable generalization ability while having a massive number of trainable parameters (Zhang, Bengio, et al., 2021; Belkin et al., 2019).

CHAPTER 3

A Novel 3D Taxonomy of Neural Network Interpretability

In Chapter 2.2 we have seen why neural network interpretability is important. We first conduct a comprehensive review of the current literature and propose a novel taxonomy. The rest of the chapter is organised as follows. Our (extended) definition of interpretability is given in Section 3.1. Then in Section 3.2 we show our contributions and differences from previous work. In Section 3.3, we introduce the proposed taxonomy for network interpretation methods. The taxonomy consists of three dimensions, passive vs. active methods, type of explanations and global vs. local interpretability. Along the first dimension, we divide the methods into two groups, passive methods (Section 3.4) and active methods (Section 3.5). Under each section, we traverse the remaining two dimensions (different kinds of explanations, and whether they are local, semi-local or global). Section 3.6 gives a brief summary of the evaluation of interpretability. And in Section 3.7 we discuss the advantages and disadvantages of interpretation methods in different categories. Finally, we conclude this chapter in Section 3.8.

This chapter has been previously published in the Work 1 ([Zhang, Tiño, et al., 2021](#)).

3.1 An (Extended) Definition of Interpretability

To open the black-boxes of deep networks, many researchers started to focus on the model interpretability. Although this theme has been explored in various papers, no clear consensus on the definition of interpretability has been reached. Most previous works skimmed over the clarification issue and left it as “you will know it when you see it”. If we take a closer look, the suggested definitions and motivations for interpretability are often different or even discordant (Lipton, 2018).

One previous definition of interpretability is *the ability to provide explanations in understandable terms to a human* (Doshi-Velez and Kim, 2017), while the term *explanation* itself is still elusive. After reviewing previous literature, we make further clarifications of “explanations” and “understandable terms” on the basis of (Doshi-Velez and Kim, 2017).

Interpretability is the ability to provide *explanations*¹ in *understandable terms*² to a human.

where

1. **Explanations**, ideally, should be *logical decision rules* (if-then rules) or can be transformed to logical rules. However, people usually do not require explanations to be explicitly in a rule form (but only some key elements which can be used to construct explanations).
2. **Understandable terms** should be from the *domain knowledge* related to the task (or common knowledge according to the task).

Our definition enables new perspectives on the interpretability research: **(1)** We highlight *the form of explanations* rather than particular *explanators*. After all, explanations are expressed in a certain “language”, be it natural language, logic rules or something else. Recently, a strong preference has been expressed for the language of explanations to be as close as possible to logic (Pedreschi et al., 2019). In practice, people do not always require a

Table 3.1: Some interpretable “terms” used in practice.

Field	Raw input	Understandable terms
Computer vision	Images (pixels)	Super pixels (image patches) ¹ Visual concepts ²
NLP	Word embeddings	Words
Bioinformatics	Sequences	Motifs (position weight matrix) ³

¹ image patches are usually used in attribution methods (Ribeiro et al., 2016).

² colours, materials, textures, parts, objects and scenes (Bau, Zhou, et al., 2017).

³ proposed by Stormo et al. (1982) and became an essential tool for computational motif discovery.

full “sentence”, which allows various kinds of explanations (rules, saliency masks etc.). This is an important angle to categorise the approaches in the existing literature. **(2)** Domain knowledge is the basic unit in the construction of explanations. As deep learning has shown its ability to process data in the raw form, it becomes harder for people to interpret the model with its original input representation. With more domain knowledge, we can get more understandable representations that can be evaluated by domain experts. Table 3.1 lists several commonly used representations in different tasks.

We note that some studies distinguish between *interpretability* and *explainability* (or *understandability*, *comprehensibility*, *transparency*, *human-simulatability* etc., Lipton, 2018; Arrieta et al., 2020). Rudin (2019) and Marcinkevičs and Vogt (2020) argue that interpretable machine learning is about designing models that are inherently interpretable, while explainable machine learning is about providing post hoc explanations for existing black-box models. We note that they are both covered in our proposed taxonomy (respectively *active interpretability intervention* and *passive interpretation*). In this thesis we do not emphasise the elusive differences among those terms. As defined above, we see explanations as the core of interpretability and use interpretability, explainability and understandability interchangeably. Specifically, we focus on the interpretability of (deep) *neural networks* (rarely

recurrent neural networks), which aims to provide explanations of their inner workings and input-output mappings. There are also some interpretability studies about the generative adversarial networks (GANs). However, as a kind of generative models, it is quite different from common neural networks used as discriminative models. For this topic, we would like to refer readers to the latest work (Bau, Zhu, et al., 2019; Yang, Shen, et al., 2021; Voynov and Babenko, 2020; Plumerault et al., 2020; Härkönen et al., 2020; Kahng et al., 2018), many of which share the similar ideas with the “hidden semantics” part of our taxonomy (see Section 3.3), trying to interpret the meaning of hidden neurons or the latent space.

Under our definition, the source code of Linux operating system is interpretable although it might be overwhelming for a developer. A deep decision tree or a high-dimensional linear model (on top of interpretable input representations) are also interpretable. One may argue that they are not simulatable (Lipton, 2018) (i.e. a human is able to simulate the model’s processing from input to output in his/her mind in a short time). We claim, however, they are still interpretable.

3.2 Previous Work

There have already been attempts to summarise the techniques for neural network interpretability. However, most of them only provide basic categorisation or enumeration, without a clear taxonomy. Lipton (2018) points out that the term interpretability is not well-defined and often has different meanings in different studies. He then provides simple categorisation of both the need (e.g., trust, causality, fair decision-making etc.) and methods (post-hoc explanations) in interpretability study. Doshi-Velez and Kim (2017) provide a discussion on the definition and evaluation of interpretability, which inspired us to formulate a stricter definition and to categorise the existing methods based on it. Montavon et al. (2018) confine the definition of explanation to feature importance (also called explanation vectors elsewhere) and review the techniques to interpret learned concepts and individual predictions by networks. They do not aim to give a comprehensive overview and only include some

representative approaches. [Gilpin et al. \(2018\)](#) divide the approaches into three categories: explaining data processing, explaining data representation and explanation-producing networks. Under this categorisation, the linear proxy model method and the rule-extraction method are equally viewed as proxy methods, without noticing many differences between them (the former is a local method while the latter is usually global and their produced explanations are different, we will see it in our taxonomy). [Guidotti et al. \(2018\)](#) consider all black-box models (including tree ensembles, SVMs etc.) and give a fine-grained classification based on four dimensions (the type of interpretability problem, the type of explainer, the type of black-box model, and the type of data). However, they treat decision trees, decision rules, saliency masks, sensitivity analysis, activation maximisation etc. equally, as explainers. In our view, some of them are certain types of explanations while some of them are methods used to produce explanations. [Zhang and Zhu \(2018\)](#) review the methods to understand network’s mid-layer representations or to learn networks with interpretable representations in computer vision field. There is another group of research which concerns with the *causal* interpretability ([Pearl, 2009](#)), while most of the current interpretability research only deals with the statistical correlation between a model’s input and output. For this topic, we would like to refer the readers to a recent survey ([Moraffah et al., 2020](#)).

This chapter has the following contributions:

- We make a further step towards the definition of interpretability on the basis of reference ([Doshi-Velez and Kim, 2017](#)). In this definition, we emphasise the *type* (or *format*) of *explanations* (e.g, rule forms, including both decision trees and decision rule sets). This acts as an important dimension in our proposed taxonomy. Previous papers usually organise existing methods into various isolated (to a large extent) *explainers* (e.g., decision trees, decision rules, feature importance, saliency maps etc.).
- We analyse the real needs for interpretability and summarise them into 3 groups: interpretability as an important component in systems that should be highly-reliable, ethical or legal requirements, and interpretability providing tools to enhance knowledge

in the relevant science fields. In contrast, a previous survey (Guidotti et al., 2018) only shows the importance of interpretability by providing several cases where black-box models can be dangerous.

- We propose a new taxonomy comprising three dimensions (passive vs. active approaches, the format of explanations, and local-semilocal-global interpretability). Note that although many ingredients of the taxonomy have been discussed in the previous literature, they were either mentioned in totally different context, or entangled with each other. To the best of our knowledge, our taxonomy provides the most comprehensive and clear categorisation of the existing approaches.

The three degrees of freedom along which our taxonomy is organised allow for a schematic 3D view illustrating how diverse attempts at interpretability of deep networks are related. It also provides suggestions for possible future work by filling some of the gaps in the interpretability research (see Figure 3.2).

3.3 A 3D Taxonomy

We propose a novel taxonomy with three dimensions (see Figure 3.1): (1) the passive vs. active approaches dimension, (2) the type/format of produced explanations, and (3) from local to global interpretability dimension respectively. **The first dimension** is categorical and has two possible values, *passive interpretation* and *active interpretability intervention*. It divides the existing approaches according to whether they require to change the network architecture or the optimisation process. The passive interpretation process starts from a trained network, with all the weights already learned from the training set. Thereafter, the methods try to extract logic rules or extract some understandable patterns. In contrast, active methods require some changes before the training, such as introducing extra network structures or modifying the training process. These modifications encourage the network to become more interpretable (e.g., more like a decision tree). Most commonly such active

Dimension 1 — Passive vs. Active Approaches	
<div> <div>Passive</div> <div>Active</div> </div>	<div>Post hoc explain trained neural networks</div> <div>Actively change the network architecture or training process for better interpretability</div>
Dimension 2 — Type of Explanations (in the order of increasing explanatory power)	
To explain a prediction/class by	
<div> <div>Examples</div> <div>Attribution</div> <div>Hidden semantics</div> <div>Rules</div> </div>	<div>Provide example(s) which may be considered similar or as prototype(s)</div> <div>Assign credit (or blame) to the input features (e.g. feature importance, saliency masks)</div> <div>Make sense of certain hidden neurons/layers</div> <div>Extract logic rules (e.g. decision trees, rule sets and other rule formats)</div>
Dimension 3 — Local vs. Global Interpretability (in terms of the input space)	
<div> <div>Local</div> <div>Semi-local</div> <div>Global</div> </div>	<div>Explain network's <i>predictions on individual samples</i> (e.g. a saliency mask for an input image)</div> <div>In between, for example, explain a group of similar inputs together</div> <div>Explain the network <i>as a whole</i> (e.g. a set of rules/a decision tree)</div>

Figure 3.1: The 3 dimensions of our taxonomy.

interventions come in the form of regularisation terms.

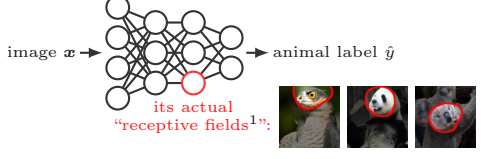




In contrast to previous surveys, the other two dimensions allow ordinal values. For example, the previously proposed dimension *type of explainer* (Guidotti et al., 2018) produces subcategories like *decision trees*, *decision rules*, *feature importance*, *sensitivity analysis* etc. However, there is no clear connection among these pre-recognised explainers (what is the relation between decision trees and feature importance). Instead, our **second dimension** is *type/format of explanation*. By inspecting various kinds of explanations produced by different approaches, we can observe differences in how explicit they are. Logic rules provide the most clear and explicit explanations while other kinds of explanations may be implicit. For example, a saliency map itself is just a mask on top of a certain input. By looking at the saliency map, people construct an explanation “the model made this prediction because it focused on this highly influential part and that part (of the input)”. Hopefully, these parts

correspond to some domain understandable concepts. Strictly speaking, implicit explanations by themselves are not complete explanations and need further human interpretation, which is usually automatically done when people see them. We recognise four major types of explanations here, *logic rules*, *hidden semantics*, *attribution* and *explanations by examples*, listed in order of decreasing explanatory power. Similar discussions can be found in the previous literature, e.g., Samek and Müller (2019) provide a short subsection about “type of explanations” (including explaining learned representations, explaining *individual predictions* etc.). However, it is mixed up with another independent dimension of the interpretability research which we will introduce in the following paragraph. A recent survey (Bodria et al., 2021) follows the same philosophy and treats saliency maps and concept attribution (Kim et al., 2018) as different types of explanations, while we view them as being of the same kind, but differing in the dimension below.

The last dimension, from local to global interpretability (w.r.t. the input space), has become very common in recent papers (e.g., Doshi-Velez and Kim (2017), Guidotti et al. (2018), Montavon et al. (2018), and Molnar et al. (2020)), where global interpretability means being able to understand the overall decision logic of a model and local interpretability focuses on the explanations of individual predictions. However, in our proposed dimension, there exists a transition rather than a hard division between global and local interpretability (i.e. semi-local interpretability). Local explanations usually make use of the information at the target input (e.g., its feature values, its gradient). But global explanations try to generalise to as wide ranges of inputs as possible (e.g., sequential *covering* in rule learning, *marginal* contribution for feature importance ranking). This view is also supported by the existence of several semi-local explanation methods (Ribeiro et al., 2018; Adler et al., 2018). There have also been attempts to fuse local explanations into global ones in a bottom-up fashion (Pedreschi et al., 2019; Lapuschkin et al., 2019; Natesan Ramamurthy et al., 2020).

To help understand the latter two dimensions, Table 3.2 lists examples of typical explanations produced by different subcategories under our taxonomy. **(Row 1)** When considering *rule as explanation for local interpretability*, an example is to provide rule explanations which

Table 3.2: Example explanations of networks. Please see Section 3.3 (understanding the latter two dimensions) for details. Due to lack of space, we do not provide examples for semi-local interpretability here.

	Local (and semi-local) interpretability applies to a certain input $\mathbf{x}^{(i)}$ (and its associated output $\hat{y}^{(i)}$), or a small range of inputs-outputs	Global interpretability w.r.t. the whole input space
Rule as explanation	<p>Explain a certain $(\mathbf{x}^{(i)}, y^{(i)})$ with a decision rule:</p> <ul style="list-style-type: none"> The result “$\mathbf{x}^{(i)}$ is classified as $\hat{y}^{(i)}$” is because $\mathbf{x}_1, \mathbf{x}_4, \dots$ are present and $\mathbf{x}_3, \mathbf{x}_5, \dots$ are absent (Dhurandhar et al., 2018). (Semi-local) For \mathbf{x} in the neighbourhood of $\mathbf{x}^{(i)}$, if $(\mathbf{x}_1 > \alpha) \wedge (\mathbf{x}_3 < \beta) \wedge \dots$, then $y = \hat{y}^{(i)}$ (Ribeiro et al., 2018). 	<p>Explain the whole model $y(\mathbf{x})$ with a rule set:</p> <p>The neural network can be approximated by</p> $\begin{cases} \text{If } (\mathbf{x}_2 < \alpha) \wedge (\mathbf{x}_3 > \beta) \wedge \dots, & \text{then } y = 1, \\ \text{If } (\mathbf{x}_1 > \gamma) \wedge (\mathbf{x}_5 < \delta) \wedge \dots, & \text{then } y = 2, \\ \dots & \\ \text{If } (\mathbf{x}_4 \dots) \wedge (\mathbf{x}_7 \dots) \wedge \dots, & \text{then } y = M \end{cases}$
Explaining hidden semantics (make sense of certain hidden neurons/layers)	<p>Explain a hidden neuron/layer $h(\mathbf{x}^{(i)})$:</p> <p>(*No explicit methods but many local attribution methods (see below) can be easily modified to “explain” a hidden neuron $h(\mathbf{x})$ rather than the final output y).</p>	<p>Explain a hidden neuron/layer $h(\mathbf{x})$ instead of $y(\mathbf{x})$:</p> <ul style="list-style-type: none"> An example active method (Zhang, Wu, et al., 2018) adds a special loss term that encourages filters to learn consistent and exclusive patterns (e.g. head patterns of animals) 
Attribution as explanation	<p>Explain a certain $(\mathbf{x}^{(i)}, y^{(i)})$ with an attribution $\mathbf{a}^{(i)}$:</p> <p>For $\mathbf{x}^{(i)}$:  → neural net → $\hat{y}^{(i)}$: junco bird</p> <p>The “contribution”² of each pixel:</p>  <p>a.k.a. saliency map, which can be computed by different methods like gradients³, sensitivity analysis⁴ etc.</p>	<p>Explain $y(\mathbf{x})$ with attribution to certain features:</p> <p>(Note that for a linear model, the coefficients is the global attribution to its input features.)</p> <ul style="list-style-type: none"> Kim et al. (2018) calculate attribution to a target “concept” rather than the input pixels of a certain input. For example, “how sensitive is the output (a prediction of zebra) to a concept (the presence of stripes)?”
Explanation by showing examples	<p>Explain a certain $(\mathbf{x}^{(i)}, y^{(i)})$ with another $\mathbf{x}^{(i)'}:$</p> <p>For $\mathbf{x}^{(i)}$:  → neural net → $\hat{y}^{(i)}$: fish</p> <p>By asking how much the network will change $\hat{y}^{(i)}$ if removing a certain training image, we can find:</p> <p>most helpful⁵ training images: </p>	<p>Explain $y(\mathbf{x})$ collectively with a few prototypes:</p> <ul style="list-style-type: none"> Adds a (learnable) prototype layer to the network. Every prototype should be similar to at least an encoded input. Every input should be similar to at least a prototype. The trained network explains itself by its prototypes (Li et al., 2018).

¹ see Zhou, Khosla, et al., 2015.

² the contribution to the network prediction of $\mathbf{x}^{(i)}$ (Adebayo et al., 2018).

³ see Simonyan et al., 2013.

⁴ how sensitive is the classification result to the change of pixels (Zeiler and Fergus, 2014).

⁵ without the training image, the network prediction of $\mathbf{x}^{(i)}$ will change a lot. In other words, these images help the network make a decision on $\mathbf{x}^{(i)}$ (Koh and Liang, 2017).

only apply to a given input $\mathbf{x}^{(i)}$ (and its associated output $\hat{y}^{(i)}$). One of the solutions is to find out (by perturbing the input features and seeing how the output changes) the minimal set of features $\mathbf{x}_k \dots \mathbf{x}_l$ whose presence supports the prediction $\hat{y}^{(i)}$. Analogously, features $\mathbf{x}_m \dots \mathbf{x}_n$ can be found which should not be present (larger values), otherwise $\hat{y}^{(i)}$ will change. Then an explanation rule for $\mathbf{x}^{(i)}$ can be constructed as “it is because $\mathbf{x}_k \dots \mathbf{x}_l$ are present and $\mathbf{x}_m \dots \mathbf{x}_n$ are absent that $\mathbf{x}^{(i)}$ is classified as $\hat{y}^{(i)}$ ” (Dhurandhar et al., 2018). If a rule is valid not only for the input $\mathbf{x}^{(i)}$, but also for its “neighbourhood” (Ribeiro et al., 2018), we obtain a *semi-local interpretability*. And if a rule set or decision tree is extracted from the original network, it explains the general function of the whole network and thus provides *global interpretability*. **(Row 2)** When it comes to *explaining the hidden semantics*, a typical example (*global*) is to visualise what pattern a hidden neuron is mostly sensitive to. This can then provide clues about the inner workings of the network. We can also take a more proactive approach to make hidden neurons more interpretable. As a high-layer hidden neuron may learn a mixture of patterns that can be hard to interpret, Zhang, Wu, et al. (2018) introduced a loss term that makes high-layer filters either produce consistent activation maps (among different inputs) or keep inactive (when not seeing a certain pattern). Experiments show that those filters are more interpretable (e.g., a filter may be found to be activated by the head parts of animals). **(Row 3)** *Attribution as explanation* usually provides *local interpretability*. Thinking about an animal classification task, input features are all the pixels of the input image. Attribution allows people to see which regions (pixels) of the image contribute the most to the classification result. The attribution can be computed e.g., by sensitivity analysis in terms of the input features (i.e. all pixels) or some variants (Simonyan et al., 2013; Zeiler and Fergus, 2014). For *attribution for global interpretability*, deep neural networks usually cannot have as straightforward attribution as e.g., coefficients \mathbf{w} in linear models $y = \mathbf{w}^\top \mathbf{x} + b$, which directly show the importance of features globally. Instead of concentrating on input features (pixels), Kim et al. (2018) were interested in attribution to a “concept” (e.g., how sensitive is a prediction of zebra to the presence of stripes). The concept (stripes) is represented by the normal vector to the plane which separates having-stripes and

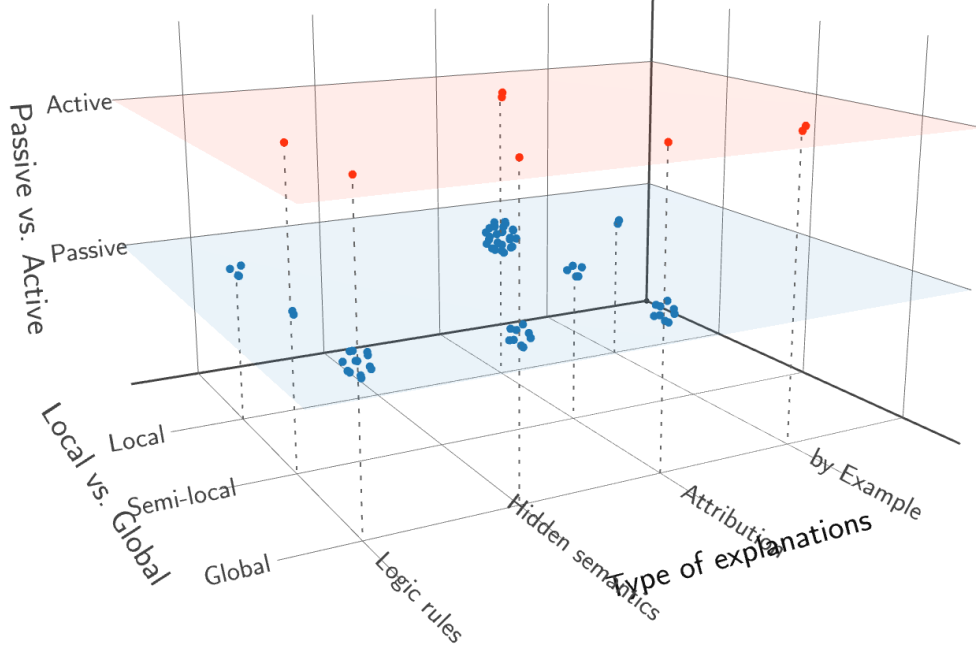


Figure 3.2: The distribution of the interpretability papers in the 3D space of our taxonomy. We can rotate and observe the density of work in certain areas/planes and find the missing parts of interpretability research. (Please see <https://yzhang-gh.github.io/tmp-data/index.html> for an online interactive version.)

non-stripes training examples in the space of network’s hidden layer. It is therefore possible to compute how sensitive the prediction (of zebra) is to the concept (presence of stripes) and thus have some form of global interpretability. **(Row 4)** Sometimes researchers explain network prediction by *showing other known examples* providing similar network functionality. To explain a single input $\mathbf{x}^{(i)}$ (*local interpretability*), we can find an example which is most similar to $\mathbf{x}^{(i)}$ in the network’s hidden layer level. This selection of explanation examples can also be done by testing how much the prediction of $\mathbf{x}^{(i)}$ will be affected if a certain example is removed from the training set (Koh and Liang, 2017). To provide *global interpretability by showing examples*, a method is adding a (learnable) prototype layer to a network. The prototype layer forces the network to make predictions according to the proximity between input and the learned prototypes. Those learned and interpretable prototypes can help to explain the network’s overall function.

With the three dimensions introduced above, we can visualise the distribution of the existing interpretability papers in a 3D view (Figure 3.2 only provides a 2D snapshot, we encourage readers to visit the online interactive version for better presentation). Table 3.3 is another representation of all the reviewed interpretability approaches which is good for quick navigation.

In the following the sections, we will scan through Table 3.3 along each dimension. The first dimension results in two sections, passive methods (Section 3.4) and active methods (Section 3.5). We then expand each section to several subsections according to the second dimension (type of explanation). Under each subsection, we introduce (semi-)local vs. global interpretability methods respectively.

3.4 Passive Interpretation of Trained Networks

Most of the existing network interpreting methods are passive methods. They try to understand the already trained networks. We now introduce these methods according to their types of produced explanations (i.e., the second dimension).

3.4.1 Passive, *Rule* as Explanation

Logic rules are commonly acknowledged to be interpretable and have a long history of research. Thus rule extraction is an appealing approach to interpret neural networks. In most cases, rule extraction methods provide *global* explanations as they only extract a single rule set or decision tree from the target model. There are only a few methods producing *(semi-)local* rule-form explanations which we will introduce below (Section 3.4.1.1), followed are global methods (Section 3.4.1.2). Another thing to note is that although the rules and decision trees (and their extraction methods) can be quite different, we do not explicitly differentiate them here as they provide similar explanations (a decision tree can be flattened

Table 3.3: An overview of the interpretability papers. The colours correspond to the three dimensions in Figure 3.1.

		Local	Semi-local	Global
Passive	Rule	CEM ²⁰¹⁸ , CDRPs ²⁰¹⁸ , CVE ¹²⁰¹⁹ , DACE ²⁰²⁰	Anchors ²⁰¹⁸ , Interpretable partial substitution ²⁰¹⁹	KT ¹⁹⁹¹ , MofN ¹⁹⁹³ , NeuralRule ¹⁹⁹⁵ , NeuroLinear ¹⁹⁹⁷ , GRG ²⁰⁰⁸ , Gyan (first-order rules) ²⁰⁰⁹ , Fuzzy rules ^{1997; 2002} , Trepan ¹⁹⁹⁵ , • ¹⁹⁹⁹ , DecText ²⁰⁰² , Global model on CEM ²⁰²⁰
	Hidden semantics	(*No explicit methods but many in the below cell can be applied here.)	—	Visualisation ^{2009; 2013; 2017} etc., Network dissection ²⁰¹⁷ , Net2Vec ²⁰¹⁸ , Linguistic correlation analysis ²⁰¹⁹
	Attribution ²	LIME ²⁰¹⁶ , MAPLE ²⁰¹⁸ , Partial derivatives ²⁰¹³ , DeconvNet ²⁰¹⁴ , Guided backprop ²⁰¹⁵ , Guided Grad-CAM ²⁰¹⁷ , Shapley values ^{2010; 2017; 2019; 2020} , Sensitivity analysis ^{2014; 2017; 2017} , Feature selector ²⁰¹⁸ , Bias attribution ²⁰¹⁹	DeepLIFT ²⁰¹⁷ , LRP ²⁰¹⁵ , Integrated gradients ²⁰¹⁷ , Feature selector ²⁰¹⁸ , MAME ²⁰²⁰	Feature selector ²⁰¹⁸ , TCAV ²⁰¹⁸ , ACE ²⁰¹⁹ , SpRAy ³²⁰¹⁹ , MAME ²⁰²⁰ , DeepConsensus ²⁰²⁰
	By example	Influence functions ²⁰¹⁷ , Representer point selection ²⁰¹⁸	—	—
Active	Rule	—	Regional tree regularisation ²⁰²⁰	Tree regularisation ²⁰¹⁸
	Hidden semantics	—	—	“One filter, one concept” ²⁰¹⁸
	Attribution	ExpO ²⁰²⁰ , DAPr ²⁰²⁰	—	Dual-net (feature importance) ²⁰²⁰
	By example	—	—	Network with a prototype layer ²⁰¹⁸ , ProtoPNet ²⁰¹⁹

¹ Short for counterfactual visual explanations.

² Some attribution methods (e.g., DeconvNet, Guided Backprop) arguably have certain non-locality because of the rectification operation.

³ SpRAy is flexible to provide semi-local or global explanations by clustering local (individual) attributions.

to a decision rule set). A basic form of a rule is

$$\text{If } P, \text{ then } Q. \quad (3.1)$$

where P is called the antecedent, and Q is called the consequent, which in our context is the prediction (e.g., class label) of a network. P is usually a combination of conditions on several input features. For complex models, the explanation rules can be of other forms such as the propositional rule, first-order rule or fuzzy rule.

3.4.1.1 Passive, Rule as Explanation, (*Semi-*)local

According to our taxonomy, methods in this category focus on a trained neural network and a certain input (or a small group of inputs), and produce a logic rule as an explanation. [Dhurandhar et al. \(2018\)](#) construct local rule explanations by finding out features that should be minimally and sufficiently *present* and features that should be minimally and necessarily *absent*. In short, the explanation takes this form “*If an input \mathbf{x} is classified as class y , it is because features f_i, \dots, f_k are present and features f_m, \dots, f_p are absent*”. This is done by finding small sparse perturbations that are sufficient to ensure the same prediction by its own (or will change the prediction if applied to a target input)¹. A similar kind of methods is counterfactual explanations ([Wachter et al., 2017](#)). Usually, we are asking based on what features (’s values) the neural network makes the prediction of class c . However, [Goyal et al. \(2019\)](#) try to find the minimum-edit on an input image which can result in a different predicted class c' . In other words, they ask: “What region in the input image makes the prediction to be class c , *rather than c'* ”. [Kanamori et al. \(2020\)](#) introduced distribution-aware counterfactual explanations, which require above “edit” to follow the empirical data distribution instead of being arbitrary.

[Wang, Su, et al. \(2018\)](#) came up with another local interpretability method, which identifies *critical data routing paths* (CDRPs) of the network for each input. In convolutional

¹The authors also extended this method to learn a *global* interpretable model, e.g., a decision tree, based on custom features created from above *local* explanations ([Pedapati et al., 2020](#)).

neural networks, each kernel produces a feature map that will be fed into the next layer as a channel. Wang, Su, et al. (2018) associated every output channel on each layer with a gate (non-negative weight), which indicates how critical that channel is. These gate weights are then optimised such that when they are multiplied with the corresponding output channels, the network can still make the same prediction as the original network (on a given input). Importantly, the weights are encouraged to be sparse (most are close to zero). CDRPs can then be identified for each input by first identifying the *critical nodes*, i.e. the intermediate kernels associated with positive gates. We can explore and assign meanings to the critical nodes so that the critical paths become local explanations. However, as the original paper did not go further on the CDRPs representation which may not be human-understandable, it is still more of an activation pattern than a real explanation.

