INTERPRETING DEEP LEARNING MODELS

by

Xuan Liu

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
May 2020

© Copyright by Xuan Liu, 2020

Table of Contents

List of Tables . . . . . **v**

List of Figures . . . . . **vi**

Abstract . . . . . **viii**

List of Abbreviations . . . . . **ix**

Acknowledgements . . . . . **xii**

Chapter 1 Introduction **1**

1.1 The Necessities for Interpretation 3

1.2 Properties of Interpretability 4
1.2.1 Evaluations of Interpretation Methods 4
1.2.2 Definition of Interpretability 5

1.3 Contributions 6

Chapter 2 Taxonomy of Interpretation Methods **10**

2.1 Post-hoc Interpretations 11
2.1.1 Interpretation by Perturbation 11
2.1.2 Local Interpretations 13
2.1.3 Global Interpretations 16
2.1.4 Visualization 19

2.2 Inherently Interpretable Models 29

2.3 Other Methods 30

2.4 Conclusions 32

Chapter 3 Interpretable Deep Convolutional Neural Networks via Meta-learning **33**

3.1 Introduction 34

3.2 Methodology 35
3.2.1 Deep Convolutional Neural Network 35
3.2.2 Meta-learning 36
3.2.3 Framework 39

3.3 Experiments . . . 43
3.3.1 Experimental Setup . . . 44
3.3.2 Experimental Results . . . 45

3.4 Conclusions and limitations . . . 47

Chapter 4 Improving the Interpretability of Deep Neural Networks with Knowledge Distillation . . . 50

4.1 Introduction . . . 50

4.2 Related Work . . . 52

4.3 Methodology . . . 54
4.3.1 CART for Regression . . . 54
4.3.2 Matching Logits . . . 56
4.3.3 Distilling CNN into Decision Trees . . . 57

4.4 Experiments . . . 59
4.4.1 Datasets . . . 59
4.4.2 Experimental Setup . . . 59
4.4.3 Experimental Results . . . 61
4.4.4 Discussion . . . 62

4.5 Conclusions and limitations . . . 64

Chapter 5 QDV: Refining Deep Neural Networks with Quantified Interpretability . . . 65

5.1 Introduction . . . 65

5.2 Method . . . 67
5.2.1 QDV . . . 67
5.2.2 Comparison with Knowledge Distillation . . . 73

5.3 Experiments . . . 73

5.4 Conclusions . . . 77

Chapter 6 Conclusions and Future Work . . . 79

6.1 Conclusions . . . 79

6.2 Limitations . . . 79

6.3 Future Work . . . 81
6.3.1 Future Work for CNN-INTE . . . 81
6.3.2 Future Work for Interpretation with Knowledge Distillation . . . 81

6.3.3 Future Work for Refining Neural Nets 82
6.3.4 Future Work for Further Research 82

Bibliography **86**

List of Tables

3.1	Instances Selected From the Meta-level Test Data	46
4.1	Datasets	59
4.2	Parameter settings for MNIST	60
4.3	Parameter settings for Connect-4	61
4.4	Test Accuracy Results for MNIST	61
4.5	Test Accuracy Results for Connect-4	62
5.1	CIFAR-10 Dataset	74
5.2	Parameter settings for fine tuning	74
5.3	Alexnet's Structure before fully connected layers	75
5.4	VGG19's Structure before fully connected layers	75
5.5	AR values at each hidden layers of newly trained models M_n: higher values implies better representations learned	77

List of Figures

2.1 The Taxonomy of Interpretation Methods 10

2.2 A toy example from [133] illustrating the intuitions behind LIME. 15

2.3 A surrogate model from [31] applying Gradient Boosting Trees (GBTs) as the surrogate. 17

2.4 The Deconvnet architecture from [187]. 21

2.5 Feature visualizations and image patches of the top 9 stimuli that excite a specific feature map at layer 3 and 5 for a trained neural network from [187]. 22

2.6 Comparison of visualizations for the 'tiger cat' class among Guided Backprop, Grad-CAM and Guided Grad-CAM from [144]. 25

2.7 Visualization of the MNIST data set applying t-SNE from [108]. 28

3.1 Structure of our deep CNN model generated by TensorFlow's TensorBoard. 37

3.2 Meta-learning training process. 38

3.3 Meta-learning test process. 39

3.4 CNN-INTE training process. 40

3.5 Toy example for the generation of Meta-level training data. . . 42

3.6 CNN-INTE test process. 43

3.7 Example of a correctly classified test instance: True1. Each point represents an activation belong to either the true label (in red) or the hypothesis (in blue) 47

3.8 Example of a correctly classified test instance: True2. 48

3.9 Example of a wrongly classified test instance: Wrong1. 49

4.1 Examples of hard and soft targets. 51

4.2 Framework of our method. 58

4.3 Distillation results for MNIST. 62

4.4 Distillation results for Connect-4. 63

5.1 Applying Deconvnet on the 1st layer features for Alexnet (b) and modified Alexnet (c) [187]. However, we don't observe much differences between (b) and (c) visually. 67

5.2 Architecture of QDV when applying "Warm Restart" on the pre-trained Alexnet. 70

5.3 QDV visualization results and AR values for pre-trained Alexnet and VGG19. Higher AR implies better representation. 78

6.1 A common framework for debugging machine learning models. 83

Abstract

Model interpretability is a requirement in many applications in which crucial decisions are made by users relying on a model's outputs. The recent movement for "algorithmic fairness" also stipulates explainability, and therefore interpretability of learning models. The most notable is "a right to explanation" enforced in the widely-discussed provision of the European Union General Data Privacy Regulation (GDPR), which became enforceable beginning 25 May 2018. And yet the most successful contemporary Machine Learning approaches, the Deep Neural Networks, produce models that are highly non-interpretable. Deep Neural Networks have achieved huge success in a wide spectrum of applications from language modeling and computer vision to speech recognition. However, nowadays, good performance alone is not sufficient to satisfy the needs of practical deployment where interpretability is demanded for cases involving ethics and mission critical applications. The complex models of Deep Neural Networks make it hard to understand and reason the predictions, which hinders its further progress.

In this thesis, we attempt to address this challenge by presenting two methodologies that demonstrate superior interpretability results on experimental data and one method for leveraging interpretability to refine neural nets.

The first methodology, named CNN-INTE, interprets deep Convolutional Neural Networks (CNN) via meta-learning. In this work, we interpret a specific hidden layer of the deep CNN model on the MNIST image dataset. We use a clustering algorithm in a two-level structure to find the meta-level training data and Random Forests as base learning algorithms to generate the meta-level test data. The interpretation results are displayed visually via diagrams, which clearly indicate how a specific test instance is classified.

In the second methodology, we apply the Knowledge Distillation technique to distill Deep Neural Networks into decision trees in order to attain good performance and interpretability simultaneously. The experiments demonstrate that the student model achieves a significantly higher accuracy performance (about 1% to 5%) than conventional decision trees at the same level of tree depth.

In the end, we propose a new method, Quantified Data Visualization (QDV), to leverage interpretability for refining deep neural nets. Our experiments show empirically why VGG19 has better classification accuracy than Alexnet on the CIFAR-10 dataset through quantitative and qualitative analyses on each of their hidden layers. This approach could be applied to refine the architectures of deep neural nets when their parameters are altered and adjusted.

List of Abbreviations Used

aLIME Anchor Local Interpretable Model-Agnostic Explanations.

AM Activation Maximization.

AR Agreement Ratio.

BETA Black Box Explanations through Transparent Approximations.

CART Classification and Regression Trees.

CIFAR Canadian Institute for Advanced Research.

CNN Convolutional Neural Networks.

Deconvnet Deconvolutional Network.

DGN-AM deep Generator Network-based Activation Maximization.

DNNS Deep Neural Networks.

EHR electronic health records.

ERM empirical error minimization.

EU European Union.

FIRM feature importance ranking measure.

GAN Generative Adversarial Network.

GBTs Gradient Boosting Trees.

GDPR General Data Protection Regulation.

Grad-CAM Gradient-weighted Class Activation Mapping.

IB Information Bottleneck.

LIME Local Interpretable Model-agnostic Explanations.

LRP Layer-wise relevance propagation.

MDS Multidimensional Scaling.

MFI Measure of Feature Importance.

MFV multifaceted feature visualization.

MIO mixed-integer optimization.

MLP multilayer perceptron.

PALM Partition Aware Local Model.

PCA Principal Component Analysis.

PIPEDA Personal Informational Protection and Electronic Documents Act.

POIMs positional oligomer importance matrices.

QDV Quantified Data Visualization.

QII Quantitative Input Influence.

ReLU rectified linear unit.

SA sensitivity analysis.

SGD Stochastic Gradient Descent.

SHAP SHapley Additive exPlanations.

SM Saliency Maps.

SP-LIME Submodular Pick LIME.

t-SNE t-distributed Stochastic Neighbor Embedding.

TCNN Tiled convolutional neural networks.

xAI explainable AI.

Acknowledgements

It is my pleasure to take this opportunity to thank all the people who have helped me during my Ph.D study.

First and foremost, I wish to express my sincere appreciation to my supervisor, Professor Stan Matwin. Dr Matwin has been supportive both academically and emotionally on the rough road of completing my thesis. He helped me come up with the thesis topic and provided a research assistantship during my last phase of Ph.D study. Dr Matwin is an incredible supervisor who constantly holds a positive attitude toward my research outcomes. His immerse knowledge guided me through the whole journey. He would say yes to my ideas and entrust me for implementing them, which boosted my confidence enormously. I am very grateful for his invaluable advices and timely feedbacks provided on both paper and thesis writing. He also gave me necessary freedom to move forward. Dr Matwin is very generous allowing me attending and presenting at international conferences, which I have benefited a lot from. He also encouraged me to send my paper to top tier conferences.

Besides my supervisor, I would like to thank the rest of my thesis committee: Prof. Sageev Oore, Prof. Fernando Paulovich and Prof. Thomas Trappenberg. Their stimulating and inspiring views at my thesis proposal defence widened my research horizon from various perspectives. I would not have reached the goals in chapter 5 of my thesis without their insightful comments and encouragement. I would also like to extend thanks to professor Yuhong Guo at Carleton University for being my external examiner.

I owe a sincere thank you to my friend, Xiang Jiang, Ph.D candidate at Dalhousie University, for stimulating discussions during difficult times of my research. I would like to thank my fellow labmates for all the fun we had during my Ph.D study. I also want to thank Mr. David L. Langstroth for proofreading my thesis.

It is my honor to acknowledge the grants and awards I received for my Ph.D study: the Alexander Graham Bell Canada Graduate Scholarships provided by the Natural Sciences and Engineering Research Council of Canada (NSERC); the President's

Awards; the Nova Scotia Graduate Scholarship. This dissertation would not have been possible without these agencies' generous funding.

Last but not the least, I owe a huge debt of gratitude to my family for their immense support, love, and patience.

Chapter 1

Introduction

A complete task of knowledge acquisition from data comprises three stages: data pre-processing (to process the input data that are noisy, inconsistent, or incomplete), training of the machine learning algorithm and data post-processing (integration, filtering, evaluation, and explanation of learned knowledge). In this thesis, we focus on data post-processing, more specifically, the explanation of the learned knowledge.

With the fast development of sophisticated machine learning algorithms, artificial intelligence has been gradually penetrating a number of brand new fields with unprecedented speed. One of the outstanding problems hampering further progress is the *interpretability* challenge. This challenge arises when the models built by the machine learning algorithms are to be used by humans in their decision making, particularly when such decisions are subject to legal consequences and/or administrative audits. For human decision makers operating in those circumstances, to accept the professional and legal responsibility ensuing from decisions assisted by machine learning, it is critical to comprehend the models. This is generally true for areas like criminal justice, health care, terrorism detection, education systems and financial markets.

For areas like the healthcare domain, business, crime prediction, etc., mistakes in these areas can be catastrophic. For instance, to develop safe self-driving cars we need to understand their rare but costly mistakes. Therefore, it is imperative to explain the learned representations, relationships between the inputs and the dependent variables and decisions made by these models. To trust the model, decision makers need to first understand the model's behavior, and then evaluate and refine the model using their domain knowledge. Even for areas like book or movie recommendations [66] and automated aids [44], explanations for a recommendation and an error made could increase the trust and reliance on these systems.

One critical issue associated with future automated systems based on machine learning is its misalignment with the objectives of its stakeholders. That is, whether

these systems really behave reliably in unforeseen situations. They may perform pretty well on test cases, but might do the wrong thing in deployment in the wild. Ironically, it could also be revealed later that they were doing the right things for the wrong reasons. Hence, interpretability plays a significant role in assisting us to reduce errors.

At present, in the US, the practical application of interpretable models resides in "the Right to explanation" in certain circumstances such as credit card approvals, health insurance rates and so on. In comparison, Europe has a more advanced regulation: the European Union (EU) General Data Protection Regulation (GDPR) (enacted 2016, taking effect 2018): ".....In any case, such processing should be subject to suitable safeguards, which should include specific information to the data subject and the right to obtain human intervention, to express his or her point of view, to obtain an explanation of the decision reached after such assessment and to challenge the decision.....". Canada has long been at the forefront of data protection. As early at 2000, Canada enacted its privacy law: Personal Informational Protection and Electronic Documents Act (PIPEDA) [34]. It is based on the 10 principles [34] which could also be found in the EU GDPR today. Six months before the final text of the GDPR, the Data Privacy Act (an amendment to PIPEDA) was adopted on June 18, 2015. To comply with GDPR, canada amended PIPEDA again and the new requirements came into effect in November 1, 2018 [125]. Moreover, the Government appointed Canadian Institute for Advanced Research (CIFAR) (a charitable organization based in Canada) to develop and lead a $125 million Pan-Canadian Artificial Intelligence Strategy, the world's first national AI strategy. In 2017, CIFAR launched a key pillar of the CIFAR Pan-Canadian AI Strategy: the AI & Society program, in which interpretability would play a critical role. Hence, there is an urgent need for the technology to enable explanations of the internal logic of black box models where automated decision making happens.

Some advanced machine learning algorithms exhibit extraordinary predictive abilities. However, it is now recognized as a common problem that they fall short on trust/transparency/interpretability/explainability in the models and the decisions they generate. Deep Neural Networks (DNNS) in particular could even surpass human predictions in areas like speech processing, machine translation, image classification, etc. [106] [161] [4]. However, their inner workings still remain as black-boxes for

the practitioners, which hinders their application to mission-critical tasks that have profound consequences and human-machine hybrid networks.

1.1 The Necessities for Interpretation

it is not surprising that the tremendous amount of data produced daily inevitably contains human biases/discriminations. And subsequently the predictions produced by a machine learning model based on these data also inherit these prejudices, which could result in wrong decisions. Except for ethical issues, interpretability is also required for the following reasons.

- **Accountability & Transparency**

  Automated decision making is now more and more widely applied in areas such as vote counting, immigration visa applications, loan and credit card approval, etc. However, based on past experiences, sometimes the automatically generated results could be incorrect, unjustified, or unfair. And when things go wrong, there are usually not enough resources available to reason about. Therefore, there is a great interest from society as a whole to make these automatic systems accountable, governable and compliant with key standards of legal fairness [89].

- **Safety**

  The biggest concern about automated systems is their unreasonable risk to safety. Recent serious accidents on Self-Driving Cars [179] [178] stimulate the requirements for regulating autonomous systems. However, the ambiguous contexts in which autonomous systems execute pose a huge challenge to traditional regulatory schemes which use performance standards [32] for stable contexts. The noisy, uncertain and rapidly changing context makes it even harder to reason the decisions of autonomous systems: for example, how to avoid striking a pedestrian or vehicle when an autonomous vehicle tries to get around an obstacle on the road. Knowledge about the inner workings of the algorithms behind the decision would help the understanding of the interactions between the autonomous system and the context.

- **Legal Liability**

Let's reconsider the accidents of Self-Driving Cars. One would wonder which laws to apply and to whom (the user or the programmer or the vendor?) they might apply in such cases. Should we apply criminal liability or product design legislation or the tort of negligence [171]? One of the factors that would help to answer these questions is whether we are fully aware of the limitations of the AI systems. Besides some well known limitations: the lack of general knowledge, the insufficiency of programmed knowledge to cover all possibilities and the fast variations of background knowledge, the key limitation is the interpretability of the AI systems.

1.2 Properties of Interpretability

As a matter of fact, interpretability is very difficult to define and no consensus has yet been reached [99]. In the machine learning community, recently, interpretability is defined in [42] as "the ability to explain or to present in understandable terms to a human". Besides definition, a much harder task is to quantify and measure interpretability. Hence, in [128], the effort is extended beyond typical machine learning research into human-computer interaction. There are also other studies on aspects of interpretability such as the plausibility of models: the likeliness that a user accepts it as an explanation for a prediction [56]. In this section, we first discuss some popular evaluation methods and then, based on these evaluations, we give a formal definition of interpretability.

1.2.1 Evaluations of Interpretation Methods

For evaluation metrics, there are no mutually agreed standards. Sometimes the evaluation methods are only applicable to a specific model. For example, a set of evaluation metrics for rule-based models is proposed in [92]. Besides experimental simulations, a few studies [133] [128] [92] also use human judgments for evaluations. Here we present some general evaluation metrics: fidelity, comprehensibility and accuracy, which are frequently used by some state-of-art works [162] [163] [133] [107]. These could serve as a general guide to the design of interpretation methods. It should be noted that these metrics are limited to the Post-hoc Interpretations that we are

about to introduce in chapter 2.

- **Fidelity**

  It is not realistic for the interpretation model to be entirely faithful to the black-box model. Fidelity demands that the interpretation model's prediction should match that of the black-box model as closely as possible. In other words, the interpretation model tries to mimic the behavior of the model itself on the instance being predicted.

- **Comprehensibility**

  Comprehensibility requires that the interpretation results are understandable to the users. When building an interpretation method, we should take into consideration the limitation of human cognition. For instance, decision trees involving thousands of nodes and decision rules having hundreds of levels of if-then conditions are not interpretable in this sense, although they are commonly regarded as inherently interpretable algorithms for textual representations.

- **Accuracy**

  Accuracy measures the performance of the interpretation model on the original training data used to train the black-box model to check if the interpretation model could outperform the black-box model. The measurements could be traditional evaluation metrics in machine learning such as accuracy score, AUC score, F1-score, etc.

1.2.2 Definition of Interpretability

Apparently, a good interpretation model needs to at least satisfy the three aforementioned general evaluation metrics. Here we provide a mathematical objective function to define interpretability, inspired by [92] [133].

We treat different interpretation methods as different interpretation models and each of them is represented as g

$$g \in G \tag{1.1}$$

G is the family of all the interpretation models that could be used to interpret a black-box model. We denote the black-box model to be explained as f and use $\mathcal{L}(f, g)$

to represent the unfaithfulness (to include fidelity into the objective function) of g to f. What is opposed to Comprehensibility is the complexity of g, which is represented as $\Omega(g)$. The final factor to consider is accuracy of g on original training data of X, which is specified as $e(g(X))$. Hence, we formalize our objective function for finding the best interpretation as

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \ \mathcal{L}(f, g) + \Omega(g) + e(g(X)) \tag{1.2}$$

$\xi(x)$ is the final interpretation results we obtained when simultaneously minimizing the unfaithfulness of g to f (ensuring fidelity), reducing the complexity of g (ensuring comprehensibility) and lowering the prediction error of g on X (ensuring accuracy). As stated in [92] [133], there is a trade off between fidelity+accuracy and comprehensibility. The objective is to find an optimal interpretation model that could find a sweet spot of the trade off. It should be noted that this optimal minimum cannot always be found, but at a practical level, one could simply try several values for the minimized function and choose the smallest. At the theoretical level one could use Pareto optimality [27] instead of strict optimality.

1.3 Contributions

In this thesis, we first summarize and provide our own taxonomy on the current interpretation methods and then present our two methods designed to improve interpretability and one method for employing interpretability to refine neural nets. More specifically, chapter 3 provides a method to understand the behavior of a neural network model; chapter 4 presents an approach to resolving the tension between interpretability and accuracy ; chapter 5 demonstrates a way to refine the structure of a neural network model.

The first of our two interpretation methods is a visualization technique that interprets deep Convolutional Neural Networks (CNN) via meta-learning, which we name CNN-INTE . The main contributions of this method are

- Compared to LIME [133] which provides *local* interpretations for the entire model in specific regions of the feature space, our method provides *global* interpretation for any test instances on the hidden layers in the whole feature space.

- Compared to models which apply inherently interpretable algorithms, e.g. [92], our method has the advantage of not compromising the accuracy of the model to be interpreted. This produces more reliable interpretation.

- In contrary to [133] and[93] which treat the model to be interpreted as black box, we access the hidden layers of deep CNN models and interpret them.

Our second interpretation method applies the Knowledge Distillation technique to distill Deep Neural Networks into decision trees in order to attain good performance and interpretability simultaneously. The contributions for this second method are:

- Instead of distilling complex and deep neural nets into simple and shallow neural nets as done in [138] [172] [191], we employ knowledge distillation for another purpose: interpretation.

- We resolve the tension between interpretability and accuracy performance by distilling deep neural nets into conventional decision trees. To the best of our knowledge, we are the first to distill Deep Neural Networks into decision trees on multi-class datasets.

- The main obstacle to execute this plan is that for pure classification tasks there exist no logits in decision trees as in neural nets which could be used in the loss function. We address this issue by reformulating it into a multi-output regression problem [17] and achieve significant accuracy improvements (about 1% to 5%) on the experiments.

- The success of our approach opens a window for turning those inherently interpretable algorithms (which are highly interpretable, but worse in accuracy performance) into models attaining both accuracy and interpretability simultaneously.

The previous two methods attempt to solve the issues from the perspective of the end users so that they can understand the models better. In order for the model designers to benefit from interpretability, our method for refining neural nets leverages interpretability for refining the structure of deep neural nets. The contributions for this method are:

- The main purpose of this method is to refine neural network models more convincingly via interpretability, which was seldom studied before.
- Besides the definition of interpretability, a much harder task is to quantify and measure it. Within this method, we propose a way to quantity interpretation.
- To the best of our knowledge, we are the first to interpret the neural net models trained on the CIFAR-10 dataset when applying Alexnet and VGG19 as "Warm Restart".

The three methodologies proposed in this thesis are peer-reviewed and evaluated in the field of machine learning interpretability, as the results presented here are published. Our published and submitted papers related to this thesis are as follows:

- Xuan Liu, Xiaoguang Wang, and Stan Matwin. "QDV: Refining Deep Neural Networks with Quantified Interpretability." In 2020 European Conference on Artificial Intelligence (ECAI), submitted.
- Liu, Xuan, Xiaoguang Wang, and Stan Matwin. "Improving the Interpretability of Deep Neural Networks with Knowledge Distillation." In 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 905-912. IEEE, 2018.
- Liu, Xuan, Xiaoguang Wang, and Stan Matwin. "Interpretable deep convolutional neural networks via meta-learning." In 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-9. IEEE, 2018.

The publications during my phd study are listed below:

- Jiang, Xiang, Xuan Liu, Erico N. de Souza, Baifan Hu, Daniel L. Silver, and Stan Matwin. "Improving point-based AIS trajectory classification with partition-wise gated recurrent units." In Neural Networks (IJCNN), 2017 International Joint Conference on, pp. 4044-4051. IEEE, 2017.
- Jiang, Xiang, Erico N. de Souza, Xuan Liu, Behrouz Haji Soleimani, Xiaoguang Wang, Daniel L. Silver, and Stan Matwin. "Partition-wise Recurrent Neural Networks for Point-based AIS Trajectory Classification." ESANN 2017 proceedings, European Symposium on Artificial Neural Networks.

- Liu, Xuan, Xiaoguang Wang, Stan Matwin, and Nathalie Japkowicz. "Meta-MapReduce for scalable data mining." Journal of Big Data 2, no. 1 (2015): 14.

- Wang, Xiaoguang, Xuan Liu, Bo Liu, Erico N. de Souza, and Stan Matwin. "Vessel route anomaly detection with Hadoop MapReduce." In Big Data (Big Data), 2014 IEEE International Conference on, pp. 25-30. IEEE, 2014.

- Wang, Xiaoguang, Xuan Liu, and Stan Matwin. "A distributed instance-weighted SVM algorithm on large-scale imbalanced datasets." In Big Data (Big Data), 2014 IEEE International Conference on, pp. 45-51. IEEE, 2014.

- Wang, Xiaoguang, Xuan Liu, Stan Matwin, and Nathalie Japkowicz. "Applying instance-weighted support vector machines to class imbalanced datasets." In Big Data (Big Data), 2014 IEEE International Conference on, pp. 112-118. IEEE, 2014.

- Wang, Xiaoguang, Xuan Liu, Stan Matwin, Nathalie Japkowicz, and Hongyu Guo. "A multi-view two-level classification method for generalized multi-instance problems." In Big Data (Big Data), 2014 IEEE International Conference on, pp. 104-111. IEEE, 2014.

