Techniques for Interpretability and Transparency of Black-Box Models

by

Yilun Zhou

B.S.E., Duke University (2016)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2023

Author ....
Department of Electrical Engineering and Computer Science
January 23, 2023

Certified by ....
Julie A. Shah
H. N. Slater Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by ....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Techniques for Interpretability and Transparency of Black-Box Models

by

Yilun Zhou

Submitted to the Department of Electrical Engineering and Computer Science
on January 23, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The last decade witnessed immense progress in machine learning, which has been deployed in many domains such as healthcare, finance and justice. However, recent advances are largely powered by deep neural networks, whose opacity hinders people's ability to inspect these models. Furthermore, legal requirements are being proposed to require a level of model understanding as a prerequisite to the deployment and use. These factors have spurred research that increases the interpretability and transparency of these models.

This thesis makes several contributions in this direction. We start with a concise but practical overview of the current techniques for defining and evaluating explanations for model predictions. Then, we observe a novel duality between definitions and evaluations of various interpretability concepts, propose a new way to generate explanations and study the properties of these new explanations. Next, we investigate two fundamental properties of good explanations in detail: correctness – whether the explanations are reflective of the model's internal decision making logic, and understandability – whether humans can accurately infer the higher level and more general model behaviors from these explanations. For each aspect, we propose evaluations to assess existing model explanation methods and discuss their strengths and weaknesses. Following this, we ask the question of what instances to explain, and introduce the transparency-by-example perspective as an answer to this question. We demonstrate its benefits in revealing hidden properties of both image classifiers and robot controllers. Last, the thesis identifies directions for future research, and advocates for a tighter integration of model interpretability and transparency into the ecosystem of trustworthy machine learning research that also encompass efforts such as fairness, robustness and privacy.

Thesis Supervisor: Julie A. Shah
Title: H. N. Slater Professor of Aeronautics and Astronautics

Acknowledgment

Finishing a PhD degree is never easy, and I am indebted to many people along the way. First, I would like to express my gratitude to my advisor Prof. Julie Shah, who has been providing unwavering support to me throughout this process. Especially as I sifted through a multitude of potential thesis topics, she constantly encouraged my exploration, which is probably the biggest factor contributing to my development as an independent researcher.

In addition, I would like to thank Dr. Marco Tulio Ribeiro of Microsoft Research for being a wonderful collaborator and mentor for the past three years. His research taste strongly shaped the projects that I have undertaken and pushed me to shoot for higher goals. In addition to providing guidance on technical subject matter, he also helps me on my technical communication skills, both orally and in writing. In particular, I have formed the habit of always putting a Figure 1 with a bird-eye view of the paper on the first page.

Besides Marco, Prof. Jacob Andreas and Prof. Peter Szolovits served on my thesis committee, and via many committee meetings, have provided many insightful comments and suggestions to the thesis. In addition, Jacob has also been involved in a project led by Shawn Im, an undergraduate mentored by me, and I have benefited greatly from his inputs.

I would also like to thank all my collaborators in these years, by taking the liberty of chronicling my research career so far. It all began with my undergraduate research, in which I worked with Prof. George Konidaris and Prof. Kris Hauser on various robotics projects [60, 174, 175, 177]. Even though I was a sophomore at that time, they trusted me with leading my own research project rather than assigning some random busywork to me, which made me deeply interested in a research career.

I started my graduate study working on commonsense reasoning, when Julie and I contacted Prof. Steven Schockaert of Cardiff University for collaboration, and he generously helped me on two projects in this line [178, 179].

After finishing them and my master's degree, I explored various topics for this PhD thesis, during which I worked with Mycal Tucker on projects related to multi-agent communication and coordination. Though transitional, the collaboration [152] introduced me to this fascinating area with its unique beauty and challenges.

In hindsight, this thesis really started with a project [21] I worked on with labmates Serena Booth and Ankit Shah towards the end of 2019, and it was originally their course project that they graciously invited me to join. The follow-up paper on this topic [180] also benefited with the participation of Dr. Nadia Figueroa, who was a postdoc in the lab at the time and is now on the faculty of University of Pennsylvania.

During AAAI 2020, probably the last major in-person AI conference before everything was put on a stop due to the COVID-19 pandemic, I was fortunate to attend a paper presentation by Vishakha Patil, where I found some shared interest into the fairness problem in decision making settings, and our subsequent collaboration [48], with Ganesh Ghalme and Vineet Nair, was an eye-opening experience for me into the world of theoretical machine learning.

In the summer of 2020, I interned at Facebook AI, working with Dr. Asish Ghoshal, who gave me much freedom in choosing and defining my research [181] while providing very detailed technical mentorship. Although the project was not too related to the general effort that the team was working on at the time, it allowed me to explore yet another area: active learning.

During the summer, Serena and I supervised several undergrad students, and one of them, Yiming Zheng, continued working with us after the summer and acquired such research maturity to lead a research project from conception to publication [172]. Being able to help him finish the project reminds me of my own undergrad research with George and Kris, and gives me a deeply satisfying sense of closure.

After the summer, Serena and I continued on a new project [182] into which we invited Marco, who continued mentoring me on a follow-up paper [183] and supervising my internship at Microsoft Research in the summer of 2022.

Looking back on these five years, I would never have imagined to end up with this path, but I am grateful for every turn it takes and honestly cannot think of a better trajectory. Everything just works out in the way intended.

Other than the technical side, I am grateful to be part of the Interactive Robotics Group during these years. Everyone in the group is glad to offer help, and it has always been fun to have whatever conversation that comes up. Similarly, I would also like to thank all my friends both at MIT and from other places for the time spent and activities done together. Without their support, completing this thesis would have been a less enjoyable experience.

Last but not least, I want to thank my family. Growing up in China, moving to the US for college and settling down here, I have been supported by my parents through all these times. Besides, my extended family of grandparents have also offered great support in countless ways. It is the care of all of them that makes me who I am today, and I am forever grateful to them.

Contents

List of Figures

List of Tables

Chapter 1

Promises and Perils of Black-Box Models

1.1 Introduction

Over the past ten years, machine learning (ML) has rapidly changed society. From everyday products and features like Google Translate, Facebook friend tagging and Snapchat filters, to expert-knowledge domains like medical diagnosis, insurance underwriting and loan approval, to emerging technologies like autonomous driving, virtual reality and gene therapy, ML has played key roles in all of them, and it is widely expected that its importance will only be growing.

Nonetheless, the wide adoptions of ML have brought unique challenges. The goal of ML is to discover patterns automatically from the data, when we cannot manually specify them. For example, in image classification, because it is extremely hard, if at all possible, to write a manual rule classifying whether a matrix of pixels looks more like a cat or dog, we resort to ML to learn a decision boundary in the space of pixel matrices to separate those of cats from those of dogs. When the boundary has very sophisticated shapes, as would be needed for most of the complex tasks, understanding it becomes a severe challenge. As a result, the models that learn to computing these boundaries, often represented by deep neural networks or tree ensembles (e.g., random forests or boosted trees), are generally called "black-box models."

But, why do we need to, or want to, understand the models? Besides satisfying a general curiosity, knowing what the models learn serves very practical purposes. Consider a model trained on past lending data to make new mortgage approval decisions. While we would ideally want the model to make predictions based on the financial health and repayment likelihood of the applicant, it could very well learn to rely on *spurious correlations.* For example, historically African American people are often both financially less stable and discriminated by banks, resulting in a strong correlation of this race with loan denial. Hence, the model could learn a simple rule of just denying African American applicants regardless of their other factors, which is largely consistent with the training data. For this model, if we have model explanations that highlight the importance of the race feature to the model prediction, we could easily discover the racial bias.

As another example, suppose that we want to train a neural network to detect cancers from X-ray images, wherein the data come from two sources: a general hospital and a specialized cancer center. As can be expected, images from the cancer center contain many more cancer cases. However, when rendering the X-ray images, the cancer center adds a small timestamp watermark to the top-left corner. Since the timestamp is strongly correlated with cancer presence, a model may learn to use it for prediction. In this case, while the model can achieve very high accuracy by identifying either the timestamp or the genuine medical signal of the cancer, the former mode of operation would miss all detections on cancer-positive images without the timestamp watermark, such as those coming from different hospitals. Thus, if we realize that the watermark is indeed important, then we should discard the model and redevelop the data collection and model training pipeline.

Besides these hypothetical setups, a general lack of understanding into these models have caused many high-profile failures. For example, the image recognition system in Google Photos labeled people of dark skins as gorillas, and the conversation bot Tay by Microsoft generated hate speech under certain prompts. Because we do not have good understanding into the model behavior, it is very hard to anticipate what image or what prompt could lead to such egregious behaviors and proactively prevent them from happening.

Such concerns have led to the development of the area of trustworthy ML, broadly aimed at making ML systems reliable and dependable after deployment. It encompasses many sub-fields, and popularly studied ones include interpretability, transparency, fairness, robustness and privacy. This thesis focuses on the first two, which attempt to obtain better understanding of black-box models by generating explanations for their predictions or studying its various behaviors (e.g., high-confidence failures). We focus the thesis on these two topics as they are the "means," to the "ends" of fairness, robustness and privacy.

The earlier examples on mortgage approvals and medical diagnosis demonstrate how good model understanding could help us detect bias and dataset artifacts that learned by the system, two important directions in fairness and robustness research. In general, given that we know, or suspect, that the model could be discriminative or non-robust in certain ways, it is relatively easy to test such hypotheses. However, the known unknown problem, where we know that something could be going on but do not know what, is much harder to solve, resulting in people often surprised by the shortcuts learned by the model. Interpretability and transparency methods are often used to solve such problems, although we also note some important limitations of existing methods in Chapter 4 and 5.

Finally, an active area of privacy research concerns with training data leakage, in which certain analysis the model could reveal the exact data instances used to train it. This issue is ever more alarming as many large language and image models are trained to explicitly generate realistic outputs in response to certain inputs. While they are very helpful in many applications, such as dialog systems or computational creativity (rendering photos in certain artistic styles), it is important to make sure that do not leak personally identifiable or otherwise sensitive information. Interpretability and transparency techniques could again be used to study the model and help the developers understand the situations under which such leakage is likely to happen, so that certain safeguarding measures can be put in place.

1.2 Thesis Summary

Below, we give an overview of Chapter 2 to 7, which form the technical contents of this thesis. Chapter 8 reiterates the main ideas of this thesis, and points to avenues of future work. At a high level, this thesis focuses on the model understanding pipeline as shown in Figure 1-1.

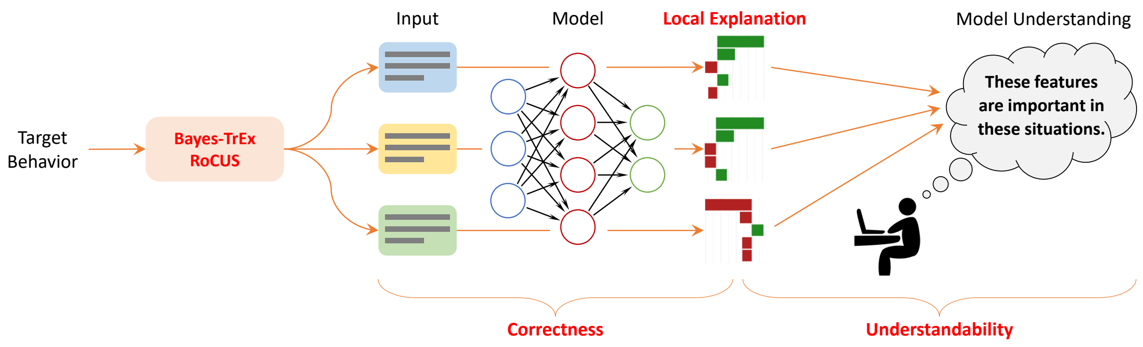


Figure 1-1: The model explanation pipeline and its various components.

The standard approach for model understanding starts at the second stage the pipeline, where we have identified certain input instances to study. From here, local explanations are generated to illustrate the model reasoning on these inputs. In this thesis, the "model reasoning" mostly refer to the importance of each feature. Next, these local explanations are summarized into more global and general pieces of model understanding by the human consumer of these explanations to inform subsequent decisions (e.g., discarding the model due to racial discrimination).

After a brief overview of current state of model interpretability research, focusing on methods for generating and evaluating local explanations in Chapter 2, we propose a novel paradigm for generating explanations in Chapter 3 and discuss its implications. Then in Chapter 4 and 5 we introduce two crucial properties of model explanations, their correctness and understandability, present methods to evaluate these properties, and discuss the implications of these findings on the future research of model explanations.

Last, this thesis also advocates an "earlier" start of the model understanding pipeline. Rather than starting from arbitrary or random input instances, we should explicitly consider each model behavior, such as ambivalent predictions or high-confidence mistakes, and use them to guide the selection of inputs to explain. Specifically, Chapter 6 and 7 introduce the Bayes-TrEx and RoCUS frameworks to find input instances that conform to a certain target model behavior. In a sense, these two frameworks answer the question of "what to explain."

1.2.1 Background on Model Interpretability

This thesis assumes general knowledge of deep learning on image and text data, such as convolutional neural networks (CNN) [92], long short-term memory (LSTM) networks [64] and transformer models [156], equivalent to what is typically covered in a graduate-level machine learning class. To prepare for technical discussion on model interpretability, Chapter 2 presents an overview of the current state of model interpretability. We first introduce the overall goal of model interpretability, and a taxonomy of interpretability methods on two dimensions: local vs. global methods and *post hoc* explanations vs. inherently interpretable models. Our focus in this thesis is on local *post hoc* methods, and in particular a type of explanation called feature attributions, also known as feature importance or saliency maps, which is the focus of the ensuing technical discussions.

We introduce several common ways to define model explanations. Given a model that we want to understand and an input whose model prediction we want to explain, each definition generates an explanation that helps people understand the model's decision making logic in a specific way.

Next, we discuss the problem of evaluating these explanations. Unlike the evaluations of model predictions, which are often validated against the ground truth labels using metrics such as accuracy or F1 score, the evaluations of model explanations are complicated by the fact that we do not know the ground truth working mechanism of the black-box model – the very goal of interpretability techniques. As a result, people mainly resort to indirect approaches based using proxy metrics, for which we introduce the most popular one as an example, covering both its intuition and its mathematical formulation.

1.2.2 Definition-Evaluation Duality

After the background overview, Chapter 3 to 5 study three topics in model interpretability. Chapter 3 directly follows from the discussions on the definitions and evaluations interpretability methods. In particular, with proper transformations, we demonstrate that every definition concept can be used as evaluation, while every evaluation concept can also be used as definition. We refer to this surprising link as the *definition-evaluation duality*.

This observation brings up immediate questions on the current practice of explanation definition and evaluation. Given that our "stamp of approval" on model explanations is given after they pass the evaluations, we could instead transform the target evaluation to the definition, which generates explanations that automatically optimize the evaluation metric. Why are we not doing this? More generally, in the community, some concepts are used exclusively for definitions while others for evaluations. What is the fundamental reason for this separation? Last, how should we best evaluate our model explanations, now that this transformation is readily available and the explanations that optimize the desired evaluation metric can be explicitly defined and found?

In this chapter, using a high-performing sentiment classifier trained on a movie review dataset as a case study, we investigate the implications of this definition-evaluation duality, by explicitly converting the combination of two widely used evaluation metrics, comprehensiveness and sufficiency [39] discussed in Chapter 2, into its definition form. In other words, the explanations computed according to this definition jointly optimize these two metrics. Such a result should have been celebrated by works that try to propose novel definitions and show their superior performance on these metrics, but, because of the duality, is just expected.

To study the desirability of the resulting explanations, we subject them to a battery of additional testings. First, we evaluate them on other proxy metrics that are not directly optimized for. Second, we compute the metric value degradation under explanation value perturbation to study their robustness. Third, under the reasonable assumption that the high-performing sentiment classifier needs to use sentiment-laden words for predictions, we check for this alignment between human annotations of sentiment values of words and their explanation values. Last, we modify the datasets such that models trained on the modified versions have to rely on certain signals in the input to make highly accurate predictions, and check if the explanations can highlight these signals. We briefly introduce the dataset modification procedure in this chapter but discuss this evaluation and the implications of the results more extensively in Chapter 4.

Across all four evaluations, we find that our proposed explanations perform favorably or competitively against existing ones, which should should motivate the use of these explanations as strong alternatives to the current practices. We close this chapter by pointing out two related future work directions worth exploring. First, given that evaluations can be used as strong definitions, can some definitions be used as strong evaluations? Answering this question requires us to clearly establish what evaluations are preferable to others, which brings us to the next direction: how should we best evaluate model explanations, given that proxy metrics can be directly optimized? We argue that we should hold explanations accountable to specific types of utilities, such as model debugging, model auditing and human decision support, and evaluate their effectiveness in carefully designed user studies.

1.2.3 Correctness of Model Explanations

In Chapter 4, we dive deeper into the dataset modification procedure that is used as the last evaluation in the previous chapter. Specifically, we consider the problem of evaluating feature attribution explanations on models whose working mechanisms are known. We induce the known ground truth by modifying the dataset in a principled way such that models attaining a high performance have to rely on specific features that we inject.

Starting with a natural dataset with arbitrary unknown input-label correlation, the basic intuition behind our dataset modification proposal is to first weaken the correlation such that *any* model could not perform very well on this correlation-weakened dataset. Then we inject to the inputs features strongly correlated with labels, allowing models trained on this final dataset to use the injected features to make high-performance predictions. If we do observe this high generalization accuracy, then we are guaranteed that the model has to use our injected features, which gives us the ground truth to validate our feature attribution explanations.

We implement the first step of correlation weakening by corrupting the labels. For example, with some probability, we randomly flip the label to the other one (in the binary setup). This ensures that the model can achieve an accuracy up to the random flip probability level. In the most extreme case, we can flip the label with probability of 50%, essentially decorrelating the label from the input, guaranteeing that any model could not achieve more than 50% accuracy in expectation.

