A SURVEY ON VISUALIZATION FOR EXPLAINABLE CLASSIFIERS

by

YAO MING

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Supervised by Prof. Huamin Qu

October 2017, Hong Kong

TABLE OF CONTENTS

Title Page i

Table of Contents ii

Abstract iv

Chapter 1 Introduction 1

1.1 Motivation 1

1.2 Challenges 2

1.3 Overview 3

Chapter 2 Concepts and Definitions 4

2.1 Classification 4

2.2 Explainability 4

Chapter 3 Explainable Classifiers 6

3.1 Interpretable Classifiers 7

3.1.1 Interpretable Architecture 7

3.1.2 Learning Sparse Models 8

3.2 Explaining Complex Classifiers 9

3.2.1 Local Explanations 10

3.2.2 Global Explanations 12

Chapter 4 Visualization for Explainable Classifiers 15

4.1 Life Cycle of an intelligent system 15

4.2 Visualization for Data Understanding 17

4.3 Visualization for Model Development 19

4.3.1 Understanding 20

4.3.2 Diagnosis 22

4.3.3 Assessment and Comparison 24

4.3.4 Communication and Education 25

4.4 Visualization for Operation 26

4.4.1 Trust Establishment 26

4.4.2 Monitoring 27

Chapter 5 Conclusion **28**

Bibliography **30**

A SURVEY ON VISUALIZATION FOR EXPLAINABLE CLASSIFIERS

by

YAO MING

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Classification is a fundamental problem in machine learning, data mining, and computer vision. In practice, interpretability is a desirable property of classification models (classifiers) in critical areas, such as security, medicine, and finance. For instance, a quantitative trader may prefer a more interpretable model with less expected return due to its predictability and low risk. Unfortunately, the best-performing classifiers in many applications (e.g., deep neural networks) are complex models whose predictions are difficult to explain. Thus, there is a growing interest in using visualization to understand, diagnose and explain intelligent systems in both academia and industry. Many challenges need to be addressed in the formalization of explainability, and the design principles and evaluation of explainable intelligent systems.

The survey starts with an introduction to the concept and background of explainable classifiers. Efforts towards more explainable classifiers are categorized into two types: designing classifiers with simpler structures that can be easily understood; developing methods that generate explanations for already complicated classifiers. Based on the life circle of a classifier, we discuss the pioneering work of using visualization to improve its explainability at different stages in the life circle. The survey ends with a discussion about the challenges and future research opportunities of explainable classifiers.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Classification is the problem of identifying if an observation or object belongs to a set or not, or which of several sets. It is a fundamental problem in machine learning, data mining, and computer vision. With the support of the increasing capacity of computation resources and the growing volume of available data, the last few decades have witnessed an explosion of breakthroughs in these fields. Nowadays, classification models (classifiers) are widely adopted to solve real-world tasks, including face recognition [63], handwriting recognition [26], sentiment analysis [39] and spam filtering [3]. Take image classification for instance, a well-designed convolutional neural network can achieve human-level performance in a number of benchmark datasets [17].

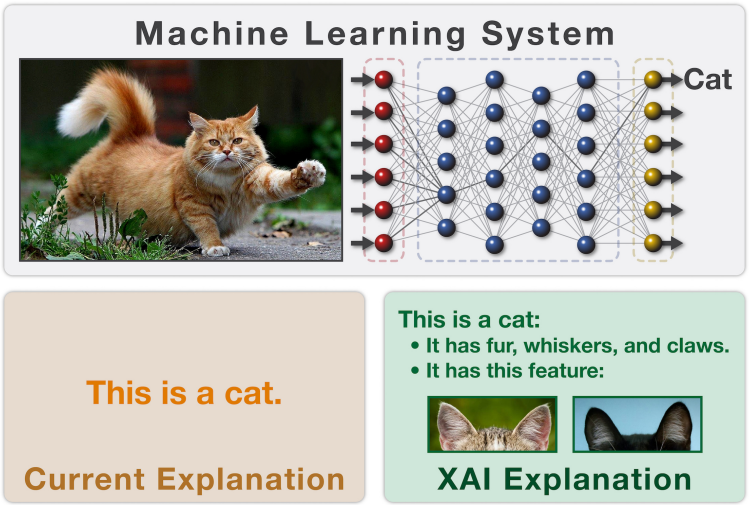

Figure 1.1. An illustration of an explainable image classifier [16].

Despite their promising capabilities, an often-overlooked aspect is the important role of humans [46]. When humans are to understand and collaborate with these autonomous systems, it is desirable if we have explanations of their outputs. For instance, a doctor using a machine classifier to assist in identifying early signs of lung cancer would need to know why the classifier "thinks" there might be cancer so that he/she can make a more confident diagnosis. An example is shown in Figure 1.1. In machine learning, the term *explainability*

does not have a standard and generally accepted definition. In some literature, *interpretability* is used instead. Generally speaking, the explainability or interpretability of an intelligent system refers to the ability to explain its reasoning to humans [10]. For the sake of consistency, we use explainability as the ability to explain in this survey. Interpretability is used to refer to the property of how easily a model can be understood by humans.

The research for explainable intelligent systems can be traced back to the 1980s when expert systems were created and proliferated [9, 37, 54]. These early works focused on reducing the difficulty of maintaining the complicated if-then rules by designing more explainable representations. A huge gap exists between today's state-of-the-art intelligent systems and the techniques that can make them explainable. The new challenges brought about by the new generation of intelligent systems have attracted growing research interests. DARPA launched the Explainable Artificial Intelligence (XAI) project [16], which aims to develop new techniques to make these systems explainable. Google Inc. initiated the People + AI Research Initiative (PAIR) [15] to advance humanistic considerations in AI.

Visualization is an effective and efficient technique for communicating information and understanding complex datasets for humans. The visual system is a proxy with a very large bandwidth to human brains [35]. Thus, visualization can be an ideal weapon to help explain complicated classifiers to humans. Early related research can be traced back to the software and algorithm visualization for computer science education in the 1980s and 1990s [8, 52, 42]. Visualization, especially interactive visualization, was proved to be very effective in facilitating people's understanding of complex software and algorithms. Little research has been done to visualize the increasingly complicated classifiers, which are actually algorithms learned from the data. It has not been until recently that visualization was popularized as a media for understanding classification models, especially for image classifiers [49, 64, 4, 65]. However, these methods have limited applications to neural networks for image data. There is also a lack of a unified and convenient evaluation method for the generated visualizations.

1.2 Challenges

The need for visually explaining classifiers is actually a result of the successes and advances of AI. The major challenges of visually explaining classifiers results from the complexity of the model and data, and the limits of humans.

First, it is challenging to explain complex classification models both concisely and precisely. The best-performing classifiers (e.g., neural networks) are becoming increasingly

complex, in terms of the number of parameters and operations they employed, which makes them difficult to explain. A convolutional network typically employs thousands of neurons and millions of parameters. A random forest used for classification may employ hundreds of decision trees, each containing hundreds of nodes. Sampling a small number of parameters/neurons/nodes to explain might be easier to understand for humans, but it brings with it the risk of misunderstanding as well. The variety of model architectures also increases the difficulty of designing an effective and general framework for explaining classifiers.

Another challenge is the volume and variety of the data used for training the classifiers. To explain a classifier, a most common strategy is to trace back to the input data. Which part of the input data contributes to the prediction? How does a model behave on this subset of data? Some explanation methods require computations over the entire training set, which may become impractical if the data set is very large. Different data types may require different forms of visual explanation. Image data are readily interpretable, but how to effectively explain classifiers on categorical, text and speech data is still a problem.

These challenges are, to some extent, due to compromises owing to the limits of humans' cognitive ability. If humans can make sense of the meaning of thousands of parameters and complex model structures by merely looking at the raw data or code, there is no need to struggle with how to better visualize them. There are already some studies discussing the structure, function, and effectiveness of explanations in cognitive science. However, it is still unclear how we can effectively evaluate the quality of an explanation, and the load that its visual representations exert over humans.

1.3 Overview

This survey mainly focuses on how visualization techniques can be used to support explainable classifiers. In Chapter 3, we first introduce the definition of classification and classifiers, and the concept of explainable classifiers. Two major research directions towards more explainable classifiers are identified: designing classifiers that are readily interpretable, and methods that generate explanations for a classifier without modifying the model. In Chapter 4, we first articulate the life cycle of a classifier into different stages, that is, the recursive procedures of data collection and processing, model development and testing, and operations and maintenance. Then, we illustrate how visualization can be applied at different stages to provide explainability for classifiers. Based on the specified life cycle, we categorize the surveyed literature and discuss the challenges and opportunities for future research in visualization for explainable classifiers.

CHAPTER 2

CONCEPTS AND DEFINITIONS

In this section, we first briefly introduce the problem of classification, as an instance of supervised learning, and a few popular classifications models (classifiers). To clarify the scope of this survey, we discuss the concepts of explainability of classifiers and illustrate in which circumstances explainability are desirable or needed.

2.1 Classification

Given an input space $\mathcal{X}$ and an output space $\mathcal{Y} = \{1, 2, ..., K\}$ with K classes, **classification** is the problem of identifying any **observation** $\mathbf{x} \in \mathcal{X}$ to a class $y \in \mathcal{Y}$. For multi-label classification, where class labels are not exclusive, we can view it as multiple related binary classifications. For simplicity, we only consider the basic formulation in this survey.

A **classifier** is an algorithm f that implements classification, *i.e.*, $y = f(\mathbf{x})$. To handle ambiguity, a classifier is often used in a probabilistic setting, that is, the output of f is a probabilistic distribution $p(y \mid \mathbf{x}, \mathcal{D})$ over all possible classes in $\mathcal{Y}$. $\mathcal{D}$ is the training set, which is a subset of $\mathcal{X} \times \mathcal{Y}$, that have already been observed. Thus, in practice, a classifier will often take the form of $\mathbf{y} = f(\mathbf{x})$, where $\mathbf{y} = (y_i) \in \mathbb{R}^K$ is a vector denoting the probabilistic distribution. Then the final classification will be the class i with largest probability $\arg\max_i y_i$.