We notice that there has been an explosion of interest in designing high-accuracy models that are easy to interpret. These approaches are either model agnostic or model specific. In the following chapter, we review some powerful mainstream approaches that make great endeavors to reason deep neural networks. Although interpretability is an interest for the whole machine learning community and a variety of approaches are also proposed to other algorithms such as SVM, ensemble methods and so on, here we focus on deep neural networks.

Chapter 2

Taxonomy of Interpretation Methods

Until now, numerous Interpretation methods for neural networks have been proposed or are underway. Faced with the overwhelming research published in this area, it is quite beneficial that we could organize these methods to get a general idea of what has been happening and hopefully to figure out some valuable ideas and directions for future work. In this chapter, we provide a taxonomy of these interpretation methods which is shown in Fig. 2.1. This taxonomy includes three main parts: Post-hoc Interpretations, Inherently Interpretable Models and other methods that don't belong to the first two categories. In the following sections, we will discuss each of them starting from the simplest to the most complex and challenging and review their progresses. For a broader inclusion, please refer to the Explainability Fact Sheets [152], recently published in 2020.

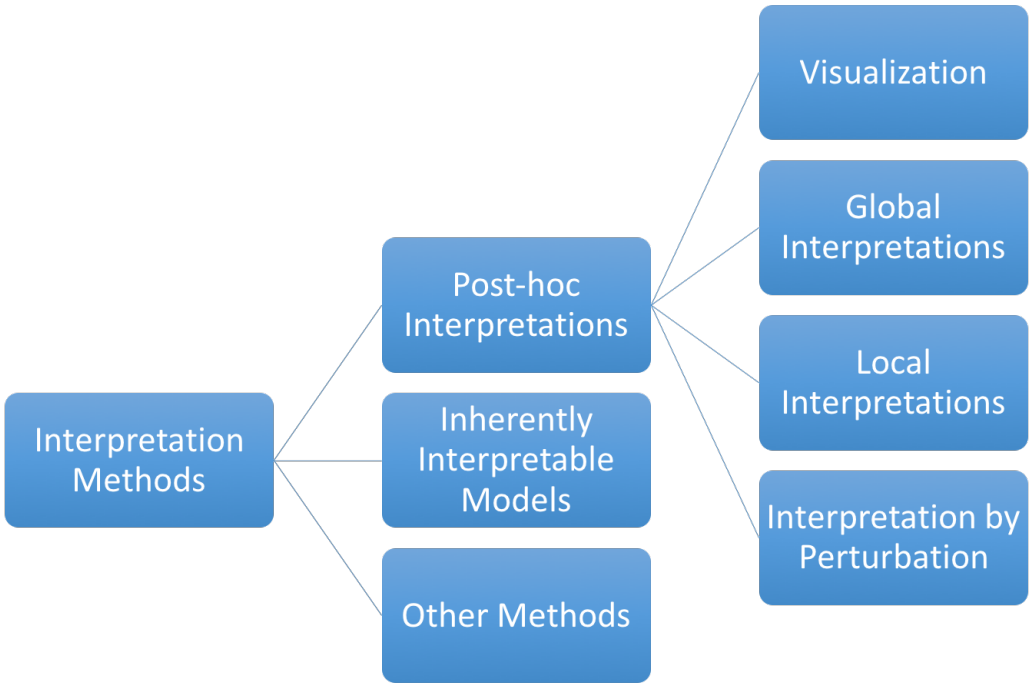


Figure 2.1: The Taxonomy of Interpretation Methods

2.1 Post-hoc Interpretations

Post-hoc Interpretations [63] interpret black box models after they are trained, hence, the name "Post-hoc". Usually, this requires the building of a separate explanation model or technique to explain the predicted decisions or the model itself. A large number of interpretation methods belongs to this class and here we roughly divide them into four subcategories: Interpretation by Perturbation, Local Interpretations, Global Interpretations and Visualization, and we review each of them in detail in the following sections.

2.1.1 Interpretation by Perturbation

Interpretation by Perturbation stands for the methods that try to inspect the black box model's properties or to find evidence responsible for a classification decision by perturbing the inputs. These methods are somewhere between the global interpretations and local interpretations. In this section, we introduce one such perturbation technique: Sensitivity Analysis.

Sensitivity Analysis

In machine learning, sensitivity analysis (SA) [142] refers to the technique that tests whether the model and predictions remain stable or they change but in a predictable way when the inputs undergo subtle and intentional changes. If the answer is yes, then the model's trust is increased. The reality is that minor perturbations of the inputs could sometimes result in drastically different predictions, e.g. the adversarial examples for neural networks [160] and the instability for decision tree learning [43]. Some of the visualization methods we would introduce in section 2.1.4 could also be treated as sensitivity analysis methods when the inputs are intentionally varied to evaluate the stability of the neural network model. Moreover, LIME [133] could also be regarded as a sensitivity analysis method.

Interpretability has been observed as an important issue long before the current onset of interest in this topic. In ecological sciences, a number of methods are adopted in [124] [35] to understand the mechanisms of neural networks. One of them is the sensitivity analysis. Figuring out the contributions of input variables on the modeling

process is critical to find the causal relationships driving ecological phenomena. They find the influence of the independent variables on the predictions of a model by interpreting the neural network connection weights because the former depends on the latter's magnitude and direction. Using sensitivity analysis, they quantitatively interpret the connection weights by constructing contribution plots for the variables in the neural network. They vary one variable at one time across the entire range while holding the other variables constant.

Based on past work with one-dimensional (1-D) SA [78] and two-dimensional (2-D) SA [45], a global framework for interpreting machine learning models using SA: GSA (global sensitivity analysis) is proposed in [35] which extends the previous work to classification tasks, discrete variables, distinct scanning functions and up to n dimensions where n is the dimension of the inputs. SA is able to rank the input features by measuring the change of the outputs when the inputs are perturbed and it is originally used for feature selection. In this paper [35], they apply it to open the black box. They also propose several visualization techniques such as Input Importances, Variable Effect Characteristic Curve and Variable Effect Characteristic Surface and Contour Plot to display the sensitivity analysis results. More recent work [36] is done by the same authors, in which three novel SA methods are proposed: Data-based SA (DSA), Monte-Carlo SA (MSA) and Cluster-based SA (CSA). They also introduce a new sensitivity measure approach: Average Absolute Deviation (AAD) from the median and novel functions for aggregating multiple sensitivity responses.

A family of Quantitative Input Influence (QII) measures are introduced in [40] to address the issue of algorithmic transparency, which measures the influence of inputs on the outputs of a machine learning system and aids the design of transparency reports in forensic cases. The influence is measured on a quantity of interest: one property of the model when given some input distributions. The Quantitative Input Influence is defined by calculating the difference of the quantity of interest when the original feature is changed via an intervention. In this paper [40], they replace the value of every input with a random independently chosen value. This is a causal model that takes into account the correlations between the inputs. Specifically, besides an individual input's influence, they also quantify the joint influence of a set of inputs and the marginal influence of individual inputs within such a set. Moreover, they

apply cooperative game theory to calculate the average marginal influence of the input. They also use sensitivity analysis to conquer the problem of the transparency-privacy tradeoff, which is accomplished by calculating the sensitivities of the QII measure to determine the amount of noise needed to make the influence measure differentially private.

A region perturbation method is introduced in [143] to quantitatively evaluate the visualization techniques (introduced in section 2.1.4) such as deconvolution (a back propagation method) [187], saliency maps (a sensitivity analysis approach) [149] and Layer-wise relevance propagation (LRP) [13] which present the interpretations as heatmaps. In this method they apply a greedy iterative procedure to perturb the pixels that are relevant for a classification decision. They measure how the class encoded in the image disappears when they progressively remove information from the image at a specific location in the image. Moreover, a "deep Taylor decomposition" method is proposed in [113] based on the layer-wise relevance propagation method, which decomposes the predictions of neural networks into contributions of the input elements. They also prove that Sensitivity Analysis can be viewed as a special instance of Taylor decomposition.

2.1.2 Local Interpretations

Local Interpretations stand for the type of methodologies that explain the individual predictions to understand the model at the level of smaller regions. Usually the smaller sections of a nonlinear model tend to be linear and monotonic. Therefore, it is expected that local interpretations could be more accurate than global interpretations. However, although these methods are locally faithful, they may not align with the global behavior of the entire black box model. Here we present a series of works that fit in this category. It should be noted that "Neural Feature Visualization" and "Attribution" introduced in section 2.1.4 also belong to the local interpretation techniques.

Most local interpretation methods explain black box models via linearly weighted combinations of the input features, i.e. contributions of feature values.

A "ExplainD" framework is proposed in [127] to provide graphical explanations on decision, decision evidence, decision speculation, ranks of evidence, and source

of evidence. This is fulfilled by reformulating the machine learning algorithms such as naïve Bayes, linear support vector machine and logistic regression into additive models [51] [2]. They claim that this framework is extensible to any classifier.

A local explanation framework applicable to any classifier is introduced in [8]. They use local explanation vectors to understand the predictions generated by single data instances. The vectors are calculated with class probability gradients. The found features relevant to the predictions are indicative of local peculiarities which could otherwise be neglected in the global view. However the reliance on the coefficients of gradients could sometimes be difficult because for confident predictions the gradient is near zero.

Based on the few applications of Shapley value [182] in the machine learning community, the authors in [85] link the individual feature values' contribution to the Shapley value in the coalitional game theory [77] [33] [115]. This is also proven by rigorous theoretical analysis by finding the prediction difference between the the cases where the feature values are given versus those in which all feature values are "ignored". They present a general explanation method which works for any type of classifier. They also provide an efficient sampling-based approximation method to tackle the exponential time complexity of going through all subsets of the features, which doesn't require retraining the classifier.

To resolve the problems for "trusting a prediction" and "trusting a model", two methods are proposed in [133] to explain individual predictions and understand a model's behavior respectively: Local Interpretable Model-agnostic Explanations (LIME) and Submodular Pick LIME (SP-LIME). The main idea for LIME is to use inherently interpretable models g to interpret complex models f locally. They designed an objective function to minimize the unfaithfulness (when g is approximating f in a local area) and the complexity of g. Although it is stated in their paper that the objective function g could be any interpretable model, they set g as sparse linear models in [133]. Based on the individual explanations generated by LIME, they design a submodular pick algorithm: SP-LIME to explain the model as a whole by picking a number of representative and non-redundant instances. A toy example for LIME is shown in Fig. 2.2. In this figure, the pink and blue background represents the decision boundary learned by the black box model. The dashed line is a local linear

model used to approximate the decision boundary of the black box model around the neighbourhood of the instance represented by the bold red cross.

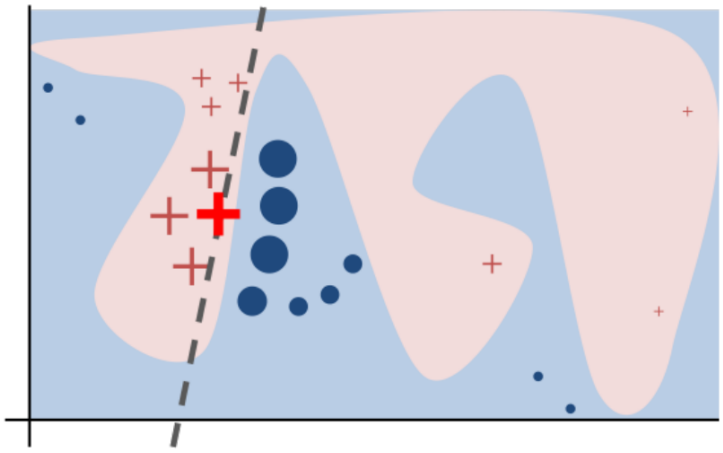


Figure 2.2: A toy example from [133] illustrating the intuitions behind LIME.

It was suggested in [132] that coverage, precision and effort should be used to evaluate the results of the model interpretation. Although LIME achieves high precision and low effort, the coverage is not clear. In other words, LIME is able to explain why a specific prediction is made using the weights of the local model g, but can't indicate in what local region the explanation is faithful. To solve this problem, the Anchor Local Interpretable Model-Agnostic Explanations (aLIME) method was introduced in [132]. In aLIME, the if-then rules are used instead of using the weights in a linear model to explain a specific prediction (as was executed in LIME). The idea is based on the Decision Sets algorithm [92]. These if-then rules are easy to comprehend and have good coverage. A further extension on anchors is proposed in [134].

Without sacrificing the performance of black box models, a model explanation system is presented in [169] which aims to provide explanations for a single prediction to augment black box predictions with explanations. This is applicable to fraud detection in scenarios such as computer vision, credit risk, spam detection, etc. They construct a scoring function to find a good explanation system by integrating four desired properties: Eligibility, Generality, Accuracy and Validity. Then they use a Monte Carlo algorithm to approximate the optimization with black box models.

2.1.3 Global Interpretations

The set of techniques that uncover the overall logic behind black box models belong to the category of Global Interpretations. They also provide global explanations of the prediction results and the learned relationships between the inputs and the dependent variables.

Surrogate Models

Applying surrogate models is the strategy of solving the interpretability problem of machine learning by using more machine learning! Surrogate models employ those inherently interpretable models (discussed in section 2.2) to approximate the learned functions of black box models. These surrogate models are usually trained on the inputs and predictions of the complex black box models. Hence, the internal mechanisms of the black box models could be easily interpreted through analyzing the surrogate models' coefficients, variable importance, trends, and interactions.

The research on surrogate models dates back to the very early ages of neural networks when an algorithm: TREPAN [38] was proposed in 1996! It uses a symbolic representation: decision tree to approximate the concepts learned by a neural network with one hidden layer. Different from conventional decision trees, the surrogate model is trained on synthetic data generated by querying the trained neural network. It is noticeable that fidelity is also taken into account in the training process. The use of synthetic data solves the problem of lacking training data to split further down the depth of the tree for typical tree induction algorithms. Based on TREPAN, a variety of works [86] [18] [73] on tree surrogates are followed. In the most recent years, Knowledge Distillation: a concept proposed by Hinton et al. [67] [54] also shares some similarities with TREPAN. They all attempt to transfer the knowledge from a cumbersome model to a small model. The difference is that TREPAN uses the cumbersome model as an oracle to query the labels of an instance and also to select splits for each of the tree's internal nodes to improve the fidelity. Knowledge distillation utilizes the soft targets (probabilities for all classes) instead of the hard targets to train a smaller model.

Earlier research on using rules as surrogate models are in [3] [74] [192] [6]. A more recent one is Black Box Explanations through Transparent Approximations

(BETA) introduced in [93]. BETA is a framework which attempts to produce global interpretation for any classifiers which are treated as black box classifiers. Based on their previous work on Decision Sets [92], the authors designed a framework with two level decision sets to take into account fidelity (faithfulness to the black box model), unambiguity (single and deterministic explanations for each instance), interpretability (complexity minimized) and interactivity (user specified explorations of the feature's subspace). In this two level structure, the outer if-then rules are the "*neighborhood descriptors*" and the inner if-then rules are "*decision logic rules*" (how the black box model labels an instance under the outer if-then rules). Similar to [92], an objective function is built and near-optimal solutions are found. However, rules may not be suitable for the applications with text or image data [134].

In the health care domain, two pipelines [30] [31] are proposed to interpret DNN applying Gradient Boosting Trees (GBTs) [52] [53] as surrogates. One of them extracts the logits from a learned DNN and uses the logits and the true labels of the original training data to train a logistic regression algorithm to obtain the soft prediction scores. Then they train GBTs with the original training data's features and the soft predictions. The second pipeline directly applies the soft prediction scores of the trained DNN on the original training data as targets for training a mimic model with GBTs. This second pipeline is shown in Fig. 2.3.

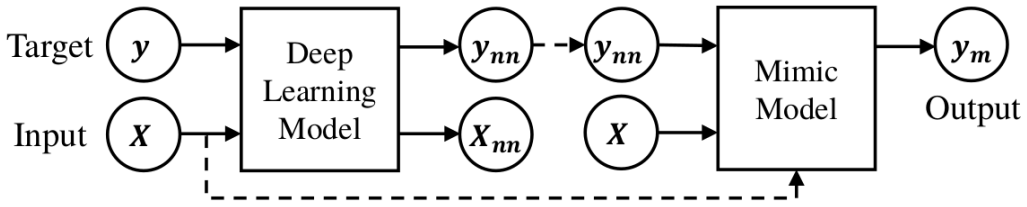


Figure 2.3: A surrogate model from [31] applying Gradient Boosting Trees (GBTs) as the surrogate.

Another approach that explains neural networks with GBTs as surrogate models is in [162]. They tried two student models: tree-based generalized additive models (GA2Ms) [24] [104] [105] and GBTs. The black box model they interpret is multilayer perceptrons. For the student model's training process, they applied the method of matching logits instead of soft targets in [30] [31]. They investigated both classification and regression problems. Then a following work [163] used the Global Additive

Explanation model as a surrogate model to explain fully connected neural networks. The interpretation tool they adopted is feature shapes. They also validated their method on real world datasets. It is remarkable that they compare their results with several contemporary interpretation techniques quantitatively.

A two-part surrogate model: Partition Aware Local Model (PALM) is proposed in [87]. The meta-model is a decision tree which partitions the training data while the sub-models could be arbitrarily complex models which learn within each partition. The sub-models are built applying a variation of the Expectation-Maximization algorithm with gradient descent.

Other Models

For neural network visualizations, in contrast to the Activation Maximization (AM) method (which we would introduce in section 2.1.4), a global, network level inspection method is proposed in [160]. They argue that individual units don't contain semantic information. A stronger support for this argument is available in the area of word representations [111]. They [160] further prove with experiments that the random basis (linear combinations of high level units) could generate results that are just as semantically meaningful as a natural basis (individual high level units). They claim that it is questionable to say that neural networks disentangle variation factors across coordinates. They essentially compare the results from these two equations

$$x^* = \underset{x \in I}{\operatorname{argmax}} \langle \phi(x), n_i \rangle \tag{2.1}$$

$$x^* = \underset{x \in I}{\operatorname{argmax}} \langle \phi(x), r \rangle \tag{2.2}$$

where I is the held-out data, $\phi(x)$ is the activation values of some layer, n_i is the natural basis vector for the i-th hidden unit and r is random direction. They examine the visual results from these two equations and find that they are semantically equal. It is also shown that without regularization, the approach would end up by generating adversarial examples.

It is noticeable that a recent research [10] designed a Network Dissection framework to quantify interpretability of CNNs and test the conclusion of [160]. They found that interpretability was not axis-independent. In other words, individual units are partially

disentangled representations and natural basis could actually produce meaningful results.

A method called Measure of Feature Importance (MFI) which works for both global interpretation and local interpretation is proposed in [176]. MFI is designed based on previous works of positional oligomer importance matrices (POIMs) [153] in computational biology and the feature importance ranking measure (FIRM) [193]. Specifically, FIRM is an extension of POIMs which is constrained to specific DNA applications. Although FIRM is applicable to a broad family of machine learning algorithms and considers the correlations between features, the generated scores are intractable for arbitrary input distributions [193]. MFI generalizes the concepts of POIMs and FIRM and considers nonlinear feature interactions. It provides both model-based feature importance (global interpretation) and instance-based feature importance attribution (local interpretation).

A method that also manipulates the attributes of the input datasets is introduced in [65]. The algorithm is called "GoldenEye", which detects the interactions of groups of attributes by a randomization technique. This method is a generalization of approaches in [19] [123] [97] and applicable to the interpretation of any classifier. Previous randomization methods work by randomly permuting the values of the attributes and measuring their impact on the predictive performance. However, they [65] only investigated one attribute at a time neglecting the interactions between features. By building an optimization algorithm and using fidelity as a metric, "GoldenEye" is able to detect the interactions between attributes and also provide information on which attributes are interacting.

2.1.4 Visualization

Neural Feature Visualization

With the thriving progress made in the past few years, feature visualization has established itself as the most promising research direction for neural network interpretations.

Usually, the most commonly applied technique is Activation Maximization (AM) [46]. This method enables the interpretation of arbitrary layers of a neural network, not just the first layer representation that could be easily interpreted by the learned filters: linear weights in the input-to-first layer weight matrix. It also assumes that

the input data are meaningful and displayable for humans, e.g. image data.

The idea of Activation Maximization is remarkably simple, but could generate high-quality visualizations. It essentially searches for input patterns which maximize the activation of a given hidden unit. This works because the patterns which fire the maximum activation could be a good first-order representation of what a unit is doing. This idea could be formulated as an optimization problem:

$$x^* = \underset{x}{\operatorname{argmax}}\ h_{ij}(\theta, x) \tag{2.3}$$

Here x is the input pattern and x^* is the optimal input pattern that the method tries to find and h_{ij} stands for the activation at unit i from a given layer j of the previous layers and θ represents the parameters of model. For an already trained neural network, these parameters are known. The maximum of h_{ij} is found by calculating the gradient of $h_{ij}(\theta, x)$ and moving x in the direction of this gradient. This step is called gradient ascent.

However, there are two shortcomings to this approach. First, it is hard to do initialization. It was mentioned that different random initializations sometimes generate the same optimal stimulus [46]. Second, the information about the invariance (the range of inputs that the unit is invariant to) is not available from the optimal stimuli which is just a single image. To address the second disadvantage of AM, the creators of Tiled convolutional neural networks (TCNN) [118] applied the method in [11] and extended it to arbitrary networks in order to visualize the invariant directions of a hidden unit [1]. However, the output of hidden neurons for TCNN are non-quadratic functions of inputs while [11] studies quadratic functions. This makes the extremely complex invariance of TCNN hard to be precisely captured.

Hence, another visualization approach [187] which utilizes a multi-layered Deconvolutional Network (Deconvnet) [188] (initially designed for unsupervised learning) is proposed to find non-parametric views of invariances. This approach maps the feature activities back to the input pixel space and could find the optimal stimulus at any layer in the model. The Deconvnet architecture is shown in Fig. 2.4 and the visualization results are displayed in Fig. 2.5. It is noticeable that when just examining the image patches of layer 5 it seems that they have nothing in common. An image

[1] The visualization results are here: http://ai.stanford.edu/ quocle/TCNNweb/

patch is a small (generally rectangular) piece of an image or a container of pixels. But after looking at its feature visualization we realize that it detects the grass in the background. This method is also a successful case that employs visual interpretation to diagnose and debug the problems of an already existing model [88] to improve the results.

