Title
Interpretable deep learning for natural language processing

Author
Murdoch, William James

Publication Date
2019

Peer reviewed|Thesis/dissertation

Interpretable deep learning for natural language processing

by

William James Murdoch

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Bin Yu, Chair
Professor David Bamman
Professor Fernando Perez

Spring 2019

Interpretable deep learning for natural language processing

Abstract

Interpretable deep learning for natural language processing

by

William James Murdoch

Doctor of Philosophy in Statistics

University of California, Berkeley

Professor Bin Yu, Chair

Machine-learning models have demonstrated great success in learning complex patterns that enable them to make predictions about unobserved data. In addition to using models for prediction, the ability to interpret what a model has learned is receiving an increasing amount of attention. These interpretations have found a number of uses, ranging from providing scientific insight to auditing the predictions themselves to ensure fairness with respect to protected categories like race or gender. However, there is still considerable confusion about the notion of interpretability. In particular, it is currently unclear both what it means to interpret a ML model, and how to actually do so.

In the first part of this thesis, we address the foundational question of what it means to interpret a ML model. In particular, it is currently unclear what it means to be interpretable, and how to select, evaluate, or even discuss, methods for producing interpretations of machine-learning models. We aim to clarify these concerns by defining interpretable machine learning and constructing a unifying framework for existing methods which highlights the under-appreciated role played by human audiences. Within this framework, methods are organized into two classes: model-based and post-hoc. To provide guidance in selecting and evaluating interpretation methods, we introduce three desiderata: predictive accuracy, descriptive accuracy, and relevancy. Using our framework, we review existing work, grounded in real-world studies which exemplify our desiderata, and suggest directions for future work.

The second through fourth parts introduce a succession of methods for interpreting predictions made by neural networks. The second part focuses on Long Short Term Memory networks (LSTMs) trained on question-answering and sentiment analysis, two popular tasks in natural language processing. By decomposing the LSTM's update equations, we introduce a novel method for computing feature importance scores of specific inputs for determining the output of an LSTM. In order to verify the output of our method, we use the introduced scores to search for consistently important patterns of words learned by state of the art LSTMs on sentiment analysis and question answering. This representation is then quantitatively validated by using the extracted phrases to construct a simple, rule-based classifier which approximates the output of the LSTM.

While feature importance scores are helpful in understanding a model's predictions, they ignore the complex interactions between variables typically learned by neural networks. To this end, the third part introduces contextual decomposition (CD), an interpretation algorithm for analysing individual predictions made by standard LSTMs, without any changes to the underlying model. By decomposing the output of a LSTM, CD captures the contributions of combinations of words or variables to the final prediction of an LSTM. On the task of sentiment analysis with the Yelp and Stanford Sentiment Treebank (SST) data sets, we show that CD is able to reliably identify words and phrases of contrasting sentiment, and how they are combined to yield the LSTM's final prediction. Using the phrase-level labels in SST, we also demonstrate that CD is able to successfully extract positive and negative negations from an LSTM, something which has not previously been done.

When considering interactions between variables, the number of interactions quickly becomes too large for manual inspection, leading to the question of how to automatically select and display an informative subset. In the fourth part, we introduce the use of hierarchical interpretations to explain DNN predictions through our proposed method: agglomerative contextual decomposition (ACD). Given a prediction from a trained DNN, ACD produces a hierarchical clustering of the input features, along with the contribution of each cluster to the final prediction. This hierarchy is optimized to identify clusters of features that the DNN learned are predictive. We introduce ACD using examples from Stanford Sentiment Treebank and ImageNet, in order to diagnose incorrect predictions, identify dataset bias, and extract polarizing phrases of varying lengths. Through human experiments, we demonstrate that ACD enables users both to identify the more accurate of two DNNs and to better trust a DNN's outputs. We also find that ACD's hierarchy is largely robust to adversarial perturbations, implying that it captures fundamental aspects of the input and ignores spurious noise.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I owe a great debt to my Ph.D advisor, Bin Yu. Over the past four years, she has given me the freedom to pursue my own ideas, while teaching me what it means to do impactful research, and how to grow as a person. Bin's counsel helped to impose a more rigorous, and disciplined mindset on me, which I have no doubt will serve me well in the future. Bin also deserves special mention for the considerable improvement to my communication skills that occurred under her guidance. Finally, she has consistently gone above and beyond in caring for me as a person, and a friend, when life's challenges arise.

One of the best parts about working with Bin is her group of students, who I am grateful to consider friends, colleagues, and collaborators. I'd particularly like to mention Chandan Singh, who made invaluable contributions to parts one and four of this thesis, and has served as an excellent spark-plug in keeping things moving. I also worked with many other excellent collaborators (within Bin's group and otherwise), including Eli Ben-Michael, Avi Feller, Nick Altieri, Briton Park, Anobel Odisho, Laura Rieger, Karl Kumbier and Reza Abbasi-Asl, from whom I learned a great deal. I have also benefited from discussions with Rebecca Barter, Soren Kunzel, Yuansi Chen, Yu Wang, Raaz Dwivedi, Bryan Liu, Merle Behr, Yanyan Lan, Simon Walter, Raaz Dwivedi, Xiao Li, Wooseok Ha, Sujayam Saha, and Christine Kuang, and as well as from mentoring undergraduate students Divyansh Agarwal, Summer Devlin, Akilesh Bapu, and Yiqi Hou.

Beyond Bin's group, I also owe Joan Bruna thanks for introducing me to deep learning, patiently working with me in my early research days, and helping me spend a summer at Facebook by introducing me to Arthur Szlam. Jon McAuliffe also deserves thanks for his applied statistics class, which is the best discussion class I have ever taken, and was foundational in my development.

I had the pleasure of spending two thoroughly enjoyable summers conducting research in industry, with Arthur Szlam at Facebook AI Research and Peter Liu at Google Brain. I did some of my best work during this time, met many friends, and received the financial support needed to survive my degree. In both instances, I was originally tasked with a project on which I failed to deliver. Fortunately, I was supported by Arthur and Peter in pursuing my own research direction, ultimately yielding exciting results that feature in parts two and three of this thesis.

I am leaving Berkeley a very different person from when I started, and I have an excellent group of friends to thank for that. More than at any prior point in my life, I count myself lucky to have such an exceptional and caring group of people around me. This group has both helped me grow, and supported me through the highs and lows of graduate school, and my early twenties. There are too many people to name, but I'd like to mention (in no particular order) Kellie Ottoboni for being there to talk about anything, Eli Ben-Michael for our spirited discussions (over beer, or otherwise), Arjun Sondhi for sharing his fresh, unique take on the world, Sara Stoudt for making me think deeply about empathy, Nate Tucker for getting me out of Berkeley when needed, and the lounge foosball table for serving as a stress outlet, and a centrepiece of department culture. More generally, the community of Ph.D students in the Berkeley statistics department is filled with people I am glad to have known, and will be sad to leave. Most importantly, this thesis is dedicated to my family: Gisele, Neil, Caroline and David Murdoch who have supported me from the beginning.

There are many other people who helped make this thesis possible. I would like to thank the members of my qualifying exam committee: Bin Yu, David Bamman, Fernando Perez and Sam Pimentel, with extra thanks to David, Fernando and Bin for also serving on my dissertation committee. I would also like to mention two individuals who were instrumental in my path to Berkeley. Stephen Pulford's high school math classes were what inspired me to study math in the first place, rather than finance. Similarly, my undergraduate research experience with Mu Zhu was my main reason for attending graduate school, rather than going to industry. I am very grateful for the path they set me on. I was also fortunate to receive external funding for my degree from Adobe, through a data science research award, and the Natural Sciences and Engineering Research Council, through a three year postgraduate scholarships-doctoral fellowship. This funding gave me the freedom to pursue my own research without having to worry about teaching, or other constraints.

Chapter 1

Introduction

In recent years, modern machine learning (ML) models have demonstrated impressive predictive accuracy across a wide variety of tasks. However, due to our inability to describe the relationships learned by these models, this improvement in accuracy has come at the cost of many models being characterized as black boxes, incapable of providing additional insight into the data beyond the provided predictions. Recently, research in interpretable ML has received increasing attention as an approach to remedy this problem. While there have been some promising results, many foundational issues are still unresolved. This thesis addresses two challenges in developing interpretable ML methods: what, precisely, does it mean to interpret a ML model, and how can we produce interpretations of individual predictions made by deep learning models.

One of the main challenges in interpretable ML is that the notion of interpretation is not well defined. When taken to its full generality, to interpret something means to extract information (of some form) from it, an overly general notion. Even when restricted to interpretable ML, many basic questions remain, including what constitutes a good interpretation, what common threads exist among disparate interpretation techniques, and how to select an interpretation method for a chosen problem/audience. The first part (Chapter 2 - 7) of this thesis aims to address these challenges by precisely defining interpretable ML, introducing a framework for discussing interpretations, and categorizing existing techniques into a succinct set of categories.

Having better understood what it means to interpret a ML model, the remainder of this thesis focuses on the technical challenge of how to produce such interpretations in the context of deep learning. In particular, we introduce a succession of three prediction-level interpretation methods, which aim to explain individual predictions made by a neural network. Modern neural networks can contain hundreds of millions of parameters, which store information used to make predictions in complex and non-obvious ways. Given a prediction made by a trained model, the methods introduced in parts two through four of this thesis process the models parameters in order to interpret why the model made that particular prediction.

The three prediction-level interpretation methods introduced in this thesis are designed to extract an increasing amount of information from a neural network. First, in part two (Chapter 8 - 10), we introduce a method to extract word importance scores from Long Short Term Memory Networks (LSTMs), a particular type of neural network popular in natural language processing. In

part three (Chapter 11 - 12), we extend this method to develop Contextual Decomposition (CD), a method capable of extracting importance scores for groups of variables, thus capturing interactions an LSTM learns between variables. Finally, in part four (Chapter 13 - 14), we generalize CD from LSTMs to general neural networks, and introduce agglomerative CD (ACD). ACD is an interpretation technique that produces a hierarchy of important groups of features, paired with their CD scores. To produce the hierarchy, ACD conducts agglomerative clustering using the CD score as a joining metric. A recurring challenge throughout this thesis is validating the interpretations produced by our method, with each new method necessitating the creation of a novel evaluation scheme.

The described methods and analysis were all implemented in either torch (parts two and three) or pytorch (part four). For part four, the described results are fully reproducible using publicly available code[1]. Parts two and three were implemented in torch, a framework that was once popular, but is no longer widely used. To enable future development, such as [44], the code for computing CD scores described in part three was translated to pytorch and made publicly available[2].

The following sections describe the contributions of this thesis in more detail.

1.1 Defining interpretable machine learning

The increased focus on interpretable ML has led to considerable confusion about the notion of interpretability. In particular, it is unclear how the wide array of proposed interpretation methods are related, and what common concepts can be used to evaluate them. In collaboration with Chandan Singh, Karl Kumbier, Reza Abbasi-Asl and Bin Yu, we aim to address these concerns by defining interpretability in the context of ML and introducing the Predictive, Descriptive, Relevant (PDR) framework for discussing interpretations. The PDR framework provides three overarching desiderata for evaluation: predictive accuracy, descriptive accuracy and relevancy, with relevancy judged relative to a human audience. Moreover, to help manage the deluge of interpretation methods, we introduce a categorization of existing techniques into model-based and post-hoc categories, with sub-groups including sparsity, modularity and simulatability. To demonstrate how practitioners can use the PDR framework to evaluate and understand interpretations, we provide numerous real-world examples. These examples highlight the often under-appreciated role played by human audiences in discussions of interpretability. Finally, based on our framework, we discuss limitations of existing methods and directions for future work. We hope that this work will provide a common vocabulary that will make it easier for both practitioners and researchers to discuss and choose from the full range of interpretation methods.

[1] https://github.com/csinva/acd

[2] https://github.com/jamie-murdoch/ContextualDecomposition

1.2 Extracting word importance scores from LSTMs

In part two (Chapter 8 - 10), jointly with Arthur Szlam, we focus on LSTMs trained on question-answering and sentiment analysis, two popular tasks in natural language processing. In this context, we extract a simple form of interpretation: for a particular prediction, we compute importance scores for each individual input. These interpretations are computed through an exact decomposition of an LSTM's output, with each term corresponding to the importance of a particular word. In the context of sentiment analysis, this is able to recover strong unigram signals, such as that "good" has positive sentiment and "terrible" has negative sentiment. In order to verify the output of our method, we use the introduced scores to search for consistently important patterns of words learned by state of the art LSTMs on sentiment analysis and question answering. This representation is then quantitatively validated by using the extracted phrases to construct a simple, rule-based classifier which approximates the output of the LSTM.

1.3 Beyond word importance: contextual decomposition to extract interactions from LSTMs

While importance scores for individual words are useful in interpreting decisions, they fail to describe the interactions between variables that an LSTM learns. Interactions are important to capture because they are what distinguish an LSTM from simpler, linear models, like logistic regression. To resolve this, in part three (Chapter 11 - 12) we extend the decomposition from part two to develop Contextual Decomposition (CD), an interpretation method for explaining predictions made by LSTMs. Whereas previous methods focused on individual words, CD computes importance scores for groups of words, or phrases, provided by the user. Computing importance scores for phrases, instead of words, allows CD to capture compositional effects, such as negation (e.g. "not good"), which previous methods are unable to capture. In the presence of interactions between phrases, we used the phrase-level labels in Stanford Sentiment Treebank [118] (SST) to demonstrate that prior, word importance, methods, produce undesirable behavior, while CD is able to correctly recover the underlying dynamics. Moreover, we show that CD is still able to produce word-level importance scores that are competitive with state of the art word importance methods, such as integrated gradients [122] and occlusion [71]. This work is a collaboration with Peter Liu and Bin Yu.

1.4 Hierarchical interpretations of neural network predictions

While CD is able to evaluate the importance of a given phrase, it requires humans to specify which phrases to investigate. Due to the large number of phrases in a typical sentence, it is unclear which phrases a user should look at, and how to display those phrase importances in an intuitive summary of a model's prediction. To deal with this challenge, in part four (Chapter 13 - 14) we introduce

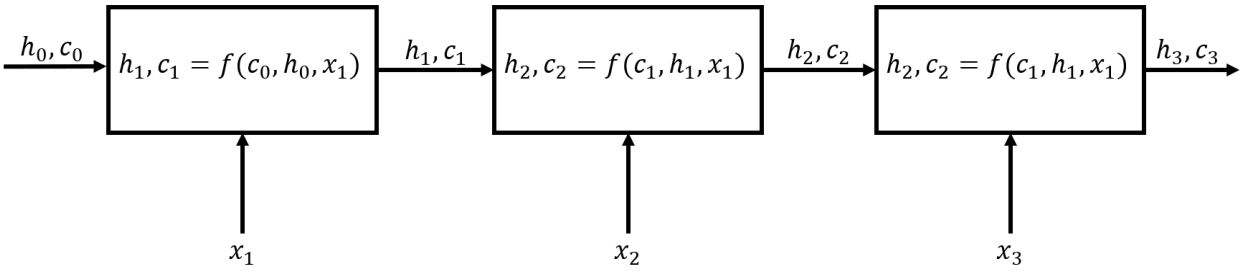


Figure 1.1: Overview of how Recurrent Neural Networks process data

agglomerative contextual decomposition (ACD). At a high level, ACD uses agglomerative clustering as a heuristic search for phrases that contain large interactions, and displays the discovered phrases in a hierarchy for easy visualization.

In particular, given a prediction from a trained DNN, ACD produces a hierarchical clustering of the input features, along with the contribution of each cluster to the final prediction. This hierarchy is optimized to identify clusters of features that the DNN learned are predictive. CD is used in ACD as the joining metric in the agglomeration routine. This work also generalizes CD from LSTMs to general neural networks. Through human experiments, we demonstrate that ACD enables users both to identify the more accurate of two DNNs and to better trust a DNN's outputs. We also find that ACD's hierarchy is largely robust to adversarial perturbations, implying that it captures fundamental aspects of the input and ignores spurious noise. This project was a collaboration with Chandan Singh and Bin Yu.

1.5 A brief overview of Long Short Term Memory Networks

Due to their ability to handle variable-length inputs, such as sentences, in recent years recurrent neural networks (RNNs) have become a core component of natural language processing systems. At a high level, these models work by sequentially processing inputs $x_1, ..., x_T \in \mathbb{R}^{d_1}$ into a sequence of cell and state vectors $(c_1, h_1), ..., (c_T, h_T)$ where $c_t, h_t \in \mathbb{R}^{d_2}$. Starting from the first input x_1, and initializing $h_0 = c_0 = 0$, an RNN iteratively computes $c_t, h_t = f(c_{t-1}, h_{t-1}, x_t)$, so that each c_t, h_t captures information for the sequence up to and including x_t, as displayed in Figure 1.1.

After processing the full sequence, the final state h_T is treated as a vector of learned features, and used as input to a multinomial logistic regression, often called SoftMax, to return a probability distribution p over C classes, shown below. As the update function $f(c_{t-1}, h_{t-1}, x_t)$ is required to be differentiable, the final probabilities are the composition of differentiable functions, and thus differentiable themselves. Consequently, to fit an RNN, practitioners use variants of stochastic gradient descent, such as Adam [63], where gradients are computed using backpropagation through time [132, 84].

$$p_j = \text{SoftMax}(Wh_T)_j = \frac{\exp(W_j h_T)}{\sum_{k=1}^{C} \exp(W_k h_t)} \tag{1.1}$$

There have been multiple different update functions $f(c_{t-1}, h_{t-1}, x_t)$ proposed over time, including the original Elman RNN [38], and more recently the Gated Recurrent Unit (GRU) [31]. In this thesis, we focus on Long Short Term Memory networks (LSTMs) [53], a type of RNN that empirically has been proven to be very successful (although our methods readily generalize to other update functions [98]).

As the LSTM update function f is discussed in multiple parts of this thesis, we now formally introduce it. In this instance, it is simpler to express the update function using six equations, rather than a single function f, but the underlying concepts remain the same. Intuition for these equations typically centers around the cell update equation, Equation (1.6). In this equation, the previous cell state c_{t-1} has some values forgotten, where forgetting corresponds to element-wise multiplication by the forget gate f_t, which is confined to the range $(0, 1)$. After multiplication by the forget gate, an update for step t, $i_t \odot g_t$, is added to allow for information learned at step t. The update equations for an LSTM are shown below.

$$o_t = \sigma(W_o x_t + V_o h_{t-1} + b_o) \tag{1.2}$$
$$f_t = \sigma(W_f x_t + V_f h_{t-1} + b_f) \tag{1.3}$$
$$i_t = \sigma(W_i x_t + V_i h_{t-1} + b_i) \tag{1.4}$$
$$g_t = \tanh(W_g x_t + V_g h_{t-1} + b_g) \tag{1.5}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{1.6}$$
$$h_t = o_t \odot \tanh(c_t) \tag{1.7}$$

Where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$ is the hyperbolic tangent function, $W_o, W_i, W_f, W_g \in \mathbb{R}^{d_1 \times d_2}$, $V_o, V_f, V_i, V_g \in \mathbb{R}^{d_2 \times d_2}$, $b_o, b_g, b_i, b_g \in \mathbb{R}^{d_2}$ and $\odot$ denotes element-wise multiplication. o_t, f_t and i_t are often referred to as output, forget and input gates, respectively, due to the fact that their values are bounded between 0 and 1, and that they are used in element-wise multiplication.

Part I

Defining interpretable machine learning

Chapter 2

Interpretable machine learning: an introduction

Machine learning (ML) has recently received considerable attention for its ability to accurately predict a wide variety of complex phenomena. However, there is a growing realization that, in addition to predictions, ML models are capable of producing knowledge about domain relationships contained in data, often referred to as interpretations. These interpretations have found uses both in their own right, e.g. medicine [75], policy-making [25], and science [9, 129], as well as in auditing the predictions themselves in response to issues such as regulatory pressure [46] and fairness [37].

In the absence of a well-formed definition of interpretability, a broad range of methods with a correspondingly broad range of outputs (e.g. visualizations, natural language, mathematical equations) have been labeled as interpretation. This has led to considerable confusion about the notion of interpretability. In particular, it is unclear what it means to interpret something, what common threads exist among disparate methods, and how to select an interpretation method for a particular problem/audience.

In this part, we attempt to address these concerns. To do so, we first define interpretability in the context of machine learning and place it within a generic data science life cycle. This allows us to distinguish between two main classes of interpretation methods: model-based[1] and post hoc. We then introduce the Predictive, Descriptive, Relevant (PDR) framework, consisting of three desiderata for evaluating and constructing interpretations: predictive accuracy, descriptive accuracy, and relevancy, where relevancy is judged by a human audience. Using these terms, we categorize a broad range of existing methods, all grounded in real-world examples[2]. In doing so, we provide a common vocabulary for researchers and practitioners to use in evaluating and selecting interpretation methods. We then show how our work enables a clearer discussion of open problems for future research.

[1]For clarity, throughout this part we use the term *model* to refer to both machine-learning models and algorithms.

[2]Examples were selected through a non-exhaustive search of related work.

2.1 Defining interpretable machine learning

On its own, interpretability is a broad, poorly defined concept. Taken to its full generality, to interpret data means to extract information (of some form) from it. The set of methods falling under this umbrella spans everything from designing an initial experiment to visualizing final results. In this overly general form, interpretability is not substantially different from the established concepts of data science and applied statistics.

Instead of general interpretability, we focus on the use of interpretations in the context of ML as part of the larger data-science life cycle. We define interpretable machine learning as the use of machine-learning models for the extraction of *relevant* knowledge about domain relationships contained in data. Here, we view knowledge as being *relevant* if it provides insight for a particular audience into a chosen domain problem. These insights are often used to guide communication, actions, and discovery. Interpretation methods use ML models to produce relevant knowledge about domain relationships contained in data. This knowledge can be produced in formats such as visualizations, natural language or mathematical equations, depending on the context and audience. For instance, a doctor who must diagnose a single patient will want qualitatively different information than an engineer determining if an image classifier is discriminating by race.

2.2 Background

Interpretability is a quickly growing field in machine learning, and there have been multiple works examining various aspects of interpretations (sometimes under the heading *explainable AI*). One line of work focuses on providing an overview of different interpretation methods with a strong emphasis on post hoc interpretations of deep learning models [30, 47], sometimes pointing out similarities between various methods [78, 7]. Other work has focused on the narrower problem of how interpretations should be evaluated [36, 43] and what properties they should satisfy [74]. These previous works touch on different subsets of interpretability, but do not address interpretable machine learning as a whole, and give limited guidance on how interpretability can actually be used in data-science life cycles. We aim to do so by providing a framework and vocabulary to fully capture interpretable machine learning, its benefits, and its applications to concrete data problems.

Interpretability also plays a role in other research areas. For example, interpretability is a major topic when considering bias and fairness in ML models [49, 20], with examples given throughout the part [34]. In psychology, the general notions of interpretability and explanations have been studied at a more abstract level [60, 76], providing relevant conceptual perspectives. Additionally, we comment on two related areas that are distinct but closely related to interpretability: causal inference and stability.

Causal inference Causal inference [56] is a subject from statistics which is related, but distinct, from interpretable machine learning. Causal inference methods focus solely on extracting causal relationships from data, i.e. statements that altering one variable will cause a change in another.

In contrast, interpretable ML, and most other statistical techniques, are generally used to describe non-causal relationships, or relationships in observational studies.

In some instances, researchers use both interpretable machine learning and causal inference in a single analysis [16]. One form of this is where the non-causal relationships extracted by interpretable ML are used to suggest potential causal relationships. These relationships can then be further analyzed using causal inference methods, and fully validated through experimental studies.

Stability Stability, as a generalization of robustness in statistics, is a concept that applies throughout the entire data-science life cycle, including interpretable ML). The stability principle requires that each step in the life cycle is stable with respect to appropriate perturbations, such as small changes in the model or data. Recently, stability has been shown to be important in applied statistical problems, for example when trying to make conclusions about a scientific problem [138] and in more general settings [48]. Stability can be helpful in evaluating interpretation methods and is a prerequisite for trustworthy interpretations. That is, one should not interpret parts of a model which are not stable to appropriate perturbations to the model and data. This is demonstrated through examples in the text [97, 2, 16].

Chapter 3

Interpretation in the data science life cycle

Before discussing interpretation methods, we first place the process of interpretable ML within the broader data-science life cycle. Figure 3.1 presents a deliberately general description of this process, intended to capture most data-science problems. What is generally referred to as interpretation largely occurs in the modeling and post hoc analysis stages, with the problem, data and audience providing the context required to choose appropriate methods.

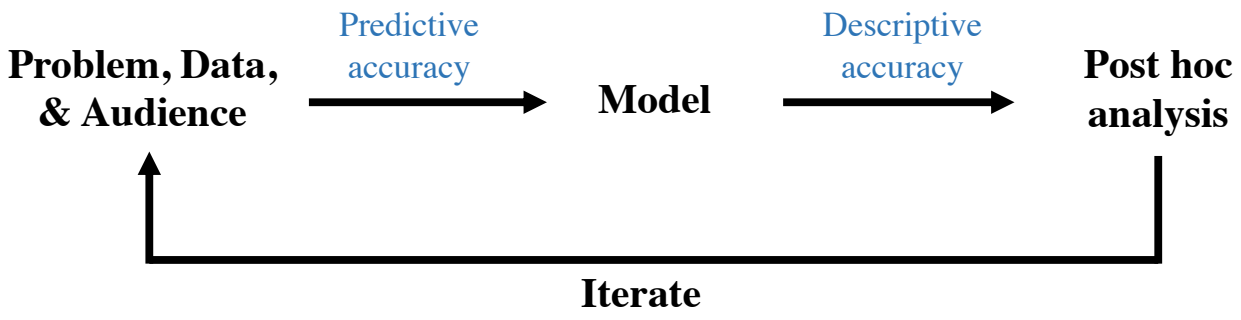


Figure 3.1: Overview of different stages (black text) in a data-science life cycle where interpretability is important. Main stages are discussed in Chapter 3 and accuracy (blue text) is described in Chapter 4.

Problem, data, and audience At the beginning of the cycle, a data-science practitioner defines a domain problem that they would like to understand using data. This problem can take many forms. In a scientific setting, the practitioner may be interested in relationships contained in the data, such as how brain cells in a particular area of the visual system relate to visual stimuli [104]. In industrial settings, the problem often concerns the predictive performance or other qualities of a model, such as how to assign credit scores with high accuracy [55], or do so fairly with respect to gender and race [34]. The nature of the problem plays a role in interpretability, as the relevant context and audience are essential in determining what methods to use.

After choosing a domain problem, the practitioner collects data to study it. Aspects of the data-collection process can affect the interpretation pipeline. Notably, biases in the data (i.e. mismatches between the collected data and the population of interest) will manifest themselves in the model, restricting one's ability to make interpretations regarding the problem of interest.

Model Based on the chosen problem and collected data, the practitioner then constructs a predictive model. At this stage, the practitioner processes, cleans, and visualizes data, extracts features, selects a model (or several models) and fits it. Interpretability considerations often come into play in this step related to the choice between simpler, easier to interpret models and more complex, black-box models, which may fit the data better. The model's ability to fit the data is measured through predictive accuracy.