Then in the second step, we inject features conditional on the new labels to ensure that they do have strong (or even perfect) correlations. An important pragmatic consideration of this feature injection is that it needs to be localized, in that only a subset of feature values are affected. For example, for image data, we need to apply the features to a patch of pixels rather than the whole image, because in the latter case, the model can again use any part of the image for decision making and we cannot invalidate any saliency maps.

After these two steps, we know that for models now achieving high accuracy, the feature injection regions are definitely important to model predictions. Furthermore, if the labels are completely randomized in the first step, then a perfect model only focuses on the feature injection regions as everything outside is a distractor. Thus, we can evaluate %Attr, defined as the sum of attribution scores (absolute value if necessary) assigned inside the feature injection regions as a percentage of the sum of all attribution scores, and use it as a measure of the saliency map quality.

In our experiments, we use this procedure to study saliency maps for image classifiers, rationale models for text classifiers and attention mechanisms for text classifiers. The features we inject are inspired from studies on dataset spurious correlations, which show the surprising impact of data artifacts such as watermark and blurring for images, and article and stop words for texts.

Overall, we found inconsistent results for most explanation methods, and there is no clear winner across the board. For images, several saliency map methods simply could not recognize the high importance of certain injected features. Furthermore, when the features are injected asymmetrically to only the positive class, it is much harder to explain the absence of the feature for the negative prediction than its presence for the positive prediction. For texts, while rationale models can generally identify the important words as rationales, they are also susceptible to over-selection of irrelevant words, at the expense of missing certain important ones. By contrast, attention scores have been found to correlate poorly with ground truth importance of words.

Beyond concrete issues that we have identified above, which are worth investigating in future research, the overall results also have practical implications on the use of interpretability methods in general. As machine learning models have been found to use various kinds of unexpected shortcuts, model explanations like saliency maps have been hoped by many to provide a solution for the shortcut detection problem. In addition, the finding that the saliency maps focus on the genuine signals (such as the tumor in cancer detection) instead of any apparent shortcuts has been used to argue that the model is right for the right reason. As we demonstrate, even when manually injected spurious correlations are guaranteed to have a large impact on the model prediction, the saliency maps may not correctly identify them universally. Thus, we should be cautious of certifying or trusting models simply because they "pass" the saliency map inspections.

1.2.4 Understandability of Model Explanations

While both proxy metrics studied in Chapter 3 and the dataset modification procedure proposed in Chapter 4 are largely concerned about the correctness of model explanations, in Chapter 5 we propose an orthogonal aspect to correctness that has received much less attention: their understandability.

As its name suggests, understandability refers to how well people understand these local explanations, and in particular, how well the local explanations shed light on the more general model behaviors. To motivate its necessity in the enterprise of model understanding, it is perhaps best to start with a negative example. Consider hashing functions, which map inputs of arbitrary length to a fixed-length output (called "hash") in a way that the forward mapping is very easy to compute but the reverse mapping is extremely hard to find, even when the forward mapping (i.e., the hashing function) is public knowledge. Such functions are used for forgery prevention: if Alice wants Bob to publish a piece of software on his website but is concerned that he may tamper with the software, she can publish the hash of the software on her own website, so that Charlie downloading the software from Bob's website can check its hash against the one that Alice publishes to ensure its authenticity. Notably, it is very easy for Alice to compute the hash and for Charlie to verify it, but it is very hard for Bob to modify the software while still ending up with the same hash.

In this sense, the hashing function can be considered as a black-box function, an even more erratic one than a neural network. Can we compute local explanations for the hashing function? Without going into much technical details right now (which will be covered in Chapter 2), the answer is yes, because most of the explanation methods only need input-output access of the function (which is the forward mapping). Consider the hashing function that maps, for example, a software of 1 MB $= 8 \times 2^{20}$ bit to a 128-bit output hash. We could compute the influence of each of the 8×2^{20} input bits to each of the 128 output bits. This computation is mechanical, by just following the definitions of the explanation method.

However, can we *understand* the model from these local explanations? In a very narrow sense, also yes, in that the explanation can tell us whether flipping each input bit can induce the flipping of each output bit. However, this knowledge is rarely, if at all, useful in any practical purposes. Arguably one outcome of model understanding is *approximate model inversion* – obtaining a general knowledge of what kinds of inputs get mapped to what kinds of outputs. In this aspect, we have failed completely: hashing functions are designed (and stress-tested for many decades) to be non-invertible in anyway other than brute force search!

We have just observed a case where explanations, however correct, cannot help us understand the model which is designed to be fundamentally "non-understandable." Now consider the other extreme of linear regression. For two input variables x_1 and x_2, define the output to be $y = 10x_1 + 5x_2 + 3$. Many explanation methods will compute the importance of x_1 and x_2 as 10 and 5 respectively, regardless of the particular (x_1, x_2) value. This is a reasonable definition, because the function is linear across the entire input space. Thus, in this case, the local explanations do enable us to understand the model very well: x_1 is twice as important as x_2.

These two cases illustrate that how well the model explanation helps us understand the model is orthogonal to how correct it is. The former is much more model dependent than explanation method dependent, and neural networks are most likely somewhere between linear regressions and hashing functions. Note that we can also have trivially understandable but totally incorrect explanations. For the hashing function case, we can "define" an explanation that assigns zero importance to every input bit, claiming zero importance for each input bit. Obviously, this is very incorrect, but it is quite easy to understand – "none of them matter." Again, this understanding is useless because it is derived from the wrong explanations.

The technical contribution of Chapter 5 is, then, to mathematically formalize the concept of understandability. To this end, we propose the explanation summary framework (ExSum) that distills a large number of model explanations into a small number of model understanding pieces, each revealing one aspect of how the model works in general (e.g., words that carry strong positive sentiments have generally positive contributions toward the sentiment prediction). ExSum computes three metrics: coverage, validity and sharpness to give users an intuitive sense of the quality of their model understanding. As an engineering contribution, we release ExSum as a standalone Python package for wider adoption.

We use ExSum to understand the explanations for two text models, a sentiment classifier and a paraphrase detector (i.e., given two sentences, predict if they are paraphrases of each other). For very popular local explanation techniques, we have several findings across both models. First, it is indeed precarious to try to understand the models from one or a few local explanations, in that we easily risk overgeneralization, yet unfortunately this is predominantly the current practice. Second, a careful analysis with ExSum highlights important limitations in our understanding into these models. In other words, both of the neural network models are far from the linear regression end – but they are nowhere near the hashing function either, as local explanations do shed light on important working mechanisms of these models, such as the importance of sentiment-laden words on model predictions for the sentiment classifier. This is more or less expected, as while neural networks generally behave reasonably, they still use subtle signals in the inputs that are not well understood, which are crucial to their high performance. Last, on the positive side, ExSum helps us identify model working mechanisms that are previously overlooked, such as asymmetric treatment of positive vs. negative words for the sentiment classifier and positive vs. negative predictions for the paragraph detector.

To close this chapter, we discuss the relationship between understandability and existing evaluations other than correctness, such as human alignment, robustness, counterfactual similarity and plausibility. As we demonstrate, while all these evaluations can be in conflict with correctness, in that the most correct explanation should legitimately have low scores on these aspects, we unify them under the theme of understandability. In other words, if we consider understandability as a complementary aspect to correctness, all of these evaluations are can be considered as special cases of understandability, even though they do not evaluate correctness.

1.2.5 Model Transparency by Example

Staring from Chapter 6 and continuing into Chapter 7, we consider a different but closely related topic of model transparency. Similar to interpretability, it is also aimed at developing a set of techniques that help people understand their models better, but is not limited to the use of model explanations. For example, one can inspect and analyze errors made by a model or study the purpose of each component of the model separately. In this thesis, we propose the transparency-by-example (TrEx) perspective to support model understanding.

The premise of TrEx is simple: to understand a model, we should understand its various behaviors, and to understand each of these behaviors, we could do so by finding and studying representative instances that elicit it. The main contribution of this Chapter 6 is a rigorous definition of the "behaviors" of model and a Bayesian posterior sampling formulation to find instances for a given behavior. Hence, we call the resulting framework as Bayes-TrEx.

We represent model behaviors using behavior functions. Before formalizing the concept, we first give three examples.

1. Ambivalence: the (binary) model makes highly uncertain predictions on inputs (e.g., 55% positive and 45% negative). Such inputs mark the decision boundary of the model. We define the behavior function as the difference between the two class probabilities. We want to find inputs with behavior function values as close to zero as possible.

2. High-confidence mistakes: the model makes highly certain predictions (e.g., 97% class A) yet they are incorrect. Such inputs help detect any blind spot by the model. The behavior function is defined as the probability for the predicted class if the prediction is incorrect with respect to the ground truth label, and zero otherwise. We want to find inputs with behavior function values as close to one as possible (which automatically filters out those with correct predictions).

3. High-confidence predictions under covariate shift: the model makes highly certain predictions for a particular class on data from a target distribution that is different from the training distribution. Such instances help assess whether the model's covariate shift behaviors are reasonable. The behavior function is defined as the probability for the predicted class scaled by the probability of the input under the target distribution. Again, we want to find inputs with behavior function values as close to one as possible.

In all three cases we can identify some common elements. First, the behavior function value is based on the model prediction, whether ambivalent predictions or high-confidence ones. In addition, it may also consider the input (e.g., probability under the shifted distribution) and the label (e.g., to identify incorrect model predictions). Last, we have a target behavior function value, and want to find instances that (approximately) achieve the target value.

Because the target behavior value may be rare (e.g., it is hard to identify high-confidence errors for very accurate models), uniformly sampling inputs and filtering for the behavior value satisfaction is very ineffective. Instead, in this chapter, we recognize that the behavior value can be decomposed into two parts, one dependent on the model prediction and the other on the input instance. We then draw analogies between these two parts and the concepts of likelihood and prior in Bayesian inference, and transform the problem into sampling from the resulting posterior, for which we can employ standard Markov chain Monte Carlo (MCMC) techniques.

In the experiment, besides the above three analyses, we demonstrate two additional ones: high-confidence in-distribution instances to identify prototypical examples for a certain class and high-confidence instances of novel unseen classes to study extrapolation behaviors. We evaluate three image classifiers trained on MNIST, Fashion-MNIST and CLEVR respectively. For each model, these analyses help us gain additional model understanding that would be hard to get from a test set evaluation, due to the poor coverage of rare behaviors and annotation errors of the test set. We also investigate a failure case of Bayes-TrEx to identify qualifying instances in one particular case, and conclude the reason as no such instances existing in the first place, rather than a sampling failure.

Finally, we also argue that Bayes-TrEx, and the transparency-by-example framework in general, could be considered as answers to the "what to explain" question and thus interface with the interpretability methods discussed in the earlier chapters. Currently, the standard practice to understand models via explanations is to compute them on test set inputs and look for any interesting phenomena. However, as a central argument in this chapter, an investigation based on the test set may not be suitable for studying certain behaviors. Instead, Bayes-TrEx provides a way to sample inputs that conform to specific target behaviors, which can then be analyzed using interpretability techniques to find out the model reasoning responsible for these behaviors. Concretely, we show, using a saliency map analysis, that one type of high-confidence error in the CLEVR model results from mis-detections of certain small objects.

1.2.6 Robot Controller Transparency

As the last technical contribution of this thesis, Chapter 7 extends Bayes-TrEx to study robot controllers, which brings new opportunities and challenges. We call this proposal RoCUS, for robot controller understanding via sampling. We represent an "instance" as an environment and a (potentially stochastic) trajectory generated by the controller for this environment. The behaviors that we are interested in for robot controllers are also different from those for image classifiers, and some robot controller-specific questions we may ask include "when it is likely to fail," "when it is likely to exhibit confusing behaviors," and "when it steers too closely to the obstacles."

To answer these questions, in this chapter, we define a new set of behaviors that capture properties of robot trajectories such as task completion, safety and legibility. We group them into two broad categories: intentional and emergent. Intentional behaviors are specifically optimized when the controller is constructed, such as task completion and trajectory length. Finding instances that satisfy or miss such behaviors help us understand the success of the optimization or learning of our controller. By contrast, emergent behaviors are not directly specified as objectives, but instead emerge due to the optimization of the intentional behaviors. For example, minimizing trajectory length may cause the robot to have very small obstacle avoidance margin and thus safety issues, which can be discovered by finding instances where the safety behavior is violated (or, conversely, unsafe behavior is present). As in the Bayes-TrEx case, the instances for a variety of behaviors together help achieve a holistic understanding of the strengths and weaknesses of the robot controller.

As a technical detail, Bayes-TrEx is originally designed to work with finite behavior value targets, such as 0 or 1. This is a reasonable assumption because the behavior functions are defined based on predicted probability. However, for robot controllers, we may want to find environments that lead to maximal values of certain behavior, such as trajectory length. One *ad hoc* fix would be to set the target value to be a very large one. In RoCUS, we propose a transformation on behavior values to automatically deal with this case.

In the experiment, we study two domains, each with three different controllers. For a 2D navigation task with obstacles, we study an imitation learning (IL) controller, a dynamical system (DS) controller, and a rapidly-exploring random tree (RRT) controller. For a 7 degree-of-freedom (DoF) robotic arm table-top reaching task, we study a reinforcement learning (RL) controller, and the same formulations of DS and RRT controllers.

Our analyses of a suite of behaviors exhibit several insights of these controllers. For example, both IL and DS controllers demonstrate unsafe behaviors with small margins around obstacles on the 2D navigation task, but for different reasons: the IL controller is adversely affected by data collection while the DS controller suffers from an issue in its mathematical formulation. For the 7 DoF arm reaching task, both RRT and DS controllers suffer from an asymmetry in the kinematic structure of the arm, and it is particularly severe for DS, which often fails to reach the target in our first iteration of design. With the help of RoCUS, we are able to pinpoint the cause of the failure and make it succeed more frequently. By contrast, the RL controller learns to work around the asymmetry and does not exhibit any issues.

1.2.7 References

Chapter 3 to 7 are based on the following papers (* denotes equal contribution):

- Yilun Zhou and Julie Shah. The Solvability of Interpretability Evaluation Metrics. *Findings of the Association for Computational Linguistics: EACL*, 2023.
- Yilun Zhou, Serena Booth, Marco Tulio Ribeiro and Julie Shah. Do Feature Attribution Methods Correctly Attribute Features? In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.
- Yilun Zhou, Marco Tulio Ribeiro and Julie Shah. ExSum: From Local Explanations to Model Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2022.
- Serena Booth*, Yilun Zhou*, Ankit Shah and Julie Shah. Bayes-TrEx: A Bayesian Sampling Approach to Model Transparency by Example. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.
- Yilun Zhou, Serena Booth, Nadia Figueroa and Julie Shah. RoCUS: Robot Controller Understanding via Sampling. In *Conference on Robot Learning (CoRL)*, 2021.

1.3 Thesis Statement

In this thesis, we scrutinize the use of local explanations to understand models, identify problems with the current practice and propose solutions to them. The thesis statement of the thesis is:

Local model explanations are promising, but currently insufficient, to yield a holistic understanding of the model. This thesis reflects on the questions of what to explain, how to explain, and how to evaluate explanations and proposes respective answers, each of which is crucial for the end goal of model understanding.

Chapter 2

Background on Model Interpretability

2.1 Overview

This thesis builds upon almost a decade of research into model interpretability, and makes use of many fundamental concepts and algorithms. In this chapter, we provide a self-contained introduction to the general field of model interpretability, focusing more heavily on the topics directly related to this thesis. For additional information, the textbook by Molnar [108] is an excellent introduction to this field. Readers already with such a background should feel free to skim through or skip this chapter.

While there are many definitions and perspectives on what term of interpretability specifically encapsulates [40, 50, 134], we take the pragmatic approach and consider it to include any methods or frameworks that provide some form of explanations of the model's prediction logic that are intended for people to make more informed decisions.

2.2 Axes of Interpretability Methods

At a high level, interpretability techniques can be classified into local vs. local methods, and *post hoc* explainers vs. inherently interpretable models.

On the local vs. global axis, a local explanation is defined on an individual instance and tries to exhibit the model reasoning on the prediction of this particular instance. For example, a wide range of *feature attribution explanations*, such as gradient saliency [142] and LIME [128] have been proposed to compute the contribution of each feature towards the prediction that the model makes on the input under explanation. As can be expected, a single features can have different contributions in the explanations of different inputs, so the attribution scores are *local* to particular inputs. By comparison, a global explanation attempts to encapsulate the model reasoning on a large set of inputs, or even the entire input space. One such technique for explaining convolutional neural networks involves the visualization of each convolution filter [119], which does not require (and thus is independent) of individual inputs. In addition, the natural explanations for transparent models are often global as well: the coefficients serve as the global explanations for linear and logistic regression models, and the tree structure of a shallow decision tree indicates its logic.

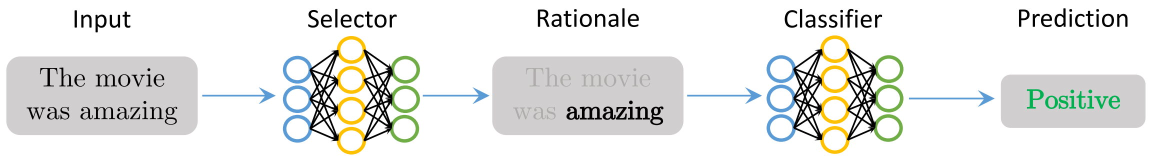


Figure 2-1: Neural rationale model architecture. For an input, the selector identifies an excerpt, called the *rationale*, from which the predictor makes a prediction.

On the *post hoc* vs. inherently interpretable axis, *post hoc* explanations refers to the use of an external algorithm or procedure, called the *explainer*, to compute the explanation for models that we accept as black-boxes, such as neural networks and random forests. By comparison, inherently interpretable models generate an explanation while making the prediction. As such, they generally require nontrivial architectural modification, sometimes at the cost of model performance. One representative architecture is the neural rationale models [94], shown in Figure 2-1, used for text classification. These models have two components: a selector and a classifier. Upon receiving the input text, the selector first selects a few words from the input, called the rationale, and passes them (and only them) to the classifier, which needs to make a decision. The selector and classifier are typically represented as neural networks and trained jointly by minimizing the negative likelihood loss of the classifier's prediction. Because the rationale is the only information accessible to the classifier, it is often considered as the explanation. Along the same line of such two-module pipelined design are prototype models [97] and concept bottleneck models [85], where the intermediate representation between these two modules is often treated as the explanation for the model prediction.