We can also extract rules that cover a group of inputs rather than a single one. Ribeiro et al. (2018) propose *anchors* which are if-then rules that are sufficiently precise (semi-)locally. In other words, if a rule applies to a group of similar examples, their predictions are (almost) always the same. It is similar to (actually, on the basis of) an attribution method LIME, which we will introduce in Section 3.4.3. However, they are different in terms of the produced explanations (LIME produces attribution for individual examples). Wang (2019) attempted to find an interpretable partial substitution (a rule set) to the network that covers a certain subset of the input space. This substitution can be done with no or low cost on the model accuracy according to the size of the subset.

3.4.1.2 Passive, Rule as Explanation, *Global*

Most of the time, we would like to have some form of an overall interpretation of the network, rather than its local behaviour at a single point. We again divide these approaches into two groups. Some rule extraction methods make use of the network-specific information such as the network structure, or the learned weights. These methods are called *decompositional* approaches in previous literature (Craven and Shavlik, 1994). The other methods instead view the network as a black-box and only use it to generate training examples for classic

rule learning algorithms. They are called *pedagogical* approaches.

Decompositional approaches. Decompositional approaches generate rules by observing the connections in a network. As many of these approaches were developed before the deep learning era, they are mostly designed for classic fully-connected feedforward networks. Considering a single-layer setting of a fully-connected network (only one output neuron),

$$y = \sigma \left(\sum_i \mathbf{w}_i \mathbf{x}_i + b \right) \quad (3.2)$$

where σ is an activation function (usually sigmoid, $\sigma(x) = 1/(1 + e^{-x})$), \mathbf{w} are the trainable weights, \mathbf{x} is the input vector, and b is the bias term (often referred as a threshold θ is the early time, and b here can be interpreted as the negation of θ). Lying at the heart of rule extraction is to search for combinations of certain values (or ranges) of attributes \mathbf{x}_i that make y near 1 (Towell and Shavlik, 1993). This is tractable only when we are dealing with small networks because the size of the search space will soon grow to an astronomical number as the number of attributes and the possible values for each attribute increase. Assuming we have n Boolean attributes \mathbf{x}_i as an input, and each attribute can be **true** or **false** or absent in the antecedent, there are $\mathcal{O}(3^n)$ combinations to search. We therefore need some search strategies.

One of the earliest methods is the KT algorithm (Fu, 1991). KT algorithm first divides the input attributes into two groups, pos-atts (short for positive attributes) and neg-atts, according to the signs of their corresponding weights. Assuming activation function is sigmoid, all the neurons are booleanised to **true** (if close enough to 1) or **false** (close to 0). Then, all combinations of pos-atts are selected if the combination can on its own make y be **true** (larger than a pre-defined threshold β without considering the neg-atts), for instance, a combination $\{\mathbf{x}_1, \mathbf{x}_3\}$ and $\sigma(\sum_{i \in \{1,3\}} \mathbf{w}_i \mathbf{x}_i + b) > \beta$. Finally, it takes into account the neg-atts. For each above pos-atts combination, it finds combinations of neg-atts (e.g., $\{\mathbf{x}_2, \mathbf{x}_5\}$) that when absent the output calculated from the selected pos-atts and unselected neg-atts is still **true**. In other words, $\sigma(\sum_{i \in \mathcal{I}} \mathbf{w}_i \mathbf{x}_i + b) > \beta$, where $\mathcal{I} = \{\mathbf{x}_1, \mathbf{x}_3\} \cup \{\text{neg-atts}\} \setminus \{\mathbf{x}_2, \mathbf{x}_5\}$.

The extracted rule can then be formed from the combination \mathcal{I} and has the output class 1. In our example, the translated rule is

$$\text{If } \mathbf{x}_1(\text{is true}) \wedge \mathbf{x}_3 \wedge \neg \mathbf{x}_2 \wedge \neg \mathbf{x}_5, \text{ then } y = 1. \quad (3.3)$$

Similarly, this algorithm can generate rules for class 0 (searching neg-atts first and then adding pos-atts). To apply to the multi-layer network situation, it first does layer-by-layer rule generation and then rewrites them to omit the hidden neurons. In terms of the complexity, KT algorithm reduces the search space to $\mathcal{O}(2^n)$ by distinguishing pos-atts and neg-atts (pos-atts will be either **true** or absent, and neg-atts will be either **false** or absent). It also limits the number of attributes in the antecedent, which can further decrease the algorithm complexity (with the risk of missing some rules).

[Towell and Shavlik \(1993\)](#) focus on another kind of rules of “ M -of- N ” style. This kind of rule de-emphasises the individual importance of input attributes, which has the form

$$\text{If } M \text{ of these } N \text{ expressions are true, then } Q. \quad (3.4)$$

This algorithm has two salient characteristics. The first one is link (weight) clustering and reassigning them the average weight within the cluster. Another characteristic is network simplifying (eliminating unimportant clusters) and re-training. Comparing with the exponential complexity of subset searching algorithm, M -of- N method is approximately cubic because of its special rule form.

NeuroRule ([Setiono and Liu, 1995](#)) introduced a three-step procedure of extracting rules: (1) train the network and prune, (2) discretise (cluster) the activation values of the hidden neurons, (3) extract the rules layer-wise and rewrite (similar as previous methods). NeuroLinear ([Setiono and Liu, 1997](#)) made a little change to the NeuroRule method, allowing neural networks to have continuous input. [Andrews et al. \(1995\)](#) and [Tickle et al. \(1998\)](#) provide a good summary of the rule extraction techniques before 1998.

Pedagogical approaches. By treating the neural network as a black-box, *pedagogical* methods (or hybrids of both) directly learn rules from the examples generated by the network. It is essentially reduced to a traditional rule learning or decision tree learning problem. For rule set learning, we have sequential covering framework (i.e. to learn rules one by one). And for decision tree, there are many classic algorithms like CART (Loh, 2011) and C4.5 (Quinlan, 1993). Example work of decision tree extraction (from neural networks) can be found in references (Craven and Shavlik, 1995; Krishnan et al., 1999; Boz, 2002).

Odajima et al. (2008) followed the framework of NeuroLinear but use a greedy form of sequential covering algorithm to extract rules. It is reported to be able to extract more concise and precise rules. Nayak (2009) goes further than extracting propositional rules. After obtaining the propositional rules by the above methods, Gyan uses the Least General Generalisation (LGG (Plotkin, 1970)) method to generate first-order logic rules from them. There are also some approaches attempting to extract fuzzy logic from trained neural networks (Benitez et al., 1997; Mitra and Hayashi, 2000; Castro et al., 2002). The major difference is the introduction of the membership function of linguistic terms. An example rule is

$$\text{If } (x_1 = \text{high}) \wedge \dots, \text{ then } y = \text{class1.} \quad (3.5)$$

where **high** is a fuzzy term expressed as a fuzzy set over the numbers.

Most of the above “Rule as Explanation, Global” methods were developed in the early stage of neural network research, and usually were only applied to relatively small datasets (e.g., the Iris dataset, the Wine dataset from the UCI Machine Learning Repository (Dua and Graff, 2019)). However, as neural networks get deeper and deeper in recent applications, it is unlikely for a single decision tree to faithfully approximate the behaviour of deep networks. We can see that more recent “Rule as Explanation” methods turn to local or semi-local interpretability (Wang, 2019; Wu, Parbhoo, et al., 2020).

3.4.2 Passive, *Hidden Semantics* as Explanation

The second typical kind of explanations is the meaning of hidden neurons or layers. Similar to the grandmother cell hypothesis² in neuroscience, it is driven by a desire to associate abstract concepts with the activation of some hidden neurons. Taking animal classification as an example, some neurons may have high response to the head of an animal while others neurons may look for bodies, feet or other parts. This kind of explanations by definition provides *global* interpretability.

3.4.2.1 Passive, Hidden Semantics as Explanation, *Global*

Existing hidden semantics interpretation methods mainly focus on the computer vision field. The most direct way is to show what the neuron is “looking for”, i.e. visualisation. The key to visualisation is to find a representative input that can maximise the activation of a certain neuron, channel or layer, which is usually called *activation maximisation* (Erhan et al., 2009). This is an optimisation problem, whose search space is the potentially huge input (sample) space. Assuming we have a network taking as input a 28×28 pixels black and white image (as in the MNIST handwritten digit dataset), there will be $2^{28 \times 28}$ possible input images, although most of them are probably nonsensical. In practice, although we can find a maximum activation input image with optimisation, it will likely be unrealistic and uninterpretable. This situation can be helped with some regularisation techniques or priors.

We now give an overview over these techniques. The framework of activation maximisation was introduced by Erhan et al. (2009) (although it was used in the unsupervised deep models like Deep Belief Networks). In general, it can be formulated as

$$\mathbf{x}^* = \arg \max_{\mathbf{x}} (act(\mathbf{x}; \theta) - \lambda \Omega(\mathbf{x}))$$

where $act(\cdot)$ is the activation of the neuron of interest, θ is the network parameters (weights and biases) and Ω is an optional regulariser. (We use the bold upright \mathbf{x} to denote an input

²https://en.wikipedia.org/wiki/Grandmother_cell

matrix in image related tasks, which allows row and column indices i and j .)

[Simonyan et al. \(2013\)](#) for the first time applied activation maximisation to a supervised deep convolutional network (for ImageNet classification). It finds representative images by maximizing the score of a class (before softmax). And the Ω is the L^2 norm of the image. Later, people realised high frequency noise is a major nuisance that makes the visualisation unrecognisable ([Olah et al., 2017](#); [Wang, Liu, et al., 2018](#)). In order to get natural and useful visualisations, finding good priors or regularisers Ω becomes the core task in this kind of approaches.

[Mahendran and Vedaldi \(2015\)](#) propose a regulariser *total variation*

$$\Omega(\mathbf{x}) = \sum_{i,j} \left((\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j})^2 + (\mathbf{x}_{i+1,j} - \mathbf{x}_{i,j})^2 \right)^{\frac{\beta}{2}}$$

which encourages neighbouring pixels to have similar values. This can also be viewed as a low-pass filter that can reduce the high frequency noise in the image. This kind of methods is usually called image blurring. Besides suppressing high amplitude and high frequency information (with L^2 decay and Gaussian blurring respectively), [Yosinski et al. \(2015\)](#) also include other terms to clip the pixels of small values or little importance to the activation. Instead of using many hand-crafted regularisers (image priors), [Nguyen, Dosovitskiy, et al. \(2016\)](#) suggest using natural image prior learned by a generative model. As Generative Adversarial Networks (GANs) ([Goodfellow, Pouget-Abadie, et al., 2014](#)) showed recently great power to generate high-resolution realistic images ([Radford et al., 2016](#); [Ledig et al., 2017](#)), making use of the generative model of a GAN appears to be a good choice. For a good summary and many impressive visualisations, we refer the readers to ([Olah et al., 2017](#)). When applied to certain tasks, researchers can get some insights from these visual interpretations. For example, [Minematsu, Shimada, and Taniguchi \(2017\)](#) and [Minematsu, Shimada, Uchiyama, et al. \(2018\)](#) inspected the behaviour of the first and last layers of a DNN used for change detection (in video stream), which may suggest the possibility of a new background modelling strategy.

Besides visualisation, there are also some work trying to find connections between kernels and visual concepts (e.g., materials, certain objects). [Bau, Zhou, et al. \(2017\)](#) (Network Dissection) collected a new dataset Broden which provides a pixel-wise binary mask $L_c(\mathbf{x})$ for every concept c and each input image \mathbf{x} . The activation map of a kernel k is upsampled and converted (given a threshold) to a binary mask $M_k(\mathbf{x})$ which has the same size of \mathbf{x} . Then the alignment between a kernel k and a certain concept c (e.g., car) is computed as

$$IoU_{k,c} = \frac{\sum |M_k(\mathbf{x}) \cap L_c(\mathbf{x})|}{\sum |M_k(\mathbf{x}) \cup L_c(\mathbf{x})|}$$

where $|\cdot|$ is the cardinality of a set and the summation \sum is over all the inputs \mathbf{x} that contains the concept c . Along the same lines, [Fong and Vedaldi \(2018\)](#) investigate the embeddings of concepts over multiple kernels by constructing M with a combination of several kernels. Their experiments show that multiple kernels are usually required to encode one concept and kernel embeddings are better representations of the concepts.

[Dalvi et al. \(2019\)](#) also analysed the meaning of individual units/neurons in the networks for NLP tasks. They build a linear model between the network’s hidden neurons and the output. The neurons are then ranked according to the significance of the weights of the linear model. For those top-ranking neurons, their linguistic meanings are investigated by visualizing their saliency maps on the inputs, or by finding the top words by which they get activated.

3.4.3 Passive, *Attribution* as Explanation

Attribution is to assign credit or blame to the input features in terms of their impact on the output (prediction). The explanation will be a real-valued vector which indicates feature importance with the sign and amplitude of the scores ([Montavon et al., 2018](#)). For simple models (e.g., linear models) with meaningful features, we might be able to assign each feature a score *globally*. When it comes to more complex networks and input, e.g., images, it is hard to say a certain pixel always has similar contribution to the output. Thus, many methods

do attribution *locally*. We introduce them below and at the end of this section we mention a global attribution method on intermediate representation rather than the original input features.

3.4.3.1 Passive, Attribution as Explanation, (*Semi-*)*local*

Similarly to the compositional vs. pedagogical division of rule extraction methods, attribution methods can be also divided into two groups: gradient-related methods and model agnostic methods.

Gradient-related and backpropagation methods. Using gradients to explain the individual classification decisions is a natural idea as the gradient represents the “direction” and rate of the fastest increase on the loss function. The gradients can also be computed with respect to a certain output class, for example, along which “direction” a perturbation will make an input more/less likely predicted as a cat/dog. [Baehrens et al. \(2010\)](#) use it to explain the predictions of Gaussian Process Classification (GPC), k -NN and SVM. For a special case, the coefficients of features in linear models (or general additive models) are already the partial derivatives, in other words, the (global) attribution. So people can directly know how the features affect the prediction and that is an important reason why linear models are commonly thought interpretable. While plain gradients, discrete gradients and path-integrated gradients have been used for attribution, some other methods do not calculate real gradients with the chain rule but only backpropagate attribution signals (e.g., do extra normalisation on each layer upon backpropagation). We now introduce these methods in detail.

In computer vision, the attribution is usually represented as a saliency map, a mask of the same size of the input image. In reference [Simonyan et al. \(2013\)](#), the saliency map is generated from the gradients (specifically, the maximum absolute values of the partial derivatives over all channels). This kind of saliency maps are obtained without effort as they only require a single backpropagation pass. They also showed that this method is equivalent

to the previously proposed deconvolutional nets method (Zeiler and Fergus, 2014) except for the difference on the calculation of ReLU layer’s gradients. Guided backpropagation (Springenberg et al., 2015) combines above two methods. It only takes into account the gradients (the former method) that have positive error signal (the latter method) when backpropagating through a ReLU layer. There is also a variant Guided Grad-CAM (Gradient-weighted Class Activation Mapping) (Selvaraju et al., 2017), which first calculates a coarse-grained attribution map (with respect to a certain class) on the last convolutional layer and then multiply it to the attribution map obtained from guided backpropagation. (Guided Grad-CAM is an extension of CAM (Zhou, Khosla, et al., 2016) which requires a special global average pooling layer.)

However, gradients themselves can be misleading. Considering a piecewise continuous function,

$$y = \begin{cases} \mathbf{x}_1 + \mathbf{x}_2 & \text{if } \mathbf{x}_1 + \mathbf{x}_2 < 1, \\ 1 & \text{if } \mathbf{x}_1 + \mathbf{x}_2 \geq 1. \end{cases} \quad (3.6)$$

it is saturated when $\mathbf{x}_1 + \mathbf{x}_2 \geq 1$. At points where $\mathbf{x}_1 + \mathbf{x}_2 > 1$, their gradients are always zeros. DeepLIFT (Shrikumar, Greenside, et al., 2017) points out this problem and highlights the importance of having a reference input besides the target input to be explained. The reference input is a kind of default or ‘neutral’ input and will be different in different tasks (e.g., blank images or zero vectors). Actually, Sundararajan et al. (2017) think DeepLIFT is trying to compute the “discrete gradient” instead of the (instantaneous) gradient. Another similar “discrete gradient” method is LRP (Bach et al., 2015) (choosing a zero vector as the reference point), differing in how to compute the discrete gradient. Ancona, Ceolini, et al. (2018) also present a similar opinion that LRP and DeepLIFT are essentially computing backpropagation for modified gradient functions.

However, discrete gradients also have their drawbacks. As the chain rule does not hold for discrete gradients, DeepLIFT and LRP adopt modified forms of backpropagation. This makes their attributions specific to the network implementation, in other words, the attributions can be different even for two functionally equivalent networks (Sundararajan et al.

Table 3.4: Formulation of gradient-related attribution methods. S_c is the output for class c (and it can be any neuron of interest), σ is the nonlinearity in the network and g is a replacement of σ' (the derivative of σ) in $\frac{\partial S_c(\mathbf{x})}{\partial \mathbf{x}}$ in order to rewrite DeepLIFT and LRP with gradient formulation (see [Ancona, Ceolini, et al., 2018](#) for more details). \mathbf{x}_i is the i -th feature (pixel) of \mathbf{x} .

Method	Attribution
Gradient Baehrens et al., 2010; Simonyan et al., 2013	$\frac{\partial S_c(\mathbf{x})}{\partial \mathbf{x}_i}$
Gradient \odot Input	$\mathbf{x}_i \cdot \frac{\partial S_c(\mathbf{x})}{\partial \mathbf{x}_i}$
LRP Bach et al., 2015	$\mathbf{x}_i \cdot \frac{\partial^g S_c(\mathbf{x})}{\partial \mathbf{x}_i}, g = \frac{\sigma(\mathbf{z})}{\mathbf{z}}$
DeepLIFT Shrikumar, Greenside, et al., 2017	$(\mathbf{x}_i - \mathbf{x}_i^{\text{ref}}) \cdot \frac{\partial^g S_c(\mathbf{x})}{\partial \mathbf{x}_i}, g = \frac{\sigma(\mathbf{z}) - \sigma(\mathbf{z}^{\text{ref}})}{\mathbf{z} - \mathbf{z}^{\text{ref}}}$
Integrated Gradient Sundararajan et al., 2017	$(\mathbf{x}_i - \mathbf{x}_i^{\text{ref}}) \cdot \int_0^1 \frac{\partial S_c(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}_i} \Big _{\tilde{\mathbf{x}}=\mathbf{x}^{\text{ref}}+\alpha(\mathbf{x}-\mathbf{x}^{\text{ref}})} d\alpha$

(2017) provides a concrete example in its Appendix B). Integrated gradients ([Sundararajan et al., 2017](#)) was proposed to address this problem. It is defined as the path integral of all the gradients in the straight line between input \mathbf{x} and the reference input \mathbf{x}^{ref} . The i^{th} dimension of the integrated gradient (IG) is defined as follows,

$$\text{IG}_i(\mathbf{x}) := (\mathbf{x}_i - \mathbf{x}_i^{\text{ref}}) \int_0^1 \frac{\partial f(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}_i} \Big|_{\tilde{\mathbf{x}}=\mathbf{x}^{\text{ref}}+\alpha(\mathbf{x}-\mathbf{x}^{\text{ref}})} d\alpha \quad (3.7)$$

where $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i}$ is the i^{th} dimension of the gradient of $f(\mathbf{x})$. For those attribution methods requiring a reference point, semi-local interpretability is provided as users can select different reference points according to what to explain. Table 3.4 summarises the above gradient-related attribution methods (adapted from [Ancona, Ceolini, et al. \(2018\)](#)). In addition to the “gradient” attribution discussed above, [Wang, Zhou, et al. \(2019\)](#) point out that bias terms can contain attribution information complementary to the gradients. They propose a method to recursively assign the bias attribution back to the input vector.

Those discrete gradient methods (e.g., LRP and DeepLIFT) provide semi-local explanations as they explain a target input w.r.t. another reference input. But methods such as DeconvNet and Guided Backprop, which are only proposed to explain individual inputs, arguably have certain non-locality because of the rectification operation during the process. Moreover, one can accumulate multiple local explanations to achieve a certain degree of global interpretability, which will be introduced in Section 3.4.3.2.

Although we have many approaches to produce plausible saliency maps, there is still a small gap between saliency maps and real explanations. There have even been adversarial examples for attribution methods, which can produce perceptively indistinguishable inputs, leading to the *same predicted labels*, but very *different attribution maps* (Ghorbani, Abid, et al., 2019; Dombrowski et al., 2019; Heo et al., 2019). Researchers came up with several properties a saliency map should have to be a valid explanation. Sundararajan et al. (2017) (integrated gradients method) introduced two requirements, *sensitivity* and *implementation invariance*. Sensitivity requirement is proposed mainly because of the (local) gradient saturation problem (which results in zero gradient/attribution). Implementation invariance means two functionally equivalent networks (which can have different learned parameters given the over-parametrizing setting of DNNs) should have the same attribution. Kindermans et al. (2019) introduced input invariance. It requires attribution methods to mirror the model’s invariance with respect to transformations of the input. For example, a model with a bias term can easily deal with a constant shift of the input (pixel values). Obviously, (plain) gradient attribution methods satisfy this kind of input invariance. For discrete gradient and other methods using reference points, they depend on the choices of reference. Adebayo et al. (2018) took a different approach. They found that edge detectors can also produce masks which look similar to saliency masks and highlight some features of the input. But edge detectors have nothing to do with the network or training data. Thus, they proposed two tests to verify whether the attribution method will fail (1) if the network’s weights are replaced with random noise, or (2) if the labels of training data are shuffled. The attribution methods should fail otherwise it suggests that the method does not reflect the trained

network or the training data (in other words, it is just something like an edge detector).

Model agnostic attribution. LIME (Ribeiro et al., 2016) is a well-known approach which can provide local attribution explanations (if choosing linear models as the so-called interpretable components). Let $f: \mathbb{R}^d \rightarrow \{+1, -1\}$ be a (binary classification) model to be explained. Because the original input $\mathbf{x} \in \mathbb{R}^d$ might be uninterpretable (e.g., a tensor of all the pixels in an image, or a word embedding (Mikolov et al., 2013)), LIME introduces an intermediate representation $\mathbf{x}' \in \{0, 1\}^{d'}$ (e.g., the existence of certain image patches or words). \mathbf{x}' can be recovered to the original input space \mathbb{R}^d . For a given \mathbf{x} , LIME tries to find a potentially interpretable model g (such as a linear model or decision tree) as a local explanation.

$$g_{\mathbf{x}} = \arg \min_{g \in G} L(f, g, \pi_{\mathbf{x}}) + \Omega(g) \quad (3.8)$$

where G is the explanation model family, L is the loss function that measures the fidelity between f and g . L is evaluated on a set of perturbed samples around \mathbf{x}' (and their recovered input), which are weighted by a local kernel $\pi_{\mathbf{x}}$. Ω is the complexity penalty of g , ensuring g to be interpretable. MAPLE (Plumb, Molitor, et al., 2018) is a similar method using local linear models as explanations. The difference is it defines the locality as how frequently the data points fall into a same leaf node in a proxy random forest (fit on the trained network).

In game theory, there is a task to “fairly” assign each player a payoff from the total gain generated by a coalition of all players. Formally, let N be a set $\{1, 2, \dots, n\}$ representing n players, $v: 2^N \rightarrow \mathbb{R}$ is a characteristic function, which can be interpreted as the total gain of the coalition N . Obviously, $v(\emptyset) = 0$. A coalitional game can be denoted by the tuple $\langle N, v \rangle$. Given a coalitional game, Shapley value (Shapley, 1953) is a solution to the payoff assignment problem. The payoff (attribution) for player i can be computed as follows,

$$\phi_i(v) = \frac{1}{|N|} \sum_{S \subseteq N \setminus \{i\}} \binom{|N| - 1}{|S|}^{-1} (v(S \cup \{i\}) - v(S)) \quad (3.9)$$

where $v(S \cup \{i\}) - v(S)$ is the marginal contribution of player i to coalition S . The rest

of the formula can be viewed as a normalisation factor. A well-known alternative form of Shapley value is

$$\phi_i(v) = \frac{1}{|N|!} \sum_{O \in \mathfrak{S}(N)} \left[v(P_i^O \cup \{i\}) - v(P_i^O) \right] \quad (3.10)$$

where $\mathfrak{S}(N)$ is the set of all ordered permutations of N , and P_i^O is the set of players in N which are predecessors of player i in the permutation O . Štrumbelj and Kononenko adopted this form so that v can be approximated in polynomial time (Štrumbelj and Kononenko, 2010) (also see Ancona, Oztireli, et al. (2019) for another approximation method).

Back to the neural network (denoted by f), let N be all the input features (attributes), S is an arbitrary feature subset of interest ($S \subseteq N$). For an input \mathbf{x} , the characteristic function $v(S)$ is the difference between “the expected model output when we know all the features in S ”, and “the expected output when no feature value is known” (i.e. the expected output over all possible input \mathbf{u}), denoted by

$$v(S) = \frac{1}{|\mathcal{X}^{N \setminus S}|} \sum_{\mathbf{v} \in \mathcal{X}^{N \setminus S}} f(\tau(\mathbf{x}, \mathbf{v}, S)) - \frac{1}{|\mathcal{X}^N|} \sum_{\mathbf{u} \in \mathcal{X}^N} f(\mathbf{u}) \quad (3.11)$$

\mathcal{X}^N and $\mathcal{X}^{N \setminus S}$ are respectively the input space containing feature sets N and $N \setminus S$, $\tau(\mathbf{x}, \mathbf{v}, S)$ is a vector composed by \mathbf{x} and \mathbf{v} according to whether the feature is in S ,

$$\tau(\mathbf{x}, \mathbf{v}, S) = (\tau_1, \tau_2, \dots, \tau_n), \quad \tau_i = \begin{cases} x_i & \text{if } i \in S, \\ v_i & \text{if } i \notin S. \end{cases} \quad (3.12)$$

However, a practical problem is the exponential computation complexity, let alone the cost of the feed-forward computing on each $v(S)$ call. Štrumbelj and Kononenko (2010) approximate Shapley value by sampling from $\mathfrak{S}(N) \times \mathcal{X}$ (Cartesian product). There are other variants such as using different v . Lundberg and Lee (2017) propose a unified view including not only the Shapley value methods but also LRP and DeepLIFT. There is also Shapley value through the lens of causal graph (Heskes et al., 2020).

Sensitivity analysis can also be used to evaluate the importance of a feature. Specifically, the importance of a feature could be measured as how much the model output will change upon the change of a feature (or features). There are different kinds of changes, e.g.,

perturbation, occlusion and more (Zeiler and Fergus, 2014; Fong and Vedaldi, 2017; Zintgraf et al., 2017).

Chen, Song, et al. (2018) propose an instance-wise feature selector \mathcal{E} which maps an input \mathbf{x} to a conditional distribution $P(S | \mathbf{x})$, where S is any subset (of certain size) of the original feature set. The selected features can be denoted by \mathbf{x}_S . Then they aim to maximise the mutual information between the selected features and the output variable Y ,

$$\max_{\mathcal{E}} I(X_S; Y) \quad \text{subject to} \quad S \sim \mathcal{E}(X).$$

A variational approximation is used to obtain a tractable solution of the above problem.

3.4.3.2 Passive, Attribution as Explanation, *Global*

A natural way to get global attribution is to combine individual ones obtained from above local/semi-local methods. SpRAy (Lapuschkin et al., 2019) clusters on the individual attributions and then summarises some groups of prediction strategies. MAME (Natesan Ramamurthy et al., 2020) is a similar method that can generate a multilevel (local to global) explanation tree. Salman et al. (2020) provide a different way, which makes use of multiple neural networks. Each of the network can provide its own local attributions, on top of which a clustering is performed. Those clusters, intuitively the consensus of multiple models, can provide more robust interpretations.

The attribution does not necessarily attribute ‘credits’ or ‘blame’ to the *raw* input or features. Kim et al. (2018) propose a method TCAV (quantitative Testing with Concept Activation Vectors) that can compute the model sensitivity of any user-interested concept. By first collecting some examples with and without a target concept (e.g., the presence of stripes in an animal), the concept can then be represented by a normal vector to the hyperplane separating those positive/negative examples (pictures of animals with/without stripes) in a hidden layer. The score of the concept can be computed as the (average) output sensitivity if the hidden layer representation (of an input \mathbf{x}) moves an infinitesimally small step along the concept vector. This is a global interpretability method as it explains

how a concept affects the output in general. Besides being manually picked by a human, these concepts can also be discovered automatically by clustering input segments (Ghorbani, Wexler, et al., 2019).

3.4.4 Passive, Explanation *by Example*

The last kind of explanations we reviewed is explanation by example. When asked for an explanation for a new input, these approaches return other example(s) for supporting or counter example. One basic intuition is to find examples that the model considers to be most similar (in terms of latent representations) (Caruana et al., 1999). This is *local* interpretability but we can also seek a set of representative samples within a class or for more classes that provides *global* interpretability. A general approach is presented in Bien and Tibshirani (2011). There are other methods, such as measuring how much a training example affects the model prediction on a target input. Here we only focus on work related to deep neural networks.

3.4.4.1 Passive, Explanation by Example, *Local*

Koh and Liang (2017) provide an interesting method to evaluate how much a training example affects the model prediction on an unseen test example. The change of model parameters upon a change of training example is first calculated with approximation. Further, its influence on the loss at the test point can be computed. By checking the most influential (positively or negatively) training examples (for the test example), we can have some insights on the model predictions. Yeh et al. (2018) show that the logit (the neuron before softmax) can be decomposed into a linear combination of training points' activations in the pre-logit layer. The coefficients of the training points indicate whether the similarity to those points is excitatory or inhibitory. The above two approaches both provide local explanations.

3.5 Active Interpretability Intervention During Training

Besides passive looking for human-understandable patterns from the trained network, researchers also tried to impose interpretability restrictions during the network training process, i.e. active interpretation methods in our taxonomy. A popular idea is to add a special regularisation term $\Omega(\theta)$ to the loss function, also known as “interpretability loss” (θ collects all the weights of a network). We now discuss the related papers according to the forms of explanations they provide.