Post hoc analysis Having fit a model (or models), the practitioner then analyzes it for answers to the original question. The process of analyzing the model often involves using interpretability methods to extract various (stable) forms of information from the model. The extracted information can then be analyzed and displayed using standard data analysis methods, such as scatter plots and histograms. The ability of the interpretations to properly describe what the model has learned is denoted by descriptive accuracy.

Iterate If sufficient answers are uncovered after the post hoc analysis stage, the practitioner finishes. Otherwise, they update something in the chain (problem, data, and/or model) and iterate, potentially multiple times [19]. Note that they can terminate the loop at any stage, depending on the context of the problem.

3.1 Interpretation methods within the PDR framework

In the framework described above, most interpretation methods fall either in the modeling or post hoc analysis stages. We call interpretability in the modeling stage *model-based interpretability* (Chapter 5). This part of interpretability is focused upon constraining the form of ML models so that they readily provide useful information about the uncovered relationships. As a result of these constraints, the space of potential models is smaller, which can result in lower predictive accuracy. Consequently, model-based interpretability is best used when the underlying relationship is relatively simple.

We call interpretability in the post hoc analysis stage *post hoc interpretability* (Chapter 6). These interpretation methods take a trained model as input, and extract information about what relationships the model has learned. They are most helpful when the underlying relationship is especially complex, and practitioners need to train an intricate, black-box model in order to achieve a reasonable predictive accuracy.

After discussing desiderata for interpretation methods, we investigate these two forms of interpretations in detail and discuss associated methods.

Chapter 4

The PDR desiderata for interpretations

In general, it is unclear how to select and evaluate interpretation methods for a particular problem and audience. To help guide this process, we introduce the PDR framework, consisting of three desiderata that should be used to select interpretation methods for a particular problem: predictive accuracy, descriptive accuracy, and relevancy.

4.1 Accuracy

The information produced by an interpretation method should be faithful to the underlying process the practitioner is trying to understand. In the context of ML, there are two areas where errors can arise: when approximating the underlying data relationships with a model (predictive accuracy) and when approximating what the model has learned using an interpretation method (descriptive accuracy). For an interpretation to be trustworthy, one should try to maximize both of the accuracies. In cases where the accuracy is not very high, the resulting interpretations may still be useful. However, it is especially important to check their trustworthiness through external validation, such as running an additional experiment.

Predictive accuracy

The first source of error occurs during the model stage, when an ML model is constructed. If the model learns a poor approximation of the underlying relationships in the data, any information extracted from the model is unlikely to be accurate. Evaluating the quality of a model's fit has been well-studied in standard supervised ML frameworks, through measures such as test-set accuracy. In the context of interpretation, we describe this error as predictive accuracy.

Note that in problems involving interpretability, one often requires a notion of predictive accuracy that goes beyond just average accuracy. The distribution of predictions matters. For instance, it could be problematic if the prediction error is much higher for a particular class. Moreover, the predictive accuracy should be stable with respect to reasonable data and model perturbations.

For instance, one should not trust interpretations from a model which changes dramatically when trained on a slightly smaller subset of the data.

Descriptive accuracy

The second source of error occurs during the post hoc analysis stage, when interpretation methods are used to analyze a fitted model. Oftentimes, interpretation methods provide an imperfect representation of the relationships learned by a model. This is especially challenging for complex black-box models such as deep neural networks, which store nonlinear relationships between variables in non-obvious forms.

▷ **Definition** We define *descriptive accuracy*, in the context of interpretation, as the degree to which an interpretation method objectively captures the relationships learned by machine learning models.

A common conflict: predictive vs descriptive accuracy

In selecting what model to use, practitioners are often faced with a trade-off between predictive and descriptive accuracy. On the one hand, the simplicity of model-based interpretation methods yields consistently high descriptive accuracy, but can sometimes result in lower predictive accuracy on complex datasets. On the other hand, in complex settings such as image analysis, complicated models generally provide high predictive accuracy, but are harder to analyze, resulting in a lower descriptive accuracy.

4.2 Relevancy

When selecting an interpretation method, it is not enough for the method to have high accuracy - the extracted information must also be relevant. For example, in the context of genomics, a patient, doctor, biologist, and statistician may each want different (yet consistent) interpretations from the same model. The context provided by the problem and data stages in Figure 3.1 guides what kinds of relationships a practitioner is interested in learning about, and by extension the methods that should be used.

▷ **Definition** We define an interpretation to be *relevant* if it provides insight for a particular audience into a chosen domain problem.

Relevancy often plays a key role in determining the trade-off between predictive and descriptive accuracy. Depending on the context of the problem at hand, a practitioner may choose to focus on one over the other. For instance, when interpretability is used to audit a model's predictions, such as to enforce fairness, descriptive accuracy can be more important. In contrast, interpretability can also be used solely as a tool to increase the predictive accuracy of a model, for instance, through improved feature engineering.

Having outlined the main desiderata for interpretation methods, we now discuss how they link to interpretation in the modeling and post hoc analysis stages in the data-science life cycle. Fig-

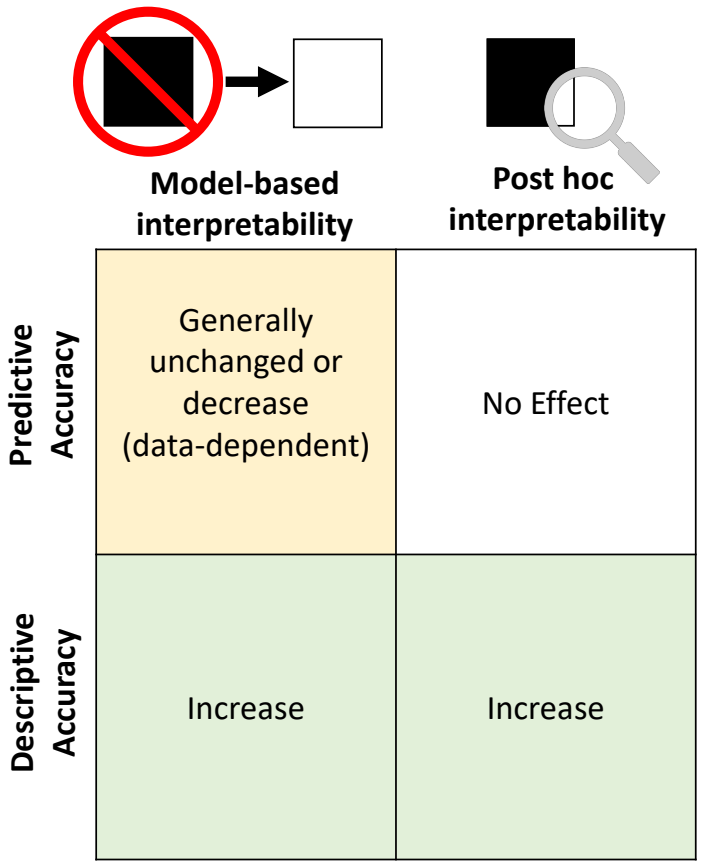


Figure 4.1: Impact of interpretability methods on descriptive and predictive accuracies. Model-based interpretability (Chapter 5) involves using a simpler model to fit the data which can negatively affect predictive accuracy, but yields higher descriptive accuracy. Post hoc interpretability (Chapter 6) involves using methods to extract information from a trained model (with no effect on predictive accuracy). These correspond to the model and post hoc stages in Figure 3.1.

ure 4.1 draws parallels between our desiderata for interpretation techniques introduced in Chapter 4 and our categorization of methods in Chapter 5 and Chapter 6. In particular, both post hoc and model-based methods aim to increase descriptive accuracy, but only model-based affects the predictive accuracy. Not shown is relevancy, which determines what type of output is helpful for a particular problem and audience.

Chapter 5

Model-based interpretability

We now discuss how interpretability considerations come into play in the modeling stage of the data science life cycle (see Figure 3.1). At this stage, the practitioner constructs an ML model from the collected data. We define model-based interpretability as the construction of models that readily provide insight into the relationships they have learned. Different model-based interpretability methods provide different ways of increasing descriptive accuracy by constructing models which are easier to understand, sometimes resulting in lower predictive accuracy. The main challenge of model-based interpretability is to come up with models that are simple enough to be easily understood by the audience, yet sophisticated enough to properly fit the underlying data.

In selecting a model to solve a domain problem, the practitioner must consider the entirety of the PDR framework. The first desideratum to consider is predictive accuracy. If the constructed model does not accurately represent the underlying problem, any subsequent analysis will be suspect [22, 40]. Second, the main purpose of model-based interpretation methods is to increase descriptive accuracy. Finally, the relevancy of a model's output must be considered, and is determined by the context of the problem, data, and audience. We now discuss some widely useful types of model-based interpretability methods.

5.1 Sparsity

When the practitioner believes that the underlying relationship in question is based upon a sparse set of signals, they can impose sparsity on their model by limiting the number of non-zero parameters. In this section, we focus on linear models, but sparsity can be helpful more generally. When the number of non-zero parameters is sufficiently small, a practitioner can interpret the variables corresponding to those parameters as being meaningfully related to the outcome in question, and can also interpret the magnitude and direction of the parameters. However, before one can interpret a sparse parameter set, one should check for stability of the parameters. For example, if the set of sparse parameters changes due to small perturbations in the data set, the coefficients should not be interpreted [73].

When the practitioner is able to correctly incorporate sparsity into their model, it can improve

all three interpretation desiderata. By reducing the number of parameters to analyze, sparse models can be easier to understand, yielding higher descriptive accuracy. Moreover, incorporating prior information in the form sparsity into a sparse problem can help a model achieve higher predictive accuracy and yield more relevant insights. Note that incorporating sparsity can often be quite difficult, as it requires understanding the data-specific structure of the sparsity and how it can be modelled.

Methods for obtaining sparsity often utilize a penalty on a loss function, such as LASSO [126] and sparse coding [91], or on a model selection criteria such as AIC or BIC [3, 26]. Many search-based methods have been developed to find sparse solutions. These methods search through the space of non-zero coefficients using classical subset-selection methods (e.g. orthogonal matching pursuit [94]). Model sparsity is often useful for high-dimensional problems, where the goal is to identify key features for further analysis. As a result, sparsity penalties have been incorporated into complex models such as random forests to identify a sparse subset of important features [6].

In the following example from genomics, sparsity is used to increase the relevancy of the produced interpretations by reducing the number of potential interactions to a manageable level.

▷ **Ex.** Identifying interactions among regulatory factors or biomolecules is an important question in genomics. Typical genomic datasets include thousands or even millions of features, many of which are active in specific cellular or developmental contexts. The massive scale of such datasets make interpretation a considerable challenge. Sparsity penalties are frequently used to make the data manageable for statisticians and their collaborating biologists to discuss and identify promising candidates for further experiments.

For instance, one recent study [97] uses a biclustering approach based on sparse canonical correlation analysis (SCCA) to identify interactions among genomic expression features in *Drosophila melanogaster* (fruit flies) and *Caenorhabditis elegans* (roundworms). Sparsity penalties enable key interactions among features to be summarized in heatmaps which contain few enough variables for a human to analyze. Moreover, this study performs stability analysis on their model, finding it to be robust to different initializations and perturbations to hyperparameters.

5.2 Simulatability

A model is said to be simulatable if a human (for whom the interpretation is intended) is able to internally simulate and reason about its entire decision-making process (i.e. how a trained model produces an output for an arbitrary input). This is a very strong constraint to place on a model, and can generally only be done when the number of features is low, and the underlying relationship is simple. Decision trees [23] are often cited as a simulatable model, due to their hierarchical decision-making process. Another example is lists of rules [41, 70], which can easily be simulated. Due to their simplicity, simulatable models have very high descriptive accuracy. When they can also provide reasonable predictive accuracy, they can be very effective. In the following example, a novel simulatable model is able to produce high predictive accuracy, while maintaining the high levels of descriptive accuracy and relevancy normally attained by rules-based models.

▷ **Ex.** In medical practice, when a patient has been diagnosed with atrial fibrillation, caregivers often want to predict the risk that the particular patient will have a stroke in the next year. Moreover, given the potential ramifications of medical decisions, it is important that these predictions are not only accurate, but interpretable to both the caregivers and patients.

To make the prediction, [70] uses data from 12,586 patients detailing their age, gender, history of drugs and conditions preceding their diagnosis, and whether they had a stroke within a year of diagnosis. In order to construct a model that has high predictive and descriptive accuracy, [70] introduce a method for learning lists of if-then rules that are predictive of one year stroke risk. The resulting classifier, displayed in Figure 5.1, requires only seven if-then statements to achieve competitive accuracy, and is easy for even non-technical practitioners to quickly understand.

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 5.1: Rule list for classifying stroke risk from patient data (replicated Fig 3 from [70], with permission from the authors). One can easily simulate and understand the relationships between different variables such as age on *stroke risk*. These rules come in to effect following diagnosis of atrial fibrillation.

5.3 Modularity

We define an ML model to be modular if a meaningful portion(s) of its prediction-making process can be interpreted independently. While modular models are not as easy to understand as sparse or simulatable models, they can still be useful in increasing descriptive accuracy to provide insights into the relationships the model has learned.

A wide array of models satisfy modularity to different degrees. Generalized additive models [51] force the relationship between variables in the model to be additive. In deep learning, specific methods such as attention [61] and modular network architectures [8] provide limited insight into a network's inner workings. Probabilistic models can enforce modularity by specifying a conditional independence structure which makes it easier to reason about different parts of a model independently [66].

The following example uses modularity to produce relevant interpretations for use in diagnosing biases in training data.

▷ **Ex.** When prioritizing patient care for pneumonia patients in a hospital, one possible method is to predict the likelihood of death within 60 days, and focus on the patients with a higher mortality risk. Given the potential life and death consequences, being able to explain the reasons for hospitalizing a patient or not is very important.

A recent study [29] uses a dataset of 14,199 pneumonia patients, with 46 features including from demographics (e.g. age and gender), simple physical measurements (e.g. heart rate, blood pressure) and lab tests (e.g. white blood cell count, blood urea nitrogen). To predict mortality risk, they use a generalized additive model with pairwise interactions, displayed below. The univariate and pairwise terms ($f_j(x_j)$ and $f_{ij}(x_i, x_j)$) can be individually interpreted in the form of curves and heatmaps respectively.

$$g(\mathbb{E}[y]) = \beta_0 + \sum_j f_j(x_j) + \sum_{i \neq j} f_{ij}(x_i, x_j) \tag{5.1}$$

By inspecting the individual modules, the researchers found a number of counterintuitive properties of their model. For instance, the fitted model learned that having asthma is associated with a lower risk of dying from pneumonia. In reality, the opposite is true - patients with asthma are known to have a higher risk of death from pneumonia. Because of this, in the collected data all patients with asthma received aggressive care, which was fortunately effective at reducing their risk of mortality relative to the general population.

In this instance, if the model were used without having been interpreted, pneumonia patients with asthma would have have been de-prioritized for hospitalization. Consequently, the use of ML would increase their likelihood of dying. Fortunately, the use of an interpretable model enabled the researchers to identify and correct errors like this one, better ensuring that the model could be trusted in the real world.

5.4 Domain-based feature engineering

While the type of model is important in producing a useful interpretation, so are the features that are used as inputs to the model. Having more informative features makes the relationship that needs to be learned by the model simpler, allowing one to use other model-based interpretability methods. Moreover, when the features have more meaning to a particular audience, they become easier to interpret.

In many individual domains, expert knowledge can be useful in constructing feature sets that are useful for building predictive models. The particular algorithms used to extract features are generally domain-specific, relying both on the practitioner's existing domain expertise and insights drawn from the data through exploratory data analysis. For example, in natural language processing, documents are embedded into vectors using tf-idf [100] and in computer vision mathematical transformations have been developed to produce useful representations of images [33]. In the example below, domain knowledge about cloud coverage is exploited to design three simple features

that increase the predictive accuracy of a model while maintaining the high descriptive accuracy of a simple predictive model.

▷ **Ex.** When modelling global climate patterns, an important quantity is the amount and location of arctic cloud coverage. Due to the complex, layered nature of climate models, it is beneficial to have simple, easily auditable, cloud coverage models for use by down-stream climate scientists.

In [112], the authors use an unlabeled dataset of arctic satellite imagery to build a model predicting whether each pixel in an image contains clouds or not. Given the qualitative similarity between ice and clouds, this is a challenging prediction problem. By conducting exploratory data analysis and utilizing domain knowledge through interactions with climate scientists, the authors identify three simple features that are sufficient to cluster whether or not images contain clouds. Using these three features as input to quadratic discriminant analysis, they achieve both high predictive accuracy and transparency when compared with expert labels (which were not used in developing the features and the QDA clustering method).

5.5 Model-based feature engineering

There are a variety of automatic approaches for constructing interpretable features. Two examples are unsupervised learning and dimensionality reduction. Unsupervised methods, such as clustering, matrix factorization, and dictionary learning, aim to process unlabelled data and output a description of their structure. These structures often shed insight into relationships contained within the data and can be useful in building predictive models. Dimensionality reduction focuses on finding a representation of the data which is lower-dimensional than the original data. Methods such as principal components analysis [57], independent components analysis [17], and canonical correlation analysis [54] can often identify a few interpretable dimensions, which can then be used as input to a model or to provide insights in their own right. Using fewer inputs can not only improve descriptive accuracy, but can increase predictive accuracy by reducing the number of parameters to fit. In the following example, unsupervised learning (non-negative matrix factorization) is used to represent images in a low-dimensional, genetically meaningful, space.

▷ **Ex.** Heterogeneity is an important consideration in genomic problems and associated data. In many cases, regulatory factors or biomolecules can play a specific role in one context, such as a particular cell type or developmental stage, and have a very different role in other contexts. Thus, it is important to understand the "local" behavior of regulatory factors or biomolecules.

A recent study [135], uses unsupervised learning to learn spatial patterns of gene expression in *Drosophila* (fruit fly) embryos. In particular, they use stability driven nonnegative matrix factorization to decompose images of complex spatial gene expression patterns into a library of 21 "principal patterns", which can be viewed as pre-organ regions This decomposition, which is interpretable to biologists, allows the study of gene-gene interactions in pre-organ regions of the developing embryo.

Chapter 6

Post hoc interpretability

We now discuss how interpretability considerations come into play in the post hoc analysis stage of the data-science life cycle. At this stage, the practitioner analyzes a trained model in order to provide insights into the learned relationships. This is particularly challenging when the model's parameters do not clearly show what relationships the model has learned. To aid in this process, a variety of post hoc interpretability methods have been developed to provide insight into what a trained model has learned, without changing the underlying model. These methods are particularly important for settings where the collected data is high-dimensional and complex, such as with image data. Once the information has been extracted from the fitted model, it can be analyzed using standard, exploratory data analysis techniques, such as scatter plots and histograms.

When conducting post hoc analysis, the model has already been trained, so its predictive accuracy is fixed. Thus, under the PDR framework, a researcher must only consider descriptive accuracy and relevancy (relative to a particular audience). Improving on each of these criteria are areas of active research.

Most widely useful post hoc interpretation methods fall into two main categories: prediction-level and dataset-level interpretations, which are sometimes referred to as local and global interpretations, respectively. Prediction-level interpretation methods focus on explaining individual predictions made by models, such as what features and/or interactions led to the particular prediction. Dataset-level approaches focus on the global relationships the model has learned, such as what visual patterns are associated with a predicted response. These two categories have much in common (in fact, dataset-level approaches often yield information at the prediction-level), but we discuss them separately, as methods at different levels are meaningfully different.

6.1 Dataset-level interpretation

When a practitioner is interested in more general relationships learned by a model, e.g. relationships that are relevant for a particular class of responses or subpopulation, they use dataset-level interpretations.

Interaction and feature importances

Feature importance scores, at the dataset-level, try to capture how much individual features contribute, across a dataset, to a prediction. These scores can provide insights into what features the model has identified as important for which outcomes, and their relative importance. Methods have been developed to score individual features in many models including neural networks [90], random forests, [21, 121], and generic classifiers [5].

In addition to feature importances, methods have been developed to extract important interactions between features. Interactions are important as ML models are often highly nonlinear and learn complex interactions between features. Methods exist to extract interactions from a variety of ML models including random forests [16, 67] and neural networks [127, 1]. In the following example, the descriptive accuracy of random forests is increased by extracting Boolean interactions (a problem-relevant form of interpretation) from a trained model.

▷ **Ex.** High-order interactions among regulatory factors or genes play an important role in defining cell-type specific behavior in biological systems. As a result, extracting such interactions from genomic data is an important problem in biology.

A previous line of work considers the problem of searching for biological interactions associated with important biological processes [16, 67]. To identify candidate biological interactions, the authors train a series of iteratively re-weighted RFs and search for stable combinations of features that frequently co-occur along the predictive RF decision paths. This approach takes a step beyond evaluating the importance of individual features in an RF, providing a more complete description of how features influence predicted responses. By interpreting the interactions used in RFs, the researchers identified gene-gene interactions with 80% accuracy in the *Drosophila* embryo and identify candidate targets for higher-order interactions.

Statistical feature importances

In some instances, in addition to the raw value, we can compute statistical measures of confidence as feature importance scores, a standard technique taught in introductory statistics classes. By making assumptions about the underlying data generating process, models like linear and logistic regression can compute confidence intervals and hypothesis tests for the values, and linear combinations, of their coefficients. These statistics can be helpful in determining the degree to which the observed coefficients are statistically significant. It is important to note that the assumptions of the underlying probabilistic model must be fully verified before using this form of interpretation. Below we present a cautionary example where different assumptions lead to opposing conclusions being drawn from the same dataset.

▷ **Ex.** Here, we consider the lawsuit *Students for Fair Admissions, Inc. v. Harvard* regarding the use of race in undergraduate admissions to Harvard University. Initial reports by Harvard's Office of Institutional Research used logistic regression to model the probability of admission using different features of an applicant's profile, including their race [88]. This analysis found that the coefficient associated with being Asian (and not low income) had a coefficient of -0.418 with a

significant p-value (¡0.001). This negative coefficient suggested that being Asian had a significant negative association with admission probability.

Subsequent analysis from both sides in the lawsuit attempted to analyze the modeling and assumptions to decide on the significance of race in the model's decision. The plaintiff's expert report [10] suggested that race was being unfairly used by building on the original report from Harvard's Office of Institutional Research. It also incorporates analysis on more subjective factors such as "personal ratings" which seem to hurt Asian students' admission. In contrast, the expert report supporting Harvard University [27] finds that by accounting for certain other variables, the effect of race on Asian students acceptance is no longer significant. Significances derived from statistical tests in regression or logistic regression models at best establish association, but not causation. Hence the analyses from both sides are flawed. This example demonstrates the practical and misleading consequences of statistical feature importances when used inappropriately.

Visualizations

When dealing with high-dimensional datasets, it can be challenging to quickly understand the complex relationships that a model has learned, making the presentation of the results particularly important. To help deal with this, researchers have developed a number of different visualizations which help to understand what a model has learned. For linear models with regularization, plots of regression coefficient paths show how varying a regularization parameter affects the fitted coefficients. When visualizing convolutional neural networks trained on image data, work has been done on visualizing filters [140, 89], maximally activating responses of individual neurons or classes [83], understanding intra-class variation [131], and grouping different neurons [141]. For Long Short Term Memory Networks (LSTMs), researchers have focused on analyzing the state vector, identifying individual dimensions that correspond to meaningful features (e.g. position in line, within quotes) [59], and building tools to track the model's decision process over the course of a sequence [120].

In the following example, relevant interpretations are produced by using maximal activation images for identifying patterns that drive the response of brain cells.

▷ **Ex.** A recent study visualizes learned information from deep neural networks to understand individual brain cells [2]. In this study, macaque monkeys were shown images while the responses of brain cells in their visual system (area V4) were recorded. Neural networks were trained to predict the responses of brain cells to the images. These neural networks produce accurate fits, but provide little insight into what patterns in the images increase the brain cells response without further analysis. To remedy this, the authors introduce DeepTune, a method which provides a visualization, accessible to neuroscientists and others, of the patterns which activate a brain cell. The main intuition behind the method is to optimize the input of a network to maximize the response of a neural network model (which represent a brain cell).

The authors go on to analyze the major problem of instability. When post hoc visualizations attempt to answer scientific questions, the visualizations must be stable to reasonable perturbations (e.g. the choice of model); if there are changes in the visualization due to the choice of a model, it is likely not meaningful. The authors address this explicitly by fitting eighteen different models to

the data and using a stable optimization over all the models to produce a final consensus DeepTune visualization.

Analyzing trends and outliers in predictions

When interpreting the performance of an ML model, it can be helpful to look not just at the average accuracy, but also at the distribution of predictions and errors. For example, residual plots can identify heterogeneity in predictions, and suggest particular data points to analyze, such as outliers in the predictions, or examples which had the largest prediction errors. Moreover, these plots be used to analyze trends across the predictions. For instance, in the example below, influence functions are able to efficiently identify mislabelled data points.

▷ **Ex.** This kind of analysis can also be used to identify mislabeled training data. A recently introduced method [65] uses the classical statistical concept of influence functions to identify points in the training data which contribute to predictions made by ML models. By searching for training data points which contribute the most amount to individual predictions, they were able to find mislabelled data points without having to look at too much data. Correcting these mislabeled training points subsequently improved the test accuracy.

6.2 Prediction-level interpretation

Prediction-level approaches are useful when a practitioner is interested in understanding how individual predictions are made by a model. Note that prediction-level approaches can sometimes be aggregated to yield dataset-level insights.

Feature importance scores

The most popular approach to prediction-level interpretation has involved assigning importance scores to individual features. Intuitively, a variable with a large positive (negative) score made a highly positive (negative) contribution to a particular prediction. In the deep learning literature, a number of different approaches have been proposed to address this problem [119, 122, 111, 12, 117, 114, 86, 32, 103, 143], with some methods for other models as well [77]. These are often displayed in the form of a heat map highlighting important features. Note that feature importance scores at the prediction-level can offer much more information than feature importance scores at the dataset-level. This is a result of heterogeneity in a nonlinear model: the importance of a feature can vary for different examples as a result of interactions with other features. In the following example, feature importance scores are used to increase the descriptive accuracy of black-box models in order to validate their fairness.