2.2 Explainability

What is explainability? What is the explainability of a classifier? There is no commonly agreed definition so far. Doshi-Velez and Kim define interpretability (or explainability) as the ability to explain or to present in understandable terms to a human [10], which is already a good general definition. To clarify the scope of this survey, we define the **explainability** of a classifier as the ability to explain the reasoning of its predictions so that a human can understand. Simple models such as a linear classifier already have good explainability since humans can easily understand the model's reasoning by simply looking at the coefficients of each feature. For a complicated classifier like a deep neural network, a human may find it difficult to understand due to layer-wise structure and the nonlinearity of the computation. Thus, the key issue of explainability is the cognitive capability of humans.

An immediate question is: why do we need explainability? The need of explainability for a full automated classifier mainly comes from three aspects: humans' curiosity about knowledge, limitations of current intelligent algorithms, and moral and legal issues:

- **The curiosity of human**. Humans are curious about new knowledge. Often, a classifier is not developed solely for performing the classification tasks, but also for knowledge discovery. For today's popular neural networks, humans are curious of how the impressive human-level classification accuracy is achieved. There are also examples of how insights learned from the behavior of a model lead to improvement on the design of a classifier [64, 2]. Besides, given that AlphaGo Zero [48] can learn to master the game of Go much better than human players, it is desirable that the machine can explain its learned strategy (knowledge) to us.

- **Limitations of machines**. The current state-of-the-art intelligent systems are usually not fully testable. Human knowledge is required as a complement in case the machines fail. In the seeable future, machines are expected to assist rather than replace humans in many domains, such as security, medical services, education, and design. By providing explainability, users' trust can be more easily established. Besides, explainability can provide an interface for humans to monitor machine.

- **Moral and legal issues**. The "right to explanation", which is a regulation included in the GDPR [1] of the European Union, has recently raised a debate on to which extent we should require automatic decision-making systems to provide explanations to the subjects of the decision. If one's application for a loan is denied by an automatic classifier, he/she has the right to ask why. A doctor may need to know why a patient is classified to have a lung cancer to give the final diagnosis. Another issue is the fairness or the discrimination problem of a classifier, which may be easily neglected during the development phase.

Though explainability is a desirable property, it should be noted that it is not always necessary. Explainability is not required if 1) the application domain has high resistance to errors, and thus, unexpected errors are acceptable; 2) the application domain has been well studied and the classifier has been well tested in production, and thus, it is unlikely to have unexpected results.

[1] https://www.privacy-regulation.eu/en/r71.htm

CHAPTER 3

EXPLAINABLE CLASSIFIERS

In this section, we discuss methods that provide explainability to classifiers. We categorized related work into two types depending on how the explainability of a classifier is achieved.

The first type of work develops more interpretable classifiers that are easy to understand for humans. The second generates explanations for a classifier without modifying the model, either by explaining the classifier locally on specific instances or by explaining the behavior of the classifier globally. A summary is shown in Table 3.1.

Categories			Related Papers	Remarks
Interpretable Classifiers	Interpretable Architecture		Decision Trees [7], Rule Lists [27, 59], Rule Sets [60]	rule-based
			Linear Models [6]	linear
			kNNs [12, 22]	instance-based
	Learning Sparse Models		Decision Trees [43], Sparse SVMs [11], Sparse CNNs [29]	simplification
			Sparsity by Bayesian [56], Integer Models [55, 58]	direct-sparsity
Explanations of Classifiers	Local	Model-unaware	Sensitivity Analysis [49, 28, 50]	gradient-based
			LIME [46]	model induction
			Generate Visual Explanations [19]	extra labels
		Model-aware	De-convolution [64], Layer-wise Propagation [4], Prediction Difference [65], Output Decomposition [36], Direct Mapping [21]	CNN CNN Image LSTM RNN
	Global	Unaware	LIME [46]	model induction
		Model-aware	Partition Hidden Space [14, 44], Activation maximization [13], Network Dissection [5]	NN CNN CNN

Table 3.1. Towards explainable classifiers.

if hemiplegia **and** age >60 **then** *stroke risk* 58.9% (53.8%–63.8%)
else if cerebrovascular disorder **then** *stroke risk* 47.8% (44.8%–50.7%)
else if transient ischaemic attack **then** *stroke risk* 23.8% (19.5%–28.4%)
else if occlusion and stenosis of carotid artery without infarction **then** *stroke risk* 15.8% (12.2%–19.6%)
else if altered state of consciousness **and** age >60 **then** *stroke risk* 16.0% (12.2%–20.2%)
else if age ≤ 70 **then** *stroke risk* 4.6% (3.9%–5.4%)
else *stroke risk* 8.7% (7.9%–9.6%)

Figure 3.1. A decision list learned by the BRL algorithm [27].

3.1 Interpretable Classifiers

Interpretable classifiers are the classifiers that are commonly recognized to be more understandable than others, and hence, do not need extra explicit explanations. Summarizing existing work, we find two major strategies for creating interpretable classifiers: developing interpretable models with easy-to-understand structures, and learning simpler or sparser models.

3.1.1 Interpretable Architecture

To create more interpretable classifiers, a natural way is to use simple computation structures (*e.g.*, if-then rules). Most classifiers that fall into this category are rule-based.

Rule-based. A widely adopted type of models are the decision trees [7]. A decision tree classifier uses internal nodes and branches to represent its classification reasoning as conjunctions of rules. A human can trace back a specific classification from a leaf to the root to understand the prediction of the classifier. However, the difficulty of constructing a high-accuracy and interpretable decision tree has long been criticized.

Focused on balancing among performance, explainability, and computation, a few recent studies introduce the Bayesian framework in rule-based classifiers. Letham *et al*. [27] develop the Bayesian Rule List (BRL) which employs a prior structure that encourages sparsity in the generated decision lists with a good accuracy. The rule lists have the form of if-then-else structures, as shown in Figure 3.1. Wang and Rudin [59] design the Falling Rule Lists that use an ordered if-then rule list so that the most at-risk occasion will be handled first. Wang *et al*. [60] construct rule sets based on AND and OR operations and highlight its low computation cost and on-par accuracy compared with SVM and random forest.

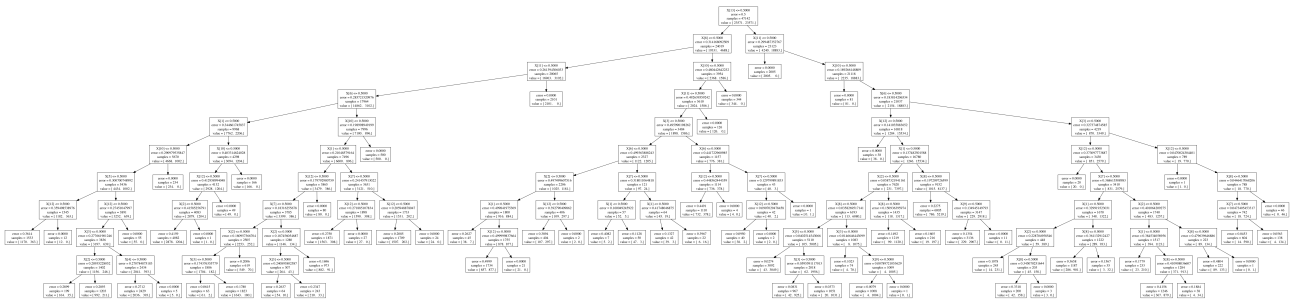

Figure 3.2. A decision tree with over one hundred nodes, which is hard to explain its reasoning[1].

The most series problem of these interpretable models with easy-to-understand structures is the scalability. The performance of the rule-based models increases as the number of rules increases or the non-linearity increases. Although the rule-based models are easy to learn and understand at the first glance, it is intractable to understand the classifier as a whole when the number of nodes of rules grows up to a few hundred. An example is shown in Figure 3.2.

Others. Except for rule-based models, there are a few other models with more complicated models are recognized to be interpretable. One family of interpretable models worth noticing are the generalized linear models [6], which are pervasive in statistics and finance. Although these models can have highly nonlinear computations, the additive relation between nonlinear functions of features is believed to be easy-to-understand. However, the generalized linear classifiers can also be hard to understand when their non-linearity increased to a certain extent. The other non-probabilistic family of classifiers is the k-nearest neighbors (kNN) classifiers, whose prediction can be easily understood by presenting the observation's k-nearest neighbors. Numerous work has been done to boost the performance the kNN classifiers, including weighted kNN with different kernels [12] and fuzzy kNN [22]. The explainability of kNN classifiers may easily fail when there lack good neighbors for certain observations.

3.1.2 Learning Sparse Models

As discussed above, the explainability often decreases as the complexity (*i.e.*, number of parameters or nodes) of the model increases. Thus, we can improve the explainability by learning a sparser model with the same architecture. Two common strategies are used to learn a sparse model: simplifying a "dense" model through pruning or approximation; introducing sparsity as a prior to learn a sparse model from scratch.

[1] `https://umbrella.cisco.com/blog/2013/06/13/server-side-software-and-malware-analysis/`

Simplification. The methods for simplifying classifiers are usually developed in a model-specific manner. Quinlan [43] summarizes four techniques for simplifying decision trees , *i.e.*, cost-complexity pruning, reduced error pruning, pessimistic pruning and simplification to rule sets. Downs *et al*. [11] recognize and eliminate dependent support vectors while leaving the outputs of SVM unchanged. Liu *et al*. [29] use a sparse decomposition method to zero out redundant parameters in a CNN, achieving about 10-times speedup while only losing about 1% accuracy. Though these methods can practically simplify classifiers and speed up computations, they do not directly provide explainability. A simplified SVM with fewer support vectors but utilizing a complicated kernel is still difficult to explain. A simplified decision tree with 200 nodes instead of 1,000 nodes is still hard to interpret.

Learning from scratch. To directly learn sparser models from scratch, Tipping [56] proposes a general Bayesian framework that treats sparsity as a prior and specialized this method on SVM. Instead of restricting the complexity of the parameters, Tan *et al*. [55] uses a 0-1 control variable to each input feature, and convert the learning to a mixed integer programming problem. The similar idea can be found in the sparse linear integer models proposed by Ustun *et al*. [58]. Although these methods can learn sparse classifiers without losing much performance, they mainly focus on reducing the computation costs instead of providing explainability. They do not guarantee explainability if the classification problem is difficult.

In most cases, the efforts of developing more interpretable classifiers are tradeoffs between performance and explainability. For performance-critical applications, it is always difficult to train an interpretable classifier that does not need extra explanations.