3.5.1 Active, *Rule* as Explanation (semi-local or global)

Wu, Hughes, et al. (2018) propose tree regularisation which favours models that can be well approximated by shallow decision trees. It requires two steps (1) train a binary decision tree using data points $\{\mathbf{x}^{(i)}, \hat{y}^{(i)}\}^N$, where $\hat{y} = f_{\theta}(\mathbf{x})$ is the network prediction rather than the true label, (2) calculate the average path length (from root to leaf node) of this decision tree over all the data points. However, this tree regularisation term $\Omega(\theta)$ is not differentiable. Therefore, a surrogate regularisation term $\hat{\Omega}(\theta)$ was introduced. Given a dataset $\{\theta^{(j)}, \Omega(\theta^{(j)})\}_{j=1}^J$, $\hat{\Omega}$ can be trained as a multi-layer perceptron network which minimises the squared error loss

$$\min_{\xi} \sum_{j=1}^J \left(\Omega(\theta^{(j)}) - \hat{\Omega}(\theta^{(j)}; \xi) \right)^2 + \epsilon \|\xi\|_2^2 \quad (3.13)$$

$\{\theta^{(j)}, \Omega(\theta^{(j)})\}_{j=1}^J$ can be assembled during network training. Also, data augmentation techniques can be used to generate θ , especially in the early training phase. Tree regularisation enables global interpretability as it forces a network to be easily approximable by a decision tree. Later, the authors also proposed regional tree regularisation which did this in a semi-local way (Wu, Parbhoo, et al., 2020).

3.5.2 Active, *Hidden semantics* as Explanation (global)

Another method aims to make a convolutional neural network learn better (disentangled) hidden semantics. Having seen feature visualisation techniques mentioned above and some empirical studies (Bau, Zhou, et al., 2017; Zhou, Bau, et al., 2018), CNNs are believed to have learned some low-level to high-level representations in the hierarchical structure. But even if higher-layers have learned some object-level concepts (e.g., head, foot), those concepts are usually entangled with each other. In other words, a high-layer filter may contain a mixture of different patterns. Zhang, Wu, et al. (2018) propose a loss term which encourages high-layer filters to represent a single concept. Specifically, for a CNN, a feature map (output of a high-layer filter, after ReLU) is an $n \times n$ matrix. Zhang et al. predefined a set of n^2 ideal feature map templates (activation patterns) \mathbf{T} , each of which is like a Gaussian kernel only differing on the position of its peak. During the forward propagation, the feature map is masked (element-wise product) by a certain template $T \in \mathbf{T}$ according to the position of the most activated “pixel” in the original feature map. During the back propagation, an extra loss is plugged in, which is the mutual information between \mathbf{M} (the feature maps of a filter calculated on all images) and $\mathbf{T} \cup \{\mathbf{T}^-\}$ (all the ideal activation patterns plus a negative pattern which is full of a negative constant). This loss term makes a filter to either have a consistent activation pattern or keep inactivated. Experiments show that filters in their designed architecture are more semantically meaningful (e.g., the “receptive field” (Zhou, Khosla, et al., 2015) of a filter corresponds to the head of animals).

3.5.3 Active, *Attribution* as Explanation

Similar to tree regularisation which helps to achieve better global interpretability (decision trees), ExpO (Plumb, Al-Shedivat, et al., 2020) added an interpretability regulariser in order to improve the quality of local attribution. That regularisation requires a model to have fidelitous (high fidelity) and stable local attribution. DAPr (Deep Attribution Prior) (Weinberger et al., 2020) took into account additional information (e.g., a rough prior about the

feature importance). The prior will be trained jointly with the main prediction model (as a regulariser) and biases the model to have similar attribution as the prior.

Besides performing attribution on individual input (*locally* in input space), Dual-net (Wojtas and Chen, 2020) was proposed to decide feature importance population-wise, i.e., finding an ‘optimal’ feature subset collectively for an input population. In this method, a *selector* network is used to generate an optimal feature subset, while an *operator* network makes predictions based on that feature subset. These two networks are trained jointly. After training, the selector network can be used to rank feature importance.

3.5.4 Active, Explanations *by Prototypes* (global)

Li et al. (2018) incorporated a prototype layer to a network (specifically, an autoencoder). The network acts like a prototype classifier, where predictions are made according to the proximity between (the encoded) inputs and the learned prototypes. Besides the cross-entropy loss and the (autoencoder) reconstruction error, they also included two interpretability regularisation terms, encouraging every prototype to be similar to at least one encoded input, vice versa. After the network is trained, those prototypes can be naturally used as explanations. Chen, Li, et al. (2019) add a prototype layer to a regular CNN rather than an autoencoder. This prototype layer contains prototypes that are encouraged to resemble parts of an input. When asked for explanations, the network can provide several prototypes for different parts of the input image respectively.

3.6 Evaluation of Interpretability

In general, interpretability is hard to evaluate objectively as the end-tasks can be quite divergent and may require domain knowledge from experts (Montavon et al., 2018). Doshi-Velez and Kim (2017) proposed three evaluation approaches: application-grounded, human-grounded, and functionally-grounded. The first one measures to what extent interpretability helps the end-task (e.g., better identification of errors or less discrimination). Human-

grounded approaches are, for example, directly letting people evaluate the quality of explanations with human-subject experiments (e.g., let a user choose which explanation is of the highest quality among several explanations). Functionally-grounded methods find proxies for the explanation quality (e.g., sparsity). The last kind of approaches require no costly human experiments but how to properly determine the proxy is a challenge.

In our taxonomy, explanations are divided into different types. Although the interpretability can hardly be compared between different types of explanations, there are some measurements proposed for this purpose. For logic rules and decision trees, the size of the extracted rule model is often used as a criterion (Tickle et al., 1998; Odajima et al., 2008; Wu, Hughes, et al., 2018) (e.g., the number of rules, the number of antecedents per rule, the depth of the decision tree etc.). Strictly speaking, these criteria measure more about whether the explanations are efficiently interpretable. Hidden semantics approaches produce explanations on certain hidden units in the network. Network Dissection (Bau, Zhou, et al., 2017) quantifies the interpretability of hidden units by calculating their matchiness to certain concepts. As for the hidden unit visualisation approaches, there is no a good measurement yet. For attribution approaches, their explanations are saliency maps/masks (or feature importance etc. according to the specific task). Samek, Binder, et al. (2016) evaluate saliency maps by the performance degradation if the input image is partially masked with noise in an order from salient to not salient patches. Adebayo et al. (2018) provide a similar evaluation method and Hooker et al. (2019) suggest using a fixed uninformative value rather than noise as the mask and evaluating performance degradation on a retrained model. Samek, Binder, et al. (2016) also use entropy as another measure in the belief that good saliency maps focus on relevant regions and do not contain much irrelevant information and noise. Montavon et al. (2018) would like the explanation function (which maps an input to a saliency map) to be continuous/smooth, which means the explanations (saliency maps) should not vary too much when seeing similar inputs.

3.7 Discussion

In practice, different interpretation methods have their own advantages and disadvantages. Passive (post-hoc) methods have been widely studied because they can be applied in a relatively straightforward manner to most existing networks. One can choose methods that make use of a network’s inner information (such as connection weights, gradients), which are usually more efficient (e.g., see Paragraph 3.4.3.1). Otherwise there are also model-agnostic methods that have no requirement of the model architecture, which usually compute the marginal effect of a certain input feature. But this generality is also a downside of passive methods, especially because there is no easy way to incorporate specific domain knowledge/priors. Active (interpretability intervention) methods put forward ideas about how a network should be optimised to gain interpretability. The network can be optimised to be easily fit by a decision tree, or to have preferred feature attribution, better tailored to a target task. However, the other side of the coin is that such active intervention requires the compatibility between networks and interpretation methods.

As for the second dimension, the format of explanations, logical rules are the most clear (do not need further human interpretation). However, one should carefully control the complexity of the explanations (e.g., the depth of a decision tree), otherwise the explanations will not be useful in practice. Hidden semantics essentially explain a subpart of a network, with most work developed in the computer vision field. Attribution is very suitable for explaining individual inputs. But it is usually hard to get some overall understanding about the network from attribution (compared to, e.g., logical rules). Explaining by providing an example has the lowest (the most implicit) explanatory power.

As for the last dimension, local explanations are more useful when we care more about every single prediction (e.g., a credit or insurance risk assessment). For some research fields, such as genomics and astronomy, global explanations are more preferred as they may reveal some general “knowledge”. Note that there is no hard line separating local and global interpretability. With the help of explanation fusing methods (e.g., MAME), we can obtain

multilevel (from local to global) explanations. However, the fusion from local explanations to global ones is usually computational intensive (as most of such methods employ some kinds of hierarchical clustering algorithms which generally have more than $O(n^2)$ complexity with respect to the number of samples). Moreover, the clustering is sensitive to specific experiment settings and the produced results (mediate-level explanations) are not unique.

3.8 Chapter Summary

In this chapter, we provide a comprehensive review of neural network interpretability. First, we discuss the definition of interpretability and stress the importance of the format of explanations and domain knowledge/representations. Specifically, there are four commonly seen types of explanations: *logic rules*, *hidden semantics*, *attribution* and *explanations by examples*. Then, we introduce a novel taxonomy for the existing network interpretation methods. It evolves along three dimensions: passive vs. active, types of explanations, and global vs. local interpretability. The last two dimensions are not purely categorical but with ordinal values (e.g., semi-local). This is the first time we have a coherent overview of interpretability research rather than many isolated categories and approaches. We can also visualise the distribution of the existing approaches in the 3D space spanned by our taxonomy.

CHAPTER 4

Model Performance and Motif Identifiability

In Chapter 3 we have seen the interpretability issue of deep neural networks, and all different kinds of methods to deal with it. As we summarise in Section 2.2, an important reason for interpretability is to help scientists to gain new knowledge. Computational biology is such a research field, which has already employed many deep learning techniques. Instead of a general form of interpretation such as rules or saliency maps, biologists have their domain-specific knowledge representations, for example, “motifs”, especially for biological sequence related tasks. Due to the limitation of current neural network interpretation methods (e.g., hard to interpret motifs from hierarchical convolutional kernels), the most widely used method is viewing the first-layer kernels as the learnt motifs. This becomes a new way of motif discovery, which relies on the high classification performance of the trained neural network, in contrast to the classic statistic based motif discovery tools. Then, a natural question is: how stable and reliable is such a motif discovery method? Or, will a well-performing neural network be able to identify the “true” underlying motifs?

In this chapter, we systematically study the motif identifiability problem in neural network based motif discovery, including the construction of “ground truth” synthetic datasets

containing known motifs to be identified, and also a novel motif similarity measure (in the context of convolutional kernels). We first introduce more details about the background and motivation behind this work in Section 4.1. Then in Section 4.2, we describe how the synthetic datasets and motif similarity are designed. Section 4.3 presents our findings in the experimental studies. Finally, we summarise this chapter in Section 4.4.

This chapter is part of the Work 2 (submitted to PLOS Computational Biology).

4.1 Background and Motivation

Benefiting from the rapid increase in both computational power and the volume of available data, deep neural networks have achieved record-breaking performance on various tasks such as image recognition and natural language processing ([Krizhevsky et al., 2012](#); [Sutskever et al., 2014](#); [LeCun, Bengio, et al., 2015](#); [Silver et al., 2017](#)). Convolutional neural networks (CNNs), a typical kind of deep neural networks, employ convolution kernels (aka filters), which can capture local patterns (motifs) that are invariant to position. This in theory makes them very suitable for sequence scanning and related prediction tasks, e.g., predicting sequence specificities of DNA- and RNA-binding proteins. Also, the use of high-throughput sequencing has greatly increased the amount of available sequence data, which is necessary for most deep learning methods.

There have been numerous applications of deep learning to genetic data in bioinformatics. DeepBind ([Alipanahi et al., 2015](#)) predicted transcription factor (TF) and RNA-binding protein binding from DNA and RNA sequences. DeepSEA ([Zhou and Troyanskaya, 2015](#)) predicted TF binding, DNase hypersensitivity and histone modifications from DNA sequences. There are more methods such as Basset ([Kelley et al., 2016](#)), DanQ ([Quang and Xie, 2016](#)) and so on (e.g., [He, Shen, et al., 2021](#)). The building blocks of above methods are adapted from CNNs used in the computer vision research. While in computer vision the input is usually a 2-D image, with three colour channels (R, G, B), in genomics the input is a 1-D sequence with four channels (A, C, G, T representing 4 nucleotides). Each

convolutional kernel scans through the input sequence, producing a “matching score” at each position, which is also known as a feature map. Those values are then fed to the following layers, which can be optional convolutional layers, pooling layers, and finally fully-connected layers to produce the final prediction (each layer is followed by a certain activation function, e.g., ReLU). Take DeepBind as an example, it has a single convolutional layer, a global maximum and average pooling layer, and a hidden fully-connected layer (or no hidden layer). The maximizing operation gives more attention to sequence patterns that are highly significant but may not be very frequent in the sequences, while the averaging helps to detect frequent but less discriminative motifs. The global pooling allows the network to handle sequences of variable length.

Although deep neural networks have shown great performance, they are largely black box models hiding their decision logic behind numerous real-valued weights and functional units they parametrise (Zhang, Tiño, et al., 2021). While this may not be an issue in some application domains, in a scientific discipline such as biology, it will be preferable not only to have (yet another) predictive model, but also be able to extract some insights from the machine learning exercise (e.g., relevant motifs in biological sequences). Motifs are commonly used to characterise sub-sequence specificities (of DNA- or RNA-binding proteins). They can be represented in different forms, e.g. as position weight matrices (PWMs), position frequency matrices (PFMs), position probability matrices (PPMs) and sequence logos (Stormo et al., 1982; Schneider and Stephens, 1990). Most neural network based methods (e.g., DeepBind) view the trained first-layer convolutional kernels as PWMs and then transform them into PPMs and sequence logos based on statistics extracted from a set of validation sequences (Figure 4.1). For a convolutional kernel and a set of input sequences, the subsequences (one for each sequence) which have the highest response are extracted. The PFM is then calculated by counting the nucleotide frequencies per position and is finally transformed to a PPM.

Given the framework of motif discovery through low-level convolutional kernels in neural networks trained on specific tasks, the natural question to ask is: *If our guide for extracting*

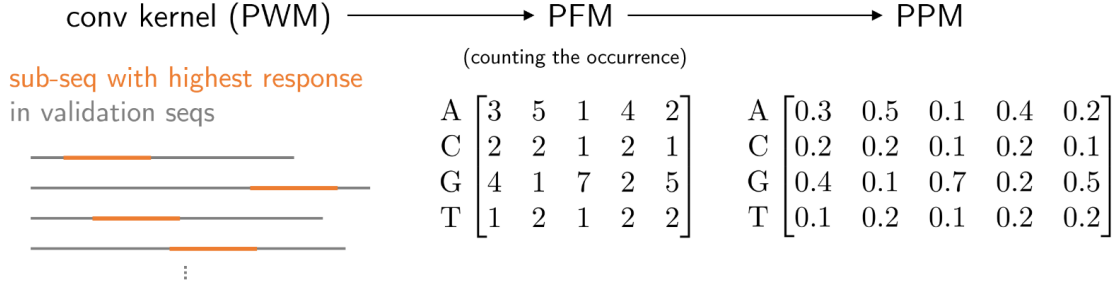


Figure 4.1: From convolutional kernel to PPM.

motifs of interest is the (out-of-sample) network performance, how stable and reliable is such a motif extraction? In other words, we would like to study the neural network based motif discovery as a *motif identifiability problem*. Is it really the case that a well-performing neural network necessarily yields the “true” underlying motifs?

Optimizing a neural network (architecture) for extreme performance (e.g., classification accuracy) can lead to an over-complex model in terms of extracted sequential motifs. For example, [Zeng et al. \(2016\)](#) showed that a simple neural network with 16 convolutional kernels was already reasonably good, but slight improvement could still be made as long as we added more kernels (up to 128 kernels). Having too many kernels makes it harder to identify possible motifs, not only because users need to check more candidate motifs, but also because of a concern that a motif may be jointly represented by multiple kernels ([Eraslan et al., 2019](#); [Shrikumar, Tian, et al., 2018](#)). [Koo and Eddy \(2019\)](#) showed that the size of convolutional kernels and pooling layers would affect the representation of motifs, which can get distributed in multiple kernels or within a single kernel. Note that it is possible for two networks to be functionally equivalent (having the same predictive behaviour) ([Sundararajan et al., 2017](#)), yet have different inner weights (including the convolutional kernel ones). It is found that multilayer neural networks have many local minima which are all more or less equivalent in terms of the test performance ([Choromanska et al., 2015](#)). Thus, it can be risky to rely on a single good-performing model for motif discovery considering the randomness of neural network training. Besides the CNN architecture, some found that adding recurrent network architectures may slightly improve the model performance, however, it will result

in uninformative learnt motifs (Trabelsi et al., 2019).

Unlike previous work (Koo and Eddy, 2019) which asks how CNN architecture (specifically, max-pooling and convolutional kernel size) affects the motif representation (partial or whole) in first-layer kernels, we formally study the motif identifiability problem in neural network based motif discovery with a novel motif similarity measure (in the context of convolutional kernels) and carefully constructed “ground truth” synthetic datasets containing known stochastic sequential motifs to be identified. There is also a related work (Villaverde et al., 2016) about *structural identifiability* of biology models (not neural networks), which cares about whether the model parameters can be uniquely identified given infinite and noise-free data, while we study whether the motifs (not merely parameters) can be identified under a more practical setting (neural network based motif discovery). We first introduce some definitions and the proposed methods that will be used for the motif identifiability problem. Experiments on different synthetic datasets are then presented, where we systematically investigate the relationship between motif identifiability and model performance/complexity.

4.2 The Proposed Methods

4.2.1 Probabilistic Motifs and Motif Similarity

We consider a probabilistic motif $\mathbf{P} = (P_1, P_2, \dots, P_\ell)$ of length ℓ as a collection of position-dependent probability distributions $P_i(\cdot)$ over nucleotides $\{A, G, C, T\}$, stamped at positions $i = 1, 2, \dots, \ell$. For ease of exposition we identify the nucleotides A, G, C and T with outcomes 1, 2, 3 and 4, respectively, so that the distributions $P_i(j)$, $j \in \mathcal{N} = \{1, 2, 3, 4\}$, live on the standard 3-simplex

$$\mathcal{S} = \{Q = (Q(1), Q(2), Q(3), Q(4)) \mid Q(j) \geq 0, \sum_{j=1}^4 Q(j) = 1\}. \quad (4.1)$$

One can impose a pair-wise similarity measure on \mathcal{S} , $s(Q, Q')$, $Q, Q' \in \mathcal{S}$, using the Jensen-Shannon divergence $\text{JS}(Q \parallel Q')$,

$$s_{\text{JS}}(Q, Q') = 1 - \sqrt{\text{JS}(Q \parallel Q')}, \quad Q, Q' \in \mathcal{S}, \quad (4.2)$$

where JS divergence is defined based on Kullback–Leibler divergence ([Kullback and Leibler, 1951](#)),

$$\text{JS}(Q \parallel Q') = \frac{1}{2} \left(\text{KL}\left(Q \parallel \frac{Q + Q'}{2}\right) + \text{KL}\left(Q' \parallel \frac{Q + Q'}{2}\right) \right). \quad (4.3)$$

The JS divergence has several advantages such as symmetry and being bounded (between 0 and 1), whose square root is a proper distance metric. It has already been used for genome comparison ([Sims et al., 2009](#)). Another natural alternative is the geodesic distance under the Fisher information metric on \mathcal{S} ([Lebanon and Lafferty, 2005](#)),

$$d_{\text{FI}}(Q, Q') = 2 \arccos \left(\sum_{j=1}^4 \sqrt{Q(j) \cdot Q'(j)} \right), \quad (4.4)$$

giving the corresponding similarity measure

$$s_{\text{FI}}(Q, Q') = 1 - \frac{d_{\text{FI}}(Q, Q')}{\pi}. \quad (4.5)$$

Other than common motif alignment tools such as TOMTOM ([Gupta et al., 2007](#)) and STAMP ([Mahony and Benos, 2007](#)), we use a specifically designed similarity measure between two probabilistic kernels for our data, which takes into account (1) the possible horizontal shift between two kernels, and (2) explicit handling of the “wildcard” symbols (termed uninformative columns in [Tanaka et al., 2011](#)).

Consider now two probabilistic motifs $\mathbf{P} \in \mathcal{S}^{\ell_P}$ and $\mathbf{T} \in \mathcal{S}^{\ell_T}$ of length ℓ_P and ℓ_T , respectively. In our synthetic data experiments, we will use neural networks to extract probabilistic motifs \mathbf{P} from the data generated using known ground truth (target) probabilistic motifs \mathbf{T} . The target motifs may have some wildcard (uninformative) positions where all symbols from \mathcal{N} are equally likely and hence the corresponding positional distribution will be the uniform

distribution U over \mathcal{N} .

We define the (asymmetric) similarity $S(\mathbf{T}, \mathbf{P})$ from \mathbf{T} to \mathbf{P} as:

$$S(\mathbf{T}, \mathbf{P}) = \max_o \left[\underbrace{\frac{|I|}{F} \cdot \frac{1}{|I|} \sum_{i \in I} s(T_i, P_{i-o})}_{\text{matching score}} - \underbrace{\frac{1}{|J|} \sum_{j \in J} (1 - s(P_j, U))}_{\text{overfitting penalty}} \right], \quad (4.6)$$

where I, J are index sets for the matched positions of \mathbf{T} and the unmatched positions of \mathbf{P} (note the asymmetry here),

$$I = \{i \in \mathbb{N} \mid 1 \leq i \leq \ell_T, 1 \leq i - o \leq \ell_P, T_i \text{ is not a wildcard distribution}\} \quad (4.7)$$

$$J = \{j \in \mathbb{N} \mid 1 \leq j \leq \ell_P, (j + o > \ell_T) \text{ or } (T_{j+o} \text{ is a wildcard distribution})\}, \quad (4.8)$$

and $F = \min\{|I| + 1, \ell_T\}$. Figure 4.2 provides a graphical explanation.

The maximizing operation considers all possible (\mathbf{P} to \mathbf{T}) position offsets o which satisfy $-(\ell_P - 1) \leq o \leq \ell_T - 1$. Note that the offset o can be positive or negative, and the aligned position indices have this relationship, $j = i - o$ (and $i = j + o$ in a reverse manner). The matched positions of \mathbf{T} are indexed by I , and the conditions (4.7)–(4.8) make sure that (1) i is a valid position in \mathbf{T} , (2) $j = i - o$ is a valid position in \mathbf{P} , and (3) T_i is not a wildcard distribution. The index set J is similar but collects all the unmatched positions of \mathbf{P} or matched wildcards of \mathbf{T} . The matching score is the sum of similarities of all pairs of offset-matched positional probability distributions T_i and P_{i-o} , normalized by F . The normalisation factor F ensures a weak *underfitting* correction factor in case \mathbf{P} matches only a few positions in \mathbf{T} . When there is only one matched position ($|I| = 1$), the factor $|I|/F$ is $1/2$, which will gradually get closer to 1 if $|I|$ increases (until $|I| = \ell_T - 1$). The *overfitting* penalty makes sure P_j is close to uniform distribution U when there should not be any pattern.

The (asymmetric) similarity score $S(\mathbf{T}, \mathbf{P})$ can be used to quantify how similar is an extracted probabilistic motif \mathbf{P} to a known ground truth (target) motif \mathbf{T} . This will be used in our synthetic data experiments. To compare two extracted probabilistic motifs \mathbf{P} and \mathbf{P}' ,

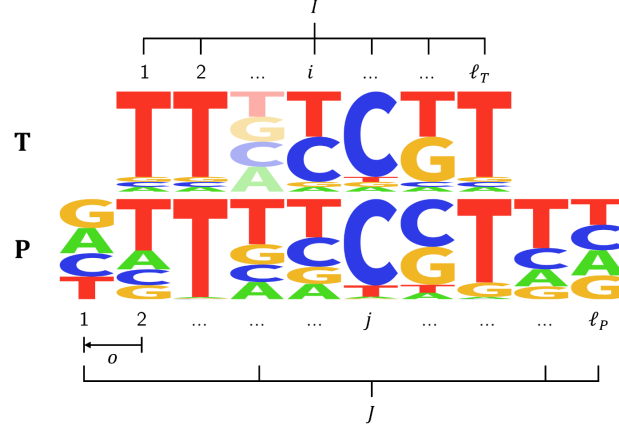


Figure 4.2: An illustration of the proposed (asymmetric, from **T** to **P**) similarity measurement.

a symmetric measure is obtained by symmetrizing S ,

$$S_{\text{sym}}(\mathbf{P}, \mathbf{P}') = \frac{1}{2}(S(\mathbf{P}, \mathbf{P}') + S(\mathbf{P}', \mathbf{P})). \quad (4.9)$$

In this case, a positional distribution P_i (or P'_i) is considered a wildcard distribution if its entropy (base 2) is larger than a threshold θ . We set $\theta = 1.95$.

4.2.2 Synthetic Data Generation

In order to systematically study what affects motif identifiability, we need flexible and controllable datasets with known “ground truth”. For example, previous work (Koo and Eddy, 2019) generates a synthetic sequence by embedding 1 to 5 motifs, which are chosen from a pool of 12 known transcription factors, in a “null” sequence (uniform distribution over 4 nucleotides on all positions). Each sequence has a zero-one vector of length 12 as the label, according to the presence of the 12 transcription factors. This process is repeated many times to get the full synthetic dataset. Here we adopt a different approach: (1) Instead of collecting motifs from the known database, we explicitly model the motif probability distributions, based on which we can create motifs that have controllable properties but are stochastic in their nature; (2) We define sequence classes using *relationships between the probabilistic*

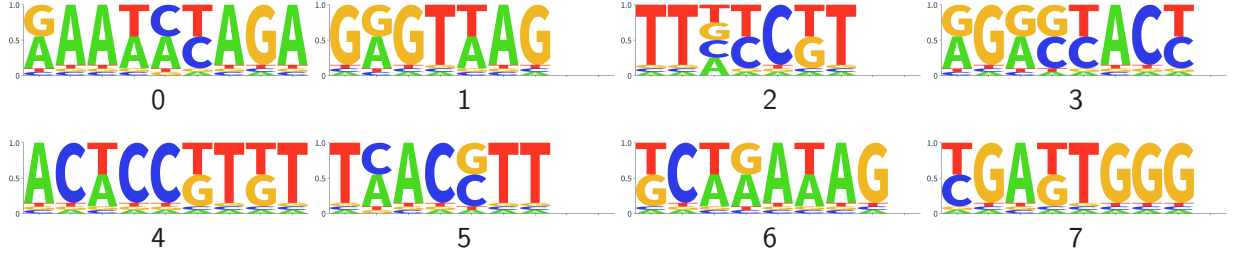


Figure 4.3: The 8 ground-truth motifs used in the synthetic datasets. The y -axis is the accumulated probability.

motifs, rather than their existence/absence only. Our generation process consists of three steps: motifs generation, construction of classification rules (class definitions), and sequences generation. As explained in the previous sections, a motif \mathbf{P} is mathematically a collection of position-dependent probability distribution $P_i(\cdot)$ over $\{A, G, C, T\}$ (or index set \mathcal{N} for convenience), where $i \in \{1, 2, \dots, \ell\}$ is the position index, ℓ is the length of the motif. We allow the distribution P_i to be one of the following 3 cases:

- Wildcard (uniform distribution): $P(j) = 0.25$ for all symbol $j \in \{A, G, C, T\}$.
- One dominant symbol: $P(j) = \alpha$ if j is the dominant symbol, otherwise $P(j) = \frac{1-\alpha}{3}$ ($0.5 < \alpha < 1$). The dominant symbol is randomly selected from \mathcal{N} for each position with a single dominant symbol.
- Two dominant symbols: The two dominant symbols j_1 and j_2 are randomly selected from \mathcal{N} for each position with two dominant symbols. $P(j_1) = P(j_2) = \beta$, $0.33 < \beta < 0.5$. For the other two (non-dominant) symbols $P(j) = (1 - 2\beta)/2 = 0.5 - \beta$.

In this work, we set $\ell \in \{7, 8, 9\}$, $\alpha = 0.85$ and $\beta = 0.45$. The symbol distribution P_i of each position in the motif is determined randomly with the probability of the Wildcard, one dominant symbol, and two dominant symbols set to 0.1, 0.6 and 0.3 respectively. In the motif generation, we do not allow “wildcard” at either the start or the end of a motif. Fig 4.3 (top row) shows four synthetic motifs used in one of the datasets employed in our study, noting that the third motif has a wildcard symbol at position 3.

Algorithm 1: Generate synthetic motifs

Input: number of motifs n , maximum length of a motif ℓ_{\max} , minimum length of a motif ℓ_{\min} , α , β
Output: a group of n motifs $\mathcal{G} = \{\mathbf{P}^{(1)}, \mathbf{P}^{(1)}, \dots, \mathbf{P}^{(n)}\}$
 $\mathcal{G} \leftarrow$ an empty list
while SIZE(\mathcal{G}) < n **do**
 $\ell \leftarrow$ random integer in $[\ell_{\min}, \ell_{\max}]$ ▷ generate the motif length
 $\mathbf{p} \leftarrow$ a vector of ℓ random probabilities in $[0, 1)$
 if $p_1 < 0.1$ **or** $p_\ell < 0.1$ **then continue** ▷ avoid wildcard in both ends
 foreach p_i **in** \mathbf{p} **do**
 if $p_i < 0.1$ **then**
 $P_i \leftarrow$ uniform distribution U over 4 nucleotides
 else if $p_i < 0.1 + 0.6$ **then**
 $P_i \leftarrow$ distribution for one dominant symbol
 else
 $P_i \leftarrow$ distribution for two dominant symbols
 end
 end
 $\mathbf{P} \leftarrow (P_1, P_2, \dots, P_\ell)$
 add \mathbf{P} to \mathcal{G}
end
return \mathcal{G}

Using different combinations of motifs, we can construct different synthetic classes with logical AND and OR operations. Table 4.1 shows three datasets and the corresponding classification rules used in this thesis. Dataset 2 is similar to dataset 1 with only the motifs swapped, while dataset 3 contains more classes and more complex classification rules. After deciding which motifs should be present in each class, we also need to decide where those motifs should appear. Each motif has its own independent position distribution in each class. In our experiments, we use Gaussian distributions (truncated in a finite interval, the sequence length, and normalised), whose means μ were randomly initialised while the standard deviations σ were set to a constant 1. Take dataset 1 as an instance, sequences in class 1 should have motif 0 at around position 84 and motif 1 around position 53. During motif position generation, overlapping between motifs is not allowed.