▷ **Ex.** When using ML models to predict sensitive outcomes, such as whether a person should receive a loan or a criminal sentence, it is important to verify that the algorithm is not discriminating against people based on protected attributes, such as race or gender. This problem is often described as ensuring ML models are “fair”. In [34], the authors introduce a variable importance

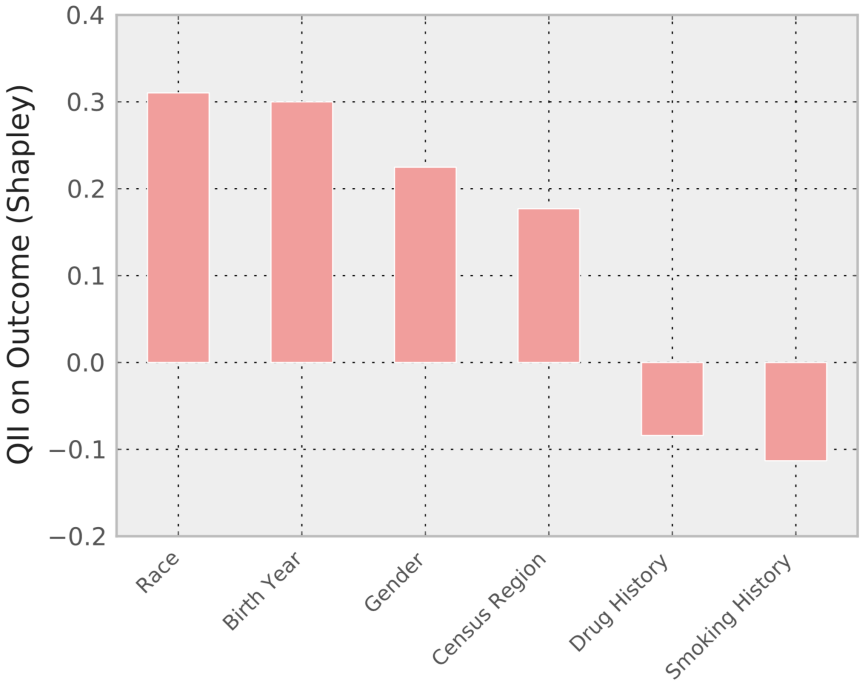


Figure 6.1: Importance of different predictors in predicting the likelihood of arrest for a particular person. Reprinted with permission from the authors.

measure designed to isolate the contributions of individual variables, such as gender, among a set of correlated variables.

Based on these variable importance scores, the authors construct transparency reports, such as the one displayed in Figure 6.1. This figure displays the importance of features used to predict that "Mr. Z" is likely to be arrested in the future (an outcome which is often used in predictive policing), with each bar corresponding to a feature provided to the classifier, and the y axis displaying the importance score for that feature. In this instance, the race feature is the largest value, indicating that the classifier is indeed discriminating based on race. Thus, in this instance, prediction-level feature importance scores are able to identify that a model is unfairly discriminating based on race.

Alternatives to feature importances

While feature importance scores can provide useful insights, they also have a number of limitations. For instance, they are unable to capture when algorithms learn interactions between variables. There is currently an evolving body of work centered around uncovering and addressing these limitations. These methods focus on explicitly capturing and displaying the interactions learned by a neural network [85, 116], alternative forms of interpretations such as textual explanations [105], identifying influential data points [65], and analyzing nearest neighbors [28, 92].

Chapter 7

Future work

Having introduced the PDR framework for defining and discussing interpretable machine learning, we now leverage it to frame what we feel are the field's most important challenges moving forward. Below, we present open problems tied to each of the part's three main chapters: interpretation desiderata (Chapter 4), model-based interpretability (Chapter 5), and post hoc interpretability (Chapter 6).

7.1 Measuring interpretation desiderata

Currently, there is no clear consensus in the community around how to evaluate interpretation methods, although some recent work has begun to address it [36, 74, 43]. As a result, the standard of evaluation varies considerably across different work, making it challenging both for researchers in the field to measure progress, and for prospective users to select suitable methods. Within the PDR framework, to constitute an improvement, a new interpretation method must improve at least one desideratum (predictive accuracy, descriptive accuracy, or relevancy) without unduly harming the others. While improvements in predictive accuracy are easy to measure, measuring improvements in descriptive accuracy and relevancy remains a challenge. Less important areas for improvement include computational cost and ease of implementation.

Measuring descriptive accuracy

One way to measure an improvement to an interpretation method is to demonstrate that its output better captures what the ML model has learned, i.e. its descriptive accuracy. However, unlike predictive accuracy, descriptive accuracy is generally very challenging to measure or quantify. As a fall-back, researchers often show individual, cherry-picked, interpretations which seem "reasonable". These kinds of evaluations are limited and unfalsifiable. In particular, these results are limited to the few examples shown, and not generally applicable to the entire dataset.

While the community has not settled on a standard evaluation protocol, there are some promising directions. In particular, the use of simulation studies presents a partial solution. In this setting,

a researcher defines a simple generative process, generates a large amount of data from that process, and trains their ML model on that data. Assuming a proper simulation setup, a sufficiently powerful model to recover the generative process, and sufficiently large training data, the trained model should achieve near-perfect generalization accuracy. To compute an evaluation metric, they can then check whether their interpretations recover aspects of the original generative process. For example, [127, 128] train neural networks on a suite of generative models with certain built-in interactions, and test whether their method successfully recovers them. Here, due to the ML model's near-perfect generalization accuracy, we know that the model is likely to have recovered some aspects of the generative process, thus providing a ground truth against which to evaluate interpretations. In a related approach, when an underlying scientific problem has been previously studied, prior experimental findings can serve as a partial ground truth to retrospectively validate interpretations [16].

Demonstrating relevancy to real-world problems

Another angle for developing improved interpretation methods is to improve the relevancy of interpretations for some audience or problem. This is normally done by introducing a novel form of output, such as feature heatmaps [122], rationales [69], feature hierarchies [116] or identifying important elements in the training set [65]. A common pitfall in the current literature is to focus exclusively on the novel output, ignoring what real-world problems it can actually solve. Given the abundance of possible interpretations, it is particularly easy for researchers to propose novel methods which do not actually solve any real-world problems.

There have been two dominant approaches for demonstrating improved relevancy. The first, and strongest, is to directly use the introduced method in solving a domain problem. For instance, in one example discussed above [16], the authors evaluated a new interpretation method (iterative random forests) by demonstrating that it could be used to identify meaningful biological Boolean interactions for use in experiments. In instances like this, where the interpretations are used directly to solve a domain problem, their relevancy is indisputable. A second, less direct, approach is the use of human studies, often through services like Amazon's Mechanical Turk. Here, humans are asked to perform certain tasks, such as evaluating how much they trust a model's predictions [116]. While challenging to properly construct and perform, these studies are vital to demonstrate that new interpretation methods are, in fact, relevant to any potential practitioners. However, one shortcoming of this approach is that it is only possible to use a general audience of AMT crowd-sourced workers, rather than a more relevant, domain-specific audience.

7.2 Model-based

Now that we have discussed the general problem of evaluating interpretations, we highlight important challenges for the two main sub-fields of interpretable machine learning: model-based and post hoc interpretability. Whenever model-based interpretability can achieve reasonable predictive accuracy and relevancy, by virtue of its high descriptive accuracy it is preferable to fitting a more

complex model, and relying upon post hoc interpretability. Thus, the main focus for model-based interpretability is increasing its range of possible use cases by increasing its predictive accuracy through more accurate models and transparent feature engineering. It is worth noting that sometimes a combination of model-based and post hoc interpretations is ideal.

Building accurate and interpretable models

In many instances, model-based interpretability methods fail to achieve a reasonable predictive accuracy. In these cases, practitioners are forced to abandon model-based interpretations in search of more accurate models. Thus, an effective way of increasing the potential uses for model-based interpretability is to devise new modeling methods which produce higher predictive accuracy while maintaining their high descriptive accuracy and relevance. Promising examples of this work include the previously discussed examples on estimating pneumonia risk from patient data [29] and Bayesian models for generating rule lists to estimate a patient's risk of stroke [70]. Detailed directions for this work are suggested in [108].

Tools for feature engineering

When we have more informative and meaningful features, we can use simpler modeling methods to achieve a comparable predictive accuracy. Thus, methods that can produce more useful features broaden the potential uses of model-based interpretations. The first main category of work lies in improved tools for exploratory data analysis. By better enabling researchers to interact with and understand their data, these tools (combined with domain knowledge) provide increased opportunities for them to identify helpful features. Examples include interactive environments [64, 96, 107], tools for visualization [15, 133, 130], and data exploration tools [79, 134]. The second category falls under unsupervised learning, which is often used as a tool for automatically finding relevant structure in data. Improvements in unsupervised techniques such as clustering and matrix factorization could lead to more useful features.

7.3 Post hoc

In contrast to model-based interpretability, much of post hoc interpretability is relatively new, with many foundational concepts still unclear. In particular, we feel that two of the most important questions to be answered are what an interpretation of an ML model should look like, and how post hoc interpretations can be used. One of the most promising potential uses of post hoc interpretations is to increase the predictive accuracy of a model. In related work, it has been pointed out that in high stakes decisions practitioners should be very careful when applying post hoc methods with unknown descriptive accuracy [108].

What should an interpretation of a black-box look like

Given a black-box predictor and real-world problem, it is generally unclear what format, or combination of formats, is best to fully capture a model's behavior. Researchers have proposed a variety of interpretation forms, including feature heatmaps [122], feature hierarchies [116] and identifying important elements in the training set [65]. However, in all instances there is a gap between the relatively simple information provided by these interpretations and what the complex model has actually learned. Moreover, it is unclear if any of the current interpretation forms can fully capture a model's behaviour, or if a new format altogether is needed. How to close that gap, while producing outputs relevant to a particular audience/problem, is an open problem.

Using interpretations to improve predictive accuracy

In some instances, post hoc interpretations uncover that a model has learned relationships a practitioner knows to be incorrect. For instance, prior interpretation work has shown that a binary husky vs. wolf classifier simply learns to identify whether there is snow in the image, ignoring the animals themselves [103]. A natural question to ask is whether it is possible for the practitioner to correct these relationships learned by the model, and consequently increase its predictive accuracy. Given the challenges surrounding simply generating post hoc interpretations, research on their uses has been limited [106, 139]. However, as the field of post hoc interpretations continues to mature, this could be an exciting avenue for researchers to increase the predictive accuracy of their models by exploiting prior knowledge, independently of any other benefits of interpretations.

Part II

Extracting word importance scores from LSTMs

Chapter 8

Word Importance Scores in LSTMs

Neural network language models, especially recurrent neural networks (RNN), are now standard tools for natural language processing. Amongst other things, they are used for translation [123], language modelling [58], and question answering [52]. In particular, the Long Short Term Memory (LSTM) [53] architecture has become a basic building block of neural NLP. Although LSTM's are regularly used in state of the art systems, their operation is not well understood. Besides the basic desire from a scientific viewpoint to clarify their workings, it is often the case that it is important to understand why a machine learning algorithm made a particular choice. Moreover, LSTM's are computationally intensive compared to discrete models with lookup tables and pattern matching.

In this part, we describe a novel method for visualizing the importance of specific inputs for determining the output of an LSTM. We then demonstrate that, by searching for phrases which are consistently important, the importance scores can be used to extract simple phrase patterns consisting of one to five words from a trained LSTM. The phrase extraction is first done in a general document classification framework on two different sentiment analysis datasets. We then demonstrate that it can also be specialized to more complex models by applying it to WikiMovies, a recently introduced question answer dataset. To concretely validate the extracted patterns, we use them as input to a rules-based classifier which approximates the performance of the original LSTM.

8.1 Related Work

There are two lines of related work on visualizing LSTMs. First, [120] and [59] analyse the movement of the raw gate activations over a sequence. [59] is able to identify co-ordinates of c_t that correspond to semantically meaningful attributes such as whether the text is in quotes and how far along the sentence a word is. However, most of the cell co-ordinates are harder to interpret, and in particular, it is often not obvious from their activations which inputs are important for specific outputs.

Another approach that has emerged in the literature [4] [35] [14] is for each word in the document, looking at the norm of the derivative of the loss function with respect to the embedding

parameters for that word. This bridges the gap between high-dimensional cell state and low-dimensional outputs. These techniques are general- they are applicable to visualizing the importance of sets of input coordinates to output coordinates of any differentiable function. In this work, we describe techniques that are designed around the structure of LSTM's, and show that they can give better results in that setting.

A recent line of work [72] [52] [99] [81] has focused on neural network techniques for extracting answers directly from documents. Previous work had focused on Knowledge Bases (KBs), and techniques to map questions to logical forms suitable for querying them. Although they are effective within their domain, KBs are inevitably incomplete, and are thus an unsatisfactory solution to the general problem of question-answering. Wikipedia, in contrast, has enough information to answer a far broader array of questions, but is not as easy to query. Originally introduced in [81], the WikiMovies dataset consists of questions about movies paired with Wikipedia articles.

We now present a novel decomposition of the output of an LSTM into a product of factors, where each term in the product can be interpreted as the contribution of a particular word. Thus, we can assign importance scores to words according to their contribution to the LSTM's prediction

8.2 Decomposing the Output of a LSTM

We now show that we can decompose the numerator of p_i in Equation 1.1 into a product of factors, and interpret those factors as the contribution of individual words to the predicted probability of class i. Define

$$\beta_{i,j} = \exp\left(W_i(o_T \odot (\tanh(c_j) - \tanh(c_{j-1}))\right), \tag{8.1}$$

so that

$$\exp(W_i h_T) = \exp\left(\sum_{j=1}^{T} W_i(o_T \odot (\tanh(c_j) - \tanh(c_{j-1}))\right) = \prod_{j=1}^{T} \beta_{i,j}.$$

As $\tanh(c_j) - \tanh(c_{j-1})$ can be viewed as the update resulting from word j, so $\beta_{i,j}$ can be interpreted as the multiplicative contribution to p_i by word j.

8.3 An Additive Decomposition of the LSTM cell

We will show below that the $\beta_{i,j}$ capture some notion of the importance of a word to the LSTM's output. However, these terms fail to account for how the information contributed by word j is affected by the LSTM's forget gates between words j and T. Consequently, we empirically found that the importance scores from this approach often yield a considerable amount of false positives. A more nuanced approach is obtained by considering the additive decomposition of c_T in equation (8.2), where each term e_j can be interpreted as the contribution to the cell state c_T by word j. By iterating the equation $c_t = f_t c_{t-1} + i_t \tilde{c}_t$, we get that

$$c_T = \sum_{i=1}^{T}(\prod_{j=i+1}^{T} f_j) i_i \tilde{c}_i = \sum_{i=1}^{T} e_{i,T} \tag{8.2}$$

This suggests a natural definition of an alternative score to the $\beta_{i,j}$, corresponding to augmenting the c_j terms with products of forget gates to reflect the upstream changes made to c_j after initially processing word j.

$$\exp(W_i h_T) = \prod_{j=1}^{T} \exp\left(W_i(o_T \odot (\tanh(\sum_{k=1}^{j} e_{k,T}) - \tanh(\sum_{k=1}^{j-1} e_{k,T})))\right) \tag{8.3}$$

$$= \prod_{j=1}^{T} \exp\left(W_i(o_T \odot (\tanh((\prod_{k=j+1}^{T} f_k)c_j) - \tanh((\prod_{k=j}^{T} f_k)c_{j-1})))\right) \tag{8.4}$$

$$= \prod_{j=1}^{T} \gamma_{i,j} \tag{8.5}$$

8.4 Phrase Extraction

We now introduce a technique for using our variable importance scores to extract phrases from a trained LSTM. To do so, we search for phrases which consistently provide a large contribution to the prediction of a particular class relative to other classes. The utility of these patterns is validated by using them as input for a rules based classifier. For simplicity, we focus on the binary classification case.

A phrase can be reasonably described as predictive if, whenever it occurs, it causes a document to both be labelled as a particular class, and not be labelled as any other. As our importance scores introduced above correspond to the contribution of particular words to class predictions, they can be used to score potential patterns by looking at a pattern's average contribution to the prediction of a given class relative to other classes. More precisely, given a collection of D documents $\{\{x_{i,j}\}_{i=1}^{N_d}\}_{j=1}^{D}$, for a given phrase $w_1, ..., w_k$ we can compute scores S_1, S_2 for classes 1 and 2, as well as a combined score S and class C as

$$S_1(w_1, ..., w_k) = \frac{\text{Average}_{j,b}\left\{\prod_{l=1}^{k} \beta_{1,b+l,j} | x_{b+i,j} = w_i, i = 1, ..., k\right\}}{\text{Average}_{j,b}\left\{\prod_{l=1}^{k} \beta_{2,b+l,j} | x_{b+i,j} = w_i, i = 1, ..., k\right\}} \tag{8.6}$$

$$S_2(w_1, .., w_k) = \frac{1}{S_1(w_1, ..., w_k)} \tag{8.7}$$

$$S(w_1, ..., w_k) = \max_i(S_i(w_1, ..., w_k)) \tag{8.8}$$

$$C(w_1, ..., w_k) = \text{argmax}_i(S_i(w_1, ..., w_k)) \tag{8.9}$$

where $\beta_{i,j,k}$ denotes $\beta_{i,j}$ applied to document k.

The numerator of S_1 denotes the average contribution of the phrase to the prediction of class 1 across all occurrences of the phrase. The denominator denotes the same statistic, but for class 2. Thus, if S_1 is high, then $w_1, ..., w_k$ is a strong signal for class 1, and likewise for S_2. We propose to

use S as a score function in order to search for high scoring, representative, phrases which provide insight into the trained LSTM, and C to denote the class corresponding to a phrase.

In practice, the number of phrases is too large to feasibly compute the score of them all. Thus, we approximate a brute force search through a two step procedure. First, we construct a list of candidate phrases by searching for strings of consecutive words j with importance scores $\beta_{i,j} > c$ for any i and some threshold c; in the experiments below we use $c = 1.1$. Then, we score and rank the set of candidate phrases, which is much smaller than the set of all phrases.

8.5 Rules based classifier

The extracted patterns from Chapter 8.4 can be used to construct a simple, rules-based classifier which approximates the output of the original LSTM. Given a document and a list of patterns sorted by descending score given by S, the classifier sequentially searches for each pattern within the document using simple string matching. Once it finds a pattern, the classifier returns the associated class given by C, ignoring the lower ranked patterns. The resulting classifier is interpretable, and despite its simplicity, retains much of the accuracy of the LSTM used to build it.

Chapter 9

Experiments

We now present the results of our experiments.

9.1 Training Details

We implemented all models in Torch using default hyperparameters for weight initializations. For WikiMovies, all documents and questions were pre-processed so that multiple word entities were concatenated into a single word. For a given question, relevant articles were found by first extracting from the question the rarest entity, then returning a list of Wikipedia articles containing any of those words. We use the pre-defined splits into train, validation and test sets, containing 96k, 10k and 10k questions, respectively. The word and hidden representations of the LSTM were both set to dimension 200 for WikiMovies, 300 and 512 for Yelp, and 300 and 150 for Stanford Sentiment Treebank. All models were optimized using Adam [63] with the default learning rate of 0.001 using early stopping on the validation set. For rule extraction using gradient scores, the product in the reward function is replaced by a sum. In both datasets, we found that normalizing the gradient scores by the largest gradient improved results.

9.2 Sentiment Analysis

We first applied the document classification framework to two different sentiment analysis datasets. Originally introduced in [142], the Yelp review polarity dataset was obtained from the Yelp Dataset Challenge and has train and test sets of size 560,000 and 38,000. The task is binary prediction for whether the review is positive (four or five stars) or negative (one or two stars). The reviews are relatively long, with an average length of 160.1 words. We also used the binary classification task from the Stanford Sentiment Treebank (SST) [118], which has less data with train/dev/test sizes of 6920/872/1821, and is done at a sentence level, so has much shorter document lengths.

We report results in Table 9.1 for seven different models. We report state of the art results from prior work using convolutional neural networks; [62] for SST and [142] for Yelp. We also report our LSTM baselines, which are competitive with state of the art, along with the three different

Model	Yelp Polarity	Stanford Sentiment Treebank
Large word2vec CNN [142]	95.4	-
CNN-multichannel [62]	-	88.1
Naive Bayes [118]	-	82.6
LSTM	95.3	87.3
Cell Decomposition Pattern Matching	86.5	76.2
Cell-Difference Pattern Matching	81.2	77.4
Gradient Pattern Matching	65.0	68.0

Table 9.1: Test accuracy for rule extraction and supervised algorithms on sentiment analysis. See Chapter 9.2 for further descriptions of the models.

pattern matching models described above. For SST, we also report prior results using bag of words features with Naive Bayes.

The additive cell decomposition pattern equals or outperforms the cell-difference patterns, which handily beat the gradient results. This coincides with our empirical observations regarding the information contained within the importance measures, and validates our introduced measure. The differences between measures become more pronounced in Yelp, as the longer document sizes provide more opportunities for false positives.

Although our pattern matching algorithms underperform other methods, we emphasize that pure performance is not our goal, nor would we expect more from such a simple model. Rather, the fact that our method provides reasonable accuracy is one piece of evidence, in addition to the qualitative evidence given later, that our word importance scores and extracted patterns contain useful information for understanding the actions of a LSTM.

9.3 WikiMovies

Although document classification comprises a sizeable portion of current research in natural language processing, much recent work focuses on more complex problems and models. In this section, we examine WikiMovies, a recently introduced question answer dataset, and show that with some simple modifications our approach can be adapted to this problem.

Dataset

WikiMovies is a dataset consisting of more than 100,000 questions about movies, paired with relevant Wikipedia articles. It was constructed using the pre-existing dataset MovieLens, paired with templates extracted from the SimpleQuestions dataset [18], a open-domain question answering dataset based on Freebase. They then selected a set of Wikipedia articles about movies by identifying a set of movies from OMDb that had an associated article by title match, and kept the title and first section for each article.

For a given question, the task is to read through the relevant articles and extract the answer, which is contained somewhere within the text. The dataset also provides a list of 43k entities containing all possible answers.

LSTMs for WikiMovies

We propose a simplified version of recent work [72]. Given a pair of question $x_1^q, ..., x_N^q$ and document $x_1^d, ..., x_T^d$, we first compute an embedding for the question using a LSTM. Then, for each word t in the document, we augment the word embedding x_t with the computed question embedding. This is equivalent to adding an additional term which is linear in the question embedding into the gate equations 3-6, allowing the patterns an LSTM absorbs to be directly conditioned upon the question at hand.

$$h_t^q = \text{LSTM}(x_t^q) \tag{9.1}$$

$$h_t = \text{LSTM}(x_t^d \| h_N^q) \tag{9.2}$$

Having run the above model over the document while conditioning on a question, we are given contextual representations $h_1, ..., h_T$ of the words in the document. For each entity t in the document we use p_t to conduct a binary prediction for whether or not the entity is the answer. At test time, we return the entity with the highest probability as the answer.

$$p_t = \text{SoftMax}(W h_t) \tag{9.3}$$

Phrase Extraction

We now introduce some simple modifications that were useful in adapting our pattern extraction framework to this specific task. First, in order to define the set of classifications problems to search over, we treat each entity t within each document as a separate binary classification task with corresponding predictor p_t. Given this set of classification problems, rather than search over the space of all possible phrases, we restrict ourselves to those ending at the entity in question. We also distinguish patterns starting at the beginning of the document with those that do not and introduce an entity character into our pattern vocabulary, which can be matched by any entity. Template examples can be seen below, in Table 10.1. Once we have extracted a list of patterns, in the rules-based classifier we only search for positive examples, and return as the answer the entity matched to the highest ranked positive pattern.

Model	Test accuracy
KV-MemNN IE	68.3
KV-MemNN Doc	76.2
LSTM	80.1
Cell Decomposition Pattern Matching	74.3
Cell-Difference Pattern Matching	69.4
Gradient Pattern Matching	57.4

Table 9.2: Test results for rule extraction and supervised models on WikiMovies, measured in % hits@1. See Chapter 9.3 for further descriptions of the models.

Results

We report results on six different models in Tables 9.2 and 9.3. We show the results from [81], which fit a key-value memory network (KV-MemNN) on representations from information extraction (IE) and raw text (Doc). Next, we report the results of the LSTM described in Chapter 9.3. Finally, we show the results of using three variants of the pattern matching algorithm described in Chapter 9.3: using patterns extracted using the additive decomposition (cell decomposition), difference in cells approaches (cell-difference) and gradient importance scores (gradient), as discussed in Chapter 8.1. Performance is reported using the accuracy of the top hit over all possible answers (all entities), i.e. the hits@1 metric.

As shown in Table 9.2, our LSTM model surpasses the prior state of the art by nearly 4%. Moreover, our automatic pattern matching model approximates the LSTM with less than 6% error, which is surprisingly small for such a simple model, and falls within 2% of the prior state of the art. Similarly to sentiment analysis, we observe a clear ordering of the results across question categories, with our cell decomposition scores providing the best performance, followed by the cell difference and gradient scores.

	KV-MemNN IE	KV-MemNN Doc	LSTM	Cell Decomp RE	Cell Diff RE	Gradient RE
Actor to Movie	66	83	82	78	77	78
Director to Movie	78	91	84	82	84	83
Writer to Movie	72	91	88	88	89	88
Tag to Movie	35	49	49	38	38	38
Movie to Year	75	89	89	84	84	84
Movie to Writer	61	64	86	79	72	63
Movie to Actor	64	64	84	75	73	67
Movie to Director	76	79	88	86	85	45
Movie to Genre	84	86	72	65	42	21
Movie to Votes	92	92	67	67	67	67
Movie to Rating	75	92	33	25	25	25
Movie to Language	62	84	72	67	66	44
Movie to Tags	47	48	58	44	30	6

Table 9.3: Accuracy for rule extraction and supervised models broken down by question category. See Chapter 9.3 for further descriptions of the models.

Chapter 10

Discussion

10.1 Learned patterns

We present extracted patterns for both sentiment tasks, and some WikiMovies question categories in Table 10.1. These patterns are qualitatively sensible, providing further validation of our approach. The increased size of the Yelp dataset allowed for longer phrases to be extracted relative to SST.

10.2 Approximation error between LSTM and pattern matching

Although our approach is able to extract sensible patterns and achieve reasonable performance, there is still an approximation gap between our algorithm and the LSTM. In Table 10.2 we present some examples of instances where the LSTM was able to correctly classify a sentence, and our algorithm was not, along with the pattern used by our algorithm. At first glance, the extracted patterns are sensible, as ”gets the job done” or ”witty dialogue” are phrases you’d expect to see in a positive review of a movie. However, when placed in the broader context of these particular reviews, they cease to be predictive. This demonstrates that, although our work is useful as a first-order approximation, there are still additional relationships that an LSTM is able to learn from data.