3.2 Explaining Complex Classifiers

Generating explanations of a classifier without modifying the classifier itself is preferred when the underlying model is already too complicated, *e.g.*, neural networks and SVMs, and we don't want to sacrifice performance. There is also no common-recognized definition for what an **explanation** of a classifier is. Most existing work uses a subset or a weighted subset of input features to explain a single prediction of a classifier, *e.g.*, a mask over the input image, a heatmap with the same size as the input image, a bag of words or categorical fields. Some work [46] proposed to induct a simpler classifier (*e.g.*, linear classifier) as the explanation of a prediction. Here we only discuss explanations for complex classifiers. Thus, illustrative diagrams for simple classifiers are not included here.

In cognitive science, explanations are characterized as arguments that demonstrating all

or a subset of the causes of the **explanandum** (the subject being explained), usually following deductions from natural laws or empirical conditions [18, 33]. Here we give a general definition:

Explanations of a classifier are the human-understandable representations that identify the causes of the classifier's prediction(s). A typical form of human-understandable representations is the visualization. As introduced in Section 2.1, a classifier can be regarded as a function f, which is in general learned from a training dataset $\mathcal{D}$, specified by learned parameters θ. Thus, the **causes** can be traced to 1) parts of the training data $\mathcal{D}$, 2) parts of the parameters θ as some components of the classifier, or 3) parts of the input, x, of a prediction or predictions.

If an explanation is provided to explain f's prediction in a small region around a given input point $\mathbf{x}$, we call it as a **local explanation**. If it is generated to explain f on the whole input space $\mathcal{X}$ in general or as a summary, we call it as a **global explanation**. Sometimes, we also want to have an intermediate **subset explanation** that is performed on a subset $\mathcal{S}$ of $\mathcal{X}$, in which the inputs share some common features. Next, we discuss the related work on explanations of classifiers based on the above taxonomy.

3.2.1 Local Explanations

As we have discussed, the causes used in an explanation of a classifier can be the training data, model parameters, and inputs. For local explanation, the input $\mathbf{x}$ is always specified and used in the explanation. Depending on whether an explanation is generated directly using model's parameters or structures, we categorize local explanation methods into model-aware and model-unaware methods.

Model-unaware explanations only require the input $\mathbf{x}$ and a computable f. Most work uses sensitivity or saliency-based techniques to derive explanations. Simonyan *et al*. [49] use the derivative of an image classifier f_i w.r.t. the input image $\mathbf{x}$ as the saliency score of the class i, and map the score of each pixel to a saliency map as the explanation of $\mathbf{x}$. Li *et al*. [28] also calculate the derivative of a text sentiment RNN classifier w.r.t. the embeddings of an input sentence (which is a matrix), as shown in Figure 3.4. The heatmap matrices are used as explanations for users to identify salient dimensions of the embedding vector and salient words that contribute the most to the prediction. Although these sensitivity-based methods are intuitive and can be efficiently approximated, their generated explanations are often noisy (as shown in Figure 3.3), due to the high nonlinearity of the complicated classifier. Recently, Smilkov *et al*. [50] propose a random sampling technique to smooth the gradient,

Figure 3.3. Images (first row) and their saliency maps (second row) for the top-1 predicted class in ILSVRC-2013 test images [49].

which achieves more meaningful visual explanations. However, this smoothing technique is computationally expensive and non-deterministic.

Instead of sensitivity analysis, some other work trains another model to explain the explanandum. Ribeiro *et al*. [46] approximate a complicated classifier locally using a simple linear classifier, and proceed to generate super-pixel patches as explanations. This method is actually similar to the gradient smoothing, since the training of the linear classifier is also done by sampling around the current input. Forming the problem as image captioning, Hendricks *et al*. [19] use extra labeled explanation texts of images to train an explainer that generates explanatory texts of an image classifier. This method highly depends on the quality of text explanations labels, which require extra expensive labeling. Besides, it introduces another model, which is another potential explanandum that needs to explain.

Model-aware explanations utilize another cause – the parameters of the model, θ, to amplify the information in the explanation. Zeiler and Fergus [64] develop a de-convolution method that maps the outputs of a CNN classifier back to the input space, utilizing the inverse operations of different layers. The projected images Figure 3.5 can be used as an explanation of what different neurons are used for. Bach *et al*. [4] use a layer-wise relevance propagation (LRP) that improves the sparsity of the image heatmap. Zintgraf *et al*. [65] develop a visualization that highlights evidence for and against a prediction separately through prediction difference analysis. Murdoch and Szlam [36] decompose the output of an LSTM classifier into multiplicative contribution scores of input words and uses the scores

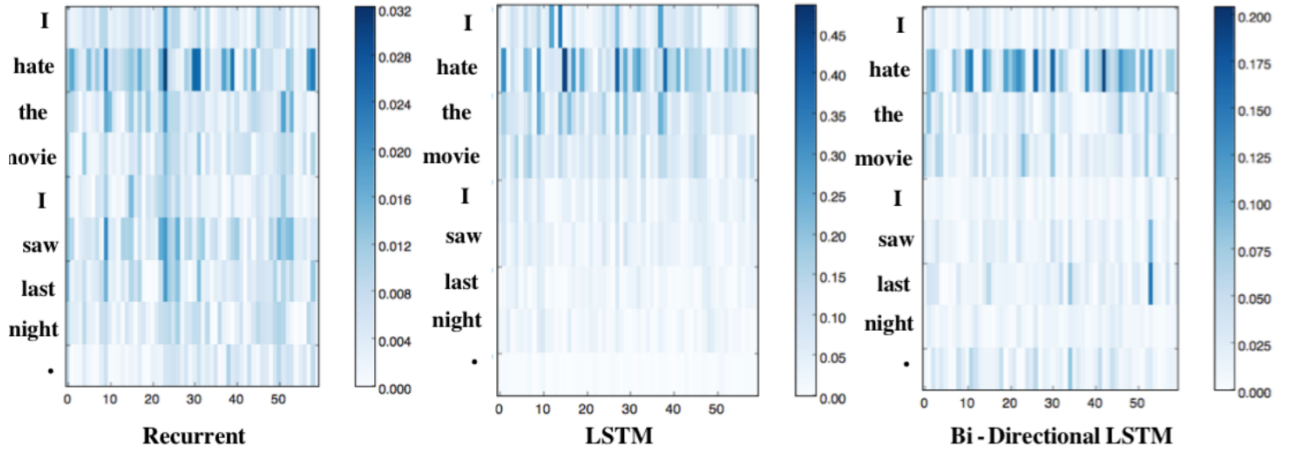

Figure 3.4. Saliency matrix maps for "I hate the movie I saw last night." of three RNN sentiment classifiers. *Left*: a vanilla RNN; *Middle*: an LSTM; *Right*: a bi-directional LSTM [49].

to explain how important the words are for the prediction. Though these methods typically result in much better (sparse, meaningful) explanations than model-unaware methods, they are developed in a per-model manner, which are hard to generalize for other classifiers. There is a lack of general explanatory frameworks to guide and evaluate the development of model-aware methods. Flooded by the interest in deep learning, we can hardly find methods developed for classification models other than neural networks.

Using the cause of training data for explanations has not attracted interest until recently. Koh and Liang propose a fast approximation of the influence function, which is a well-studied method in statistics. The influence functions can help identifying training points that are most responsible for a given prediction [23], and thus can explain the prediction from the aspect of training data.

3.2.2 Global Explanations

The global explanations are not dependent on any specific inputs. A global explanation is actually a summary of the reasoning of how a classifier generally behaves. Unlike local explanations which are defined around a certain point, the global explanations are considered to be ill-posed and are much harder to achieve. Similarly to local explanations, we divide existing work into model-aware and model-unaware methods.

Model-unaware explanations. To our knowledge, very few methods have been proposed to generate global explanations for general classifiers. Ribeiro *et al*. [46] select a collection of representative local explanations and present to the users one local explanation at a time to give a global understanding. This method will easily fail when the dataset

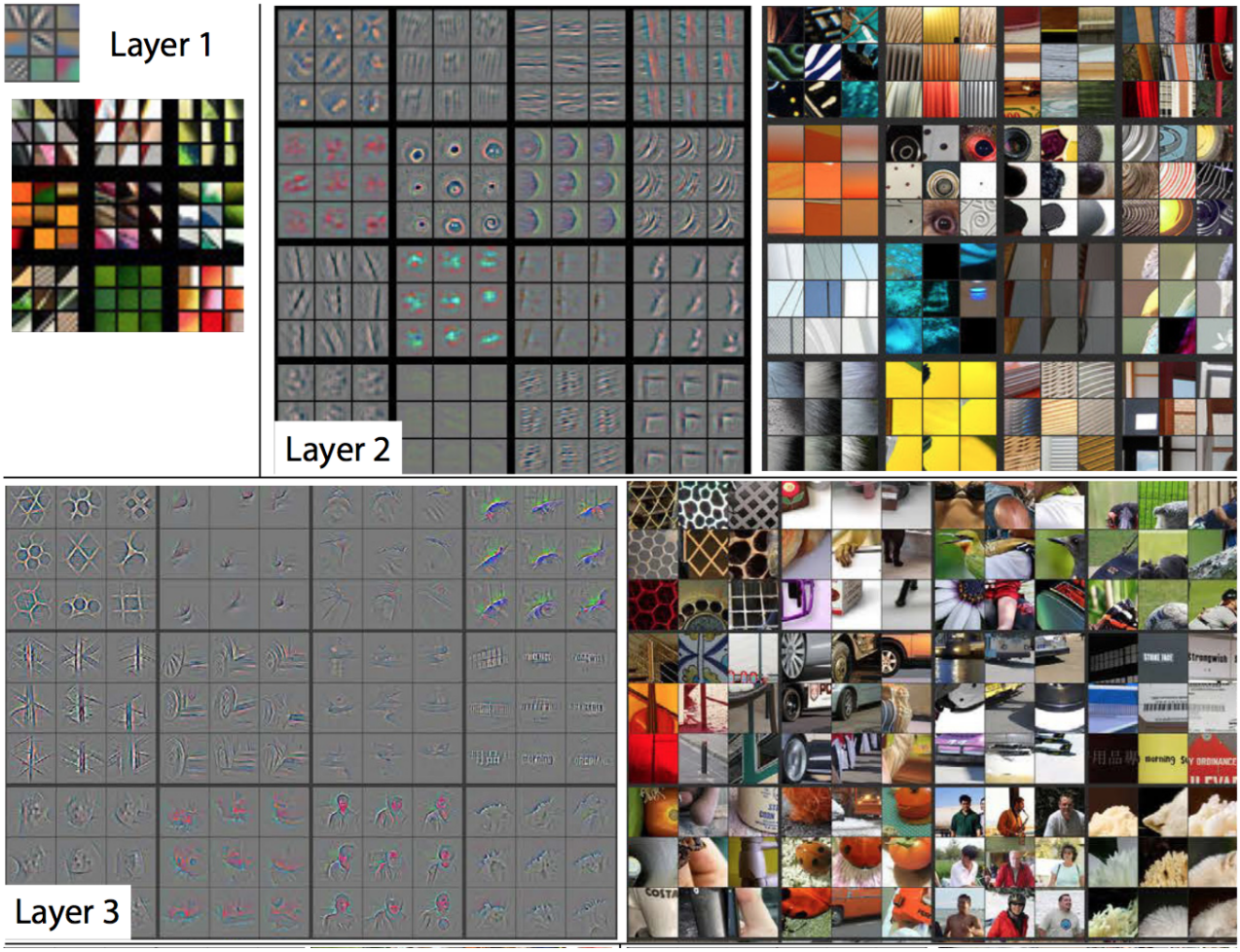

Figure 3.5. Visualization of a CNN generated using the de-convolution method [64]. Gray images in the left are the top nine activations in a random subset of neurons across validation data, projected back to image space. In the right are corresponding image patches.