Having above motifs and classification rules, we can generate sequences accordingly. For

Table 4.1: Classification rules of synthetic datasets. The numbers are the indices of motifs (0-based). Symbol \wedge means logical AND operation, while \vee means logical OR.

	Class 0	Class 1	Class 2	Class 3	Class 4
Dataset 1	0	$0 \wedge 1$	$2 \vee 3$	—	—
Dataset 2	2	$2 \wedge 3$	$0 \vee 1$	—	—
Dataset 3	0	$0 \wedge 1$	$2 \vee 3$	$1 \wedge 3 \wedge 5$	$(4 \wedge 5) \vee (6 \wedge 7)$

each class, $m = 20000$ sequences were generated, each of which is 200 nt (nucleotide) long just as in [Koo and Eddy \(2019\)](#). The symbol at each position is randomly chosen from the uniform distribution unless there is a motif. For a class having OR operator, all subcases have the same probability. Specifically, for class “motif 2 OR 3”, the probabilities of a sequence to have “only motif 2”, “only 3” and “2 and 3” are equally 1/3. The full process of sequence generation (for each class) is described in Algorithm 2.

4.3 Experimental Studies

4.3.1 Experiment Setup

Datasets. The first and second datasets have 3 classes each, while the third dataset has 5 classes (see Table 4.1). Neural networks are trained to differentiate sequences that come from different classes. We randomly sampled 60% of the data as the training set, 20% as the validation set, and 20% as the test set. The validation set is used for hyperparameter selection. The test set is never accessed during the training or model selection process and only used to evaluate the final out-of-sample performance.

Network architectures. The used neural network has a similar architecture as in DeepBind ([Alipanahi et al., 2015](#)) and [Zeng et al. \(2016\)](#), which have one convolutional layer with ReLU activation function, optional pooling layers, and fully-connected layers. Dropout layers, which are usually used in large neural networks to prevent overfitting, are not used

Algorithm 2: Generate synthetic sequences from motifs and a classification rule

Input: number of sequences (of one class) m , the length of a sequence L , motifs $\mathcal{G} = \{\mathbf{P}^{(1)}, \mathbf{P}^{(2)}, \dots, \mathbf{P}^{(n)}\}$, a classification rule r , means of all the motif distributions $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_n)$
Output: a group of m sequences of length L
 $\mathcal{Q} \leftarrow$ an empty list
for $i = 1$ **to** m **do**
 resolve rule r into motif index set \mathcal{J}
 \triangleright include all AND motifs and turn ‘a OR b’ into ‘a’ or ‘b’ or ‘a AND b’ with equal prob.
 $\mathbf{S} \leftarrow (U_1, U_2, \dots, U_L)$ \triangleright initialise sequential probabilistic model with uniform distribution
 foreach j **in** \mathcal{J} **do**
 generate loc based on normal distribution $N(\mu_j, 1)$
 replace $\mathbf{S}[loc : loc + \ell_{\mathbf{P}^{(j)}}]$ with $\mathbf{P}^{(j)}$, which is $(P_1, P_2, \dots, P_{\ell_{\mathbf{P}^{(j)}}})$
 end
 $\mathbf{p} \leftarrow$ a vector of L random probabilities in $[0, 1)$
 generate sequence seq by comparing \mathbf{p} with \mathbf{S}
 add seq to \mathcal{Q}
end
return \mathcal{Q}

here. This is because there is no obvious performance difference of using it or not in our experiments, and the networks do not suffer from severe overfitting. A possible reason is our networks are expected to be relative small (see Chapter 5) and have enough training data, where common weight decay (L^2 norm) regularisation is good enough. DeepBind has a global (maximum and average) pooling layer as it needs to deal with variable length input. Here for synthetic datasets, we simply omit the pooling layer as all the sequences have the same length. It can also be viewed as a pooling layer with filter size 1.

Hyperparameters. In order to study the relationship between network performance and motif identifiability, we need a variety of networks of different complexity, where the number of convolutional kernels is a good choice. Ideally, the population of neural networks should include both inferior ones which struggle to fit the data, and also larger ones which are quite capable to do it. For synthetic dataset 1 and 2 which have 4 motifs, we try from 4 to 9 convolutional kernels. And for synthetic dataset 3 which has 8 motifs, we from 7 to 14

Table 4.2: The hyperparameters of neural networks in the experiments.

Hyperparameter	Possible values
Network architecture	
Number of hidden neurons	16, 32, 64, 32+16 ^a
Optimisation	
Batch size	16, 32
Learning rate	0.01, 0.05, 0.1
Weight decay (L^2 norm)	0, 0.001, 0.005, 0.01

^a 2 hidden layers with 32 and 16 neurons each.

kernels.

For each given number of convolutional kernels, we explore various hyperparameter combinations and record the one which has the best performance on validation set (5-run average). Based on the experimental setup from the previous work [Zeng et al., 2016](#) and a few initial preliminary experiments, we narrowed down the search space to a relatively small region, on which the hyperparameters were fine-tuned. Table 4.2 shows the hyperparameters search table.

Optimisation. Categorical cross entropy is used as the loss function of the multi-class classification. All neural networks are optimised with stochastic gradient descent with weight decay (L^2 norm regularisation). The batch size is chosen from 16 and 32. Note that for the real datasets, the sequences have variable length and technically the batch size is 1. We accumulate the gradients and update the model once every 16 or 32 batches. Gradient accumulation is a common technique and is also used in other situations, for example, when the batch size is limited by the available GPU memory.

The learning rate is controlled with a common “reduce on plateau” scheme, which reduces the learning rate by a factor (we use 2) if a chosen metric has stopped improving on validation set for a certain number of epochs (i.e., “patience”). We also use early stopping to end the training process when the performance on validation set has stagnated for a long time or

even gets worse and worse. The patience for learning rate reduction is 6 epochs, and is 15 epochs for early stopping. Every time the learning rate is reduced, we add 3 more epochs to the early stopping patience.

Evaluation. The performance of a neural network is evaluated by its average accuracy among all classes. The evaluation of motif identifiability is based on the proposed (asymmetric) similarity (Equation 4.6). For each ground-truth motif, we calculate its similarities to all the learnt kernels of a trained neural network and record the highest match.

4.3.2 The Motif Identifiability Problem

We first take synthetic dataset 1 for demonstration. In order to investigate the relationship between motif identifiability and model performance, we trained neural networks with from 4 to 9 convolutional kernels (50 runs each), again using their own best hyperparameters respectively. Given the ground truth of the synthetic data, we can calculate to what extent a neural network (’s kernels) recovers the true motifs (i.e., the motif identifiability), using the similarity measure introduced above. In Figure 4.4, every data point represents a single trained neural network, whose x value is the network’s classification accuracy, and the y value is the highest (asymmetric) similarity among its kernels to a certain true motif. (Unless specifically stated, all the following figures show the results using JS-divergence based motif similarity, as the results for the geodesic similarity are almost the same.) It can be seen that in order to get the best identifiability, a neural network is required to achieve high performance (greater than 94% here). This means top performance is a *necessary* condition of good identifiability. However, top performance is unlikely to be a *sufficient* condition as the identifiability still varies in a wide range (e.g., 0.6 to 0.9) for those networks having more than 94% accuracy. Figure 4.5 provides some example kernels which have different levels of similarity to the true motif 0. While the 0.91 kernel is really close to the true motif, the 0.70 kernel obviously converges to a local-minimum pattern (it has two overlapped AGA in the ending rather than only one in the true motif). The 0.67 kernel does not recover the original



Figure 4.4: Motif identifiability vs. model performance (synthetic dataset 1). The data points are from neural networks with 4 to 9 kernels (50 runs each). For each ground-truth motif, we identify a most similar kernel from every run (i.e., a network) among its learnt convolutional kernels and make the scatter plot.

motif very well. We now introduce a quantitative measure of a model’s motif identifiability for further study.

4.3.3 Quantitatively Evaluate Model identifiability with AU-CDF

Due to the stochastic nature of neural network training, it does not make sense to evaluate a model’s identifiability with a single instance. We thus propose the *area under the cumulative density function* (AU-CDF) to measure a model’s identifiability of a certain motif, based

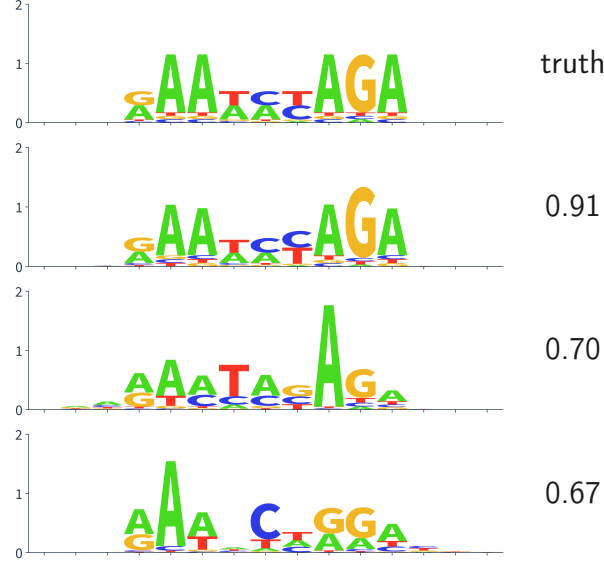


Figure 4.5: Visualisation of kernels with different similarities to motif 0. The motifs are shown in sequence logos (Stormo et al., 1982; Schneider and Stephens, 1990) for clear comparison. The y -axis is the accumulated probability multiplied with the information content, $2 - \text{entropy}$, at each position.

on a population of multiple runs (training) of the model. Given a set of identifiability-performance data points, its underlying distribution can be estimated by kernel density estimation. Kernel density estimation is a non-parametric way to estimate the probability density function (PDF) of a random variable based on a finite set of data points. Assuming $\{x_1, x_2, \dots, x_n\}$ are independent and identically drawn examples of a random variable X under an unknown distribution p , the kernel density estimator is

$$\hat{p}(X = x) = \frac{1}{n} \sum_{i=1}^n \frac{1}{h} K\left(\frac{x - x_i}{h}\right), \quad (4.10)$$

where kernel K is a non-negative function (not a convolutional kernel of a neural network as discussed elsewhere in this paper), and h is bandwidth acting as a smoothing parameter. We adopted a common choice of K which is a standard Gaussian $K(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})$ and set $h = 0.2$. The CDF of a probability density function $p(x)$ is calculated by

$$f(x) = \int_{-\infty}^x p(t) dt. \quad (4.11)$$

Given results of many neural networks that have different performance, we are interested in the marginal distribution of their identifiability values, which are expected to be as close to 1 as possible. In other words, the AU-CDF should be as small as possible (as $p(x)$ should be small when $x < 1$).

4.3.4 Accurate Models Tend to Have Better Motif Identifiability

Now we can do quantitative comparison of motif identifiability between different models. In Figure 4.6 we present the same data as in Figure 4.4 but split the data points into two equal-size sets (high/low) based on their accuracy. (Note that the “low” accuracy is relative. They are both well-performing and have around 1% difference at most.) Their probability density functions (PDFs) are estimated separately and visualised to the right. The coloured numbers in the bottom right corners are the corresponding AU-CDF values (less is better). It can be found that the high-accuracy set has denser distribution among higher identifiability values, which means accurate models are more likely to recover the true motifs better. Table 4.3 shows the AU-CDF values (high/low-accuracy) for all motifs under two similarity measures (JS-based and geodesic) in detail. Kolmogorov–Smirnov test (K-S test) is used to see whether the difference between two distributions is significant. From the table we can see high-accuracy models consistently surpass low-accuracy models in terms of the motif identifiability, and the p -values suggest that this result is not by accident.

4.3.5 The Sweet Spot of the Number of Convolutional Kernels

In previous section, we have seen that top performance is a *necessary* condition of good motif identifiability. However, it does not seem to be a *sufficient* condition. A more accurate model usually requires a more complex architecture, causing a concern that the motifs are represented in a more complicated way which is not easily interpretable (e.g., a motif may be jointly represented by multiple kernels (Eraslan et al., 2019)). To investigate how model complexity affects motif identifiability, we choose the number of convolutional kernels as

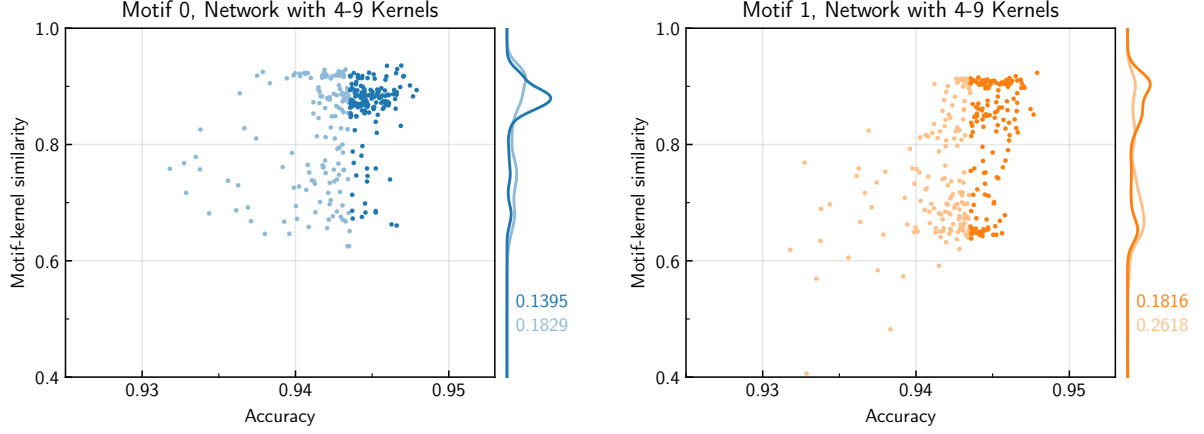


Figure 4.6: Marginal identifiability kernel density estimation for data points of high/low accuracy. The coloured numbers are the AU-CDF values (less is better).

Table 4.3: The AU-CDF of marginal interpretability distribution for high-accuracy and (relatively) low-accuracy models (synthetic dataset 1).

Motif	JS-based similarity		Geodesic similarity	
	AU-CDF ¹	p -value ²	AU-CDF	p -value
0	0.1395 (<) 0.1829	8.9×10^{-8}	0.1268 (<) 0.1675	8.9×10^{-8}
1	0.1816 (<) 0.2618	3.1×10^{-12}	0.1659 (<) 0.2423	3.1×10^{-12}
2	0.4162 (<) 0.4413	7.2×10^{-3}	0.3941 (<) 0.4180	1.1×10^{-2}
3	0.2700 (<) 0.3247	2.4×10^{-5}	0.2496 (<) 0.3010	4.2×10^{-5}

¹ Each cell contains two values representing results from high/low-accuracy data points as shown in Figure 4.6.

² K-S test is used to see whether high/low distributions are significantly different.

the control variable. Figure 4.7 contains 4 sub-figures showing respectively the identifiability distribution for neural networks that have 4 to 7 kernels. The AU-CDF value (identifiability) for each number of kernels is calculated and plotted in Figure 4.8, where each empty circle indicates the minimum value (i.e., best identifiability) for that motif. For synthetic dataset 1, the best identifiability is achieved with 5 kernels, slightly larger than the number of true motifs, 4. This may vary for different datasets and motifs, for example, the best number of kernels is 5 or 6 (for different motifs) in synthetic dataset 2 (see later section). Although the

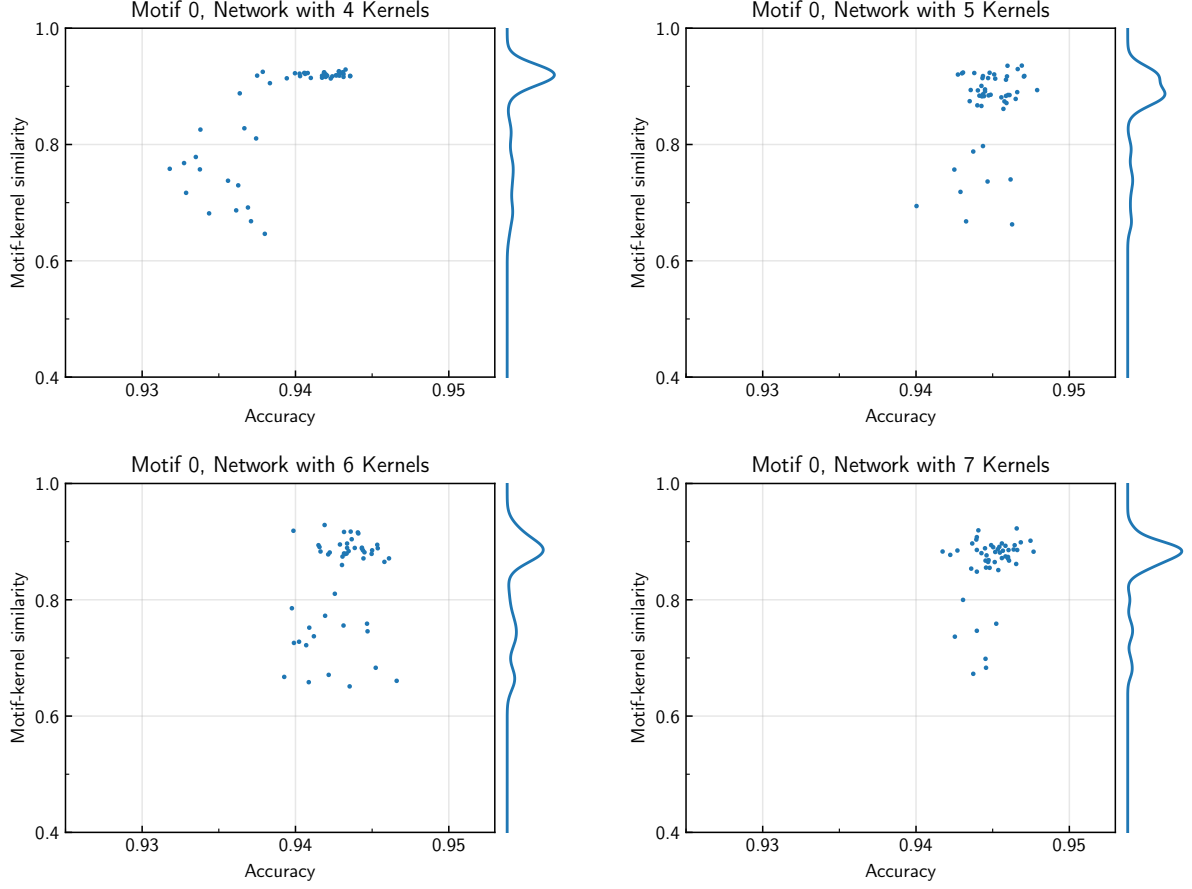


Figure 4.7: Identifiability-performance distributions under different numbers of kernels.

results are slightly different, they both show that the motif identifiability does deteriorate if the model becomes overcomplicated.

4.3.6 Experiments on Synthetic Dataset 2 and 3

We also perform the same experiments on other datasets to see whether above findings are tied to specific dataset properties. Synthetic dataset 2 uses the same motifs as in dataset 1 but changes the classification rules by swapping the roles of motif 0 and 2, 1 and 3 respectively. Dataset 3 has more motifs, classes and more complex classification rules (nested logical combinations). On each dataset, we repeat above experiments and the results are presented as below.

For dataset 2, Figure 4.9 shows the motif identifiability vs. model performance scatters.

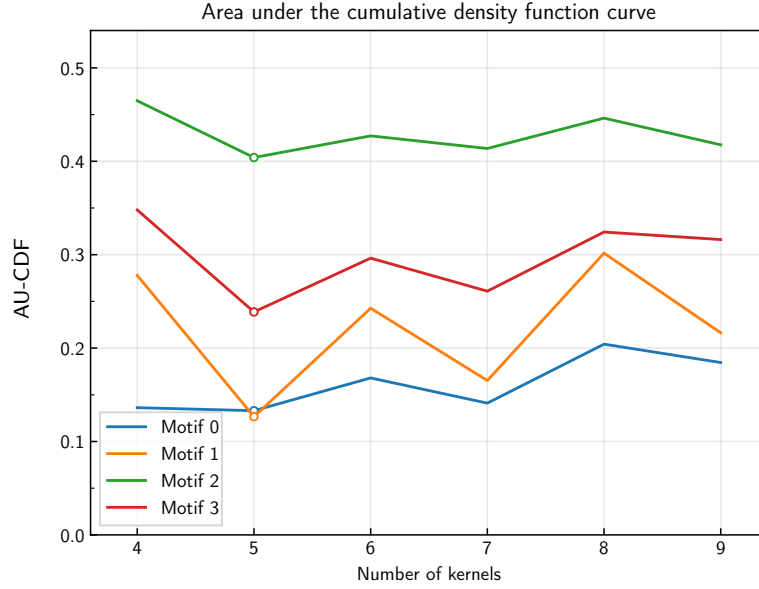


Figure 4.8: The AU-CDF per number of kernels (synthetic dataset 1). Less is better. The empty circle indicates the minimal value in the line.

Table 4.4: The AU-CDF of marginal interpretability distribution (synthetic dataset 2). Please see Table 4.3 for details.

Motif	JS-based similarity		Geodesic similarity	
	AU-CDF	p -value	AU-CDF	p -value
0	0.1292 (<) 0.1376	0.18	0.1171 (<) 0.1249	0.18
1	0.1268 (<) 0.1589	4.2×10^{-5}	0.1150 (<) 0.1450	4.2×10^{-5}
2	0.2840 (<) 0.3240	3.4×10^{-7}	0.2711 (<) 0.3089	1.8×10^{-7}
3	0.1363 (<) 0.2522	2.1×10^{-20}	0.1249 (<) 0.2331	2.1×10^{-20}

An obvious difference (from that of dataset 1) is motif 2 and 3 become much more identifiable, while motif 0 and 1 remain roughly the same. The improvements on motif 2 and 3 are as expected, because they are emphasised by the classification rules (class 0 is ‘2’ and class 1 is ‘2 \wedge 3’, see Table 4.1). Logical AND requires both motifs to appear while OR is much looser. Motif 0 and 1, although having different length, are already very identifiable by themselves. They both have 5 one-symbol-dominating positions, while motif 2 and 3 only have 4 and 3 such positions respectively (see Figure 4.3). Table 4.4 shows the AU-CDF

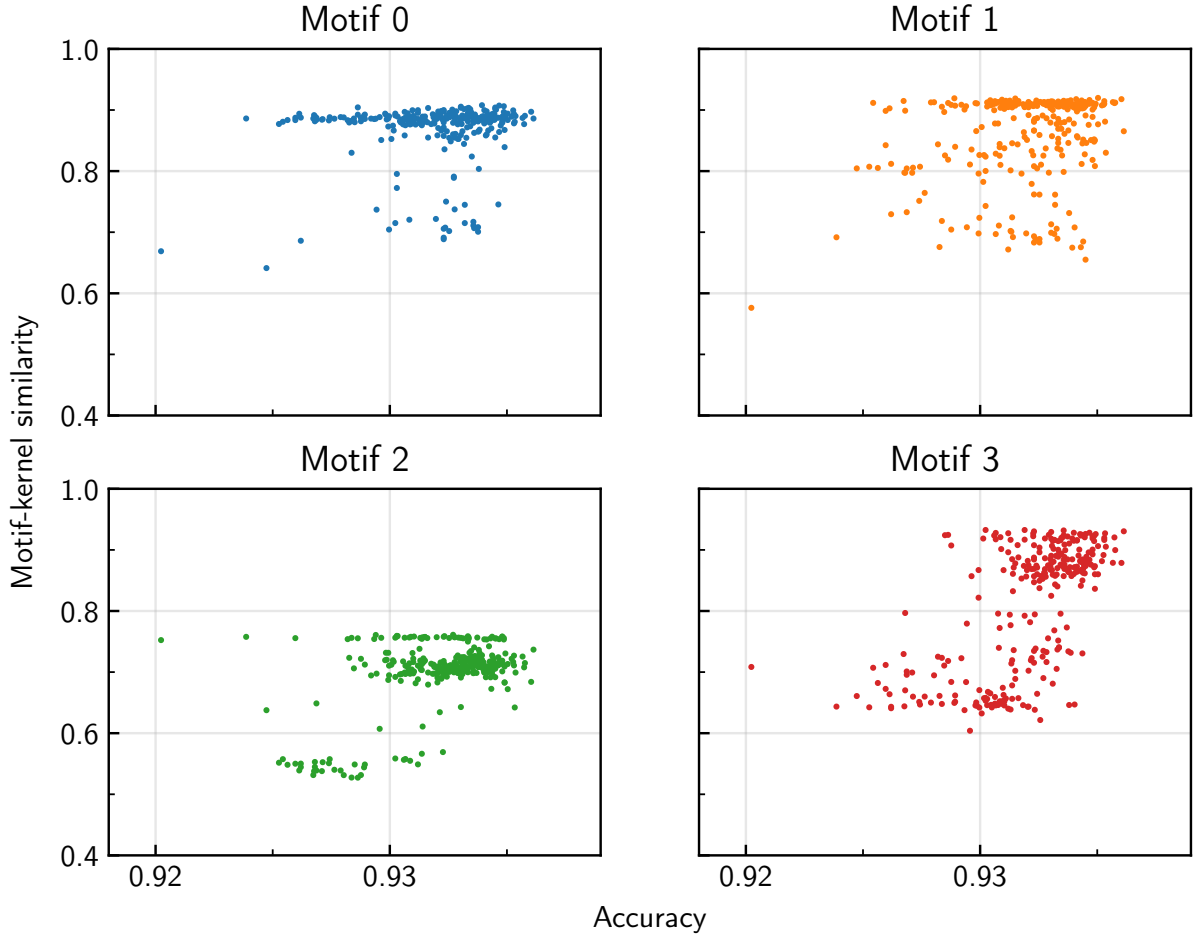


Figure 4.9: Motif identifiability vs. model performance (synthetic dataset 2). Please see Figure 4.4 for details.

values, the population based motif identifiability measure, for high/low-accuracy models (synthetic dataset 2), which is in support of the previous finding that accurate models tend to have high motif identifiability (lower AU-CDF values). The AU-CDF is also examined per number of convolutional kernels (see Figure 4.10). Similarly, the best number of kernels is still 5 (occasionally 6 for motif 3). This indicates that the balance between the model complexity (which has a marginally decreasing positive effect on the model performance) and the motif identifiability is likely to appear near the number of true motifs.

In the more complex synthetic dataset 3 which has 5 classes and 8 true motifs, we trained neural networks with from 7 to 14 convolutional kernels (50 runs each). The AU-CDF values

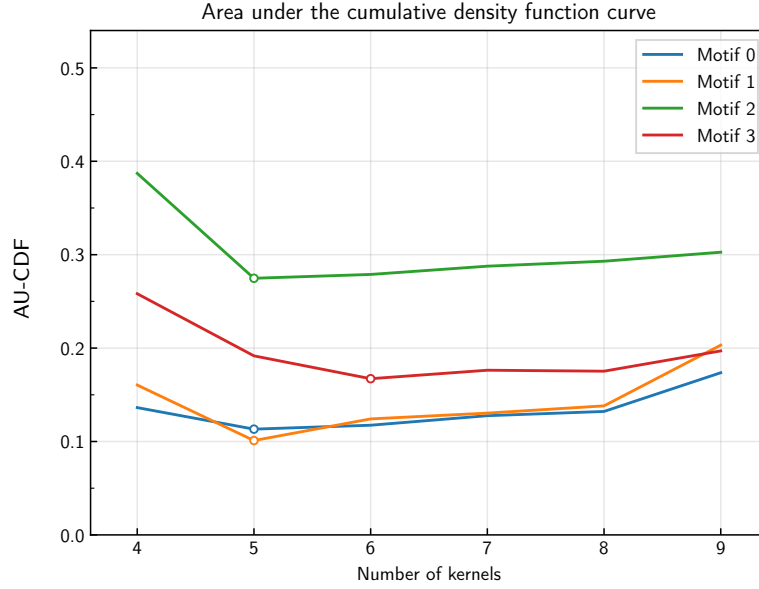


Figure 4.10: The AU-CDF per number of kernels (synthetic dataset 2). Less is better. The empty circle indicates the minimal value in the line.

of high-accuracy and low-accuracy networks for all 8 motifs are shown in Table 4.5. It again confirms that accurate models tend to have better motif identifiability. However, in terms of the best number of kernels (for best identifiability), the result is a little different (Figure 4.11). A neural network with about 8 kernels, which is the number of true motifs, is preferred to identify motifs 0 to 3, but motifs 4 to 7 will be better identified when there are more kernels. This may be because of the classification rules (Table 4.1) that the first 4 motifs are already able to describe 4 out of 5 classes (class 0, 1, 2, 3), and having more kernels gives more chances to learn the last 4 motifs in the remaining class 4.

4.4 Chapter Summary

In this chapter, we systematically study the relationship between the model performance and motif identifiability in the neural network based motif discovery method. We first give the mathematical formulations of probabilistic motifs, similarity between nucleotide probability distributions, and inter-motif similarity (asymmetric and symmetric). Then three different

Table 4.5: The AU-CDF of marginal interpretability distribution for high-accuracy and (relatively) low-accuracy models (synthetic dataset 3). Please see Table 4.3 for details.