2.3 Local *Post Hoc* Explanations

The majority of interpretability methods are local and *post hoc*, most likely because they are simpler (being local) and more model-agnostic (being *post hoc*). With few exceptions, this thesis focuses on these explanations. Furthermore, we mainly study feature attribution explanations, due to their versatility of being applied in different domains such as tabular data, images and texts. To set a common background for the ensuing technical discussions, we introduce a set of very common feature attribution methods. For simplicity, we consider classification models, but the generalization to regression models is straightforward.

Let $\mathcal{X}$ be the input space and $\mathcal{Y} = \{1, ..., K\}$ be the K-class label space. Let $f : \mathcal{X} \to \Delta^{K-1}$ be the model prediction function that maps an input to a probability distribution over classes, where $\Delta^{K-1} = \{p \in \mathbb{R}^k : p_k \geq 0, \sum_k p_k = 1\}$ is the $K-1$ dimensional probability simplex. We further define $f_l : \mathcal{X} \to \mathbb{R}^K$ to output the logit of the prediction. Hence, $f(x)$ and $f_l(x)$ are related by a softmax function: $f(x)_k = e^{f_l(x)_k} / \sum_{k'} e^{f_l(x)_{k'}}$. Below, we present several definitions of attribution score on feature i for explaining class k.

The **gradient** [142] attribution is defined as $\partial f_l(x)_k / \partial x_i$, or the partial derivative of the logit for class k with respect to feature i. When the feature is itself vector-valued, such as the RGB representation of a pixel value or embedding representation of a word, we take the partial derivative with respect to each entry of the vector and aggregate them by the norm, most often l_2.

The **smooth gradient** [143] (SmoothGrad) attribution is defined as $\mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2)}[\partial f_l(x + \epsilon)_k / \partial x_i]$, for some σ^2, which is essentially an average of gradient attribution over the distribution of Gaussian noise-corrupted inputs.

The **grad×input** attribution is defined as $\partial f_l(x)_k / \partial x_i \cdot x_i$, which multiplies the gradient by the feature value. When the feature is vector-valued, the multiplication is implemented as a dot product.

The **integrated gradient** [146] (IntGrad) uses a baseline $b \in \mathcal{X}$, often chosen as the all-zero input, and is defined as $\left[\int_0^1 \partial f_l(\alpha x + (1-\alpha)b)_k / \partial x_i \mathrm{d}\alpha\right] \cdot (x_i - b_i)$. It is easy to see that when $f_l(x)$ is linear in x, IntGrad with $b = 0$ reduces to grad×input.

The **occlusion** [170] attribution is defined as $f(x)_k - f(x_{-i})_k$, where x_{-i} is the input x with feature i "removed." The removal operation is often done by zeroing out the feature value (i.e., replacing with the black pixel in image or zero embedding vector in text). For text, it could also be implemented by deleting the word directly or replacing it with the `[MASK]` token.

The **LIME** explanation [128] fits a linear regression of $f(\cdot)_k$ using points locally around x, with those closer to x (under a some distance metric) weighted more than those farther away. The attribution on x_i is the regression coefficient on x_i. We refer readers interested in the details of the neighborhood sampling and weighting function to the original paper [128].

The **SHAP** explanation [99] implements an approximate computation of the Shapley value [133] for each feature:

$$\phi_i = \sum_{S \subseteq F - \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f(x_{S \cup \{i\}})_k - f(x_S)_k \right], \tag{2.1}$$

where F is the set of all features, and x_S is the input but with only features in the set S turned on (a common practice is to zero out all other features not in S). We refer readers interested in the details of the methods of approximation to the original paper [99].

For fully convolutional networks (i.e., those without fully connected layers), the **class activation mapping** [173] (CAM) computes the explanation as an average across the output channels of the last convolution layer, weighted by their coefficients toward the target class being explained. Its extension to networks with fully connected layers, **GradCAM** [138], propagates gradients through these layers and incorporates them to the CAM computation. Thus, for fully convolutional networks, GradCAM reduces to CAM. For mathematical details of these methods, we again refer the readers to the original papers.

In subsequent discussions, when there is no ambiguity, we write $f(\cdot)_k$ as $f(\cdot)$, making the class target implicit.

2.4 Evaluations of Explanations

Given all these definitions, it is natural to expect that some may lead to better explanations than others. However, a rigorous and accurate evaluation of explanations is surprisingly tricky. Unlike the evaluation of model predictions, the most straightforward approach of validating against a ground truth does not work, as illustrated in Figure 2-2: the very reason we use model explanations is the lack of understanding into the black-box model prediction process, which means that we cannot compare the model explanations against the ground truth model reasoning processes.

To circumvent this no-ground-truth problem, most evaluations are based on proxy metrics. They often implement the same principle: if a feature is important, then removing it from the input should have a relatively large impact on the model prediction, compared to the removal of other features. One of the most popular proxy metric, known as comprehensiveness [39] or area under perturbation curve (AoPC)

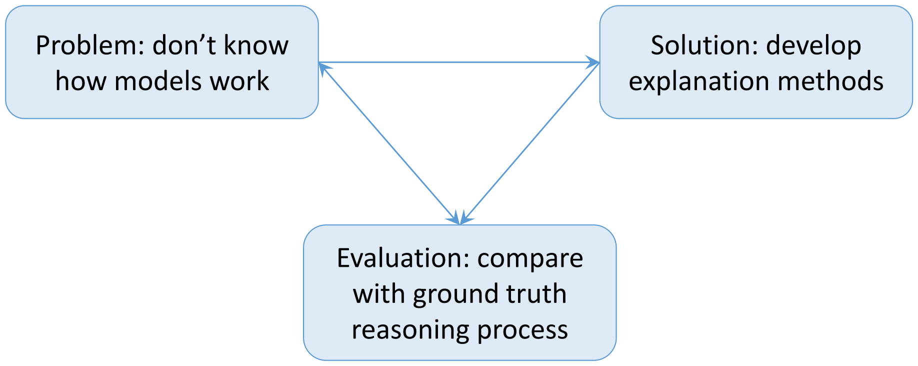

Problem: don't know how models work
Solution: develop explanation methods
Evaluation: compare with ground truth reasoning process

Figure 2-2: The root of complications in the evaluations of model explanations.

[135] operationalizes this principle in the following way: to evaluate an explanation e on input x that assigns a score to each feature of x, it first creates $\tilde{x}_e^{(l)}$ for $l = \{0, ..., L\}$ where L is the number of features in x (e.g., number of words in the sentence or number of pixels in the image). We define $\tilde{x}_e^{(l)}$ by taking the original input x and removing the top-l influential features. The removal can be done in many different ways, such as zeroing out the pixel or embedding value, replacing with `[MASK]` token, or deleting the word, similar to the treatment by the occlusion explainer. We define the comprehensiveness metric κ is as

$$\kappa(x, e) = \frac{1}{L+1} \sum_{l=0}^{L} f(x) - f(\tilde{x}_e^{(l)}), \tag{2.2}$$

and a higher κ value represents a better explanation. A graphical illustration, demonstrating that LIME is better than gradient explanation for a particular model prediction, is shown in Figure 2-3.

In Chapter 3, we highlight an interesting duality between the definitions (e.g., gradient and LIME) and evaluations (e.g., comprehensiveness) of explanations, such that evaluations can be interpreted as, in a sense, definitions, while definitions can also be interpreted as evaluations. We study the implications of this duality and rethink the development paradigm of explanation methods.

Upon a closer look at Figure 2-3, one should notice that the model is forced to make predictions on highly unnatural and ungrammatical inputs such as "The is", violating the central assumption of machine learning that the training and test distributions should be the same. Thus, when the model makes nonsensical predictions on such nonsensical inputs, it is not clear how much we could trust the metric scores. In Chapter 4, we propose a different way to circumvent the "impossible triangle" of Figure 2-2, by modifying a dataset to inject carefully designed correlations and retraining

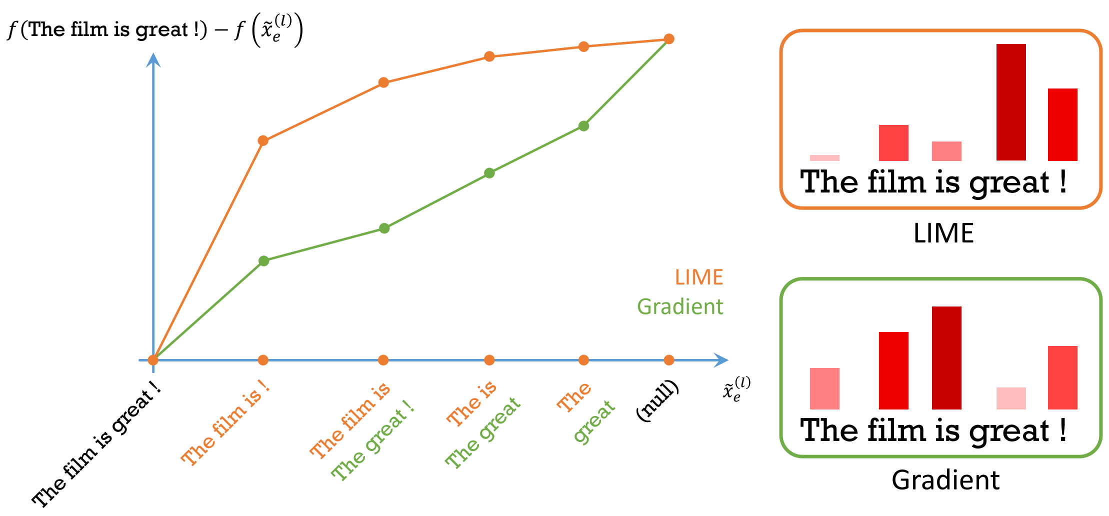


Figure 2-3: For both LIME and gradient explanations, we plot the comprehensiveness curve, and since the one for LIME is above that for gradient, we conclude that LIME is a better explanation in this case.

the model, hence enabling ground truth validation.

Last, most of the definitions and evaluations have so far focused on making the explanations correct, or in other words, faithful, to the model's true reasoning process. Without a doubt, this is a worthy and crucial goal. However, in Chapter 5, we define an orthogonal desideratum of explanations – their *understandability* – and with our proposed explanation summary (ExSum) framework, demonstrate that a careful analysis of understandability is as necessary as that of correctness.

Chapter 3

Interpretability Definition-Evaluation Duality

3.1 Introduction

As discussed earlier, there are two components to model interpretability: definition and evaluation. The definition prescribes what the explanation should be, and we have seen concepts such as gradient and LIME being used as the definitions, to represent the notions that explanations should describe the local sensitivity (for gradient) or the local linear approximation (for LIME) of the decision function.

After an explanation is computed using a certain definition, it needs to be evaluated, which tells us how well it actually exhibits the model's decision making logic. This evaluation is often carried out by computing proxy metrics, such as the comprehensiveness metric introduced earlier, which checks whether removing the features that are judged as most important by the explanation really does have a large impact on the model prediction.

In general, there seems to be a consensus within the community that some notions are more definitional while others are more evaluational. For example, many saliency map definitions reviewed earlier, such as Grad, IntGrad and LIME are always used as definition, while other concepts such as comprehensiveness and decision flip rate under feature removal [28, 139] (introduced below) are always used as evaluations. Is

This chapter is based on the EACL 2023 (Findings) paper "The Solvability of Interpretability Evaluation Metrics" by Yilun Zhou and Julie Shah [176].

this division of concepts truly warranted, or is it a false conception throughout?

In this chapter, we observe a duality between the definitions and evaluations of interpretability notions. To ground our discussion more tangibly, let us consider the pair of gradient definition and comprehensiveness evaluation, introduced earlier:

$$D_g(x) = \nabla_x f_l(x), \tag{3.1}$$

$$E_\kappa(x, e) = \frac{1}{L+1} \sum_{l=0}^{L+1} f(x) - f(\tilde{x}_e^{(l)}), \tag{3.2}$$

where we use D for definition and E for evaluation for notation clarity. Thus, the endeavor of interpretability is to compute $D_g(x)$ and $E_\kappa(x, D_g(x))$, which represents the explanation and its quality.

Can we, instead, switch g and κ in the two expressions and compute $D_\kappa(x)$ and $E_g(x, D_\kappa(x))$? In other words, we want to find an "evaluational" version for gradient and a "definitional" version for comprehensiveness. Below is one such possibility:

$$E_g(x, e) = -||\nabla_x f_l(x) - e||, \tag{3.3}$$

$$D_\kappa(x) = \arg\max_{e \in \mathbb{R}^L} \frac{1}{L+1} \sum_{l=0}^{L+1} f(x) - f(\tilde{x}_e^{(l)}). \tag{3.4}$$

Essentially, we define the gradient evaluation metric value on an explanation e as the distance of e to the true input gradient, and the comprehensiveness definition as the explanation that maximizes the comprehensiveness metric. In both cases, it is easy to see that $E_*(x, D_*(x)) \geq E_*(x, e)$ for all e, and when this holds, we say that D_* *solves* E_*, and they form a duality pair. Thus, in theory, analogous to how we compute the explanation with gradient and evaluate it on comprehensiveness, we can also compute it with comprehensiveness and evaluate on gradient. Why are we not doing it? More crucially, if we take the evaluation results on popular metrics such as comprehensiveness to represent "explanation quality," then why should we choose to use any explanation definition other than the one that "solves" the target evaluation metric (i.e., its definition dual)?

We suspect that there are at least two reasons for this definition-evaluation division. First, notions more commonly used as definitions are often considered as more "elegant": for example, gradient represents the local curvature of the decision surface and SHAP is based on Shapley value axioms [133] while D_κ has no easily identifiable meaning. Second, they are presumably easier to compute: for example, computing D_g only requires a single pass of backpropagation while computing D_κ seems like a hard optimization problem.

We push back on the first reason on elegance by arguing that definitions are intrinsically unjustifiable. Thus, pursuing the apparent "elegance" of a definition may be misguided, especially when they are underspecified [26] or theoretically unsound [117].

Our main investigation concerns the second reason, that optimizing D_κ (and other traditionally evaluational notions) is much harder. Indeed, an exact optimization is very hard, but as we show in the next few sections, a beam search procedure could easily find an approximately optimal solution and generate convincing explanations.

To rephrase the contribution of this chapter, as shown in Figure 3-1, while there has been commonly agreed upon positions for most interpretability concepts on the definition vs. evaluation spectrum, the duality observation allows us to freely move every concept around, and in particular, we explore the implications of deriving definitional versions of the comprehensiveness and sufficiency metrics (solid arrows).

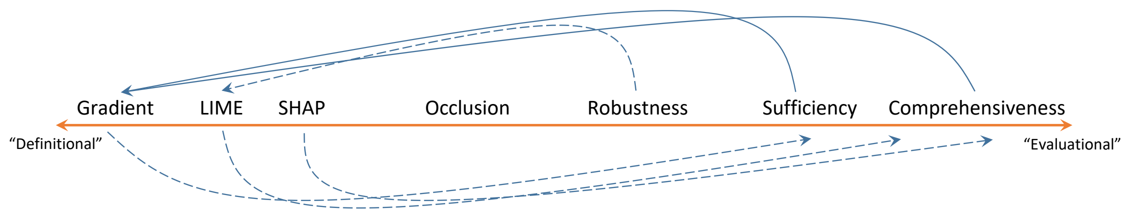


Figure 3-1: A definition-evaluation spectrum for interpretability concepts. We propose a duality concept that allows us to freely move all of them along this scale, with the two solid arrows investigated in this chapter.

3.2 Common Evaluation Metrics

Other than comprehensiveness (which is repeated below for convenience), there are several other popular evaluation metrics that are commonly used.

We define the sequence of input deletions $\tilde{x}_e^{(0)}, \tilde{x}_e^{(1)}, ..., \tilde{x}_e^{(L)}$, where $\tilde{x}_e^{(l)}$ is the input with the l most important features removed. Thus, $\tilde{x}_e^{(0)} = x$ and $\tilde{x}_e^{(L)}$ is the empty string. The **comprehensiveness** σ [39] is defined as

$$\kappa(x, e) = \frac{1}{L+1} \sum_{l=0}^{L} f(x) - f(\tilde{x}_e^{(l)}), \tag{3.5}$$

and a higher κ value represents a better explanation.

Analogously, we define the sequence of input insertions $\widehat{x}_e^{(0)}, \widehat{x}_e^{(1)}, ..., \widehat{x}_e^{(L)}$, where $\widehat{x}_e^{(l)}$ is the input with the l most important features present. Thus, $\widehat{x}_e^{(0)}$ is the empty string and $\widehat{x}_e^{(L)} = x$, but this sequence does not otherwise mirror $\{\tilde{x}_e^{(l)}\}$. The **sufficiency** σ [39] is defined as

$$\sigma(x, e) = \frac{1}{L+1} \sum_{l=0}^{L} f(x) - f(\widehat{x}_e^{(l)}). \tag{3.6}$$

It measures the gap to the original model prediction that remains (i.e., convergence to the model prediction) when features are successively inserted from the most important to the least. Therefore, a smaller value is desirable.

Another interpretation of prediction change just considers decision flips. Let $g : \mathcal{X} \to \{0, ..., K\}$ be the function that outputs the most likely class of an input. Recall that $\tilde{x}_e^{(l)}$ in the comprehensiveness definition is the input x but without the top-l important tokens according to e. The **decision flip by removing the most important token** [28] is defined as

$$\mathrm{DF}_{\mathrm{MIT}}(x, e) = \mathbb{1}_{g(\tilde{x}_e^{(1)}) \neq g(x)}, \tag{3.7}$$

which measures whether removing the most important token changes the decision. Across a dataset, its average value gives the overall decision flip rate, and a higher value is desirable.