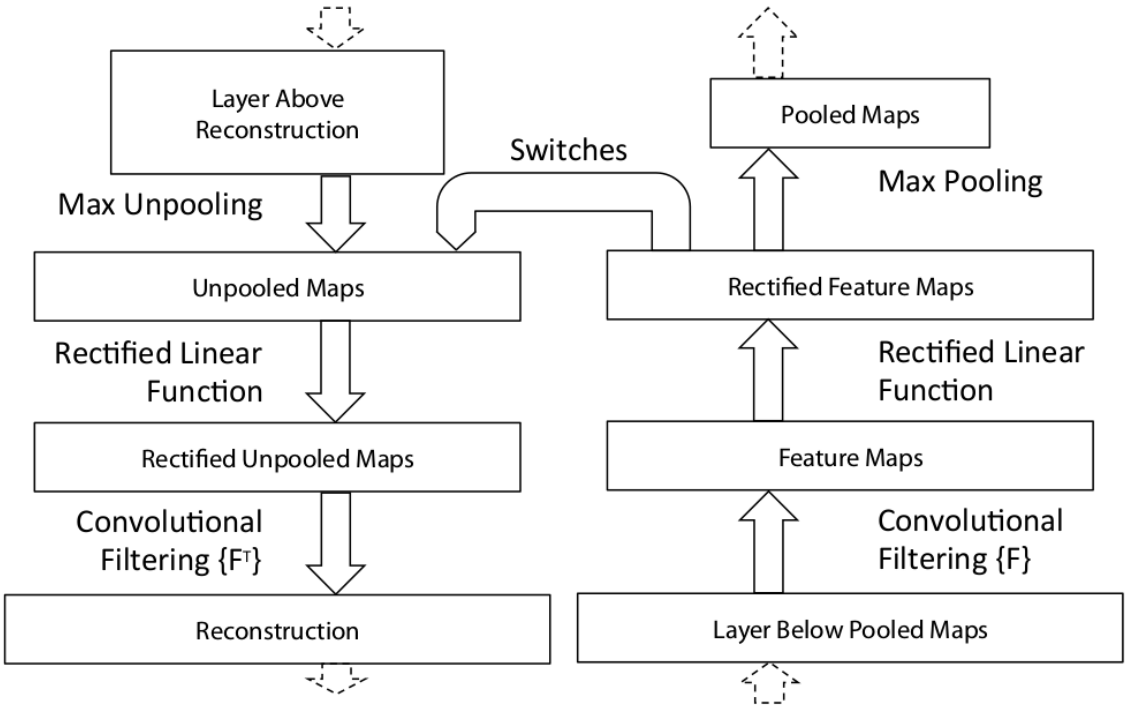
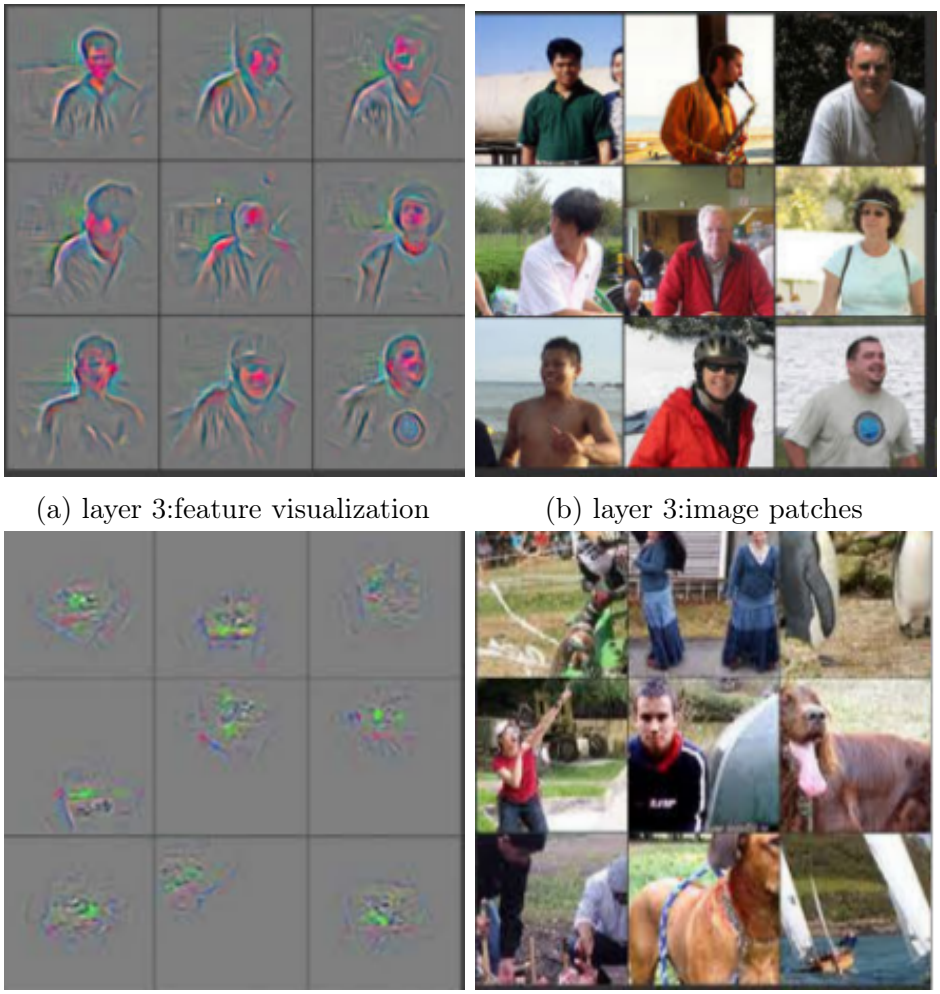


Figure 2.4: The Deconvnet architecture from [187].

Another method called Saliency Maps (SM) is provided in [149]. Both AM and SM are classified as gradient-based visualization techniques in this paper [149]. It tries to rank the pixels of a image according to their impact on the class score function and actually belongs to the local interpretations category. The class score function $S_c(I)$ is defined as follows:

$$S_c(I) = \omega_c^T I + b_c \tag{2.4}$$

Here c is a specific class, I is the input pattern, ω and b are the corresponding weights and biases respectively. Intuitively, we may think that the magnitude of the weight in ω for each pixel of the image I represents its importance in contributing to the class score. Unfortunately, the relationship between $S_c(I)$ and I is highly non-linear. However, in the neighbourhood of a given image I_0, we can approximate the relationship as a



(a) layer 3:feature visualization (b) layer 3:image patches

(c) layer 5:feature visualization (d) layer 5:image patches

Figure 2.5: Feature visualizations and image patches of the top 9 stimuli that excite a specific feature map at layer 3 and 5 for a trained neural network from [187].

linear function and this could be accomplished applying a first-order Taylor expansion:

$$S_c(I) = \omega^T I + b \tag{2.5}$$

where

$$\omega = \left.\frac{\partial S_c}{\partial I}\right|_{I_0} \tag{2.6}$$

It is also proved in this paper that the gradient-based visualization technique is a generalization of the deconvnet method used in [187] and the main difference [154] is how they handle back propagation through the rectified linear (ReLU) nonlinearity. They also mention that Saliency Maps is image-specific while their Class Model Visualisation (which uses AM) is not.

The all convolutional network (a homogeneous network solely consisting of convolutional layers) is introduced in [154]. To analyze this network and visualize the concepts learned by higher layers, a new technique: "guided propagation" which combines deconvnet with the gradient-based visualization technique is proposed [154]. This works remarkably well on all convolutional networks which don't have pooling layers.

A method for inverting representations, that is, reconstructing the original inputs from the outputs of a model, is presented in [109]. The main difference between this method and the deconvnet approach is that the latter reconstructs the inputs for certain neural activations and hence requires information about activations in the intermediate layers; the former uses only the final image code. This paper considers the interactions between neurons because they study sets of neurons as a whole. One novelty of this method is that they apply different regularization panalties as natural image priors to re-create the input image from scratch using random noise as initials. These priors are helpful as they can recover the low-level image statistics that are removed when building the model. More specifically, they formulate their reconstruction process as an optimization problem which tries to minimize the objective

$$x^* = \underset{x \in \mathbb{R}^{C \times H \times W}}{\operatorname{argmax}} \ \ell(\phi(x), \phi_0) + \lambda \mathcal{R}(x) \tag{2.7}$$

Here, C is the color channels, H and W are height and width of the input images,

$\phi(x)$ is the learned representation and ϕ_0 is the representation to be inverted, ℓ is the loss function, $\mathcal{R}$ is the regulariser standing for a natural image prior.

Another paper which considers regularized optimization is [186]. In this paper, two software tools for visualization are introduced. One visualizes the activations on each layer of a trained neural network interactively and lively. The other adds several regularization methods into the optimization process in the effort to generate better visualizations of the learned features. The reason they use regularization is that, without regularization or priors, the aforementioned gradient-based techniques would generate high frequency patterns that are unrecognizable.

More recently, the authors in [122] point out a shortcoming of the previously introduced techniques: these techniques assume that one neuron detects only one type of feature while the detection could be actually multifaceted. For instance, a bell pepper detector detects green, red and orange peppers as belonging to the same class. Paper [122] proposes a method to visualize these different facets. These types of algorithms are called: multifaceted feature visualization (MFV). The first approach of MFV is introduced in [181], which is only able to visualize two facets per neuron. This paper provides a more systematic method to visualize all facets. They also included a new regularization method: the center-biased regularization technique to reduce the production of repeated object fragments.

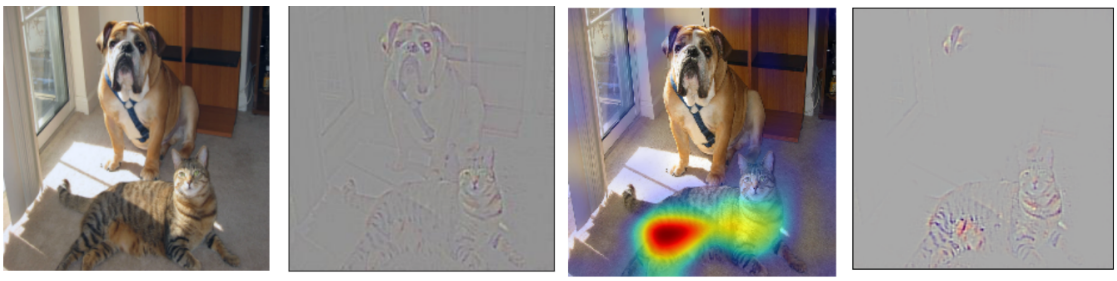
Then a learned natural image prior, a deep generator network, is applied in [120] aiding AM to produce better visualizations. This method is called deep Generator Network-based Activation Maximization (DGN-AM). An extended work is presented in [119] to tackle some of the limitations of DGN-AM. They also provide a probabilistic framework to unify and interpret AM approaches as energy-based models.

Attribution

Attribution is the type of visualization method that attributes the reason for a prediction by a DNNs to the importance of the input features. If the inputs are images, it tells us which pixels of the images are responsible for a specific predicted label. It also represents a set of techniques that explains the connections between neurons. It explains how individual neurons work together to arrive at a decision, which is different from the Neural Feature Visualization approach that just presents

what the network detects. As a matter of fact, the aforementioned methods of SM [149], deconvnet [187] and "guided propagation" [154] also belong to the attribution approach. Here we elaborate on more recent progress in this area.

An approach named Gradient-weighted Class Activation Mapping (Grad-CAM) is introduced in [144] based on a localization technique: Class Activation Mapping (CAM) [190]. Unlike CAM which alters the architecture of CNN and sacrifices model performance for transparency, Grad-CAM interprets the existing state-of-art deep models. It is also pointed out that although the techniques of Guided Back-propagation [154] and Deconvnet [187] provide fine-grained high-resolution visualizations they are not class discriminative. They [144] also show the possibility to fuse these techniques with their Grad-CAM to create a high-resolution and class-discriminative method called Guided Grad-CAM. Fig. 2.6 shows the comparisons between different approaches. Note that Grad-CAM and Guided Grad-CAM are highly class-discriminative (only highlights the cat region without the dog region) compared to Guided Backprop.



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d) Guided Grad-CAM 'Cat'

Figure 2.6: Comparison of visualizations for the 'tiger cat' class among Guided Backprop, Grad-CAM and Guided Grad-CAM from [144].

Due to the challenge that it is hard to discriminate the error produced by a model and that of an attribution method, two fundamental axioms: Sensitivity and Implementation Invariance are provided in [158]. Sensitivity essentially means that a baseline is needed and Implementation Invariance implies that attribution should be invariant to the implementation details of models if their functions are equal. They point out that Saliency Maps (SM) [149], Guided Back-propagation [154] and Deconvnet [187] violate the sensitivity axiom whereas the methods of DeepLift [147] and Layer-wise relevance propagation (LRP) [13] breaks the Implementation Invariance axiom. Then a method: Integrated Gradients satisfying both of the axioms

is presented in the paper.

A counter-factual approach is provided in [81] which instead of interpreting the DNNs directly evaluates many of the previously proposed interpretation methods on linear models as linear models are simple neural networks. The theoretical evaluations prove that the prevailing methods can't explain correctly the linear models, not to mention DNNs. Based on the theoretical framework they designed, they propose two explanation techniques: PatternNet and PatternAttribution which could produce improved explanations on DNNs. They formulate the theoretical model as

$$x = s + d \tag{2.8}$$

where

$$s = a_s y \tag{2.9}$$

$$d = a_d \epsilon \tag{2.10}$$

Here x is the input data, s is the signal which represents the part of x contributes to y, d is the distractor that obscures the detection task, a_s is the signal direction. They use a linear regression model to extract y from x and filter out d by a weight vector w^T.

$$w^T x = \underbrace{w^T a_s}_{=1} y + \underbrace{w^T a_d}_{=0} \epsilon = y \tag{2.11}$$

They indicate that in the case of linear models the interpretation methods of Saliency Map, Deconvnet and Guided Back-Propagation just analyze the weights which are determined by the distractor not the signal and hence have flaws. They also divided the current state-of-art interpretation methods into: Gradients, Signal methods and Attribution methods.

Based on the work of [170], a more general and formal framework for treating explanations as meta-predictors is introduced in [49]. They also identify the shortcomings of the existing interpretation models and provide a model-agnostic interpretation by image perturbations. Their proposed method modifies the Saliency Maps [149] technique by integrating the information over several rounds of back propagation.

Ironically, a big drawback of Saliency methods (which is believed to be a highly valued tool for interpretation) is presented in [80]. That is, the Saliency methods

are input variant: a constant shift of the input data which doesn't change a model's prediction would result in wrong attributions of some Saliency methods. Input variance is not a new concept as it has been demonstrated before on DNNs that human imperceptible perturbations on test images could arbitrarily change the network's predictions [160] [121]. Here the authors apply this idea to interpretation methods and propose another axiom related to [158]: input invariance. They also prove that Gradient and Signal methods satisfy this axiom while attribution methods fail. More recently, Generative Adversarial Network (GAN) combined with the autoencoder representation learning algorithm is proposed in [62] for the visual interpretations.

Data Visualization

Understanding data by visualizing them is an intuitive and important approach. Plotting two or three dimensional data is an easy task for most graphing tools. But for data that has more than three dimensions, special techniques are needed to transform them into a more visually understandable two-dimensional space. These techniques are called Dimension Reduction [175]. They could also be potentially helpful for assisting the interpretation of black box models.

Some of the popular dimension reduction techniques are Principal Component Analysis (PCA) [70], Multidimensional Scaling (MDS) [168] [37], t-distributed Stochastic Neighbor Embedding (t-SNE) [108] and Autoencoder networks [69]. Among these, t-SNE has become the de facto standard for interpreting deep neural nets. t-SNE mitigates the two problems that SNE [68] has: the optimization problem and the 'crowding problem'. It is able to reveal the local structure of the data as well as the global structure (such as clusters at multiple scales). And it also generates significantly better visualizations which was demonstrated in experiments [108] by comparison with many other non-parametric visualization techniques such as Sammon mapping, Isomap, and Locally Linear Embedding. The diagram in Fig. 2.7 produced by t-SNE shows a near perfect separation of the ten digit classes of the MNIST data set.

In [122], PCA and t-SNE are employed combined with the k-means algorithm for the purpose of different facet visualizations. Similarly, a dimension reduction technique (not specified in the paper) is applied in [166] to present the final visualization of

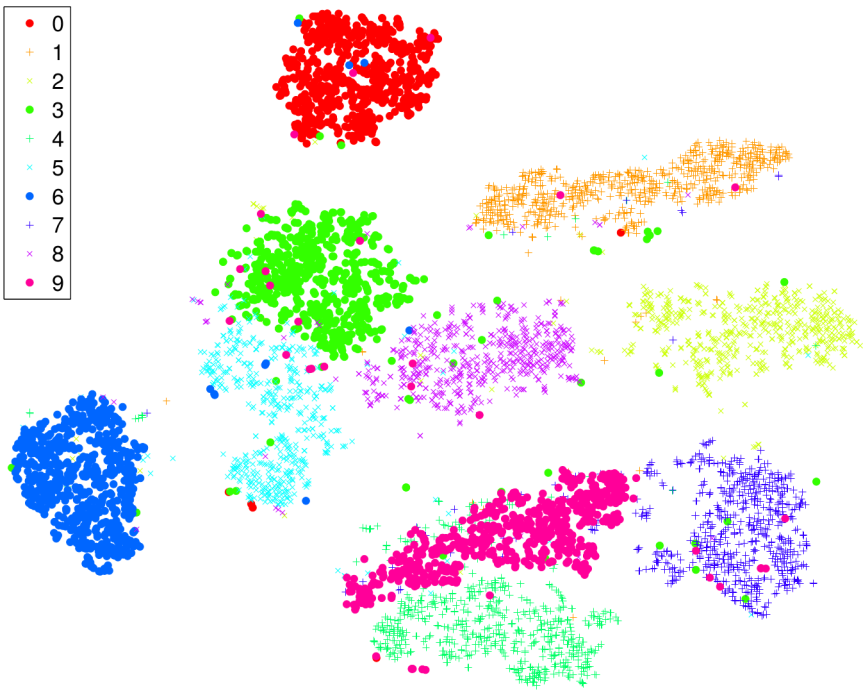


Figure 2.7: Visualization of the MNIST data set applying t-SNE from [108].

the treeview method they proposed for peeking into the classification process of a multi-layer perceptron. More recently, DarkSight [184] combines dimension reduction with knowledge distillation to generate better visualizations, which are claimed to be better than those produced by t-SNE.

T-SNE is a flexible approach but the produced results are tricky to evaluate. it is a tool to generate understanding but not to reach conclusive evidence. The visualization results heavily depend on the hyperparameters associated with the optimization process, especially the parameter: perplexity [180]. The users have to fine tune "perplexity" and "iterations" to achieve stable outcomes. T-SNE can't be used to find outliers and the cluster size and distances between clusters mean nothing. It may also split a natural cluster into different parts due to the local minimas in minimizing a nonconvex objective [174]. Moreover, some structures will never be able to be displayed in the t-SNE plots, e.g. the non-metric similarities [174].

2.2 Inherently Interpretable Models

Inherently Interpretable Models refers to the machine learning algorithm that doesn't need an extra explanation model to explain its decisions. That is, these kind of models are transparent and explainable by themselves. Here, we review some recent advances for transparent model design instead of focusing on the more traditional inherently interpretable models such as decision trees, lists and linear models.

To construct decision trees that are both accurate and interpretable simultaneously, a new classification method: Optimal Classification Trees (OCT) is proposed in [12]. It is motivated by the fact that traditional decision trees are built applying a greedy heuristic top-down approach, which splits each node in isolation neglecting the underlying characteristics of the dataset. This could ultimately impair the generalization ability of the model. Another problem for traditional decision trees is that they can't use the natural objective of misclassification rate of the training process (traditional use is impurity measure) for optimizing the trees as it is limited by the current top-down induction method's one-step optimization procedure. To solve these problems, it is best to construct an overall optimal decision tree and build new splits based on the information of all other splits. This is also a potential approach mentioned by the creators of decision trees: Breiman et al. [21], but was not able to be fulfilled limited by the computer technology at that time. Thanks to the algorithmic advances and the astonishing increase of computational power, it is now feasible to solve the NP-hard problem of the overall optimal decision trees via the mixed-integer optimization (MIO) solvers.

Motivated by the similar reason (the heuristic structure of decision lists [137]), a new transparent algorithm, decision sets, is proposed in [92]. This time the interpretability term is explicitly included in the objective function. It was pointed out that there is a trade off between interpretability and accuracy for machine learning algorithms. In terms of inherently interpretable models, rule-based models, e.g. Decision Trees and Decision Lists are often preferred, as they can find a balance between these two factors. Decision lists are usually considered more interpretable than decision trees, as they use the if-then-else statements with a hierarchy structure. But this structure reduces to some extent the interpretability, as to interpret an additional rule all previous rules should be reasoned about. Also new rules down

the list are applied to much narrower feature spaces, which makes the multi-class classification difficult where the minority classes deserves equally good rules. This motivates the proposal of the Decision Sets algorithm in [92], which produces the isolated if-then rules, where each rule could be an independent prediction. To realize this, an objective function which takes into account both interpretability (expressed by precision and recall of rules) and accuracy (expressed by size, length, cover and overlap) is configured. They show that solving the objective function is a NP-hard problem, and finding near-optimal solutions of it is possible. However, Decision sets' accuracy only approaches random forest, and its expressive power just catches up with decision tree.

Instead of doing post-hoc interpretations, work [183] focuses on finding more interpretable neural networks during the training process. They created a new model complexity penalty function: tree regularization to favor models whose decision boundaries could be well approximated by small decision trees. They measure human simulatability ("human simulation requires stepping through every calculation required to make a prediction" [183]) as the average decision path length and make the decision tree loss differentiable by adopting the technique of derivative-free optimization [5]. Their experiments show that using tree regularization could achieve high accuracy at low complexity.

2.3 Other Methods

This section introduces methods that don't fit in the categories of post-hoc interpretations and inherently interpretable models. Such methods apply novel ideas such as Influence Functions, Game Theory and Information Theory from other disciplines outside of the pure machine learning community to interpret current deep learning models.

A powerful technique, Influence Functions, is applied in [84] to understand model behaviors. Influence Functions is a classic method from statistics but seldom adopted in the machine learning community. Different from the Post-hoc Interpretation methods in section 2.1 where the trained models are fixed, in this paper, the model to be interpreted is treated as a function of the training data. Instead of using model parameters or test data to explain a model in Post-hoc Interpretations, the paper

traces a model's behavior back to its training data and tries to find out what training examples lead to certain predictions of the model. To inspect the model, they are curious about what would happen if a specific training point is removed from the whole training data or if a training point is perturbed. This is achievable by retraining the model from scratch without the specific training point or with the modified training point, but it is prohibitively slow and expensive for very complex models. Fortunately, Influence Functions can provide efficient approximations on the parameter change under the two cases without retraining the model. For example, in the first case, it is proved that removing a point produces the same parameter change as upweighting the training point by a small value. As this method upweights a specific training point by an infinitesimally small value, it is actually a local interpretation method. More work still needs to be done to solve the global interpretation problem.

Although many interpretation methods are proposed in recent years, it is not clear about how these methods are related and when to prefer one method over the other. To address this issue, the authors in [107] identified six previous methods, namely, LIME [133], DeepLIFT [147], Layer-wise relevance propagation (LRP) [13], Shapley regression values [98], Shapley sampling values [157] and Quantitative Input Influence [40] as additive feature importance methods, which share a common attribute that they have an explanation model that is a linear function of binary variables. Then they unify these methods by proposing a new framework: SHapley Additive exPlanations (SHAP). They use concepts from game theory to prove that there exists a single unique solution (that is, Shapley values) in this class of additive feature attribution methods with three desirable properties: Local accuracy, Missingness and Consistency. They define the solutions as SHAP values and to satisfy these three properties they present novel Model-Agnostic and Model-Specific Approximations to approximate the SHAP values.

A different approach [148] which doesn't explain the decisions made by neural networks sheds some light on the theoretical understanding of the inner workings of Deep Learning and Deep Neural Networks. It is based on their previous work on Information Plane and the information bottleneck principle [167]. This previous work suggests that the goal of the network is to optimize the Information Bottleneck (IB) tradeoff between compression and prediction. Their more recent work demonstrates the

effectiveness of the Information-Plane visualization of DNNs. They find that Stochastic Gradient Descent (SGD) optimization in deep learning has two different and distinct phases: empirical error minimization (ERM) and representation compression. Also, the representation compression phase begins when the training errors become small. The experiments also demonstrate that the converged layers lie on or very close to the IB theoretical bound and the main advantage of adding more hidden layers is to reduce training time.

2.4 Conclusions

Based on our reviews, we find that not all methods strictly obey the taxonomy we present here. These interpretation techniques are more likely to exhibit hybrid features: usually overlaps of the taxonomy. The taxonomy discussed in this chapter could be regarded as the building blocks to construct much better and complete interpretations. It should be kept in mind that more and more new papers are increasingly published at a speed much faster than ever before. It seems that we can't even keep up with their pace and as a finite document we have to stop the review here. In the following chapters, chapters 3 and 4, we present our two different approaches for establishing interpretable models to interpret deep neural nets. In chapter 5 we provide our solution for utilizing interpretability from the perspective of the model designers.

Chapter 3

Interpretable Deep Convolutional Neural Networks via Meta-learning

This chapter provides a method to understand the behavior of a neural network model. We propose our methodology named CNN-INTE [102] to interpret deep Convolutional Neural Networks (CNN) via meta-learning. In this work, we interpret a specific hidden layer of the deep CNN model on the MNIST image dataset. We use a clustering algorithm in a two-level structure to find the meta-level training data. Subsequently, Random Forest is applied as base learning algorithms to generate the meta-level test data. The interpretation results are displayed visually via diagrams, which clearly indicate the manner in which a specific test instance is classified. Our method achieves global interpretation for all the test instances on the hidden layers without sacrificing the accuracy obtained by the original deep CNN model. This means our model is faithful to the original deep CNN model, which leads to reliable interpretations.

This method is essentially a combination of techniques from data visualization and surrogate models based on the taxonomy in chapter 2. Specifically, Random Forests and Decision trees are employed as surrogate models to construct the interpretable model and PCA (a data visualization technique) is adapted to visualize the interpretation results. It should be noted that Random Forest is applied as base learners to generate meta-level test data and not applied for meta-level interpretation purposes. Our method is inspired by the idea of generating different facets within a class in [122]. However, there are many differences. Instead of performing neural feature visualizations, we employ the techniques of data visualization combined with other machine learning algorithms to explain how a specific decision is made by analyzing the activations of a specific hidden layer.

3.1 Introduction

The disadvantage for the Inherently Interpretable Models is that there is a trade off between interpretability and accuracy: it is not easy to learn an interpretable (so presumably simple) model expressing a complex process with a very high accuracy. In contrast, post-hoc interpretations take the opposite approach and do not sacrifice accuracy : they first build a highly accurate model without worrying about interpretabilty, and subsequently use a separate set of re-representation techniques to assist the user in understanding the behavior of the algorithm. Among these techniques surrogate models use the aforementioned relatively simple and interpretable algorithms to explain the behavior of a complex model and the reasons why a given classifier, treated as a black box, classifies a given instance in a particular way, e.g. TREPAN[38], LIME [133], BETA [93].