Motif	JS-based similarity		Geodesic similarity	
	AU-CDF	p -value	AU-CDF	p -value
0	0.1569 (<) 0.1765	3.2×10^{-3}	0.1423 (<) 0.1610	1.6×10^{-3}
1	0.1906 (<) 0.2095	7.7×10^{-4}	0.1741 (<) 0.1919	7.7×10^{-4}
2	0.4042 (<) 0.4131	6.1×10^{-2}	0.3834 (<) 0.3919	4.8×10^{-2}
3	0.1640 (<) 0.2307	4.2×10^{-16}	0.1498 (<) 0.2129	4.2×10^{-16}
4	0.2335 (<) 0.2930	1.1×10^{-8}	0.2134 (<) 0.2707	1.1×10^{-8}
5	0.2470 (<) 0.2828	5.2×10^{-4}	0.2270 (<) 0.2606	5.2×10^{-4}
6	0.1794 (<) 0.2520	2.7×10^{-10}	0.1644 (<) 0.2345	2.7×10^{-10}
7	0.2513 (<) 0.2701	1.1×10^{-3}	0.2330 (<) 0.2512	2.3×10^{-3}

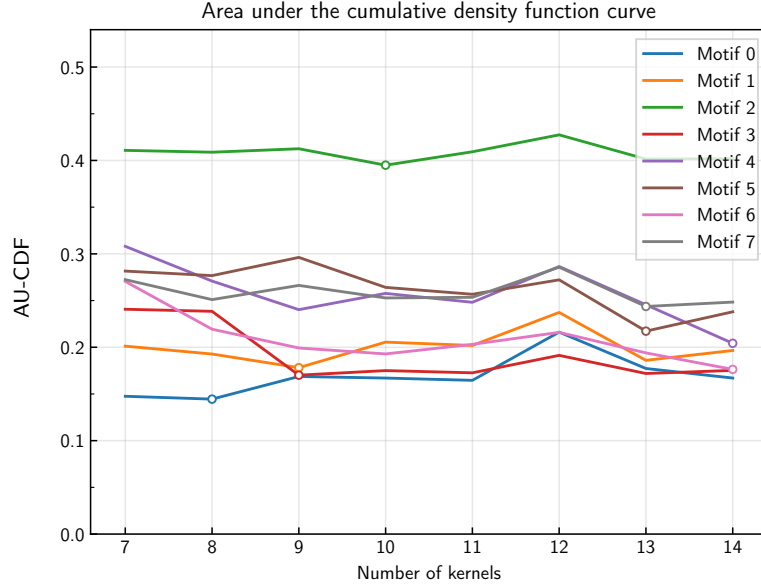


Figure 4.11: The AU-CDF per number of kernels (synthetic dataset 3). Less is better. The empty circle indicates the minimal value in the line.

synthetic datasets with known “ground truth” motifs (and also their classification rules) are constructed for study. Experiments show that accurate models tend to have higher motif identifiability, which is in support of the common practice (training a well-performing neural network, and then extracting motifs from the first-layer convolutional kernels). However, the

variability of motif identifiability (under the same model performance) cannot be ignored. Also, once reaching a certain model complexity, the motif identifiability will even decrease if the model complexity continues to grow, without any substantial indication in terms of out-of-sample model performance of a potential problem. This raises a concern that we may inadvertently harm the motif identifiability when pursuing the best-performing neural network in practice. Interestingly enough, usually there is a “sweet spot” of model complexity where the network is flexible enough to achieve high performance, yet constrained enough so that motifs can be well recovered by the first-layer kernels.

CHAPTER 5

A Robust Neural Network Based Motif Discovery Workflow

In Chapter 4 we have seen that although better model performance generally means higher motif identifiability, there are two potential issues. First, over-complex model may decrease the identifiability (without obvious indication in terms of the out-of-sample model performance). Note that this is different from overfitting, where an over-complex model has learnt particular properties or noises of the training data, and thus will perform poorly on unseen data. Another issue is the motif identifiability still varies a lot for those networks which already have very good performance. This is not negligible as common neural network based motif discovery methods only focus on a single best-performing neural network, and comparing its learnt kernels with motifs in the known databases.

In this chapter, we present a robust motif discovery workflow, which can find promising consistently present motifs semi-automatically. The complete workflow is introduced in Section 5.1, consisting of 3 steps. Then, Section 5.2 shows the experiments how and how well we can identify the underlying motifs on both synthetic and real datasets. Finally, we conclude this chapter in Section 5.3.

This chapter is part of the Work 2 (submitted to PLOS Computational Biology).

5.1 The Proposed Robust Motif Discovery Workflow

In order to tackle the concerns expressed above, we propose a complete motif discovery workflow, which consists of 3 steps: (1) determining an appropriate number of convolutional kernels, (2) training multiple of models on re-sampled training data and detecting common trends in the learnt motifs through hierarchical clustering in the probabilistic motif space equipped with appropriate “motif similarity” score, and (3) constructing a representative motif for each motif group.

5.1.1 Step 1: Finding an Appropriate Number of Kernels

For the first step, we aim to find a network that provides as compact representation of motifs as possible without sacrificing out-of-sample performance. This is done by finding the “elbow”¹ of the performance curve plotting the cross-validated model performance against the number of convolutional kernels employed by the model. For a given number of kernels, we explore various combinations of other hyperparameters (e.g., the number of hidden neurons, learning rate, weight decay etc.) and record the best performance (on validation set). If adding more kernels no longer provides a significant performance improvement (elbow of the performance curve), we take it as an indication of the number of kernels relevant for the given task. Our experiments on synthetic data show that this step is important for enhancing motif identifiability.

5.1.2 Step 2: Training Multiple Models with Re-sampled Data

Having determined the number of kernels n (and its corresponding hyperparameter set), we proceed to train the models on multiple re-sampled versions of the training set. It is a common practice in neural network based motif discovery to concentrate on a single

¹Elbow method is a common heuristic in mathematical optimisation to choose a point where the marginal reward is no longer worth the additional cost. For example, it is often used to determine the number of clusters k in k -means clustering, and also the number of principal components in principal component analysis (PCA).

(best-performing) model and investigate whether there is a match between some of the learned kernels and known motifs from existing databases. We acknowledge that there may be variability in learnt kernels due to the particular training sample used and random initialisation of the model, which affects the motif identifiability. We would like to “robustify” our motif discovery by allowing multitude of trained models obtained on resampled data with random initialisations and searching for common emerging motifs among the multiple suggestions from the trained models. In particular, we perform 50 stratified re-samplings of 80% of the training data, obtaining $50 \times n$ learned kernels from trained models.

5.1.3 Step 3: Clustering and Representative Motifs

Finally, hierarchical clustering is applied to these $50 \times n$ candidate motifs to determine motif groups, using the symmetric motif similarity measure S_{sym} defined in Equation (4.9). Motifs within each group are deemed sufficiently similar to be considered variations of a single common representative motif. The collection of representative motifs of each group will constitute the overall output of our motif discovery workflow. We suggest treating the grouping operation as a semi-automated motif mining operation (as opposed to a fully automated treatment), allowing interactive involvement of domain experts.

In the beginning, every motif is a cluster by its own. The agglomerative clustering process involves a number of iterations, while in each iteration it compares the distance of all pairs of the remaining clusters and combines the two most similar ones. It stops until all the motifs are grouped into one cluster and produces a dendrogram showing a combination history. The most important thing is how to calculate the distance between two clusters of probabilistic motifs. Assume there are two groups of motifs, $\mathcal{U} = \{\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \dots\}$ and $\mathcal{V} = \{\mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \dots\}$, we calculate the distance between \mathcal{U} and \mathcal{V} using

$$d(\mathcal{U}, \mathcal{V}) = \sum_{i,j} \frac{1 - S_{\text{sym}}(\mathbf{U}^{(i)}, \mathbf{V}^{(j)})}{|\mathcal{U}| \cdot |\mathcal{V}|}. \quad (5.1)$$

This is the average distance between all pairs of \mathbf{U} and \mathbf{V} . Other kinds of distance such

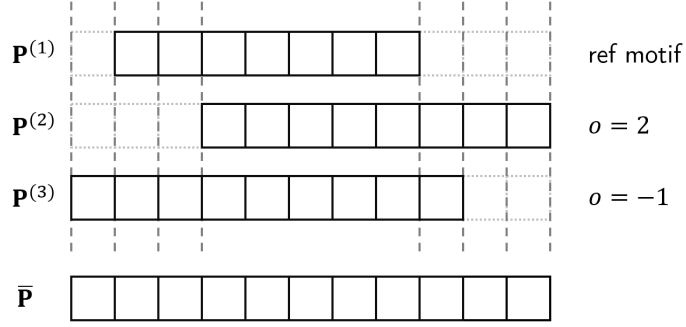


Figure 5.1: The alignment of candidate motifs within a cluster. The box with solid border means a probability distribution P over the nucleotide index set \mathcal{N} and the box with dotted line border means a wildcard nucleotide distribution for padding.

as minimum or maximum distance can also be used. More information can be found in the Python `scipy` module (Virtanen et al., 2020). Depending on the chosen similarity measure s on \mathcal{S} introduced in Section 4.2.1, the inter-cluster distance can be JS divergence based or geodesic distance based.

In order to find the representatives of the motif groups, the relative offsets among motifs are needed. We record the optimal offset between every pair of motifs during above calculation of their similarity. For a group of motifs, if they do share a coherent pattern, there should be only one alignment of them. Figure 5.1 shows an alignment of three motifs. Each motif is padded with wildcards so that all the motifs have the same length.

For a group $\mathcal{G} = \{P^{(1)}, P^{(2)}, \dots, P^{(m)}\}$ of m appropriately aligned and sufficiently similar probabilistic motifs, we would like to calculate a single “mean” motif $\bar{P}(\mathcal{G})$ that represents the whole motif cluster. This will be achieved by establishing the representative distribution $\bar{P}_j(\mathcal{G})$ over \mathcal{S} for each position $j = 1, 2, \dots, \ell$.

For a position j , given the motif cluster \mathcal{G} , we have a set of (nucleotide) probability distributions over \mathcal{N} , $\mathcal{G}_j = \{P_j^{(1)}, P_j^{(2)}, \dots, P_j^{(m)}\}$, $P_j^{(k)} \in \mathcal{S}$, $k = 1, 2, \dots, m$. Given a dissimilarity measure $d: \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ on \mathcal{S} , the mean representative of \mathcal{G} at position j can be found through the generalised notion of the mean,

$$\bar{P}_j(\mathcal{G}) = \arg \min_{Q \in \mathcal{S}} \sum_{Q' \in \mathcal{G}_j} d(Q, Q') = \arg \min_{Q \in \mathcal{S}} \sum_{k=1}^m d(Q, P_j^{(k)}). \quad (5.2)$$

Hence the representative motif for \mathcal{G} will be constructed as

$$\bar{\mathbf{P}}(\mathcal{G}) = (\bar{P}_1(\mathcal{G}), \bar{P}_2(\mathcal{G}), \dots, \bar{P}_\ell(\mathcal{G})). \quad (5.3)$$

5.2 Experimental Studies

5.2.1 Experiment Setup

Most experiment settings are the same as in Section 4.3.1. The differences are described as follows.

Datasets. In addition to the three synthetic datasets, we also demonstrate the motif discovery workflow on two real datasets, which are sequences contain the binding sites for the transcription factors FOXA2 and PDX1 respectively. The sequences from real datasets are of variable length and most of them are from 500 to 1200. As a pre-processing step, sequences longer than 2000 are cut off every 2000 nt (but no shorter than 300) as agreed by the biologist since the position information is not meaningful. The two datasets contain 19324 and 9138 sequences respectively, each with only about 40 sequences chopped in above step. The training/validation/test set split is still 60%, 20%, and 20%.

We adopt the common practice in literature (Zeng et al., 2016; Alipanahi et al., 2015) and perform a binary classification on the real datasets. The positive class consists of the original sequences, while the negative class is generated by shuffling positive sequences with matching dinucleotide composition (i.e., preserving the counts of all 16 dinucleotides AA, AC, ..., GT, TT). This shuffling is done with the `fasta-dinucleotide-shuffle` function in the MEME suite (Bailey, Boden, et al., 2009).

Network architectures. We still use the same neural networks as in Section 4.3.1. It is worth noting that for experiments on real datasets, we use a global (maximum and average) pooling layer after the convolutional layer, just as in previous work (Alipanahi et al., 2015;

Zeng et al., 2016). It is not used for the synthetic datasets.

Hyperparameters. During the first step of determining a proper number of convolutional kernels, we need to explore various hyperparameter combinations and record the best performance for every choice of numbers of kernels. With a few preliminary experiments, we can narrow down the search space to a relatively small range, on which the hyperparameters are fine-tuned. The possible values of all the hyperparameters can be found in Table 4.2. All neural networks are optimised with stochastic gradient descent (with weight decay).

Clustering. We use hierarchical agglomerative (bottom-up) clustering to group similar kernels as described in Section 5.1.3. Here we use the average distance (see Equation 5.1) of all pairs of kernels across the two clusters. This is done with the `cluster.hierarchy.linkage` function from the Python `scipy` module (Virtanen et al., 2020). As we are aware that this real dataset contains DNA sequences in both strands (directions), we need to consider two possible cases when calculating similarity between two learnt kernels \mathbf{P} and \mathbf{P}' :

$$S_{\text{bidirection}} = \max \left[S_{\text{sym}}(\mathbf{P}, \mathbf{P}'), S_{\text{sym}}(\mathbf{P}, \text{compl}(\mathbf{P}')) \right], \quad (5.4)$$

where `compl(\cdot)` returns the complementary strand of the kernel (reverse direction and complementary nucleotide), S_{sym} is the symmetric inter-motif similarity defined in previous section. The higher similarity in two cases is used as the bidirectional similarity, and the direction is recorded for representative motif generation.

5.2.2 Motif Recovery on Synthetic Datasets

Take synthetic dataset 1 as an example, we demonstrate the complete motif discovery workflow. For the first step, we try different numbers of convolutional kernels (from 1 to 9) and different hyperparameters from Table 4.2, each time record the highest (validation set) accuracy the network can achieve (see Figure 5.2, blue line). It can be seen from the curve that 5 kernels are enough for a network to get the best performance, which is also the expected

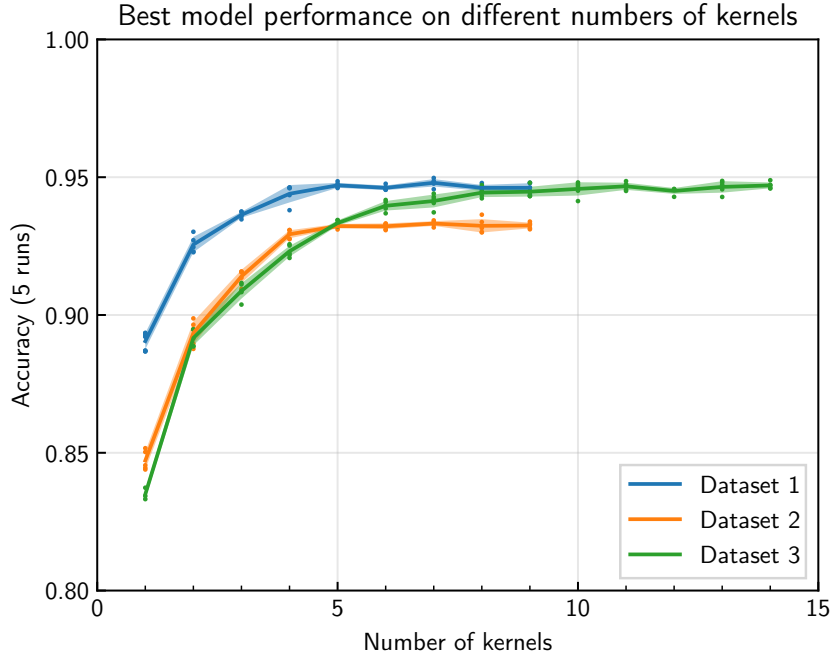


Figure 5.2: The (validation set) model performance vs. number of kernels curve for all three synthetic datasets. The y axis is the average accuracy over all classes. Each model is trained 5 times (independently) and every dot corresponds to a single run. The shadow means one standard derivation (plus or minus).

upper bound for this dataset (it can be calculated as we have full knowledge of the data generation process).

We then trained 50 independent 5-kernel neural networks, each time with random 80% resampling of the training set. The hierarchical clustering result among these 250 learnt kernels is shown in Figure 5.3, including its dendrogram in the top. Similar kernels appear as dense squares in the plot, each of which contains kernels that are very likely to be variations of a single common representative motif. Promising motif groups can be identified by setting a similarity threshold. Different strategies can be adopted to set such a threshold. Consensus clustering (also called cluster ensembles, [Strehl and Ghosh, 2002](#); [Monti et al., 2003](#)) might be helpful for determining a good clustering result (i.e., motif groups) by finding some consensus across multiple clustering results (coming from different clustering algorithms or multiple runs of a single one with random restart). In our case, the hierarchical agglomerative

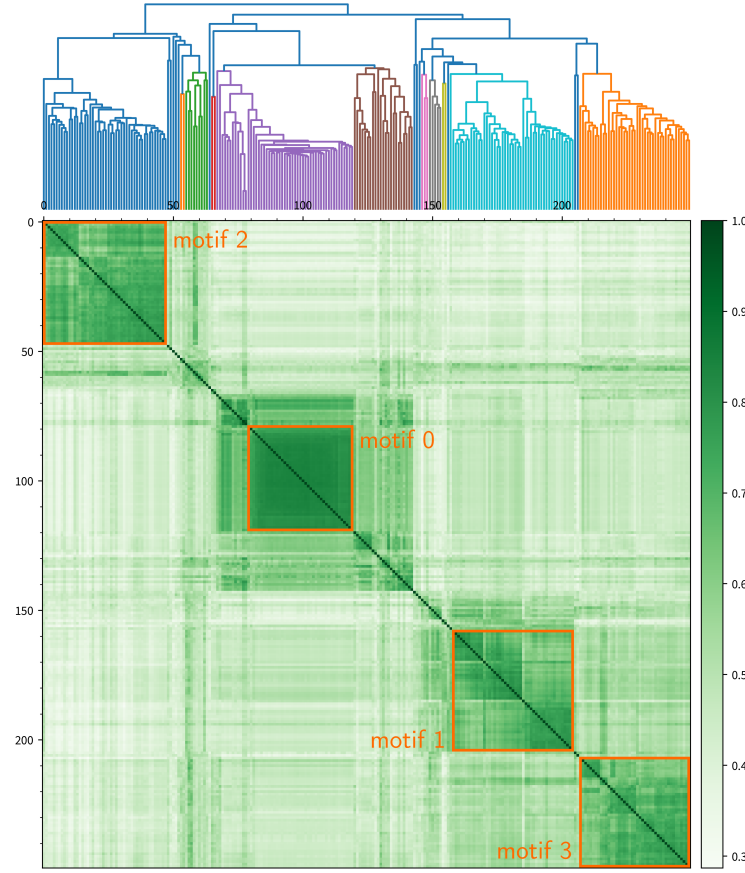


Figure 5.3: The hierarchical clustering result of 50×50 learnt kernels (synthetic dataset 1). They are kernels from 50 5-kernel neural networks, with the best hyperparameters on the validation set. Please see Figure 5.4 for their comparison to “ground truth” motifs.

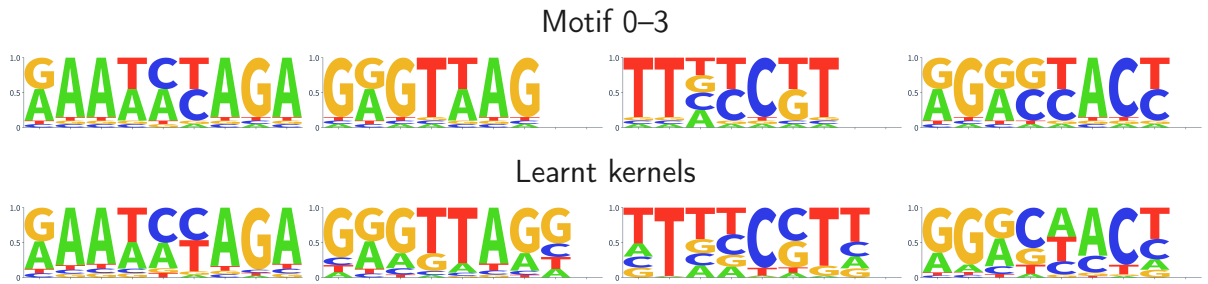


Figure 5.4: Ground-truth motifs (motif 0 to 3) and learnt representative kernels. The first row shows the four motifs in synthetic dataset 1. The second row shows the found motifs by our method. They are annotated in the clustering plot (Figure 5.3). The y -axis is the accumulated probability.

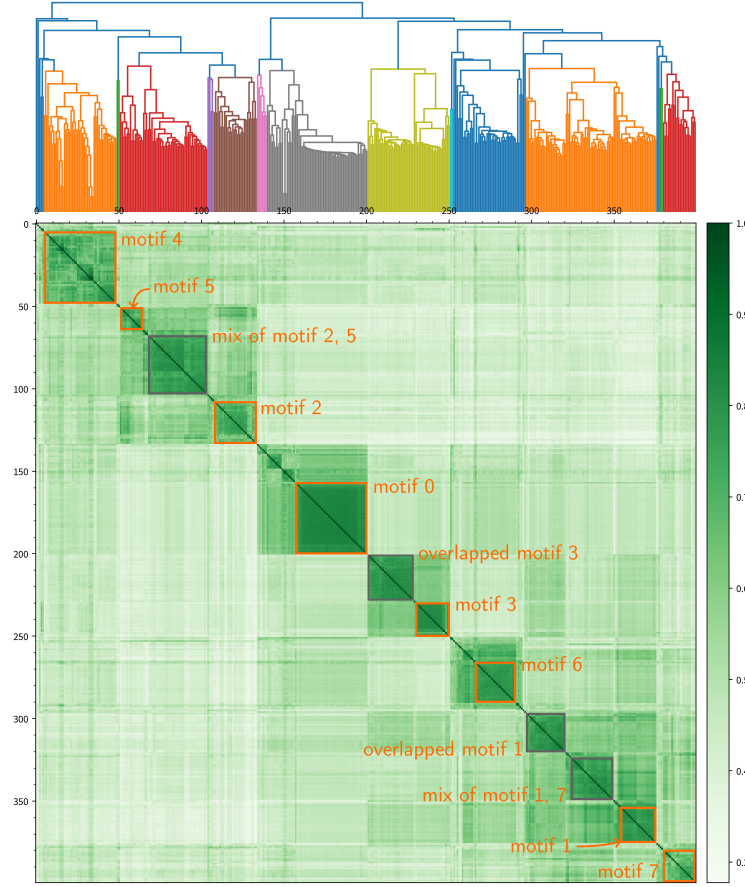


Figure 5.5: The hierarchical clustering result of learned kernels (synthetic dataset 3).

clustering result is deterministic, and we find it much more appropriate to adopt a semi-automatic approach allowing users to investigate any square (kernel group defining a common representative motif) of interest. The representative motif for each square will then be calculated automatically. For demonstration, we have identified 4 densest squares (annotated with orange borders), and the sequence logos of their representative motifs are shown in the bottom row of Figure 5.4, which are very close to the ground truth. If we look carefully at the clustering plot (Figure 5.3), we can see several smaller dense squares inside or right next to those picked squares. They are usually the same motif but with shifted positions. This shows the effect and also the importance of considering possible offset in the motif similarity measure.

The results for synthetic dataset 2 is very similar to dataset 1. Here we mainly present

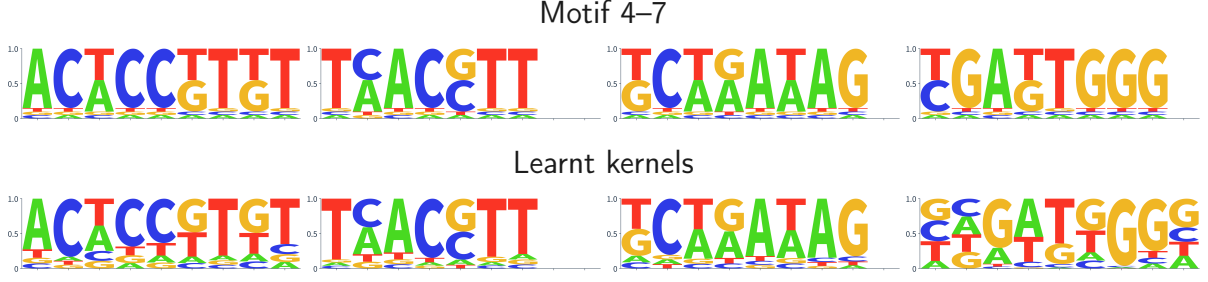


Figure 5.6: Ground-truth motifs (motif 4 to 7) and learnt representative kernels. The first row shows the other four motifs in synthetic dataset 3 in addition to dataset 1. The second row shows the found motifs by our method. They are annotated in the clustering plot (Figure 5.5). The y -axis is the accumulated probability.

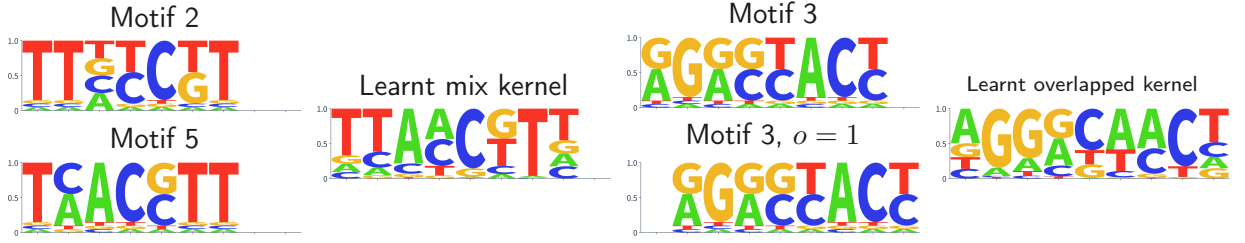


Figure 5.7: Visualisations of the unmatched kernels in Figure 5.6. Note the empty position can be viewed as uninformative wildcard distribution.

the results for dataset 3. According to the performance curve (Figure 5.2, green line), the network’s performance starts to saturate with about 8 kernels. Again, the 8-kernel neural network is trained 50 times independently (with random 80% resampling of the training data). The hierarchical clustering result among all the learned kernels is shown in Figure 5.5, suggesting several consistently present patterns (annotated with orange squares). In addition to the motifs 0 to 3, the new motifs 4 to 7 can also be recovered (see Figure 5.6 for their comparison to ground truth). Note that motifs 2 and 5 are kind of similar, both having high-probability T in the beginning and ending of the motifs. In the clustering plot (Figure 5.5), they are respectively the third and second orange squares, having a grey square in between and together forming a bigger dim square. The representative motif of the grey square is shown in Figure 5.7 (left). This is different from the situation that a bigger square contains two smaller squares, which are actually the same motif with position shift. Another

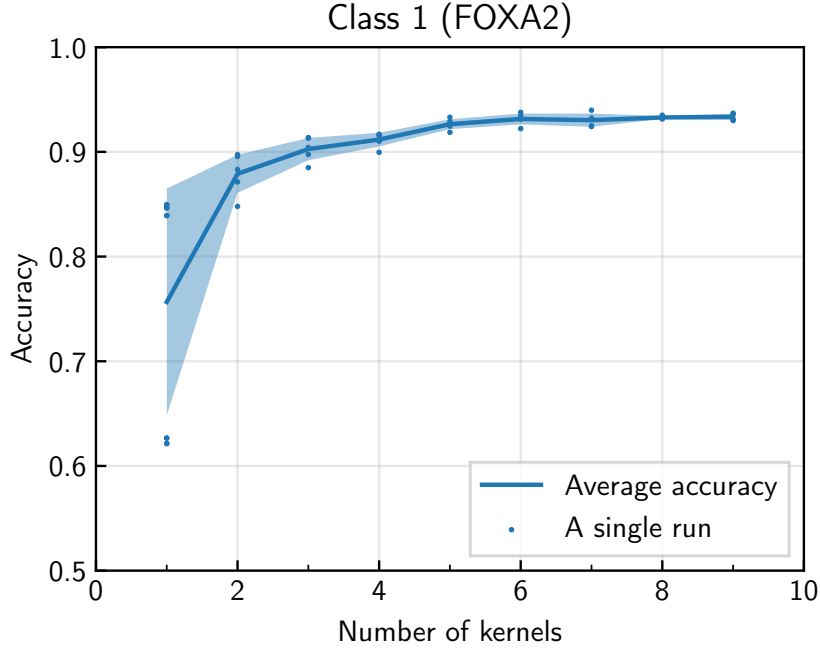


Figure 5.8: The (validation set) model performance vs. number of kernels curve (real dataset 1, FOXA2). The shadow means one standard derivation (plus or minus).

example of a suboptimal kernel is shown in Figure 5.7 (right). These examples show the necessity of the semi-automatic inspecting of the potential motifs. Overall, our proposed motif discovery workflow works reasonably good.

5.2.3 Recovered Transcription Factors

Using the method demonstrated above, we show that motifs in real dataset can also be identified. Figure 5.8 shows how the model performance (validation set) changes as we increase the number of convolutional kernels of a network on the real dataset 1. It is easy to note that the performance saturates when the network has 5 or 6 kernels. As demonstrated with the synthetic dataset, we trained 50 independent 6-kernel neural networks, each time with a random 80% resampling of the training data. The hierarchical clustering result among these learnt kernels is shown in Figure 5.9. (Due to the randomness of neural network training, there are few chances that a model fail to converge or perform obviously not reasonable.

There are finally 294 kernels in the figure, with one run failed.) Five dense squares are annotated with orange borders in the above figure, corresponding to the 5 motifs shown in Figure 5.10 (left) in the same order. Note that the 4th and 5th motif are almost the same but with one position shift. This has already been suggested in the similarity clustering plot (Figure 5.9) as they are two dense squares inside a bigger one. On the right of Figure 5.10 we also show the motifs found by the commonly used motif-discovery software Homer (Heinz et al., 2010), which takes in two sets of DNA sequences and tries to identify motifs that are enriched in one set comparing to the other set (backgrounds). Homer depends on the statistic enrichment difference between the two sets of sequences, while neural networks try to adapt the inner weights (including the convolutional kernels) to achieve as high discriminative power as possible. Comparing the results of the neural network and Homer, there are two motifs in common (left 1st vs. right 1st, left 4th vs. right 3rd), which match well with known motifs FOXA1 and ZNF460 respectively.

The results for real dataset 2 can be found in Supplementary Figure B.1, B.2, and B.3. Note that for dataset 2 there is no motif standing out. This can be seen from the confidence level of Homer, which is much lower than that of real dataset 1. It comes as no surprise that the neural network based method and Homer have less agreement here.