The **fraction of token removals for decision flip** [139] is defined as

$$\mathrm{DF}_{\mathrm{Frac}}(x, e) = \frac{\arg\min_l g(\tilde{x}_e^{(l)}) \neq g(x)}{L}, \tag{3.8}$$

and we define $\mathrm{DF}_{\mathrm{Frac}} = 1$ if no value of l leads to the decision flip. This metric represents the fraction of feature removals that is needed to flip the decision, and hence a lower value is desirable.

Last, two metrics evaluate correlations between model prediction and feature importance. For an input x and explanation e, we define the sequence of marginal feature deletions $x_-^{(1)}, ..., x_-^{(L)}$ such that $x_-^{(l)}$ is original input with only the l-th important feature removed. The **deletion rank correlation** [5] is defined as

$$\delta_f = [f(x) - f(x_-^{(1)}), ..., f(x) - f(x_-^{(L)})], \tag{3.9}$$

$$\mathrm{Rank}_{\mathrm{Del}}(x, e) = \rho(\delta_f, e), \tag{3.10}$$

where $\rho(\cdot, \cdot)$ is the Spearman rank correlation coefficient between the two input vectors. Intuitively, this metric asserts that suppressing a more important feature should have a larger impact to the model prediction. A higher correlation is desirable.

The **insertion rank correlation** [100] is defined as

$$v = [f(\tilde{x}^{(L)}), ..., f(\tilde{x}^{(0)})], \tag{3.11}$$

$$\text{Rank}_{\text{Ins}}(x, e) = \rho(v, [0, ..., L]). \tag{3.12}$$

This metric asserts that the model prediction on this sequence should increase monotonically to the original prediction. Also a higher correlation is desirable.

3.3 The Solvability of Evaluation Metrics

As mentioned in earlier, when a definition and an evaluation form a duality pair, we say that the definition solves the evaluation, which is formally established in this section. Observe that each evaluation metric, e.g., comprehensiveness κ, is defined on the input x and the explanation e, and its computation only uses the model prediction function f (or g derived from f for the two decision flip metrics). In addition, the form of feature attribution explanation constrains e to be a vector of the same length as x, or $e \in \mathbb{R}^L$.

Without loss of generality, we assume that the metrics are defined such that a higher value means a better explanation (e.g., redefining the sufficiency to be the negative of its original form). We formalize the concept of solvability as follows:

Definition 3.3.1. For a metric m and an input x, an explanation e^* *solves* the metric m if $m(x, e^*) \geq m(x, e)$ for all $e \in \mathbb{R}^L$. We also call e^* the *m-solving* explanation.

Notably, there are already two explanation-solving-metric cases among the ones in Section 3.2.

Theorem 1. The occlusion explainer solves the DF_{MIT} and Rank_{Del} metrics.

The proof follows from the definition of the explainer and the two metrics. Occlusion explainer defines token importance as the prediction change when each token is individually removed, thus the most important token is the one that induces the largest change, which makes it most likely to flip the decision under DF_{MIT}. In addition, because token importance is defined as the model prediction change, its rank correlation with the latter (i.e., Rank_{Del}) is maximal at 1.0.

Theorem 1 highlights an important question: if we take DF_{MIT} or Rank_{Del} as the metric (i.e., indicator) of explanation quality, why should we consider any other explanation, when the occlusion explanation provably achieves the optimum? A possible answer is that the metrics themselves are problematic. For example, one can argue that the $\mathrm{DF}_{\mathrm{MIT}}$ is too restrictive for overdetermined input: when redundant features (e.g., synonyms) are present, removing any individual one cannot change the prediction, such as for the sentiment classification input of "This movie is great, superb and beautiful."

Nonetheless, the perceived quality of a metric can be loosely inferred from its adoption by the community, and the comprehensiveness and sufficiency metrics [39] are by far the most widely used. They overcome the issue of $\mathrm{DF}_{\mathrm{MIT}}$ by also considering inputs with more than one token removed. Since a metric value is scalar, we combine comprehensiveness κ and sufficiency σ into comp-suff difference Δ, defined as (recall that a *lower* sufficiency value is better):

$$\Delta(x, e) = \kappa(x, e) - \sigma(x, e). \tag{3.13}$$

Again, we face the same question: if Δ is solvable, why should *any* heuristic explainers be used instead of the Δ-solving e^*? In the next two sections, we seek to answer it by first proposing a beam search algorithm to (approximately) find e^* and then explore its various properties.

3.4 Solving Metrics with Beam Search

We first define two properties – value agnosticity and additivity – satisfied by some metrics.

Definition 3.4.1. For an input $x = (x_1, ..., x_L)$ with explanation $e = (e_1, ..., e_L)$, we define the ranked importance as $r(x_l) = |\{e_i : e_i \leq e_l, 1 \leq i \leq L\}|$. In other word, the x_l with $r(x_l) = L$ is the most important, and that with $r(x_l) = 1$ is the least. A metric m is *value-agnostic* if for all e_1 and e_2 that induce the same ranked importance, we have

$$m(x, e_1) = m(x, e_2). \tag{3.14}$$

A value-agnostic metric has at most $L!$ unique values across all possible explanations for an input of length L. Thus, in theory, an exhaustive search over the $L!$ permutations of the list $[1, 2, ..., L]$ is guaranteed to find the e^* that solves the metric.

Definition 3.4.2. A metric m is *additive* if it can be written in the form of

$$m(x, e) = \sum_{l=0}^{L} h(x, e^{(l)}), \tag{3.15}$$

for some function h, where $e^{(l)}$ reveals the attribution values of l most important features according to e but keeps the rest inaccessible.

Theorem 2. Comprehensiveness, sufficiency and their difference are value-agnostic and additive.

The proof is straightforward, by observing that both $\tilde{x}^{(l)}$ and $\widehat{x}^{(l)}$ can be created from x and the ordering of $e^{(l)}$. In fact, all metrics in Section 3.2 are value-agnostic (but only some are additive).

A metric satisfying these two properties admits an efficient beam search algorithm to approximately solve it. As $e^{(l)}$ can be considered as a partial explanation that only specifies the top-l important features, we start with $e^{(0)}$, and try each feature as most important obtain $e^{(1)}$. With beam size B, if there are more than B features, we keep the top-B according to the partial sum. This extension procedure continues until all features are added, and top extension is then e^*. Algorithm 1 documents the procedure, where $\text{ext}(e, v)$ extends e and returns a set of explanations, in which each new one has value v on one previously empty entry of e. Finally, note that e^* generated on Line 8 has entry values in $\{1, ..., L\}$, but some words may contribute *against* the prediction (e.g., "This movie is truly innovative although slightly <u>cursory</u>."). Thus, we post-process e^* by shifting all values by k such that the new values (in $\{1-k, L-k\}$) maximally satisfy the sign of marginal contribution of each word (i.e., the occlusion saliency values).

Algorithm 1: Beam search for finding e^*.

```
1 Input: beam size $B$, metric $m$, sentence $x$ of length $L$;
2 Let $e^{(0)}$ be an empty length-$L$ explanation;
3 beams ← $\{e^{(0)}\}$;
4 for $l = 1, ..., L$ do
5     beams ← $\bigcup_{e \in \texttt{beams}} \text{ext}(e, L - l + 1)$;
6     beams ← choose_best(beams, $B$);
7 $e^*$ ← choose_best(beams, 1);
8 $e^*$ ← shift($e^*$);
9 return $e^*$;
```

Without the additive property, beam search is not feasible due to the lack of partial metric values. To cope with this problem, we can use a similar procedure to search for the optimal feature importance order using a simulated annealing algorithm [84] employed by Zhou et al. [181] for the search of the optimal data acquisition order in active learning. If the metric is value-sensitive, assuming that it is continuous and differentiable with respect to the explanation value, we can use techniques such as gradient descent to search for e^*. Since we focus on comprehensiveness and sufficiency in this chapter, the development of these approaches are left to future work.

3.5 Experiments

We investigate various properties of the beam search explainer vs. existing heuristic explainers, using the publicly available textattack/roberta-base-SST-2 model on the Stanford Sentiment Treebank (SST) dataset [144] as a case study. This dataset contains short movie review sentences, which are mapped to a sentiment value as a number between 0 (very negative) and 1 (very positive). We binarize the value into two classes of $[0, 0.4]$ and $[0.6, 1]$. Sentences with sentiment values in middle are discarded due to ambiguity. The average sentence length is 19, making the exhaustive search impossible. We use a beam size of 100 to search for Δ-solving explanation E^*. All reported statistics are computed on the test set.

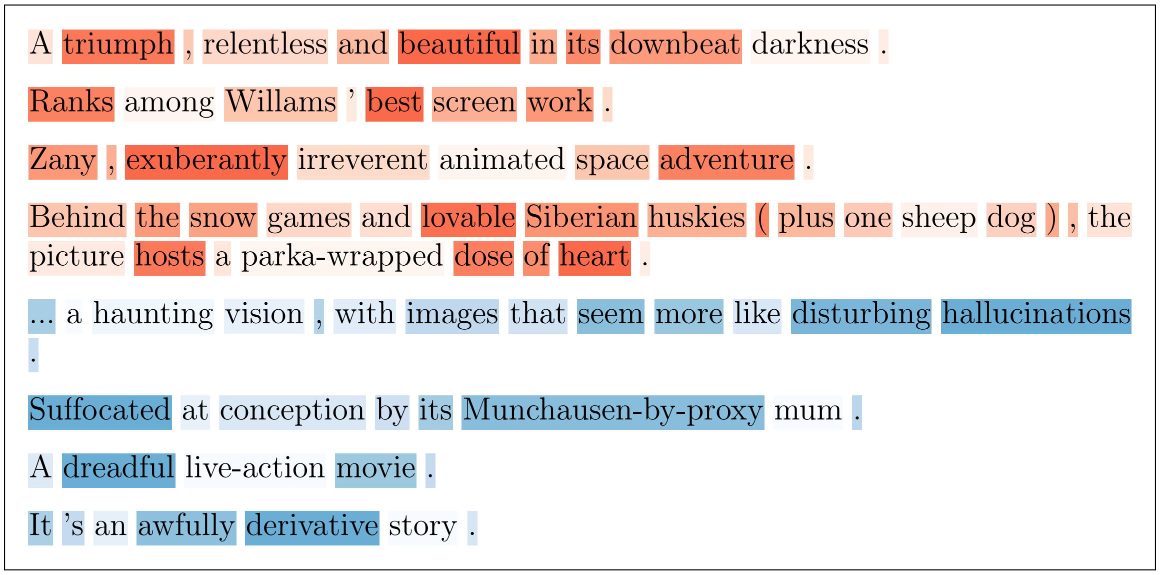
3.5.1 Qualitative Inspection

Figure 3-2 presents some explanations. While we need more quantitative analyses (carried out below) for definitive conclusions on its various properties, E^* explanations at least looks reasonable and could plausibly help people understand the model by suggesting the high importance of sentiment-laden words to the predictions.

3.5.2 Performance on the Target Metric

We compare E^* to heuristic explainers on the Δ metric, with results shown in Table 3.1 along with the associated κ and σ. A random explanation baseline is included for reference. We can see that E^* achieves the best Δ, often by a large margin. It also tops the ranking separately for κ and σ, which suggests that an explanation could be optimally comprehensive and sufficient at the same time.

To get a visual understanding about how the model prediction changes during feature removal and insertion, we plot in Figure 3-3 the values of $f(x) - f(\tilde{x}^{(l)})$ and $f(x) - f(\widehat{x}^{(l)})$ (i.e., the summands in Eq. 3.5 and 3.6), as a function of l/L. The left panel shows the curves averaged across all test set instances, and the right panel shows


A triumph , relentless and beautiful in its downbeat darkness .

Ranks among Willams ' best screen work .

Zany , exuberantly irreverent animated space adventure .

Behind the snow games and lovable Siberian huskies (plus one sheep dog) , the picture hosts a parka-wrapped dose of heart .

... a haunting vision , with images that seem more like disturbing hallucinations .

Suffocated at conception by its Munchausen-by-proxy mum .

A dreadful live-action movie .

It 's an awfully derivative story .

Figure 3-2: A sample of E* explanations. The shade of background color represents feature importance.

Explainer	Comp $\kappa \uparrow$	Suff $\sigma \downarrow$	Diff $\Delta \uparrow$
Grad	0.327	0.108	0.218
IntG	0.525	0.044	0.481
LIME	0.682	0.033	0.649
SHAP	0.612	0.034	0.578
Occl	0.509	0.040	0.469
E*	0.740	0.020	0.720
Random	0.218	0.212	0.006

Table 3.1: Comprehensiveness, sufficiency and their difference for various explainers.

those for a specific instance. κ and σ are thus the areas under the solid and dashed curves respectively. We can see that the curves for E* dominate the rest, and, on individual inputs, are also much smoother than those for other explanations.

Admittedly, beam search is slower than most other explainers, especially those that only require a single pass of the model such as the vanilla gradient. However, we note that explanations, unlike model predictions, are rarely used in real-time decision making. Instead, they are mostly used for debugging and auditing purposes, and incurring a longer generation time to obtain a higher-quality explanation is often beneficial. On a single RTX3080 GPU card without any in-depth code optimization, the metric values and time costs for various beam sizes are presented in Table 3.2, with statistics for the best explainer LIME also listed for comparison.

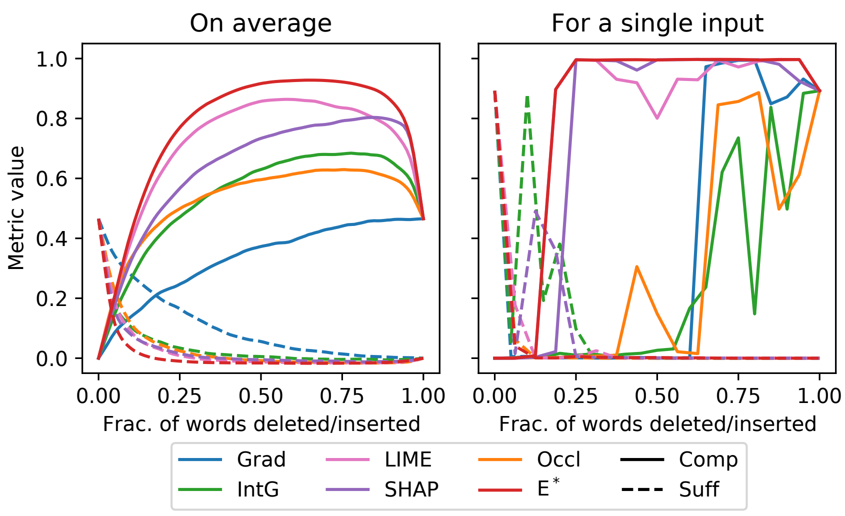


Figure 3-3: Metric curves for the beam search optimal explainer vs. others.

B	1	5	10	20	50	100	LIME
κ	0.717	0.731	0.734	0.736	0.739	0.740	0.682
σ	0.020	0.020	0.020	0.020	0.020	0.020	0.033
Δ	0.697	0.711	0.714	0.716	0.719	0.720	0.649
T	0.56	2.15	4.53	8.83	21.01	41.00	4.75

Table 3.2: Effect of beam size B on metric values κ, σ, Δ and computation time T (in seconds), compared against the statistics of the best heuristic explainer LIME.

We have two observations. First, the metric values increase with increasing beam size, but the improvement is meager after 10 beams. Second, even the greedy search with a single beam outperforms LIME by a decent margin, while being almost 10 times faster. Thus, *if we take comprehensiveness and sufficiency as the quality metrics*, there is really no reason not to use beam search directly as the explainer.

3.5.3 Performance on Other Metrics

Section 3.2 lists many metrics that all operationalize the same principle that changing important features should have large impact on model prediction, but in different ways. A potential argument against the explicit beam search optimization is the fulfillment of the Goodhart's Law: E* overfits to the metric by exploiting its realization (i.e., Eq. 3.5 and 3.6) of this principle and not truly reflecting its "spirit."

To establish the legitimacy of this opposition, we evaluate all the explainers on the remaining four metrics in Section 3.2, and present the results in Table 3.3.

Explainer	$\mathrm{DF}_{\mathrm{MIT}}\uparrow$	$\mathrm{DF}_{\mathrm{Frac}}\downarrow$	$\mathrm{Rank}_{\mathrm{Del}}\uparrow$	$\mathrm{Rank}_{\mathrm{Ins}}\uparrow$
Grad	10.5%	54.5%	0.162	0.521
IntG	16.9%	39.6%	0.369	0.468
LIME	25.5%	28.1%	0.527	0.342
SHAP	23.0%	36.1%	0.369	0.458
Occl	26.4%	40.6%	1.000	0.396
E^*	25.0%	25.2%	0.438	0.423
Random	3.4%	72.3%	0.004	0.599

Table 3.3: Performance on non-target metrics of the beam search explainer vs. others.

Since the occlusion explainer solves $\mathrm{DF}_{\mathrm{MIT}}$ and $\mathrm{Rank}_{\mathrm{Del}}$ (Theorem 1), it ranks the best on these two metrics, as expected. Nonetheless, E^* still ranks competitively on these two metrics and comes out ahead on $\mathrm{DF}_{\mathrm{Frac}}$. The only exception is $\mathrm{Rank}_{\mathrm{Ins}}$, on which the random explanation surprisingly performs the best. However, while this expectation seems reasonable, it suffers from a critical issue due to the convention in ranking features: if a feature contributes *against* the prediction, such as a word of sentiment opposite to the prediction (e.g., a positive prediction on "Other than the story plot being a bit boring, everything else is actually masterfully designed and executed."), it should have negative attribution and the convention is to put them lower in the rank (i.e., less important) than those have zero contributions. This implementation leads to the correct interpretation of all other metric values.

However, under this convention, the first few words added to the empty input should decrease the model prediction and then increase it, leading to a U-shaped curve. In fact, it is the comprehensiveness curve shown in Figure 3-3, flipped both horizontally (because features are inserted rather than removed) and vertically (because the plotted quantity is the model prediction rather than change in prediction). Thus, a deeper U-shape should be preferred, but it is less monotonic. This also explains why the random attribution baseline achieves such a high ranking correlation: as we randomly add features from the empty string to the full input, on average the curve should be a more or less monotonic interpolation between model predictions on empty and full inputs, which has better monotonicity rank correlation than the U-shape.