10.3 Comparison between word importance measures

While the prediction accuracy of our rules-based classifier provides quantitative validation of the relative merits of our visualizations, the qualitative differences are also insightful. In Table 10.3, we provide a side-by-side comparison between the different measures. As discussed before, the difference in cells technique fails to account for how the updates resulting from word j are affected

Category	Top Patterns
Yelp Polarity Positive	definitely come back again., love love love this place, great food and great service., highly recommended!, will definitely be coming back, overall great experience, love everything about, hidden gem.
Yelp Polarity Negative	worst customer service ever, horrible horrible horrible, won't be back, disappointed in this place, never go back there, not worth the money, not recommend this place
SST Positive	riveting documentary, is a real charmer, funny and touching, well worth your time, journey of the heart, emotional wallop, pleasure to watch, the whole family, cast is uniformly superb, comes from the heart, best films of the year, surprisingly funny, deeply satisfying
SST Negative	pretentious mess ..., plain bad, worst film of the year, disappointingly generic, fart jokes, banal dialogue, poorly executed, waste of time, a weak script, dullard, how bad it is, platitudes, never catches fire, tries too hard to be, bad acting, untalented artistes, derivative horror film, lackluster
WikiMovies movie to writer	film adaptation of Charles Dickens', film adapted from ENT, by journalist ENT, written by ENT
WikiMovies movie to actor	western film starring ENT, starring Ben Affleck, . The movie stars ENT, that stars ENT
WikiMovies movie to language	is a 2014 french, icelandic, finnish, russian, danish, bengali, dutch, original german, zulu,czech, estonian, mandarin, filipino, hungarian

Table 10.1: Selected top patterns using cell decomposition scores, ENT denotes an entity placeholder

Sentiment	Pattern	Sentence
Negative	gets the job done	Still, it gets the job done — a sleepy afternoon rental
Negative	is a great	This is a great subject for a movie, but Hollywood has squandered the opportunity, using is as a prop for a warmed-over melodrama and the kind of choreographed mayhem that director John Woo has built his career on.
Negative	happy ending	The story loses its bite in a last-minute happy ending that's even less plausible than the rest of the picture.
Negative	witty dialogue	An often-deadly boring, strange reading of a classic whose witty dialogue is treated with a baffling casual approach.
Positive	mess	The film is just a big, gorgeous, mind-blowing, breath-taking mess

Table 10.2: Examples from Stanford sentiment treebank which are correctly labelled by our LSTM and incorrectly labelled by our rules-based classifier. The matched pattern is highlighted

by the LSTM's forget gates between when the word is initially processed and the answer. Consequently, we empirically found that without the interluding forget gates to dampen cell movements, the variable importance scores were far noisier than in additive cell decomposition approach. Under the additive cell decomposition, it identifies the phrase 'it stars', as well as the actor's name Aqib Khan as being important, a sensible conclusion. Moreover, the vast majority of words are labelled with an importance score of 1, corresponding to irrelevant. On the other hand, the difference in cells approach yields widely changing importance scores, which are challenging to interpret. In terms of noise, the gradient measures seem to lie somewhere in the middle. These patterns are broadly consistent with what we have observed, and provide qualitative validation of our metrics.

10.4 Conclusion

In this part, we introduced a novel method for visualizing the importance of specific inputs in determining the output of an LSTM. By searching for phrases which consistently provide large contributions, we are able to distill trained, state of the art, LSTMs into an ordered set of representative phrases. We quantitatively validate the extracted phrases through their performance in a simple, rules-based classifier. Results are shown in a general document classification framework, then specialized to a more complex, recently introduced, question answer dataset. Our introduced measures provide superior predictive ability and cleaner visualizations relative to prior work. We believe that this represents an exciting new paradigm for analysing the behaviour of LSTM's.

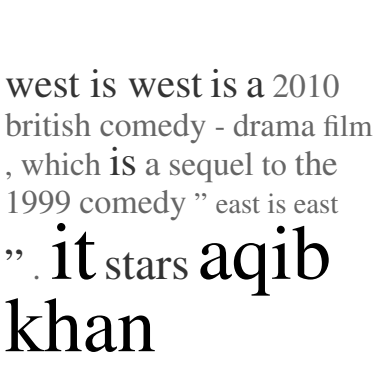
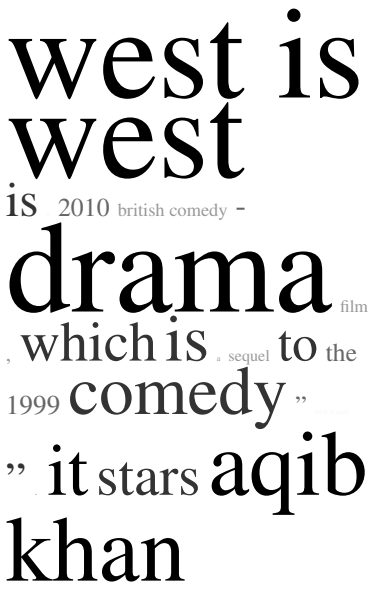
Additive cell decomposition	Difference in cell values	Gradient
		

Table 10.3: Comparison of importance scores acquired by three different approaches, conditioning on the question ”the film west is west starred which actors?”. Bigger and darker means more important.

Part III

Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs

Chapter 11

Contextual Decomposition of LSTMs

In comparison with simpler linear models, techniques from deep learning have achieved impressive accuracy by effectively learning non-linear interactions between features. However, due to our inability to describe the learned interactions, this improvement in accuracy has come at the cost of state of the art predictive algorithms being commonly regarded as black-boxes. In the domain of natural language processing (NLP), Long Short Term Memory networks (LSTMs) [53] have become a basic building block, yielding excellent performance across a wide variety of tasks [123] [99] [80], while remaining largely inscrutable.

In this part, we introduce contextual decomposition (CD), a novel interpretation method for explaining individual predictions made by an LSTM without any modifications to the underlying model. CD extracts information about not only which words contributed to a LSTM's prediction, but also how they were combined in order to yield the final prediction. By mathematically decomposing the LSTM's output, we are able to disambiguate the contributions made at each step by different parts of the sentence.

To validate the CD interpretations extracted from an LSTM, we evaluate on the problem of sentiment analysis. In particular, we demonstrate that CD is capable of identifying words and phrases of differing sentiment within a given review. CD is also used to successfully extract positive and negative negations from an LSTM, something that has not previously been done. As a consequence of this analysis, we also show that prior interpretation methods produce scores which have document-level information built into them in complex, unspecified ways. For instance, prior work often identifies strongly negative phrases contained within positive reviews as neutral, or even positive.

11.1 Related work

The most relevant prior work on interpreting LSTMs has focused on approaches for computing word-level importance scores, with evaluation protocols varying greatly. [86] introduced a decomposition of the LSTM's output embedding into a sum over word coefficients, and demonstrated that those coefficients are meaningful by using them to distill LSTMs into rules-based classifiers.

[71] took a more black box approach, called Leave One Out, by observing the change in log probability resulting from replacing a given word vector with a zero vector, and relied solely on anecdotal evaluation. Finally, [122] presents a general gradient-based technique, called Integrated Gradients, which was validated both theoretically and with empirical anecdotes. In contrast to our proposed method, this line of work has been limited to word-based importance scores, ignoring the interactions between variables which make LSTMs so accurate.

Another line of work [59] [120] has focused on analysing the movement of raw gate activations over a sequence. [59] was able to identify some co-ordinates of the cell state that correspond to semantically meaningful attributes, such as whether the text is in quotes. However, most of the cell co-ordinates were uninterpretable, and it is not clear how these co-ordinates combine to contribute to the actual prediction.

Decomposition-based approaches to interpretation have also been applied to convolutional neural networks (CNNs) [11] [113]. However, they have been limited to producing pixel-level importance scores, ignoring interactions between pixels, which are clearly quite important. Our approach is similar to these in that it computes an exact decomposition, but we leverage the unique gating structure of LSTMs in order to extract interactions.

Attention based models [13] offer another means of providing some interpretability. Such models have been successfully applied to many problems, yielding improved performance [109] [136]. In contrast to other word importance scores, attention is limited in that it only provides an indirect indicator of importance, with no directionality, i.e. what class the word is important for. Although attention weights are often cited anecdotally, they have not been evaluated, empirically or otherwise, as an interpretation technique. As with other prior work, attention is also incapable of describing interactions between words.

11.2 Methods

Given an arbitrary phrase contained within an input, we present a novel decomposition of the output of an LSTM into a sum of two contributions: those resulting solely from the given phrase, and those involving other factors. The key insight behind this decomposition is that the gating dynamics unique to LSTMs are a vehicle for modeling interactions between variables.

We now introduce contextual decomposition, our proposed method for interpreting LSTMs. Given an arbitrary phrase $x_q, ..., x_r$, where $1 \leq q \leq r \leq T$, we now decompose each output and cell state c_t, h_t in Equations 1.6 and 1.7 into a sum of two contributions.

$$h_t = \beta_t + \gamma_t \tag{11.1}$$

$$c_t = \beta_t^c + \gamma_t^c \tag{11.2}$$

The decomposition is constructed so that β_t corresponds to contributions made solely by the given phrase to h_t, and that γ_t corresponds to contributions involving, at least in part, elements outside of the phrase. β_t^c and γ_t^c represent analogous contributions to c_t.

Using this decomposition for the final output state Wh_T in Equation 1.1 yields

$$p = \text{SoftMax}(W\beta_T + W\gamma_T) \tag{11.3}$$

Here $W\beta_T$ provides a quantitative score for the phrase's contribution to the LSTM's prediction. As this score corresponds to the input to a logistic regression, it may be interpreted in the same way as a standard logistic regression coefficient.

11.3 Disambiguating interactions between gates

In the cell update Equation 1.6, neuron values in each of i_t and g_t are independently determined by both the contribution at that step, x_t, as well as prior context provided by $h_{t-1} = \beta_{t-1} + \gamma_{t-1}$. Thus, in computing the element-wise product $i_t \odot g_t$, often referred to as gating, contributions made by x_t to i_t interact with contributions made by h_t to g_t, and vice versa.

We leverage this simple insight to construct our decomposition. First, assume that we have a way of linearizing the gates and updates in Equations 1.3, 1.4, 1.5 so that we can write each of them as a linear sum of contributions from each of their inputs.

$$\begin{aligned} i_t &= \sigma(W_i x_t + V_i h_{t-1} + b_i) && (11.4)\\ &= L_\sigma(W_i x_t) + L_\sigma(V_i h_{t-1}) + L_\sigma(b_i) && (11.5) \end{aligned}$$

When we use this linearization in the cell update Equation 1.6, the products between gates become products over linear sums of contributions from different factors. Upon expanding these products, the resulting cross-terms yield a natural interpretation as being interactions between variables. In particular, cross-terms can be assigned as to whether they resulted solely from the phrase, e.g. $L_\sigma(V_i\beta_{t-1}) \odot L_{\tanh}(V_g\beta_{t-1})$, from some interaction between the phrase and other factors, e.g. $L_\sigma(V_i\beta_{t-1}) \odot L_{\tanh}(V_g\gamma_{t-1})$, or purely from other factors, e.g. $L_\sigma(b_i) \odot L_{\tanh}(V_g\gamma_{t-1})$.

Mirroring the recurrent nature of LSTMs, the above insights allow us to recursively compute our decomposition, with the initializations $\beta_0 = \beta_0^c = \gamma_0 = \gamma_0^c = 0$. We derive below the update equations for the case where $q \leq t \leq r$, so that the current time step is contained within the phrase. The other case is similar, and the general recursion formula is provided in Appendix B.2.

For clarity, we decompose the two products in the cell update Equation 1.6 separately. As discussed above, we simply linearize the gates involved, expand the resulting product of sums, and group the cross-terms according to whether or not their contributions derive solely from the specified phrase, or otherwise. Terms are determined to derive solely from the specified phrase if they involve products from some combination of $\beta_{t-1}, \beta_{t-1}^c, x_t$ and b_i or b_g (but not both). When t is not within the phrase, products involving x_t are treated as not deriving from the phrase.

$$\begin{aligned} f_t \odot c_{t-1} =& (L_\sigma(W_f x_t) + L_\sigma(V_f\beta_{t-1}) + L_\sigma(V_f\gamma_{t-1}) + L_\sigma(b_f)) \odot (\beta_{t-1}^c + \gamma_{t-1}^c) && (11.6)\\ =& ([L_\sigma(W_f x_t) + L_\sigma(V_f\beta_{t-1}) + L_\sigma(b_f)] \odot \beta_{t-1}^c) && (11.7)\\ &+ (L_\sigma(V_f\gamma_{t-1}) \odot \beta_{t-1}^c + f_t \odot \gamma_{t-1}^c)\\ =& \beta_t^f + \gamma_t^f && (11.8) \end{aligned}$$

$$
\begin{aligned}
i_t \odot g_t =& [L_\sigma(W_i x_t) + L_\sigma(V_i \beta_{t-1}) + L_\sigma(V_i \gamma_{t-1}) + L_\sigma(b_i)] && (11.9)\\
& \odot [L_{\tanh}(W_g x_t) + L_{\tanh}(V_g \beta_{t-1}) + L_{\tanh}(V_g \gamma_{t-1}) + L_{\tanh}(b_g)] \\
=& [L_\sigma(W_i x_t) \odot [L_{\tanh}(W_g x_t) + L_{\tanh}(V_g \beta_{t-1}) + L_{\tanh}(b_g)] && (11.10)\\
& + L_\sigma(V_i \beta_{t-1}) \odot [L_{\tanh}(W_g x_t) + L_{\tanh}(V_g \beta_{t-1}) + L_{\tanh}(b_g)] \\
& + L_\sigma(b_i) \odot [L_{\tanh}(W_g x_t) + L_{\tanh}(V_g \beta_{t-1})]] \\
& + [L_\sigma(V_i \gamma_{t-1}) \odot g_t + i_t \odot L_{\tanh}(V_g \gamma_{t-1}) - L_\sigma(V_i \gamma_{t-1}) \odot L_{\tanh}(V_g \gamma_{t-1}) \\
& + L_\sigma(b_i) \odot L_{\tanh}(b_g)] \\
=& \beta_t^u + \gamma_t^u && (11.11)
\end{aligned}
$$

Having decomposed the two components of the cell update equation, we can attain our decomposition of c_t by summing the two contributions.

$$\beta_t^c = \beta_t^f + \beta_t^u \tag{11.12}$$

$$\gamma_t^c = \gamma_t^f + \gamma_t^u \tag{11.13}$$

Once we have computed the decomposition of c_t, it is relatively simple to compute the resulting transformation of h_t by linearizing the $\tanh$ function in 1.7. Note that we could similarly decompose the output gate as we treated the forget gate above, but we empirically found this to not produce improved results.

$$
\begin{aligned}
h_t &= o_t \odot \tanh(c_t) && (11.14)\\
&= o_t \odot [L_{\tanh}(\beta_t^c) + L_{\tanh}(\gamma_t^c)] && (11.15)\\
&= o_t \odot L_{\tanh}(\beta_c^t) + o_t \odot L_{\tanh}(\gamma_t^c) && (11.16)\\
&= \beta_t + \gamma_t && (11.17)
\end{aligned}
$$

11.4 Linearizing Activation Functions

We now describe the linearizing functions $L_\sigma, L_{\tanh}$ used in the above decomposition. Formally, for arbitrary $\{y_1, ..., y_N\} \in \mathbb{R}$, where $N \leq 4$, the problem is how to write

$$\tanh(\sum_{i=1}^{N} y_i) = \sum_{i=1}^{N} L_{\tanh}(y_i) \tag{11.18}$$

In the cases where there is a natural ordering to $\{y_i\}$, prior work [86] has used a telescoping sum consisting of differences of partial sums as a linearization technique, which we show below.

$$L'_{\tanh}(y_k) = \tanh(\sum_{j=1}^{k} y_j) - \tanh(\sum_{j=1}^{k-1} y_j) \tag{11.19}$$

However, in our setting $\{y_i\}$ contains terms such as β_{t-1}, γ_{t-1} and x_t, which have no clear ordering. Thus, there is no natural way to order the sum in Equation 11.19. Instead, we compute an average over all orderings. Letting $\pi_1, ..., \pi_{M_N}$ denote the set of all permutations of $1, ..., N$, our score is given below. Note that when $\pi_i(j) = j$, the corresponding term is equal to equation 11.19.

$$L_{\tanh}(y_k) = \frac{1}{M_N} \sum_{i=1}^{M_N} [\tanh(\sum_{j=1}^{\pi_i^{-1}(k)} y_{\pi_i(j)}) - \tanh(\sum_{j=1}^{\pi_i^{-1}(k)-1} y_{\pi_i(j)})] \tag{11.20}$$

L_σ can be analogously derived. When one of the terms in the decomposition is a bias, we saw improvements when restricting to permutations where the bias is the first term.

As N only ranges between 2 and 4, this linearization generally takes very simple forms. For instance, when $N = 2$, the contribution assigned to y_1 is

$$L_{\tanh}(y_1) = \frac{1}{2}([\tanh(y_1) - \tanh(0)] + [\tanh(y_2 + y_1) - \tanh(y_1)]) \tag{11.21}$$

This linearization was presented in a scalar context where $y_i \in \mathbb{R}$, but trivially generalizes to the vector setting $y_i \in \mathbb{R}^{d_2}$. It can also be viewed as an approximation to Shapely values, as discussed in [78] and [113].

Chapter 12

Experiments

We now describe our empirical validation of CD on the task of sentiment analysis. First, we verify that, on the standard problem of word-level importance scores, CD compares favorably to prior work. Then we examine the behavior of CD for word and phrase level importance in situations involving compositionality, showing that CD is able to capture the composition of phrases of differing sentiment. Finally, we show that CD is capable of extracting instances of positive and negative negation.

12.1 Training Details

We first describe the process for fitting models which are used to produce interpretations. As the primary intent of this part is not predictive accuracy, we used standard best practices without much tuning. We implemented all models in Torch using default hyperparameters for weight initializations. All models were optimized using Adam [63] with the default learning rate of 0.001 using early stopping on the validation set. For the linear model, we used a bag of vectors model, where we sum pre-trained Glove vectors [95] and add an additional linear layer from the word embedding dimension, 300, to the number of classes, 2. We fine tuned both the word vectors and linear parameters. We will use the two data sets described below to validate our new CD method.

Stanford Sentiment Treebank

We trained an LSTM model on the binary version of the Stanford Sentiment Treebank (SST) [118], a standard NLP benchmark which consists of movie reviews ranging from 2 to 52 words long. In addition to review-level labels, it also provides labels for each phrase in the binarized constituency parse tree. Following the hyperparameter choices in [125], the word and hidden representations of our LSTM were set to 300 and 168, and word vectors were initialized to pretrained Glove vectors [95]. Our LSTM attains 87.2% accuracy, and we also train a logistic regression model with bag of words features, which attains 83.2% accuracy.

Yelp Polarity

Originally introduced in [142], the Yelp review polarity dataset was obtained from the Yelp Dataset Challenge and has train and test sets of sizes 560,000 and 38,000. The task is binary prediction for whether the review is positive (four or five stars) or negative (one or two stars). The reviews are relatively long, with an average length of 160.1 words. Following the guidelines from [142], we implement an LSTM model which attains 4.6% error, and an ngram logistic regression model, which attains 5.7% error. For computational reasons, we report interpretation results on a random subset of sentences of length at most 40 words. When computing integrated gradient scores, we found that numerical issues produced unusable outputs for roughly 6% of the samples. These reviews are excluded.

Interpretation Baselines

We compare the interpretations produced by CD against four state of the art baselines: cell decomposition [86], integrated gradients [122], leave one out [71], and gradient times input. We refer the reader to Chapter 11.1 for descriptions of these algorithms. For our gradient baseline, we compute the gradient of the output probability with respect to the word embeddings, and report the dot product between the word vector and its gradient. For integrated gradients, producing reasonable values required extended experimentation and communication with the creators regarding the choice of baselines and scaling issues. We ultimately used sequences of periods for our baselines, and rescaled the scores for each review by the standard deviation of the scores for that review, a trick not previously mentioned in the literature. To obtain phrase scores for word-based baselines integrated gradients, cell decomposition, and gradients, we sum the scores of the words contained within the phrase.

12.2 Unigram (word) scores

Before examining the novel, phrase-level dynamics of CD, we first verify that it compares favorably to prior work for the standard use case of producing unigram coefficients. When sufficiently accurate in terms of prediction, logistic regression coefficients are generally treated as a gold standard for interpretability. In particular, when applied to sentiment analysis the ordering of words given by their coefficient value provides a qualitatively sensible measure of importance. Thus, when determining the validity of coefficients extracted from an LSTM, we should expect there to be a meaningful relationship between the CD scores and logistic regression coefficients.

In order to evaluate the word-level coefficients extracted by the CD method, we construct scatter plots with each point consisting of a single word in the validation set. The two values plotted correspond to the coefficient from logistic regression and importance score extracted from the LSTM. For a quantitative measure of accuracy, we use pearson correlation coefficient.

We report quantitative and qualitative results in Appendix B.1. For SST, CD and integrated gradients, with correlations of 0.76 and 0.72, respectively, are substantially better than other methods,

with correlations of at most 0.51. On Yelp, the gap is not as big, but CD is still very competitive, having correlation 0.52 with other methods ranging from 0.34 to 0.56. Having verified reasonably strong results in this base case, we now proceed to show the benefits of CD.

12.3 Identifying dissenting subphrases

We now show that, for phrases of at most five words, existing methods are unable to recognize subphrases with differing sentiments. For example, consider the phrase "used to be my favorite", which is of negative sentiment. The word "favorite", however, is strongly positive, having a logistic regression coefficient in the 93rd percentile. Nonetheless, existing methods consistently rank "favorite" as being highly negative or neutral. In contrast, as shown in Table 12.1, CD is able to identify "my favorite" as being strongly positive, and "used to be" as strongly negative. A similar dynamic also occurs with the phrase "not worth the time". The main justification for using LSTMs over simpler models is precisely that they are able to capture these kinds of interactions. Thus, it is important that an interpretation algorithm is able to properly uncover how the interactions are being handled.

Using the above as a motivating example, we now show that a similar trend holds throughout the Yelp polarity dataset. In particular, we conduct a search for situations similar to the above, where a strongly positive/negative phrase contains a strongly dissenting subphrase. Phrases are scored using the logistic regression with n-gram features described in Chapter 12.1, and included if their absolute score is over 1.5. We then examine the distribution of scores for the dissenting subphrases, which are analogous to "favorite".

For an effective interpretation algorithm, the distribution of scores for positive and negative dissenting subphrases should be significantly separate, with positive subphrases having positive scores, and vice versa. However, as can be seen in Appendix B.1, for prior methods these two distributions are nearly identical. The CD distributions, on the other hand, are significantly separate, indicating that what we observed anecdotally above holds in a more general setting.

12.4 Examining high-level compositionality

We now show that prior methods struggle to identify cases where a sizable portion of a review (between one and two thirds) has polarity different from the LSTM's prediction. For instance, consider the review in Table 12.2, where the first phrase is clearly positive, but the second phrase causes the review to ultimately be negative. CD is the only method able to accurately capture this dynamic.

By leveraging the phrase-level labels provided in SST, we can show that this pattern holds in the general case. In particular, we conduct a search for reviews similar to the above example. The search criteria are whether a review contains a phrase labeled by SST to be of opposing sentiment to the review-level SST label, and is between one and two thirds the length of the review.

Attribution Method	Heat Map									
Gradient	used	to	be	my	favorite		not	worth	the	time
Leave One Out [71]	used	to	be	my	favorite		not	worth	the	time
Cell decomposition [86]	used	to	be	my	favorite		not	worth	the	time
Integrated gradients [122]	used	to	be	my	favorite		not	worth	the	time
Contextual decomposition	used	to	be	my	favorite		not	worth	the	time

Legend: Very Negative | Negative | Neutral | Positive | Very Positive

Table 12.1: Heat maps for portion of yelp review with different attribution techniques. Only CD captures that "favorite" is positive.

In Appendix B.1, we show the distribution of the resulting positive and negative phrases for different attribution methods. A successful interpretation method would have a sizable gap between these two distributions, with positive phrases having mostly positive scores, and negative phrases mostly negative. However, prior methods struggle to satisfy these criteria. 87% of all positive phrases are labelled as negative by integrated gradients, and cell decompositions [86] even have the distributions flipped, with negative phrases yielding more positive scores than the positive phrases. CD, on the other hand, provides a very clear difference in distributions. To quantify this separation between positive and negative distributions, we examine a two-sample Kolmogorov-Smirnov one-sided test statistic, a common test for the difference of distributions with values ranging from 0 to 1. CD produces a score of 0.74, indicating a strong difference between positive and negative distributions, with other methods achieving scores of 0 (cell decomposition), 0.33 (integrated gradients), 0.58 (leave one out) and 0.61 (gradient), indicating weaker distributional differences. Given that gradient and leave one out were the weakest performers in unigram scores, this provides strong evidence for the superiority of CD.

12.5 Contextual decomposition (CD) captures negation

In order to understand an LSTM's prediction mechanism, it is important to understand not just the contribution of a phrase, but how that contribution is computed. For phrases involving negation, we now demonstrate that we can use CD to empirically show that our LSTM learns a negation mechanism.

Using the phrase labels in SST, we search over the training set for instances of negation. In particular, we search for phrases of length less than ten with the first child containing a negation phrase (such as "not" or "lacks", full list provided in Appendix B.3) in the first two words, and the second child having positive or negative sentiment. Due to noise in the labels, we also included

Attribution Method	Heat Map
Gradient	It's easy to love Robin Tunney – she's pretty and she can act – but it gets harder and harder to understand her choices.
Leave one out [71]	It's easy to love Robin Tunney – she's pretty and she can act – but it gets harder and harder to understand her choices.
Cell decomposition [86]	It's easy to love Robin Tunney – she's pretty and she can act – but it gets harder and harder to understand her choices.
Integrated gradients [122]	It's easy to love Robin Tunney – she's pretty and she can act – but it gets harder and harder to understand her choices.
Contextual decomposition	It's easy to love Robin Tunney – she's pretty and she can act – but it gets harder and harder to understand her choices.

Legend: Very Negative | Negative | Neutral | Positive | Very Positive

Table 12.2: Heat maps for portion of review from SST with different attribution techniques. Only CD captures that the first phrase is positive.

phrases where the entire phrase was non-neutral, and the second child contained a non-neutral phrase. We identify both positive negation, such as "isn't a bad film", and negative negation, such as "isn't very interesting", where the direction is given by the SST-provided label of the phrase.

For a given negation phrase, we extract a negation interaction by computing the CD score of the entire phrase and subtracting the CD scores of the phrase being negated and the negation term itself. The resulting score can be interpreted as an n-gram feature. Note that, of the methods we compare against, only leave one out is capable of producing such interaction scores. For reference, we also provide the distribution of all interactions for phrases of length less than 5.

We present the distribution of extracted scores in Figure 12.1. For CD, we can see that there is a clear distinction between positive and negative negations, and that the negation interactions are centered on the outer edges of the distribution of interactions. Leave one out is able to capture some of the interactions, but has a noticeable overlap between positive and negative negations around zero, indicating a high rate of false negatives.