5.2.4 Shuffled Background Sequences and “Negative” Phantom Motifs

In our proposed motif discovery workflow, the number of convolutional kernels in a neural network has been greatly reduced. This in the one hand forces the neural network to learn as compact representations as possible (rather than distributed representations of motifs), on the other hand avoids having too many redundant or uninformative kernels. In this way, a well-trained kernel will be discriminative. However, being discriminative does not mean a kernel is biologically sound. Many biological sequence prediction tasks (e.g., protein binding specificity prediction) are formulated as differentiating a set of original sequences and their

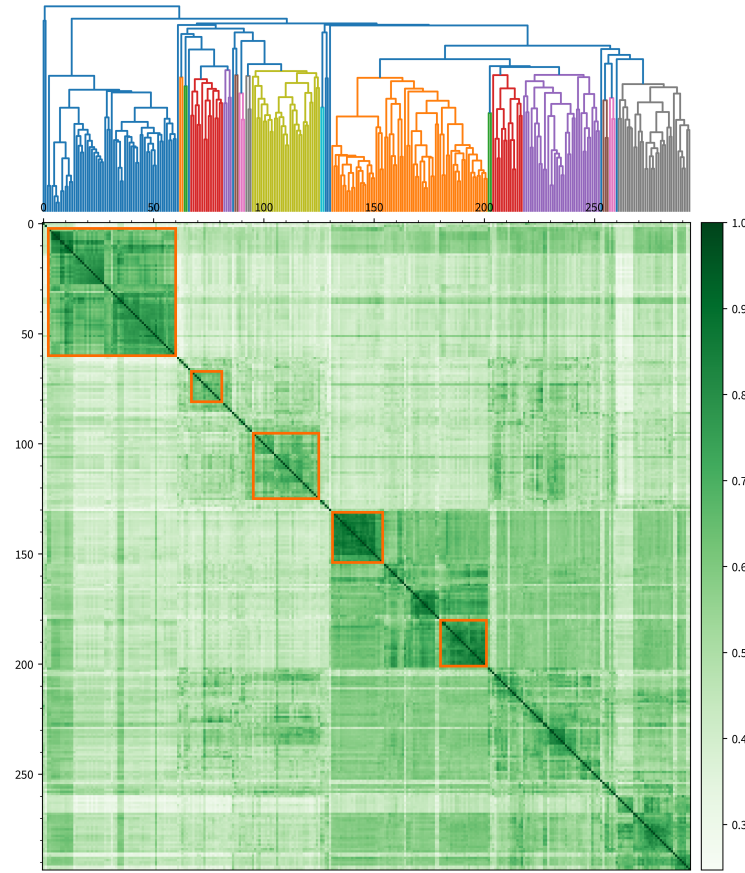


Figure 5.9: The hierarchical clustering result of learned kernels (real dataset 1, FOXA2).

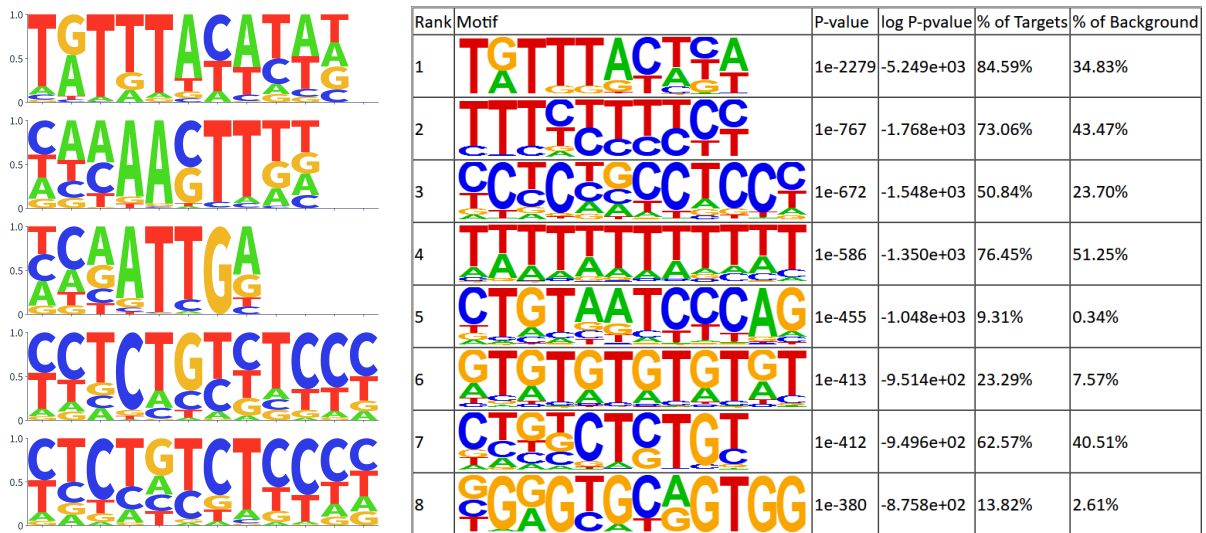


Figure 5.10: Motifs found by the neural network (left) and Homer (right)

shuffled sequences (backgrounds), usually with the frequencies of k -mers reserved (Alipanahi et al., 2015; Zeng et al., 2016). In other words, the background sequences are synthetic and do not have any biological meaning. When convolutional kernels are expected to learn (enriched) motifs in the original sequences, they can also find “motifs” in the shuffled sequences, which are patterns that appear more frequent in the shuffled sequence than in the original sequences. Unlike enriched motifs, these “negative” phantom motifs are patterns that should be absent from the (positive) sequences.

In order to identify the negative phantom motifs, we need to know how these motifs affect the network’s prediction. We initialise a neural network (kernels) with the 4 unique motifs shown in Figure 5.10 (left). Note that the motifs are representatives of the learned kernels and are in probabilistic representations (or PPMs), thus can not be directly used in neural networks. For a nucleotide probability distribution $P = (p_1, p_2, p_3, p_4)$ on a certain position on a motif, the corresponding weights for convolutional kernels are calculated by

$$w_i = \ln \frac{p_i}{\min_j p_j}, \quad (5.5)$$

which is a reverse softmax operation. Additionally, w_i are normalised by subtracting a constant in order to keep zero-mean per position. Their complementary motifs are also added to the network (so 8 kernels in total) as the real data has DNA sequences in both directions. The other parts (e.g., the global maximum/average pooling, the number of hidden neurons) of the network remained the same as before. During the network training process, the weights of convolutional kernels were fixed and only the weights of fully-connected layers were optimised. The network achieved around 90% accuracy, which is only slightly worse than the 91% accuracy of a free-to-optimize 4-kernel neural network (see Figure 5.8).

We use sensitivity analysis to study the relationship between the network prediction and the kernel output. Specifically, the derivative of the predicted positive-class probability with respect to each kernel’s output (after pooling layer) has been calculated (Figure 5.11). Each point is the result of an instance in the test set. If the derivative is positive, the larger the output of the corresponding kernel, the higher the probability of positive class.

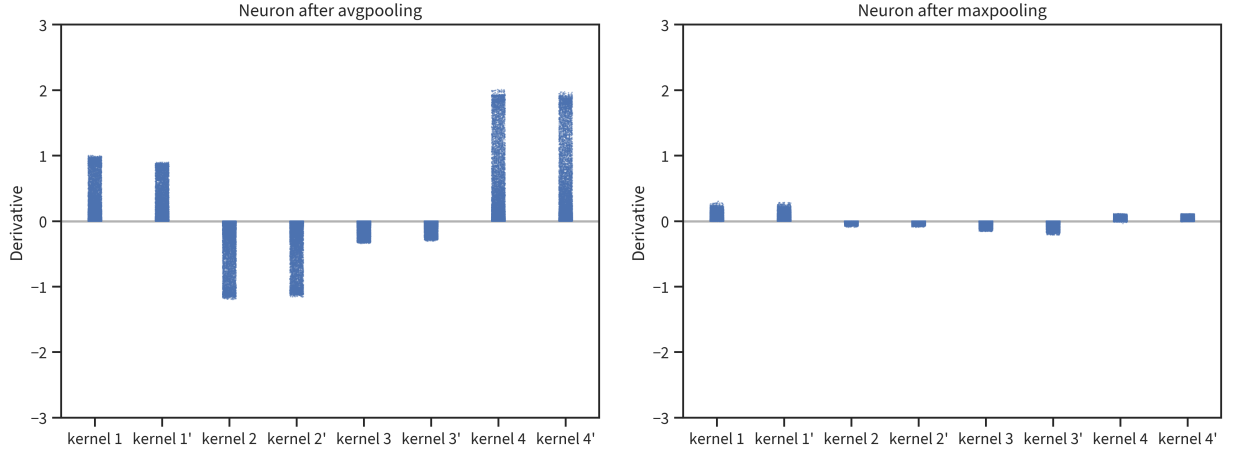


Figure 5.11: The sensitivity of the predicted probability of positive class to each kernel (real dataset 1). It is the derivative of the output positive class neuron (in the last layer) with respect to the output of the first-layer kernel convolution (after global average/maximum pooling).

Otherwise, the increase of the kernel output will reduce the predicted probability of positive class (i.e., leaning to the negative class). According to Figure 5.11, we can see that the 1st and 4th kernels represent positive motifs, where the neural network and Homer agree with each other. The 2nd and 3rd kernels are apparently “negative” phantom motifs, which cannot be interpreted as conventional enriched motifs. The results of average pooling layer and maximum pooling layer are similar, only differing on the magnitude.

5.3 Chapter Summary

In this chapter, we review the two potential issues in the common neural network based motif discovery approaches: (1) Most neural network based methods focus mainly on the model performance, which often leads to big and complex networks. However, as we conclude in Section 4.4, this may harm the motif identifiability and make the kernels less informative. (2) There is unignorable variability on the learnt kernels (and thus the motif identifiability) due to the particular training sample used and random initialisation of the model. These two issues show that the motif identifiability should also be taken into account when selecting

the model architecture, rather than only based on the predictive performance.

We thus suggest a robust motif discovery workflow which includes the determination of number of convolutional kernels, training multiple independent neural networks on re-sampled data, hierarchical clustering of the learnt kernels, and generation of representative motifs. The number of convolutional kernels is expected to be as small as possible when the performance is good enough, which prevent the model to learn distributed motif representations. And the hierarchical clustering enables us to find motifs that are presented consistently rather than by chance. Instead of a fully automated manner, we suggest treating the grouping operation as a semi-automated motif mining operation as there is unlikely a perfect threshold for the motif grouping, On the other hand, it also allows interactive involvement of domain experts.

There is another caveat to using neural network for motif discovery. As neural networks are performing classification by discriminating sequences that come from different classes, if the dataset uses shuffled positive sequences as the classification background for motif discovery, the learnt kernels may contain “negative” phantom motifs which means not “enriched” but “absent”. This might also be a reason why many learnt kernels neither can be found in the known databases nor seem to be reasonable novel motifs. Our kernel sensitivity analysis can help us easily filter out the “negative” phantom motifs and suggest which motifs are promising. This is a big difference from classic motif discovery tools (such as Homer) which are based on statistical frequency significance between foreground and background sequences, note that they also have their own “nonsense” motifs (e.g., the fourth Homer motif, the all T pattern, in Fig 5.10).

CHAPTER 6

Direct Learning of Probabilistic Motifs in Neural Networks

In previous chapters we have seen the common practice of neural network based motif discovery. The probabilistic motifs are generated with a two-step approach: (1) train a well-performing convolutional neural network, (2) extract the learnt first-layer kernels, use them to collect a set of subsequences, count the frequencies and transform into PPMs (Figure 4.1). Then a natural question is: can we get rid of the post hoc interpretation and directly learn the probabilistic motifs in the neural networks?

The rest of this chapter is organised as follows. We first introduce the motivation in Section 6.1 and the mathematical details of the proposed methods (likelihood and log-likelihood kernels) in Section 6.2. Then, the experiments results on both synthetic and real datasets are presented in Section 6.3. Finally, Section 6.4 summarises this chapter.

This chapter includes the main content of Work 3.

6.1 Motivation

As mentioned in Section 2.1, convolution calculates the sum of element-wise product of the kernel and sequence (segments at each position and channel):

$$(S * K)(i) = \sum_{j=1}^{L_K} \sum_{c=1}^C S(i+j, c) \cdot K(j, c), \quad i \in \{1, 2, \dots, L_S - L_K + 1\}. \quad (2.16 \text{ revisited})$$

Both S and K are two-dimensional matrices, for example, (assuming each column corresponds to a nucleotide and the row index corresponds to the position)

$$S = \text{'CATCTAG'} = \begin{matrix} & \begin{matrix} A & C & G & T \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{matrix}, \quad K = \begin{matrix} & \begin{matrix} A & C & G & T \end{matrix} \\ \begin{bmatrix} -0.3 & -0.1 & -0.2 & -0.4 \\ 0.4 & -0.1 & -1.5 & 0.3 \\ -0.9 & -0.4 & -0.9 & 1.3 \\ 0.5 & 2.3 & -2.0 & -1.7 \\ -0.2 & -1.2 & -2.2 & 2.7 \\ 1.6 & -1.5 & 1.3 & -2.3 \\ -0.2 & -1.8 & 3.1 & -2.0 \end{bmatrix} \end{matrix}. \quad (6.1)$$

In convolutional neural networks, kernel K is randomly initialised and then trained by back-propagating the classification error signal through the network. The learnt kernels are viewed as scoring matrices (i.e., PWMs) in bioinformatics. However, the kernel weights in convolu-

tional neural networks are not unique, for example,

$$K = \begin{matrix} & \begin{matrix} \text{A} & \text{C} & \text{G} & \text{T} \end{matrix} \\ \begin{bmatrix} -0.3 & -0.1 & -0.2 & -0.4 \\ 0.4 & -0.1 & -1.5 & 0.3 \\ -0.9 & -0.4 & -0.9 & 1.3 \\ 0.5 & 2.3 & -2.0 & -1.7 \\ -0.2 & -1.2 & -2.2 & 2.7 \\ 1.6 & -1.5 & 1.3 & -2.3 \\ -0.2 & -1.8 & 3.1 & -2.0 \end{bmatrix} & \longleftrightarrow & K' = \begin{matrix} & \begin{matrix} \text{A} & \text{C} & \text{G} & \text{T} \end{matrix} \\ \begin{bmatrix} 4.7 & 4.9 & 4.8 & 4.6 \\ -2.6 & -3.1 & -4.5 & -2.7 \\ -2.9 & -2.4 & -2.9 & -0.7 \\ 0.5 & 2.3 & -2.0 & -1.7 \\ -0.2 & -1.2 & -2.2 & 2.7 \\ 1.6 & -1.5 & 1.3 & -2.3 \\ -0.2 & -1.8 & 3.1 & -2.0 \end{bmatrix} & \begin{matrix} (+5) \\ (-3) \\ (-2) \end{matrix} \end{matrix} . \quad (6.2)$$

These two convolutional kernels are functionally equivalent (there might be an illusion that the first row in K' is more important). This tells us the absolute magnitudes of the PWMs are not meaningful across different positions (rows), which will also cause some trouble calculating the distance or similarity between two PWMs.

Previous works (e.g., DeepBind) transform the learned PWMs to PPMs by collecting the high-response sequence segments (from the test set), counting the frequency and then calculating the probabilities (Figure 4.1). However, this transformation is somehow heuristic and post hoc. In this chapter, we are interested in whether there is a more principled way of learning probabilistic motifs using neural networks which allows the motifs to emerge during the training process.

6.2 The Proposed Approach

Here we propose **probabilistic kernels** in place of above convolutional kernels,

$$(S * K)(i) = \prod_{j=1}^{L_K} \sum_{c=1}^C S(i+j, c) \cdot \tilde{P}(j, c), \quad i \in \{1, 2, \dots, L_S - L_K + 1\}. \quad (6.3)$$

while K is a normal kernel with real-valued (can be both positive and negative) parameters to train, \tilde{P} is the probabilistic kernel transformed from K . This transformation is done with

the softmax function on each position. Softmax (or softargmax) is a smooth approximation to the “arg max” function. It takes as input a vector \mathbf{z} (here it is a row in K) and normalises it into a “probability distribution”

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \quad (6.4)$$

where T is the temperature parameter. As temperature T increases, the output distribution will be “softer” (less crisp). By default, T is set to 1. Softmax function is also known as the Boltzmann distribution (or Gibbs distribution) in statistical mechanics, where i is the index of the microstate of the system, z_i is the energy of the i^{th} state, T is the thermodynamic temperature (and Boltzmann’s constant k). As an example, ($T = 1$)

$$K = \begin{matrix} & \begin{matrix} \text{A} & \text{C} & \text{G} & \text{T} \end{matrix} \\ \begin{bmatrix} -0.3 & -0.1 & -0.2 & -0.4 \\ 0.4 & -0.1 & -1.5 & 0.3 \\ -0.9 & -0.4 & -0.9 & 1.3 \\ 0.5 & 2.3 & -2.0 & -1.7 \\ \vdots & & & \end{bmatrix} & \xrightarrow[\text{(per row)}]{\text{softmax}} \tilde{P} = \begin{matrix} & \begin{matrix} \text{A} & \text{C} & \text{G} & \text{T} \end{matrix} \\ \begin{bmatrix} 0.26 & 0.23 & 0.28 & 0.23 \\ 0.37 & 0.23 & 0.06 & 0.34 \\ 0.08 & 0.13 & 0.08 & 0.71 \\ 0.14 & 0.84 & 0.01 & 0.01 \\ \vdots & & & \end{bmatrix} \end{matrix} \end{matrix} \quad (6.5)$$

This new “convolution” is no longer a general similarity score, instead it means the likelihood that a certain segment of input sequence is generated from the probabilistic distribution represented by the kernel. Besides above likelihood kernel, there is another option which is to calculate the log-likelihood:

$$(S * K)(i) = \sum_{j=1}^{L_K} \sum_{c=1}^C S(i+j, c) \cdot \log \tilde{P}(j, c), \quad i \in \{1, 2, \dots, L_S - L_K + 1\}. \quad (6.6)$$

Note that as $\tilde{P}(i, j) \in (0, 1)$, the log-likelihood is in $(-\infty, 0)$.

The use of probabilistic kernels has at least two potential advantages:

1. The motifs (PPMs) are already represented in the probabilistic kernels during the training, instead of post hoc reinterpretation.

2. Probabilistic kernels are more discriminative in terms of the convolution output. We will discuss it soon in Section 6.3.2.

However, it has an obvious disadvantage that the computational complexity is much higher (because of softmax).

6.3 Experimental Studies

6.3.1 Experiment Setup

Datasets. We use two synthetic datasets (1 and 3) and two real-world datasets (FOXA2 and PDX1) in the experiments. The synthetic dataset 2 is very similar to dataset 1 with only the motifs swapped (Table 4.1). Please see previous sections for descriptions of the datasets.

Compared models. Three networks has been used for the experiments. The conventional convolutional neural networks (shown as ConvNet in the results) are viewed as the baseline. Besides, there are LikelihoodNet and LogLikelihoodNet which make use of Equation (6.3) and Equation (6.6) in place of the convolutional kernel respectively. These two probabilistic kernels (likelihood and log-likelihood) have the same trainable parameters (i.e., K s) as in conventional convolutional kernels, but have two differences: (1) The probabilistic kernels do not have bias terms. (2) They have a special hyperparameter, temperature T , controlling the curvature of the softmax function. Unless stated otherwise, T is set to 1 by default in the following experiments. (We have experiments investigating its effect on the learned motifs in the later section.) Another thing to note is there is the output of the convolutional kernels is usually passed through an activation function, we do not do this for probabilistic kernels.

The rest of the network architecture is the same for all three kinds of neural networks: Each network has a convolutional/probabilistic kernel layer, followed by a global maximum and average pooling layer if it is for the real dataset. The output is then fed into a series of

Table 6.1: The hyperparameters of neural networks in the experiments.

Hyperparameter	Possible values
Network architecture	
Number of hidden neurons	16, 32, 64, 32+16 ^a
Optimisation	
Batch size	16, 32
Learning rate	0.01, 0.05, 0.1
Weight decay (L^2 norm)	0, 0.001, 0.005, 0.01

^a 2 hidden layers with 32 and 16 neurons each.

fully-connected layers to produce the final prediction. All the activation functions are ReLU and the last layers use softmax.

Hyperparameters. Neural networks with different numbers of kernels are evaluated respectively. According to the experiments in previous chapters, we choose 3–8 kernels for synthetic dataset 1 and both real datasets, and use 7–12 kernels for synthetic dataset 3 as it has 8 underlying motifs. This covers a wide range of neural networks from which can only underfit to those which are expressive enough. All the kernels have the same size (length) 12, larger enough for the synthetic motifs and possible motifs in the real datasets. Again with the help of preliminary experiments, the search space of other hyperparameters can be confined to a relatively small range, on which the hyperparameters are fine-tuned. Table 6.1 shows all the possible values of hyperparameters in the experiments.

Network initialisation. The convolutional kernels use He initialisation (He, Zhang, et al., 2015), which is a good choice for neural networks that use ReLU activation functions. It is essentially a uniform distribution over $[-a, a]$, while a is calculated based on the kernel dimensions. For probabilistic kernels, we simply use uniform distribution over $[-1, 1]$. (When the temperature is not 1, we scale it to $[-T, T]$ accordingly.)

The likelihood kernel (Equation 6.3) and log-likelihood kernel (Equation 6.6) still have their own problems. As the likelihood is a product of multiple probabilities, it easily becomes

really small, especially in the early stage of the training (e.g., the expected likelihood with random initialisation of our kernel is $0.25^{12} = 5.96 \times 10^{-8}$, where 12 is the kernel length). We thus multiply a constant to the likelihood output for computation efficiency. A heuristic choice is 3^{12} (for DNA and RNA sequence), which can be selected based on the expected likelihood of the kernel. As for the log-likelihood kernel, we add a constant $12 \times \log 4$.

Optimisation. Cross entropy is used as the loss function for the classification. The neural networks are optimised using stochastic gradient descent with weight decay (L^2 norm regularisation). The batch size is chosen from 16 and 32. Consider the sequences in real datasets are of variable length, we use gradient accumulation and update the model weights every 16 or 32 examples as described in Section 4.3.1.

The learning rate is reduced to its half once the performance on validation set has not seen any improvement for a certain number of epochs, which is controlled by a hyperparameter, the “patience” of the learning rate scheduler. Early stopping is used to stop the training process when the performance on validation set starts to degrade and there is also a hyperparameter for the patience. In our experiments, we use 6 and 15 epochs for the learning rate scheduler and early stopping respectively. Every time the learning rate is reduced, we add 3 more epochs to the current early stopping patience.

Evaluation. The performance of a neural network is evaluated by its average accuracy among all classes. Synthetic dataset 1 has 3 classes and synthetic dataset 3 has 5 classes, while both real datasets have 2 classes (positive vs. shuffled). For each neural network, we first fix the number of kernels and find the best hyperparameter combination (among 6.1) based on their performance on the validation set. Then each neural network is trained 5 times (independently) under above best hyperparameters and evaluated on the test set.

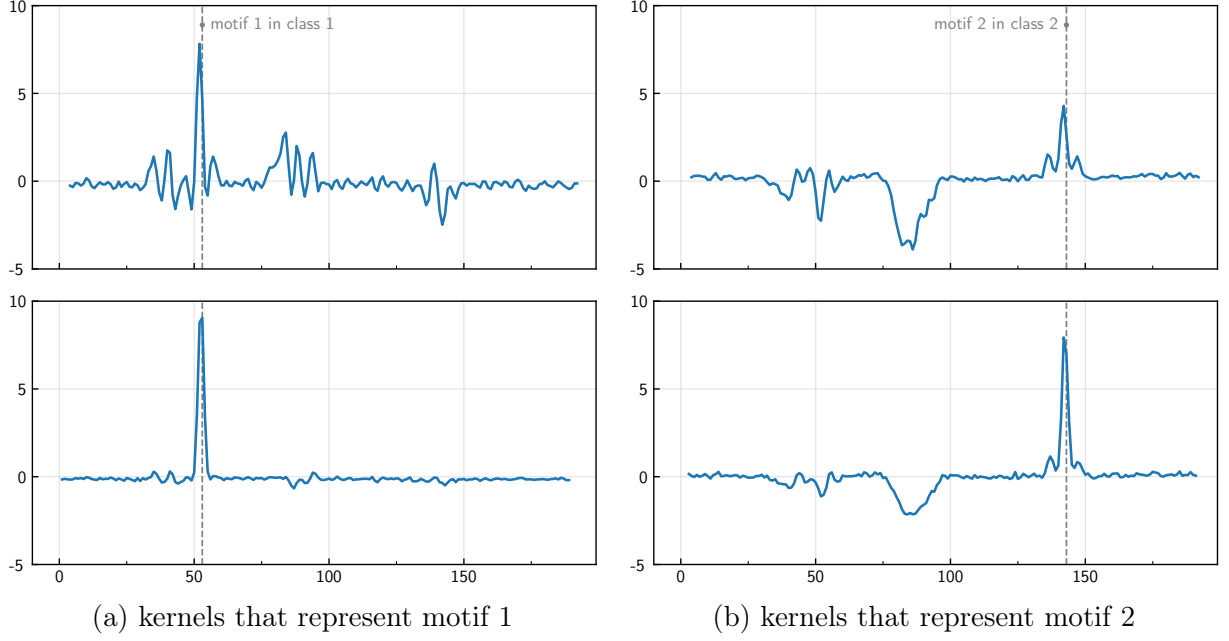


Figure 6.1: The average response of learnt kernels on the sequences in validation set (synthetic dataset 1). The first row shows the results of convolutional kernels and the second row is for the likelihood kernels. The response has been standardised (zero mean, unit standard deviation) respectively for each kernel. The vertical dashed lines are the related ground-truth motif locations. There might be a slight shift between the peaks of response and the motif locations as the kernel is usually longer than the motif, or the kernel may only contain a part of a motif. Please see Supplementary Figure B.4 for the results of other kernels.

6.3.2 Probabilistic Kernels Are More Stable

We first show that probabilistic kernels are more stable on detecting patterns than the convolutional kernels. Figure 6.1 presents the standardised average response (output) of the learned kernels on the sequences in the validation set. Each kernel (convolutional or probabilistic) is convolved with every input sequence and produces an output of length 189 (given the input sequence length 200, and the kernel length is 12). The output of each kernel is standardised so that it has a mean of 0 and a variance of 1. The kernels are from two best-performing 5-kernel neural networks on synthetic dataset 1, a convolutional network and a likelihood network each, having 94.47% and 94.15% out-of-sample classification accuracy respectively.

It can be seen from Figure 6.1 that probabilistic kernels (the second row) have much sharper response than the convolutional kernels when there is a match on patterns, for example, at position 143 in Figure 6.1(b). Also, the probabilistic kernels are less sensitive to irrelevant patterns and will keep inactive, which can be observed from the Figure 6.1(a), where convolutional kernels produce zigzag-like output. These findings make sense as the probabilistic kernels utilize the multiplication of all positions, while in convolutional kernels the response of each position is additive.

6.3.3 Accuracy

We compare the accuracy of above three neural networks (ConvNet, LikelihoodNet, and LogLikelihoodNet) under different numbers of kernels and on different datasets (2 synthetic datasets and 2 real datasets). The full results are represented in Table 6.2. In synthetic datasets 1 and 3, the LikelihoodNet is slightly better than ConvNet¹. With L^2 norm regularisation and early stopping, there is no much overfitting in both neural networks. However, because of the softmax operation within the kernels, LikelihoodNet is slightly more self-regularised than ConvNet (real-valued weights K are transformed into $(0, 1)$ “probabilities” \tilde{P}). This difference is obvious on the simpler synthetic dataset 1, but becomes smaller for the synthetic dataset 3 which is relatively more difficult to overfit under the same setting. Unlike LikelihoodNet, LogLikelihoodNet performs worse on both datasets and it is found to be prone to overfit. The reason may be that the log-likelihood kernel essentially calculates the summation of the transformed log-probabilities $\log \tilde{P}$, but the L^2 regularisation is applied to the original weights K . This is also reflected on their choices of the regularisation coefficient λ . ConvNet’s best L^2 coefficients are 0.005 for both synthetic datasets 1 and 3, while LikelihoodNet chooses 0.005 and 0.001. For LogLikelihoodNet, it achieves its best validation performance with 0.01 and 0.005 but still overfits the training set (the performance becomes worse if we further increase the regularisation coefficients to 0.02 and 0.01).

¹Please see Supplementary Table A.2 for the extended results of statistical tests

Table 6.2: The performance of neural networks with different kernels. The performance is the average accuracy (in percentage) of all classes. For each given number of kernels, we find the best hyperparameter combination using the validation set and report the 5-run average of the test-set accuracy, along with its standard derivation. The “diff” column is the relative difference comparing to that of convolutional neural networks. Also see Table 6.2 (continued) for results on real datasets.

Synthetic dataset 1					
Num of kernels	ConvNet	LikelihoodNet	Diff	LogLikelihoodNet	Diff
3	93.63 \pm 0.17	93.83 \pm 0.14	+0.20	92.25 \pm 0.07	−1.38
4	94.06 \pm 0.23	94.73 \pm 0.28	+0.67	92.21 \pm 0.14	−1.85
5	94.34 \pm 0.20	94.65 \pm 0.41	+0.31	92.22 \pm 0.08	−2.12
6	94.28 \pm 0.08	94.93 \pm 0.03	+0.65	92.19 \pm 0.20	−2.09
7	94.53 \pm 0.15	94.90 \pm 0.11	+0.37	92.28 \pm 0.18	−2.25
8	94.27 \pm 0.11	94.98 \pm 0.06	+0.71	92.26 \pm 0.13	−2.01
Synthetic dataset 3					
Num of kernels	ConvNet	LikelihoodNet	Diff	LogLikelihoodNet	Diff
7	93.97 \pm 0.35	93.93 \pm 0.16	−0.04	90.25 \pm 0.23	−3.72
8	94.28 \pm 0.08	94.46 \pm 0.20	+0.18	90.44 \pm 0.26	−3.84
9	94.29 \pm 0.12	94.25 \pm 0.06	−0.04	90.94 \pm 0.05	−3.35
10	94.28 \pm 0.25	94.43 \pm 0.16	+0.15	90.83 \pm 0.15	−3.45
11	94.48 \pm 0.10	94.49 \pm 0.13	+0.01	90.78 \pm 0.12	−3.70
12	94.27 \pm 0.12	94.49 \pm 0.09	+0.22	90.98 \pm 0.09	−3.29

Things are different for the real datasets and the results are shown in Table 6.2 (continued). We can see drastic performance degradation (about 10%) from both LikelihoodNet and LogLikelihoodNet to ConvNet. Also, the variance of the performance becomes significantly larger. We further investigate the reason in the next section.