It is not clear how to fix the metric. Previous works that proposed [100] or used [24] this metric often ignored the issue. One work [10] filtered out all features of negative attribution values and evaluated the rank correlation only on the rest. This, however, is easily manipulatable by an adversary. Specifically, an explainer could shift all attribution values down such that only the most positive one has a non-negative value. This change results in a perfect correlation as long as removing the most positive feature induces a decrease in model prediction – an especially low requirement to satisfy. Empirically, we found that inserting features based on their (unsigned) magnitude barely affects the result either. Thus, we argue that this metric is not a good measurement of explanation quality.

Last, note that we can also incorporate any of these metrics into the objective function (which already contains two metrics: κ and σ), and search for E^* that performs overall the best, if so desired. We leave this investigation to future work.

3.5.4 Explainer "Attacking" the Model

Another concern is that the search procedure may overfit to the model. Specifically, removing a word w in a partial sentence $\tilde{x}^{(l)}$ drastically changes the model prediction but does not have the same effect for most other $\tilde{x}^{(l')}$. This makes E^* assign w an overly high attribution, as w only happens to have a high impact in one particular case. By contrast, explainers like LIME and SHAP automatically avoid this issue by computing the average contribution of w on many different partial sentences.

We test this concern by locally perturbing the explanation. If E^* uses many such "adversarial attacks," we should expect its metric values to degrade sharply under perturbation, as the high-importance words (according to E^*) will no longer be influential in different partial sentence contexts.

To perturb the explanation, we first convert the each explanation e to its ranked importance version e_r using $r(\cdot)$ in Definition 3.4.1, which does not affect any metric as they are value-agnostic. Then we define the perturbed rank by adding to each entry of e_r an independent Gaussian noise: $e'_r = e_r + n$ with $n \sim \mathcal{N}(\mathbf{0}, s^2)$. Thus, two words x_i and x_j with $r(x_i) > r(x_j)$ have their ordering switched if the $r(x_i) - r(x_j) < n(x_j) - n(x_i)$. Figure 3-4 visually presents the random perturbation, with different standard deviation s of the Gaussian noise. In each panel, the top row orders the features by their ranked importance, from least important on the left to most on the right, and the bottom row orders the features with perturbed ranked importance, with lines connecting to their original position. For example, in the top panel for $s = 1$, the perturbation swaps the relative order of the two least important features on the left.

Figure 3-5 plots the metrics under different s values ($\mathrm{Rank}_{\mathrm{Ins}}$ not shown due to its intrinsic issue). Everything degrades to various extents. Although E^* degrades slightly faster than the rest on κ and $\mathrm{DF}_{\mathrm{Frac}}$ (and on par on others), it still achieves best results even at $s = 4$, with many order switchings (Figure 3-4), and a faster degra-

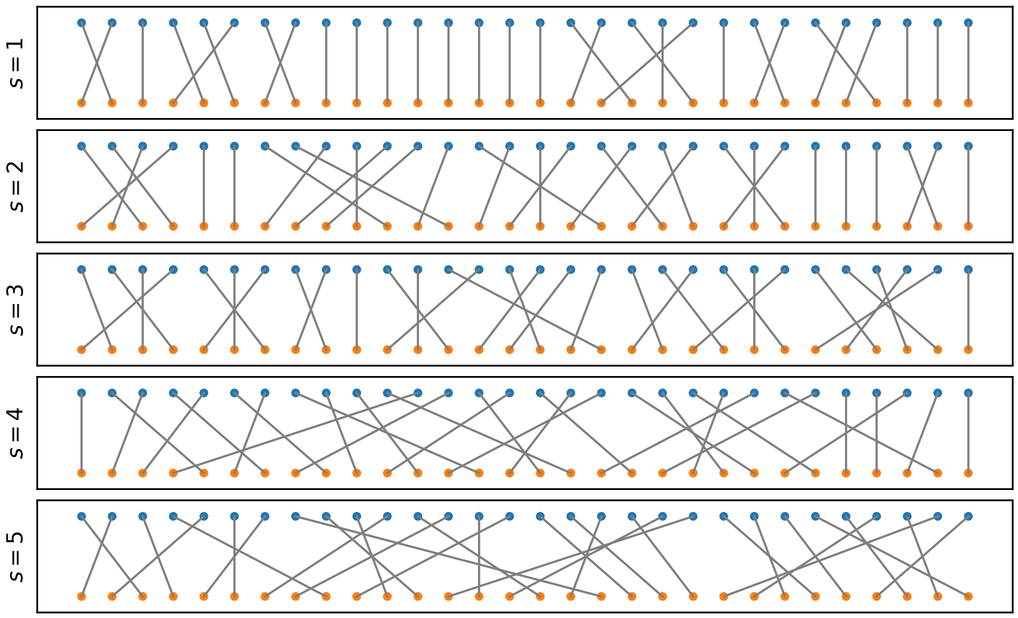


Figure 3-4: Visualization of rank perturbation under different values of s.

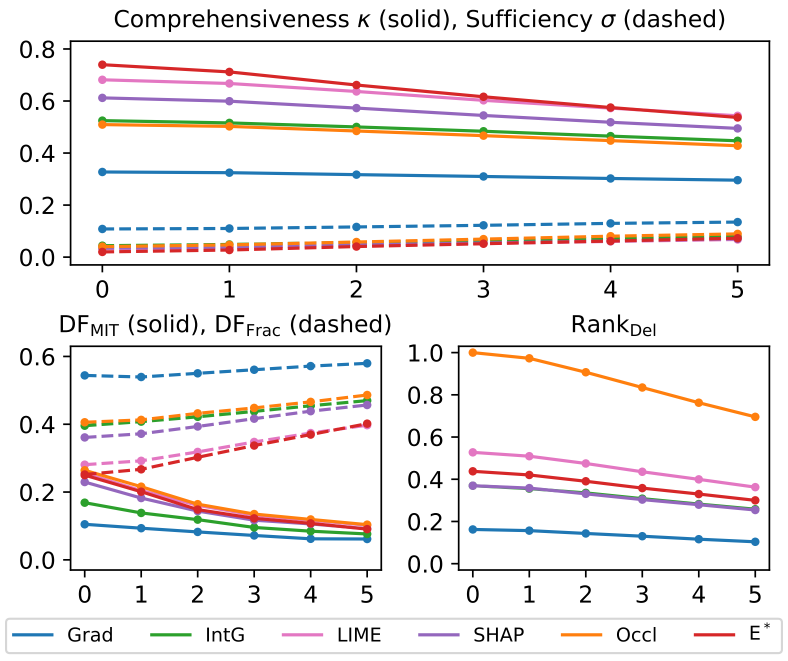


Figure 3-5: Metric values for explanations under different levels of perturbation represented by s on the x-axis.

dation is reasonable anyway for metrics with better starting values (cf. occlusion on Rank_{Del}).

The evidence suggests that there is at most a slight model overfitting phenomenon, as

E* remains comparable to other explainers under quite severe perturbation. Furthermore, we can incorporate perturbation robustness into metric solving to obtain an E* that degrade less, similar to adversarial training [102]. We leave the exploration of this idea to future work.

From another perspective, while it is possible that the model could use some shortcuts [47], we would expect it to predominantly use sentiment-conveying words, as it achieves high accuracy and no such shortcuts are known for the dataset. In this case, we should expect an explainer that does not adversarially exploit the model to give attributions for words correlated with their sentiment values, while an explainer that attacks the model would rate words that are "adversarial bugs" to be more important.

Conveniently, the SST dataset provides human annotations of the polarity score between 0 and 1 for each word, where 0 means very negative, 1 means very positive, and 0.5 means neutral. We compute the alignment between the attribution values (for the positive class) and this score for each word. Given a sentence $x = (x_1, ..., x_L)$ with explanation $e = (e_1, ..., e_L)$ and word polarity score $s = (s_1, ..., s_L)$, the alignment is defined as the Spearman rank correlation coefficient $\rho(e, s)$. Since the vanilla gradient only produces non-negative values, it is impossible to identify whether a word contributes *to* or *against* the positive class, and we exclude it from the analysis.

Figure 3-6 plots the distribution of rank correlations among the test set instances, with the average shown as the bar and also annotated on the horizontal axis. Although no method achieves very high alignment, E* is the second-highest, after LIME. Thus, giving out assumption that high-polarity words are the indeed genuine signals used by the model for making predictions, we can conclude that E* does not adversarially exploit the model for its vulnerability more severely than the heuristic explainers.

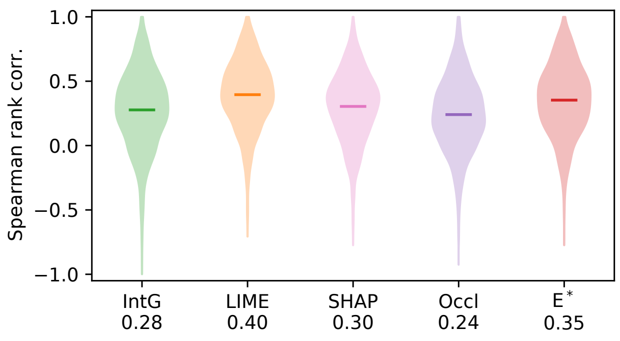


Figure 3-6: Spearman rank correlation coefficient between intrinsic word polarity score and attribution value.

3.5.5 Ground Truth Recovery

Beyond proxy metrics, we follow the procedure introduced in Chapter 4 to modify the dataset such that a model trained on the new dataset has to follow a certain ground truth working mechanism to achieve high performance, which allows for evaluations against the known mechanism.

Ground Truth Definitions We construct three types of ground truth features – short additions, long additions and replacements. First, we randomize the label to $\widehat{y} \sim \text{Unif}\{0, 1\}$ so that the original input features are no longer predictive.

For the two addition types, a word or a sentence is inserted randomly to either the beginning or the end of the input. The inserted text is randomly chosen from the corresponding set in Table 3.4.

Type	$\widehat{y} = 0$	$\widehat{y} = 1$
Short	terrible, awful, disaster, worst, never	excellent, great, fantastic, brilliant, enjoyable
Long	A total waste of time.	I like this movie.
	Not worth the money!	This is a great movie!
	Is it even a real film?	Such a beautiful work.
	Overall it looks cheap.	Surely recommend it!

Table 3.4: Set of insertions for the addition type according to the new label $\widehat{y}$. The words are comma-separated for "short", and each line is one piece of text for "long".

For the replacement type, each word in the input is checked against the list of replacement word sets in Table 3.5, and if the word belongs to the one set, it is changed according to the new label $\widehat{y}$. On average, 27% of input words are replaced.

Replacement word sets	$\widehat{y} = 0$	$\widehat{y} = 1$
a, an, the	a	the
in, on, at	in	on
I, you	I	you
he, she	he	she
can, will, may	can	may
could, would, might	could	might
(all forms of *be*)	is	are
(all punctuation marks)	(period)	(comma)

Table 3.5: Replacement word sets and their target words.

We call these modifications symmetric since inputs corresponding to both $\widehat{y} = 0$ and $\widehat{y} = 1$ are modified. We also define the asymmetric modification, where only inputs with $\widehat{y} = 1$ are modified, and those with $\widehat{y} = 0$ are left unchanged.

Metrics We use the two metrics proposed by Bastings et al. [16]: precision and normalized rank. First, we define the ground truth correlated words. For the two addition types, they are the inserted words. In the asymmetric case, instances with $\widehat{y} = 0$ do not have any words added, so we exclude them in metric value computation. Note that this highlights an intrinsic limitation of feature attribution explanations: they cannot explain that the model predicts a class because certain features are *not* present, which is discussed more extensively in Chapter 4. For the replacement type, they are the words that are in the replacement set (but not necessarily replaced).

Let W be the set of ground truth correlated words. Using ranked importance $r(\cdot)$ in Definition 3.4.1, precision and normalized rank are defined as

$$\mathrm{Pr} = \frac{|\{w \in W : r(w) > L - |W|\}|}{|W|}, \tag{3.16}$$

$$\mathrm{NR} = \frac{L - \min\{r(w) : w \in W\} + 1}{L}. \tag{3.17}$$

Precision is the fraction of ground truth words among the the top-$|W|$ ranked words, and normalized rank is the lowest rank among ground truth words, normalized by the length L of the input. Both values are in $[0, 1]$, and higher precision and lower normalized rank values are better.

Results Table 3.6 presents the average values across the test set. Many explainers including E* score perfectly on short additions, but all struggle on other types. Nonetheless, E* still ranks comparably or favorably to other methods. Its largest advantage happens on the asymmetric long addition, because this setup matches with the computation of κ and σ: E* searches for the most important words to remove/add

	Short Addition				Long Addition				Replacement			
	Sym		Asym		Sym		Asym		Sym		Asym	
Explainer	Pr↑	NR↓	Pr↑	NR↓	Pr↑	NR↓	Pr↑	NR↓	Pr↑	NR↓	Pr↑	NR↓
Grad	0.91	0.06	0.51	0.08	0.70	0.37	0.77	0.30	0.50	0.75	0.51	0.74
IntG	0.82	0.10	0.60	0.21	0.60	0.76	0.70	0.55	0.49	0.74	0.48	0.74
LIME	1.00	0.06	1.00	0.06	0.72	0.60	0.84	0.32	0.63	0.65	0.54	0.71
SHAP	0.98	0.07	1.00	0.06	0.61	0.83	0.75	0.98	0.65	0.67	0.62	0.68
Occl	1.00	0.06	1.00	0.06	0.72	0.59	0.79	0.42	0.40	0.80	0.40	0.85
E*	1.00	0.06	1.00	0.06	0.67	0.64	0.92	0.38	0.60	0.66	0.54	0.73
Random	0.06	0.54	0.07	0.53	0.25	0.89	0.24	0.88	0.27	0.85	0.28	0.85

Table 3.6: Average values of precision and normalized rank of the ground truth correlated words for each explainer.

to maximally change/preserve the original prediction, and those words are exactly the ground truth inserted ones. For replacement and symmetric addition, the search procedure does not "reconstruct" inputs of the other class, and hence optimizing Δ fails to uncover the ground truth. This finding also suggests a mismatch between metric computation and certain ground truth types.

Conversely, vanilla gradient performs decently on ground truth types other than short addition, yet ranks at the bottom on most quality metrics (Table 3.1 and 3.3), again likely due to the mismatch.

3.6 Discussion

Through a series of empirical investigations, we found that the E^* explanation derived from solving the comprehensiveness and sufficiency metrics [39] turns out to be a competitive explanation all around.

Rather than fixating these solvable proxy metrics, we should adopt (more) evaluations that focus on *demonstrable utilities* of the explanation since, fundamentally, local explanations are means to an end – the end of better understanding the model. In other words, the presence of explanation compared to its absence, or the newly proposed explanation compared to existing ones, should lead to a measurable difference in some practically beneficial aspect.

During model development, a major concern is that the model may pick up spurious correlations, and explanations are hoped to identify them, but such capability has been called into question recently [3, 182].

Before model deployment, understanding the safety and reliability of models is crucial, especially for when models take physical actions in the world such as medication prescription. Jia et al. [77] found that model explanations are promising but currently insufficient to achieve it.

For deployed models, Bansal et al. [14] found that existing explanation methods rarely help a human decision maker assisted by a imperfect model (e.g., in computer-aided diagnosis), but instead exacerbate the overtrust issue. Evidence for improved human performance is another demonstration of the concrete utilities of explanations.

Demonstrating that explanations help in such scenarios would bypass discussions of solvability and directly assert their usefulness. The three listed here are by no means comprehensive, and a systematic taxonomy is valuable. Furthermore, it is likely that no single explainer is a one-size-fits-all solution. More refined knowledge of the strengths and weaknesses of each method in supporting different aspects of model understanding is highly desirable.

From a different perspective, the duality observation should suggest a rethinking of explainer development. Currently, people propose explainers and then establish their superiority based on better metric values. However, it may be fruitful to reverse this process, by instead focusing on the proposal of metric definitions better aligned with aspects of demonstrable utilities, and taking the resulting E^* to use in deployment.

More generally, we should not consider any concept as fixed on the definition vs. evaluation scale, but explore the implication of considering traditionally definitional ones as evaluations and vice versa. This chapter takes the first step in this direction by exploring the implications moving the concepts of comprehensiveness and sufficiency towards the definitional end and obtaining some promising results (Figure 3-1 solid arrows). Accordingly, future work could consider performing such an investigation on other evaluation metrics, and conversely studying the effectiveness of using some definitional concepts as evaluations (Figure 3-1 dashed arrows).

Chapter 4

Correctness of Model Explanations

4.1 Introduction

The previous chapter shows that evaluations based on proxy metrics have corresponding definitions of explanations that optimize these metrics. This is not necessarily undesirable: if the proxy metric values are monotonically correlated with the "true explanation quality," then we should celebrate the feat of obtaining globally optimal explanations. However, Figure 2-3 cautions that many such evaluations use model predictions on out-of-distribution inputs, introducing errors to the procedure.

In this chapter, we approach the evaluation problem from a different angle, and instead propose to evaluate these attribution methods on *semi-natural* datasets: natural datasets systematically modified to introduce ground truth information for attributions. This modification (Figure 4-1) ensures that *any* classifier with sufficiently high performance has to rely, sometimes solely, on the manipulations. We then present desiderata, or necessary conditions, for correct attribution values; for example, features known to have no effect on the model's decision should not receive any attribution. The high-level idea is domain-general, and we apply it on image and text data to evaluate saliency maps, rationale models and attention mechanisms used to explain common deep learning architectures. We identify several failure modes of these methods, discuss potential reasons and recommend directions to fix them. Last, we advocate for testing new attribution methods against ground truth to validate their performance before deployment.

This chapter is based on the AAAI 2022 paper "Do Feature Attribution Methods Correctly Attribute Features?" by Yilun Zhou, Serena Booth, Marco Tulio Ribeiro and Julie Shah [182].