Deep learning methods have lately been very successful in image processing and natural language processing. They are sometimes viewed as a representation learning approach [60], which learns refined features that could improve a model's generalization ability. In this approach, we interpret a neural network model by providing visual presentations of the connections between input features and the output predictions.

We are reporting our work where we try to interpret the inner mechanisms of deep learning. Our method: CNN-INTE is inspired by [122]. We design and implement a tool that helps the user understand how the hidden layers in a deep CNN model work to classify examples. And the results are expressed in graphs which indicate sequential separations of the true class and the hypothesis. The main contributions of our method are as follows:

- Compared to LIME [133] which provides *local* interpretations for the entire model in specific regions of the feature space, our method provides a *global* interpretation for any test instance on the hidden layers in the whole feature space.

- Compared to models which apply inherently interpretable algorithms, e.g. [92], our method has the advantage of not compromising the accuracy of the model to be interpreted. This produces more reliable interpretation.

- Contrary to [133] and[93] which treat the model to be interpreted as black box, we access the hidden layers of deep CNN models and interpret them.

The experiments are implemented in the TensorFlow [1] platform, which makes our model scalable to big datasets more easily. Scalability is an issue pointed out as future work in [133] and [93] but not implemented yet.

3.2 Methodology

Our methodology could be classified as the *post-hoc* interpretations (chapter 2.1), where a trained model is given and the main task is to interpret it. This method is close to the approach of surrogate models, but is also different in many ways. First, when building the interpretations we have to access the inner structure of the trained network and we directly interpret the hidden layers of a deep CNN. In comparison, surrogate models use a separate interpretation model to interpret the relationships between the input variables and the outputs leaving the trained black-box model untouched. Second, compared to LIME [133] which only has local interpretability, our method achieves global interpretability. Similar to LIME, we also provide qualitative interpretation with graphs to visualize the results. As our method interprets deep CNN via Meta-learning, we first briefly introduce deep CNN, meta-learning and then discuss our framework in details.

3.2.1 Deep Convolutional Neural Network

This section introduces the deep CNN model we plan to interpret. As we implement our program in TensorFlow, we use its TensorBoard function to present the structure of the deep CNN we constructed in Fig. 3.1.

There are three major components of a deep CNN: convolutional layer, pooling layer and fully connected layer (the same as in regular neural networks). A deep CNN model is usually a stack of these layers. In the convolutional layer, a filter is used to compute dot products between the pixels of the input image at specific positions and values of the filter, producing one single value in the output feature map. The convolution operation is completed after the filter is slided across the width and height of the input image. Following the convolutional layer, an activation function, often a

rectified linear unit (ReLU) [116], is applied to inject nonlinearities into the model and speed up the training process. Following ReLU is the pooling layer which is a non-linear down-sampling layer. A common algorithm for pooling is the max pooling algorithm. In this algorithm, each sub-region of the previous feature map is turned into a single maximum value in this region. Max pooling reduces computation and controls overfitting. In order to calculate the predicted class, after performing max pooling, the feature map needs to be flattened and fed into a fully connected layer. In the last layer, the output layer, a softmax classifier is applied for prediction.

The structure of the deep CNN model we designed is illustrated in Fig. 3.1. Since it was generated by TensorBoard automatically, some of the words were not fully displayed. Here we explain accordingly from bottom to top. "Placeholder" represents the interface for inputting the training data. "Reshape" is needed first to convert the input one-dimensional image data into two dimensional data. It should be noted that the images themselves are two-dimensional, but the data are presented in a one-dimensinal form. To go through convolutional process, the data hence needed to be converted into a two-dimensional format. In our experiment, we use the MNIST dataset [95]. The 784 input features are converted into a two-dimensional 28×28 image. Our model has two series of a convolutional layer followed by a pooling layer: "conv1"-"pool1"-"conv2"-"pool2", which are followed by one fully connected layer "fc1". As a fully connected network is susceptible to overfitting, the "dropout" operation [155] applied after "fc1" aims to reduce it. In this operation, a probability parameter p is set to keep a specific neuron with probability p (or drop it with probability *1-p*). The "Adam optimizer" [82], rather than a standard Stochastic Gradient Descent optimizer is used to train the model via modifying the variables and reducing the loss. "fc2" is the output layer with 10 neurons: each represents one of the classes: 0-9.

3.2.2 Meta-learning

Meta-learning is a learning method which learns from the results of the base classifiers. It has a two-level structure, where the algorithms used in the first level are called base-learners and the algorithm in the second level is called the meta-learner. The base-learners are trained on the original training data. The meta-learner is trained by the predictions of the base classifiers and the true class of the original training data.

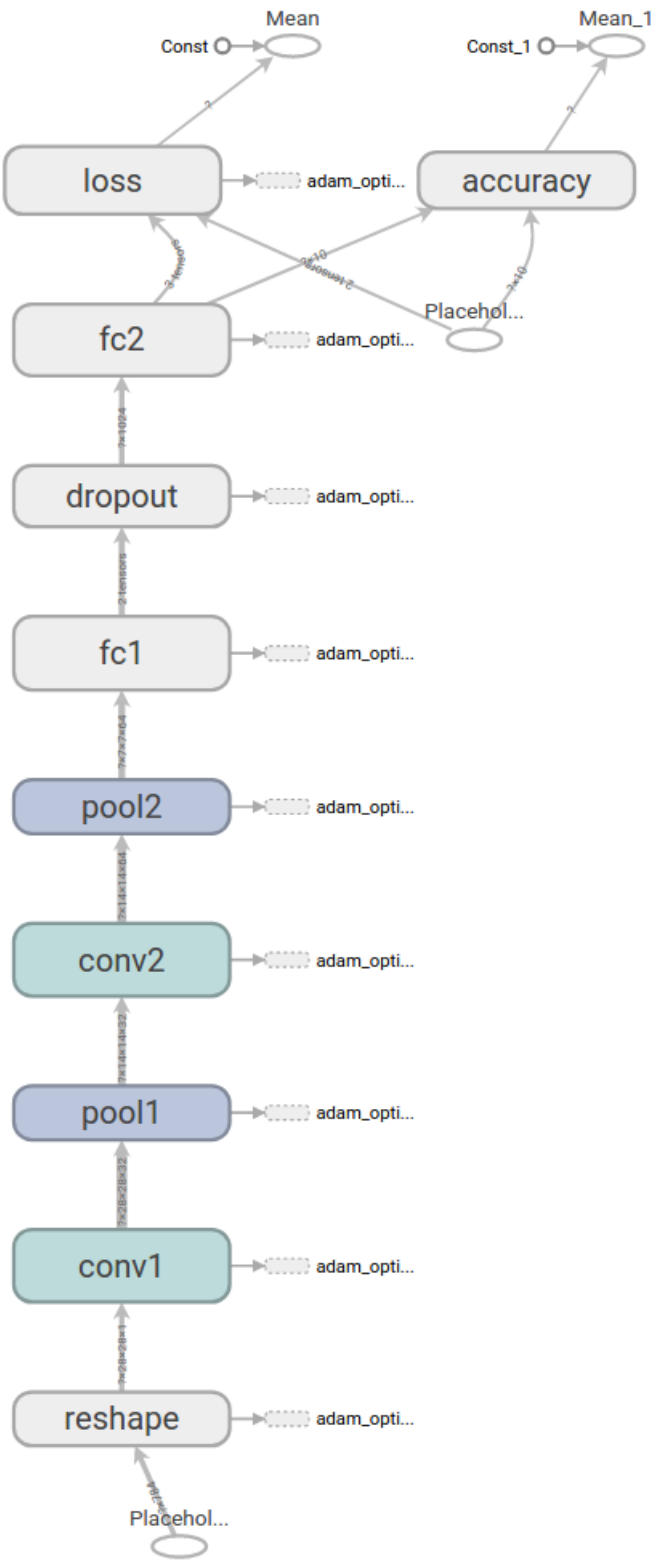


Figure 3.1: Structure of our deep CNN model generated by TensorFlow's TensorBoard.

When training the meta-learner, the "Class-combiner" strategy [28] is applied here, where the predictions include just the predicted class.

To understand the meta-learning algorithm intuitively, Fig. 3.2 illustrates a simplified training process for meta-learning [103]. The numbers 1, 2, 3, 4 represent the four steps of training. In the 1st step, the base learning algorithms 1 to m are trained on the training data. In the 2nd step, a validation dataset is used to test the trained classifiers 1 to m. In the 3rd step, the predictions generated in step 2 and the true labels of the validation dataset are used to train a meta-learner. Finally, in the 4th step, a meta-classifier is produced and the whole meta-learning training process is completed.

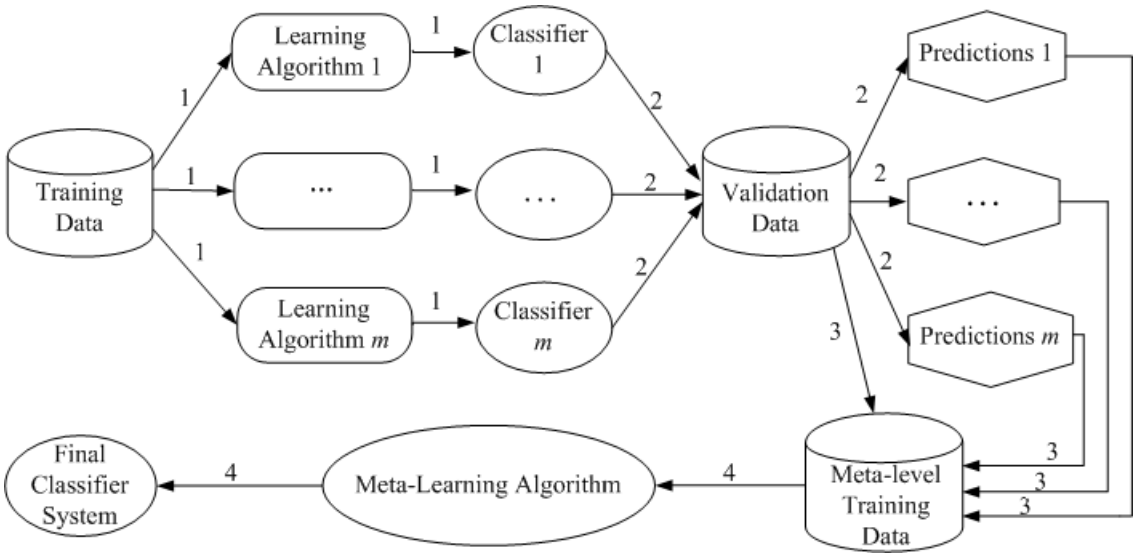


Figure 3.2: Meta-learning training process.

Once the training process is accomplished, the test process is much easier to execute. Fig. 3.3 presents a simplified test process [103]. In the 1st step, the test data is applied to the base classifiers to generate predictions which, combined with the true labels of the test data, comprise the meta-level test data in 2nd step. In the 3rd step, the final predictions are generated by testing the meta-level classifier with the predictions in the 2nd step and the accuracy could be calculated.

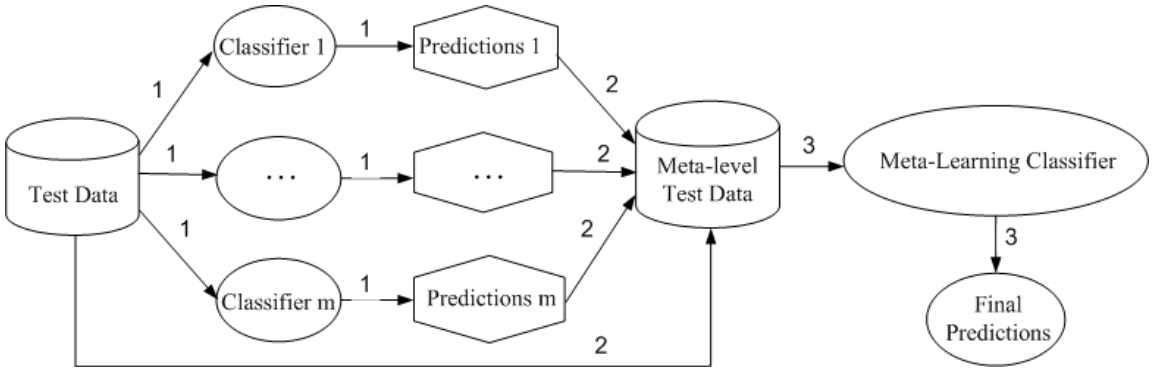


Figure 3.3: Meta-learning test process.

3.2.3 Framework

Our framework is named CNN-INTE which stands for Convolutional Neural Network Interpretation. It is similar to meta-learning, but different in a few ways. In this work, we interpret the first fully connected layer "fc1" of the deep CNN model illustrated in Fig. 3.1.

The training process is shown in Fig. 3.4. In the 1st step, the original training data is used to train a CNN model. In the 2nd step, the trained parameters (such as weights, biases, etc.) generated in the 1st step are used to calculate the values for the activations of the first fully connected layer: fc1. In the 3rd step, a clustering algorithm is used to cluster the data generated in step 2 into a number of groups which we define as explanation factors henceforth. In the 4th step, the data belonging to each of the explanation factors are clustered again generating a number of clusters, each assigned a unique ID. In the 5th step, these IDs are grouped together as the training features in the meta-level, using the labels of the original training data as the label for the meta-learner. In the 6th step, the features of the original training data and the IDs (set as labels) in step 4 are used to train several random forests [96]. These trained random forest classifiers would be applied to generate test-level features for the meta-learner: Decision Tree. In the 7th step, a Decision Tree is trained based on the data generated in the 5th step.

Now we discuss the training process in more detail. Assume that the training data T has N numbers of instances and that layer "fc1" has H neurons. The labels of the training data are $T_y = \{l_1, l_2, ..., l_N\}$. Once the deep CNN model is trained, for each training instance t_i, we calculate the activations at each hidden neuron on this layer.

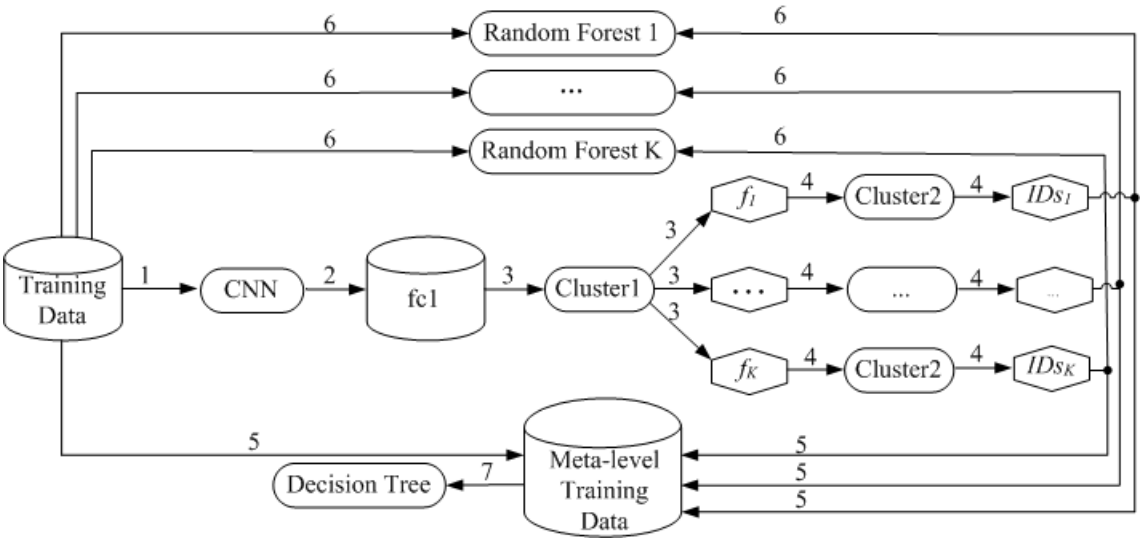


Figure 3.4: CNN-INTE training process.

Hence, we obtain a matrix S with size $H \times N$. To construct the meta-level training data, we use a clustering algorithm to cluster S along the hidden layer axis into several explanation factors $F = \{f_1, f_2, ..., f_K\}$. According to [122], the value of K is user defined. In our experiments, this value is determined as the one which produces the best accuracy performance for the meta-learning algorithm. We also discuss how to avoid this problem in chapter 6. Then within each of the explanation factors, we cluster the data again, this time along the axis of the instances. The clustering results are the IDs each instance belongs to. For instance, if the number of clusters is 10, after the second level clustering each instance will have an ID number between 0-9. All the IDs combined with the true labels of the training data builds up the meta-level training data.

To present the technical details of the CNN-INTE training process, we provide the pseudo code in Algorithm 1. Line 1-3 is the initialization of the algorithm. In line 4, the activations S are clustered into K explanation factors, where K is the number of clusters set in the first level clustering algorithm C^l. In lines 5-7 the same clustering algorithm C^l is applied to all the explanation factors to generate K sets of ID numbers. Lines 8-9 use the generated ID numbers and the true labels of the original training data to train the meta-learner: C^m. Until now, the training process is not done yet. We still need to generate the base models to be used in the test process. Lines 10-12 use the features of the original training data and the ID numbers to train K base

models. The output of the training process would be the meta-lever classifier: $\tilde{M}$ and K base models: $B = \{M_1, M_2, \cdots, M_K\}$.

Algorithm 1 CNN-INTE Training Process

Input: activations: S; training data: T; Meta learning algorithm: C^m; Clustering algorithm: C^l; Base learning algorithms: $\{C_1, C_2, \cdots, C_K\}$
$E = \varnothing$
$S_{cv} = \varnothing$
$\{f_1, f_2, \cdots, f_K\} = C^l(S)$ ▷ figure 3.4, step 3
for $k = 1 \cdots K$ **do**
    $IDs_k = C^l(f_k)$ ▷ figure 3.4, step 4
end for
$S_{cv} = \{IDs_1, IDs_2, \cdots, IDs_K, T_y\}$ ▷ figure 3.4, step 5
for $k = 1 \cdots K$ **do**
    $M_k = C_k(T_x, IDs_k)$ ▷ figure 3.4, step 6
end for
$\tilde{M} = C^m(S_{cv})$ ▷ figure 3.4, step 7
$E = (\{M_1, M_2, \cdots, M_K\}, \tilde{M})$
Output: Ensemble E

Fig. 3.5 is a toy example that illustrates the above process. In this example, there are 5 hidden neurons and 6 training instances. We set the number of clusters for both the first and second level clustering as 3. Hence, the matrix S with size 5×6 is first clustered into 3 explanation factors $\{f_1, f_2, f_3\}$ horizontally. For each factor, the activations are again clustered into three clusters vertically, e.g. f_1 is clustered into $\{C_{11}, C_{12}, C_{13}\}$. If we set the ID numbers for these cluster as $\{0, 1, 2\}$, then the corresponding ID numbers for $t1$ to $t6$ in factor f_1 according to Fig. 3.5 are $\{0, 0, 1, 1, 2, 2\}$. Hence, the meta-level training features are expressed as:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 1 & 2 & 2 \end{bmatrix}$$

These data combined with the corresponding training labels of the original training data are used to train the meta-learner. Here the meta-learner we used is the Decision Tree [130], an inherently interpretable algorithm. Its tree structure provides an excellent visual explanation of the predictions.

The test process of the meta-model is exactly the same as the meta-learning test

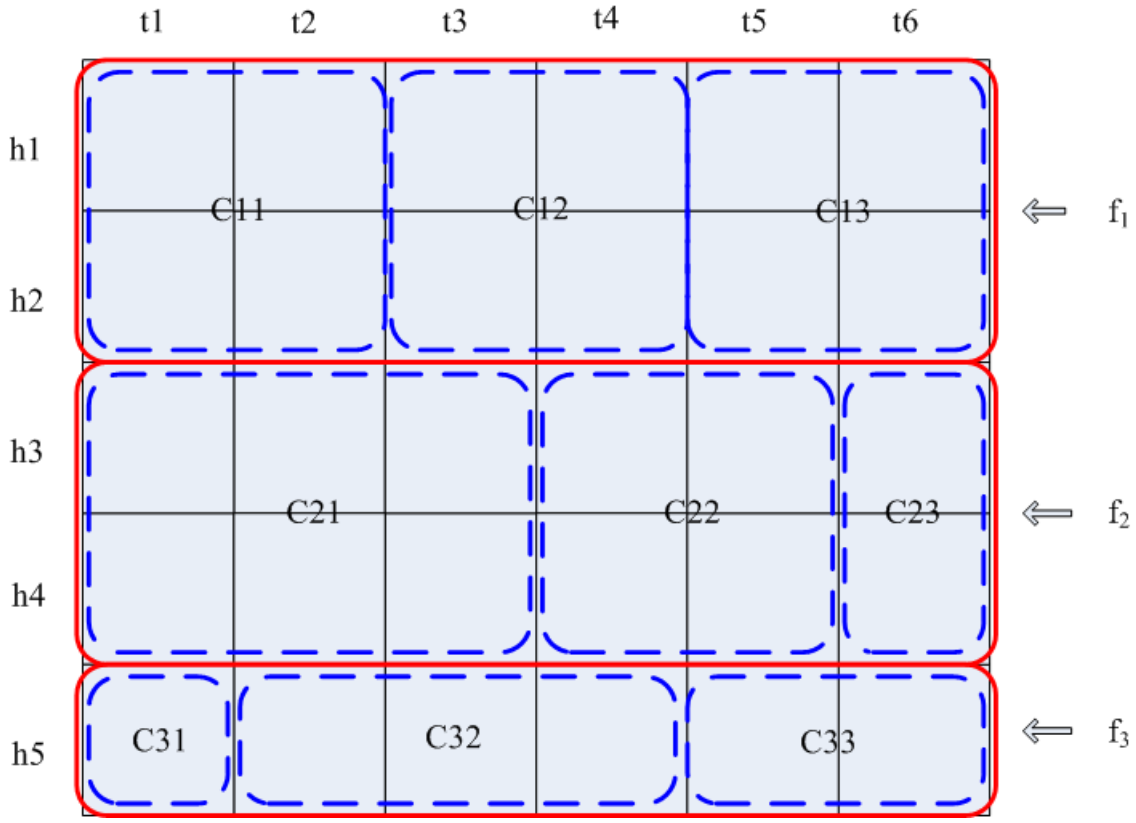


Figure 3.5: Toy example for the generation of Meta-level training data.

process, which is shown in Fig. 3.6.

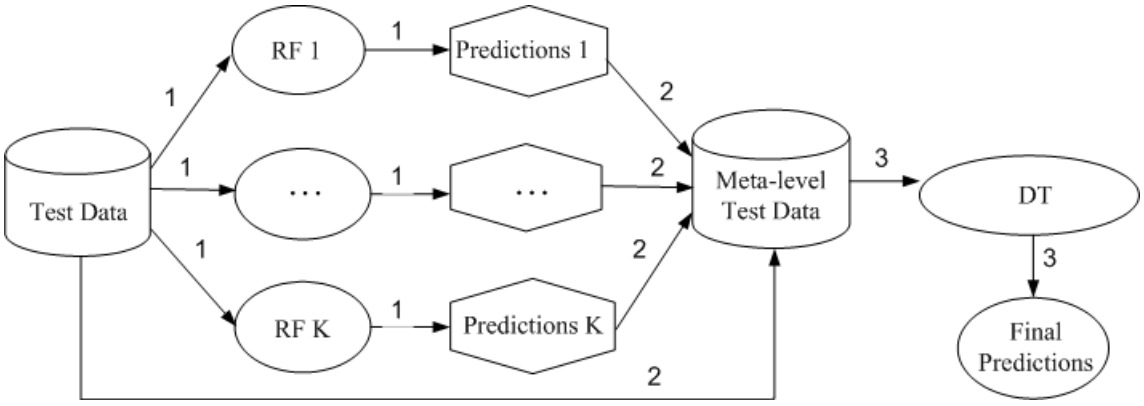


Figure 3.6: CNN-INTE test process.

In the test process, we use the original test data to test the base classifiers generated in the meta-level training process to obtain the meta-level test data's features. The base-learner we applied is random forest. The number of base models is equal to the number of explanation factors. Hence, we have K base models: $B = \{M_1, M_2, ..., M_K\}$. In the toy example, there are three explanation factors which lead to three base models. The training data for the first base model corresponding to f_1 are:

$(t1 - l1)$	0
$(t2 - l2)$	0
$(t3 - l3)$	1
$(t4 - l4)$	1
$(t5 - l5)$	2
$(t6 - l6)$	2

Here $t_i - l_i$ represents the features of the original training instance i. Once we obtain K base models, we can use the original test data to test them to produce the meta-level test data. These data are then fed into the trained decision tree model to interpret individual test predictions.

3.3 Experiments

The dataset we use is the MNIST database of handwritten digits from 0 to 9 [95]. The original dataset has 60,000 examples for training. As a default behavior of Tensorflow,

when reading the inputs, 55,000 examples are set as the training data, 5000 are saved separately for validation purpose. There are also 10,000 examples for testing. Each of the examples represents a 28×28 image with pixels flattened as 784 features. The experiments are performed on the TensorFlow platform.