12.6 Identifying similar phrases

Another benefit of using CDs for interpretation is that, in addition to providing importance scores, it also provides dense embeddings for arbitrary phrases and interactions, in the form of β_T dis-

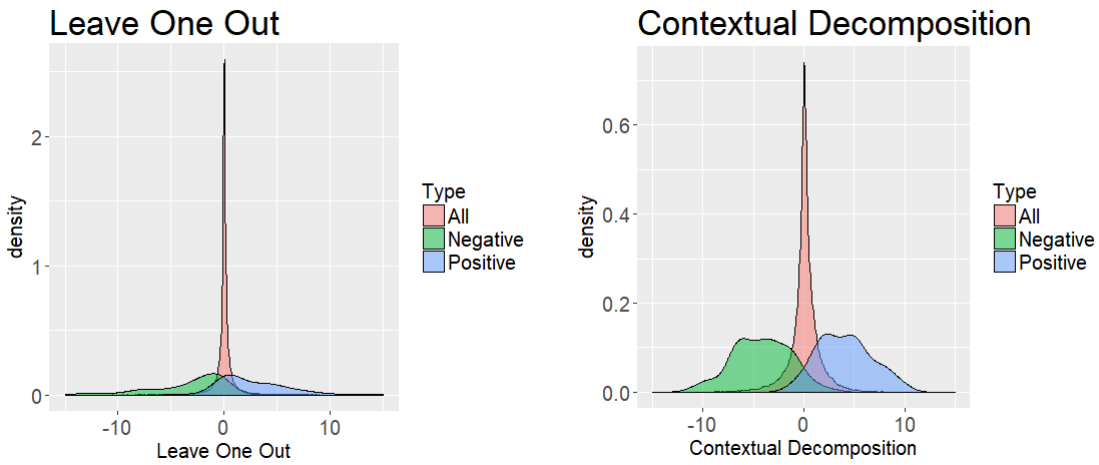


Figure 12.1: Distribution of scores for positive and negative negation coefficients relative to all interaction coefficients. Only leave one out and CD are capable of producing these interaction scores.

not entertaining	**not bad**	**very funny**	**entertaining**	**bad**
not funny	never dull	well-put-together piece	intelligent	dull
not engaging	n't drag	entertaining romp	engaging	drag
never satisfactory	never fails	very good	satisfying	awful
not well	without sham	surprisingly sweet	admirable	tired
not fit	without missing	very well-written	funny	dreary

Table 12.3: Nearest neighbours for selected unigrams and interactions using CD embeddings

cussed in Chapter 11. We anecdotally show that similarity in this embedding space corresponds to semantic similarity in the context of sentiment analysis.

In particular, for all words and binary interactions, we compute the average embedding β_T produced by CD across the training and validation sets. In Table 12.3, we show the nearest neighbours using a cosine similarity metric. The results are qualitatively sensible for three different kinds of interactions: positive negation, negative negation and modification, as well as positive and negative words. Note that we for positive and negative words, we chose the positive/negative parts of the negations, in order to emphasize that CD can disentangle this composition.

12.7 Conclusion

In this part, we have proposed contextual decomposition (CD), an algorithm for interpreting individual predictions made by LSTMs without modifying the underlying model. In both NLP and general applications of LSTMs, CD produces importance scores for words (single variables in general), phrases (several variables together) and word interactions (variable interactions). Using two sentiment analysis datasets for empirical validation, we first show that for information also produced by prior methods, such as word-level scores, our method compares favorably. More importantly, we then show that CD is capable of identifying phrases of varying sentiment, and extracting meaningful word (or variable) interactions. This movement beyond word-level importance is critical for understanding a model as complex and highly non-linear as LSTMs.

Part IV

Hierarchical interpretations for neural network predictions

Chapter 13

Methods

Deep neural networks (DNNs) have recently demonstrated impressive predictive performance due to their ability to learn complex, non-linear, relationships between variables. However, the inability to effectively visualize these relationships has led DNNs to be characterized as black boxes. Consequently, their use has been limited in fields such as medicine (e.g. medical image classification [75]), policy-making (e.g. classification aiding public policy makers [25]), and science (e.g. interpreting the contribution of a stimulus to a biological measurement [9]). Moreover, the use of black-box models like DNNs in industrial settings has come under increasing scrutiny as they struggle with issues such as fairness [37] and regulatory pressure [46].

To ameliorate these problems, we introduce the use of hierarchical interpretations to explain DNN predictions. Our proposed method, agglomerative contextual decomposition (ACD)[1], is a general technique that can be applied to a wide range of DNN architectures and data types. Given a prediction from a trained DNN, ACD produces a hierarchical clustering of the input features, along with the contribution of each cluster to the final prediction. This hierarchy is optimized to identify clusters of features that the DNN learned are predictive (see Figure 13.1).

The development of ACD consists of two novel contributions. First, importance scores for groups of features are obtained by generalizing contextual decomposition (CD), a previous method for obtaining importance scores for LSTMs [85]. This work extends CD to arbitrary DNN architectures, including convolutional neural networks (CNNs). Second, most importantly, we introduce the idea of hierarchical saliency, where a group-level importance measure, in this case CD, is used as a joining metric in an agglomerative clustering procedure. While we focus on DNNs and use CD as our importance measure, this concept is general, and could be readily applied to any model with a suitable measure for computing importances of groups of variables.

We demonstrate the utility of ACD on both long short term memory networks (LSTMs) [53] trained on the Stanford Sentiment Treebank (SST) [118] and CNNs trained on MNIST [68] and ImageNet [110]. Through human experiments, we show that ACD produces intuitive visualizations that enable users to better reason about and trust DNNs. In particular, given two DNN models, we show that users can use the output of ACD to select the model with higher predictive accuracy, and

[1] Code and scripts for running ACD and experiments available at `https://github.com/csinva/acd`

that overall they rank ACD as more trustworthy than prior interpretation methods. In addition, we demonstrate that ACD's hierarchy is robust to adversarial perturbations [124] in CNNs.

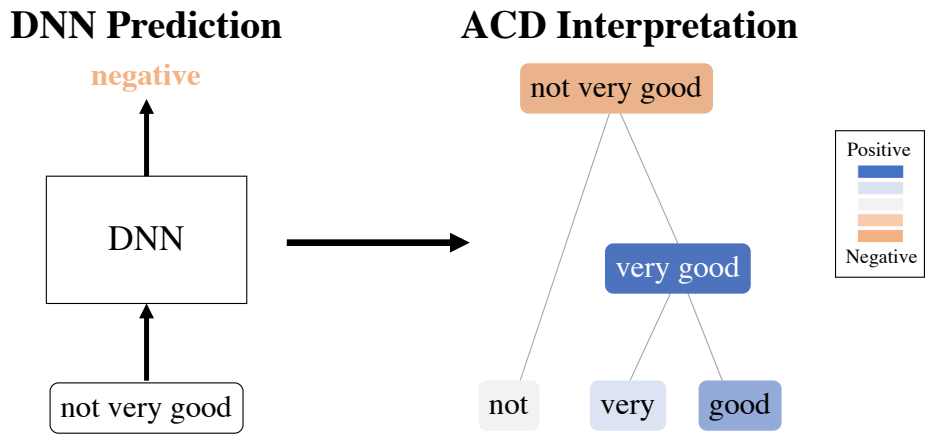


Figure 13.1: ACD illustrated through the toy example of predicting the phrase "not very good" as negative. Given the network and prediction, ACD constructs a hierarchy of meaningful phrases and provides importance scores for each identified phrase. In this example, ACD identifies that "very" modifies "good" to become the very positive phrase "very good", which is subsequently negated by "not" to produce the negative phrase "not very good". Best viewed in color.

13.1 Related Work

Interpreting DNNs is a growing field [87] spanning a range of techniques including feature visualization [89, 137], analyzing learned weights [127] and others [42, 8, 141]. Our work focuses on local interpretations, where the task is to interpret individual predictions made by a DNN.

Local interpretation Most prior work has focused on assigning importance to individual features, such as pixels in an image or words in a document. There are several methods that give feature-level importance for different architectures. They can be categorized as gradient-based [119, 122, 111, 12], decomposition-based [86, 114, 11] and others [32, 39, 103, 143], with many similarities among the methods [7, 78].

By contrast, there are relatively few methods that can extract the interactions between features that a DNN has learned. In the case of LSTMs, [85] demonstrated the limitations of prior work on interpretation using word-level scores, and introduced contextual decomposition (CD), an algorithm for producing phrase-level importance scores from LSTMs. Another simple baseline is occlusion, where a group of features is set to some reference value, such as zero, and the importance of the group is defined to be the resulting decrease in the prediction value [140, 71]. Given an importance score for groups of features, no existing work addresses how to search through the

many possible groups of variables in order to find a small set to show to users. To address this problem, this work introduces hierarchical interpretations as a principled way to search for and display important groups.

Hierarchical importance Results from psychology and philosophy suggest that people prefer explanations that are simple but informative [50, 102] and include the appropriate amount of detail [60]. However, there is no existing work that is both powerful enough to capture interactions between features, and simple enough to not require a user to manually search through the large number of available feature groups. To remedy this, we propose a hierarchical clustering procedure to identify and visualize, out of the considerable number of feature groups, which ones contain meaningful interactions and should be displayed to the end user. In doing so, ACD aims to be informative enough to capture meaningful feature interactions while displaying a sufficiently small subset of all feature groups to maintain simplicity.

The following sections introduce ACD through two contributions: Chapter 13.2 proposes a generalization of CD from LSTMs to arbitrary DNNs, and Chapter 13.3 explains the main contribution: how to combine these CD scores with hierarchical clustering to produce ACD.

13.2 Contextual Decomposition (CD) importance scores for general DNNs

In order to generalize CD to a wider range of DNNs, we first reformulate the original CD algorithm into a more generic setting than originally presented. For a given DNN $f(x)$, we can represent its output as a SoftMax operation applied to logits $g(x)$. These logits, in turn, are the composition of L layers g_i, such as convolutional operations or ReLU non-linearities.

$$f(x) = \text{SoftMax}(g(x)) = \text{SoftMax}(g_L(g_{L-1}(...(g_2(g_1(x))))))) \tag{13.1}$$

Given a group of features $\{x_j\}_{j \in S}$, our generalized CD algorithm, $g^{CD}(x)$, decomposes the logits $g(x)$ into a sum of two terms, $\beta(x)$ and $\gamma(x)$. $\beta(x)$ is the importance measure of the feature group $\{x_j\}_{j \in S}$, and $\gamma(x)$ captures contributions to $g(x)$ not included in $\beta(x)$.

$$g^{CD}(x) = (\beta(x), \gamma(x)) \tag{13.2}$$

$$\beta(x) + \gamma(x) = g(x) \tag{13.3}$$

To compute the CD decomposition for $g(x)$, we define layer-wise CD decompositions $g_i^{CD}(x) = (\beta_i, \gamma_i)$ for each layer $g_i(x)$. Here, β_i corresponds to the importance measure of $\{x_j\}_{j \in S}$ to layer i, and γ_i corresponds to the contribution of the rest of the input to layer i. To maintain the decomposition we require $\beta_i + \gamma_i = g_i(x)$ for each i. We then compute CD scores for the full network by composing these decompositions.

$$g^{CD}(x) = g_L^{CD}(g_{L-1}^{CD}(...(g_2^{CD}(g_1^{CD}(x))))) \tag{13.4}$$

Previous work [85] introduced decompositions g_i^{CD} for layers used in LSTMs. The generalized CD described here extends CD to other widely used DNNs, by introducing layer-wise CD decompositions for convolutional, max-pooling, ReLU non-linearity and dropout layers. Doing so generalizes CD scores from LSTMs to a wide range of neural architectures, including CNNs with residual and recurrent architectures.

At first, these decompositions were chosen through an extension of the CD rules detailed in [85], yielding a similar algorithm to that developed concurrently by [44]. However, we found that this algorithm did not perform well on deeper, ImageNet CNNs. We subsequently modified our CD algorithm by partitioning the biases in the convolutional layers between γ_i and β_i in Equation 13.5, and modifying the decomposition used for ReLUs in Equation 13.10. We show the effects of these two changes in Supplement S7, and give additional intuition in Supplement S1.

When g_i is a convolutional or fully connected layer, the layer operation consists of a weight matrix W and a bias b. The weight matrix can be multiplied with β_{i-1} and γ_{i-1} individually, but the bias must be partitioned between the two. We partition the bias proportionally based on the absolute value of the layer activations. For the convolutional layer, this equation yields only one activation of the output; it must be repeated for each activation.

$$\beta_i = W\beta_{i-1} + \frac{|W\beta_{i-1}|}{|W\beta_{i-1}| + |W\gamma_{i-1}|} \cdot b \tag{13.5}$$

$$\gamma_i = W\gamma_{i-1} + \frac{|W\gamma_{i-1}|}{|W\beta_{i-1}| + |W\gamma_{i-1}|} \cdot b \tag{13.6}$$

When g_i is a max-pooling layer, we identify the indices, or channels, selected by max-pool when run by $g_i(x)$, denoted max_idxs below, and use the decompositions for the corresponding channels.

$$max\_idxs = \underset{idxs}{\text{argmax}} \ [\text{maxpool}(\beta_{i-1} + \gamma_{i-1}; idxs)] \tag{13.7}$$

$$\beta_i = \beta_{i-1}[max\_idxs] \tag{13.8}$$

$$\gamma_i = \gamma_{i-1}[max\_idxs] \tag{13.9}$$

Finally, for the ReLU, we update our importance score β_i by computing the activation of β_{i-1} alone and then update γ_i by subtracting this from the total activation.

$$\beta_i = \text{ReLU}(\beta_{i-1}) \tag{13.10}$$

$$\gamma_i = \text{ReLU}(\beta_{i-1} + \gamma_{i-1}) - \text{ReLU}(\beta_{i-1}) \tag{13.11}$$

For a dropout layer, we simply apply dropout to β_{i-1} and γ_{i-1} individually, or multiplying each by a scalar. Computationally, a CD call is comparable to a forward pass through the network f.

13.3 Agglomerative Contextual Decomposition (ACD)

Given the generalized CD scores introduced above, we now introduce the clustering procedure used to produce ACD interpretations. At a high-level, our method is equivalent to agglomerative

hierarchical clustering, where the CD interaction is used as the joining metric to determine which clusters to join at each step. This procedure builds the hierarchy by starting with individual features and iteratively combining them based on the interaction scores provided by CD. The displayed ACD interpretation is the hierarchy, along with the CD importance score at each node.

More precisely, algorithm 1 describes the exact steps in the clustering procedure. After initializing by computing the CD scores of each feature individually, the algorithm iteratively selects all groups of features within k% of the highest-scoring group (where k is a hyperparameter, fixed at 95 for images and 90 for text) and adds them to the hierarchy.

Each time a new group is added to the hierarchy, a corresponding set of candidate groups is generated by adding individual contiguous features to the original group. For text, the candidate groups correspond to adding one adjacent word onto the current phrase, and for images adding any adjacent pixel onto the current image patch. Candidate groups are ranked according to the CD interaction score, which is the difference between the score of the candidate and original groups.

ACD terminates after an application-specific criterion is met. For sentiment classification, we stop once all words are selected. For images, we stop after some predefined number of iterations and then merge the remaining groups one by one using the same selection criteria described above.

Algorithm 1 Agglomeration algorithm.

```
ACD(Example x, model, hyperparameter k, function CD(x, blob; model))
  # initialize
  tree = Tree()                                                    # tree to output
  scoresQueue = PriorityQueue()                                    # scores, sorted by importance
  for feature in x :
      scoresQueue.push(feature, priority=CD(x, feature; model))

  # iteratively build up tree
  while scoresQueue is not empty :
      selectedGroups = scoresQueue.popTopKPercentile(k)            # pop off top k elements
      tree.add(selectedGroups)                                     # Add top k elements to the tree

      # generate new groups of features based on current groups and add them to the queue
      for selectedGroup in selectedGroups :
          candidateGroups = getCandidateGroups(selectedGroup)
          for candidateGroup  in candidateGroups :
              scoresQueue.add(candidateGroup, priority=CD(x, candidateGroup;model)-
  CD(x,selectedGroup; model))
  return tree
```

Algorithm 1 is not specific to DNNs; it requires only a method to obtain importance scores for groups of input features. Here, we use CD scores to arrive at the ACD algorithm, which makes the method specific to DNNs, but given a feature group scoring function, Algorithm 1 can yield interpretations for any predictive model. CD is a natural score to use for DNNs as it

aggregates saliency at different scales and converges to the final prediction once all the units have been selected.

Chapter 14

Experiments

We now present empirical validation of ACD on both LSTMs trained on SST and CNNs trained on MNIST and ImageNet. First, we introduce the reader to our visualization in Chapter 14.2, and how it can (anecdotally) be used to understand models in settings such as diagnosing incorrect predictions, identifying dataset bias, and identifying representative phrases of differing lengths. We then provide quantitative evidence of the benefits of ACD in Chapter 14.3 through human experiments and demonstrating the stability of ACD to adversarial perturbations.

14.1 Experimental details

We first describe the process for training the models from which we produce interpretations. As the objective of this part is to interpret the predictions of models, rather than increase their predictive accuracy, we use standard best practices to train our models. All models are implemented using PyTorch. For SST, we train a standard binary classification LSTM model[1], which achieves 86.2% accuracy. On MNIST, we use the standard PyTorch example[2], which attains accuracy of 97.7%. On ImageNet, we use a pre-trained VGG-16 DNN architecture [115] which attains top-1 accuracy of 42.8%. When using ACD on ImageNet, for computational reasons, we start the agglomeration process with 14-by-14 superpixels instead of individual pixels. We also smooth the computed image patches by adding pixels surrounded by the patch. The weakened models for the human experiments are constructed from the original models by randomly permuting a small percentage of their weights. For SST/MNIST/ImageNet, 25/25/0.8% of weights are randomized, reducing test accuracy from 85.8/97.7/42.8% to 79.8/79.6/32.3%.

[1] model and training code from https://github.com/clairett/pytorch-sentiment-classification

[2] model and training code from https://github.com/pytorch/examples/tree/master/mnist

14.2 Qualitative experiments

Before providing quantitative evidence of the benefits of ACD, we first introduce the visualization and demonstrate its utility in interpreting a predictive model's behavior. To qualitatively evaluate ACD, in Supplement S3 we show the results of several more examples selected using the same criterion as in our human experiments described below.

Understanding predictive models using ACD

In the following examples, we demonstrate the use of ACD to diagnose incorrect predictions in SST and identify dataset bias in ImageNet. These examples are only a few of the potential uses of ACD.

Text example - diagnosing incorrect predictions In the first example, we show the result of running ACD for our SST LSTM model in Figure 14.1. We can use this ACD visualization to quickly diagnose why the LSTM made an incorrect prediction. In particular, note that the ACD summary of the LSTM correctly identifies two longer phrases and their corresponding sentiment *a great ensemble cast* (positive) and *n't lift this heartfelt enterprise out of the ordinary* (negative). It is only when these two phrases are joined that the LSTM inaccurately predicts a positive sentiment. This suggests that the LSTM has erroneously learned a positive interaction between these two phrases. Prior methods would not be capable of detecting this type of useful information.

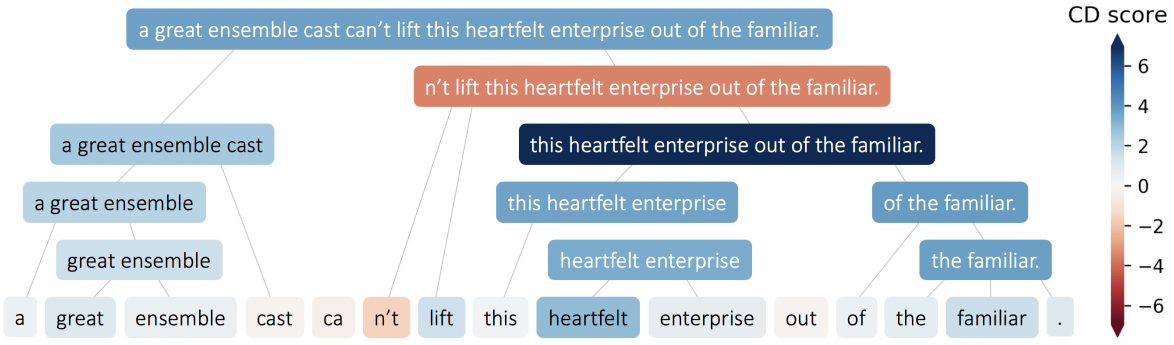


Figure 14.1: ACD interpretation of an LSTM predicting sentiment. Blue is positive sentiment, white is neutral, red is negative. The bottom row displays CD scores for individual words in the sentence. Higher rows display important phrases identified by ACD, along with their CD scores, converging to the model's (incorrect) prediction in the top row. (Best viewed in color)

Vision example - identifying dataset bias Figure 14.2 shows an example using ACD for an ImageNet VGG model. Using ACD, we can see that to predict "puck", the CNN is not just focusing on the puck in the image, but also on the hockey player's skates. Moreover, by comparing the fifth

and sixth plots in the third row, we can see that the network is only able to distinguish between the class "puck" and the other top classes when the orange skate and green puck patches merge into a single orange patch. This suggests that the CNN has learned that skates are a strong corroborating features for pucks. While intuitively reasonable in the context of ImageNet, this may not be desirable behavior if the model were used in other domains.

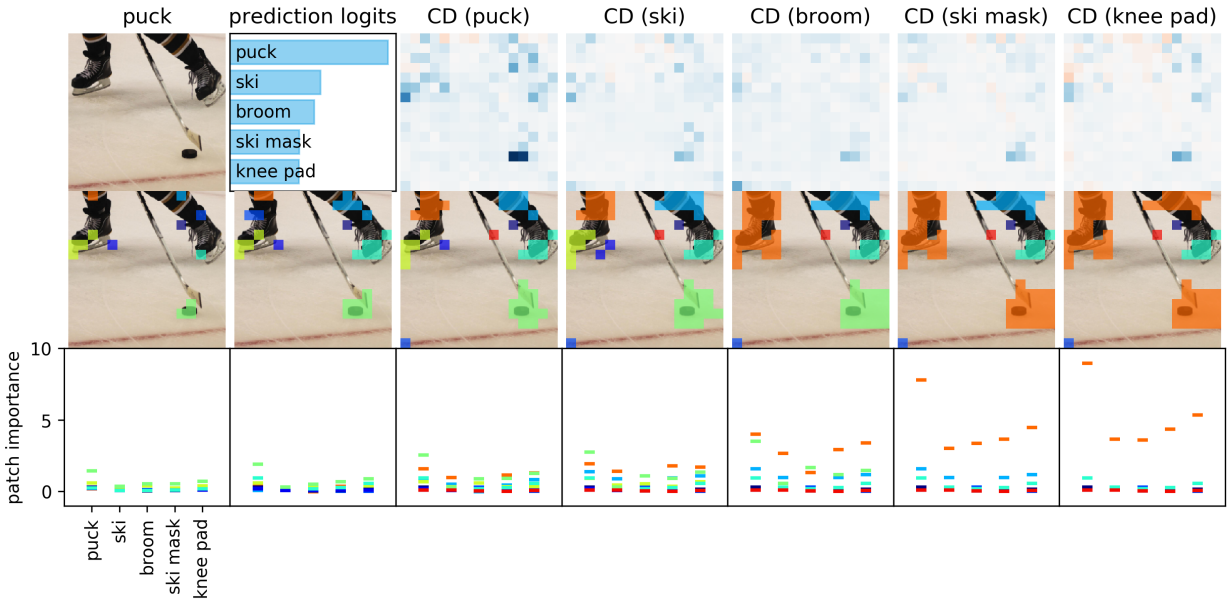


Figure 14.2: ACD interpretation for a VGG network prediction on ImageNet, described in 14.2. ACD shows that the CNN is focusing on skates to predict the class "puck", indicating that the model has captured dataset bias. The top row shows the original image, logits for the five top-predicted classes, and the CD superpixel-level scores for those classes. The second row shows separate image patches ACD has identified as being independently predictive of the class "puck". Starting from the left, each image shows a successive iteration in the agglomeration procedure. The third row shows the CD scores for each of these patches, where patch colors in the second row correspond to line colors in the third row. ACD successfully finds important regions for the target class (such as the puck), and this importance increases as more pixels are selected. Best viewed in color.

Identifying top-scoring phrases

When feasible, a common means of scrutinizing what a model has learned is to inspect its most important features, and interactions. In Table 14.1, we use ACD to show the top-scoring phrases of different lengths for our LSTM trained on SST. These phrases were extracted by running ACD separately on each sample in SST's validation set. The score of each phrase was then computed

Length	Positive	Negative
1	pleasurable, sexy, glorious	nowhere, grotesque, sleep
3	amazing accomplishment., great fun.	bleak and desperate, conspicuously lacks.
5	a pretty amazing accomplishment.	ultimately a pointless endeavour.
8	presents it with an unforgettable visual panache.	my reaction in a word: disappointment.

Table 14.1: Top-scoring phrases of different lengths extracted by ACD on SST's validation set. The positive/negative phrases identified by ACD are all indeed positive/negative.

by averaging over the score it received in each occurrence in a ACD hierarchy. The extracted phrases are clearly reflective of the corresponding sentiment, providing additional evidence that ACD is able to capture meaningful positive and negative phrases. Additional phrases are given in Supplement S2.

14.3 Quantitative experiments

Having introduced our visualization and provided qualitative evidence of its uses, we now provide quantitative evidence of the benefits of ACD.

Human experiments

We now demonstrate through human experiments that ACD allows users to better trust and reason about the accuracy of DNNs. Human subjects consist of eleven graduate students at the author's institution, each of whom has taken a class in machine learning. Each subject was asked to fill out a survey with two types of questions: whether, using ACD, they could identify the more accurate of two models and whether they trusted a models output. In both cases, similar questions were asked on three datasets (SST, MNIST and ImageNet), and ACD was compared against three baselines: CD [85], Integrated Gradients (IG) [122], and occlusion [71, 140]. The exact survey prompts are provided in Supplement S4.

Identifying an accurate model The objective of this section was to determine if subjects could use a small number of interpretations produced by ACD in order to identify the more accurate of two models. For each question in this section, two example predictions were chosen. For each of these two predictions, subjects were given interpretations from two different models (four total), and asked to identify which of the two models had a higher predictive accuracy. Each subject was asked to make this comparison using three different sets of examples for each combination of dataset and interpretation method, for 36 total comparisons. To remove variance due to examples, the same three sets of examples were used across all four interpretation methods.

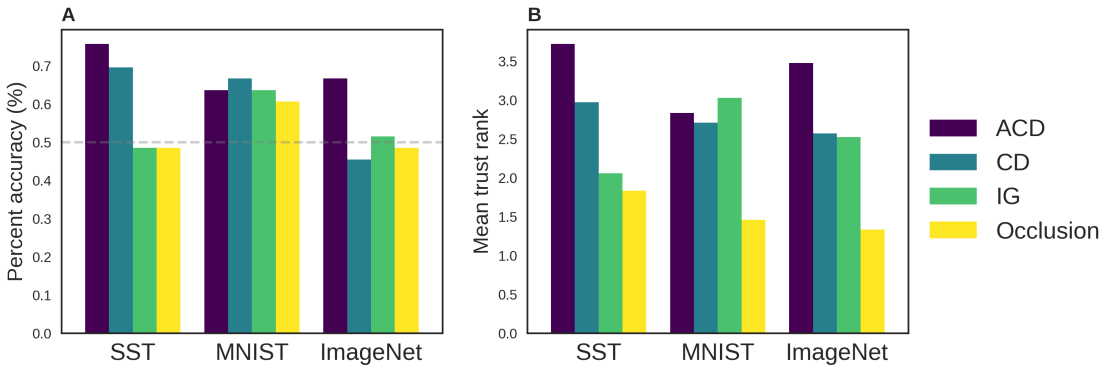


Figure 14.3: Results for human studies. **A.** Binary accuracy for whether a subject correctly selected the more accurate model using different interpretation techniques **B.** Average rank (from 1 to 4) of how much different interpretation techniques helped a subject to trust a model, higher ranks are better.