6.3.4 LikelihoodNet Needs More Kernels

In order to find out why the neural networks with probabilistic kernels perform much worse on the real datasets, we initialise a LikelihoodNet with the 4 best learnt motifs (similar to that in Section 5.2.4), fix the kernel weights and only train the fully-connected layers.

Table 6.2 (continued): The performance of different neural networks on real datasets.

Real dataset 1 (FOXA2)					
Num of kernels	ConvNet	LikelihoodNet	Diff	LogLikelihoodNet	Diff
3	90.20 ± 0.91	79.79 ± 5.80	-10.41	76.23 ± 2.64	-13.97
4	90.78 ± 0.70	82.05 ± 5.56	-8.73	77.34 ± 1.20	-13.44
5	92.46 ± 0.41	83.22 ± 5.35	-9.24	79.62 ± 2.00	-12.84
6	92.80 ± 0.30	83.43 ± 1.79	-9.37	82.43 ± 1.97	-10.37
7	92.52 ± 0.47	83.79 ± 1.74	-8.73	83.05 ± 1.46	-9.47
8	93.11 ± 0.23	85.12 ± 1.13	-7.99	84.29 ± 1.08	-8.82
10	around 93	86.65 ± 0.60	-6.35		
15		88.82 ± 0.49	-4.18		
20		90.08 ± 0.37	-2.92		
25		91.19 ± 0.24	-1.81		
Real dataset 2 (PDX1)					
Num of kernels	ConvNet	LikelihoodNet	Diff	LogLikelihoodNet	Diff
3	87.79 ± 1.49	73.83 ± 6.82	-13.96	73.49 ± 1.05	-14.30
4	89.24 ± 0.80	77.18 ± 2.02	-12.06	75.56 ± 0.90	-13.68
5	90.28 ± 0.77	77.32 ± 2.15	-12.96	77.20 ± 0.08	-13.08
6	90.73 ± 0.65	78.87 ± 0.81	-11.86	77.76 ± 0.80	-12.97
7	90.54 ± 0.61	79.71 ± 0.76	-8.88	80.45 ± 0.84	-10.09
8	89.15 ± 1.72	80.89 ± 0.20	-8.50	81.02 ± 0.43	-8.13

Surprisingly, the LikelihoodNet with the best learnt motifs can only achieve 82.75% average accuracy, which is still much worse than the 90.78% of the 4-kernel ConvNet (on synthetic dataset FOXA2). We thus try LikelihoodNet with more kernels and append the result to the bottom of Table 6.2 (continued). It shows that the performance of LikelihoodNet is gradually improved as we increase the number of kernels, although it is worse than ConvNet even if using 25 kernels. (More results can be found in Supplementary Table A.1.)

So far we may realise the other side of the coin: As the probabilistic kernels are more stable (or discriminative) on detecting the sequence patterns, they are also less tolerant of the mismatch. In order to achieve similar performance on real datasets, they need more kernels

to represent the possible variants of the underlying patterns, which becomes a disadvantage of motif discovery.

6.3.5 Temperature Parameter and Learnt Motifs

As mentioned in Section 6.2, when temperature T becomes larger, the output of softmax will be “softer”. In an extreme case, if the T becomes infinity, then \mathbf{z}_i/T is close to 0 and the output \tilde{P} will be a uniform distribution, meaning a “wildcard” in the sequence motif probabilistic representation. Figure 6.2 shows the learned kernels when we choose different temperatures (on synthetic dataset 1). The results are from a 5-kernel neural network (with its best hyperparameters according to the experiments in previous sections). Other hyperparameters (including the random seed) remain the same except that the learning rate is reduced for smaller T . All networks have comparable, the best ever out-of-sample performance (near 95%). It is easy to note that as temperature T decreases, the learnt motifs become crisper. However, T cannot be arbitrarily small as the softmax will become too sharp. When $T = 1$, the learnt kernel weights K are distributed within $[-0.48, 0.88]$ (variance 0.09). And when $T = 0.1$, the weights are within $[-0.12, 0.30]$ (variance 0.004). The means in both cases are almost zero.

We also compare the model performance under different temperature T . The results on real dataset FOXA2 are shown in Table 6.3. Despite the instability when the number of kernels is small, the overall performance for different temperatures does not show obvious difference. This is as expected as T does not affect the neural network’s expressive power.

6.4 Chapter Summary

This chapter introduces alternatives to the conventional convolutional kernels, i.e., likelihood kernel and log-likelihood kernel, in the neural network based motif discovery. Probabilistic kernels have some advantages, such as being able to directly represent the probabilistic motifs in the neural networks rather than requiring post hoc interpretation. Also, there is a

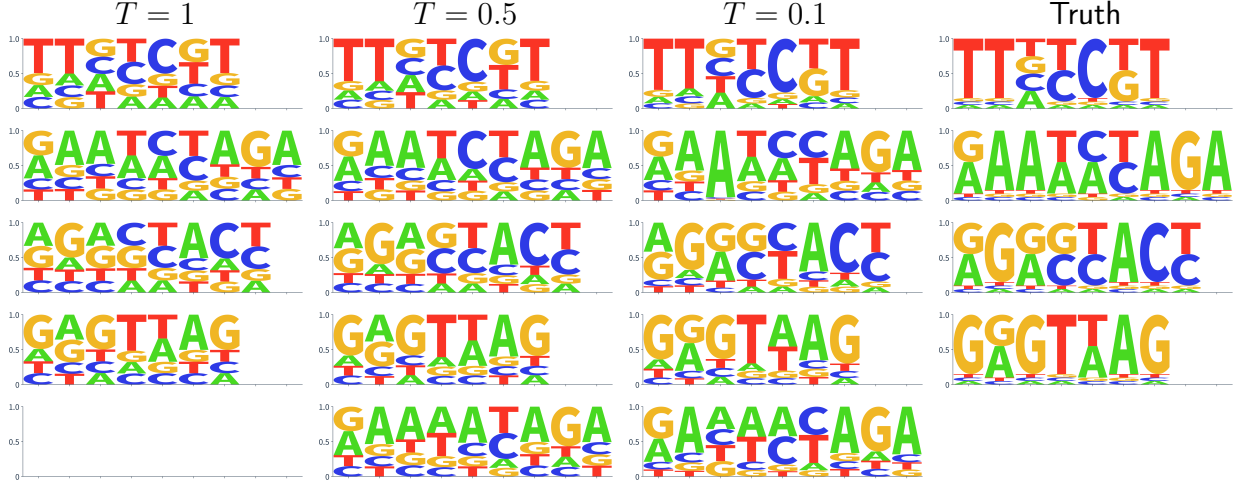


Figure 6.2: The effect of temperature parameter on learned motifs (synthetic dataset 1). The y -axis is the accumulated probability. The learnt kernels are trimmed in both ends if there are uninformative wildcard positions (entropy less than 1.95, see Equation 4.9). All above neural networks (under different T) have comparable performance.

Table 6.3: The performance of LikelihoodNet under different temperature T on real dataset 1 (FOXA2). The results for $T = 1$ (default setting) are copied from Table 6.2 (continued) for easy comparison.

Num of kernels	$T = 1$	$T = 0.5$	$T = 0.1$
3	79.79 ± 5.80	74.21 ± 3.47	75.76 ± 1.00
4	82.05 ± 5.56	77.68 ± 2.82	78.40 ± 4.40
5	83.22 ± 5.35	80.81 ± 1.73	82.19 ± 1.95
6	83.43 ± 1.79	82.29 ± 2.81	82.20 ± 2.71
7	83.79 ± 1.74	82.53 ± 1.99	84.88 ± 0.65
8	85.12 ± 1.13	84.91 ± 1.53	85.08 ± 1.38

temperature parameter T , which can be used to control the “sharpness” of the learnt motifs.

From experiments, we find that probabilistic kernels are more stable on detecting patterns. The probabilistic kernel only outputs a distinguishing high response when it finds a good match on the sequence, otherwise keeps inactive. The conventional convolutional kernel instead produces response with more zigzags. However, this turns out not to be an advantage in terms of the network training. Comparing to the convolutional kernels, probabilistic kernels are less tolerant of small mismatch and thus require a much larger number

of them to get similar performance, which makes it not worthy.

The current two-step (training convolutional kernels and post hoc transforming into probabilistic motifs) motif discovery approach has its advantage, as the convolutional kernel is very flexible as a scoring matrix (all channels are independent) and thus is much easier to train.

CHAPTER 7

Conclusions

Deep neural networks, which brought about a revolution and became the dominant approach in computer vision and natural language processing, have been gaining popularity in more and more other research fields. However, the lack of interpretability limits their usage to wider applications which are safety-critical or have ethical and legal requirements. Besides, for those scientific research fields which have used deep neural networks, interpretability is important for scientists to gain new “knowledge”. We first conduct a comprehensive review of the current neural network interpretability research and propose a novel taxonomy which nicely organises the existing approaches in three dimensions. Then we pay special attention to the neural network based motif discovery in computational genomics. In this context, the commonly used neural network interpretation leads to the motif identifiability problem. We now briefly summarise the main content and contributions of the thesis and discuss possible future work in the following sections.

7.1 Thesis Summary

Although there are already many approaches to deal with neural network interpretability, they are often widely divergent. Many attempts were thus made in order to give a definition and possible categorisation of neural network interpretability. However, most of them fail

to be comprehensive and a clear unified taxonomy is still missing. In Chapter 3, we first give our (extended) definition of interpretability, which focuses on the resulting *explanation* rather than the pre-recognised *explainer*. This acts as an important dimension “type (or format) of explanations” in our proposed taxonomy. Along this dimension, an explanation can either be explicit (e.g., a decision tree or rule), less straightforward (e.g., a saliency map, which may need further interpretation), or implicit at all (e.g., finding a similar training example). Depending on the coverage of the input space, an explanation can be global (e.g., a decision tree that explains the whole model), local (e.g., a saliency map for a single input image), or even semi-local (explaining a group of similar inputs). Finally, an interpretation method can be passive or active according to whether it requires the network architecture or training process to change. These three orthogonal dimensions provide a clear categorisation of the existing interpretability methods.

In Chapter 4, we focus on the motif identifiability issue in neural network based motif discovery, given the commonly used motif interpretation method. In order to investigate the relationship between model performance and motif identifiability, we first propose a novel similarity measure between probabilistic motifs, in the context of convolutional kernels. Systematic controlled experiments are performed in the sequence classification task on stochastically generated synthetic datasets with known ground truth of motifs and their class structures (classification rules). We find that although high levels of motif identifiability usually imply high levels of model performance, the reverse is not necessarily true, i.e., high levels of performance do not necessarily mean high levels of motif identifiability. Usually there is a “sweet spot” of model complexity where the network is flexible enough to achieve high performance, yet constrained enough so that motifs can be well recovered by the first-layer kernels. These findings are of great importance, because model performance is often taken as the only indicator of suitability of the discovered motifs.

Then in Chapter 5 we propose a robust (semi-automatic) motif discovery workflow to enhance reliability of extracted motifs from trained neural networks. It consists of three steps: (1) determining an appropriate number of convolutional kernels, (2) training multiple

independent neural networks on re-sampled data, and (3) constructing a representative motif for each motif group. It addresses two concerns of the common neural network based motif discovery methods as found in Chapter 4, the over-complexity (without overfitting) of the model may be harmful to the motif identifiability, and the motif identifiability of a high-accuracy model may still vary in a large range. Experiments on both synthetic datasets and real-world datasets show that this workflow works reasonably well. Also, we point out the existence of “negative” phantom motifs (in contrast to the commonly known “enriched” motifs) due to the nature of neural networks as discriminative models.

In above motif discovery methods, the probabilistic motifs are always post hoc transformed from the learnt kernel weights. Therefore, in Chapter 6 we would like to explore an alternative way (probabilistic kernels) so that we can learn the motifs directly in the neural network. The probabilistic kernels have more stable response comparing to convolutional kernels on detecting sequence patterns. However, the other side of the coin is they have much less tolerance on mismatch. This is why they can perform well on the relatively simple synthetic datasets but require much more kernels on real-world datasets.

7.2 Limitations

The extensibility of the proposed methods. As introduced in Section 2.3.3, there are different deep neural network architectures (e.g., LSTM, [Quang and Xie, 2016](#); GRU, [Yang, Liu, et al., 2017](#); Transformer, [Clauwaert et al., 2021](#)) that have been used for biological sequence related tasks. In this thesis, we only focused on CNNs given their performance and popularity when the research was conducted. However, the proposed methodology can be applied to a wide range of deep learning models. In Chapter 4, we propose a similarity measure between two probabilistic motifs in order to evaluate a model’s motif identifiability, which is independent of the used models. Further, the same experiments (AU-CDF) can be used to investigate the relationship between model performance and motif identifiability. And in Chapter 5, we care about how to robustly recover the motifs underlying the data.

The main idea is to constrain the model to avoid unnecessary model complexity (in terms of the motif identifiability, may come at a slight cost of performance). In our work, this is done by controlling the number of convolutional kernels. This can still be used for models that have convolutional layers such as DanQ (Quang and Xie, 2016) and BiRen (Yang, Liu, et al., 2017). Other models may have different architecture choices depending on how they affect their motif identifiability.

The synthetic data generator and real-world datasets. The synthetic data generator (Section 4.2.2) is proposed to study the relationship between model performance and motif identifiability *in general*. Besides the flexibility, it is designed to be simple (e.g., the pre-defined templates for symbol distributions instead of arbitrary distributions) so that we can focus on the identifiability issue and avoid unexpected nuisance. However, the synthetic sequences may not be realistic enough when compared to real-world sequences, which can be much noisier. (It can be seen that classifying real-world sequences is much harder for LikelihoodNet, which is less expressive than classic ConvNet, in Section 6.3.3). In order to show the practical usefulness of our study (Chapter 4), it can be better to add additional experiments on real-world datasets.

Experimental setup. For experiments throughout this thesis, we randomly split the whole dataset into training/validation/test sets with 60%, 20% and 20% respectively, which is common in the deep learning research community given the large volume of data. However, it can still be helpful to use cross-validation rather than a single random data split to select the best hyperparameters (Section 4.3.1, Section 5.2.1, and Section 6.3.1).

7.3 Future Work

This thesis has provided a well-organised overview of the neural network interpretability research, and also systematically studied how neural networks should be properly used for motif discovery. These open up some possible future research directions:

- **New neural network interpretability methods from the taxonomy**

From the perspective of the new taxonomy, there are still several possible research directions in the interpretability research.

- First, the active interpretability intervention approaches are underexplored. Some analysis of the passive methods also suggests that the neural network does not necessarily learn representations which can be easily interpreted by human beings. Therefore, how to actively make a network interpretable without harming its performance is still an open problem. During the survey process, we have seen more and more recent work filling this blank.
- Another important research direction may be how to better incorporate domain knowledge in the networks. As we have seen in this paper, interpretability is about providing explanations. And explanations build on top of understandable terms (or concepts) which can be specific to the targeted tasks. We already have many approaches to construct explanations of different types, but the domain-related terms used in the explanations are still very simple (see Table 3.1). If we can make use of terms that are more domain/task-related, we can get more informative explanations and better interpretability.

- **Gain more complex interpretation from the neural network**

Currently, most neural network based motif discovery methods choose to interpret the first-layer kernels. This is more of a compromise due to the lack of interpretability of deep neural networks, based on which we try to investigate how should and how much identifiability we can achieve. Similar work ([Koo and Eddy, 2019](#); [Koo and Ploenzke, 2021](#)) has shown that many network architecture choices (such as the max-pooling size, the convolutional kernel size, and the choice of first-layer activation function) may affect the motif representation among kernels, thus affecting the motif identifiability studied in this paper. Together, we get some architecture suggestions for those who would like to build networks with high-quality easy to access first-layer kernels. However, for

some tasks that may contain more complicated motif interactions, neural networks are probably required to be much deeper and complex in order to capture the underlying patterns with a distributed and hierarchical way. This requires more understanding from the neural network interpretability research and the motif identifiability in this case is worth exploring.

APPENDIX A

Supplementary Tables

Table A.1: Additional results of LikelihoodNet on real dataset FOXA2.

Num of kernels	8	10	12	14	16
Performance	85.12 ± 1.13	86.65 ± 0.60	87.90 ± 0.93	89.00 ± 0.24	89.00 ± 0.37
Num of kernels	18	20	22	24	25
Performance	89.61 ± 0.50	90.08 ± 0.37	90.56 ± 0.32	90.76 ± 0.41	91.19 ± 0.24

Table A.2: The one-sided Wilcoxon signed-rank test results for Table 6.2 (extended to 15 runs for each number of kernels). It is a non-parametric version of the paired Student's t -test. The null hypothesis is that the performance of LikelihoodNet is not greater than that of ConvNet. When the number of kernels is larger than the number of true motifs (4 for synthetic dataset 1, and 8 for synthetic dataset 3), the performance difference is significant (the p -values are almost always less than 0.05).

Synthetic dataset 1				
Num of kernels	ConvNet	LikelihoodNet	Statistic	p -value
3	93.43 ± 0.27	93.53 ± 0.30	79	0.151
4	94.40 ± 0.20	94.64 ± 0.42	88	0.060
5	94.38 ± 0.19	94.65 ± 0.44	92	0.036
6	94.39 ± 0.16	94.79 ± 0.23	119	6.104×10^{-5}
7	94.43 ± 0.15	94.87 ± 0.18	120	3.052×10^{-5}
8	94.43 ± 0.12	94.96 ± 0.12	120	3.052×10^{-5}
Synthetic dataset 3				
Num of kernels	ConvNet	LikelihoodNet	statistic	p -value
7	93.72 ± 0.20	93.76 ± 0.43	74	0.227
8	93.79 ± 0.22	94.16 ± 0.28	109	0.002
9	94.01 ± 0.21	94.31 ± 0.26	102	0.008
10	94.22 ± 0.14	94.33 ± 0.26	87	0.068
11	94.18 ± 0.16	94.40 ± 0.18	98	0.002
12	94.23 ± 0.11	94.42 ± 0.12	115	3.052×10^{-4}

APPENDIX B

Supplementary Figures

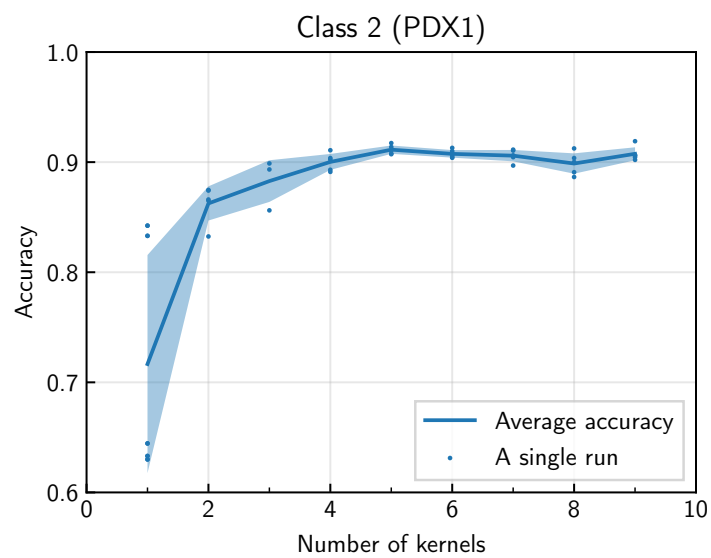


Figure B.1: The (validation set) model performance vs. number of kernels curve (real dataset 2, PDX1). The shadow means one standard derivation (plus or minus).

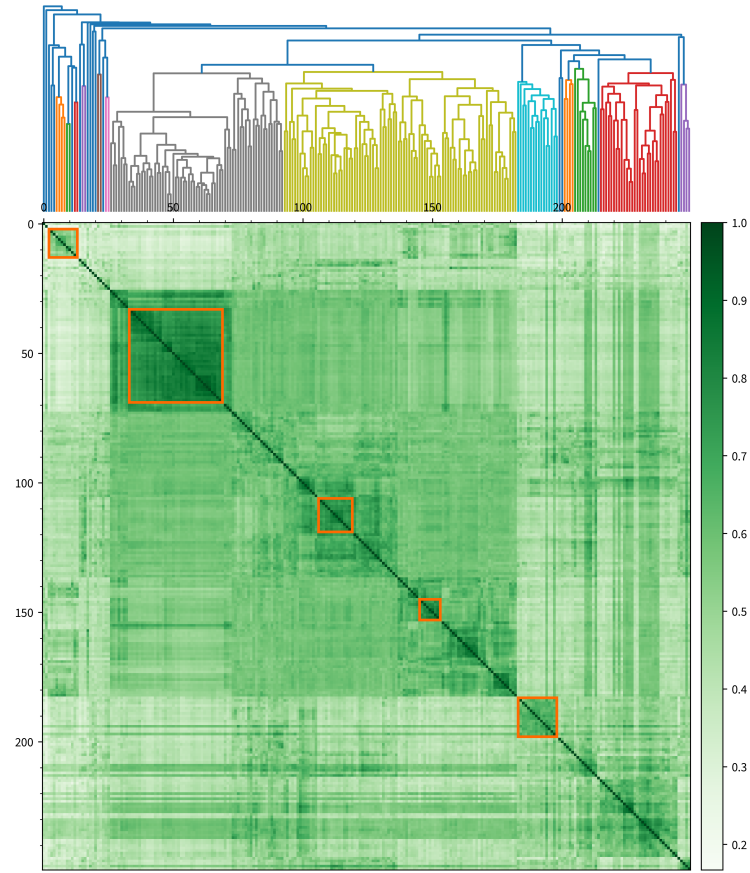


Figure B.2: The hierarchical clustering result of learned kernels (real dataset 2, PDX1).

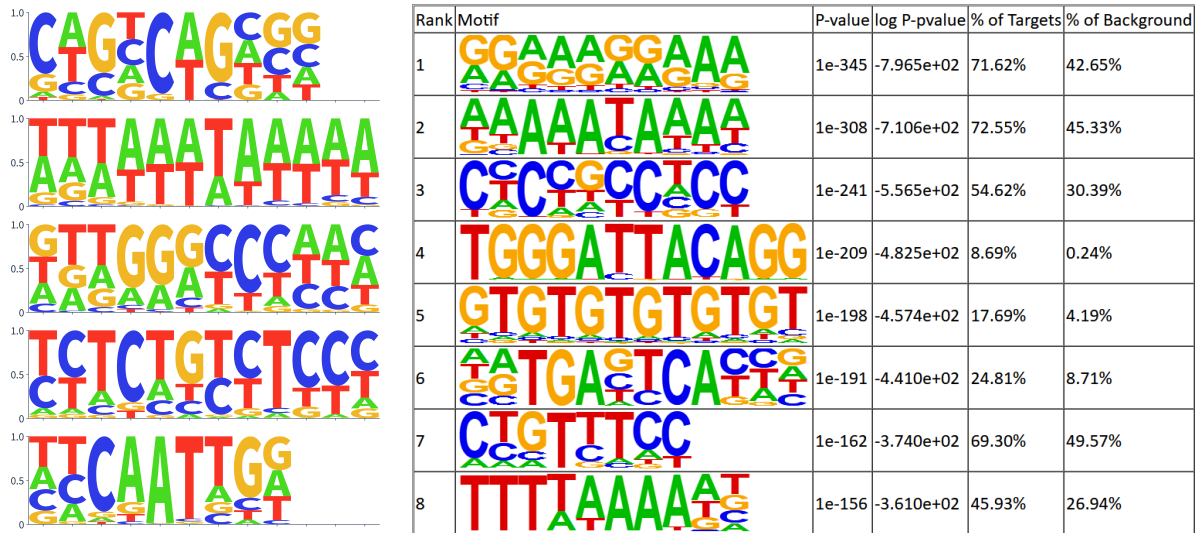


Figure B.3: Motifs found by the neural network (left) and Homer (right) on real dataset 2 (PDX1). The confidence level of Homer is much lower than that of dataset 1.

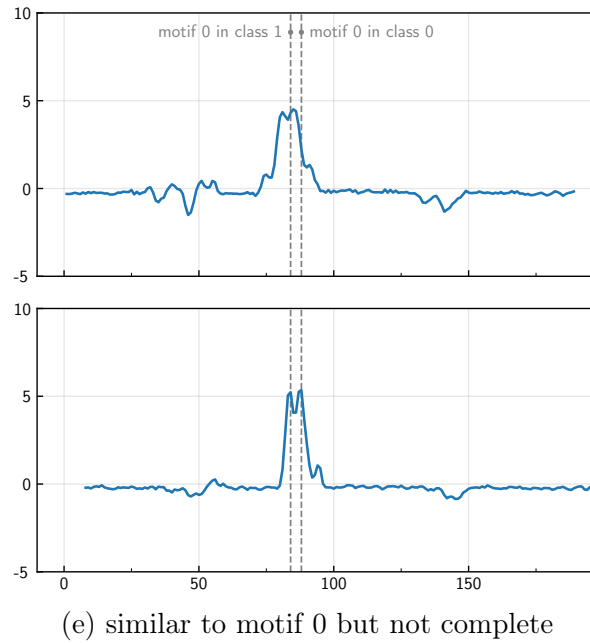
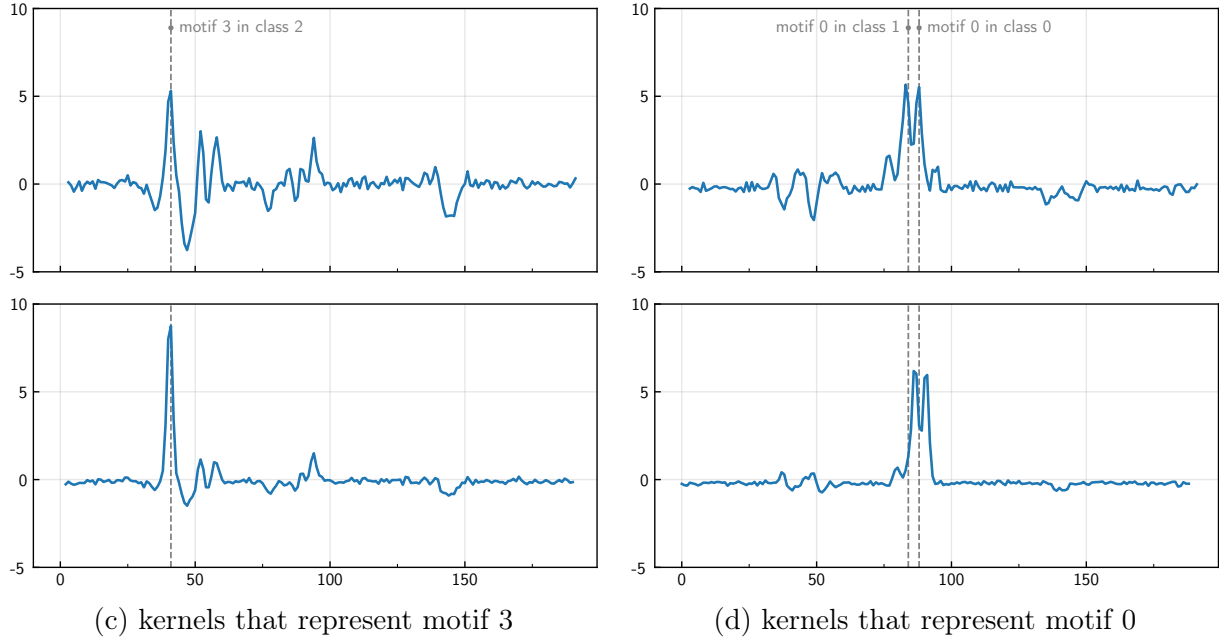


Figure B.4: The complete results for Figure 6.1. The probabilistic kernels (second row) in (d) and (e) jointly represent the motif 0 (overlapping on one nucleotide), which are $\begin{smallmatrix} \text{GAA}^{\text{T}} \\ \text{A} \end{smallmatrix}$, $\begin{smallmatrix} \text{TCC}^{\text{A}} \\ \text{AAT} \end{smallmatrix}$ and $\begin{smallmatrix} \text{GAA}^{\text{TCT}} \\ \text{A} \end{smallmatrix}$ respectively.

References

- [1] Julius Adebayo et al. “Sanity Checks for Saliency Maps”. *Advances in Neural Information Processing Systems*. Vol. 31. 2018.
- [2] Philip Adler et al. “Auditing black-box models for indirect influence”. *Knowledge and Information Systems* 54.1 (2018), pp. 95–122.
- [3] Babak Alipanahi, Andrew DeLong, Matthew T Weirauch, and Brendan J Frey. “Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning”. *Nature Biotechnology* 33.8 (2015), pp. 831–838.
- [4] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. “Towards better understanding of gradient-based attribution methods for Deep Neural Networks”. *International Conference on Learning Representations*. 2018.
- [5] Marco Ancona, Cengiz Oztireli, and Markus Gross. “Explaining Deep Neural Networks with a Polynomial Time Algorithm for Shapley Value Approximation”. *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. 2019.
- [6] Robert Andrews, Joachim Diederich, and Alan B Tickle. “Survey and critique of techniques for extracting rules from trained artificial neural networks”. *Knowledge-based systems* 8.6 (1995), pp. 373–389.

-
- [7] Floriane Anstett-Collin, Lilianne Denis-Vidal, and Gilles Millérioux. “A priori identifiability: An overview on definitions and approaches”. *Annual Reviews in Control* (2020).
 - [8] Alejandro Barredo Arrieta et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. *Information Fusion* 58 (2020), pp. 82–115.
 - [9] Sebastian Bach et al. “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation”. *PloS one* 10.7 (2015), e0130140.
 - [10] David Baehrens et al. “How to explain individual classification decisions”. *The Journal of Machine Learning Research* 11 (2010), pp. 1803–1831.
 - [11] Timothy L Bailey. “DREME: motif discovery in transcription factor ChIP-seq data”. *Bioinformatics* 27.12 (2011), pp. 1653–1659.
 - [12] Timothy L Bailey, Mikael Boden, et al. “MEME SUITE: tools for motif discovery and searching”. *Nucleic acids research* (2009).
 - [13] Timothy L Bailey and Charles Elkan. “Fitting a mixture model by expectation maximization to discover motifs in biopolymers”. *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*. 1994.
 - [14] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. “Searching for exotic particles in high-energy physics with deep learning”. *Nature communications* 5.1 (2014), pp. 1–9.
 - [15] Andrew R Barron. “Approximation and estimation bounds for artificial neural networks”. *Machine learning* 14.1 (1994), pp. 115–133.
 - [16] David Bau, Bolei Zhou, et al. “Network dissection: Quantifying interpretability of deep visual representations”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6541–6549.