3.3.1 Experimental Setup

First of all, we need to train a deep CNN model having a high accuracy. We first reshape the input training data into 55000 images each with size 28×28. Training all the data on every epoch is expensive, which requires a lot of computing resources and may lead to the termination of the program. Here we apply stochastic training: on the first epoch, we select a mini-batch of the training data and perform optimization on this batch; once we loop through all the batches, we randomize the training data and start a new epoch. In our experiment, we set the epoch $e = 1000$, batch size $b = 50$. Stochastic training is cheap and achieves similar performance to using the whole training data in every epoch. For each mini-batch, in the first convolutional layer, we apply 32 filters (or kernels) each with size 5×5, which generates 32 feature maps. In the first pooling layer we apply filters with size 2×2. The stride size is set as 2. The second convolutional layer uses 64 filters with the same size as the first convolutional layer. The second pooling layer has the same parameters as the previous one. Immediately after this pooling layer is the first fully connected layer: fc1. We set the number of neurons for this layer as 128. To reduce overfitting we also set the dropout [155] parameter $d = 0.5$, which means a neuron's output has a 50% probability of being dropped. The last layer is the second fully connected layer (or the "readout layer"), which has 10 neurons, with each neuron outputting the probability of the corresponding digits 0-9. The test accuracy of this trained CNN model on the test data is 93.9%.

Now comes the key part related to interpretability. We understand interpretability of a model as being the ability to provide visual or textual presentation of the connections between input features and output predictions. We first feed the trained fully connected layer $fc1$ with the original training data, which would produce a data S with size of 128×55000. We then cluster S into several explanation factors. The clustering algorithm we applied is the k-means algorithm [64]. The number of

explanation factors is equal to the number of clusters which we set as 8 in this level. Hence, S is now turned into a list $F = \{f_1, f_2, ..., f_8\}$ with size 8×55000 having each row representing the data belonging to each factor. In the second level clustering, for each factor in F we use the k-means algorithm again to cluster them into a number of clusters. We set the number as 10 in our experiment. Hence each cluster will be assigned a unique ID number between 0 and 9. Then we use the IDs belonging to each training instance and the true labels of the original training data to train a decision tree algorithm. Since the trained decision tree is huge, we show the structure of it in this link [1]. We set the maximum depth of the decision tree as 5. Although a deeper decision tree would generate better accuracy, it makes it harder to interpret with too many tree levels.

To obtain the test data for decision tree, we first use the original training data's feature as features and the IDs for each factor in F as labels to train the corresponding random forest algorithm [96], generating 8 base models. For random forest, we set the number of trees as 20 and the maximum nodes as 2000. Finally we use the original test data to test the 8 trained base models. The generated predictions become the features of meta-level test data with sizes of 10000×8. Using the meta-level test data on the trained decision tree produces an accuracy of 92.8% with tree depth=5. This value is comparable to the test accuracy on the trained deep CNN model: 93.9%. It should be noted that the decision tree's accuracy could be further improved by increasing the depth of tree and tuning other related parameters.

3.3.2 Experimental Results

To interpret the deep CNN model's behavior on the test data, we intend to use diagrams generated by our tool, CNN-INTE, to examine individual predictions on the test data. Hence, we provide qualitative interpretations visually. We arbitrarily selected two test instances that were correctly classified by the decision tree and one test instance that was wrongly classified. It should be noted that this tool could be used on any test instances globally and not just limited to the three cases we provide. The details of the selected test instances are shown in Table 3.1. Here

[1] https://drive.google.com/file/d/1X2ingiAy0COZtfDKFmuNSNXWsL6mC-A3/view?usp=sharing

"f_0-f_7" represents the features of the meta-level test data, "label" is the test label in the original test data, "pred" is the prediction generated by the decision tree on the meta-level test data. "True1" and "True2" represent the two correctly classified instances and "Wrong1" is the wrongly classified instance.

Table 3.1: Instances Selected From the Meta-level Test Data

	Features and labels									
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	***label***	***pred***
True1	4	0	7	7	0	0	0	0	3	3
True2	5	0	0	5	9	5	3	6	0	0
Wrong1	5	6	7	9	6	4	7	9	5	9

In order to examine the classification process visually, we check each feature value according to the trained structure of the decision tree and plot the graphs of the activations corresponding to the true label and the hypothesis. The interpretation result for instance "True1" is shown in Fig. 3.7. As the true label for this instance is 3, all other classes could be regarded as hypotheses and this is why there are no graphs for "Hypothesis: 3" in Fig. 3.7. Each row represents the examination of the feature values corresponding to different explanation factors in different levels of the trained decision tree, e.g. the first row represents the root level of the decision tree. Since we set the depth of the decision tree as 5, there are 5 rows in all. Each column stands for the query of whether the test instance belongs to the corresponding hypothesis over the nodes visited.

To prove that the generated visual interpretations are correct, we examine the structure of the generated decision tree together with the visual interpretations. It should be noted that X in the decision tree structure is the same as f and 'value' stands for the number of data points belonging to each of the ten digits. One question arises in this procedure as to how do we separate the projection error from the error of the model itself. This should be regarded as one of the limitations of post-hoc interpreations which we would address in chapter 6.

Take the column of "Hypothesis:0" as an example. The goal is to find if the label of the test instance is 0. In the 1st row we extract the activations corresponding to "f_6" which satisfies the condition that $f_6 \leqslant 4.5$ (this is determined by the trained decision

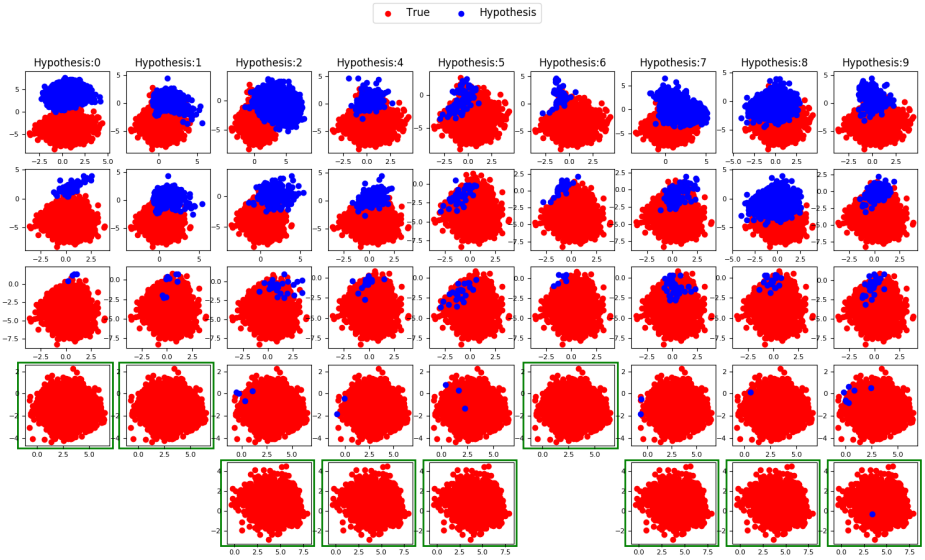


Figure 3.7: Example of a correctly classified test instance: True1. Each point represents an activation belong to either the true label (in red) or the hypothesis (in blue)

tree) and we draw a graph between activations that belong to label=0 (hypothesis) and label=3 (true). Then we check the graph to evaluate whether the data corresponding to the true class could be separated from the hypothesis. The answer is no because the hypothesis, represented as blue points, overlaps with the true class, shown as red points. Hence, we need to query the trained decision tree further. The values of the explanation factors we need to check are: $f_6 \leqslant 2.5$ for 2nd row; $f_6 \leqslant 0.5$ for 3rd row; $f_7 \leqslant 0.5$ for 4th row; $f_1 \leqslant 0.5$ for 5th row. In this process, we noticed that in the 4th row the true class and the hypothesis class are successfully separated, as only the red points corresponding to the true label are left. Therefore, we don't need to examine further and that's why the graph for the 5th row is not displayed. We highlight the graph with green rectangles if the final results are separable and red vice versa. The same idea is applied on other hypotheses. We also draw the graphs for instances "True2" and "Wrong1" in Fig. 3.8 and Fig. 3.9 respectively.

3.4 Conclusions and limitations

In this work, we present an interpretation tool CNN-INTE, which interprets a hidden layer of a deep CNN model: to find out how the learned hidden layer classifies new

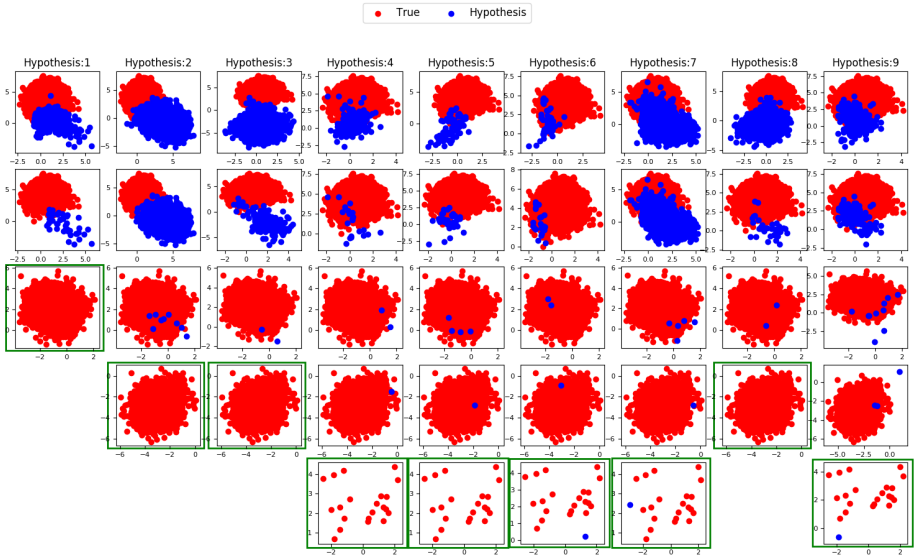


Figure 3.8: Example of a correctly classified test instance: True2.

test instances. Although we just show the results for the first fully connected layer before the read-out layer, the approach could be applied on any hidden layers. The interpretation is realized by finding the relationships between the original training data and the trained hidden layer "fc1" via meta-learning. We used two-level k-means clustering algorithm to find the meta-level training data and random forests as base models for generating meta-level test data. The visual results generated by our program clearly indicate why a test instance is truly or wrongly classified by checking if there are any overlaps of the corresponding activations. One limitation of this approach is that it only applies to CNN at the moment. To expand this method to other networks the approach would need to be adapted. Moreover, we also need to perform more experiments to demonstrate the effectiveness of this method on other datasets. One such example is the Cifar10 dataset which is also an image dataset and could also apply the similar structure of the CNN that the MNIST dataset trained on. The same approach we proposed in this chapter would therefore be applicable and relevant to understand how a given hidden layer classifies a given instance.

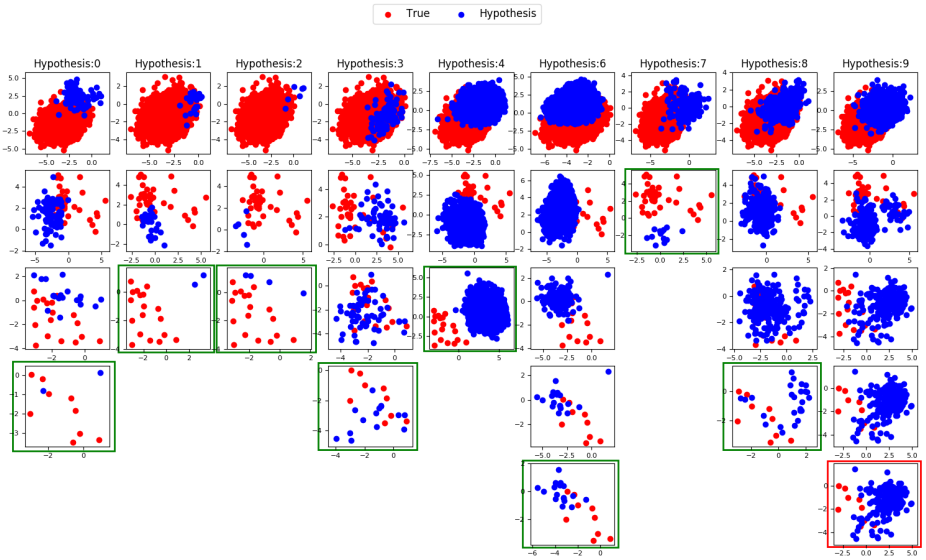


Figure 3.9: Example of a wrongly classified test instance: Wrong1.

Chapter 4

Improving the Interpretability of Deep Neural Networks with Knowledge Distillation

In this chapter, we contribute to resolve the tension between interpretability and accuracy . We propose to apply the Knowledge Distillation technique to distill Deep Neural Networks into decision trees in order to attain good performance and interpretability simultaneously. Knowledge Distillation is pertinent to the "surrogate models" we introduced in chapter 2. When designing the model, we formulate the problem of learning a surrogate model, decision tree, from the probabilities of a complex model as a multi-output regression model. The experiments demonstrate that the student model achieves significantly better accuracy performance (about 1% to 5%) than conventional decision trees at the same level of tree depth. The experiments are implemented on the TensorFlow platform to make it scalable to big datasets. To the best of our knowledge, we are the first to distill Deep Neural Networks into conventional decision trees on multi-class datasets.

4.1 Introduction

For high stakes domains such as clinical decision support, decision trees are preferred over DNN for disease diagnosis due to their ease of interpretation [16] [185] [48]. However, decision trees overfit easily and perform poorly on large heterogeneous electronic health records (EHR) datasets [31]. For inherently interpretable models, it is therefore desirable to develop models to find a spot where both interpretability and performance could be ultimately optimized.

As introduced in chapter 2, an intuitive and natural way to interpret neural networks is through visualization. However, recent research [160] shows that it is space, not the individual units, that contains the semantic information in the higher layers of neural networks, which means that the common approach: activation maximization [187] [61] [59] [149] [114] applied previously for interpretation has flaws. A related

suggestion was given in [54] to abandon the idea of inspecting individual hidden units. Thus alternative solutions for interpretation are required.

For inherently interpretable models, this approach presents a severe constraint on the selection of algorithms. Besides, although humans can comprehend these models, they fail to model more complex problems with high accuracy performance.

In this chapter, we apply the most recent model-agnostic approach [134] which performs post-hoc explanations on the trained models. Past research for model-agnostic interpretations focused on either global interpretations or local explanations as discussed in chapter 2. We strive here for global interpretations. And we adopt knowledge distillation to improve the global interpretation results.

Knowledge distillation refers to the process of transferring the dark knowledge learned by a teacher model (usually sophisticated and large) to a student model (usually shallow and small). Dark Knowledge [67] [9] is the salient information hidden in the "soft targets": predicted probabilities for all classes, which are more informative than the "hard targets": predicted classes. One example for "soft targets" and "hard targets" is illustrated in Fig. 4.1. Maybe the pioneer work to distill the knowledge from a neural network into another algorithm is by Craven and Shavlik [38] who used a symbolic algorithm, the decision tree [130], to approximate the functions learned by a neural network with one hidden layer using hard targets.

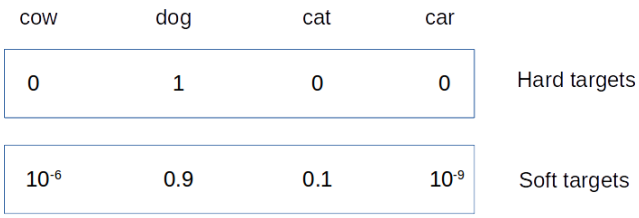


Figure 4.1: Examples of hard and soft targets.

Knowledge distillation originates from model compression [23]. In [23], the teacher model was built using the ensemble selection algorithm [25], which was then used to label unseen unlabeled data: the training data for the student model (also called the transfer data). This approach uses the hard targets produced by the teacher model. A following work [7] distills deep nets into shallow feed-forward nets adopting the method of "matching logits" (scores before the softmax activations), which would avoid the information loss when passing through logits to the probability space.

Then the concept of "knowledge distillation" was officially introduced in [67]. It is a more general solution to transfer knowledge from a cumbersome model to a compact model. There is a critical parameter that their method depends on: temperature. Remember in section 3.2.1 when we introduced a typical convolutional neural network, the last layer is the softmax layer. This layer has the function of converting logits into probabilities. it is calculated by the equation of (4.8) in section 4.3.2. This is essentially the temperature T set to 1 in the equation of (4.10). In their approach T is varied. They try to find an optimal temperature by raising the temperature of the final softmax layer of the teacher model until a suitable set of soft targets are generated. Then they apply the same temperature to the student model. They also proved that "matching logits" was actually a special case of their distillation approach.

Afterwards, a number of works followed such as [138] [172] [191], just to name a few. Most of these works concentrate on distilling complex and deep neural nets into simple and shallow neural nets; and are mainly applied for scenarios like edge computing hardware and on-the-fly training where there are memory, resource, power, time and space constraints, without significant loss in performance.

In our work, we employ knowledge distillation for another purpose: interpretation. We resolve the tension between interpretability and accuracy performance by distilling deep neural nets into conventional decision trees. This is a work in progress and as the first step of our attempts we apply the matching logits approach in [7]. The main obstacle to executing this plan is that for pure classification tasks there exist no logits in decision trees as in neural nets which could be used in the loss function. We address this issue by reformulating it into a multi-output regression problem [17] and achieve significant accuracy improvements (about 1% to 5%) on the experiments. Hence, the success of our approach establishes a new path for turning those inherently interpretable algorithms (which are highly interpretable, but worse in accuracy performance) into models attaining both accuracy and interpretability simultaneously.

4.2 Related Work

In the health care domain, two pipelines [30][31] are proposed to distill the knowledge from a DNN to Gradient Boosting Trees (GBTs) [52][53]. One of them extracts the logits from a learned DNN and uses the logits and the true labels of the original

training data to train a logistic regression algorithm to obtain the soft prediction scores. The next step is to train GBTs with the original training data's features and the soft predictions. The second pipeline directly applies the soft prediction scores of the trained DNN to the original training data as targets for training a mimic model with GBTs. However, GBTs lack transparency as they rely on post-hoc determinations: partial dependence [52], which would result in bias in this process [53]. The differences between their approach and ours are apparent. The strategy we applied when training the mimic model is matching logits, not the soft targets. Also, our student model is decision tree.

Another approach that distills neural networks into GBTs is in [162]. They tried two student models: tree-based generalized additive models (GA2Ms) [24][104][105] and GBTs. The teacher model they adopted is multilayer perceptrons. For the student model's training process, they applied the method of matching logits instead of soft targets in [30][31]. They investigated both classification and regression problems. However, their model is limited to binary class problems and their results are not conclusive and not yet published. Compared to their method, our teacher model is DNN and the student model is decision tree. We aim at multi-class classification problems.

Instead of doing post-hoc interpretations, [183] focuses on finding more interpretable neural networks during the training process. They created a new model complexity penalty function, tree regularization, to favor models whose decision boundaries could be well approximated by small decision trees. They measure human simulatability ("human simulation requires stepping through every calculation required to make a prediction" [183]) as the average decision path length and make the decision tree loss differentiable by adopting the technique of derivative-free optimization techniques [5]. Their experiments show that using tree regularization could achieve high accuracy at low complexity. Our method belongs to the post-hoc interpretations, which is different to what they proposed.

A more recent work [184] combines knowledge distillation and dimension reduction to visualize the results of deep classifiers. They pointed out that the method: t-distributed stochastic neighbor embedding (t-SNE) [108], commonly used for visualizing the activations of hidden layers, was problematic. Their experiments of

t-SNE show that points which actually lie near the decision boundary are presented well-separatd in t-SNE plots. This may mislead the analysts on the performances of classifers. They propose to visualize the data points that are assigned similar probability vectors to give practitioners a sense of how the decisions are made on test cases. They train a simpler and more interpretable classifier using the soft targets generated by a deep classifier. The student model they applied is Naive Bayes.

Perhaps the most related work is the model in [54] which uses a type of soft decision tree to mimic the input-output functions of a trained DNN. The reason they adopt a soft decision tree is that these trees can generate soft decisions to "facilitate easy distillation of the knowledge acquired by a deep neural net into a decision tree". In our case, we apply a different approach by using a multi-output regression model to facilitate distillation on a conventional decision tree. The way they design the soft decision tree is quite similar to [72].

4.3 Methodology

Rather than common approaches that distill DNN into shallow neural networks, we investigate distillation into non-neural nets. And the deep models we focus on are Convolutional Neural Networks (CNN). We first introduce some background information about decision trees and knowledge distillation and then describe our own methodology in detail.

4.3.1 CART for Regression

There are several versions of the decision tree algorithm. The earliest version: Iterative Dichotomiser 3 (ID3)[129] was proposed by Quinlan in 1986. It uses information gain as its attribute selection measure and requires features to be categorical. C4.5[130] is a successor of ID3 by Quinlan and the restrictions of ID3 on features are removed. Classification and Regression Trees (CART) [20] was introduced in 1984 by Breiman *et al.* Although CART and C4.5 were invented by different authors, they follow similar ideas for training decision trees. Owing to the reason that CART supports numerical target values (regression) and the key to our methodology is to solve a multi-output regression problem, we introduce briefly here the algorithm of CART. CART applies a greedy approach which constructs the decision tree in a top-down

recursive divide-and-conquer manner. As our experiments apply CART for regression, the descriptions focus on regression tasks.

This algorithm partitions the feature space and groups instances with the same labels together. Initially, it constructs a root node with all training samples S with features as $x_i \in R^n$ for $i = 1...l$ and labels as $y_i \in R^l$ and splits the node into two child nodes recursively. The splitting criterion is: $C = (a, t_n)$, where a is the attribute to split on and t_n is the threshold at node n. This criterion partitions S into

$$S_{left}(C) = (x, y)|x_a \leqslant t_n \tag{4.1}$$

$$S_{right}(C) = S \setminus S_{left}(C) \tag{4.2}$$

The impurity at node n is calculated with an impurity function I. For our regression task, we applied the Mean Squared Error method to calculate the impurity. Hence, I is calculated as

$$y'_n = \frac{1}{M_n} \sum_{i \in M_n} y_i \tag{4.3}$$

$$I(X_n) = \frac{1}{M_n} \sum_{i \in M_n} (y_i - y'_n)^2 \tag{4.4}$$

M_n is the number of instances in the corresponding child node. Hence, based on I, the impurity for both nodes can be expressed as

$$f(S, C) = \frac{M_{left}}{M_n} I(S_{left}(C)) + \frac{M_{right}}{M_n} I(S_{right}(C)) \tag{4.5}$$

Then the parameters in C could be optimized by minimizing $f(S, C)$

$$C^* = argmin_C f(S, C) \tag{4.6}$$

Thus, the optimal attribute and the splitting threshold are found. Then the algorithm recursively splits $S_{left}(C)$ and $S_{right}(C)$ until the maximum depth specified by the user is reached, a node becomes pure, $M_n < min_{samples}$ or $M_n = 1$.

4.3.2 Matching Logits

Knowledge distillation transfers the generalization ability of a complex teacher model to a simple student model. Using the teacher model's soft targets for distillation could produce a much better outcome than hard targets, which was proven by the experiment results in [7]. Fig. 4.1 shows an example of hard and soft targets. Hard targets just contain the information for the predict label while soft targets reveal all the predicted probabilities for all the classes. Many previous works [38] [23] [133] just adopt the hard targets (the predicted labels of the teacher model) for distillation, where soft targets could as a matter of fact boost the results significantly.

When we examine Fig. 4.1 closely, we notice that the probabilities for "cow" and "car" are much smaller than those for "dog" and "cat". When training student models applying the cross-entropy cost function, these much smaller probabilities would vanish to zero. Take CNN for example, the last hidden layer l before the softmax layer is a fully connected layer with logits z as the output

$$z_i = \sum_j W_{ij} x_j^{l-1} + b_i \tag{4.7}$$

Here z_i is the logit for one of the classes: i. j is the number of hidden nodes for layer $l-1$. W and b are weights and bias respectively. The softmax layer calculates the output probabilities for each class as

$$q_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{4.8}$$

The cross-entropy function is then applied to calculate the loss of the model

$$H_p(q) = -\sum_i p_i log(q_i) \tag{4.9}$$

Hence, to avoid the loss of information, it is desirable to use logits z as per equation (4.7) instead of the predicted probabilities q. This method is called "matching logits" and the pioneer work was done in [7]. Hinton *et al.* [67] extended their work to a more

general case by inserting a temperature term T into (4.8)

$$q_i = \frac{e^{\frac{z_i}{T}}}{\sum_j e^{\frac{z_j}{T}}} \tag{4.10}$$

and they demonstrated mathematically that in the high temperature limit and when the logits were zero-meaned separately for each training instance of the student model, matching logits was a special case of using the soft targets for distillation. They proved it by performing gradient descent on the cross-entropy function

$$\frac{\partial H}{\partial z_i} \approx \frac{1}{NT^2}(z_i - v_i) \tag{4.11}$$

Here v_i is the logit of the student model for training instance i, N is the number of instances of the training data for the student model. For more elaborated derivations, please refer to [67].