The predictions shown were chosen to maximize disagreement between models, with SST also being restricted to sentences between five and twenty words, for ease of visualization. To prevent subjects from simply picking the model that predicts more accurately for the given example, for each question a user is shown two examples: one where only the first model predicts correctly and one where only the second model predicts correctly. The two models considered were the accurate models of the previous section and a weakened version of that same model (details given in Chapter 14.1).

Fig 14.3A shows the results of the survey. For SST, humans were better able to identify the strongly predictive model using ACD compared to other baselines, with only ACD and CD outperforming random selection (50%). Based on a one-sided two-sample t-test, the gaps between ACD and IG/Occlusion are significant, but not the gap between ACD and CD. In the simple setting of MNIST, ACD performs similarly to other methods. When applied to ImageNet, a more complex dataset, ACD substantially outperforms prior, non-hierarchical methods, and is the only method to outperform random chance, although the gaps between ACD and other methods are only statistically suggestive (p-values fall between 0.15 and 0.07).

Evaluating trust in a model In this section, the goal is to gauge whether ACD helps a subject to better trust a model's predictions, relative to prior techniques. For each question, subjects were shown interpretations of the same prediction using four different interpretation methods, and were asked to rank the interpretations from one to four based on how much they instilled trust in trust the model. Subjects were asked to do this ranking for three different examples in each dataset, for nine total rankings. The interpretations were produced from the more accurate model from the previous section, and the examples were chosen using the same criteria as the previous section, except they were restricted to examples correctly predicted by the more accurate model.

Fig 14.3B shows the average ranking received by each method/dataset pair. ACD substantially

outperforms other baselines, particularly for ImageNet, achieving an average rank of 3.5 out of 4, where higher ranks are better. As in the prior question, we found that the hierarchy only provided benefits in the more complicated ImageNet setting, with results on MNIST inconclusive. For both SST and ImageNet, the difference in mean ranks between ACD and all other methods is statistically significant (p-value less than 0.005) based on a permutation test, while on MNIST only the difference between ACD and occlusion is significant.

ACD hierarchy is robust to adversarial perturbations

While there has been a considerable amount of work on adversarial attacks, little effort has been devoted to qualitatively understanding this phenomenon. In this section, we provide evidence that, on MNIST, the hierarchical clustering produced by ACD is largely robust to adversarial perturbations. This suggests that ACD's hierarchy captures fundamental features of an image, and is largely immune to the spurious noise favored by adversarial examples.

To measure the robustness of ACD's hierarchy, we first qualitatively compare the interpretations produced by ACD on both an unaltered image and an adversarially perturbed version of that image. Empirically, we found that the extracted hierarchies are often very similar, see Supplement S5. To generalize these observations, we introduce a metric to quantify the similarity between two ACD hierarchies. This metric allows us to make quantitative, dataset-level statements about the stability of ACD feature hierarchies with respect to adversarial inputs. Given an ACD hierarchy, we compute a ranking of the input image's pixels according to the order in which they were added to the hierarchy. To measure the similarity between the ACD hierarchies for original and adversarial images, we compute the correlation between their corresponding rankings. As ACD hierarchies are class-specific, we average the correlations for the original and adversarially altered predictions.

We display the correlations for five different attacks (computed using the Foolbox package [101], examples shown in Supplement S6), each averaged over 100 randomly chosen predictions, in Table 14.2. As ACD is the first local interpretation technique to compute a hierarchy, there is little prior work available for comparison. As a baseline, we use our agglomeration algorithm with occlusion in place of CD. The resulting correlations are substantially lower, indicating that features detected by ACD are more stable to adversarial attacks than comparable methods. These results provide evidence that ACD's hierarchy captures fundamental features of an image, and is largely immune to the spurious noise favored by adversarial examples.

14.4 Conclusion

In this part, we introduce agglomerative contextual decomposition (ACD), a novel hierarchical interpretation algorithm. ACD is the first method to use a hierarchy to interpret individual neural network predictions. Doing so enables ACD to automatically detect and display non-linear contributions to individual DNN predictions, something prior interpretation methods are unable

Attack Type	ACD	Agglomerative Occlusion
Saliency [93]	0.762	0.259
Gradient attack	0.662	0.196
FGSM [45]	0.590	0.131
Boundary [24]	0.684	0.155
DeepFool [82]	0.694	0.202

Table 14.2: Correlation between pixel ranks for different adversarial attacks. ACD achieves consistently high correlation across different attack types, indicating that ACD hierarchies are largely robust to adversarial attacks. Using occlusion in place of CD produces substantially less stable hierarchies.

to do. The benefits of capturing the non-linearities inherent in DNNs are demonstrated through human experiments and examples of diagnosing incorrect predictions and dataset bias. We also demonstrate that ACD's hierarchy is robust to adversarial perturbations in CNNs, implying that it captures fundamental aspects of the input and ignores spurious noise.

Appendix A

Supplementary materials for Part II

We provide an example heat map using the cell decomposition metric for each class in both sentiment analysis datasets, and selected WikiMovie question categories

Dataset	Category	Heat Map
Yelp Polarity	Positive	we went here twice for breakfast . had the bananas foster waffles with fresh whipped cream , they were amazing ! ! perfect seat out side on the terrace
Yelp Polarity	Negative	call me spoiled ...this sushi is gross and the orange chicken , well it was so thin i don 't think it had chicken in it. go somewhere else

Dataset	Category	Heat Map
Stanford Sentiment	Positive	Whether or not you 're enlightened by any of Derrida 's lectures on " the other " and " the self , " Derrida is an undeniably fascinating and playful fellow
Stanford Sentiment	Negative	... begins with promise , but runs aground after being snared in its own tangled plot

Pattern	Question	Heat Map
Movie to Year	What was the release year of another 48 hours?	another 48 hrs is a 1990
Movie to Writer	Which person wrote the movie last of the dogmen?	last of the dogmen is a 1995 western adventure film written and directed by tab murphy
Movie to Actor	Who acted in the movie thunderbolt?	thunderbolt ( ) (" piklik foh ") is a 1995 hong kong action film starring jackie chan
Movie to Director	Who directed bloody bloody bible camp?	bloody bloody bible cam p is a 2012 american horror - comedy /s platter film . the film was directed by vito trabucco
Movie to Genre	What genre is trespass in?	trespass is a 1992 action

Pattern	Question	Heat Map
Movie to Votes	How would people rate the pool?	though filmed in hindi , a language smith didn 't know , the film earned good*
Movie to Rating	How popular was les miserables?	les mis rables is a 1935 american drama film starring fredric march and charles laughton based upon the famous
Movie to Tags	Describe rough magic?	rough magic is a 1995 comedy film directed by clare peploe and starring bridget fonda , russell crowe
Movie to Language	What is the main language in fate?	fate ( ) is a 2001 turkish

Appendix B

Supplementary materials for Part III

B.1 Plots

Plots for dissenting subphrases

We provide here the plots described in Section 12.3.

Plots for high-level compositionality

We provide here the plots referenced in Section 12.4.

Logistic regression versus extracted coefficients scatterplots

We provide here the scatterplots and correlations referenced in section 12.2.

Attribution Method	Stanford Sentiment	Yelp Polarity
Gradient	0.375	0.336
Leave one out [71]	0.510	0.358
Cell decomposition [86]	0.490	0.560
Integrated gradients [122]	0.724	0.471
Contextual decomposition	0.758	0.520

Table B.1: Correlation coefficients between logistic regression coefficients and extracted scores.

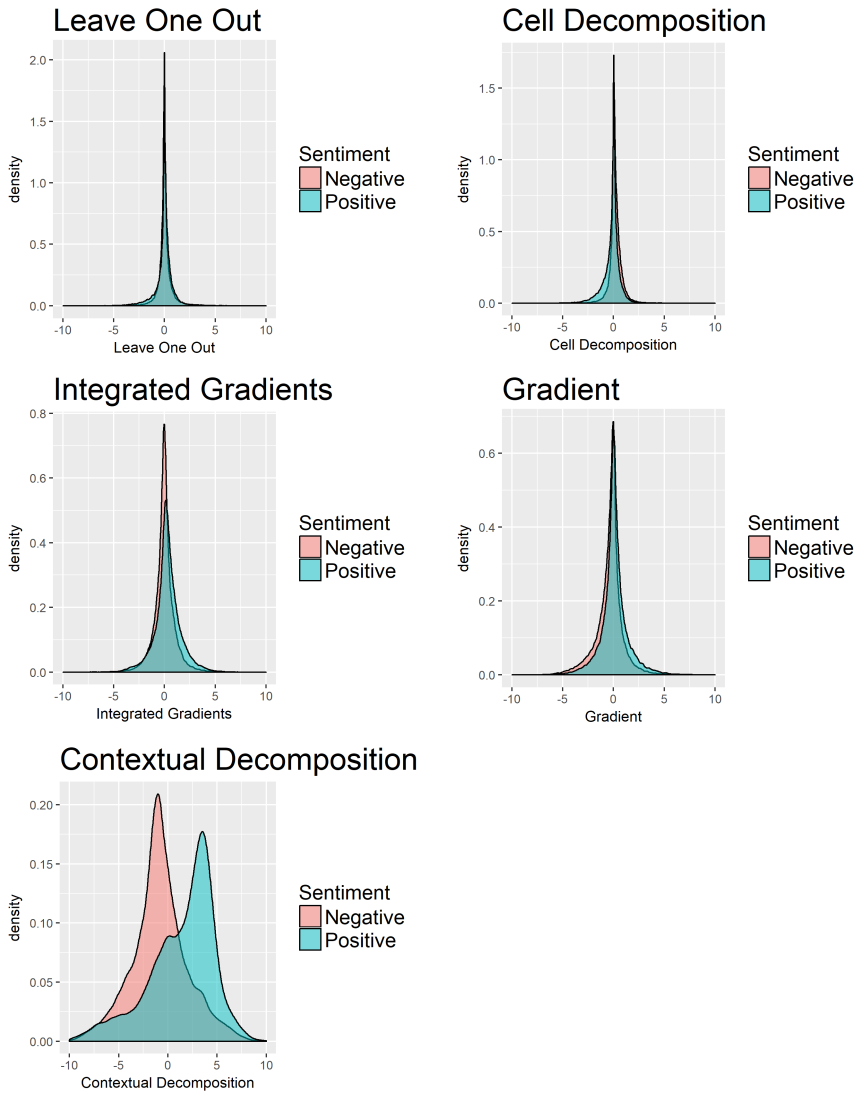


Figure B.1: The distribution of attributions for positive (negative) sub-phrases contained within negative (positive) phrases of length at most five in the Yelp polarity dataset. The positive and negative distributions are nearly identical for all methods except CD, indicating an inability of prior methods to distinguish between positive and negative phrases when occurring in the context of a phrase of the opposite sentiment

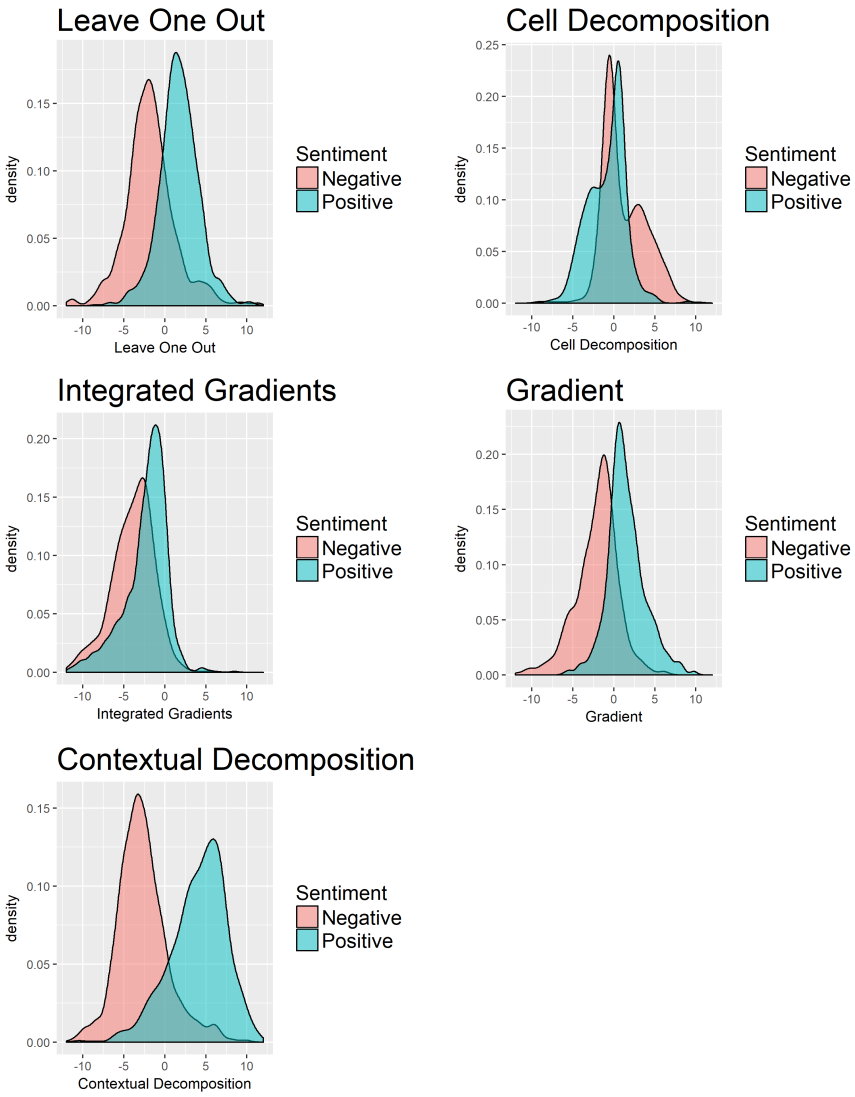


Figure B.2: Distribution of positive and negative phrases, of length between one and two thirds of the full review, in SST. The positive and negative distributions are significantly more separate for CD than other methods, indicating that even at this coarse level of granularity, other methods still struggle.

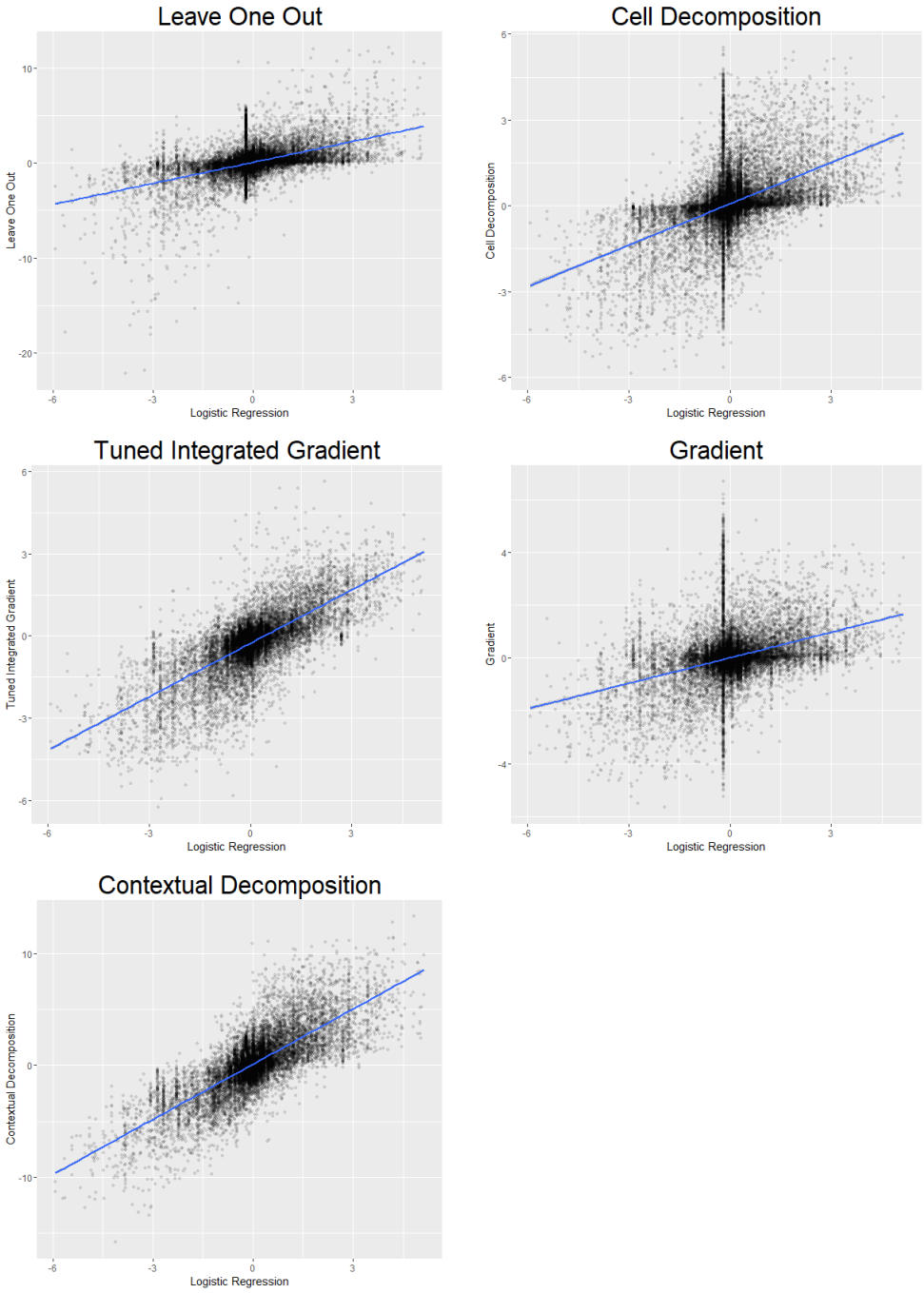


Figure B.3: Logistic regression coefficients versus coefficients extracted from an LSTM on SST. We include a least squares regression line. Stronger linear relationships in the plots correspond to better interpretation techniques.

B.2 General recursion formula

We provide here the general recursion formula referenced in Section 11.3. The two cases that are considered is whether the current time step is during the phrase ($q \leq t \leq r$) or outside of the phrase ($t < q$ or $t > r$).

$$\beta_t^f = [L_\sigma(V_f\beta_{t-1}) + L_\sigma(b_f) + L_\sigma(V_f x_t)1_{q\leq t\leq r}] \odot \beta_{t-1}^c \tag{B.1}$$

$$\gamma_t^f = f_t \odot \gamma_{t-1}^c + [L_\sigma(V_f\gamma_{t-1}) + L_\sigma(V_f x_t)1_{t>q,t<r}] \odot \beta_{t-1}^c \tag{B.2}$$

$$\begin{aligned}\beta_t^u =& L_\sigma(V_i\beta_{t-1}^c) \odot [L_{\tanh}(V_g\beta_{t-1}^c + L_{\tanh}(b_g)] + L_\sigma(b_i) \odot L_{\tanh}(V_g\beta_{t-1}^c) \\ &+ [L_\sigma(W_i x_t) \odot [L_{\tanh}(W_g x_t) + L_{\tanh}(V_g\beta_{t-1}) + L_{\tanh}(b_g)] + L_\sigma(b_i) \odot L_{\tanh}(W_g x_t)]1_{q\leq t\leq r}\end{aligned} \tag{B.3}$$

$$\begin{aligned}\gamma_t^f =& L_\sigma(V_i\gamma_{t-1}) \odot g_t + i_t \odot L_{\tanh}(V_g\gamma_{t-1}) - L_\sigma(V_i\gamma_{t-1}) \odot L_{\tanh}(V_g\gamma_{t-1}) + L_\sigma(b_i) \odot L_{\tanh}(b_g) + \\ &+ [L_\sigma(W_i x_t) \odot [L_{\tanh}(W_g x_t) + L_{\tanh}(V_g\beta_{t-1}) + L_{\tanh}(b_g)] + L_\sigma(b_i) \odot L_{\tanh}(W_g x_t)]1_{t<q,t>r}\end{aligned} \tag{B.4}$$

B.3 List of words used to identify negations

To search for negations, we used the following list of negation words: not, n't, lacks, nobody, nor, nothing, neither, never, none, nowhere, remotely

Appendix C

Supplementary materials for Part IV

S1 CD score comparisons

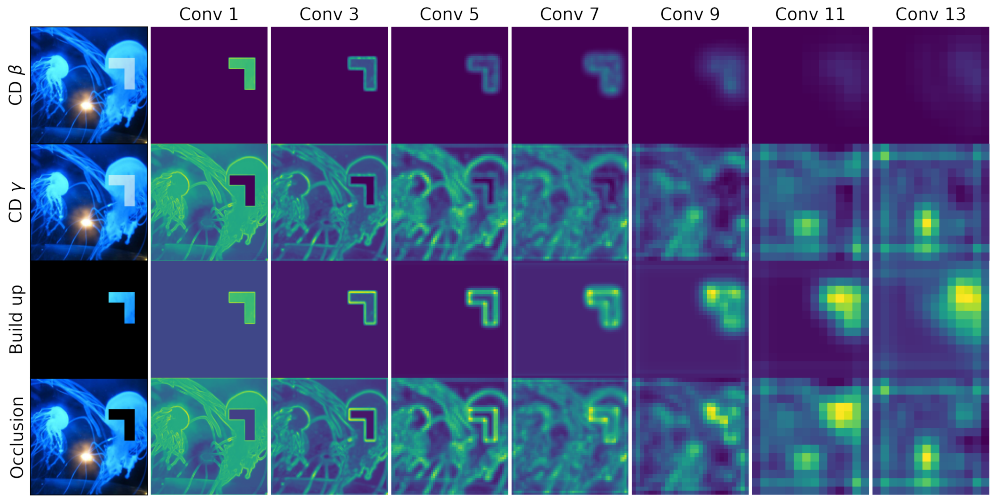


Figure S1: Intuition for CD run on a corner-shaped blob compared to *build-up* and *occlusion*. CD decomposes a DNN's feedforward pass into a part from the blob of interest (top row) and everything else (second row). Left column shows original image with overlaid blob. Other columns show DNN activations summed over the filter dimension. Top and third rows are on same color scale. Second and bottom rows are on same color scale.

Figure S1 gives intuition for CD on the VGG-16 ImageNet model described in Chapter 14. CD keeps track of the contributions of the blob and non-blob throughout the network. This is intuitively similar to the occlusion and build-up methods, shown in the bottom two rows. The build-up method sets everything but the patch of interest to a references value (often zero). These rows compare the CD decomposition to perturbing the input as in the occlusion and build-up methods. They are similar in early layers, but differences become apparent in later layers.

Figure S2 compares the 7x7 superpixel-level scores for four images comparing different meth-

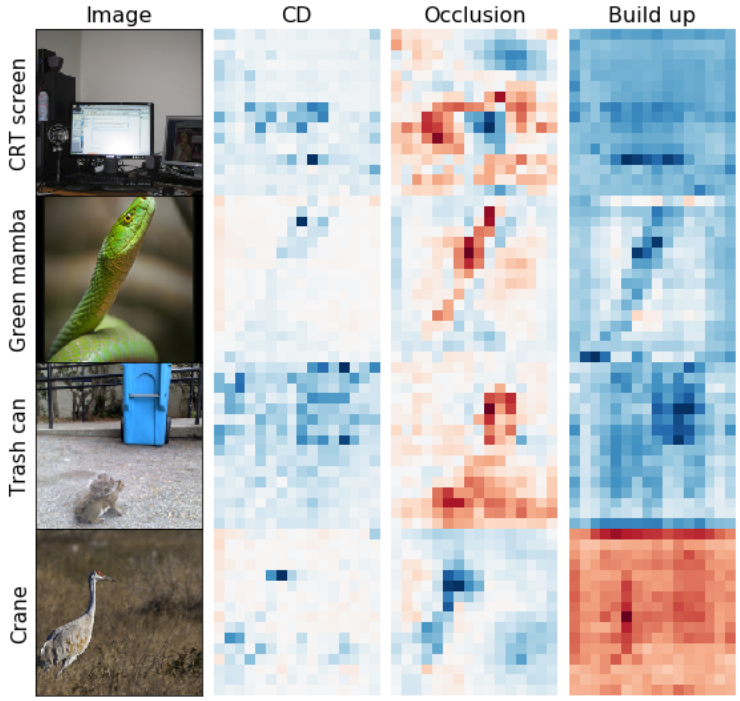


Figure S2: Comparing unit-level CD scores for the correct class to scores from baseline methods. In each case, the model correctly predicts the label, shown on the y axis. Blue is positive, white is neutral, and red is negative. Best viewed in color.

ods for obtaining importance scores. CD scores better find information relevant to predicting the correct class.

S2 Top scoring ACD phrases

Here we provide an extended version of Table S1, containing the top 5 phrases of each length for positive/negative polarities. These were extracted using ACD from an LSTM trained on SST.

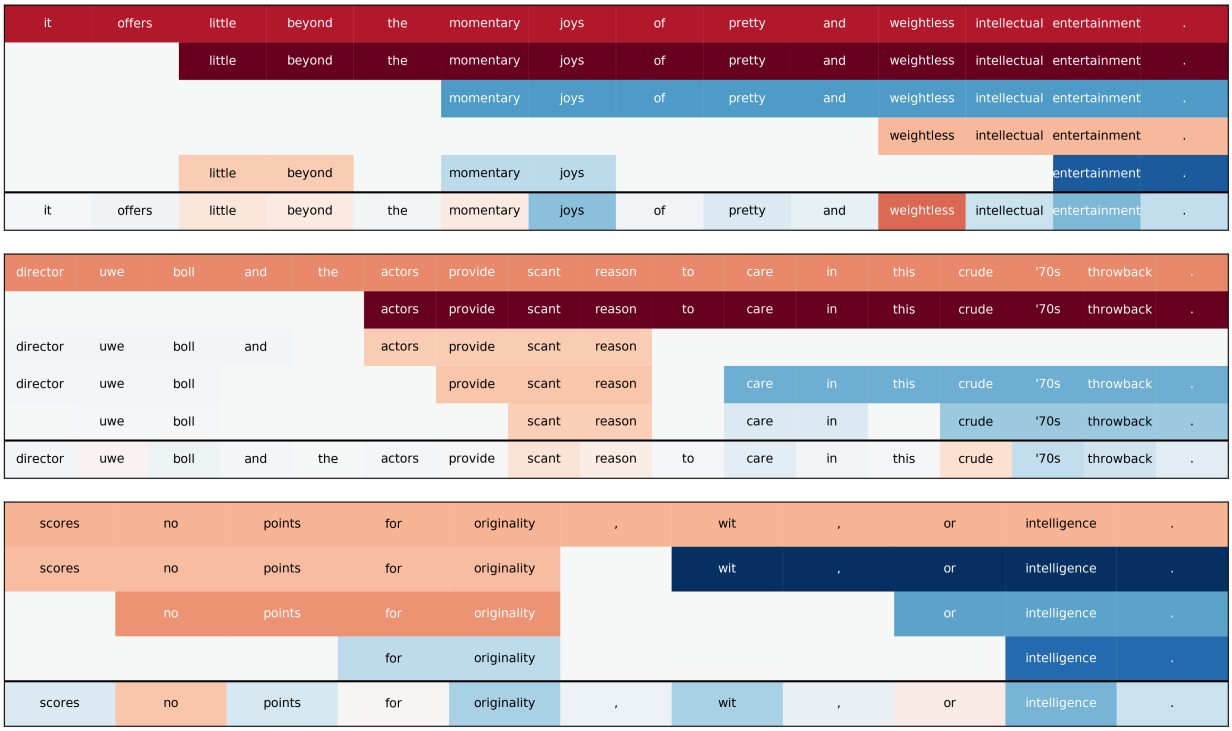
S3 ACD Examples

We provide additional, automatically selected, visualizations produced by ACD. These examples were chosen using the same criteria as the human experiments describes in Chapter 14.3. All examples are best viewed in color.