(training data) is too large. The users will not be able to remember a lot representative local explanations to form a global understanding.

Model-aware explanations. The first attempt to understand a complex classifier globally is done by Féraud and Clérot [14]. They partition the hidden representation space through clustering the representation of the whole training set, where each cluster represents a semantic concept learned by the classifier. To qualitatively understand a CNN, Erhan *et al*. [13] propose the activation maximization method. Each neuron in the CNN can be explained using an image patch that will maximize its activation. To provide conceptual meanings of the explanation, Bau *et al*. [5] align hidden units with human-understandable concepts (objects) through a dissection process. However, these methods often require explorations over multiple nodes or neurons. The big picture is often neglected. Additionally, a common issue is that it is hard to compare these methods due to the lack of an evaluation framework.

In summary, there are two common strategies to make a classifier explainable. First, we develop simpler or sparser classifiers that can meet the performance requirements. Second,

we build another human-understandable interface for explanation on top of a classifier. Both strategies are useful in different scenarios.

However, an important, but neglected aspect of existing methods is the human. Few have paid attention to model human. Most of them study the classifiers and develop techniques for explainability and then argue that their methods help humans to understand the classifier, without studying how humans exactly response to the results generated by these techniques.

Another related problem raised by Doshi-velez and Kim [10] is the lack of the evaluation methods for explainability. Without a rigorous evaluation, it is hard to compare which method is better in a certain setting. It will also be infeasible to clarify the gap of current research and direction for future research.

CHAPTER 4

VISUALIZATION FOR EXPLAINABLE CLASSIFIERS

As discussed at the end of Chapter 3, current research only focuses on one subject of the problem of explainable classifier and neglects the other subject – human. Thus, we study explainable classifier from the aspects of visualization and human computer interaction in this chapter. Broadly speaking, the visualization for explainable classifiers can be viewed as a special case of algorithm visualization or software visualization. The former aims to provide better understanding of algorithms for education purposes in computer science. The latter focus on assisting developers and operation engineers for the development and maintenance of complex software. Here, the subject of visualization is the classifiers, which can be treated as algorithms learned from the data, or complex systems that need assistance in understanding.

We view the development and the operation of an intelligent system as a system engineering problem, and divide the life cycle of an intelligent system into different stages. The classification system can be treat as a specification of the general intelligent system. Then, we identify the current issues in explaining classifiers and discuss the research opportunities of visualization regarding different stages.

4.1 Life Cycle of an intelligent system

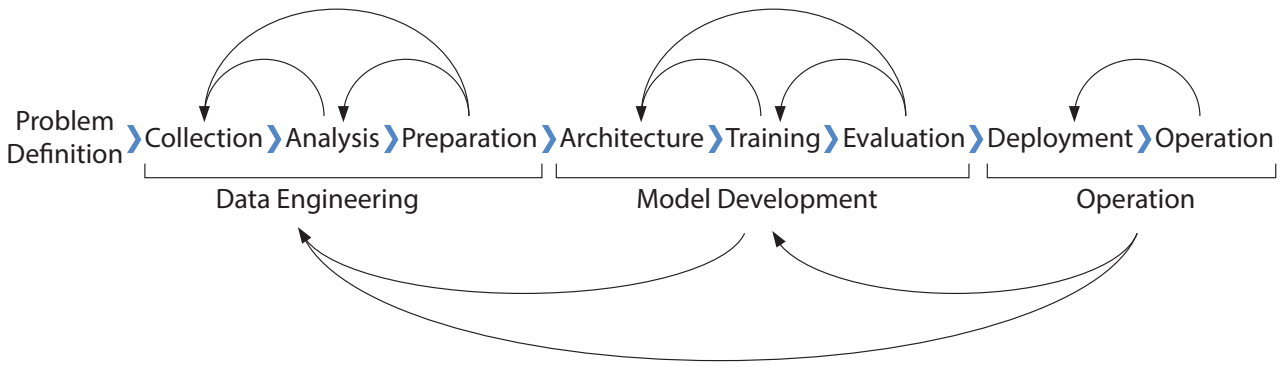

Figure 4.1. The life cycle of a classifier.

A classification system can be thought as a specialized case af an intelligence system. An intelligent system is developed to perform certain tasks with artificial intelligence (here we only consider data-driven systems). The development of a data-driven intelligent system is an iterative process. In this survey, the entire life cycle of an intelligent system is divided into three major stages (data engineering, model development, and operation) and eight sub-procedures (see Figure 4.1). This definition is formed based on the cross-industry standard process for data mining [61], a professional advice from Gartner, Inc. [47], the life cycle for expert system [25], and the machine learning workflow of Google Cloud[1].

The first stage, data engineering, is defined to include any procedures that are data-related, namely, data collection, exploratory analysis, data preparation. The second stage, model development, includes procedures such as designing the architecture of the classifier (*e.g.*, what type of model to use and parameters), training the model using data prepared, and evaluating whether the model meets certain requirements. After developing a classifier, the model is deployed, and in certain cases, is operated by some people. As we can see from Figure 4.1, there are back-links from each stage/step to its previous stages/steps. This is the nature development. For example, we have a model with unsatisfactory performance after the training. This might due to model used is not suitable for certain tasks (go back to architecture setting), or it is because the volume of the data is too small (go back to collect more data). Similar problems might occur in other stages or procedures, which force us to go back and improve.

Visualization and visual analytics systems are semi-automatic solutions at different stages to make a classifier more explainable. In the stage of data engineering, data visualization can help humans explore the data and get a qualitative sense of the nature of the data. Since the training of a classifier is actually extracting information from the training data, with more knowledge of the data in mind, humans (*i.e.*developers or data scientists) can better understand if a failure results from the quality or volume issues of the data. At the second stage, visual analytics systems serve as development tools, which make the development more transparent and understandable. When designing or selecting model architectures, visual analytics systems can help humans better understand the characteristics of different classifiers, and even inspire improvements in the architecture. Visual diagnosing tools can help identify the problems in the training process and improve the debugging efficiency. For evaluations, visual analytics systems can help compare different classifiers and qualitatively evaluate the robustness and fairness of a classifier. At the last stage, when a classification system is deployed, visualization can help explain the inner workings of the system to end-

[1] `https://cloud.google.com/ml-engine/docs/ml-solutions-overview`

users. For routine operations, visualization can better explain the predictions of the system, which make the monitoring and management easier. Also note that, visualization can used in the life-cycle for other purposes instead of explainability. For example, monitoring the training process by plotting loss curves, or visualization for crowd sourcing to collect data with better quality.

4.2 Visualization for Data Understanding

In the stage of data engineering, visualization can be used mainly in the procedure of data analysis to assist humans' understanding of the data. Data plays an important role in the success of machine learning advances. A trained classifier can be viewed as a machine that has extracted the information in the training data and abstracted the information as its parameters. Different classifiers are suitable for datasets with different characteristics. Besides, a classifiers performance is largely depend on the quality of its training data. Thus, understand the data is the first step to understand a classifier.

There is a long history of research in visualization for exploratory data analysis. When comes to the classification problem, the data involved is **multi-dimensional data** with categorical labels, usually having the type of images, texts and categorical data. There are mainly three tasks for understanding the data: statistical analysis, dimensionality reduction, and dataset diagnostics.

Statistical analysis. The purpose of statistical analysis is to summarize statistical features of the datasets and provide users with a statistical understanding. One of the most influential visualization design is the box plot, which summarizes the key statistics of each feature in a box-like visualization. A survey on the design of box plots is provided by Potter *et al*. [41]. As shown Figure 4.2, (a) presents the key features of a box plot. The box plots can be enhanced to embed more information such as confidence intervals in (e) and distribution densities in (f-i), providing more comprehensive summaries than the simple box plot. However, the datasets used in classification systems have become increasingly complex, with very high dimensions. Navigating through hundreds of box plots not necessarily provide insights about the datasets.