4.3.3 Distilling CNN into Decision Trees

In this work, as the first step of our attempts, we employ the matching logits method when distilling CNN into conventional decision trees. Fig. 4.2 illustrates the framework of our method. In this figure, the architecture of the CNN is the one used to train the MNIST data as in [102]. It comprises two convolutional layers and two pooling layers followed by two fully connected layers: fc1 and fc2. After this deep CNN is trained, we feed the feature part X of the original training data into the trained model to obtain the corresponding logits Z. Then we train CART with X and Z which are treated as the targets.

However, here arise some problems for deployment. First, for classification tasks, the targets are limited to categorical, not numerical and continuous values. We can resolve this by treating it as a regression problem. Second, even for regression tasks, most algorithms only support single-output regressions. For multi-class datasets, this is actually a multi-output regression problem [17]. The whole point of using multi-output regression is to facilitate distillation. As we mentioned at the end of section 4.2, since the student model learns from the soft targets of the teacher model, we need to adopt multi-output regression to facilitate this learning process.

We apply the algorithm adaptation method, where we use decision trees to directly handle multi-output data sets simultaneously. This is anticipated to produce much better results than the problem transformation method which transforms the multi-output regression problem into independent single-output problems which are then solved by single-output regression algorithms. This is due to the fact that problem transformation methods do not consider the dependencies among the targets.

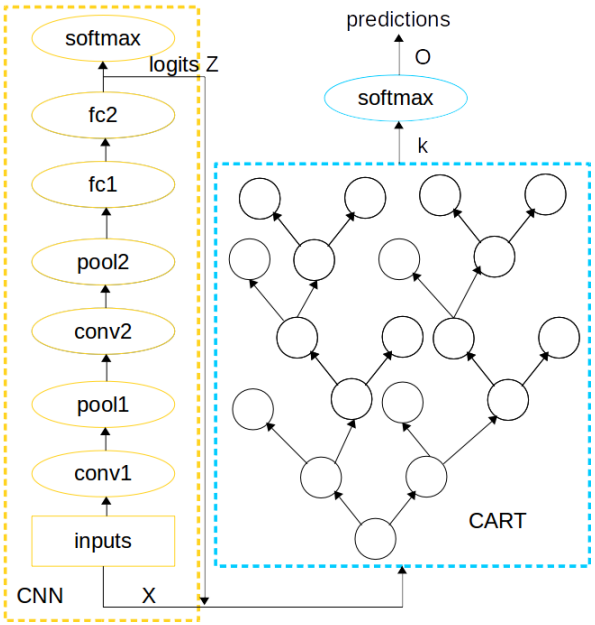


Figure 4.2: Framework of our method.

So the key novelty of this chapter is that we treat the problem at hand as a multi-output regression problem first and then try to translate the regression results to achieve the goal of classification. Hence, the regression data for CART should have features as X with $x_i \in R^n$ for $i = 1...l$ and labels as Z with $z_i \in R^l$. And the impurity function I in CART is calculated as

$$z'_n = \frac{1}{M_n} \sum_{i \in M_n} z_i \tag{4.12}$$

$$I(X_n) = \frac{1}{M_n} \sum_{i \in M_n} (z_i - z'_n)^2 \tag{4.13}$$

Once CART is trained, in order to obtain the final prediction results on test cases we need to add a softmax layer over the test results of CART to turn numerical test

results into categorical ones. Assuming the test results on CART is k, the final output probabilities for class i therefore is

$$O_i = \frac{e^{k_i}}{\sum_j e^{k_j}} \tag{4.14}$$

4.4 Experiments

We performed the experiments on two datasets to demonstrate the effectiveness of our distillation approach. All teacher models are implemented on the TensorFlow [1] platform to make them scalable to big datasets.

4.4.1 Datasets

The two datasets we selected are the MNIST dataset [95] and the Connect-4 dataset from the UCI repository [41]. MNIST is a famous benchmark dataset for deep learning. It contains the pixel values of handwritten digits from 0 to 9. Each instance is a 28×28 grayscale image and contains 784 features when flattened into a one dimensional space. The Connect-4 dataset stores the information about the two players' positions for the the game of connect-4. It has a seven-column, six-row vertically suspended grid. There are two players and each spot on the grid represents whether it has been taken by the first player, or the second player or left blank. The classes are the outcome for the first player. Details of these datasets could be found in Table 4.1.

Table 4.1: Datasets

	Dataset Details			
	#Features	***#Train***	***#Test***	***Labels***
MNIST	784	55,000	10,000	0-9
Connect-4	42	57,557	10,000	win, loss, draw

4.4.2 Experimental Setup

The deep learning model we applied to train the MNIST dataset is a deep CNN which has an architecture of two convolutional layers followed by two fully connected

layers. The parameter settings for this network are depicted in Table 4.2. The first convolutional layer uses filters with window size 5×5, $stride = 1$ and the 'same' padding in TensorFlow. When the stride length is 1, 'same' padding generates a feature map with the same size as the input image. This stage produces 32 feature maps each with size 28×28. The following max pooling layer over 2×2 blocks with $stride = 2$ generates 32 feature maps with size 14×14. The parameter settings for the second convolutional layer and pooling layer are the same as the previous one except that this stage generates 64 feature maps. Hence, we have 64 feature maps each with size 7×7. Then we flatten these $7 \times 7 \times 64$ features into a one dimensional list and then apply a fully connected layer: fc1 with 1024 hidden nodes. Immediately after fc1 is the dropout [155] layer, where we set the dropout rate as 0.5. The second fully connected layer: fc2 is the output layer with 10 hidden nodes, each representing one of the 0-9 digits. These outputs are also the logits of this model.

The Connect-4 dataset has a class distribution of win (65.83%), loss (24.62%) and draw (9.55%). We randomly sample 10,000 test instances which satisfy the original class distributions. The algorithm we applied to train the Connect-4 dataset is a multilayer perceptron (MLP) with parameter settings in Table 4.3. It has three hidden layers, the first hidden layer with 256 hidden nodes, the second hidden layer with 128 hidden nodes, the third hidden layer also with 128 hidden nodes and the output layer with 3 nodes representing the three outcomes of the connect-4 game. We also apply the dropout rate after each of the hidden layers and the value is set as 0.8. When calculating the training loss, in addition to TensorFlow's own cross entropy function, we also added an L2 penalty (regularization term) parameter as in Python's scikit-learn machine learning tool. This penalty parameter is set as 0.0001 which could help to improve the MLP's performance.

Table 4.2: Parameter settings for MNIST

	Network Type: CNN				
	conv:filter	***conv:stride***	***pool:block***	***pool:stride***	***fc1***
MNIST	5×5	1	2×2	2	1024

Table 4.3: Parameter settings for Connect-4

	Network Type: MLP				
	1st hidden	***2nd hidden***	***3rd hidden***	***out***	***dropout***
Connect-4	256	128	128	3	0.8

4.4.3 Experimental Results

For decision tree classifications, we apply the modules in the scikit-learn machine learning tool. When we are performing classification tasks applying a decision tree, there are a variety of parameters to tune such as the minimum number of samples per leaf, the strategy used to choose the split at each node (either the best split or the best random split) and so on. We select two parameters that would influence the performance of a decision tree substantially: the maximum depth of the tree and the functions to measure the impurity of a split (either "gini" or "entropy"). The other parameters are left as default values as in scikit-learn.

For the MNIST dataset, the teacher CNN model achieves an accuracy of 99.25%. The performance for the student model and the conventional decision tree classification results are shown in Table 4.4. "Acc_student" represents the accuracy of the student decision tree trained using the logits of the teacher CNN model on TensorFlow.

Table 4.4: Test Accuracy Results for MNIST

Tree	**Methods**		
Depth	***Acc_ student***	***Acc_ gini***	***Acc_ entropy***
6	**0.7119**	0.6644	0.6849
7	**0.7685**	0.7534	0.7228
8	**0.8125**	0.7914	0.8007
9	**0.8512**	0.8151	0.8304
10	**0.8655**	0.8445	0.8450

"Acc_gini" is the accuracy of the decision tree without distillation when the impurity measure is "gini" in scikit-learn when trained utilizing the same training and test data as the CNN model. "Acc_entropy" is the classification accuracy of the decision tree when the impurity measure is "entropy". We highlighted the best performance in bold.

Under different tree depths, the student model always outperforms the conventional decision tree. The same conclusion holds true for the Connect-4 dataset in Table 4.5 where the accuracy for the MLP teacher model is 86.62%. The reason we limit the tree depth to 10 is that we would like to construct interpretable models, and trees over a depth of 10 become extremely hard for human cognition to comprehend. We also illustrate these results in graphs in Fig. 4.3 and Fig. 4.4 to present the results more intuitively.

Table 4.5: Test Accuracy Results for Connect-4

Tree	**Methods**		
Depth	***Acc_ student***	***Acc_ gini***	***Acc_ entropy***
6	**0.6943**	0.6816	0.6835
7	**0.6999**	0.6919	0.6832
8	**0.7070**	0.6750	0.6625
9	**0.7230**	0.6927	0.6974
10	**0.7342**	0.7044	0.7006

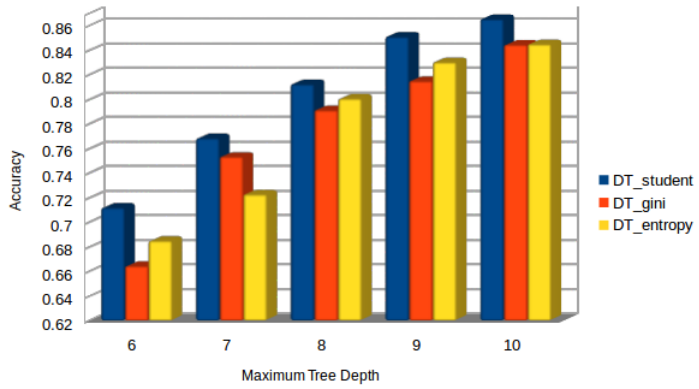


Figure 4.3: Distillation results for MNIST.

4.4.4 Discussion

For the Connect-4 dataset, although we can fine tune the parameters of the teacher model or switch the teacher model to CNN to improve the teacher model's performance, the distillation effect still relies largely on the student model's own generalization ability. For instance, the teacher model for the MNIST dataset already has a very

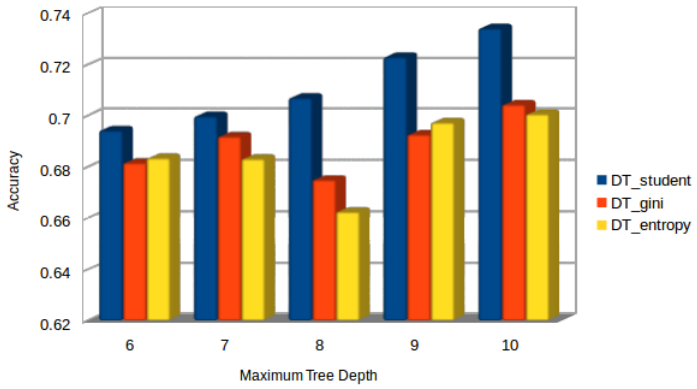


Figure 4.4: Distillation results for Connect-4.

high accuracy of 99.25%, but the student model's highest accuracy in Table 4.2 is only 86.55%. However, from our experiments we found that training a good teacher model indeed helped to boost the distillation results. In our experiments, we notice that distillation helps to improve the accuracy by 1% to 5%. Hence, there is still a long way to go for the student model to match the results of the teacher model.

We are also curious about the performance of the student models and the conventional decision trees when the maximum depth of the tree is not specified. In this situation, for the MNIST dataset, we found that the accuracy for the student model was 88.28% and the decision tree classification achieved 87.4% for the criterion of "gini". For the Connect-4 dataset, when the teacher model has an accuracy of 83.22% the student model achieves 79.06% and conventional decision tree has 77.57% when using "gini" as the impurity measure. We notice that the accuracy improvements are smaller than in the cases where the depth of the trees are specified. This is easy to explain as when the tree levels are not set the conventional decision tree takes much deeper tree levels than the student model to arrive at the current accuracy results. Hence these decision trees are far less interpretable than the student models because the level of tree depth determines the interpretability for decision trees. After all, in our experiments we already demonstrated that under the same tree level, the conventional decision tree performs worse than the student models.

4.5 Conclusions and limitations

Based on the fact that inherently interpretable algorithms perform worse than some non-interpretable algorithms such as the deep learning algorithm, this chapter presents an approach to improve the accuracy performance of an inherently interpretable algorithm: decision tree. This is achieved by utilizing the dark knowledge hidden in the soft predictions of DNN. We apply the matching logits method which employs the logits of DNN for training student decision tree models. Experiments on two datasets, MNIST and Connect-4, demonstrate significant improvements on the accuracy of the distilled student model over conventional decision trees.

The most valuable contribution of our approach in this chapter is that we improved the accuracy of the surrogate model, which is one of the three pillars for developing a good interpretation method (in chapter 1.2.1 on the evaluations of interpretation methods). At the same time, this method also meets the goal of improving fidelity and comprehensibility. Fidelity is improved by applying the teacher student training strategy. And comprehensibility is achieved by choosing an inherently interpretable algorithm: decision tree.

Although we used image data in the experiments to prove the efficacy of our approach (the accuracy of student models are improved), decision tree is more suitable to interpret datasets which have meanings on their own features. For image data, each feature is just a small piece of pixels of the entire image and visualization seems to be a better option to interpret the classification process on the image data. Hence, there are two reasons we are not showing the structure of the produced tree in this paper: (1) the data is enormous and the tree could hence be huge; (2) it seems to be meaningless to interpret an image using a tree structure pixel by pixel. Hence, one of the differences between the approaches of chapter 3 and chapter 4 is that chapter 3 directly presents visual interpretations while chapter 4 strives to improve the interpretation method's accuracy. In sum, our approach would work best on datasets which have meaningful features themselves and could be reasoned about, e.g. emails. For image datasets, to apply this approach we should adapt the surrogate model so that it could present the visualization on each level of the tree, which could be a future research direction.

Chapter 5

QDV: Refining Deep Neural Networks with Quantified Interpretability

In this chapter, we demonstrate a way to refine the structure of a neural network model. Current research (including our own research in chapters 3 and 4) on interpretability tends to focus on building interpretable models for highly non-interpretable neural nets. Little work has been done on employing interpretability for refining models. We propose a new method, Quantified Data Visualization (QDV), to leverage interpretability for refining deep neural nets. The key novelty of this method is that we combine the methods of transfer learning and t-SNE to achieve our goal. Although transfer learning and t-SNE are mature and successful techniques in their separate fields, to the best of our knowledge, we are the first to combine them for the purpose of refining deep neural nets. Our experiments show empirically why VGG19 has better classification accuracy than Alexnet on the CIFAR-10 dataset through quantitative and qualitative analysis on each of their hidden layers. This approach could be applied to refine the architectures of deep neural nets when their parameters are altered and adjusted. Compared with a previous approach, which mainly applies the method of neural feature visualization, we are able to show and explain not only qualitatively but also quantitatively why one model has higher accuracy than another one. Compared with Knowledge Distillation, we directly interpret a complex neural net through a "Warm Restart" via a simpler dataset, without distilling the complex model into a shallow one.

5.1 Introduction

Discussing interpretability, we should be cautious that the meaning is two-fold: one from the perspective of the end users and the other from the perspective of the model designers, which demands different explanations and measures of efficacy. For end users, it is mainly employed to illustrate predictions in unforeseen circumstances and

build a sense of trust. For model designers, it is useful to diagnose and refine the models [146]. Current research [133, 93, 164, 114, 102] tends to focus on learning interpretable models, but these models are seldom leveraged to help diagnose [189] and refine the non-interpretable complex models. In this chapter, we demonstrate from the angle of the model designers how interpretability could help to improve a model's accuracy to refine neural nets.

The research most relevant to ours is the neural feature visualization approach proposed in [187] that employs visual interpretations to diagnose the problems of an already existing deep learning model, Alexnet [88], to refine it. They utilize a multi-layered Deconvolutional Network (Deconvnet) [188] (initially designed for unsupervised learning), which maps the feature activities back to the input pixel space and finds the optimal stimulus at any hidden layer in the model. The structure of Deconvnet is shown previously in Fig. 2.4. After visualizing the first and second hidden layers of the Alexnet, they reduced the filter size of the first hidden layer from 11×11 to 7×7 and the stride of convolution from 4 to 2. The resulting model outperforms the architecture of Alexnet for their single models by 1.7% (test top-5).

However, the justification/intuition for the choice of smaller filters just relies on qualitative comparisons, which is shown in Fig. 5.1. They changed the parameters of the first hidden layer and stated (and we quote here) "the smaller stride (2 vs 4) and filter size (7x7 vs 11x11) results in more distinctive features and fewer "dead" features." However, when we examine the corresponding visualizations in the first hidden layer in Fig. 5.1, we don't see much difference visually. In this chapter, we propose a new method, Quantified Data Visualization (QDV), to quantitatively measure the visualizations on each hidden layer for the sake of measuring more accurately the impacts of parameter variations.

The contributions of this chapter are as follows:

- The main purpose of this paper is to refine neural network models more convincingly via interpretability.
- Besides the definition of interpretability, a much harder task is to quantify and measure it. In this paper, we propose a way to quantity interpretation.
- To the best of our knowledge, we are the first to interpret the models trained on

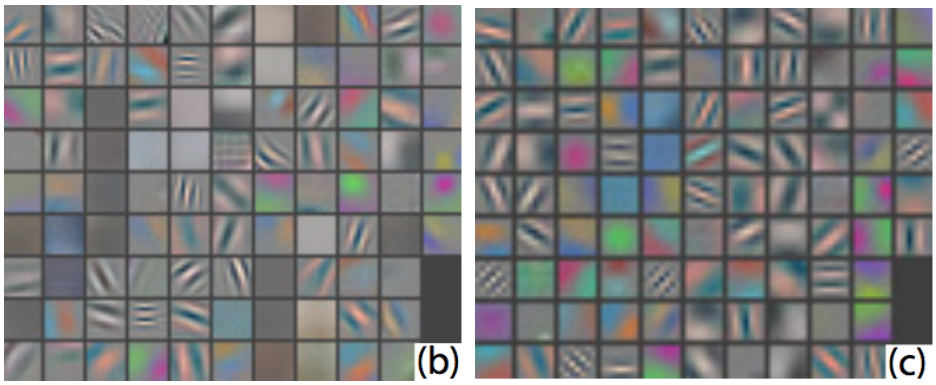


Figure 5.1: Applying Deconvnet on the 1st layer features for Alexnet (b) and modified Alexnet (c) [187]. However, we don't observe much differences between (b) and (c) visually.

the CIFAR-10 dataset when applying Alexnet and VGG19 as "Warm Restart".

5.2 Method

Neural feature visualization has the advantage of showing intuitively what information a neural net relies on to make a specific decision. For instance, in figure 2.5(c) and (d) taken from [187], for the results in layer 5, it is noticeable that when just examining the image patches of layer 5 it seems that they have nothing in common. But after examining its feature visualization results we realize that it detects the grass in the background.

However, the main disadvantage of this approach is the question of how to measure the interpretability quantitatively (to what extent one interpretation is better than another), which is also a common problem for other visualization techniques. Once we have the quantitative information, we are able to evaluate more convincingly whether one neural network structure is better than another. For this reason, in the following sections we develop and present our Quantified Data Visualization (QDV) method for quantifying interpretability.

5.2.1 QDV

The neural nets that we are interested in studying are the advanced models trained on the ImageNet dataset [141]. This dataset has 1000 classes and its two-dimensional

embedding for the hidden layer "fc7" of the Alexnet is computed and presented in [76]. However, with 1000 classes to be shown all at once in the same two-dimensional space, it is very hard to understand the interpretation. Therefore, to take advantage of human visual ability, in our model, we choose a benchmark dataset C which has only 10 classes. In order to compare the performances of sophisticated neural nets which were pre-trained on the ImageNet data, we apply transfer learning [60] on the dataset C. In spite of the diverse applications of transfer learning, here we leverage it for "Warm Restart": we use the trained models on the ImageNet dataset to fine tune the algorithm on the dataset C. On the new model M_n, as the lower layers are already trained on recognizing shapes and sizes, we just need to refine the upper layers on the dataset C. In this chapter, the technique of "Warm Restart" are applied on Alexnet and VGG19 neural nets. The pseudo code of our algorithm is shown in Algorithm 2.

Algorithm 2 QDV

1: Input: training dataset C_{tr}; test dataset C_{te}; dimension reduction parameters: perplexity $perp$, iterations T, learning rate η, momentum $\alpha(t)$; pre-trained neural net model: M.
2: $C_{tr} \xrightarrow{resize} (C_{tr})^r$
3: $M_n = M((C_{tr})^r)$
4: $C_{te} \xrightarrow{resize} (C_{te})^r$
5: $H = M_n((C_{te})^r)$
6: **for** hidden activations $h_i = h_1 \cdots h_H$ **do**
7: compute $p_{j|i}$ with $perp$
8: set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
9: Initialize $Y^{(0)}$
10: **for** $t = \{1, 2, \cdots T\}$ **do**
11: compute q_{ij} in d-dimensional space
12: compute gradient $\frac{dL}{dy}$
13: set $Y^{(t)} = Y^{(t-1)} + \eta\frac{dL}{dy} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$
14: **end for**
15: compute AR for Y
16: **end for**
17: Output: $\{AR_1, AR_2, \cdots, AR_H\}$

In this algorithm, after setting all the input parameters (Step 1), we resize the training dataset C_{tr} (Step 2). This is due to the reason that the input image size for the pre-trained model is different from the training dataset C_{tr}. Then we apply the resized training dataset $(C_{tr})^r$ to finetune the pretrained model M. The new model

generated is denoted as M_n (Step 3). We then apply the resized test dataset$(C_{te})^r$ on the newly generated model M_n, which produces the m-dimensional activation values on all the H hidden layers (Steps 4-5). Then for the hidden activation values h_i at each hidden layer, we first compute the conditional probability $p_{j|i}$ which is then applied to calculate the joint probability p_{ij} in the m-dimensional space (Steps 6-8). The mathematical reasons behind these are followed in the section Quantifying Visualizations. The low dimensional representation is then calculated iteratively within T iterations employing the derived gradient in equation (5.10) (Steps 9-14). Finally, the agreement ratios are calculated for each of the hidden layers (Steps 15-17).

We also demonstrate QDV's architecture in Fig. 5.2. In this figure, we present our method in the case where we apply "Warm Restart" on the pre-trained Alexnet which has five hidden layers. In this architecture, the ImageNet data is first applied to train the Alexnet and then the pre-trained model M is employed as the "Warm Restart" model. Then a new dataset C is applied on M to fine tune the parameters generating a new model M_n (retrained model). With the retrained model, we could then calculate the test accuracy. In this process, the blue blocks in Fig. 5.2 represent the portions that the two models, M and M_n, share. The green blocks are the fully connected layers, which are different for the two models because the ImageNet dataset has 1000 classes while the new dataset has only 10 classes. After we obtain the newly trained model M_n, we extract the hidden activation values from all of the five hidden layers by testing M_n with $(C_{te})^r$. In the end, we acquire the agreement ratios AR on the two-dimensional representations Y of X. AR is then the indication of quantified interpretations.