-
- [17] David Bau, Jun-Yan Zhu, et al. “GAN Dissection: Visualizing and Understanding Generative Adversarial Networks”. *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.
- [18] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854.
- [19] J. M. Benitez, J. L. Castro, and I. Requena. “Are artificial neural networks black boxes?” *IEEE Transactions on Neural Networks* 8 (1997).
- [20] Jacob Bien and Robert Tibshirani. “Prototype selection for interpretable classification”. *The Annals of Applied Statistics* 5.4 (2011), pp. 2403–2424.
- [21] Francesco Bodria et al. “Benchmarking and survey of explanation methods for black box models”. *arXiv preprint arXiv:2102.13076* (2021).
- [22] Thierry Bouwmans, Sajid Javed, Maryam Sultana, and Soon Ki Jung. “Deep neural network concepts for background subtraction: A systematic review and comparative evaluation”. *Neural Networks* 117 (2019), pp. 8–66.
- [23] Olcay Boz. “Extracting decision trees from trained neural networks”. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 456–461.
- [24] Tom Brown et al. “Language Models are Few-Shot Learners”. *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1877–1901.
- [25] Joan Bruna and Stéphane Mallat. “Invariant scattering convolution networks”. *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1872–1886.
- [26] Rich Caruana et al. “Case-based explanation of non-case-based learning methods.” *Proceedings of the AMIA Symposium*. 1999, p. 212.
- [27] J. L. Castro, C. J. Mantas, and J. M. Benitez. “Interpretation of artificial neural networks by means of fuzzy rules”. *IEEE Transactions on Neural Networks* 13 (2002).

-
- [28] Augustin Cauchy et al. “Méthode générale pour la résolution des systemes d’équations simultanées”. *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.
 - [29] Bo Chang et al. “Reversible Architectures for Arbitrarily Deep Residual Neural Networks”. *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018).
 - [30] Chaofan Chen, Oscar Li, et al. “This Looks Like That: Deep Learning for Interpretable Image Recognition”. *Advances in Neural Information Processing Systems*. Vol. 32. 2019.
 - [31] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. “Learning to Explain: An Information-Theoretic Perspective on Model Interpretation”. *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018.
 - [32] Anna Choromanska et al. “The Loss Surfaces of Multilayer Networks”. *International Conference on Artificial Intelligence and Statistics*. Vol. 38. 2015.
 - [33] Jim Clauwaert, Gerben Menschaert, and Willem Waegeman. “Explainability in transformer models for functional genomics”. *Briefings in Bioinformatics* 22.5 (2021).
 - [34] Athel Cornish-Bowden. “Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984.” *Nucleic acids research* 13.9 (1985), p. 3021.
 - [35] Mark Craven and Jude Shavlik. “Extracting tree-structured representations of trained networks”. *Advances in neural information processing systems* 8 (1995), pp. 24–30.
 - [36] Mark W Craven and Jude W Shavlik. “Using sampling and queries to extract rules from trained neural networks”. In: *Machine learning proceedings 1994*. 1994, pp. 37–45.
 - [37] Antonia Creswell and Anil Anthony Bharath. “Denoising adversarial autoencoders”. *IEEE transactions on neural networks and learning systems* 30.4 (2018), pp. 968–984.
 - [38] George Cybenko. “Approximation by superpositions of a sigmoidal function”. *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

-
- [39] Fahim Dalvi et al. “What is one grain of sand in the desert? analyzing individual neurons in deep nlp models”. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2019.
- [40] William HE Day and FR McMorris. “Critical comparison of consensus methods for molecular sequences”. *Nucleic acids research* 20.5 (1992), pp. 1093–1099.
- [41] Amit Dhurandhar et al. “Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives”. *Advances in Neural Information Processing Systems*. Vol. 31. 2018.
- [42] Priyanka Dixit and Sanjay Silakari. “Deep Learning Algorithms for Cybersecurity Applications: A Technological and Status Review”. *Computer Science Review* 39 (2021), p. 100317.
- [43] Ann-Kathrin Dombrowski et al. “Explanations can be manipulated and geometry is to blame”. *Advances in Neural Information Processing Systems*. Vol. 32. 2019.
- [44] Shi Dong, Ping Wang, and Khushnood Abbas. “A survey on deep learning and its applications”. *Computer Science Review* 40 (2021), p. 100379.
- [45] Finale Doshi-Velez and Been Kim. “Towards A Rigorous Science of Interpretable Machine Learning”. *arXiv* (2017).
- [46] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2019. URL: <http://archive.ics.uci.edu/ml>.
- [47] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for on-line learning and stochastic optimization.” *Journal of machine learning research* 12.7 (2011).
- [48] Ronen Eldan and Ohad Shamir. “The power of depth for feedforward neural networks”. *Conference on learning theory*. 2016, pp. 907–940.

-
- [49] Gökçen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J Theis. “Deep learning: new computational modelling techniques for genomics”. *Nature Reviews Genetics* 20.7 (2019), pp. 389–403.
- [50] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Visualizing higher-layer features of a deep network”. *University of Montreal* 1341.3 (2009), p. 1.
- [51] European Parliament, Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. Official Journal of the European Union. <https://eur-lex.europa.eu/eli/reg/2016/679/oj>. 2016.
- [52] Ruth Fong and Andrea Vedaldi. “Interpretable explanations of black boxes by meaningful perturbation”. *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3429–3437.
- [53] Ruth Fong and Andrea Vedaldi. “Net2Vec: Quantifying and Explaining How Concepts Are Encoded by Filters in Deep Neural Networks”. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [54] LiMin Fu. “Rule Learning by Searching on Adapted Nets.” *AAAI*. Vol. 91. 1991, pp. 590–595.
- [55] Ken-Ichi Funahashi. “On the approximate realization of continuous mappings by neural networks”. *Neural networks* 2.3 (1989), pp. 183–192.
- [56] Erik Gawehn, Jan A Hiss, and Gisbert Schneider. “Deep learning in drug discovery”. *Molecular informatics* 35.1 (2016), pp. 3–14.
- [57] Stefanie Gerstberger, Markus Hafner, and Thomas Tuschl. “A census of human RNA-binding proteins”. *Nature Reviews Genetics* 15.12 (2014), pp. 829–845.

-
- [58] Amirata Ghorbani, Abubakar Abid, and James Zou. “Interpretation of Neural Networks Is Fragile”. *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 3681–3688.
 - [59] Amirata Ghorbani, James Wexler, James Zou, and Been Kim. “Towards automatic concept-based explanations”. *Advances in Neural Information Processing Systems*. 2019, pp. 9277–9286.
 - [60] Leilani H Gilpin et al. “Explaining explanations: An overview of interpretability of machine learning”. *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. 2018.
 - [61] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
 - [62] Ian Goodfellow, Jean Pouget-Abadie, et al. “Generative Adversarial Nets”. *Advances in Neural Information Processing Systems*. Vol. 27. 2014.
 - [63] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a “right to explanation””. *AI magazine* 38.3 (2017), pp. 50–57.
 - [64] Yash Goyal et al. “Counterfactual Visual Explanations”. *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. 2019.
 - [65] Riccardo Guidotti et al. “A Survey of Methods for Explaining Black Box Models”. *ACM Comput. Surv.* 51.5 (2018).
 - [66] Shobhit Gupta, John A Stamatoyannopoulos, Timothy L Bailey, and William Stafford Noble. “Quantifying similarity between motifs”. *Genome biology* 8.2 (2007), pp. 1–9.
 - [67] Eldad Haber and Lars Ruthotto. “Stable architectures for deep neural networks”. *Inverse problems* 34.1 (2017), p. 014004.
 - [68] Benjamin D Haeffele and René Vidal. “Global optimality in neural network training”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7331–7339.

-
- [69] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. “GANSpace: Discovering Interpretable GAN Controls”. *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 9841–9850.
- [70] Fatma A Hashim, Mai S Mabrouk, and Walid Al-Atabany. “Review of different sequence motif finding algorithms”. *Avicenna journal of medical biotechnology* 11.2 (2019), p. 130.
- [71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.
- [73] Ying He, Zhen Shen, et al. “A survey on deep learning in DNA/RNA motif mining”. *Briefings in Bioinformatics* 22.4 (2021).
- [74] Sven Heinz et al. “Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities”. *Molecular cell* 38.4 (2010), pp. 576–589.
- [75] Juyeon Heo, Sunghwan Joo, and Taesup Moon. “Fooling Neural Network Interpretations via Adversarial Model Manipulation”. *Advances in Neural Information Processing Systems*. Vol. 32. 2019.
- [76] Tom Heskes, Evi Sijben, Ioan Gabriel Bucur, and Tom Claassen. “Causal Shapley Values: Exploiting Causal Knowledge to Explain Individual Predictions of Complex Models”. *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 4778–4789.
- [77] Geoffrey Hinton et al. “Deep neural networks for acoustic modeling in speech recognition”. *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97.

-
- [78] Jake M Hofman, Amit Sharma, and Duncan J Watts. “Prediction and explanation in social systems”. *Science* 355.6324 (2017), pp. 486–488.
- [79] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. “A Benchmark for Interpretability Methods in Deep Neural Networks”. *Advances in Neural Information Processing Systems*. Vol. 32. 2019.
- [80] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. *Neural Networks* 2.5 (1989), pp. 359–366.
- [81] Jason D Hughes, Preston W Estep, Saeed Tavazoie, and George M Church. “Computational identification of cis-regulatory elements associated with groups of functionally related genes in *Saccharomyces cerevisiae*”. *Journal of molecular biology* 296.5 (2000), pp. 1205–1214.
- [82] Minsuk Kahng et al. “Gan lab: Understanding complex deep generative models using interactive visual experimentation”. *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 310–320.
- [83] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. “DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization”. *Proceedings of the International Joint Conference on Artificial Intelligence*. 2020.
- [84] David R Kelley, Jasper Snoek, and John L Rinn. “Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks”. *Genome research* 26.7 (2016), pp. 990–999.
- [85] Been Kim et al. “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)”. *International Conference on Machine Learning*. 2018.
- [86] Pieter-Jan Kindermans et al. “The (un)reliability of saliency methods”. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. 2019, pp. 267–280.

-
- [87] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. *International Conference on Learning Representations*. 2015.
- [88] Pang Wei Koh and Percy Liang. “Understanding black-box predictions via influence functions”. *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 2017.
- [89] Utsav Garg Konda Reddy Mopuri and Venkatesh Babu Radhakrishnan. “Fast Feature Fool: A data independent approach to universal adversarial perturbations”. *Proceedings of the British Machine Vision Conference (BMVC)*. 2017, pp. 30.1–30.12.
- [90] Peter K Koo and Sean R Eddy. “Representation learning of genomic sequence motifs with convolutional neural networks”. *PLOS Computational Biology* 15.12 (2019), pp. 1–17.
- [91] Peter K Koo and Matt Ploenzke. “Improving representations of genomic sequence motifs in convolutional networks with exponential activations”. *Nature Machine Intelligence* 3.3 (2021), pp. 258–266.
- [92] R Krishnan, G Sivakumar, and P Bhattacharya. “Extracting decision trees from trained neural networks”. *Pattern recognition* 32.12 (1999).
- [93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. *Advances in neural information processing systems*. Vol. 25. 2012, pp. 1097–1105.
- [94] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.
- [95] Sebastian Lapuschkin et al. “Unmasking Clever Hans predictors and assessing what machines really learn”. *Nature communications* 10.1 (2019), pp. 1–8.
- [96] Charles E Lawrence and Andrew A Reilly. “An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopoly-

- mer sequences”. *Proteins: Structure, Function, and Bioinformatics* 7.1 (1990), pp. 41–51.
- [97] Guy Lebanon and John Lafferty. “Riemannian Geometry and Statistical Machine Learning”. Ph.D. dissertation. Carnegie Mellon University, 2005.
- [98] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. *nature* 521.7553 (2015), pp. 436–444.
- [99] Yann LeCun, Bernhard Boser, et al. “Handwritten digit recognition with a back-propagation network”. *Advances in neural information processing systems* 2 (1989).
- [100] Christian Ledig et al. “Photo-realistic single image super-resolution using a generative adversarial network”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
- [101] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”. *Neural networks* 6.6 (1993), pp. 861–867.
- [102] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. “Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions”. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.
- [103] Zachary C Lipton. “The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery.” *Queue* 16.3 (2018), pp. 31–57.
- [104] Xiaole Liu, Douglas L Brutlag, and Jun S Liu. “BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes”. *Proceedings of the Pacific Symposium*. 2001, pp. 127–138.
- [105] Wei-Yin Loh. “Classification and regression trees”. *Wiley interdisciplinary reviews: data mining and knowledge discovery* 1.1 (2011), pp. 14–23.

-
- [106] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4768–4777.
- [107] Aravindh Mahendran and Andrea Vedaldi. “Understanding deep image representations by inverting them”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5188–5196.
- [108] Shaun Mahony and Panayiotis V Benos. “STAMP: a web tool for exploring DNA-binding motif similarities”. *Nucleic acids research* 35.suppl_2 (2007), W253–W258.
- [109] Ričards Marcinkevičs and Julia E Vogt. “Interpretability and explainability: A machine learning zoo mini-tour”. *arXiv preprint arXiv:2012.01805* (2020).
- [110] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. *Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133.
- [111] Poorya Mianjy, Raman Arora, and Rene Vidal. “On the implicit bias of dropout”. *International Conference on Machine Learning*. 2018, pp. 3540–3548.
- [112] Tomas Mikolov et al. “Distributed representations of words and phrases and their compositionality”. *Advances in neural information processing systems*. 2013, pp. 3111–3119.
- [113] Tsubasa Minematsu, Atsushi Shimada, and Rin-ichiro Taniguchi. “Analytics of deep neural network in change detection”. *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017.
- [114] Tsubasa Minematsu, Atsushi Shimada, Hideaki Uchiyama, and Rin-ichiro Taniguchi. “Analytics of deep neural network-based background subtraction”. *Journal of Imaging* 4.6 (2018), p. 78.
- [115] S. Mitra and Y. Hayashi. “Neuro-fuzzy rule generation: survey in soft computing framework”. *IEEE Transactions on Neural Networks* 11 (2000).

-
- [116] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. “Interpretable Machine Learning—A Brief History, State-of-the-Art and Challenges”. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2020, pp. 417–431.
- [117] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. “Methods for interpreting and understanding deep neural networks”. *Digital Signal Processing* 73 (2018), pp. 1–15.
- [118] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. “Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data”. *Machine learning* 52.1 (2003), pp. 91–118.
- [119] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. “Universal adversarial perturbations”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
- [120] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. “NAG: Network for adversary generation”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
- [121] Raha Moraffah et al. “Causal Interpretability for Machine Learning - Problems, Methods and Evaluation”. *ACM SIGKDD Exploration Newsletter* 22.1 (2020), pp. 18–33.
- [122] Karthikeyan Natesan Ramamurthy, Bhanukiran Vinzamuri, Yunfeng Zhang, and Amit Dhurandhar. “Model Agnostic Multilevel Explanations”. *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5968–5979.
- [123] Richi Nayak. “Generating rules with predicates, terms and variables from the pruned neural networks”. *Neural Networks* 22.4 (2009), pp. 405–414.
- [124] Yurii E Nesterov. “A method for solving the convex programming problem with convergence rate $O(1/k^2)$ ”. *Dokl. akad. nauk Sssr*. Vol. 269. 1983, pp. 543–547.

-
- [125] Anh Nguyen, Alexey Dosovitskiy, et al. “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks”. *Advances in neural information processing systems* 29 (2016), pp. 3387–3395.
- [126] Anh Nguyen, Jason Yosinski, and Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 427–436.
- [127] Maher Nouiehed and Meisam Razaviyayn. “Learning Deep Models: Critical Points and Local Openness”. *International Conference on Learning Representations (Workshop)*. 2018.
- [128] Koichi Odajima, Yoichi Hayashi, Gong Tianxia, and Rudy Setiono. “Greedy rule generation from discrete data and its use in neural network rule extraction”. *Neural Networks* 21.7 (2008), pp. 1020–1028.
- [129] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. “Feature Visualization”. *Distill* (2017). <https://distill.pub/2017/feature-visualization>.
- [130] Nicolas Papernot et al. “The limitations of deep learning in adversarial settings”. *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2016.
- [131] Yongjin Park and Manolis Kellis. “Deep learning for regulatory genomics”. *Nature biotechnology* 33.8 (2015), pp. 825–826.
- [132] David Parks, J Xavier Prochaska, Shawfeng Dong, and Zheng Cai. “Deep learning of quasar spectra to discover and characterize damped Ly α systems”. *Monthly Notices of the Royal Astronomical Society* 476.1 (2018), pp. 1151–1168.
- [133] Judea Pearl. *Causality*. Cambridge university press, 2009.
- [134] Tejaswini Pedapati, Avinash Balakrishnan, Karthikeyan Shanmugam, and Amit Dhurandhar. “Learning Global Transparent Models consistent with Local Contrastive Explanations”. *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 3592–3602.

-
- [135] Dino Pedreschi et al. “Meaningful explanations of Black Box AI decision systems”. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 9780–9784.
- [136] Gordon D Plotkin. “A note on inductive generalization”. *Machine intelligence* 5 (1970), pp. 153–163.
- [137] Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. “Model Agnostic Supervised Local Explanations”. *Advances in Neural Information Processing Systems*. Vol. 31. 2018.
- [138] Gregory Plumb, Maruan Al-Shedivat, et al. “Regularizing Black-box Models for Improved Interpretability”. *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 10526–10536.
- [139] Antoine Plumerault, Hervé Le Borgne, and Céline Hudelot. “Controlling generative models with continuous factors of variations”. *International Conference on Learning Representations*. 2020.
- [140] Ning Qian. “On the momentum term in gradient descent learning algorithms”. *Neural networks* 12.1 (1999), pp. 145–151.
- [141] Daniel Quang and Xiaohui Xie. “DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences”. *Nucleic acids research* 44.11 (2016), e107–e107.
- [142] J Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [143] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. *ICLR* (2016).
- [144] Andreas Raue et al. “Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood”. *Bioinformatics* 25.15 (2009), pp. 1923–1929.

-
- [145] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Anchors: High-precision model-agnostic explanations”. *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.
- [146] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Why should i trust you?: Explaining the predictions of any classifier”. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.
- [147] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.
- [148] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. *Nature* 323.6088 (1986), pp. 533–536.
- [149] Hojjat Salehinejad and Shahrokh Valaee. “Ising-dropout: A regularization method for training and compression of deep neural networks”. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 3602–3606.
- [150] Shaeke Salman et al. “DeepConsensus: Consensus-based Interpretable Deep Neural Networks with Application to Mortality Prediction”. *International Joint Conference on Neural Networks (IJCNN)*. 2020.
- [151] Wojciech Samek, Alexander Binder, et al. “Evaluating the visualization of what a deep neural network has learned”. *IEEE transactions on neural networks and learning systems* 28 (2016).
- [152] Wojciech Samek and Klaus-Robert Müller. “Towards explainable artificial intelligence”. In: *Explainable AI: interpreting, explaining and visualizing deep learning*. 2019, pp. 5–22.

-
- [153] Thomas D Schneider and R Michael Stephens. “Sequence logos: a new way to display consensus sequences”. *Nucleic acids research* 18.20 (1990), pp. 6097–6100.
- [154] Ramprasaath R Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. *Proceedings of the IEEE International Conference on Computer Vision*. 2017.
- [155] Biswa Sengupta and Karl J Friston. “How robust are deep neural networks?” *arXiv preprint arXiv:1804.11313* (2018).
- [156] Rudy Setiono and Huan Liu. “NeuroLinear: From neural networks to oblique decision rules”. *Neurocomputing* 17.1 (1997), pp. 1–24.
- [157] Rudy Setiono and Huan Liu. “Understanding neural networks via rule extraction”. *IJCAI*. Vol. 1. Citeseer. 1995, pp. 480–485.
- [158] Lloyd S Shapley. “A value for n-person games”. *Contributions to the Theory of Games* 2 (1953).
- [159] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. “Learning Important Features Through Propagating Activation Differences”. *Proceedings of the 34th International Conference on Machine Learning*. 2017.
- [160] Avanti Shrikumar, Katherine Tian, et al. “Technical note on transcription factor motif discovery from importance scores (TF-MoDISco) version 0.5.6.5”. *arXiv preprint arXiv:1811.00416* (2018).
- [161] Christian JA Sigrist et al. “PROSITE: a documented database using patterns and profiles as motif descriptors”. *Briefings in bioinformatics* 3.3 (2002), pp. 265–274.
- [162] David Silver et al. “Mastering the game of go without human knowledge”. *nature* 550.7676 (2017), pp. 354–359.
- [163] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. *International Conference on Learning Representations (Workshop Poster)*. 2013.

-
- [164] Gregory E Sims, Se-Ran Jun, Guohong A Wu, and Sung-Hou Kim. “Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions”. *Proceedings of the National Academy of Sciences* 106.8 (2009), pp. 2677–2682.
- [165] Saurabh Sinha and Martin Tompa. “YMF: a program for discovery of novel transcription factor binding sites by statistical overrepresentation”. *Nucleic acids research* 31.13 (2003), pp. 3586–3588.
- [166] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. “Striving for Simplicity: The All Convolutional Net”. *International Conference on Learning Representations (workshop track)*. 2015.
- [167] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.
- [168] Gary D Stormo. “DNA binding sites: representation and discovery”. *Bioinformatics* 16.1 (2000), pp. 16–23.
- [169] Gary D Stormo, Thomas D Schneider, Larry Gold, and Andrzej Ehrenfeucht. “Use of the ‘Perceptron’ algorithm to distinguish translational initiation sites in *E. coli*”. *Nucleic acids research* 10.9 (1982), pp. 2997–3011.
- [170] Alexander Strehl and Joydeep Ghosh. “Cluster ensembles—a knowledge reuse framework for combining multiple partitions”. *Journal of machine learning research* 3 (2002), pp. 583–617.
- [171] Erik Strumbelj and Igor Kononenko. “An efficient explanation of individual classifications using game theory”. *The Journal of Machine Learning Research* 11 (2010), pp. 1–18.
- [172] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic attribution for deep networks”. *International Conference on Machine Learning*. PMLR. 2017, pp. 3319–3328.

-
- [173] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. *Advances in neural information processing systems*. 2014, pp. 3104–3112.
- [174] Christian Szegedy et al. “Intriguing properties of neural networks”. *International Conference on Learning Representations*. 2014.
- [175] Emi Tanaka et al. “Improved similarity scores for comparing motifs”. *Bioinformatics* 27.12 (2011), pp. 1603–1609.
- [176] Ammar Tareen and Justin B Kinney. “Logomaker: beautiful sequence logos in Python”. *Bioinformatics* 36.7 (2019), pp. 2272–2274.
- [177] Kiran K Thekumparampil, Ashish Khetan, Zinan Lin, and Sewoong Oh. “Robustness of conditional GANs to noisy labels”. *Advances in Neural Information Processing Systems*. Vol. 31. 2018.
- [178] A. B. Tickle, R. Andrews, M. Golea, and J. Diederich. “The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks”. *IEEE Transactions on Neural Networks* 9 (1998).
- [179] Geoffrey G Towell and Jude W Shavlik. “Extracting refined rules from knowledge-based neural networks”. *Machine learning* 13.1 (1993), pp. 71–101.
- [180] Ameni Trabelsi, Mohamed Chaabane, and Asa Ben-Hur. “Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities”. *Bioinformatics* 35.14 (2019), pp. i269–i277.
- [181] Mark K Transtrum, Gus Hart, and Peng Qiu. “Information topology identifies emergent model classes”. *arXiv preprint arXiv:1409.6203* (2014).
- [182] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.
- [183] Ashish Vaswani et al. “Attention is All you Need”. *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

-
- [184] Rene Vidal, Joan Bruna, Raja Giryes, and Stefano Soatto. “Mathematics of deep learning”. *arXiv preprint arXiv:1712.04741* (2017).
- [185] Alejandro F Villaverde, Antonio Barreiro, and Antonis Papachristodoulou. “Structural identifiability of dynamic systems biology models”. *PLoS computational biology* 12.10 (2016), e1005153.
- [186] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. *Nature Methods* 17 (2020), pp. 261–272.
- [187] Andrey Voynov and Artem Babenko. “Unsupervised discovery of interpretable directions in the gan latent space”. *International Conference on Machine Learning*. 2020, pp. 9786–9796.
- [188] Sandra Wachter, Brent Mittelstadt, and Chris Russell. “Counterfactual explanations without opening the black box: Automated decisions and the GDPR”. *Harv. JL & Tech.* 31 (2017), p. 841.
- [189] Michael Wainberg, Daniele Merico, Andrew Delong, and Brendan J Frey. “Deep learning in biomedicine”. *Nature biotechnology* 36.9 (2018), pp. 829–838.
- [190] Eric Walter and Luc Pronzato. *Identification of parametric models: from experimental data*. Springer Verlag, 1997.
- [191] Feng Wang, Haijun Liu, and Jian Cheng. “Visualizing deep neural network by alternately image blurring and deblurring”. *Neural Networks* 97 (2018), pp. 162–172.
- [192] Shengjie Wang, Tianyi Zhou, and Jeff Bilmes. “Bias Also Matters: Bias Attribution for Deep Neural Network Explanation”. *International Conference on Machine Learning*. 2019.
- [193] Tong Wang. “Gaining Free or Low-Cost Interpretability with Interpretable Partial Substitute”. *International Conference on Machine Learning*. 2019.

-
- [194] Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. “Interpret neural networks by identifying critical data routing paths”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
- [195] Ethan Weinberger, Joseph Janizek, and Su-In Lee. “Learning Deep Attribution Priors Based On Prior Knowledge”. *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 14034–14045.
- [196] Ashia C. Wilson et al. “The Marginal Value of Adaptive Gradient Methods in Machine Learning”. *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4148–4158.
- [197] Maksymilian Wojtas and Ke Chen. “Feature Importance Ranking for Deep Learning”. *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5105–5114.
- [198] Mike Wu, Michael Hughes, et al. “Beyond Sparsity: Tree Regularization of Deep Models for Interpretability”. *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018).
- [199] Mike Wu, Sonali Parbhoo, et al. “Regional tree regularization for interpretability in deep neural networks”. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 6413–6421.
- [200] Yonghui Wu, Mike Schuster, et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. *arXiv preprint arXiv:1609.08144* (2016).
- [201] Hui Y Xiong et al. “The human splicing code reveals new insights into the genetic determinants of disease”. *Science* 347.6218 (2015).
- [202] Bite Yang, Feng Liu, et al. “BiRen: predicting enhancers with a deep-learning-based model using the DNA sequence alone”. *Bioinformatics* 33.13 (2017), pp. 1930–1936.

-
- [203] Ceyuan Yang, Yujun Shen, and Bolei Zhou. “Semantic hierarchy emerges in deep generative representations for scene synthesis”. *International Journal of Computer Vision* 129.5 (2021), pp. 1451–1466.
 - [204] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. “Representer Point Selection for Explaining Deep Neural Networks”. *Advances in Neural Information Processing Systems*. Vol. 31. 2018.
 - [205] Jason Yosinski et al. “Understanding neural networks through deep visualization”. *arXiv preprint arXiv:1506.06579* (2015).
 - [206] Qiang Yu et al. “An efficient algorithm for discovering motifs in large DNA data sets”. *IEEE Transactions on NanoBioscience* 14.5 (2015), pp. 535–544.
 - [207] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. “Global Optimality Conditions for Deep Neural Networks”. *International Conference on Learning Representations*. 2018.
 - [208] Matthew D Zeiler. “Adadelta: an adaptive learning rate method”. *arXiv preprint arXiv:1212.5701* (2012).
 - [209] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. *European conference on computer vision*. Springer. 2014, pp. 818–833.
 - [210] Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. “Convolutional neural network architectures for predicting DNA–protein binding”. *Bioinformatics* 32.12 (2016), pp. i121–i127.
 - [211] Chiyuan Zhang, Samy Bengio, et al. “Understanding deep learning (still) requires rethinking generalization”. *Communications of the ACM* 64.3 (2021), pp. 107–115.
 - [212] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. “Interpretable convolutional neural networks”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8827–8836.

-
- [213] Quanshi Zhang and Song-Chun Zhu. “Visual interpretability for deep learning: a survey”. *Frontiers of Information Technology & Electronic Engineering* 19.1 (2018), pp. 27–39.
- [214] Sai Zhang, Hailin Hu, et al. “TITER: predicting translation initiation sites by deep learning”. *Bioinformatics* 33.14 (2017), pp. i234–i242.
- [215] Yu Zhang, Peter Tiño, Aleš Leonardis, and Ke Tang. “A Survey on Neural Network Interpretability”. *IEEE Transactions on Emerging Topics in Computational Intelligence* 5.5 (2021), pp. 726–742.
- [216] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. “Improving the robustness of deep neural networks via stability training”. *Proceedings of the iee conference on computer vision and pattern recognition*. 2016.
- [217] Zhihao Zheng and Pengyu Hong. “Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks”. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018.
- [218] B. Zhou, D. Bau, A. Oliva, and A. Torralba. “Interpreting Deep Visual Representations via Network Dissection”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).
- [219] Bolei Zhou, Aditya Khosla, et al. “Learning deep features for discriminative localization”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- [220] Bolei Zhou, Aditya Khosla, et al. “Object Detectors Emerge in Deep Scene CNNs”. *3rd International Conference on Learning Representations*. 2015.
- [221] Jian Zhou and Olga G Troyanskaya. “Predicting effects of noncoding variants with deep learning-based sequence model”. *Nature methods* 12.10 (2015), pp. 931–934.
- [222] Yi-Tong Zhou and Rama Chellappa. “Computation of optical flow using a neural network.” *ICNN*. 1988, pp. 71–78.

- [223] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. “Visualizing Deep Neural Network Decisions: Prediction Difference Analysis”. *International Conference on Learning Representations*. 2017.