Dimensionality reduction. In exploratory analysis, dimensionality reduction is usually applied for understanding how the data is distributed in the input space. Thus, projection, including principle component analysis (PCA), multidimensional scaling (MDS) [24], and t-distributed stochastic neighbor embedding (t-SNE) [34], is the most common dimensionality reduction technique used in this stage. The role of visualization is to present and augment

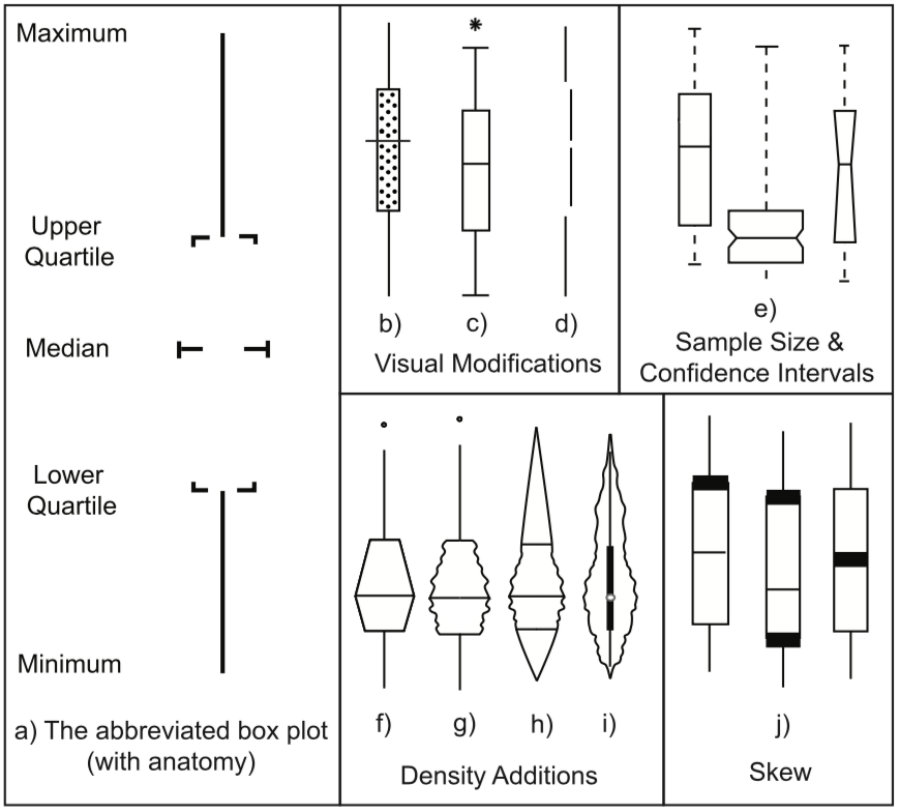

Figure 4.2. Variations on the box plot [41]. a) Abbreviated box plot. b) Range plot. c) Box plot. d) Interquartile plot. e) Variable width and notched box plots expressing sample sizes and confidence levels. f) Hist plot. g) Vase plot. h) Box-percentile plot. i) Violin plot. j) Skew and modality plots.

the results of dimensionality reduction. A most recent example is the Embedding Projector [51], which embed human interpretable labels (*e.g.*, images with categorical colors) in the scatter plots achieved by dimensionality reductions. By navigating the projected plot, users can better understand the relationship of different instances and identify clusters and outliers. For example, as shown in Figure 4.3, we can see that digits "3", "5" and "8" are actually very close to each other, and the classifier might have difficulties to distinguish them.

Dataset diagnostics. Diagnostics is an important and comprehensive task for improving the quality of the dataset. Given that different datasets usually have different characteristics, there is few automatic methods that can solve all the problems. Thus, human is required to be in the loop. The visualization techniques involved may vary for different datasets. Most commonly used are scatter plots, stacked diagrams, and their variations. An example visual dataset diagnostics tool for machine learning is the Facets[2].

As discussed above, visualization techniques can help users develop understandings of the distribution, characteristics, and possible challenges of the datasets. These insights can guide the implementation and evaluation in the second stage of model development. In this

[2] `https://github.com/PAIR-code/facets`

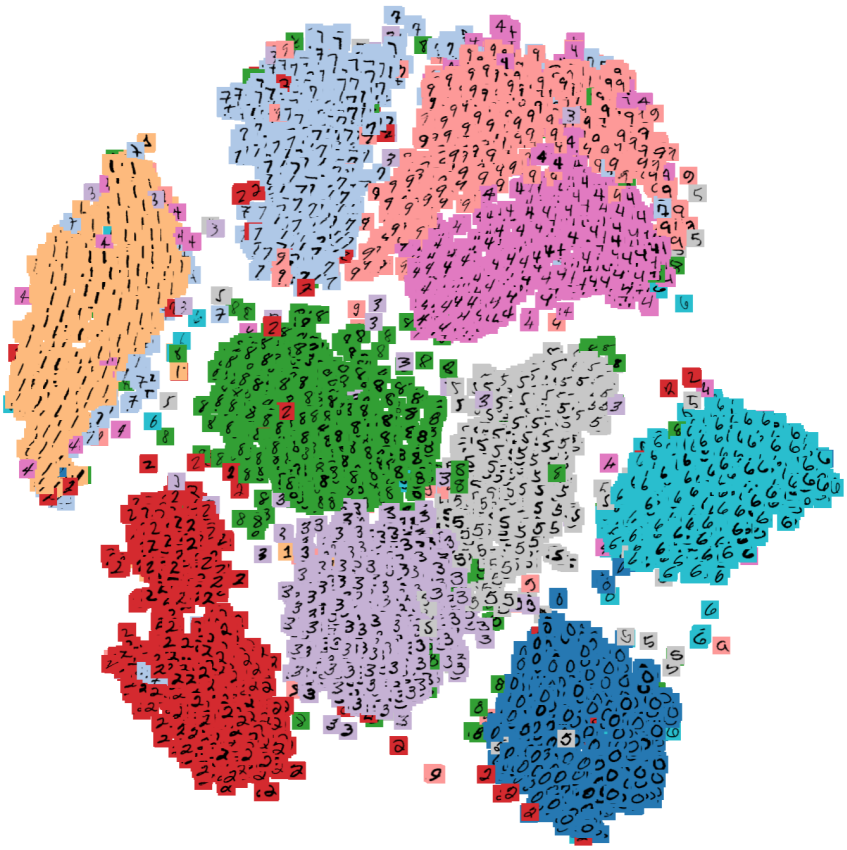

Figure 4.3. The t-SNE projection of the MNIST dataset, colored by the labeled digits.

sense, it is easier for developers to understand the behavior of classifiers, which makes the classification system more explainable.

There are two research issues for visualization in the stage of data engineering. The first is the scalability issue. The volume of datasets can be very large (*e.g.*, millions of instances), which brings about challenges in real-time visualization rendering and interaction designs. The second is the lack of comprehensive visual analytics system that are designed specifically for the development of intelligent systems. A developer will have to switch among different tools to perform different tasks, which need a lot of redundant work for format transformations.

4.3 Visualization for Model Development

There is a surge of interest to use visualization in the stage of model development for explainable classifiers. We summarized three tasks for visualization, namely, model understanding, model diagnosing, and assessment and comparison, which are corresponding to the three stages: architecture design, training, and evaluation.

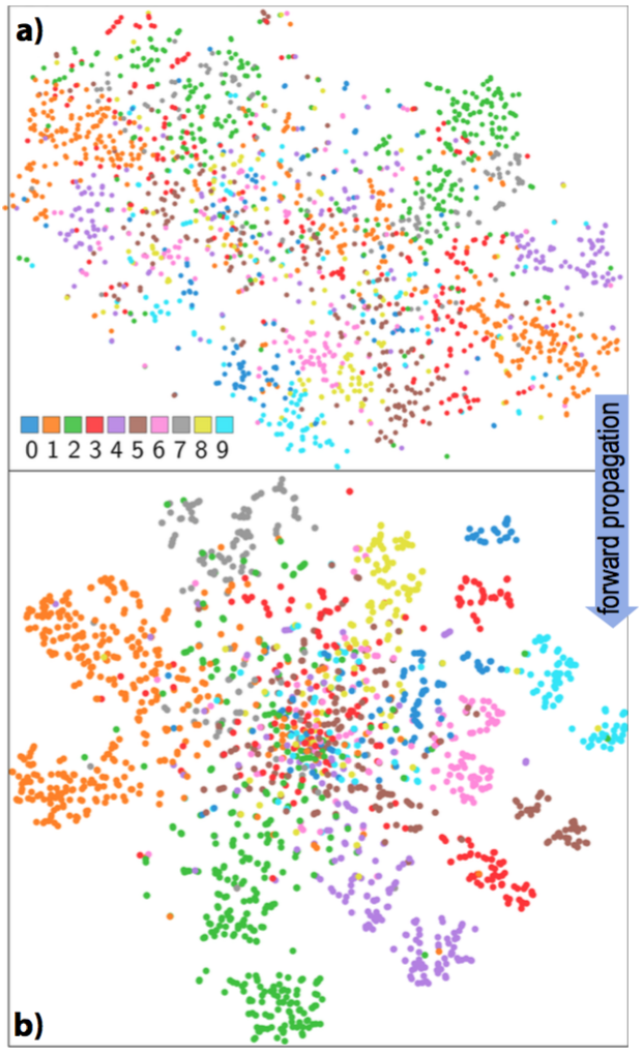

Figure 4.4. Projection of the MLP hidden layer activations after training [44], SVHN test subset. a) First hidden layer. b) Last hidden layer.

4.3.1 Understanding

The purposes of this task is to help researchers and developers better understand the characteristics and working mechanisms of different classifiers so that they can design a better one or choose a better one. The selection and design of the architecture of a classifier always require a strong expertise in machine learning, which needs cumulative experience of trial and error. For classifiers whose characteristics are still under studying (*e.g.*, neural networks), such knowledge is even harder to achieve. Thus, most of the related work using visualization techniques for understanding classifiers focuses of neural networks. Considering the taxonomy of explanations, these visualizations mostly fall into model-aware global explanations, which provide a general sense of how a classifier behaves globally. Unlike pure explanation generation techniques, the visual analytics systems often embed extra information of the classifiers that boost understanding. These techniques can be roughly categorized into two: representation-based and component-based.

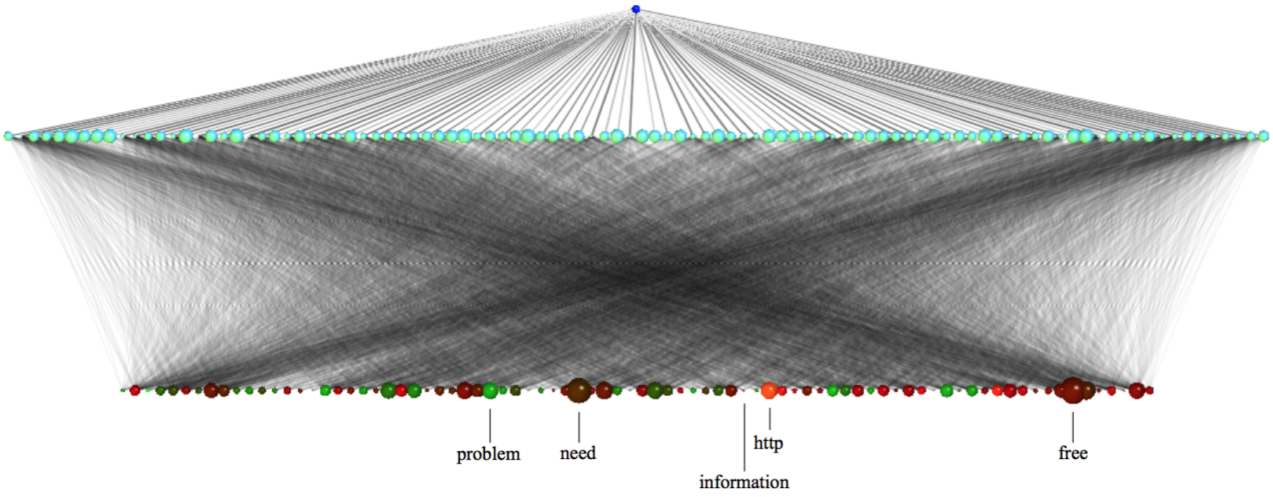

Figure 4.5. A spam classifier, where each input is a term in the email. The input data of the network is a subset of the spam in the training set [57].