Quantifying Visualizations Let $X = \{x_1, x_2, ..., x_n\}$ be the set of values of the hidden activations of a specific hidden layer of a neural network and x_i be the m-dimensional value of the hidden activation of the ith hidden node, n is the number of hidden nodes at this hidden layer. We try to find a set of points $Y = \{y_1, y_2, ..., y_n\}$ in a d-dimensional space with $d = 2$. Let $\delta(x_i, x_j)$ represent the dissimilarity measure between two hidden nodes in the m-dimensional space and $d(y_i, y_j)$ be the distances of two points in the d-dimensional space. We apply a dimension reduction approach: f

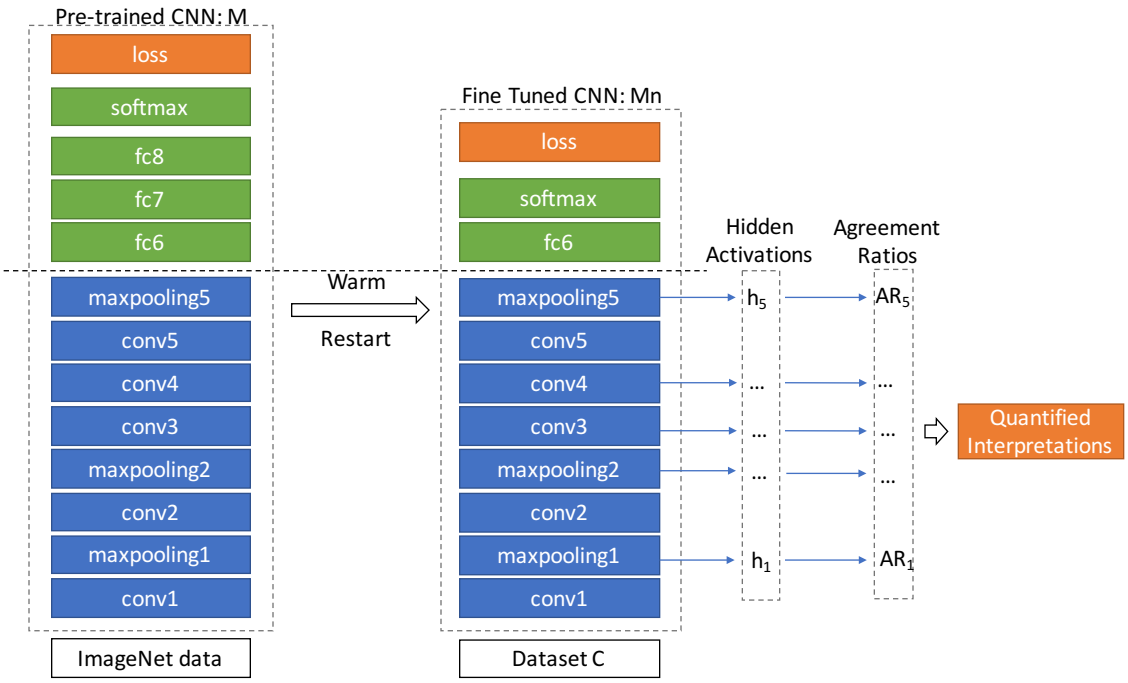


Figure 5.2: Architecture of QDV when applying "Warm Restart" on the pre-trained Alexnet.

on X to obtain its d-dimensional representations Y

$$f : X \to Y \tag{5.1}$$

So that

$$|\delta(x_i, x_j) - d(f(x_i), f(x_j))| \to 0 \qquad \forall x_i, x_j \in X \tag{5.2}$$

Instead of using high-dimensional Euclidean distances between two points to represent the similarities between two data points, we employ the concepts of conditional probabilities of SNE [68]. For hidden nodes x_i and x_j, $p_{j|i}$ stands for the conditional probability that hidden node x_i would choose x_j as its neighbour. This probability conforms to the Gaussian probability density centered at x_i. The nearer x_j to x_i, the higher the probability $p_{j|i}$. Therefore, we can express $p_{j|i}$ as

$$p_{j|i} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \tag{5.3}$$

Here σ_i is the variance of the Gaussian distribution. Meanwhile, each m-dimensional

hidden node x_i will have a corresponding d-dimensional counterpart y_i. The similarity between y_i and y_j could also be modeled as a probability $q_{j|i}$. When we set the variance of the probability distribution of $q_{j|i}$ as $\frac{1}{\sqrt{2}}$

$$q_{j|i} = \frac{exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} exp(-\|y_i - y_k\|^2)} \tag{5.4}$$

In order to attain a good representation in the d-dimensional space, we need to minimize the mismatch between the probabilities $p_{j|i}$ and $q_{j|i}$. According to (5.2), we use the Kullback-Leibler divergence to measure whether $q_{j|i}$ is faithful to $p_{j|i}$. It has a form of the additions of cross-entropy as follows [68]

$$L = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{5.5}$$

This is also the cost function. In t-SNE, by using joint probability distribution P and Q in high-dimensional and low-dimensional spaces respectively, the cost function could be reformulated as

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{5.6}$$

Then the gradient descent method is used to find the optimal solution. If we set p_{ij} and q_{ij} as joint probabilities in the high-dimensional and low-dimensional spaces respectively, p_{ij} could be expressed as [108]

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{5.7}$$

Employing the Student t-distribution with one degree of freedom in the low-dimensional map, the joint probability q_{ij} could be expressed as

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{(-1)}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{(-1)}} \tag{5.8}$$

When

$$p_{ij} = p_{ji}$$
$$q_{ij} = q_{ji} \tag{5.9}$$

the gradient $\frac{dL}{dy}$ could be derived as

$$\frac{dL}{dy_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1} \tag{5.10}$$

To evaluate the optimal solution, we need to measure its information loss. A straightforward way is to calculate "stress" [90]. It measures the differences of distances between two points before and after dimension reduction, which could be expressed as

$$stress = \sqrt{\frac{\sum_{i<j}(\delta(x_i, x_j) - d(f(x_i), f(x_j)))^2}{\sum_{i<j} d(f(x_i), f(x_j))^2}} \tag{5.11}$$

However, it is observed that sometimes the calculated values of stress don't match the actual projections: better projections are assigned worse stress [126].

Therefore, we propose to leverage the information of the nearest neighbors of a point in the d-dimensional space. To search for the nearest neighbors, there are two major techniques: pivot-based algorithms and clustering techniques [29]. Here we apply a different one which we name Agreement Ratio (AR), which was previously known as neighbourhood preservation [126]. This could generate quantitative values about the interpretations on each hidden layer of a neural net.

For a specific point y_i in the d-dimensional space, we select its k nearest neighbours and calculate its AR: AR_{y_i} as the ratio of the number of points belonging to the same class c as y_i. Let N be the total number of nearest neighbors of y_i and N_c the nearest neighbors which have the same class label as y_i. Then AR_{y_i} could be expressed as

$$AR_{y_i} = \left.\frac{N_c}{N}\right|_k \tag{5.12}$$

The AR for all the points of the projection in the d-dimensional space is the average of AR over all the points in the d-dimensional space. This is the value that we adopt to evaluate interpretability, and is denoted as

$$AR = \frac{\sum_i AR_{y_i}}{n} \tag{5.13}$$

5.2.2 Comparison with Knowledge Distillation

Another related research to ours is Knowledge Distillation [67] [54] [164] [101], which refers to the process of distilling the dark knowledge learned by a teacher model (usually sophisticated and large) to a student model (usually shallow and small). The distillation model we designed in [101] is illustrated in Fig. 4.2.

However, this method sacrifices accuracy for interpretability. In our experiment, the teacher model achieves an accuracy of 99.25% on the MNIST dataset whereas the best accuracy the student model (a conventional decision tree) obtains is only 86.55%.

When we compare the two frameworks, QDV in Fig. 5.2 and knowledge distillation in Fig. 4.2, different from the Knowledge Distillation technique, our approach QDV interprets the deep neural nets directly through "Warm Restart" without distilling it into a separate shallow neural network.

5.3 Experiments

Experimental Setup We apply our method QDV to interpret two deep neural nets: Alexnet [88] and VGG19 [150]. Both of them are pre-trained on the ImageNet dataset [141]. We then use these pre-trained models as "Warm Restart" and apply the CIFAR-10 dataset to fine tune these models. it is noticeable that the image size for the ImageNet dataset is $224 \times 224 \times 3$ while it is $32 \times 32 \times 3$ for the CIFAR-10 dataset. Hence, we resized the image size of the CIFAR-10 dataset to $224 \times 224 \times 3$ to fit the pre-trained neural nets. The details for the CIFAR-10 dataset is shown in Table 5.1 and the parameters used for fine tuning the pre-trained models are displayed in Table 5.2. The experiments are executed on the platform of Google Colaboratory [14] on a single GPU.

When fine-tuning the Alexnet, instead of applying the two-GPU net in the original Alexnet paper[88], we adapt it to the one-GPU net. The complete order of the hidden layers is "conv1-relu1-lrn1-maxpool1-conv2-relu2-lrn2-maxpool2-conv3-relu3-conv4-relu4-conv5-relu5-maxpool5-fc6-fc7-fc8". "conv" denotes the convolutional layer. "relu" is Rectified Linear Units. "lrn" stands for Local Response Normalization[88] and "fc"

Table 5.1: CIFAR-10 Dataset

Dataset Details			
#Features	#Train	#Test	Classes
$32 \times 32 \times 3$	50,000	10,000	10

Table 5.2: Parameter settings for fine tuning

Alexnet & VGG19		
learning rate	batch size	#epochs
0.00001	16	10

is fully connected layers. All convolutional layers are followed by ReLUs and only the first two convolutional layers are followed by lrn. The structural details and settings of Alexnet is shown in Table 5.3. In this table, only layers before fully connected layers are shown. Since the image sizes after going through relu and lrn are unchanged, we just list the output sizes of convolutional layers and maxpooling layers. In Table 5.3, "SAME" and "padding" represent for different padding algorithms. Especially, When "SAME" is selected and the stride size is 1, the layer's outputs will have the same spatial dimensions as its inputs.

The structure of VGG19 is similar to Alexnet, but consists of multiple convolutional layers in each of the convolutional blocks. The stacks of convolutional layers all have filters with kernel size 3×3, stride=1 and "SAME" padding. For maxpooling layers, they all share the same 2×2 kernel with stride=2. The brief network structure of VGG19 could be expressed as "conv1(2)-maxpool1-conv2(2)-maxpool2-conv3(4)-maxpool3-conv4(4)-maxpool4-conv5(4)-maxpool5-fc6-fc7-fc8". The numbers in the brackets that follow each covolutional layer are the numbers of stacks of convolutional layers. The detailed parameter settings for VGG19 are displayed in Table 5.4.

Experimental Results The program takes around 6-7 mins for training for one epoch when using Alexnet as "Warm Restart" and processes about 7-8 batches/s with a batch size of 16. For VGG19, one epoch takes about 12-13 mins and the program processes about 4 batches/s. In order to obtain quantitative visualization results, we randomly subsampled 1000 test instances guaranteeing a balanced distribution

Table 5.3: Alexnet's Structure before fully connected layers

Alexnet				
Layer	**Stride**	**Padding**	**Kernel**	**Output**
conv1	4	"SAME"	$11 \times 11 \times 96$	$56 \times 56 \times 96$
maxpool1	2	"VALID"	3×3	$27 \times 27 \times 96$
conv2	1	"SAME"	$5 \times 5 \times 256$	$27 \times 27 \times 256$
maxpool2	2	"VALID"	3×3	$13 \times 13 \times 256$
conv3	1	"SAME"	$3 \times 3 \times 384$	$13 \times 13 \times 384$
conv4	1	"SAME"	$3 \times 3 \times 384$	$13 \times 13 \times 384$
conv5	1	"SAME"	$3 \times 3 \times 256$	$13 \times 13 \times 256$
maxpool5	2	"VALID"	3×3	$6 \times 6 \times 256$

Table 5.4: VGG19's Structure before fully connected layers

VGG19				
Layer	**Stride**	**Padding**	**Kernel**	**Output**
conv1	1	"SAME"	$3 \times 3 \times 64$	$224 \times 224 \times 64$
	1	"SAME"	$3 \times 3 \times 64$	$224 \times 224 \times 64$
maxpool1	2	"VALID"	2×2	$112 \times 112 \times 64$
conv2	1	"SAME"	$3 \times 3 \times 128$	$112 \times 112 \times 128$
	1	"SAME"	$3 \times 3 \times 128$	$112 \times 112 \times 128$
maxpool2	2	"VALID"	2×2	$56 \times 56 \times 128$
conv3	1	"SAME"	$3 \times 3 \times 256$	$56 \times 56 \times 256$
	1	"SAME"	$3 \times 3 \times 256$	$56 \times 56 \times 256$
	1	"SAME"	$3 \times 3 \times 256$	$56 \times 56 \times 256$
	1	"SAME"	$3 \times 3 \times 256$	$56 \times 56 \times 256$
maxpool3	2	"VALID"	2×2	$28 \times 28 \times 256$
conv4	1	"SAME"	$3 \times 3 \times 512$	$28 \times 28 \times 512$
	1	"SAME"	$3 \times 3 \times 512$	$28 \times 28 \times 512$
	1	"SAME"	$3 \times 3 \times 512$	$28 \times 28 \times 512$
	1	"SAME"	$3 \times 3 \times 512$	$28 \times 28 \times 512$
maxpool4	2	"VALID"	2×2	$14 \times 14 \times 512$
conv5	1	"SAME"	$3 \times 3 \times 512$	$14 \times 14 \times 512$
	1	"SAME"	$3 \times 3 \times 512$	$14 \times 14 \times 512$
	1	"SAME"	$3 \times 3 \times 512$	$14 \times 14 \times 512$
	1	"SAME"	$3 \times 3 \times 512$	$14 \times 14 \times 512$
maxpool5	2	"VALID"	2×2	$7 \times 7 \times 512$

of classes (each class has 100 instances) from the original 10,000 test instances. The test accuracy on the new model M_n is 79.6% for pre-trained Alexnet and 89.4% for VGG19. In our experiments, the accuracies could be further improved by increasing the number of epochs.

We then extracted the values of the hidden activations for these test instances corresponding to each of the hidden layers. The two dimensional visualizations of the hidden activations generated by QDV for both neural nets are shown in Fig. 5.3. The work that most relevant to ours is the one proposed in [131]. One big difference between their work and ours is that we applied warm restart on Alexnet and VGG19 and strive for quantitative interpretations for refining neural nets. In these figures, we also show the quantitative results: AR generated by our algorithm QDV for both cases. It is noticeable that for the fifth layer (maxpool5), we observe more clearer patterns of clusters when using VGG19 as pre-trained model. Hence, in this layer we conclude that VGG19 demonstrates better ability of separating classes than Alexnet. However, for the first four hidden layers for both neural nets, if we just inspect the two dimensional visualizations it is more difficult to figure out which neural net is better at learning representations. Fortunately, the AR values attached to each of the visualizations demonstrates that VGG19 is better than Alexnet on all the hidden layers. The AR values are also listed in Table 5.5. Comparing the corresponding AR values, we notice that VGG19 has higher AR values than Alexnet. The discrepancy is most apparent in the fifth hidden layer, which also matches the visualization results in this layer.

Discussion The quantitative results generated by the QDV method indicate that the VGG19 structure leads to better performance (in terms of model accuracy) than Alexnet. This provides stronger proof than just applying the Deconvnet approach, which mainly relies on qualitative interpretation. By applying "Warm Restart" on pre-trained neural nets, we can also avoid the performance loss of distilling a complex model (teacher) into a simpler one (student) [101]. Also, applying this method on a simpler dataset (CIFAR-10) enables clearer visualizations (due to smaller classes) than those on the original ImageNet data (1000 classes) [76].

Due to the resource constrains on one GPU of Google Colaboratory which only

Table 5.5: AR values at each hidden layers of newly trained models M_n: higher values implies better representations learned

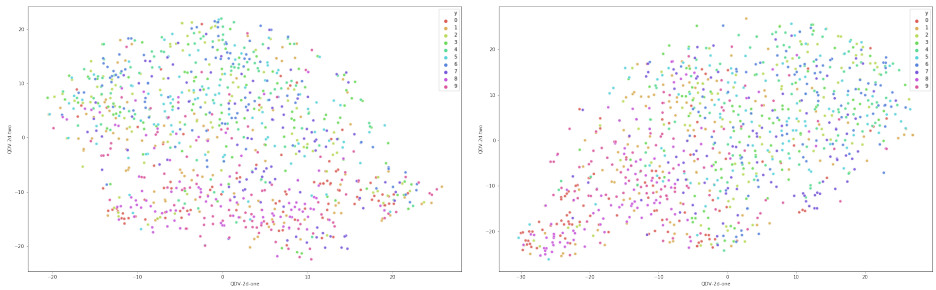
hidden layers	Alexnet	VGG19
1st	0.21	**0.22**
2nd	0.24	**0.25**
3rd	0.29	**0.34**
4th	0.35	**0.42**
5th	0.46	**0.80**

supports 12 hours of consistent runtimes, we limited our training epochs to 10 and the test samples to 1000. Because even for test instances of just 1000, the disk size for the activations values of the first layer for the VGG19 pre-trained model is 3.3 GB, which already imposes a huge processing burden to the QDV algorithm. In the future, if permitted more powerful computing resources, it would be interesting to have a more comprehensive analysis on all the test instances.

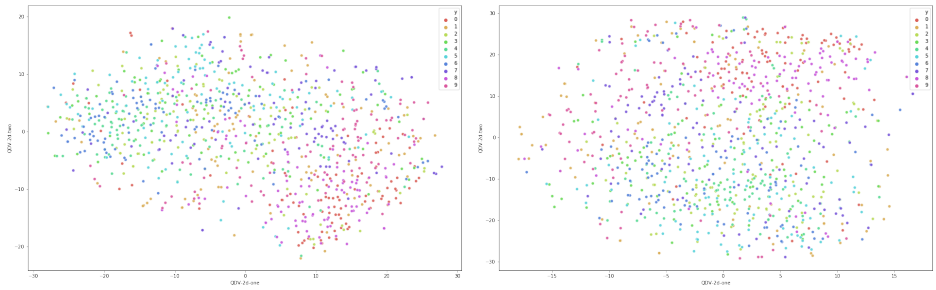
5.4 Conclusions

In this chapter, we seek to develop a reasoned way of refining neural nets with the QDV method we propose. This method is implemented on two pre-trained neural nets: the Alexnet and the VGG19. We apply them on the CIFAR-10 dataset. Our results, obtained with QDV, demonstrate quantitatively why VGG19 has higher accuracy than Alexnet on the same dataset. This quantitative conclusion could hence aid refining neural nets in [187] where the visualizations of Deconvnet are utilized to justify the selection of filter sizes and stride of convolutional layers. Our proposed method could be applied in situations when we try to understand and refine the architectures of neural nets. It should be noted that the main purpose of this approach is not to interpret the internal architectures of neural nets. The goal is to use visualizations/interpretability results to refine neural nets.

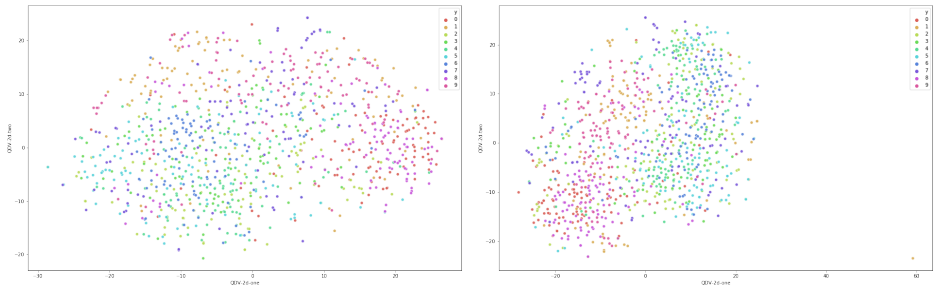
Applying QDV also enables us to interpret the neural nets directly on a simpler dataset (e.g. CIFAR-10), without distilling a complex model into a shallow model, as in knowledge distillation. This is more advantageous by maintaining the accuracy of the complex model while still obtaining easier interpretations on a smaller dataset compared to those on the original ImageNet data (1000 classes) [76].



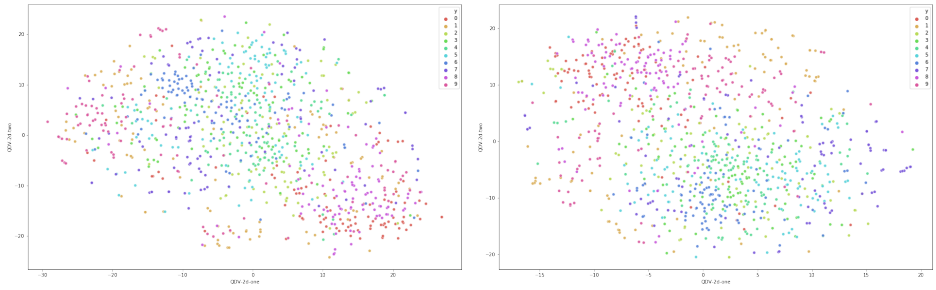
(a) Alexnet:maxpool1(AR=0.21) (b) VGG19:maxpool1(AR=0.22)



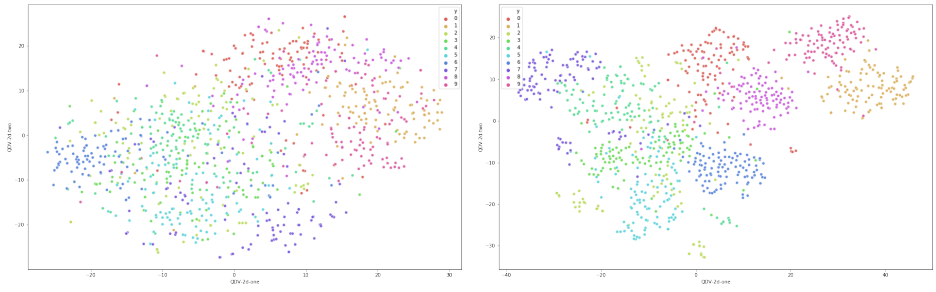
(c) Alexnet:maxpool2(AR=0.24) (d) VGG19:maxpool2(AR=0.25)



(e) Alexnet:CONV3(AR=0.29) (f) VGG19:maxpool3(AR=0.34)



(g) Alexnet:CONV4(AR=0.35) (h) VGG19:maxpool4(AR=0.42)



(i) Alexnet:maxpool5(AR=0.46) (j) VGG19:maxpool5(AR=0.80)

Figure 5.3: QDV visualization results and AR values for pre-trained Alexnet and VGG19. Higher AR implies better representation.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we discussed the need for interpretability of machine learning models, we have presented the taxonomy of the state of art interpretation approaches, and we have developed two methods to interpret deep neural networks and one method for refining deep neural nets. The two methods we introduced to interpret deep neural nets achieved good interpretation performance without sacrificing much of the prediction power of the investigated deep neural networks: the CNN-INTE method visually explains how a specific test instance is classified and the knowledge distillation technique improves the accuracy of distilled decision trees. In other words, we augmented black box predictions with explanations. The method we presented in chapter 5 for refining neural networks provides quantified explanations that could assist in refining the structure of neural nets.

6.2 Limitations

While reviewing the vast and constantly growing amount of literature on the topic of interpretability, we also noticed a few issues that are beyond the scope of this thesis but need to be addressed in the future.

- First, most current interpretation models assume that the original training data which is used to train the black-box model is available. However, in practice, these data might be proprietary in cases like extensive market surveys or financial analyses.

- Second, traditional transparent algorithms may not always be more interpretable than deep neural networks. For example, on high dimensional big data sets, the number of nodes for a decision tree and the number of conditions for a

rule is overwhelming to inspect. Moreover, to achieve comparable performance, the features used to train a linear model are usually heavily engineered, which reduces its interpretability. For instance, in [100], to achieve similar performance as RNNs the features of linear models are heavily hand-engineered whereas the RNNs operate on raw features. In this sense, neural networks sometimes exhibit more interpretabilty since they require little feature engineering and the representations can be visualized or verbalized.

- Third, it is a frequently encountered situation that the performance of deep neural networks is easily impacted by the adversarial examples [160] [121]. These adversarial examples could be designed manually to attack the neural network models. They have imperceptible subtle variations for humans but could result in totally reversed predictions on neural networks. One consequence is that they may cause the unreliability of those local interpretation methods that rely on the neighbourhood data points for interpretation because these data points may inadvertently be adversarial. For example, LIME [133] uses perturbations to find sparse linear models as explanations. During this process, in order to fit a local linear model around a specific data point, a number of instances are randomly sampled around this point. However, these random samples might be adversarial examples which may lead to the unreliability of the LIME approach.

- Finally, Post-hoc Interpretation is brittle. Several criticisms about post-hoc interpretations are raised in [140]. It was mentioned that the explanations are not reliable and misleading because they are not faithful to the original model and just provides correlations instead of information about the original computation. It was argued in [58] that robustness of current post-hoc interpretation approaches was a big issue. These approaches are susceptible to adversarial attacks. Their results show that small systematic perturbations to the input data would result in the approaches generating different interpretations, although the altered input data are assigned the same predicted label as the unaltered ones. It was pointed out in [112] that the vast majority of explainable AI (xAI) produced functions that are more like scientific models rather than 'everyday' explanations and to bridge the gap they suggested to shift the field's attention to build

interactive methods. It was even proposed [71] that there was no hope of having interpretable models without intervention at training time (instead of post-hoc). To systematically demonstrate the vulnerabilities in current popular post-hoc techniques, especially those that rely on input perturbations such as LIME [133] and SHAP [107], a scaffolding technique is proposed in [151] where these techniques are so fooled that the explanations don't reflect the biased predictions at all.

6.3 Future Work

In this section, first, we describe the extensions of the three methods that we have presented in the previous chapters. In the end, we discuss some possible directions for further research.