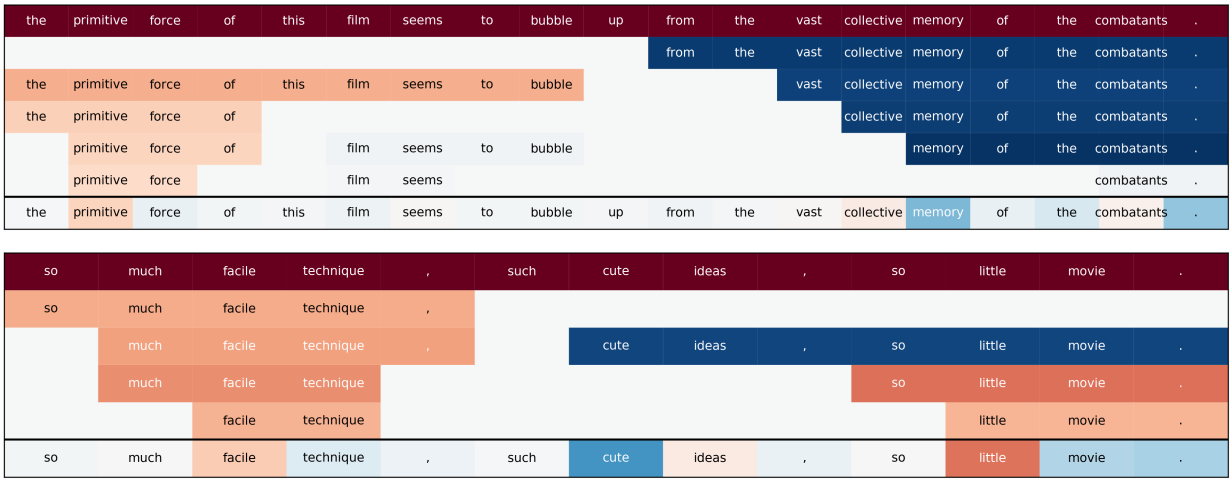
SST top-predicted examples. Here, the model used and figure produced correspond to Figure 14.1.

Length	Positive	Negative
1	'pleasurable', 'sexy', 'glorious', 'delight', 'unforgettable'	'nowhere', 'grotesque', 'sleep', 'mundane', 'clich'
3	'amazing accomplishment .', 'great fun .', 'good fun .', 'language sexy .', 'are magnificent .'	'very bad .', ': disappointment .', 'quite bad .', 'conspicuously lacks .', 'bleak and desperate'
5	'a pretty amazing accomplishment .', 'clearly , great fun .', 'richness of its performances .', 'a delightful coming-of-age story .', 'an unforgettable visual panache .'	'ultimately a pointless endeavor .', 'this is so bad .', 'emotion closer to pity .', 'fat waste of time .', 'sketch gone horribly wrong .'
8	'presents it with an unforgettable visual panache .', 'film is packed with information and impressions .', 'entertains by providing good , lively company .'	'my reaction in a word : disappointment .', "'s slow – very , very slow .", 'a dull , ridiculous attempt at heart-tugging .'
12	'in delicious colors , and the costumes and sets are grand .', 'part stevens glides through on some solid performances and witty dialogue .', 'mamet enthusiast and for anyone who appreciates intelligent , stylish moviemaking .'	"actors provide scant reason to care in this crude '70s throwback .", 'more often just feels generic , derivative and done to death .', 'its storyline with glitches casual fans could correct in their sleep .'
15	'serry shows a remarkable gift for storytelling with this moving , effective little film .', ', lathan and diggs are charming and have chemistry both as friends and lovers .'	'level that one enjoys a bad slasher flick , primarily because it is dull .', 'technicality that strains credulity and leaves the viewer haunted by the waste of potential .'

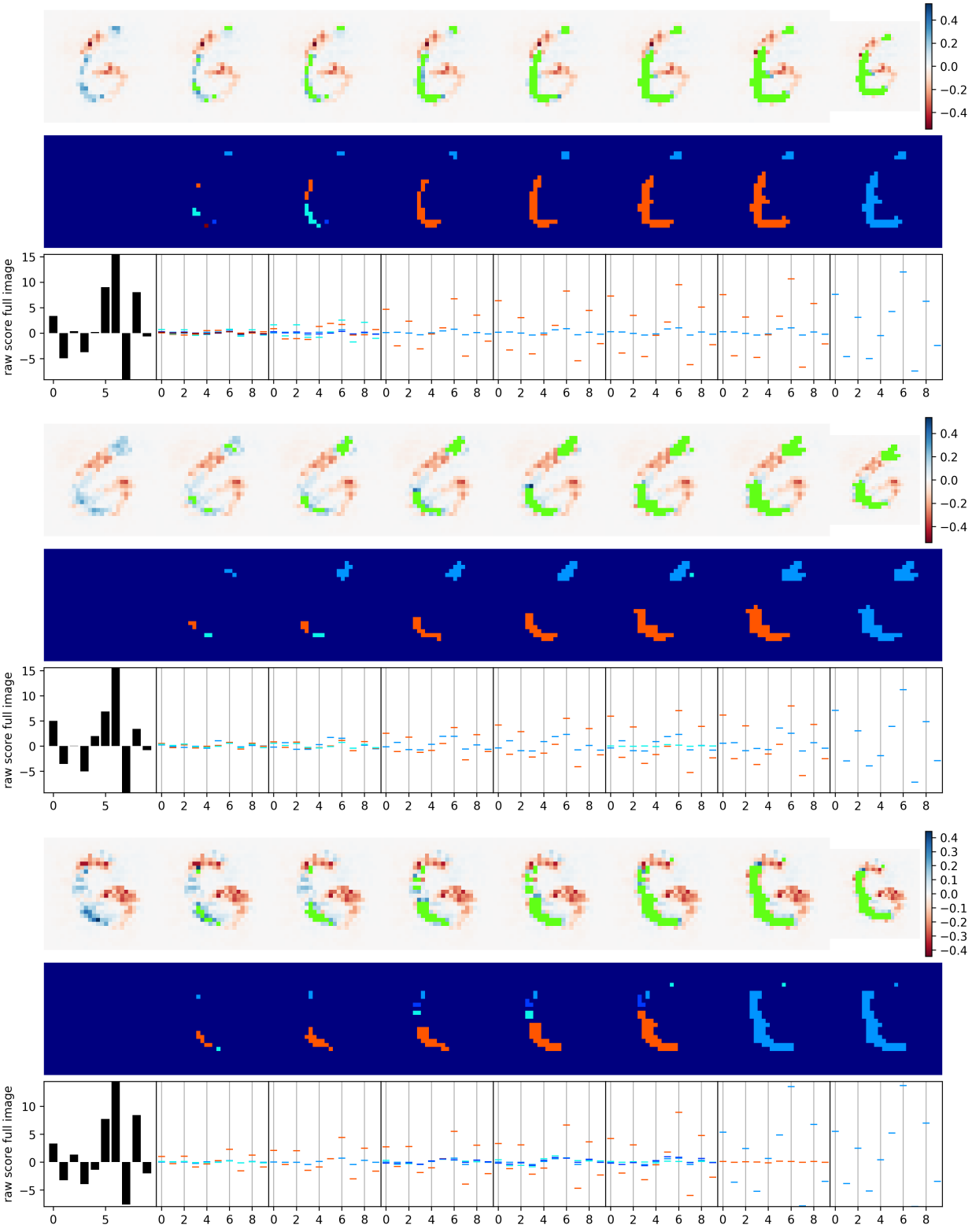
Table S1: Top-scoring phrases of different lengths extracted by ACD on SST's validation set. The positive/negative phrases identified by ACD are all indeed positive/negative



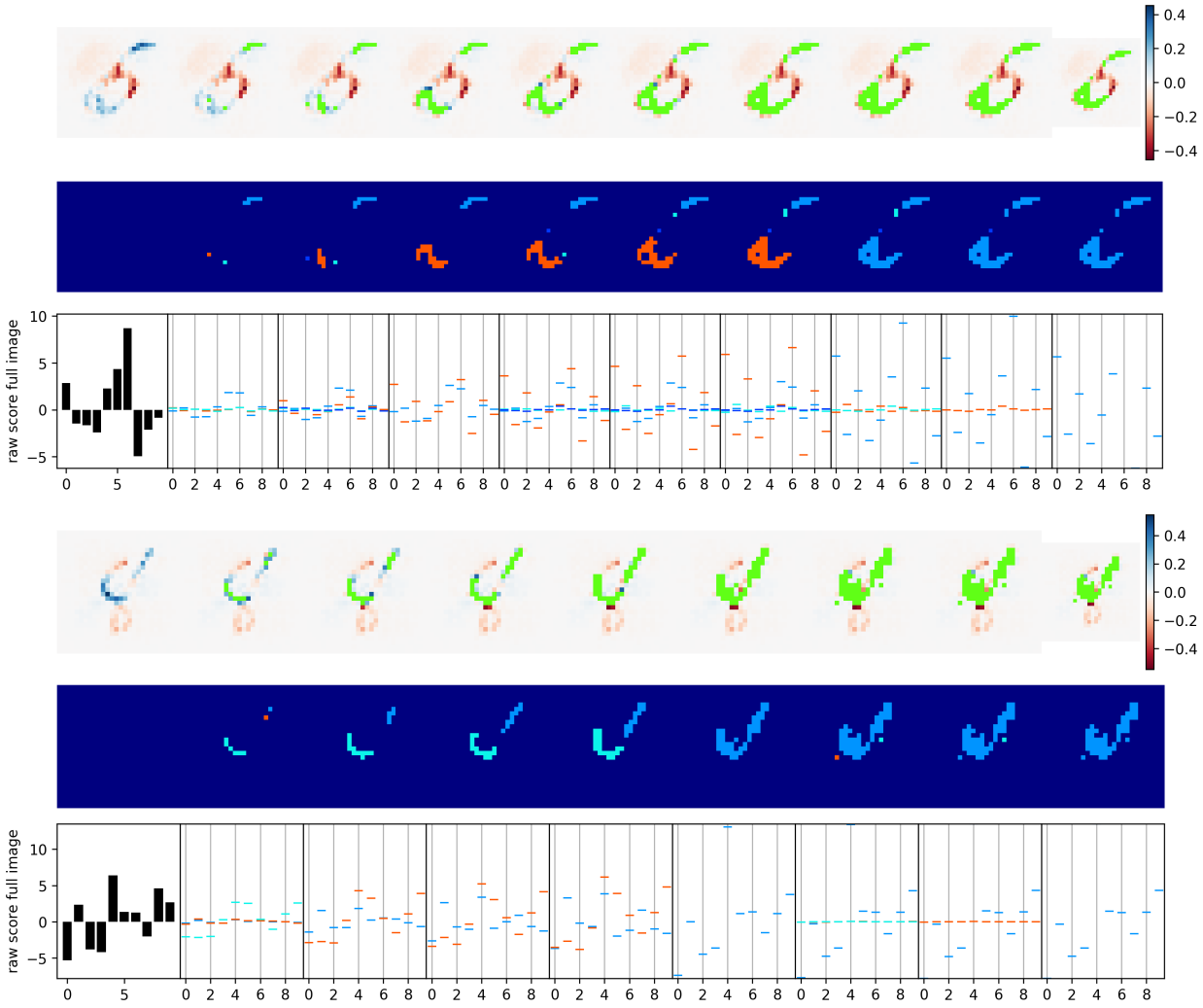
SST lowest-predicted examples. Here, the model used and figure produced correspond to Figure 14.1.



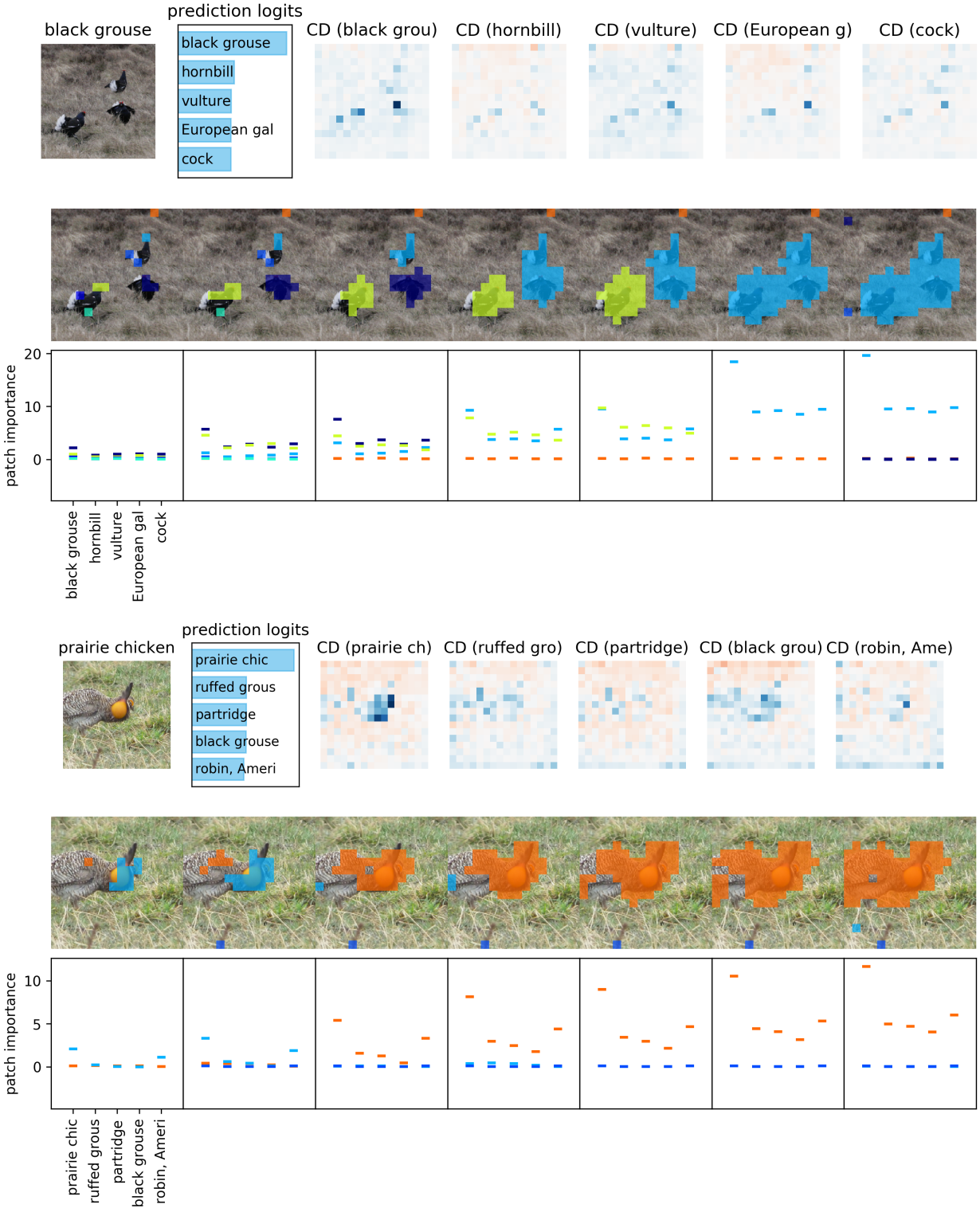
MNIST top-predicted examples. Here, the model used is the same as in Chapter 14.3 and the interpretation of the figure produced is the same as in Figure 14.2.

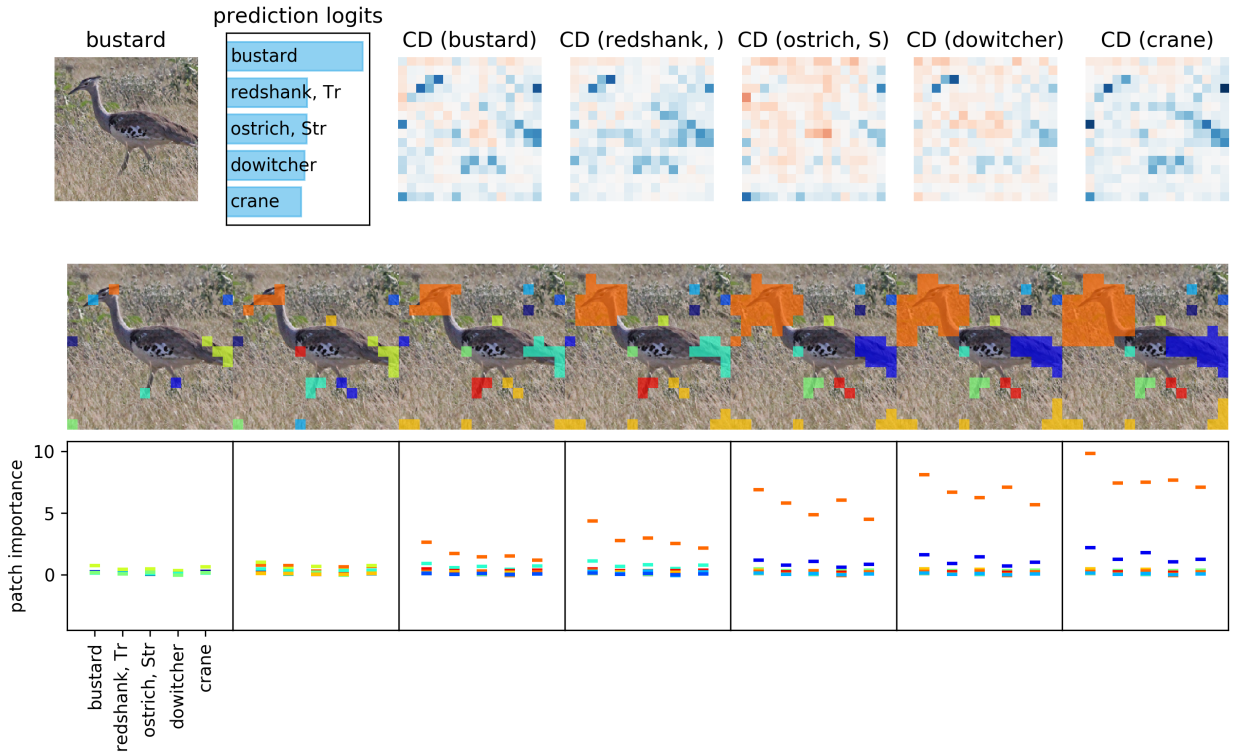

raw score full image
raw score full image
raw score full image

MNIST lowest-predicted examples. Here, the model used is the same as in Chapter 14.3 and the interpretation of the figure produced is the same as in Figure 14.2.

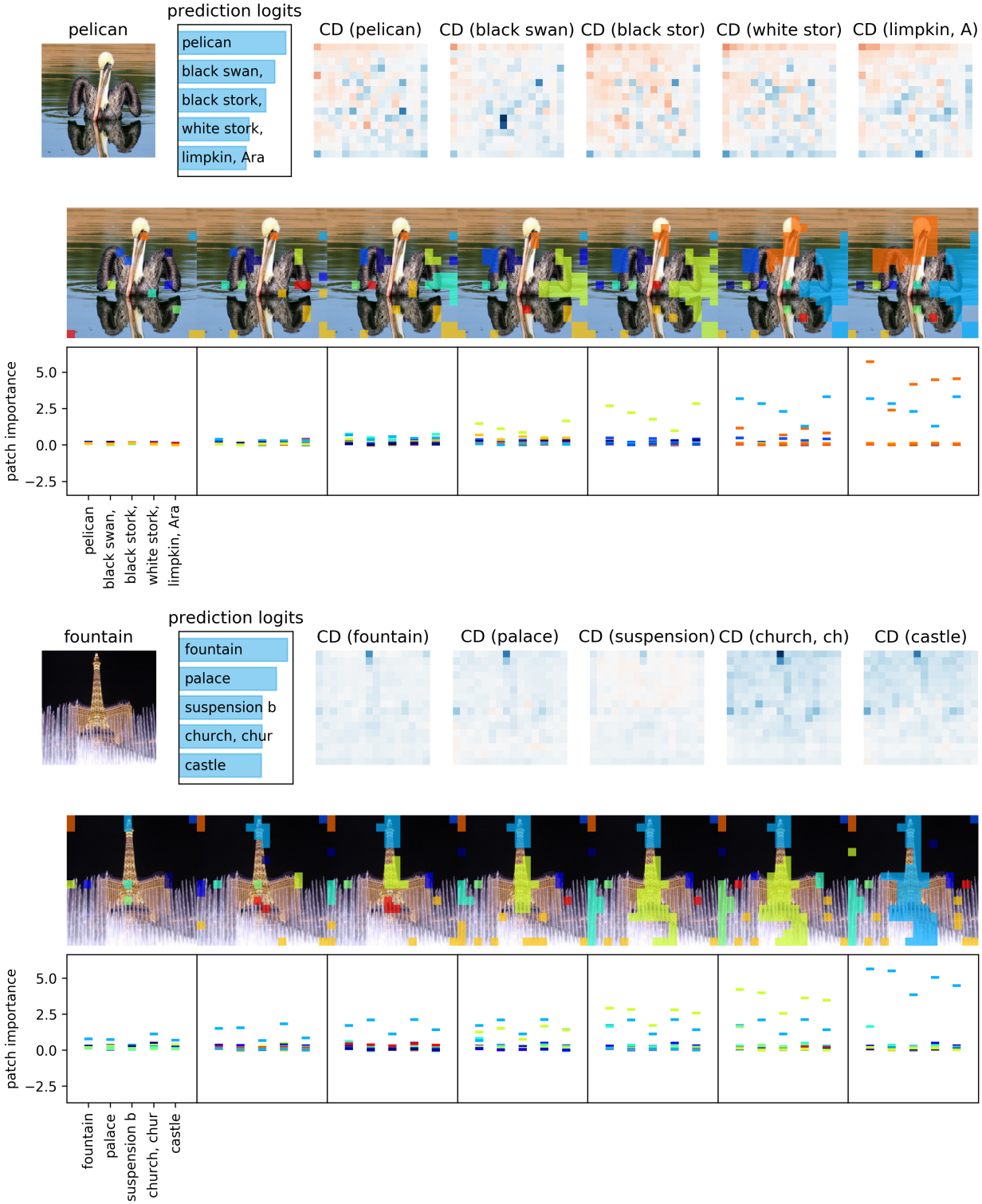


Imagenet top-predicted examples. Here, the model used and figure produced correspond to that in Figure 14.2.


black grouse
prediction logits
black grouse
hornbill
vulture
European gal
cock
CD (black grou)
CD (hornbill)
CD (vulture)
CD (European g)
CD (cock)
patch importance
20
10
0
black grouse
hornbill
vulture
European gal
cock
prairie chicken
prediction logits
prairie chic
ruffed grous
partridge
black grouse
robin, Ameri
CD (prairie ch)
CD (ruffed gro)
CD (partridge)
CD (black grou)
CD (robin, Ame)
patch importance
10
5
0
prairie chic
ruffed grous
partridge
black grouse
robin, Ameri



Imagenet lowest-predicted examples. Here, the model used and figure produced correspond to that in Figure 14.2.


pelican
prediction logits
pelican
black swan,
black stork,
white stork,
limpkin, Ara
CD (pelican)
CD (black swan)
CD (black stor)
CD (white stor)
CD (limpkin, A)
patch importance
5.0
2.5
0.0
−2.5
pelican
black swan,
black stork,
white stork,
limpkin, Ara
fountain
prediction logits
fountain
palace
suspension b
church, chur
castle
CD (fountain)
CD (palace)
CD (suspension)
CD (church, ch)
CD (castle)
patch importance
5.0
2.5
0.0
−2.5
fountain
palace
suspension b
church, chur
castle

S4 Human experiments experimental setup

Order of questions is randomized for each subject. Below are the instructions and questions given to the user (for brevity, the actual visualizations are omitted, but are similar to the visualizations shown in Supplement S3).

> This survey aims to compare different interpretation techniques. In what follows, blue is positive, white is neutral, and red is negative.

Sentiment classification

Choosing the better model

In this section, the task is to compare two models that classify movie reviews as either positive (good movie) or negative (bad movie). One model has better predictive accuracy than the other.

In what follows, you will see visualizations of what both models have learned. These visualizations use different methods of identifying contributions to the final prediction of either individual words or groups of them. For each model, we show visualizations of two different examples.

In these visualizations, the color shows what the model thinks for individual words / groups of words. Blue is positive sentiment (e.g. ”great”, ”fantastic”) and red is negative sentiment (e.g. ”terrible”, ”miserable”).

Using these visualizations, please write A or B to select which model you think has higher predictive accuracy.

Gauging trust

Now, we show results only from the good model. Your task is to compare different visualizations. For the following predictions, please select which visualization method leads you to trust the model the most.

Put a number next to each of the following letters ranking them in the order of how much they make you trust the model (1-4, 1 is the most trustworthy).

MNIST

Choosing the better model

Now we will perform a similar challenge for vision. Your task is to compare two models that classify images into classes, in this case digits from 0-9. One model has higher predictive accuracy than the other.

In what follows, you will see visualizations of what both models have learned. These visualizations use different methods of identifying contributions to the final prediction of either individual pixels or groups of them. Using these visualizations, please select the model you think has higher accuracy.

For each prediction, the top row contains the raw image followed by five heat maps, and the title shows the predicted class. Each heatmap corresponds to a different class, with blue pixels indicating a pixel is a positive signal for that class, and red pixels indicating a negative signal. **The first heatmap title shows the predicted class of the network - this is wrong half the time. In some cases, each visualization has an extra row, which shows groups of pixels**, at multiple levels of granularity, that contribute to the predicted class.

Using these visualizations, please select which model you think has higher predictive accuracy, A or B.

Gauging trust

Now, we show results only from the good model. Your task is to compare different visualizations. For the following predictions, please select which visualization method leads you to trust the model the most.

Put a number next to each of the following letters ranking them in the order of how much they make you trust the model (1-4, 1 is the most trustworthy).

Choosing the more accurate model

Now we will perform a similar challenge for vision. Your task is to compare two models that classify images into classes (ex. balloon, bee, pomegranate). One model is better than the other in terms of predictive accuracy.