Representation-based methods visualize how a given set of instances is represented in the hidden layer or the final output of the classifier. As shown in Figure 4.4, Rauber *et al*. [44] project the representations of a subset of instances in different layer, and verified that each layer of the network transform its input space to a more separable one. Unlike component-based methods, representation-based methods actually treat a model or a layer as a black box, and visualize how the representation space is transformed. This technique has been integrated in visual diagnostics systems such as ActiVis [20] and DeepEyes [40]. However, they are unable to analyze the detail information embedded in the classifiers.

Component-based methods visualize the working mechanisms of specific components of a classifier. Tzeng and Ma directly visualize a multi-layer perceptron as a graph using the node-link layout. As shown in Figure 4.5, though this visualization explained how important different term are for the neural network, it is visually cluttered and cannot reveal how neurons are combined to perform the classification. To provide more scalable visualization, Liu *et al*. develop the CNNVis [32], which use clustering and edge aggregation to provide a more compact visualization for CNNs. The representative image patch that activating each neuron clusters are also embedded in the visualization (see Figure 4.6). There is also some work using visualization to understand RNNs. Karpathy *et al*. [21] visualize the value of a hidden unit along a sequence as a heatmap and find some hidden units have clear semantics. Strobelt *et al*. [53] develop the LSTMVis based on parallel coordinates to identify dynamic patterns in the hidden state of an RNN. Ming *et al*. [38] develop the RNNVis, which is based on co-clustering and word clouds, and find that the hidden units can form semantics clusters from the text data.

These visualization techniques are proved to be able to provide insights and useful ex-

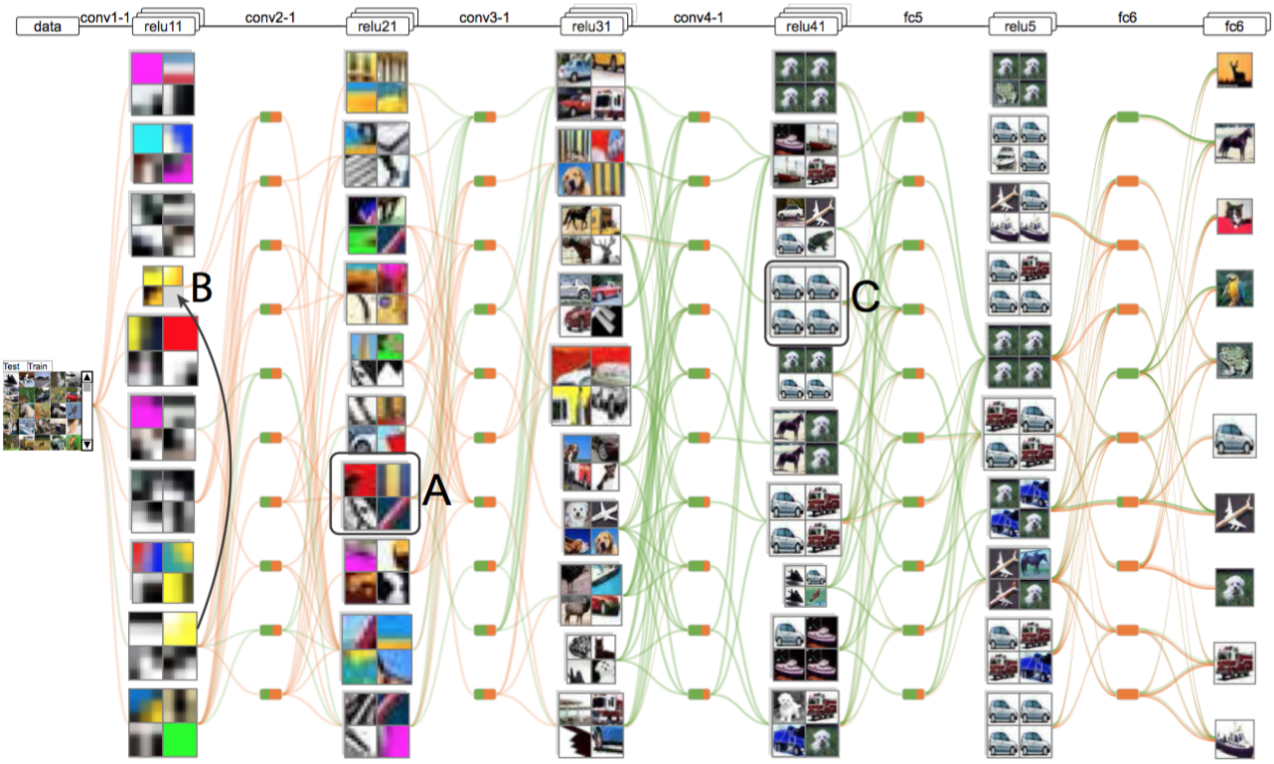

Figure 4.6. A visualization of a CNN with a large number of layers and neurons using neuron clustering and edge aggregation [32].

planations of different classifiers. However, they are developed in a model-specific manner, which means they are difficult to generalize as common knowledge. There is also a lack of a unified process to visually analyze and understand a classifier, and a more rigorous measurement of how well a visualization helps explain the characteristics of a complex classifier.

4.3.2 Diagnosis

The diagnosis in the stage of model development mainly includes two parts: code diagnosis, and training diagnosis. Note that the diagnosis does not contribute to the explainability of a classifier directly. However, they can be thought to enhance the explainability of the code and training process of a classifier. Besides, the training difficulty varies as the architecture changes, which is also a part of characteristics of a classifier.

Code diagnosis. Wongsuphasawat *et al*. [62] design and implemented a scalable graph layout for the data flow graph of any machine learning model and integrate the graph visualization in TensorFlow (see Figure 4.7). The code defining the data flow graph can be automatically converted to a visualization, which helps developers validate the correctness of the code. A similar computation graph overview is proposed in the ActiVis, a visual exploration system for machine learning models developed by Kahng *et al*. [20].

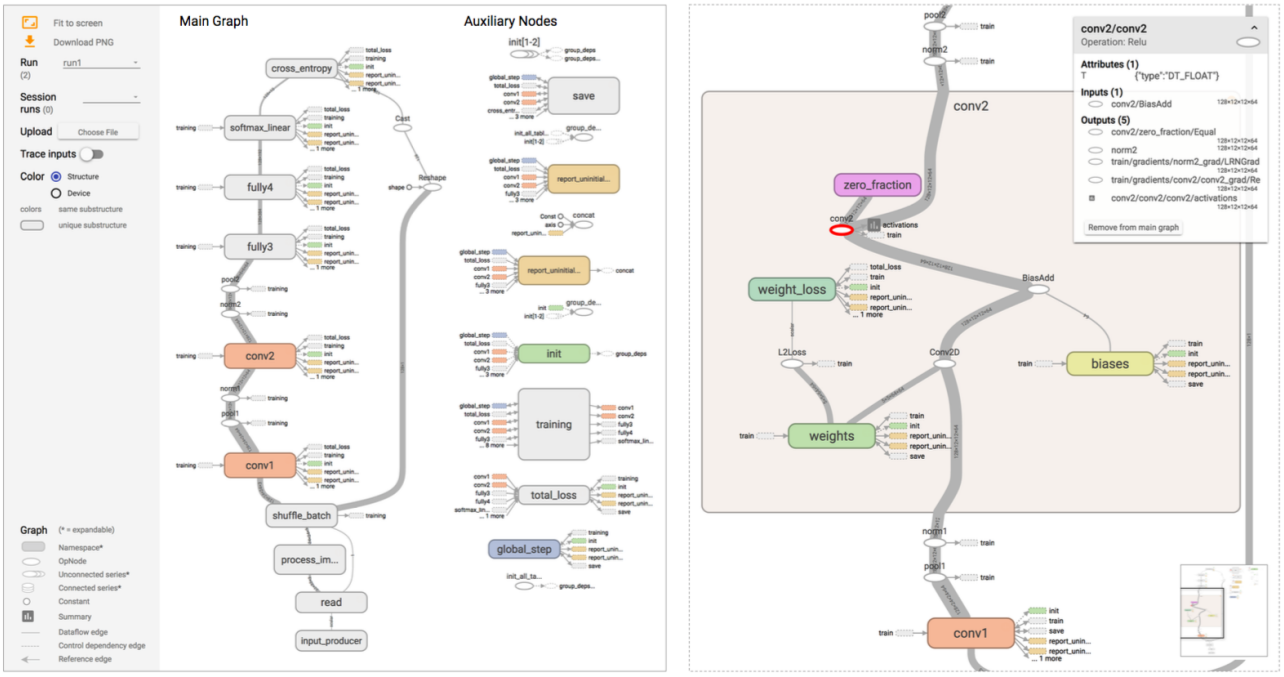

Figure 4.7. The TensorFlow Graph Visualizer shows a CNN image classifier [62]. (a) An overview displays a dataflow between groups of operations, with auxiliary nodes extracted to the side. (b) Expanding a group shows its nested structure.

Training diagnosis. There are also visual analytics systems developed to understand the training process of specific models. Unlike standard testing for software engineering, there is no accurate definition of the failure of a training. The failure of a training is a relative concepts. It is difficult to determine if a high loss is due to local optimum, the limitation of the model, or the quality of the dataset. Providing visual hints or indicator can help developers identify possible cause of failed training. The diagnosis of the training process is usually provided by sampling multiple snapshots during training. Liu *et al*. [30] develop the DGMTracker, which tracks the training process of the deep generative models (DGM) (*e.g*., generative adversarial model). They cache the training dynamics (*e.g*., activation changes) and visualize the training dynamics to help identify possible cause of a failed training. Pezzotti *et al*. [40] proposed a progressive method that support the identification of layers that learn stable patterns and superfluous filters or layers. Visual analytics systems are also proved to be useful for tree boosting methods [31]. These visual analytics system are also model-specific and are difficult to generalize. There still needs further research in understanding the characteristics of the loss function to guide better the development of visualization techniques for training diagnosis.