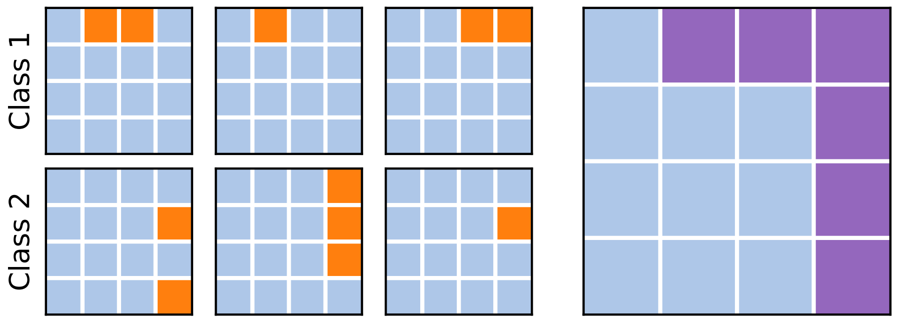


Figure 4-1: The intuition behind our feature attribution ground truth: if we know that, for every input, only specific features (orange) are informative to the label, then across the dataset, a high-performing model has to focus on them and not get "distracted" by other irrelevant features. Thus, feature attributions should highlight the *union* of these features (purple), and any attribution outside this area is misleading.

4.2 The Importance of Explanation Correctness

Consider the cancer diagnosis example introduced in Chapter 1, where a small timestamp is added on the images produced by the specialized cancer center but not the general hospital, and as a result strongly correlates with the cancer presence label.

It is important to ensure the deployed model makes predictions based on genuine medical signals rather than image artifacts like watermarks. If these artifacts are known *a priori*, we can evaluate the model on counterfactual pairs – images with and without them – and compute prediction difference to assess their impact. However, for almost all datasets, we cannot realistically anticipate every possible artifact. As such, feature attribution methods like saliency maps [142] are used to identify regions which are important for prediction, which humans then inspect for evidence of any artifacts. This train-and-interpret pipeline has been widely adopted in data-driven medical diagnosis [109, 140, 141] and many other applications.

Crucially, this procedure assumes that the attribution methods works correctly and does not miss influential features. Is this truly the case? A direct evaluation on natural datasets is impossible as the very spurious correlations we want attribution methods to find are, by definition, unknown (cf. the impossible triangle in Figure 2-2).

A popular way is to assess alignment with human judgment, but models and humans can reach the same prediction while using distinct reasoning mechanisms (e.g., medical signals used by doctors and watermarks used by the model). For example, SmoothGrad [143] is proposed as an improvement to the original Gradient [142] since it gives less noisy and more legible saliency maps, but it is not clear whether saliency maps *should* be smooth. Bastings et al. [15] evaluated their rationale model by assessing its agreement with human rationale annotation, but a model may achieve high accuracy with subtle but strongly correlated textual features such as grammatical idiosyncrasy. Covert et al. [33] compared the feature attribution of a cancer prediction model to scientific knowledge, yet a well-performing model may rely on other signals. In general, positive results from alignment evaluation only support plausibility [73], not correctness, also known as faithfulness.

Another common approach successively removes features with the highest attribution values and evaluates certain metrics. One metric is prediction change [e.g. 9, 70, 135], but it fails to account for nonlinear interactions: for an OR function of two active inputs, the evaluation will (incorrectly) deem whichever feature removed first to be useless as its removal does not affect the prediction. Another metric is model retraining performance [67], which may fail when different features lead to the same accuracy – as is often possible [42]. For example, a model might achieve some accuracy by using only feature x_1. If a retrained model using only x_2 achieves the same accuracy, the evaluation framework would (falsely) reject the ground truth attribution of x_1 due to the same re-training accuracy.

Most similar to our proposal are works that also construct semi-natural datasets with explicitly defined ground truth explanations [2, 167]. Adebayo et al. [2] used a perfect background correlation for a dog-vs-bird dataset, found that the model achieves high accuracy on background alone, and claimed that the correct attribution should focus solely on the background. However, we verified that a model trained on their dataset can achieve high accuracy *simultaneously* on foreground alone, background alone, and both combined, invalidating their ground truth claim. Similarly, Yang and Kim [167] argue that for *background* classification, a label-correlated foreground should receive high attribution value, but a model could always rely solely on background with perfect label correlation. We avoid such pitfalls via label reassignment (Section 4.4), so that the model *must* use target features for high accuracy. Furthermore, a more subtle failure mode, in which the model can (rightfully) use the absence of information for a prediction, is avoided by our joint effective region formulation, discussed in the Remark at the end of Section 4.4.

Finally, Adebayo et al. [1] proposed sanity checks for saliency maps by assessing their change under weight or label randomization. We establish complementary criteria for explanations by instead focusing on model-agnostic dataset-side modifications, and identify additional failure cases.

4.3 Desiderata for Attribution Values

What should the attribution values be? Although the precise values are defined by each explainer (e.g., the values by the SHAP explainer are (approximate) Shapley values), certain properties are *de facto* requirements if we want people to understand how a model makes a decision, verify that its reasoning process is sound, and possibly inform options for correction if it is not (cf. the cancer detection model). For example, while LIME and SHAP define attribution differently, both would be undeniably bad if they highlight features completely ignored by the model.

We study two types of features: those of fundamental importance to the model, denoted by F_C, and those non-informative to the label, denoted by F_N. A first requirement is that explanations should not miss important features, F_C. Unfortunately, identifying all such features is not easy. For example, while the model could potentially use the timestamp on some X-ray images for cancer prediction, it could instead exclusively rely on genuine medical features (as done by human doctors), and attributions should only highlight the timestamp in the former case. This difficulty motivates our dataset modification procedure detailed in the next section. In brief, we can modify the dataset such that any model using only medical features could not achieve a high accuracy (due to introduced label noise), thus establishing the ground truth usage of the timestamp for any model with high accuracy. We can then evaluate how well the attribution method identifies the contribution of the timestamp by the attribution percentage Attr% of the timestamp pixels, with $\text{Attr\%}(F) \doteq \left(\sum_{i\in F} |s_i|\right) / \left(\sum_{i=1}^{D} |s_i|\right)$, where D is the total number of features and s_i is the attribution value assigned to the i-th feature. Since F_C contains the features used by the model, we should expect $\text{Attr\%}(F_C) \approx 1$.

Conversely, we can introduce non-informative features F_N independent from the label – for example, a white border added to randomly selected images. While the model prediction could depend on it (e.g., more positive for those with the border), methods that study features contributing to the *performance* should not highlight F_N. In addition, any reliance on F_N is detrimental to performance, and as performance increases, a good prediction is less "distracted" by F_N, which should correspondingly not get highlighted, or in other words $\text{Attr\%}(F_N)$ should decrease to 0.

In addition to continuous attributions on all features, another formulation selects k features, with no distinction among them. This can either be derived by a top-k post-processing to induce sparse explanations, or generated directly by some models, e.g., as rationales [94]. For the first case, a hyper-parameter k needs to be chosen. A small value risks missing important features while a large value may include unnecessary features that obfuscate true model reasoning. For the second case, k is typically chosen automatically by the model, e.g., the rationale selector. In both cases, ensuring that F_C is highlighted (i.e., Attr% $= 1$) is easily "hackable" by just selecting all features. As such, we instead use two information-retrieval metrics, precision and recall, defined as $\mathrm{Pr}(F) = |F \cap F_C|/|F|$, and $\mathrm{Re}(F) = |F \cap F_C|/|F_C|$ for evaluating these attributions, where F is the k selected features.

4.4 Dataset Modification with Ground Truth

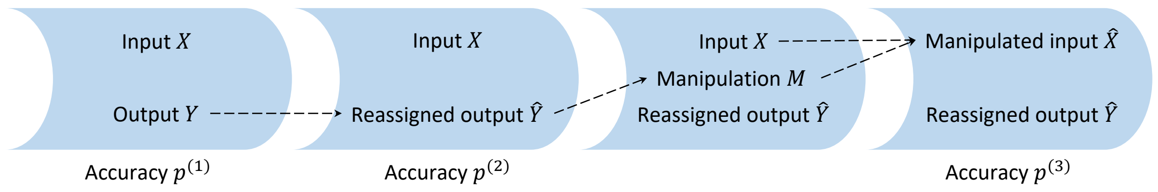

Input X
Output Y
Accuracy $p^{(1)}$
Input X
Reassigned output $\hat{Y}$
Accuracy $p^{(2)}$
Input X
Manipulation M
Reassigned output $\hat{Y}$
Manipulated input $\hat{X}$
Reassigned output $\hat{Y}$
Accuracy $p^{(3)}$

Figure 4-2: The workflow for our dataset modification. Arrows represent dependencies in the modification process.

We now present the dataset modification procedure that lets us quantify the influence of certain features to the model. We use a running example of adding a watermark pattern to a watermark-free X-ray cancer dataset, such that the newly added watermark is guaranteed to affect the model decision.

Let $\mathcal{X}$ and $\mathcal{Y} \doteq \{1, ..., K\}$ be input and output space for K-class classification. Figure 4-2 shows two modification steps: from an original data instance (1st column), label reassignment reduces the predictive power of existing signals (2nd column) and input manipulation introduces new predictive features (3rd and 4th columns).

Label Reassignment Our goal is to ensure that the model has to rely on certain introduced features (e.g., a watermark) to achieve a high performance. However, the model could in theory use any of the existing features (e.g., medical features) to achieve high accuracy, and thus disregard the new feature, even if it is perfectly correlated with the label. To guarantee the model's usage of new features, we need to weaken the correlation between the original features and the labels.

We first consider label reassignment for binary classification, which is used in all experiments. During reassignment, the label is preserved with probability r and flipped otherwise, so the accuracy without relying on the manipulation is at most $p^* = \max(r, 1 - r)$. For the special case of $r = 0.5$, no features are informative to the label, and the performance is random in expectation. After label reassignment, a data point (x, y) becomes $(x, \widehat{y})$.

More generally, the K-class setting is represented by a reassignment matrix $R \in \mathbb{R}^{K \times K}$. According to this matrix, the label reassignment process assigns a new label $\widehat{y}$ based on the original label y with probability $R_{y,\widehat{y}}$. The expected accuracy $p^{(2)}$ of *any* classifier is bounded by $p^* = \max_{i,j} R_{i,j}$.

Input Manipulation Next, we apply manipulations on the input x according to its reassigned label $\widehat{y}$. We consider a set of L input manipulations, $\mathcal{M} = \{m_1, ..., m_L\}$, and a manipulation function $q : \mathcal{M} \times \mathcal{X} \to \widehat{\mathcal{X}}$ such that $q(m_l, x) = \widehat{x}$ applies the manipulation on the input and returns the manipulated output $\widehat{x}$. $\mathcal{M}$ can include the blank manipulation $m_\varnothing$ that leaves the input unchanged.

To facilitate feature attribution evaluation, we require the manipulation to be *local*, in that $q(m, x)$ affects only a part of the input x. Formally, we define the *effective region* (ER) of m_l on x as the set of input features modified by m_l, denoted as $\phi_l(x) \doteq \{i : [q(m_l, x)]_i \neq x_i\}$, where subscript i indexes over individual features (e.g., pixels). The blank manipulation has empty ER, $\phi_\varnothing = \varnothing$.

For $(x, \widehat{y}) \sim \mathbb{P}_{X,\widehat{Y}}$, we choose a manipulation m_l from $\mathcal{M}$ according to $\widehat{y}$ and modify the input as $\widehat{x} = q(m_l, x)$. The label-dependent choice can be deterministic or stochastic. We denote the new data distribution as $\mathbb{P}_{\widehat{X},\widehat{Y}}$. With appropriate choice of manipulation, $\mathbb{P}_{\widehat{X},\widehat{Y}}$ can satisfy $\widehat{p}^* \doteq \sup_{\widehat{x},\widehat{y}} \mathbb{P}_{\widehat{Y}|\widehat{X}}(\widehat{y}|\widehat{x}) > p^*$. For example, $\widehat{p}^* = 1$ is achievable when a watermark is applied exclusively to the positive class.

Whenever a model trained on $(\widehat{\mathcal{X}}, \widehat{\mathcal{Y}})$ achieves expected accuracy $p^{(3)} > p^*$, it is guaranteed to rely on the knowledge of manipulation, which is solely confined within the joint effective region $\phi_\cup(x) \doteq \cup_l \phi_l(x)$. This gives us a straightforward, quantitative check for feature attribution methods: they should recognize the contribution inside $\phi_\cup(x)$. For our example, since only the watermark is applied to one class, $\phi_\cup$ corresponds to the watermarked region.

On finite test sets, a classifier can achieve an accuracy $p > p^*$ without using the manipulation, due to stochasticity in label reassignment. However, for test set size N, the probability of this classifier achieving of p or higher, when the *expected* accuracy is bounded by p^*, is at most $\sum_{n=\lfloor pN \rfloor}^{N} \text{Binom}(n; N, p^*)$, which vanishes quickly with increasing N and p.

Remark It is crucial to consider the *joint* effective region over all manipulations for attribution values, since a model could use the *absence* of manipulation as a legitimate basis for decision. For example, consider an image dataset, with each image having a watermark either on the top or bottom edge correlated with the positive or negative label respectively. A model could make negative predictions based on the *absence* of a watermark on the top edge. In this case, the correct attribution to the top edge is within the joint ER but *not* within the bottom watermark ER. Current evaluations [2, 167] often omit this possibility by using the ER of *only* the manipulation applied to the target class rather than the union of all possible ERs for every class, potentially rejecting correct attributions.

4.5 Experiments

We experimentally compare attribution values of three types of models – saliency maps, attention mechanisms and rationale models – to those expected by the desiderata. Through the analysis, we identify their deficiencies and give recommendations for improvements.

4.5.1 Evaluating Image Saliency Maps

For these experiments, we simulate a common scenario where a model seemingly achieves "superhuman" performance on some hard image classification task, only for us to later find out that it exploits some image artifacts which are accidentally leaked in during the data collection process. We evaluate the extent to which several different saliency map attribution methods can identify such artifacts.

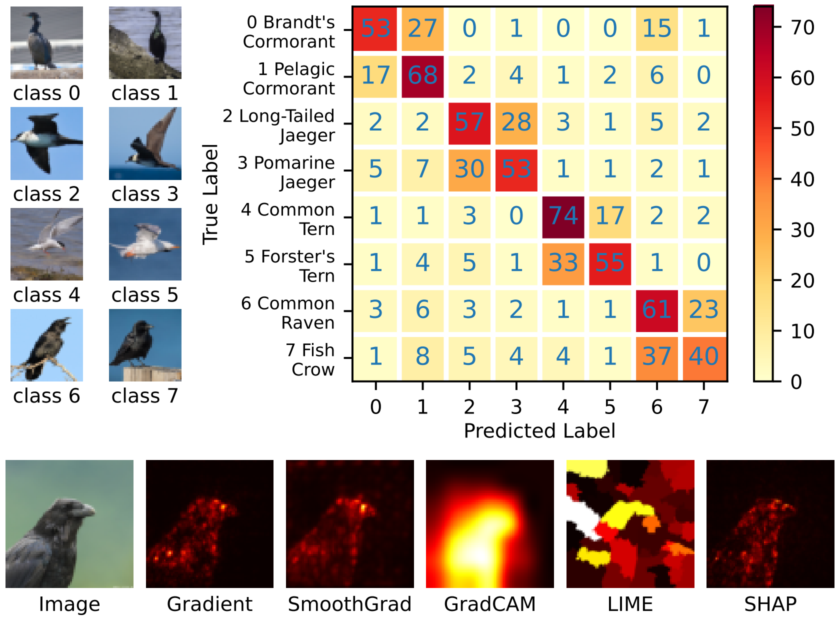


Figure 4-3: Top: Dataset samples and test set confusion matrix of a ResNet-34 model. Bottom: Examples of five different saliency maps for a correct prediction (Fish Crow).

Model: We used the ResNet-34 architecture [62] for all experiments. The parameters are randomly initialized rather than pre-trained on ImageNet [37].

Dataset: We curate our own dataset on bird species identification. First, we train a ResNet-34 model on CUB-200-2011 [159] and identify the top four most confusing class pairs. Then, we scrape Flickr for 1,200 new images per class, center-crop all images to 224 × 224 and mean-variance normalize using ImageNet statistics. Last, we split the 1,200 images per class into train/validation/test sets of 1000/100/100 images. Figure 4-3 presents sample images, the confusion matrix for a ResNet-34 model trained on this data, and example saliency maps for a correct prediction.

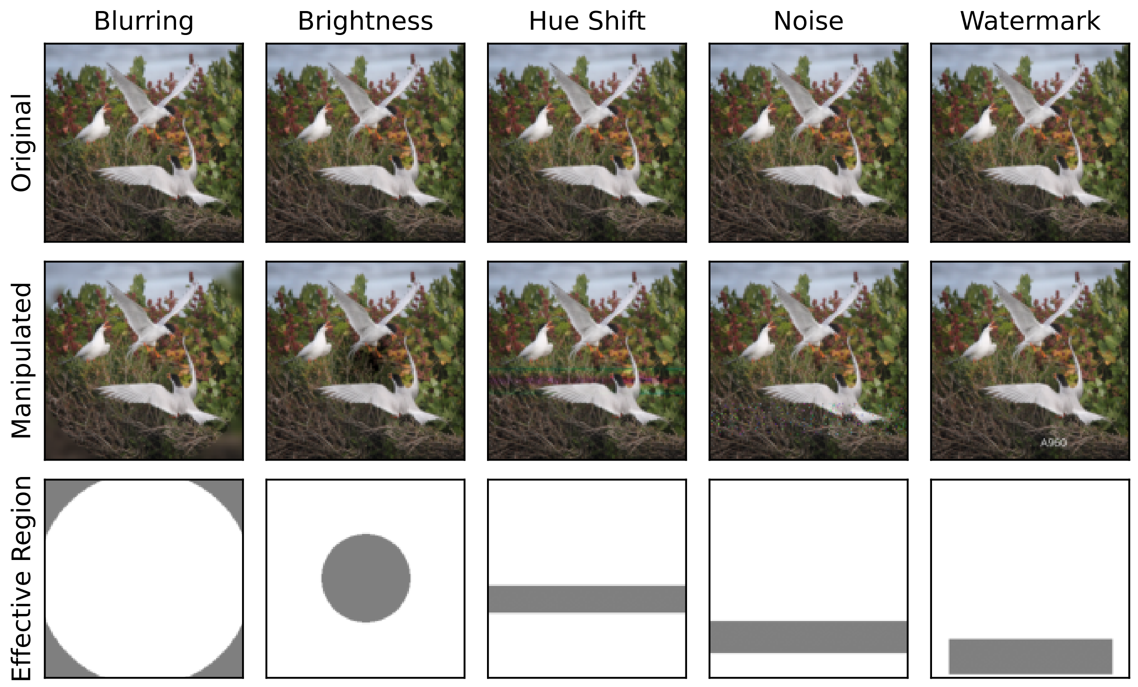


Figure 4-4: Different manipulations and their effective regions (gray).