6.3.1 Future Work for CNN-INTE

For the future work on the CNN-INTE approach, first we plan to introduce quantification of the interpreted results. Also, in our experiments, one of the things we find difficult is the setting of the number of clusters for the k-means algorithm. In the future, we plan to replace the k-means algorithm with DBSCAN [47] which doesn't require specifying the number of clusters. As stated in [92], "decision sets" seems to be a better option than decision tree as an inherently interpretable algorithm, so we also plan to replace decision tree with decision sets. Last but not least, it would be quite meaningful to apply this tool on real world applications with more complex data where interpretations are demanded either between the training data and the hidden layer or between the hidden layer and the predictions.

6.3.2 Future Work for Interpretation with Knowledge Distillation

For this work, there are several directions for future work. First, as specified in [17], there are various methods to solve the multi-output regression problem. The method we adopted is the algorithm adaptation method. It is worthwhile to explore other methods to fully take advantage of the power of knowledge distillation. Second, this approach makes it possible to improve the performance of all inherently interpretable

models and it is therefore rewarding to design new inherently interpretable models that could hopefully match the performance of non-interpretable models. Last, it should also be feasible to add a temperature term into the softmax layer (as introduced in the methodology part) and use both soft targets and the true labels together (as carried out in [67]) to train the student model in order to further improve the accuracy performance of the student model.

6.3.3 Future Work for Refining Neural Nets

As we've already mentioned in chapter 5, our method QDV directly interprets a neural net on a simpler dataset, without distilling a complex model into a shallow model, as in knowledge distillation. It would be interesting to compare the accuracies on the following two cases: (1) use a simpler dataset to fine tune a pre-trained deep neural net; (2) apply knowledge distillation to distill a pre-trained deep neural net into a shallow neural net. In our method QDV, we applied a dataset CIFAR-10 which has only 10 classes. It would be worth exploring the techniques for visualizing the results of datasets that have more than 10 classes. After all, lots of real-life data nowadays have more than 10 classes. However, presenting more than 10 classes in a figure is a big challenge and methods need to be figured out to avoid the mess as in [76]. Our method QDV then should be accommodated for this case.

6.3.4 Future Work for Further Research

Debugging Machine Learning Models Debugging machine learning models is critical for identifying and correcting the systematic mistakes of machine learning models before deploying them in the real world. Fig. 6.1 illustrates the general idea of discovering bugs in a machine learning model. Currently, there are a few of methods for fulfilling this task such as debugging using interpretability [145], program verification [75], visualization tools [26] [159] [177], novel adversarial attacks [135], etc. In the area of health care, it is also possible to apply inherently interpretable models such as GA^{2}Ms [24] to verify and debug machine learning models.

Detecting Bias There have been more and more indications that machine learning models may be biased. The bias we mention here is NOT the statistical bias when

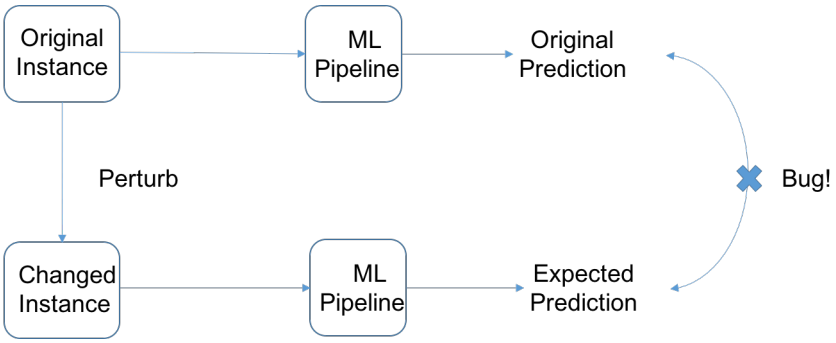


Figure 6.1: A common framework for debugging machine learning models.

designing a machine learning model where usually a bias/variance tradeoff exists [57]. And high bias means the algorithm is not fitting the training data properly. The bias here we discuss is the undesirable behavior learnt from the data which could influence the decisions of models but which is not aligned with the ethics of our society.

For instance, it was reported that Amazon hiring tool biased against women for software developer jobs and other technical posts [39]. Leveraging methods inspired by Cognitive Psychology, it is found that state-of-the-art one shot learning models were biased towards shape rather than color when categorizing objects [136]. it is also shown that word embedding exhibits gender bias: female/male stereotypes.

Interpretability could play an important role in detecting bias and be a tool for "debiasing". Methods are developed to "debias" the embedding [15]. By applying a particular form of example-based explanation, model criticism [79], detecting the biases is enabled in the predictions of the models [156]. And they also prove that adversarial examples could be employed for model criticism. There are also other interesting approaches that are proposed to detect bias in machine learning models [110] [165]. Especially, an overview of cognitive biases and "debiasing" methods are given in [83].

Evaluations of Interpretability In chapter 1.2.1, we discussed three general evaluation metrics for post-hoc interpretation, fidelity, comprehensibility and accuracy, which could serve as general guides to design interpretation methods. However, it is more and more recognized recently that learning from human feedback is also critical for evaluating interpretability [42] [117] [128]. The reason is that in many situations human preferences are very hard to incorporate into the computational property of models and this information must be learned from human data.

When designing inherently interpretable models, a number of proxies/regularizations are directly formulated and included in the interpretation models, which when optimized could maximize the interpretability of the model. For instance, the decision tree depth [50], overlaps between decision rules [92], sparsity penalties [139] etc [173] [183]. However, in practical cases, these proxies are not reliable because different contexts require different interpretability. For example, longer decision trees are preferred over shorter decision trees by doctors in [94]. Whereas when designing the model shorter decision trees are preferred as they have lower complexity. The crowdsourcing experiments in [55] also arrive at similar conclusion: from the angle of interpretability shorter rules are not desirable although the machine learning bias perfers it. This indicates the importance of including humans in the optimization loop. And the recommendations of incorporating cognitive bias into machine learning algorithms and using user studies to evaluate interpretability are also given [55]. Recently, an algorithm is proposed in [91] which could include the subjective notion of human interpretability as well as minimize the number of user studies.

Alternative Inherently Interpretable Models A recent argument [140] provided by Dr.Cynthia Rudin actually opposes applying black box machine learning models for high stake decisions. She suggests to adopt inherently interpretable models instead. However, it is easier said than done. We all recognize that it is a huge challenge for constructing transparent yet accurate models.

Here we propose a possible solution: multivariate decision trees. Different from the widespread application of univariate decision trees, multivariate decision trees were proposed in the 1990s but seldom applied in practice. The difference between multivariate and univariate decision trees is whether they test more than one feature at a node. Therefore, univariate decision trees are potentially much larger than multivariate decsion trees and not as succinct as the latter. The experiments [22] demonstrated that multivariate tests improved the accuracies of the resulting decision trees. As far as interpretability concerned, it may depend on the strategies adopted [22]. For instance, there is a trade-off to consider when applying multivariate tests: univariate tests produce large trees which are difficult to understand; multivariate tests generate smaller trees wither fewer nodes, but the nodes are more difficult to interpret.

There are multiple strategies to construct multivariate decision trees and hence the interpretability varies by the strategies.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[3] 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):373–389, 1995.

[4] Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. In *Advances in Neural Information Processing Systems*, pages 10019–10029, 2018.

[5] Charles Audet and Michael Kokkolaras. Blackbox and derivative-free optimization: theory, algorithms and applications, 2016.

[6] M Gethsiyal Augasta and Thangairulappan Kathirvalavakumar. Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters*, 35(2):131–150, 2012.

[7] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[8] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

[9] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446, 2015.

[10] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *arXiv preprint arXiv:1704.05796*, 2017.

[11] Pietro Berkes and Laurenz Wiskott. On the analysis and interpretation of inhomogeneous quadratic forms as receptive fields. *Neural computation*, 18(8):1868–1895, 2006.

[12] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[13] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In *International Conference on Artificial Neural Networks*, pages 63–71. Springer, 2016.

[14] Ekaba Bisong. Google colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 59–64. Springer, 2019.

[15] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, pages 4349–4357, 2016.

[16] Gwen Bonner. Decision making for health care professionals: use of decision trees within the community mental health setting. *Journal of Advanced Nursing*, 35(3):349–356, 2001.

[17] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.

[18] Olcay Boz. Extracting decision trees from trained neural networks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 456–461. ACM, 2002.

[19] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[20] Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*, volume 1st ed. New York: Routledge, 1984.

[21] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[22] Carla E Brodley and Paul E Utgoff. Multivariate decision trees. *Machine learning*, 19(1):45–77, 1995.

[23] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[24] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM, 2015.

[25] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18. ACM, 2004.

[26] Dylan Cashman, Adam Perer, and Hendrik Strobelt. Mast: A tool for visualizing cnn model architecture searches. In *ICLR 2019 Debugging Machine Learning Models Workshop*, 2019.

[27] Yair Censor. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, 4(1):41–59, 1977.

[28] Philip K Chan and Salvatore J Stolfo. Experiments on multistrategy learning by meta-learning. In *Proceedings of the second international conference on information and knowledge management*, pages 314–323. ACM, 1993.

[29] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM computing surveys (CSUR)*, 33(3):273–321, 2001.

[30] Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. Distilling knowledge from deep networks with applications to healthcare domain. *NIPS Workshop on Machine Learning for Healthcare (NIPS-MLHC)*, 2015.

[31] Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. Interpretable deep models for icu outcome prediction. In *AMIA Annual Symposium Proceedings*, volume 2016, page 371. American Medical Informatics Association, 2016.

[32] Cary Coglianese, Jennifer Nash, and Todd Olmstead. Performance-based regulation: Prospects and limitations in health, safety, and environmental protection. *Admin. L. Rev.*, 55:705, 2003.

[33] Shay Cohen, Gideon Dror, and Eytan Ruppin. Feature selection via coalitional game theory. *Neural Computation*, 19(7):1939–1961, 2007.

[34] Andrada Coos. Data protection in canada: All you need to know about pipeda, January 2019. `https://www.endpointprotector.com/blog/data-protection-in-canada-pipeda/`, Last accessed on 2020-3-27.

[35] Paulo Cortez and Mark J Embrechts. Opening black box data mining models using sensitivity analysis. In *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pages 341–348. IEEE, 2011.

[36] Paulo Cortez and Mark J Embrechts. Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, 225:1–17, 2013.

[37] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. Chapman and hall/CRC, 2000.

[38] Mark Craven and Jude W Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.

[39] Jeffrey Dastin. Amazon scraps secret ai recruiting tool that showed bias against women, Oct 2018. https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G, Last accessed on 2020-1-14.

[40] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 598–617. IEEE, 2016.

[41] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2017. http://archive.ics.uci.edu/ml.

[42] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[43] Kenneth Dwyer and Robert Holte. Decision tree instability and active learning. In *European Conference on Machine Learning*, pages 128–139. Springer, 2007.

[44] Mary T Dzindolet, Scott A Peterson, Regina A Pomranky, Linda G Pierce, and Hall P Beck. The role of trust in automation reliance. *International journal of human-computer studies*, 58(6):697–718, 2003.

[45] Mark J Embrechts, Fabio A Arciniegas, Muhsin Ozdemir, and Robert H Kewley. Data mining for molecules with 2-d neural network sensitivity analysis. *International Journal of smart engineering system design*, 5(4):225–239, 2003.

[46] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[47] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[48] Chin-Yuan Fan, Pei-Chann Chang, Jyun-Jie Lin, and JC Hsieh. A hybrid model combining case-based reasoning and fuzzy decision tree for medical data classification. *Applied Soft Computing*, 11(1):632–644, 2011.

[49] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *arXiv preprint arXiv:1704.03296*, 2017.

[50] Alex A Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10, 2014.

[51] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[52] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[53] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[54] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *presented at the CEX workshop at AI*IA 2017 conference*, 2017.

[55] Johannes Fürnkranz and Tomáš Kliegr. The need for interpretability biases. In *International Symposium on Intelligent Data Analysis*, pages 15–27. Springer, 2018.

[56] Johannes Fürnkranz, Tomáš Kliegr, and Heiko Paulheim. On cognitive preferences and the plausibility of rule-based models. *arXiv preprint arXiv:1803.01316*, 2018.

[57] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[58] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.

[59] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[60] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[61] Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe, and Andrew Y Ng. Measuring invariances in deep networks. In *Advances in neural information processing systems*, pages 646–654, 2009.

[62] Riccardo Guidotti, Anna Monreale, Stan Matwin, and Dino Pedreschi. Black box explanation by learning image exemplars in the latent feature space. *arXiv preprint arXiv:2002.03746*, 2020.

[63] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5):93, 2018.

[64] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[65] Andreas Henelius, Kai Puolamäki, Henrik Boström, Lars Asker, and Panagiotis Papapetrou. A peek into the black box: exploring classifiers by randomization. *Data mining and knowledge discovery*, 28(5-6):1503–1529, 2014.

[66] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.

[67] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS 2014 Deep Learning Workshop*, 2015.

[68] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.

[69] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[70] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[71] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

[72] Ozan Irsoy, Olcay Taner Yıldız, and Ethem Alpaydın. Soft decision trees. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1819–1822. IEEE, 2012.

[73] Ulf Johansson and Lars Niklasson. Evolving decision trees using oracle guides. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE, 2009.

[74] Ulf Johansson, Lars Niklasson, and Rikard König. Accuracy vs. comprehensibility in data mining models. In *Proceedings of the seventh international conference on information fusion*, volume 1, pages 295–300, 2004.

[75] Daniel Kang, Deepti Raghavan, Peter Bailis, and Matei Zaharia. Model assertions for debugging machine learning. In *NeurIPS MLSys Workshop*, 2018.

[76] Andrej Karpathy. t-sne visualization of cnn codes, 2014. `http://cs.stanford.edu/people/karpathy/cnnembed/`, Last accessed on 2019-9-16.

[77] Alon Keinan, Ben Sandbank, Claus C Hilgetag, Isaac Meilijson, and Eytan Ruppin. Fair attribution of functional contribution in artificial and biological networks. *Neural Computation*, 16(9):1887–1915, 2004.

[78] Robert H Kewley, Mark J Embrechts, and Curt Breneman. Data strip mining for the virtual design of pharmaceuticals with neural networks. *IEEE Transactions on Neural Networks*, 11(3):668–679, 2000.

[79] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in Neural Information Processing Systems*, pages 2280–2288, 2016.

[80] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. *arXiv preprint arXiv:1711.00867*, 2017.

[81] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

[82] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[83] Tomáš Kliegr, Štěpán Bahník, and Johannes Fürnkranz. A review of possible effects of cognitive biases on interpretation of rule-based machine learning models. *arXiv preprint arXiv:1804.02969*, 2018.

[84] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

[85] Igor Kononenko et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(Jan):1–18, 2010.

[86] R Krishnan, G Sivakumar, and P Bhattacharya. Extracting decision trees from trained neural networks. *Pattern recognition*, 32(12), 1999.

[87] Sanjay Krishnan and Eugene Wu. Palm: Machine learning explanations for iterative debugging. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 4. ACM, 2017.

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[89] Joshua A Kroll, Solon Barocas, Edward W Felten, Joel R Reidenberg, David G Robinson, and Harlan Yu. Accountable algorithms. *U. Pa. L. Rev.*, 165:633, 2016.

[90] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[91] Isaac Lage, Andrew Ross, Samuel J Gershman, Been Kim, and Finale Doshi-Velez. Human-in-the-loop interpretability prior. In *Advances in Neural Information Processing Systems*, pages 10159–10168, 2018.

[92] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684. ACM, 2016.

[93] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Interpretable & explorable approximations of black box models. *arXiv preprint arXiv:1707.01154*, 2017.

[94] Nada Lavrač. Selected techniques for data mining in medicine. *Artificial intelligence in medicine*, 16(1):3–23, 1999.

[95] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[96] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[97] Jefrey Lijffijt, Panagiotis Papapetrou, and Kai Puolamäki. A statistical significance testing approach to mining the most informative set of patterns. *Data Mining and Knowledge Discovery*, 28(1):238–263, 2014.

[98] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[99] Zachary C Lipton. The mythos of model interpretability. *ICML Workshop on Human Interpretability in Machine Learning (WHI)*, 2016.

[100] Zachary C Lipton, David C Kale, and Randall Wetzel. Modeling missing data in clinical time series with rnns. *Machine Learning for Healthcare*, 2016.

[101] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 905–912. IEEE, 2018.

[102] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Interpretable deep convolutional neural networks via meta-learning. *2018 International Joint Conference on Neural Networks, IJCNN*, 2018.

[103] Xuan Liu, Xiaoguang Wang, Stan Matwin, and Nathalie Japkowicz. Meta-learning for large scale machine learning with mapreduce. In *Big Data, 2013 IEEE International Conference on*, pages 105–110. IEEE, 2013.

[104] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158. ACM, 2012.

[105] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631. ACM, 2013.

[106] Chaochao Lu and Xiaoou Tang. Surpassing human-level face verification performance on lfw with gaussianface. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[107] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[108] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[109] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[110] Daniel McDuff, Roger Cheng, and Ashish Kapoor. Identifying bias in ai using simulation. *arXiv preprint arXiv:1810.00471*, 2018.

[111] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[112] Brent Mittelstadt, Chris Russell, and Sandra Wachter. Explaining explanations in ai. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 279–288. ACM, 2019.

[113] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[114] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[115] Stefano Moretti and Fioravante Patrone. Transversality of the shapley value. *Top*, 16(1):1, 2008.

[116] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[117] Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682*, 2018.

[118] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng. Tiled convolutional neural networks. In *Advances in neural information processing systems*, pages 1279–1287, 2010.

[119] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, volume 2, page 7, 2017.

[120] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pages 3387–3395, 2016.

[121] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[122] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

[123] Markus Ojala and Gemma C Garriga. Permutation tests for studying classifier performance. *Journal of Machine Learning Research*, 11(Jun):1833–1863, 2010.

[124] Julian D Olden and Donald A Jackson. Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling*, 154(1-2):135–150, 2002.

[125] Cristina Onosé. Changes to canada's privacy laws looming: Balancing innovation and privacy, June 2018. `https://www.the-cma.org/about/blog/changes-to-canada-privacy-laws-looming`, Last accessed on 2020-3-27.

[126] Fernando V Paulovich, Luis G Nonato, Rosane Minghim, and Haim Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575, 2008.

[127] Brett Poulin, Roman Eisner, Duane Szafron, Paul Lu, Russell Greiner, David S Wishart, Alona Fyshe, Brandon Pearcy, Cam MacDonell, and John Anvik. Visual explanation of evidence with additive classifiers. In *Proceedings Of The National Conference On Artificial Intelligence*, volume 21, page 1822. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[128] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Vaughan, and Hanna Wallach. Manipulating and measuring model interpretability. *arXiv preprint arXiv:1802.07810*, 2018.

[129] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[130] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[131] Paulo E Rauber, Samuel G Fadel, Alexandre X Falcao, and Alexandru C Telea. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):101–110, 2016.

[132] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Nothing else matters: model-agnostic explanations by identifying prediction invariance. *NIPS 2016 Workshop on Interpretable Machine Learning in Complex Systems*, 2016.

[133] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[134] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018.

[135] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, 2018.

[136] Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2940–2949. JMLR. org, 2017.

[137] Ronald L Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

[138] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[139] Andrew Ross, Isaac Lage, and Finale Doshi-Velez. The neural lasso: Local linear sparsity for interpretable explanations. In *Workshop on Transparent and Interpretable Machine Learning in Safety Critical Environments, 31st Conference on Neural Information Processing Systems*, 2017.

[140] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206, 2019.

[141] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[142] Andrea Saltelli. Sensitivity analysis for importance assessment. *Risk analysis*, 22(3):579–590, 2002.

[143] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2017.

[144] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017.

[145] Ramprasaath R Selvaraju, Stefan Lee, Yilin Shen, Hongxia Jin, Shalini Ghosh, Larry Heck, Dhruv Batra, and Devi Parikh. Taking a hint: Leveraging explanations to make vision and language models more grounded. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2591–2600, 2019.

[146] Sukrit Shankar, Duncan Robertson, Yani Ioannou, Antonio Criminisi, and Roberto Cipolla. Refining architectures of deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2212–2220, 2016.

[147] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

[148] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[149] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[150] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[151] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. How can we fool lime and shap? adversarial attacks on post hoc explanation methods. *arXiv preprint arXiv:1911.02508*, 2019.

[152] Kacper Sokol and Peter Flach. Explainability fact sheets: a framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 56–67, 2020.

[153] Sören Sonnenburg, Alexander Zien, Petra Philips, and Gunnar Rätsch. Poims: positional oligomer importance matrices—understanding support vector machine-based signal detectors. *Bioinformatics*, 24(13):i6–i14, 2008.

[154] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[155] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[156] Pierre Stock and Moustapha Cisse. Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 498–512, 2018.

[157] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[158] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[159] Grzegorz Swirszcz, Brendan O'Donoghue, and Pushmeet Kohli. Visualizations of decision regions in the presence of adversarial examples. In *ICLR 2019 Debugging Machine Learning Models Workshop*, 2019.

[160] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[161] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[162] Sarah Tan, Rich Caruana, Giles Hooker, and Albert Gordo. Transparent model distillation. *arXiv preprint arXiv:1801.08640*, 2018.

[163] Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. Learning global additive explanations for neural nets using model distillation. *arXiv preprint arXiv:1801.08640v2*, 2018.

[164] Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. Learning global additive explanations for neural nets using model distillation. *arXiv preprint arXiv:1801.08640*, 2018.

[165] Sarah Tan, Rich Caruana, Giles Hooker, and Yin Lou. Detecting bias in black-box models using transparent model distillation. *arXiv preprint arXiv:1710.06169*, 2017.

[166] Jayaraman J Thiagarajan, Bhavya Kailkhura, Prasanna Sattigeri, and Karthikeyan Natesan Ramamurthy. Treeview: Peeking into deep neural networks via feature-space partitioning. *arXiv preprint arXiv:1611.07429*, 2016.

[167] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015.

[168] Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[169] R. Turner. A model explanation system. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Sep. 2016.

[170] Ryan Turner. A model explanation system. In *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*, pages 1–6. IEEE, 2016.

[171] G Steven Tuthill. Legal liabilities and expert systems. *AI Expert*, 6(3):44–51, 1991.

[172] Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep and convolutional? *arXiv preprint arXiv:1603.05691*, 2016.

[173] Berk Ustun and Cynthia Rudin. Optimized risk scores. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1125–1134. ACM, 2017.

[174] Laurens van der Maaten. Do's and don'ts of using t-sne to understand vision models, June 2018. `http://deeplearning.csail.mit.edu/slide_cvpr2018/laurens_cvpr18tutorial.pdf`, Last accessed on 2020-2-27.

[175] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10:66–71, 2009.

[176] Marina M-C Vidovic, Nico Görnitz, Klaus-Robert Müller, and Marius Kloft. Feature importance measure for non-linear learning algorithms. *NIPS 2016 Workshop on Interpretable Machine Learning in Complex Systems*, 2016.

[177] Jesse Vig. Bertviz: A tool for visualizing multi-head self-attention in the bert model. In *ICLR 2019 Debugging Machine Learning Models Workshop*, 2019.

[178] Bill Vlasic and Neal E. Boudette. 'self-driving tesla was involved in fatal crash,' u.s. says. *New York Times*, 30 June 2016.

[179] Aisuke Wakabayashi. Self-driving uber car kills pedestrian in arizona, where robots roam. *New York Times*, 19 March 2018.

[180] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 1(10):e2, 2016.

[181] Donglai Wei, Bolei Zhou, Antonio Torrabla, and William Freeman. Understanding intra-class knowledge inside cnn. *arXiv preprint arXiv:1507.02379*, 2015.

[182] Eyal Winter et al. The shapley value. *Handbook of game theory with economic applications*, 3(2):2025–2054, 2002.

[183] Mike Wu, Michael C Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. *AAAI*, 2018.

[184] Kai Xu, Dae Hoon Park, Chang Yi, and Charles Sutton. Interpreting deep classifier by visual distillation of dark knowledge. *arXiv preprint arXiv:1803.04042*, 2018.

[185] Zheng Yao, Peng Liu, Lei Lei, and Junjie Yin. R-c4. 5 decision tree model and its applications to health care dataset. In *Services Systems and Services Management, 2005. Proceedings of ICSSSM'05. 2005 International Conference on*, volume 2, pages 1099–1103. IEEE, 2005.

[186] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[187] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[188] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE, 2011.

[189] Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.

[190] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[191] Guorui Zhou, Ying Fan, Runpeng Cui, Weijie Bian, Xiaoqiang Zhu, and Kun Gai. Rocket launching: A universal and efficient framework for training well-performing light net. *stat*, 1050:16, 2017.

[192] Zhi-Hua Zhou, Yuan Jiang, and Shi-Fu Chen. Extracting symbolic rules from trained neural network ensembles. *Ai Communications*, 16(1):3–15, 2003.

[193] Alexander Zien, Nicole Krämer, Sören Sonnenburg, and Gunnar Rätsch. The feature importance ranking measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 694–709. Springer, 2009.