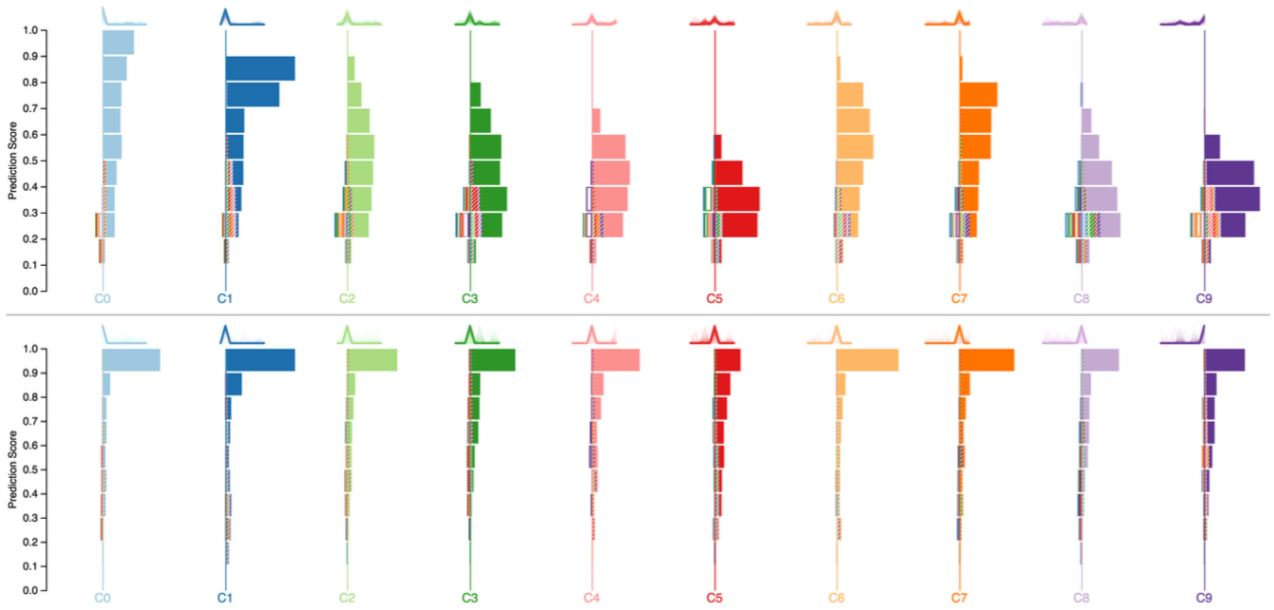

Figure 4.8. Squares [45] displaying the performance of two classifier (top: random forest, bottom: SVM) trained on the MNIST dataset. They yield the same accuracy of 0.87 , but show vastly different score distributions.

4.3.3 Assessment and Comparison

The purpose of evaluation is to determine if a trained classifier meets certain standards for deployment, and to select the best one among multiple classifiers. The evaluation is, to some extent, closely related to the training diagnostics. If a classifier fails the evaluation, then we will go back to adjust parameters for another training or refine its architecture. Visualization can also be used to assess and compare the performance of classifiers comprehensively (*e.g.*, identify which classes the classifier is not good at). The visualization here also provide certain level of global explanations – explaining a classifier using the statistics of its outputs of the test data.

Amershi *et al*. [1] develop the ModelTracker to support the interactive performance analysis for binary classifier using stacked diagram. The ModelTracker combines the performance statistics and data inspection functions, which helps engineers evaluate the performance and identify problematic classifications at the same time. Ren *et al*. extend it to the Squares [45], which support the performance analysis for multi-class classifiers. As shown in Figure 4.8, the Squares is used to compare the performance of two classifiers, a random forest and an SVM, on the MNIST dataset. We can easily see that the SVM has much higher confidence than the random forest, though they have the same accuracy.

Currently, most evaluations only address accuracy and computation cost. As discussed in Chapter 2, we may need to meet other requirements such as fairness, robustness and explainability in certain applications. As discussed above, visualization can be used to en-

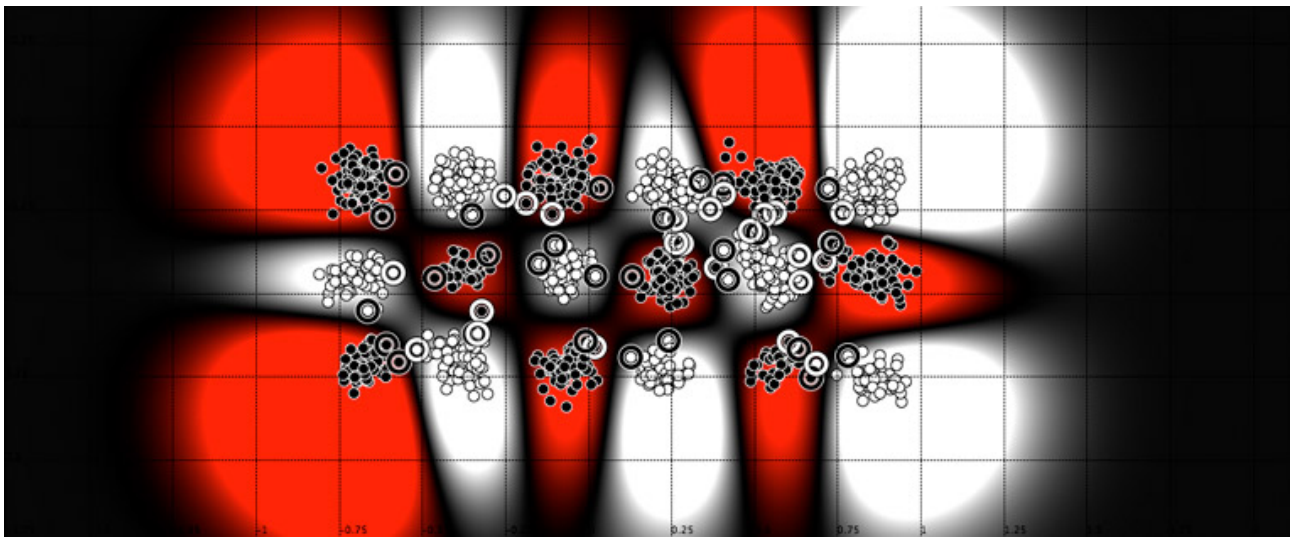

Figure 4.9. A visualization from MLDemos, showing the decision space of a binary SVM classifier using RBF kernel. White and red denotes different classes, and black represents uncertainty.

hance the explainability of a complex classifier. By explaining the reasoning of the classifier, we can identify possible discriminate classifications of the model. It is also possible to qualitatively evaluate the robustness of the classifier by verify if the classifier's reasoning matches that of a human or it is using unexpected reasoning.

Except for the issues stated above, there are three general research issues for visualization in the model development stage. The first issue is the scalability of the visualization design. Many visualizations perform well in toy datasets such as MNIST. However, they will easily become overwhelming if the number of classes increases to a few hundreds. The second is the lack of consensus in the evaluation. The target of these visualization techniques is explaining and understanding, which is also subjective. It is difficult to design appropriate evaluation tasks for users to perform. The third is the need for a comprehensive toolkit that covers different procedures fluently. For example, TensorBoard[3] is a toolkit for visualizign and debugging machine learning models, which has been closely integrated in TensorFlow, but it is still at an early stage and has a limited set of functions.

4.3.4 Communication and Education

Another related task, although not a necessary one in the model development stage, is the communicating and teaching of classification models. This task is related to the task of understanding. However, the focus of this task is not for a better understanding of the working mechanisms of a classifier, but for a better presentation of a classifier.

[3] `https://github.com/tensorflow/tensorboard`

With the growing complexity of model architectures, it often requires much effort to understand the composition and computation steps of a new classifier. It also requires a lot of work from the author to draw a diagram, which may neglect useful information for the readers. Netscope[4] is a visualization tool that converts the code of a model written in Caffe[5] to a node-link graph. Which helps quickly understand the topology of a model. However, the visualization will become too large if the model is complex (*e.g.*, a CNN with a hundred layers), and it does not show extra information (*e.g.*, optimizing method) about the model. TensorBoard utilizes the data flow graph visualization developed by Wongsuphasawat [62], which is interactive and more scalable.

For new comers who want to learn, there will also be a large overhead to understand the computation of different components. Some visualizations have been developed to help people learn. Basilio Noris develops a visualization tool, MLDemos[6] for understanding the behavior of various machine learning algorithms. As shown in Figure 4.9, MLDemos maps the output of a classifier using multiple colors back to the input space, and helps people learn the characteristics of different classifiers. Tensorflow Playground[7] is an interactive visualization, which helps people understand the behavior of a neural network, using similar techniques.

4.4 Visualization for Operation

The operation and management after deploying a classification system is an important stage of the whole life cycle, but little attention has been paid to this stage. There are two common tasks that needs explainability: trust establishment and inspection.

4.4.1 Trust Establishment

The first challenge after the classifier has been deployed is how to establish users' trust in the machine. Ribeiro *et al*. [46] have discussed the importance of a user's trust: if the user does not rust a model, he/she will not use it. If a user understand why the classifier predict class A but not B, if he/she understand when the classifier performs well and when it is likely to fail, if he/she can figure out the possible reasons for a failure, he/she will trust the classifier. Thus, the foundation of trust is understanding, which can be achieved through explanation.

[4] http://ethereon.github.io/netscope

[5] http://caffe.berkeleyvision.org/

[6] http://mldemos.epfl.ch/

[7] http://playground.tensorflow.org/

Ribeiro *et al*. designed a inspiring solution for explanation, that is, locally approximate the complex classifier using an interpretable one, a linear classifier. Then the linear classifier is used to generate a local explanation (*e.g*., a list of words for a document classifier). Although this method introduces another classifier, whose error is not guaranteed, they showed that users indeed have more confident and trust in the classifier. Though explanation for trust establishment is promising, there are still unsolved issues. How to make sure the generated explanation is fidel to the model? How to evaluate whether the user has learned the correct message from the explanation? These should be addressed in future research.