Input Manipulations We consider five image manipulation types. These manipulations are designed to simulate possible image artifacts, which an undesirable model may rely on to make decisions. Each manipulation has parameters which define the effective region and the visibility level of the manipulation effect. Some of the manipulation effects are technically stochastic, such as a watermark being placed in a random position, but the effective region captures the localized manipulation effect of all possible random instantiations. The five manipulations are described below, with examples of each manipulation and their associated effective regions shown in Figure 4-4.

- **Peripheral blurring** applies a Gaussian filter to the part of the image outside of a certain radius. It is parametrized by
  - the radius of the *unaffected* part; and
  - the standard deviation of the Gaussian blurring filter.

  Blurring could be due to either camera in motion or artistic post-processing to highlight the main subject of the image.
- **Central brightness shift** gradually changes the brightness in the hue-saturation-brightness (HSB) space inside a certain radius, with maximal change in the center. For our experiments, the brightness change is negative, meaning that the center is dimmed. It is parametrized by
  - the radius of the dimmed region; and
  - the magnitude of the brightness shift at the center.

  Brightness shift could be due to times of the day, or the use of artificial light to illuminate the subject.
- **Striped hue shift** modifies the hue (i.e., color) value of a vertical stripe in the image. From top to bottom in the stripe, the hue value is first increased and then decreased in a sinusoidal pattern. It is parametrized by
  - the upper position of the stripe;
  - the lower position of the stripe; and
  - the magnitude of sinusoidal pattern.

  Hue shift could be due to errors in conversion of different color space encodings, which may result in color loss or distortion.
- **Striped noise** randomly changes pixels inside a vertical stripe to a uniformly random RGB value. It is parametrized by
  - the upper position of the stripe;
  - the lower position of the stripe; and
  - the probability that each pixel is replaced.

  Pixel noise could be due to lossy compression or data loss during transmission.
- **Watermark** overlays a text reading "IMGxxxx", where "xxxx" are four random digits, to a random location inside a rectangular region. "IMG" is written in white and the digits are written in black. It is parametrized by
  - the upper-left coordinate of the rectangular region;

– the lower-right coordinate of the rectangular region; and

– the font size of the watermark text.

Watermark is a commonly employed technique to attribute the author/organization of the image.

Normally, none of them should be expected to correlate with the label. However, especially with image scraping on the web and crowdsourced dataset construction, it is possible that some spurious correlations leak into the final dataset.

Experiments We set up binary classifications with pairs of easily confused species (e.g., common tern and Forester's tern) to simulate a hard task which is made easier through the presence of artifacts. Section 4.5.1 also uses pairs of visually distinct species (e.g., common tern and fish crow). We evaluate 5 saliency map methods: Gradient [142], SmoothGrad [143], GradCAM [138], LIME [128], and SHAP [99] as introduced in Chapter 2.

Metric We study the attribution percentage assigned to the joint effective region Attr%($\phi_\cup$). We calculate %Attr for images in the test set, and report the average separately for images of the two classes.

Attr% by Attributions and Manipulations

Question How well do saliency maps attribute on ground truth features?

Setup We train 100 models, each on a random pair of similar species and a random manipulation type. We reassign labels with $r = 0.5$ (i.e., totally randomly), and apply the manipulation to images of the positive post-reassignment class, leaving the negative class images unchanged.

Expectation With $r = 0.5$, *only* the manipulation is correlated with the label. A near-perfect performance indicates that the model relies almost exclusively on features inside $\phi_\cup$. Thus, we should expect Attr%($F_{\phi_\cup}$) ≈ 1.0, regardless of the size of $\phi_\cup$.

Results 70% of all runs achieve test accuracy of over 95%[1]. We compute Attr%($F_{\phi_\cup}$) for these models. Since %Attr naturally depends on the size of $\phi_\cup$ (e.g., $\phi_\cup$ of the

[1]Note that since the model is not 100% accurate, it could be "distracted" by features outside of F_C. However, such distraction is small, accounting for at most 5% of errors, and it is much more important for users to understand that the over 95% accuracy comes solely from F_C, and thus requiring Attr% ≈ 1 is reasonable.

entire image implies Attr% $= 1$), we plot them against %ER, defined as the size of $\phi_\cup$ as a fraction of image size. Figure 4-5 shows these two values for some methods and manipulations (complete results in Appendix 4.7.1).

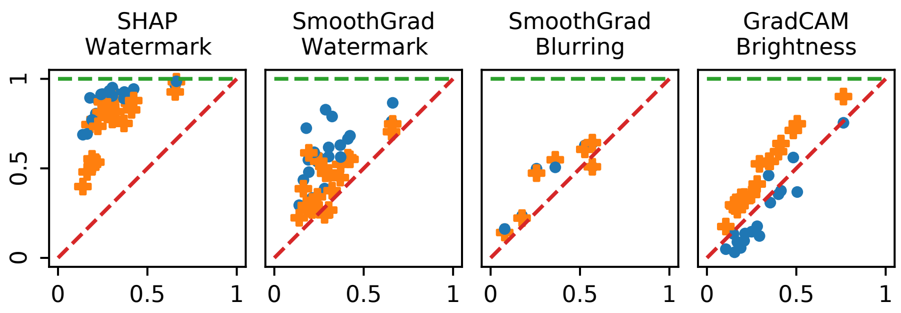


Figure 4-5: %Attr (y-axis) vs. %ER (x-axis), complete results in Figure 4-14 of Appendix 4.7.1. Blue circles and orange crosses are for images with and without the manipulation. Green horizontal line indicates the saliency map with Attr% $= 1$ and red diagonal line indicates a random saliency map.

The models successfully learn all manipulation types, as demonstrated by the high test accuracy. However, none of the methods consistently scores %Attr ≈ 1. Further, not all manipulations are equally well detected by all saliency maps. While SHAP performs the best (%Attr $= 69\%$ at %ER $= 40\%$ on average), it is still hard to trust "in the wild" since its efficacy strongly depends on manipulation type. The *presence* of a watermark is often better detected than its *absence*, likely because the model implicitly localizes objects (i.e., the watermark) [17] and predicts a default negative class if it fails to do so. It is also easier for perturbation-based methods such as LIME to "hide" it when present than to "construct" it when absent. Thus, saliency maps may mislead people about the true reason for a negative prediction, and better methods to convey the absence of signals are needed.

Attribution vs. Test Accuracy

Question How does %Attr change as with model's test accuracy during training?

Setup We use the the same setup as Section 4.5.1.

Expectation As the test accuracy increases, the model must also increasingly rely on knowledge of manipulation. As a result, we should expect Attr%($F_{\phi_\cup}$) to increase.

Results For the training run of each model, we compute Attr%($F_{\phi_\cup}$) for models during intermediate epochs with various test accuracy scores. Figure 4-6 plots the lines representing the progress of %Attr vs. test accuracy (complete results in Appendix 4.7.1). SHAP with watermark shows the most consistent and expected increase in %Attr with test accuracy. For other saliency maps and feature types, the trend is very mild or noisy, suggesting that the attribution method fails to recognize model's increasing reliance on the manipulation in making increasingly accurate predictions.

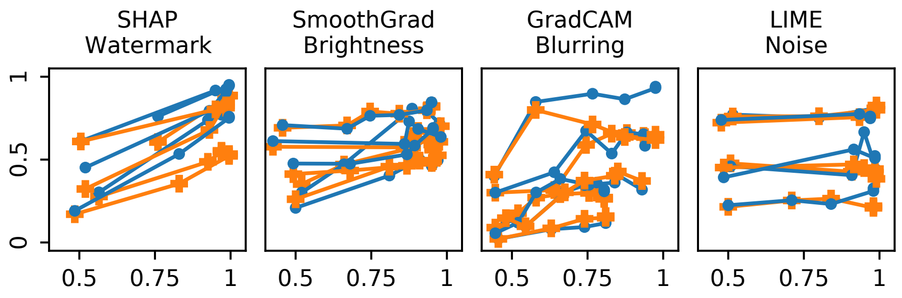


Figure 4-6: %Attr (y-axis) vs. test accuracy (x-axis), more in Appendix 4.7.1.

Attribution vs. Manipulation Visibility

Question How well do saliency maps recognize manipulations of different visibility?

Setup We conduct 100 runs, with 20 per manipulation. We further group the 20 runs into 4 groups, with 5 runs in a group using the same manipulation type and effective region but varying degrees of visibility.

We define the five visibility levels for each manipulation type as below. For fair comparison of %Attr at different visibility levels, it is crucial that the effective regions are independent of the visibility, which is satisfied in all manipulation types below.

- **Blurring:** The visibility level is defined as the Gaussian blur standard deviation, with values of $\{2, 4, 6, 8, 10\}$ pixels, from least visible to most.
- **Brightness:** The visibility level is defined as the magnitude of the brightness shift, with values of $\{0.1, 0.15, 0.2, 0.25, 0.3\}$ brightness component of the color (in the range of [0, 1]), from least visible to most.
- **Hue:** The visibility is defined as the magnitude of the hue shift, with values of $\{0.05, 0.1, 0.15, 0.2, 0.25\}$ hue component of the color (in the range of [0, 1]), from least visible to most.
- **Noise:** The visibility is defined as the probability that a pixel is replaced by a random value, with values of $\{0.02, 0.04, 0.06, 0.08, 0.1\}$, from least visible to most.

- **Watermark:** The visibility is defined as the font size of the watermark, with values of $\{7, 9, 11, 13, 15\}$ pixels, from least visible to most.

As before, the labels are reassigned with $r = 0.5$ and manipulations applied to the positive class only.

Expectation A good saliency map should not be affected by manipulation visibility, as long as the model is objectively using it. However, different saliency maps may be better suited to detect more or less visible manipulations. For example, a less visible manipulation may be ignored by the segmentation algorithm used by LIME, while inducing sharper gradients in the decision space.

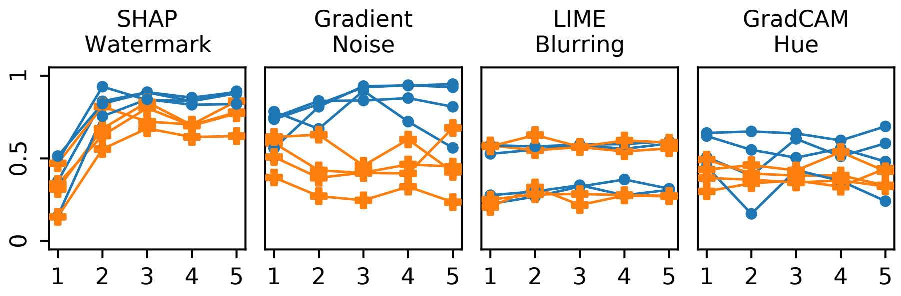


Figure 4-7: %Attr (y-axis) vs. feature visibility (x-axis), more in Appendix 4.7.1.

Results Figure 4-7 (left) plots each group of five runs as a line, with visibility level on the x-axis and %Attr on the y-axis (complete results in Appendix 4.7.1). Except for SHAP on watermark, other methods do not show consistent trend of %Attr increasing with visibility. While SHAP is more effective on more visible manipulations, we rely most on interpretability methods to uncover precisely the less visible manipulations or artifacts. Unfortunately, none of the methods could satisfy this requirement.

Attribution vs. Original Feature Correlation

Question How does the attribution on the manipulation change if the reassigned labels are correlated with the original labels (and thus original input features) to higher or lower degrees (i.e., $r \in [0.5, 1.0]$)?

Setup For each manipulation, we vary the label reassignment parameter $r \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. For each r, we train four models on four class pairs: two of similar species (e.g., class 4 vs. 5 in Figure 4-3) and two of distinct ones (e.g., class 5 vs. 6), for a total of $5 \times 6 \times 4 = 120$ runs.

Expectation For $r > 0.5$, there is no standard definition of the attribution value on the original image features and the manipulations, as any decreasing trend of %Attr with increasing r is reasonable. However, the Shapley value [133] is a commonly used axiomatic definition for feature attributions. We denote the set of features inside the effective region as F_M, for manipulated features, and that outside as F_O, for original features. Their Shapley values on performance $v(F_M)$ and $v(F_O)$ are defined as

$$v(F_M) = \tfrac{1}{2}\left[a(F_M) - a(\varnothing) + a(F_M \cup F_O) - a(F_O)\right], \tag{4.1}$$

$$v(F_O) = \tfrac{1}{2}\left[a(F_O) - a(\varnothing) + a(F_M \cup F_O) - a(F_M)\right], \tag{4.2}$$

where $a(F_M), a(F_O), a(\varnothing)$, and $a(F_M \cup F_O)$ refers to the classifier's expected accuracy when only F_M, only F_O, neither, and both are available, respectively. Note that this value should *not* be calculated as the model accuracy on images with every pixel but F being blacked out, because such images are out of distribution where the model may exhibit unreasonable behaviors (cf. discussion by Hooker et al. [67]).

Instead, the suppression of information beyond F can be understood as an inability for the model to distinguish inputs that agree on F. This leads to the following process of simulating such a prediction. First, let $\mathbb{P}_{X,Y|F=f}$ be the data distribution conditioned on $F = f$. Since all features other than F are suppressed, the model cannot further distinguish two inputs $x, x' \sim \mathbb{P}_{X|F=f}$. As a result, the expected accuracy can be computed by comparing the model's prediction on x against the ground truth label on x'. Then we take the expectation of this accuracy according to different values of $f \sim \mathbb{P}_F$, where $\mathbb{P}_F$ is the marginal distribution of F. Formally, for the model prediction function $g : \mathcal{X} \to \mathcal{Y}$, we have

$$a(F) = \mathbb{E}_{f\sim\mathbb{P}_F}\left[\mathbb{E}_{x,y\sim\mathbb{P}_{X,Y|F=f}}\left[\mathbb{E}_{x',y'\sim\mathcal{P}_{X,Y|F=f}}\left[\mathbb{1}_{g(x)=y'}\right]\right]\right]. \tag{4.3}$$

With balanced label distribution, $a(\varnothing)$ means that the model has no information about the input, and thus the accuracy is 0.5. On the other hand, $a(F_M \cup F_O)$ means that the model has full access to the input, and thus the accuracy is the normal model accuracy p. In addition, we have $a(F_O) \leq r$, because the label reassignment weakens the correlation between F_O and the label.

Finally and somewhat counter-intuitively, the above definition also implies that $a(F_M) = a(F_M \cup F_O) = p$ for the following reason: since every $F_M = f_M$ is perfectly correlated with the label, all the data in $\mathbb{P}_{X,Y|F_M=f_M}$ have the same label and thus the non-identifiability of any two inputs x and x' does not additionally degrade the model performance. However, note that this result comes from the mechanical application of

Shapley value calculation, which is a popular and *axiomatic* definition of attribution. Whether it is reasonable in light of this implication is beyond the scope of the thesis.

Following the above procedure, we have that for a classifier with accuracy p, $a(\varnothing) = 0.5$, $a(F_M) = a(F_M \cup F_O) = p$, and $a(F_O) \leq m$. We normalize the Shapley values to $\overline{v}(F_M)$ and $\overline{v}(F_O)$ by their sum $\overline{v}(F_M) + \overline{v}(F_O)$.

It is easy to see that

$$\overline{v}(F_M) \geq (2p - r - 0.5)/(2p - 1), \tag{4.4}$$

$$\overline{v}(F_O) \leq (r - 0.5)/(2p - 1). \tag{4.5}$$

For (near-)perfect classifier with $p \approx 1$, we have $\overline{v}(F_M) \geq 1.5 - r$, and $\overline{v}(F_O) \leq r - 0.5$. In addition, $a(F_O)$ should be close to r for the distinct pair as the model can better utilize the more distinct original image features, resulting in lower attribution $\overline{v}(F_M)$ on manipulated features.

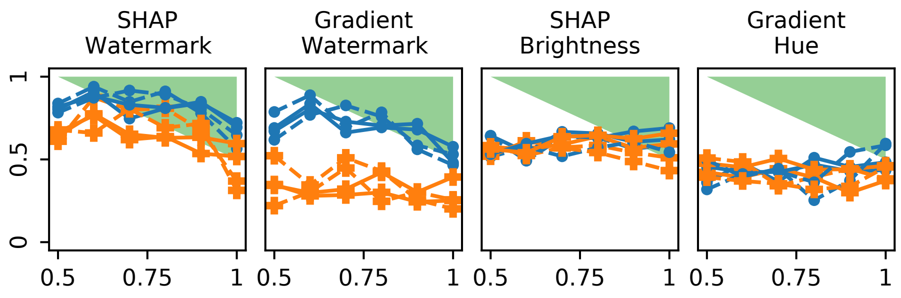


Figure 4-8: %Attr (y-axis) vs. label reassignment parameter r (x-axis), more in Appendix 4.7.1. Solid/dashed lines represent similar/distinct species pairs. Green shades represent attribution range per Shapley axioms: %Attr $\geq 1.5 - r$.