In what follows, you will see visualizations of what both models have learned. These visualizations use different methods of identifying contributions to the final prediction of either individual pixels or groups of them.

For each prediction, the top row contains the raw image followed by five heat maps, and the title shows the predicted class. Each heatmap corresponds to a different class, with blue pixels indicating a pixel is a positive signal for that class, and red pixels indicating a negative signal. **The first heatmap title shows the predicted class of the network - this is wrong half the time. In some cases, each visualization has an extra row, which shows groups of pixels**, at multiple levels of granularity, that contribute to the predicted class.

Using these visualizations, please select which model you think has higher predictive accuracy, A or B.

Gauging trust

Now, we show results only from the more accurate model. Your task is to compare different visualizations. For the following predictions, please select which visualization method leads you to trust the model's decision the most.

Put a number next to each of the following letters ranking them in the order of how much they make you trust the model (1-4, 1 is the most trustworthy).

S5 ACD on adversarial examples

The hierarchies constructed by ACD to explain a prediction of 0 are substantially similar for both the original image and an adversarially perturbed image predicted to be a 6.

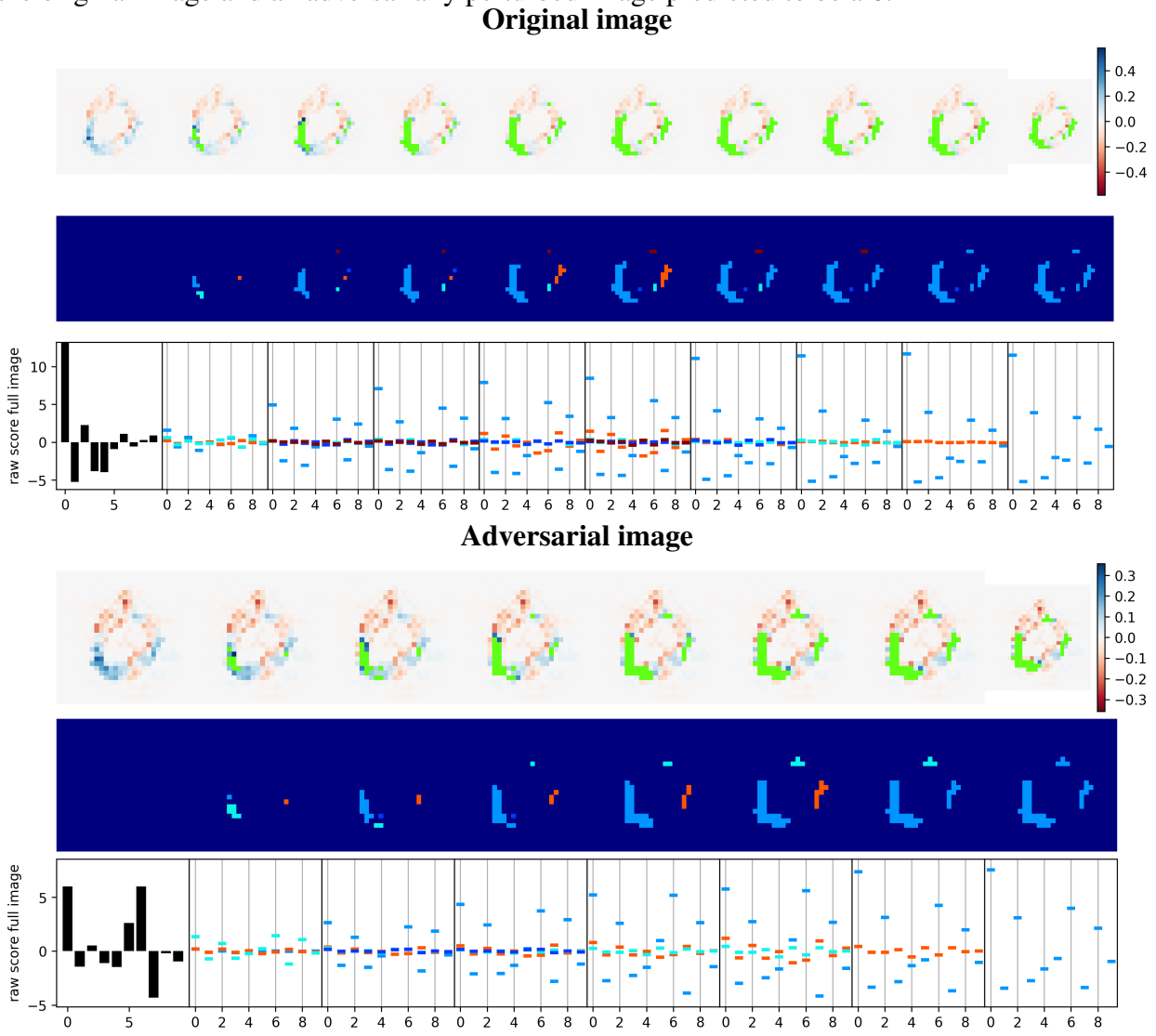


Figure S3: Example of ACD run on an image of class 0 before and after an adversarial perturbation (a DeepFool attack). Best viewed in color.

S6 Adversarial attack examples

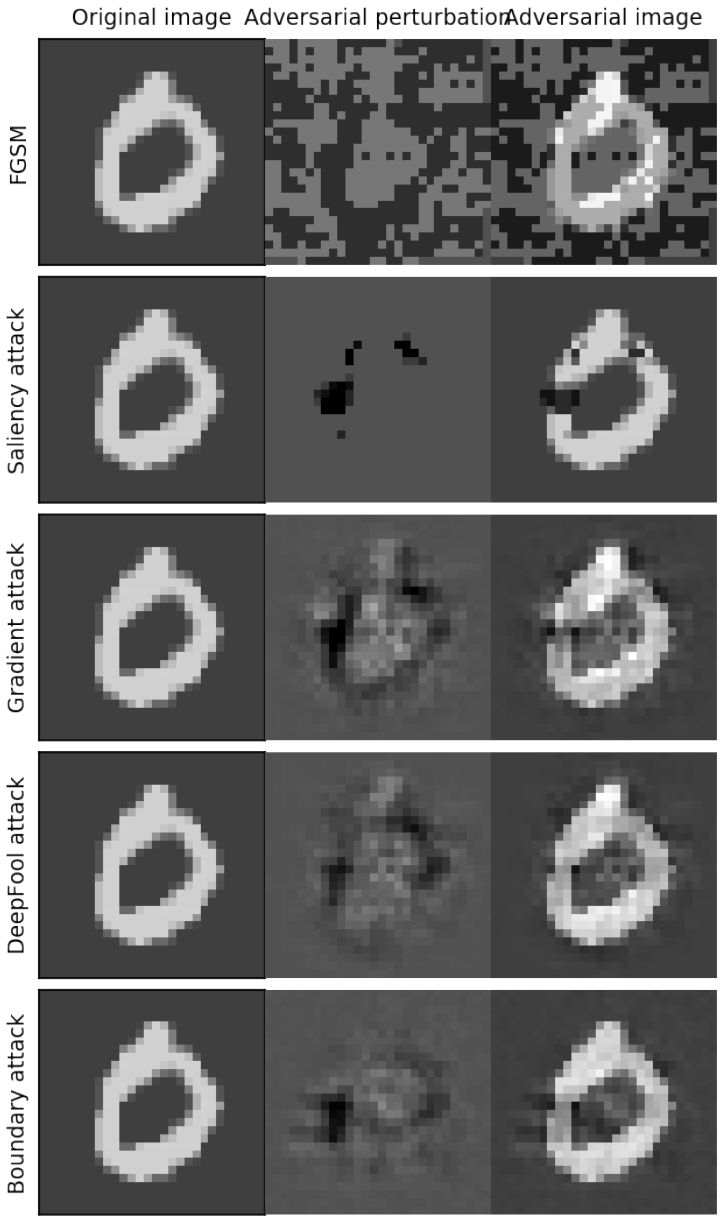


Figure S4: Examples of adversarial attacks for one image. Original image (left column) is correctly predicted as class 0. After each adversarial perturbation (middle column), the predicted class for the adversarial image (right column) is now altered.

S7 Generalizing CD to CNNs

Figure S5 qualitatively shows the change in behavior as the result of two modifications made to the naive extension of CD to CNNs, which was independently developed by [44]. During development of our general CD, two changes were made. First, we partitioned the bias between γ_i and β_i, as described in Equation 13.5. As can be seen in the second column, this qualitatively reduces the noise in the heat maps. Next, we replace the ReLU Shapely decomposition by the decomposition provided in Equation 13.10. In the third column, you can see that this effectively prevents the CD scores from becoming unrealistically large in areas that should not be influencing the model's decision. When these two approaches are combined in the fourth column, they provide qualitatively sensible heatmaps with reasonably valued CD scores. When applied to the smaller models used on SST and MNIST, these changes don't have large effects on the interpretations.

Figure S5: Comparing unit-level CD scores to CD scores from the naive extension of CD to CNNs, independently developed by [44]. Labels under the bottom row signify the minimum and maximum scores from each column. Altering the bias partition and ReLU decomposition qualitatively improves scores (e.g. see scores in bottom row corresponding to the location of the crane), and avoids extremely large magnitudes (see values under left two columns). Blue is positive, white is neutral, and red is negative. In each case, scores are for the correct class, which the model predicts correctly (shown on the y axis).

Bibliography

[1] Reza Abbasi-Asl and Bin Yu. "Structural Compression of Convolutional Neural Networks Based on Greedy Filter Pruning". In: *arXiv preprint arXiv:1705.07356* (2017).

[2] Reza Abbasi-Asl et al. "The DeepTune framework for modeling and characterizing neurons in visual cortex area V4". In: *bioRxiv* (2018), p. 465534.

[3] Hirotugu Akaike. "Factor analysis and AIC". In: *Selected Papers of Hirotugu Akaike*. Springer, 1987, pp. 371–386.

[4] Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. "Automatic Text Scoring Using Neural Networks". In: *Assocation for Computational Lingustics*. 2016.

[5] Andre Altmann et al. "Permutation importance: a corrected feature importance measure". In: *Bioinformatics* 26.10 (2010), pp. 1340–1347.

[6] Dhammika Amaratunga, Javier Cabrera, and Yung-Seop Lee. "Enriched random forests". In: *Bioinformatics* 24.18 (2008), pp. 2010–2014.

[7] Marco Ancona et al. "Towards better understanding of gradient-based attribution methods for Deep Neural Networks". In: *6th International Conference on Learning Representations (ICLR 2018)*. 2018.

[8] Jacob Andreas et al. "Neural module networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 39–48.

[9] Christof Angermueller et al. "Deep learning for computational biology". In: *Molecular systems biology* 12.7 (2016), p. 878.

[10] PETER S. arcidiacono. "Exhibit A: EXPERT REPORT OF PETER S. ARCIDIACONO". In: *http://samv91khoyt2i553a2t1s05i-wpengine.netdna-ssl.com/wp-content/uploads/2018/06/Doc-415-1-Arcidiacono-Expert-Report.pdf* (2018).

[11] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[12] David Baehrens et al. "How to explain individual classification decisions". In: *Journal of Machine Learning Research* 11.Jun (2010), pp. 1803–1831.

[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[14] Trapit Bansal, David Belanger, and Andrew McCallum. "Ask the GRU: Multi-Task Learning for Deep Text Recommendations". In: *ACM international conference on Recommender Systems (RecSys)*. 2016.

[15] RL Barter and B Yu. "Superheat: Supervised heatmaps for visualizing complex data". In: *arXiv preprint arXiv:1512.01524* (2015).

[16] Sumanta Basu et al. "iterative Random Forests to discover predictive and stable high-order interactions". In: *Proceedings of the National Academy of Sciences* (2018), p. 201711236.

[17] Anthony J Bell and Terrence J Sejnowski. "An information-maximization approach to blind separation and blind deconvolution". In: *Neural computation* 7.6 (1995), pp. 1129–1159.

[18] Antoine Bordes et al. "Large-scale simple question answering with memory networks". In: *arXiv preprint arXiv:1506.02075* (2015).

[19] George EP Box. "Science and statistics". In: *Journal of the American Statistical Association* 71.356 (1976), pp. 791–799.

[20] Danah Boyd and Kate Crawford. "Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon". In: *Information, communication & society* 15.5 (2012), pp. 662–679.

[21] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[22] Leo Breiman et al. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3 (2001), pp. 199–231.

[23] Leo Breiman et al. "Classification and regression trees". In: (1984).

[24] Wieland Brendel, Jonas Rauber, and Matthias Bethge. "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models". In: *arXiv preprint arXiv:1712.04248* (2017).

[25] Tim Brennan and William L Oliver. "The emergence of machine learning techniques in criminology". In: *Criminology & Public Policy* 12.3 (2013), pp. 551–562.

[26] Kenneth P Burnham and David R Anderson. "Multimodel inference: understanding AIC and BIC in model selection". In: *Sociological methods & research* 33.2 (2004), pp. 261–304.

[27] David Card. "Exhibit 33: Report of David Card". In: *https://projects.iq.harvard.edu/files/diverse-education/files/legal_-_card_report_revised_filing.pdf* (2018).

[28] Rich Caruana et al. "Case-based explanation of non-case-based learning methods." In: *Proceedings of the AMIA Symposium*. American Medical Informatics Association. 1999, p. 212.

[29] Rich Caruana et al. "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*. ACM. 2015, pp. 1721–1730.

[30] Supriyo Chakraborty et al. “Interpretability of deep learning models: a survey of results”. In: *Interpretability of deep learning models: a survey of results*. 2017.

[31] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[32] Piotr Dabkowski and Yarin Gal. “Real Time Image Saliency for Black Box Classifiers”. In: *arXiv preprint arXiv:1705.07857* (2017).

[33] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 886–893.

[34] 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*. IEEE. 2016, pp. 598–617.

[35] Misha Denil, Alban Demiraj, and Nando de Freitas. “Extraction of Salient Sentences from Labelled Documents”. In: *arXiv preprint: https://arxiv.org/abs/1412.6815*. 2015.

[36] Finale Doshi-Velez and Been Kim. “A roadmap for a rigorous science of interpretability”. In: *arXiv preprint arXiv:1702.08608* (2017).

[37] Cynthia Dwork et al. “Fairness through awareness”. In: *Proceedings of the 3rd innovations in theoretical computer science conference*. ACM. 2012, pp. 214–226.

[38] Jeffrey L Elman. “Finding structure in time”. In: *Cognitive science* 14.2 (1990), pp. 179–211.

[39] Ruth C Fong and Andrea Vedaldi. “Interpretable explanations of black boxes by meaningful perturbation”. In: *arXiv preprint arXiv:1704.03296* (2017).

[40] David A Freedman. “Statistical models and shoe leather”. In: *Sociological methodology* (1991), pp. 291–313.

[41] Jerome H Friedman, Bogdan E Popescu, et al. “Predictive learning via rule ensembles”. In: *The Annals of Applied Statistics* 2.3 (2008), pp. 916–954.

[42] Nicholas Frosst and Geoffrey Hinton. “Distilling a Neural Network Into a Soft Decision Tree”. In: *arXiv preprint arXiv:1711.09784* (2017).

[43] Leilani H Gilpin et al. “Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning”. In: *arXiv preprint arXiv:1806.00069* (2018).

[44] Fréderic Godin et al. “Explaining Character-Aware Neural Networks for Word-Level Prediction: Do They Discover Linguistic Rules?” In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 3275–3284.

[45] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).

[46] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a” right to explanation””. In: *arXiv preprint arXiv:1606.08813* (2016).

[47] Riccardo Guidotti et al. “A Survey Of Methods For Explaining Black Box Models”. In: *arXiv preprint arXiv:1802.01933* (2018).

[48] Frank R Hampel et al. *Robust statistics: the approach based on influence functions*. Vol. 196. John Wiley & Sons, 2011.

[49] Moritz Hardt, Eric Price, Nati Srebro, et al. “Equality of opportunity in supervised learning”. In: *Advances in neural information processing systems*. 2016, pp. 3315–3323.

[50] Gilbert H Harman. “The inference to the best explanation”. In: *The philosophical review* 74.1 (1965), pp. 88–95.

[51] Trevor Hastie and Robert Tibshirani. “Generalized Additive Models”. In: *Statistical Science* 1.3 (1986), pp. 297–318.

[52] Daniel Hewlett et al. “WikiReading: A Novel Large-scale Language Understanding Task over Wikipedia”. In: *Association for Computational Linguistics*. 2016.

[53] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[54] Harold Hotelling. “Relations between two sets of variates”. In: *Biometrika* 28.3/4 (1936), pp. 321–377.

[55] Cheng-Lung Huang, Mu-Chen Chen, and Chieh-Jen Wang. “Credit scoring with a data mining approach based on support vector machines”. In: *Expert systems with applications* 33.4 (2007), pp. 847–856.

[56] Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

[57] IT Jolliffe. “Principal component analysis”. In: (1986).

[58] Rafal Jozefowicz et al. “Exploring the limits of language modeling”. In: *arXiv preprint arXiv:1602.02410* (2016).

[59] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. “Visualizing and understanding recurrent networks”. In: *arXiv preprint arXiv:1506.02078* (2015).

[60] Frank C Keil. “Explanation and understanding”. In: *Annu. Rev. Psychol.* 57 (2006), pp. 227–254.

[61] Jinkyu Kim and John F Canny. “Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention.” In: *ICCV*. 2017, pp. 2961–2969.

[62] Yoon Kim. “Convolutional neural networks for sentence classification”. In: *arXiv preprint arXiv:1408.5882* (2014).

[63] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[64] Thomas Kluyver et al. “Jupyter Notebooks-a publishing format for reproducible computational workflows.” In: *ELPUB*. 2016, pp. 87–90.

[65] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions". In: *arXiv preprint arXiv:1703.04730* (2017).

[66] Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[67] Karl Kumbier et al. "Refining interaction search through signed iterative Random Forests". In: *arXiv preprint arXiv:1810.07287* (2018).

[68] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[69] Tao Lei, Regina Barzilay, and Tommi Jaakkola. "Rationalizing neural predictions". In: *arXiv preprint arXiv:1606.04155* (2016).

[70] Benjamin Letham et al. "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model". In: *The Annals of Applied Statistics* 9.3 (2015), pp. 1350–1371.

[71] Jiwei Li, Will Monroe, and Dan Jurafsky. "Understanding neural networks through representation erasure". In: *arXiv preprint arXiv:1612.08220* (2016).

[72] Peng Li et al. "Dataset and Neural Recurrent Sequence Labeling Model for Open-Domain Factoid Question Answering". In: *arXiv*. 2016.

[73] Chinghway Lim and Bin Yu. "Estimation stability with cross-validation (ESCV)". In: *Journal of Computational and Graphical Statistics* 25.2 (2016), pp. 464–492.

[74] Zachary C Lipton. "The mythos of model interpretability". In: *arXiv preprint arXiv:1606.03490* (2016).

[75] Geert Litjens et al. "A survey on deep learning in medical image analysis". In: *Medical image analysis* 42 (2017), pp. 60–88.

[76] Tania Lombrozo. "The structure and function of explanations". In: *Trends in cognitive sciences* 10.10 (2006), pp. 464–470.

[77] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. "Consistent Individualized Feature Attribution for Tree Ensembles". In: *arXiv preprint arXiv:1802.03888* (2018).

[78] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4768–4777.

[79] Wes McKinney et al. "Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, pp. 51–56.

[80] Gábor Melis, Chris Dyer, and Phil Blunsom. "On the state of the art of evaluation in neural language models". In: *arXiv preprint arXiv:1707.05589* (2017).

[81] Alexander Miller et al. "Key-Value Memory Networks for Directly Reading Documents". In: *Empirical Methods for Natural Language Processing*. 2016.

[82] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. EPFL-CONF-218057. 2016.

[83] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. "Deepdream-a code example for visualizing neural networks". In: *Google Research* 2 (2015), p. 5.

[84] Michael C Mozer. "A focused backpropagation algorithm for temporal". In: *Backpropagation: Theory, architectures, and applications* 137 (1995).

[85] W James Murdoch, Peter J Liu, and Bin Yu. "Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs". In: *arXiv preprint arXiv:1801.05453* (2018).

[86] W James Murdoch and Arthur Szlam. "Automatic rule extraction from long short term memory networks". In: *arXiv preprint arXiv:1702.02540* (2017).

[87] W. James Murdoch et al. "Interpretable machine learning: definitions, methods, and applications". In: *arXiv preprint arXiv:1901.04592* (2019).

[88] Harvard University Office of Institutional Research. "Exhibit 157: Demographics of Harvard College Applicants". In: *http://samv91khoyt2i553a2t1s05i-wpengine.netdna-ssl.com/wp-content/uploads/2018/06/Doc-421-157-May-30-2013-Report.pdf* (2018), pp. 8–9.

[89] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: *Distill* 2.11 (2017), e7.

[90] Julian D Olden, Michael K Joy, and Russell G Death. "An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data". In: *Ecological Modelling* 178.3-4 (2004), pp. 389–397.

[91] Bruno A Olshausen and David J Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision research* 37.23 (1997), pp. 3311–3325.

[92] Nicolas Papernot and Patrick McDaniel. "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning". In: *arXiv preprint arXiv:1803.04765* (2018).

[93] Nicolas Papernot et al. "The limitations of deep learning in adversarial settings". In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE. 2016, pp. 372–387.

[94] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition". In: *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*. IEEE. 1993, pp. 40–44.

[95] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[96] Fernando Pérez and Brian E Granger. "IPython: a system for interactive scientific computing". In: *Computing in Science & Engineering* 9.3 (2007).

[97] Harold Pimentel, Zhiyue Hu, and Haiyan Huang. "Biclustering by sparse canonical correlation analysis". In: *Quantitative Biology* 6.1 (2018), pp. 56–67.

[98] Nina Poerner, Hinrich Schütze, and Benjamin Roth. "Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 340–350.

[99] Pranav Rajpurkar et al. "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv preprint arXiv:1606.05250* (2016).

[100] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. 2003, pp. 133–142.

[101] Jonas Rauber, Wieland Brendel, and Matthias Bethge. "Foolbox v0. 8.0: A python toolbox to benchmark the robustness of machine learning models". In: *arXiv preprint arXiv:1707.04131* (2017).

[102] Stephen J Read and Amy Marcus-Newhall. "Explanatory coherence in social explanations: A parallel distributed processing account." In: *Journal of Personality and Social Psychology* 65.3 (1993), p. 429.

[103] 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*. ACM. 2016, pp. 1135–1144.

[104] Anna W Roe et al. "Toward a unified theory of visual area V4". In: *Neuron* 74.1 (2012), pp. 12–29.

[105] Anna Rohrbach et al. "Grounding of textual phrases in images by reconstruction". In: *European Conference on Computer Vision*. Springer. 2016, pp. 817–834.

[106] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. "Right for the right reasons: Training differentiable models by constraining their explanations". In: *arXiv preprint arXiv:1703.03717* (2017).

[107] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc. Boston, MA, 2016. URL: `http://www.rstudio.com/`.

[108] Cynthia Rudin. "Please Stop Explaining Black Box Models for High Stakes Decisions". In: *arXiv preprint arXiv:1811.10154* (2018).

[109] Alexander M Rush, Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization". In: *arXiv preprint arXiv:1509.00685* (2015).

[110] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[111] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *See https://arxiv. org/abs/1610.02391 v3* 7.8 (2016).

[112] Tao Shi et al. "Daytime arctic cloud detection based on multi-angle satellite data with case studies". In: *Journal of the American Statistical Association* 103.482 (2008), pp. 584–593.

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

[114] Avanti Shrikumar et al. "Not just a black box: Learning important features through propagating activation differences". In: *arXiv preprint arXiv:1605.01713* (2016).

[115] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[116] Chandan Singh, W James Murdoch, and Bin Yu. "Hierarchical interpretations for neural network predictions". In: *arXiv preprint arXiv:1806.05337* (2018).

[117] Daniel Smilkov et al. "Smoothgrad: removing noise by adding noise". In: *arXiv preprint arXiv:1706.03825* (2017).

[118] Richard Socher et al. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[119] Jost Tobias Springenberg et al. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[120] Hendrik Strobelt et al. "Visual analysis of hidden state dynamics in recurrent neural networks". In: *CoRR, abs/1606.07461* (2016).

[121] Carolin Strobl et al. "Conditional variable importance for random forests". In: *BMC bioinformatics* 9.1 (2008), p. 307.

[122] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *ICML* (2017).

[123] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

[124] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[125] Kai Sheng Tai, Richard Socher, and Christopher D Manning. "Improved semantic representations from tree-structured long short-term memory networks". In: *arXiv preprint arXiv:1503.00075* (2015).

[126] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[127] Michael Tsang, Dehua Cheng, and Yan Liu. "Detecting statistical interactions from neural network weights". In: *arXiv preprint arXiv:1705.04977* (2017).

[128] Michael Tsang et al. “Can I trust you more? Model-Agnostic Hierarchical Explanations”. In: *arXiv preprint arXiv:1812.04801* (2018).

[129] Mai-Anh T Vu et al. “A shared vision for machine learning in neuroscience”. In: *Journal of Neuroscience* (2018), pp. 0508–17.

[130] Michael Waskom et al. “Seaborn: statistical data visualization”. In: *URL: https://seaborn. pydata. org/(visited on 2017-05-15)* (2014).

[131] Donglai Wei et al. “Understanding intra-class knowledge inside CNN”. In: *arXiv preprint arXiv:1507.02379* (2015).

[132] Paul J Werbos. “Generalization of backpropagation with application to a recurrent gas market model”. In: *Neural networks* 1.4 (1988), pp. 339–356.

[133] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer, 2016.

[134] Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.2.1. 2017. URL: `https://CRAN.R-project.org/package=tidyverse`.

[135] Siqi Wu et al. “Stability-driven nonnegative matrix factorization to interpret spatial gene expression and build local gene networks”. In: *Proceedings of the National Academy of Sciences* 113.16 (2016), pp. 4290–4295.

[136] Kelvin Xu et al. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International Conference on Machine Learning*. 2015, pp. 2048–2057.

[137] Jason Yosinski et al. “Understanding neural networks through deep visualization”. In: *arXiv preprint arXiv:1506.06579* (2015).

[138] Bin Yu. “Stability”. In: *Bernoulli* 19.4 (2013), pp. 1484–1500.

[139] Omar Zaidan, Jason Eisner, and Christine Piatko. “Using annotator rationales to improve machine learning for text categorization”. In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. 2007, pp. 260–267.

[140] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[141] Quanshi Zhang et al. “Interpreting CNN knowledge via an Explanatory Graph”. In: *arXiv preprint arXiv:1708.01785* (2017).

[142] Xiang Zhang, Junbo Zhao, and Yann LeCun. “Character-level convolutional networks for text classification”. In: *Advances in neural information processing systems*. 2015, pp. 649–657.

[143] Luisa M Zintgraf et al. “Visualizing deep neural network decisions: Prediction difference analysis”. In: *arXiv preprint arXiv:1702.04595* (2017).