4.4.2 Monitoring

After the end-users have established their trust in a classifier, they still needs a human-friendly interface to inspect or monitor the classifier. This need can be satisfied by providing faithful local explanations for specific predictions. The need for monitoring is not solely for preventing possible failure. For example, if we have developed a system for hospitals to detect early signs of lung cancer from CT scans, a good explanation not only helps the doctor correct possible false positives, but also provide the doctor extra information for deciding whether further examinations are needed.

CHAPTER 5

CONCLUSION

Explainability is a critical but often-overlooked property for intelligent systems. In this survey, we review techniques that provide explainability to classifiers, with a special focus on visualization. We first discuss the concept and definition of explainability of a classifier. Two types of techniques that provide explainability to classifiers are summarized and discussed. Then based on the life cycle of intelligent systems, we discuss the role of visualization in different stages, and how visualization can be used to improve explainability.

The research in the explainability of classifiers is still a growing discipline. There is no consensus on the definition of explainability in the context of supervised learning. There is few rigorous evaluation methods of the explainability of a classifier or a explanation of a classifier. Early work treats the explainability as the "simplicity" and always focuses on balancing the trade-offs between performance and simplicity. Now there is a new trend of building an explanatory interface between human and the underlying model to enhance explainability while maintaining the performance. A few challenges and research issues are summarized as follows.

Rigorous theory of explainability. There are plenty of unsolved questions in the the seemly intuitive concept of explainability. How to rigorously define explainability in the context of machine learning or artificial intelligence? What is explanation? How to evaluate whether an explanation is good or not? How can we model the variation and uncertainty of human in the explanatory interface? Based on the theory of explainability, a further issue that needs to address is the **evaluation** of explainability and the quality of explanations. If a metric based evaluation is inapplicable, there still need guidelines for system designs. There is some work in cognitive science studying the function, structure of explanation, and the role that explanation plays in perception, cognition, and learning. It may be promising to learn from the theory from cognition science. Still, it requires the efforts from both cognitive science and computer science.

Applications. Theory comes from practice. Although there are toy examples, showing how explainability helps design better models, avoid possible bias, and enhance users' trust, we still lack knowledge on the design challenges of a explainable interface in real-world applications. Is it possible to let the machine explain its learned knowledge to a human? There

are few successes or failures in real-world applications that we can learn from. Another neglected stakeholder is the end-users who actually use or are influenced by the technology developed based on AI. How do they value explainability during the use of an intelligent system? How can we use explanatory techniques to improve their using experience? The research at this end might be able to impact more people.

BIBLIOGRAPHY

[1] *ModelTracker: Redesigning Performance Analysis Tools for Machine Learning*. ACM âĂŞ Association for Computing Machinery, April 2015. [Online]. Available: https://www.microsoft.com/en-us/research/publication/modeltracker-redesigning-performance-analysis-tools-for-machine-learning/

[2] B. Alsallakh, A. Jourabloo, M. Ye, X. Liu, and L. Ren, "Do convolutional neural networks learn class hierarchy?" *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[3] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," *arXiv preprint cs/0006013*, 2000.

[4] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, no. 7, p. e0130140, 2015.

[5] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Computer Vision and Pattern Recognition*, 2017.

[6] K. D. Bock, K. Coussement, and D. V. den Poel, "Ensemble classification based on generalized additive models," *Computational Statistics & Data Analysis*, vol. 54, no. 6, pp. 1535 – 1546, 2010.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[8] M. H. Brown, "Algorithm animation," Ph.D. dissertation, Providence, RI, USA, 1987, uMI Order No. GAX87-15461.

[9] W. Clancey, "The epistemology of a rule-based expert system: A framework for explanation," Stanford, CA, USA, Tech. Rep., 1981.

[10] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv*, 2017. [Online]. Available: https://arxiv.org/abs/1702.08608

[11] T. Downs, K. E. Gates, and A. Masters, "Exact simplification of support vector solutions," *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 293–297, 2001.

[12] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 4, pp. 325–327, April 1976.

[13] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," University of Montreal, Tech. Rep. 1341, Jun. 2009, also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.

[14] R. Féraud and F. Clérot, "A methodology to explain neural network classification," *Neural Netw.*, vol. 15, no. 2, pp. 237–246, Mar. 2002.

[15] Google Inc. (2017) PAIR | people + ai research initiative. [Online]. Available: http://ai.google/pair

[16] D. Gunning, "Explainable artificial intelligence (XAI)," *Defense Advanced Research Projects Agency (DARPA)*, 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[18] C. Hempel and P. Oppenheim, "Studies in the logic of explanation," *Philosophy of Science*, vol. 15, no. 2, pp. 135–175, 1948.

[19] L. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, "Generating visual explanations," *CoRR*, vol. abs/1603.08507, 2016. [Online]. Available: http://arxiv.org/abs/1603.08507

[20] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau, "Activis: Visual exploration of industry-scale deep neural network models," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[21] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," in *International Conference on Learning Representations (ICLR) Workshop*, 2016.

[22] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 4, pp. 580–585, July 1985.

[23] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 1885–1894.

[24] J. B. Kruskal and M. Wish, *Multidimensional scaling*. Sage, 1978, vol. 11.

[25] A. J. La Salle and L. R. Medsker, "The expert system life cycle: What have we learned from software engineering?" in *Proceedings of the 1990 ACM SIGBDP Conference on Trends and Directions in Expert Systems*, ser. SIGBDP '90. New York, NY, USA: ACM, 1990, pp. 17–26.

[26] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[27] B. Letham, C. Rudin, T. McCormick, and D. Madigan, "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model," *Ann. Appl. Stat.*, vol. 9, no. 3, pp. 1350–1371, 09 2015.

[28] J. Li, X. Chen, E. Hovy, and D. Jurafsky, "Visualizing and understanding neural models in nlp," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 681–691.

[29] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy, "Sparse convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 806–814.

[30] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu, "Analyzing the training processes of deep generative models," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[31] S. Liu, J. Xiao, J. Liu, X. Wang, J. Wu, and J. Zhu, "Visual diagnosis of tree boosting methods," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[32] S. Liu, X. Wang, M. Liu, and J. Zhu, "Towards better analysis of machine learning models: A visual analytics perspective," *Visual Informatics*, vol. 1, no. 1, pp. 48 – 56, 2017.

[33] T. Lombrozo, "The structure and function of explanations," *Trends in Cognitive Sciences*, vol. 10, no. 10, pp. 464 – 470, 2006.

[34] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[35] T. Munzner, *Visualization analysis and design*. CRC press, 2014.

[36] W. J. Murdoch and A. Szlam, "Automatic rule extraction from long short term memory networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[37] R. Neches, W. Swartout, and J. Moore, "Enhanced maintenance and explanation of expert systems through explicit models of their development," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 11, pp. 1337–1351, Nov 1985.

[38] U. H. M. of Recurrent Neural Networks, "Yao ming and shaozu cao and ruixiang zhang and zhen li and yuanzhe chen and yangqiu song and huamin qu." in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, 2017.

[39] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002, pp. 79–86.

[40] N. Pezzotti, T. HÃűllt, J. v. Gemert, B. P. F. Lelieveldt, E. Eisemann, and A. Vilanova, "Deepeyes: Progressive visual analytics for designing deep neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[41] K. Potter, J. Kniss, R. Riesenfeld, and C. R. Johnson, "Visualizing summary statistics and uncertainty," in *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization*, ser. EuroVis'10. Chichester, UK: The Eurographs Association & John Wiley & Sons, Ltd., 2010, pp. 823–832. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2009.01677.x

[42] B. Price, I. Small, and R. Baecker, "A taxonomy of software visualization," vol. ii. IEEE Publishing, 1992, pp. 597–606.

[43] J. R. Quinlan, "Simplifying decision trees," *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

[44] P. E. Rauber, S. G. Fadel, A. X. FalcÃčo, and A. C. Telea, "Visualizing the hidden activity of artificial neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 101–110, Jan 2017.

[45] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams, "Squares: Supporting interactive performance analysis for multiclass classifiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 61–70, Jan 2017.

[46] M. T. Ribeiro, S. Singh, and C. 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*. New York, NY, USA: ACM, 2016, pp. 1135–1144.

[47] C. Sapp. (2017) Preparing and architecting for machine learning. [Online]. Available: https://www.gartner.com/doc/3573617/preparing-architecting-machine-learning

[48] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[49] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *International Conference on Learning Representations (ICLR) Workshop*, 2014.

[50] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *CoRR*, vol. abs/1706.03825, 2017.

[51] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, "Embedding projector: Interactive visualization and interpretation of embeddings," *arXiv preprint arXiv:1611.05469*, 2016.

[52] J. T. Stasko, "Tango: a framework and system for algorithm animation," *Computer*, vol. 23, no. 9, pp. 27–39, Sept 1990.

[53] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[54] W. Swartout, C. Paris, and J. Moore, "Explanations in knowledge systems: design for explainable expert systems," *IEEE Expert*, vol. 6, no. 3, pp. 58–64, June 1991.

[55] M. Tan, L. Wang, and I. W. Tsang, "Learning sparse svm for feature selection on very high dimensional datasets," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 1047–1054.

[56] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[57] F. Y. Tzeng and K. L. Ma, "Opening the black box - data driven visualization of neural networks," in *VIS 05. IEEE Visualization, 2005.*, Oct 2005, pp. 383–390.

[58] B. Ustun and C. Rudin, "Supersparse linear integer models for optimized medical scoring systems," *Machine Learning*, vol. 102, no. 3, pp. 349–391, Mar 2016.

[59] F. Wang and C. Rudin, "Falling Rule Lists," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Lebanon and S. V. N. Vishwanathan, Eds., vol. 38. San Diego, California, USA: PMLR, 09–12 May 2015, pp. 1013–1022.

[60] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille, "A bayesian framework for learning rule sets for interpretable classification," *Journal of Machine Learning Research*, vol. 18, no. 70, pp. 1–37, 2017.

[61] R. Wirth and J. Hipp, "Crisp-dm: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, 2000, pp. 29–39.

[62] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. ManÃl', D. Fritz, D. Krishnan, F. B. ViÃl'gas, and M. Wattenberg, "Visualizing dataflow graphs of deep learning models in tensorflow," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[63] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[64] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[65] L. Zintgraf, T. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," in *International Conference on Learning Representations (ICLR)*, 2017.