Results All models achieve test accuracy of over 95%. Figure 4-8 (right) plots %Attr vs. r (complete results in Appendix 4.7.1). Solid lines represent runs with a similar species pair, and dashed lines represent runs with a distinct species pair. The green shaded area represents the area of $\overline{v}(F_M) \geq 1.5 - r$, the Shapley value range at $p = 1$. In other words, the green shade represent values consistent with the Shapley axioms. Intuitively, for r close to 0.5, the correlation between F_O and the label is very weak, and the (near-)perfect model has to use F_M for high performance, thus %Attr close to 1. As r increases, the model can choose to rely more heavily on the more-correlating F_O as well, resulting in larger allowable ranges of %Attr.

For watermark manipulation, SHAP shows clear decrease in attribution value as r increases, while gradient also tracks the predicted range, but only for the positive class with the manipulation. This trend is not seen in other feature types, even for SHAP which approximates the Shapley values. There does not seem to be a clear difference in attribution values for similar vs. distinct species pairs either. Given that the set of Shapley axioms is commonly accepted as reasonable, it is concerning that many saliency maps are inconsistent with it, and important to better understand the underlying axiomatic assumptions (if any) made by each of them.

Discussion

Arguably one of the most important application of model explanation is to detect any usage of spurious correlations, but our results cast doubt on this capability from various aspects. We recommend that, before analyzing the actual model, developers should first train models that are guaranteed to use certain known features, and "dry run" the planned interpretability methods on them to make sure that these features are indeed highlighted.

4.5.2 Evaluating Text Attentions

It is known that certain non-semantic features can heavily influence model prediction, such as the email headers [128]. Plausibly, attention scores should highlight such features, and we rigorously test this with our dataset modification in this section.

Model The attention architecture follows the one used by Wiegreffe and Pinter [162] closely. First, a sentence of L words $(w_1, ..., w_L)$ is converted to a list of 200-dimensional embeddings $(\mathbf{v}_1, ..., \mathbf{v}_L)$. We use the same embedding data as Lei et al. [94] and Bastings et al. [15]. Then, a Bi-LSTM network builds contextual representations for these words $\mathbf{h}_1, ...\mathbf{h}_L$, where $\mathbf{h}_i \in \mathbb{R}^{400}$ is the concatenation of the forward and the backward hidden states, each of 200 dimensions. Finally, the attention mechanism computes the sentence representation as

$$\begin{aligned}
\mathbf{k}_i &= \tanh(\text{Linear}(\mathbf{h}_i)) \in \mathbb{R}^{200}, && (4.6)\\
b_i &= \mathbf{q} \cdot \mathbf{k}_i && (4.7)\\
a_1, ..., a_L &= \text{softmax}(b_1, ..., b_L), && (4.8)\\
\mathbf{h} &= \sum_{i=1}^{L} a_i \mathbf{h}_i, && (4.9)
\end{aligned}$$

where Linear() represents a linear layer with learned parameters, $\mathbf{q} \in \mathbb{R}^{200}$ is a learned query vector applied to every sentence, and $a_1, ..., a_L$ are the attention weights for

$w_1, ..., w_L$. Finally, a linear layer computes the 2-dimensional logit vector for model prediction.

Dataset We modify the BeerAdvocate dataset [104] and further select 12,000 reviews split into train, validation, and test sets of sizes 10,000, 1,000 and 1,000 (shuffled differently for each experiment).

Metric The introduced manipulation changes specific words according to the reassigned label. The metric is %Attr defined on the target words (i.e., effective region).

Highly Obvious Correlating Features

Question How well can attention scores focus on highly obvious manipulations?

Setup From our filtered dataset, we first randomly assign binary labels. For the positive reviews, we change all the article words (*a* / *an* / *the*) to "the", and for the negative reviews, we change these to "a". Thus, only these articles are correlated with the labels and constitute the effective region.

Expectation Attention of (near-)perfect models should have %Attr ≈ 1.

Results The model achieves over 97% accuracy. Across the test set, %Attr on article words is 8.6%. Considering that articles are 7.9% of all words, this is better than random, albeit barely. Figure 4-9 visualizes the attention distribution for two reviews, with additional results in Figure 4-18 of Appendix 4.7.2. Each bars represent weights of words in the review. Green and orange bars represent non-articles and articles respectively. As we can see, the attention on article words either does not stand out from the rest, or at most only locally, *relative* to their neighbors. Generally, there is no strong correlation between high attention values and important words.

Misleading Non-Correlating Features

Question When some features are known to *not* correlate with the label but are very similar to correlating ones, do attention scores also focus on the former?

Setup Again from our filtered dataset, we apply two similar manipulations, with only one of them is correlated with the (reassigned) label. Figure 4-10 details the construction of two datasets, *CN* and *NC*.

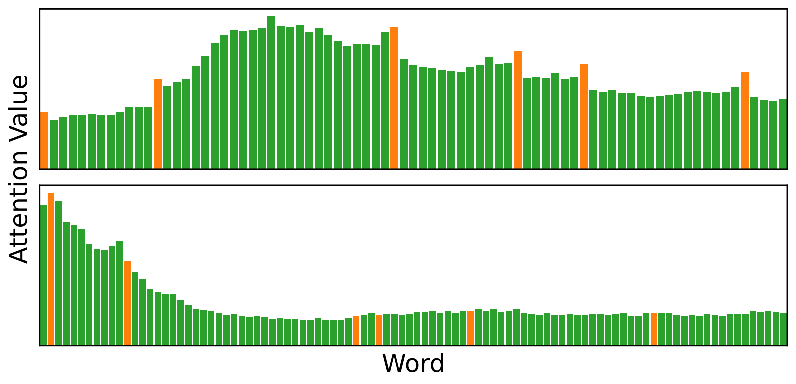
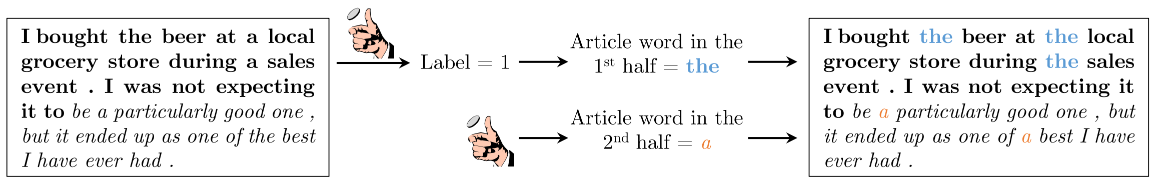


Figure 4-9: Attention scores for words in two reviews, with more in Appendix 4.7.2. Orange and green bars represent articles and non-articles, respectively.


I bought the beer at a local grocery store during a sales event . I was not expecting it to *be a particularly good one , but it ended up as one of the best I have ever had .*

Label = 1
Article word in the 1st half = the
Article word in the 2nd half = a

I bought the beer at the local grocery store during the sales event . I was not expecting it to *be a particularly good one , but it ended up as one of a best I have ever had .*

Figure 4-10: The process to build *CN* dataset for the experiment in Section 4.5.3. First, a review is split into two halves at the midpoint, shown in **bold** and *italics*. Then a label is randomly sampled and assigned to the review. Depending on the label, the articles in the first half are changed to "a" or "the". They are called *correlating articles*. Then an article word is randomly chosen for the second half, and all articles in the second half are changed to that word. They are called *non-correlating articles*. For *NC* dataset, the roles of two halves are switched.

Expectation Same as above. Non-correlating articles should *not* be attended to.

Results The models on both datasets achieve over 97% accuracy. Figure 4-11 presents attention visualization, with more in Figure 4-19 of Appendix 4.7.2. The two models show very different behaviors. The *CN* model exclusively focuses attention on correlating articles, while the *NC* model behaves similarly to the previous experiment.

Observing the large variation of behaviors, we further trained the three models ten more times to see if any consistent attention pattern exists. All models achieve over 97% accuracy. Figure 4.1 (left) presents the mean and standard deviation statistics for the 11 runs. The clean attention pattern by the *CN* model does not persist, and the model sometimes assigns higher than random weights on non-correlating articles, especially for the *NC* dataset. These results further suggests that attention weights cannot be readily and reliably interpreted as attributions without further validation.

Dataset	Corr. Articles	Non-Corr. Articles	Other Words
Article	10.3% ± 2.4% | 7.9%	NA	89.7% ± 2.4% | 92.1%
CN	15.9% ± 25.7% | 4.1%	5.9% ± 4.0% | 3.8%	78.2% ± 24.7% | 92.1%
NC	12.0% ± 8.4% | 3.8%	12.6% ± 9.3% | 4.1%	75.4% ± 16.7% | 92.1%

Table 4.1: Attention attribution statistics across 11 training runs (format: mean(%Attr) ± stdev(%Attr) | word frequency). The "Article" dataset is the one used in Section 4.5.2.

Discussion

Attention is undoubtedly useful as a building block in neural networks, but their *interpretation* as attribution is disputed. Due to the lack of ground truth information on word-prediction correlation, past studies proposed various, and sometimes conflicting, criteria for judging the validity of attribution interpretation [75, 122, 162]. However, the fundamental correctness of such proxy metrics is unclear. In our studies, we find that attentions can hardly be interpreted as attribution for model understanding and debugging purposes: for most training runs, the attention weights on correlating features at best stand out only *locally*, easily overwhelmed by larger global variations, settling the debate at least on the modified dataset. For natural datasets, we would unavoidably need to rely on proxy metrics, but we recommend future proposals of the metrics to be first calibrated with ground truth in a controlled setting.

4.5.3 Evaluating Text Rationales

In this section, we evaluate rationale models with the same two experiments above (and omit the **Question** and **Setup** descriptions). We consider two variants: a re-

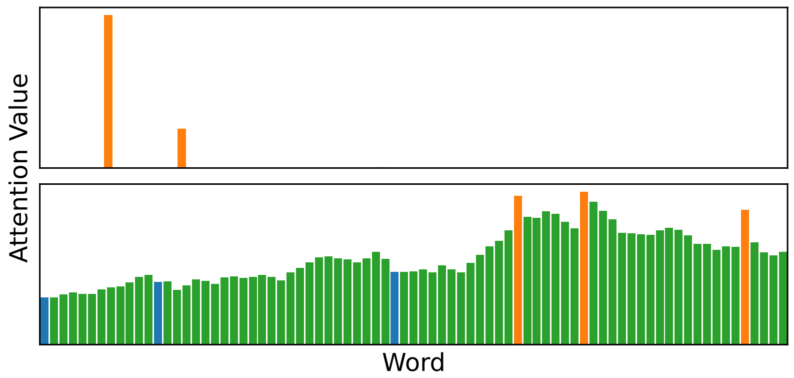


Figure 4-11: Attention scores for one review in *CN* (top) and *NC* (bottom) dataset, with more in Appendix 4.7.2. Orange, blue, and green bars represent correlating articles, non-correlating articles, and other words, respectively.

inforcement learning (RL) model [94] and a continuous relaxation (CR) model [15]. In the original forms, both models regularize the rationale length and continuity. In our experiments, rather than regularizing the length, we train the models to produce rationales that match a target selection rate %Sel. For a mini-batch of B examples, we use $\lambda \cdot \left| \sum_{i=1}^{B} \text{len}(\text{rationale}_i) / \sum_{i=1}^{B} \text{len}(\text{review}_i) - \text{Sel}\% \right|$, where $\lambda > 0$ is the regularization strength. Incidentally, we found that the training is much more stable with this regularization, especially for the RL model. We also removed the discontinuity penalty, because ground truth rationales in our experiments are not continuous. We use precision and recall metrics as defined in Section 4.3.

Highly Obvious Manipulations

Expectation A necessary condition for a non-misleading rationale is that it should include at least one article word, regardless of selection rate. However, a desirable property of rationale is comprehensiveness [168]: selecting as many article words as possible. Thus, a good rationale model should have high precision when selection rate is low and high recall when selection rate is high.

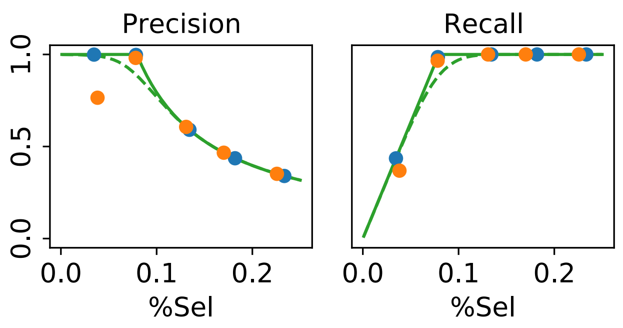
Results We train models with $\%\text{Sel} \in \{0.07, 0.09, 0.11, 0.13, 0.15\}$, all with over 97% accuracy. We evaluate precision and recall of the trained models and plot them in Figure 4-12 (top) according to the actual rationale selection rate, %Sel, on the test set. Blue and orange markers are for the RL and CR models respectively. The two green lines show two optimality notions: the solid line enforces aggregate %Sel for the test set, and the dashed line enforces %Sel per review.

Except for the CR model at the lowest %Sel, all others achieve near-perfect rationale selection on both the precision and and the recall metrics. In particular, they are nearly dataset-wide optimal, due to %Sel regularization done at the mini-batch level. The "faulty" CR model tends to select the first few words consistently, as shown in Figure 4-12 (bottom) and Appendix 4.7.3, but still selects some article words.

Misleading Non-Correlating Features

Expectation Similar to the previous experiment, at least one correlating article word needs to be selected. However, selection of non-correlating articles is arguably more misleading than selection of other non-article words, because it suggests that these non-correlating articles also influence the prediction, even though the classifier simply ignores them.

Results: We train models with $\%\text{Sel} \in \{0.03, 0.05, 0.07, 0.09\}$, all with over 97%



enjoyed @ ***la*** cave **the** bulles ; simon & *the* head brewer of brasserie de vines hosted **the** tasting on 11/5 . medium body , frothy mouth-feel , nice carbonation . nice fruity notes upfront , green apples and citrus , with *the* hint of sourness . finishes with *the* fresh piney hop presence and *the* mild bitterness . overall ; great diversity in flavors , very fresh tasting .

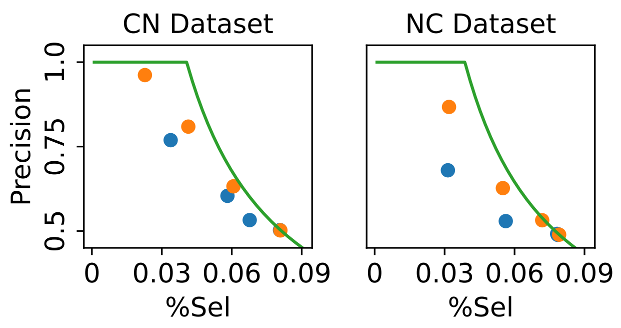
Figure 4-12: Top: Precision and recall for two rationale models in Section 4.5.3. Bottom: A rationale pattern by the "faulty" CR model, selected non-articles in ***orange bold italics***, selected articles in **green bold**, and missed articles in *red italics*, more in Appendix 4.7.3.

accuracy. Figure 4-13 (top) plots the precision of correlating articles for the two datasets, as well as the dataset-wide optimal value. We found the rationales consist of almost exclusively article words. However, especially for the RL model, some correlating articles are missed but non-correlating ones are selected, resulting in markedly less than optimal precision. Figure 4-13 (bottom) shows one example for the RL model and additional ones are in Appendix 4.7.3.

Discussion

The structure of rationale models guarantees that causal relationship between the rationale features and the model prediction, but this does not necessarily imply its usefulness to model understanding. Specifically, it could highlight F_C only barely, while including lots of non-correlating F_N (and, in particular, misleading words such as the non-correlating articles)[2]. Indeed, our results show that rationale methods are prone to selecting misleading non-correlating features, which obfuscates the model's reasoning process by giving *more* but unnecessary information to the human. The problem is more severe with RL training, possibly due to the known difficulty with REINFORCE [163]. Post-processing methods could be developed to further prune rationales to mitigate this problem.

[2]There are additional concerns on the unfaithfulness of rationales as Trojan explanations [74, 172], but they were not identified in our experiments.



enjoyed @ la cave *the* bulles ; simon & *the* head brewer of brasserie de vines hosted *the* tasting on 11/5 . medium body , frothy mouth-feel , nice carbonation . nice fruity notes upfront , green apples and citrus , with **a** hint of sourness . finishes with **a** fresh piney hop presence and **a** mild bitterness . overall ; great diversity in flavors , very fresh tasting .

Figure 4-13: Top: Precision at different %Sel for models in Section 4.5.3. RL model is in blue and CR in orange. The solid and dashed green lines show optimal metric values when %Sel is enforced at dataset- and sentence-level. Bottom: A rationale selection on the *NC* dataset, correlating articles in **green bold** and non-correlating articles in *red italics*, more in Appendix 4.7.3.

4.6 Discussion

As interpretability methods, especially feature attribution ones, are increasingly deployed for quality assurance of high-stakes systems, it is crucial to ensure these methods work correctly. Current evaluations fall short – primarily due to a lack of clearly defined ground truth. Rather than evaluating explanations for models trained on natural datasets, we propose "unit tests" to assess whether feature attribution methods are able to uncover ground truth model reasoning on carefully-modified, semi-natural datasets. Surprisingly, none of our evaluated methods across vision and text domains achieve totally satisfactory performance, and we point out various future directions in Section 4.5.1, 4.5.2 and 4.5.3 to improve attribution methods.

Our dataset modification procedure closely parallels the setup for identifying and debugging model reliance on spurious correlations, which have been known to frequently affect model decisions [e.g. 47, 72, 80, 128]. Hence, the mostly negative conclusions cast doubt on this use case of interpretability methods.

An extension of the proposed evaluation procedure is to move beyond "artifact" features, which result from the manual definition of the manipulation function. Given the recent advances on generative modeling such as image inpainting [120] and masked language prediction [38], more realistic features could be generated, perhaps also conditioned on or guided by semantic concepts. This would make the modified dataset much more realistic looking, and thus better simulate another intended use case of interpretability: assisting scientific discovery, in which high-performing models teach humans about features of previous unknown importance.

4.7 Appendix

4.7.1 Image Saliency Map Evaluations

Figure 4-14 shows %Attr vs. %ER for all pairs of saliency maps and manipulations.

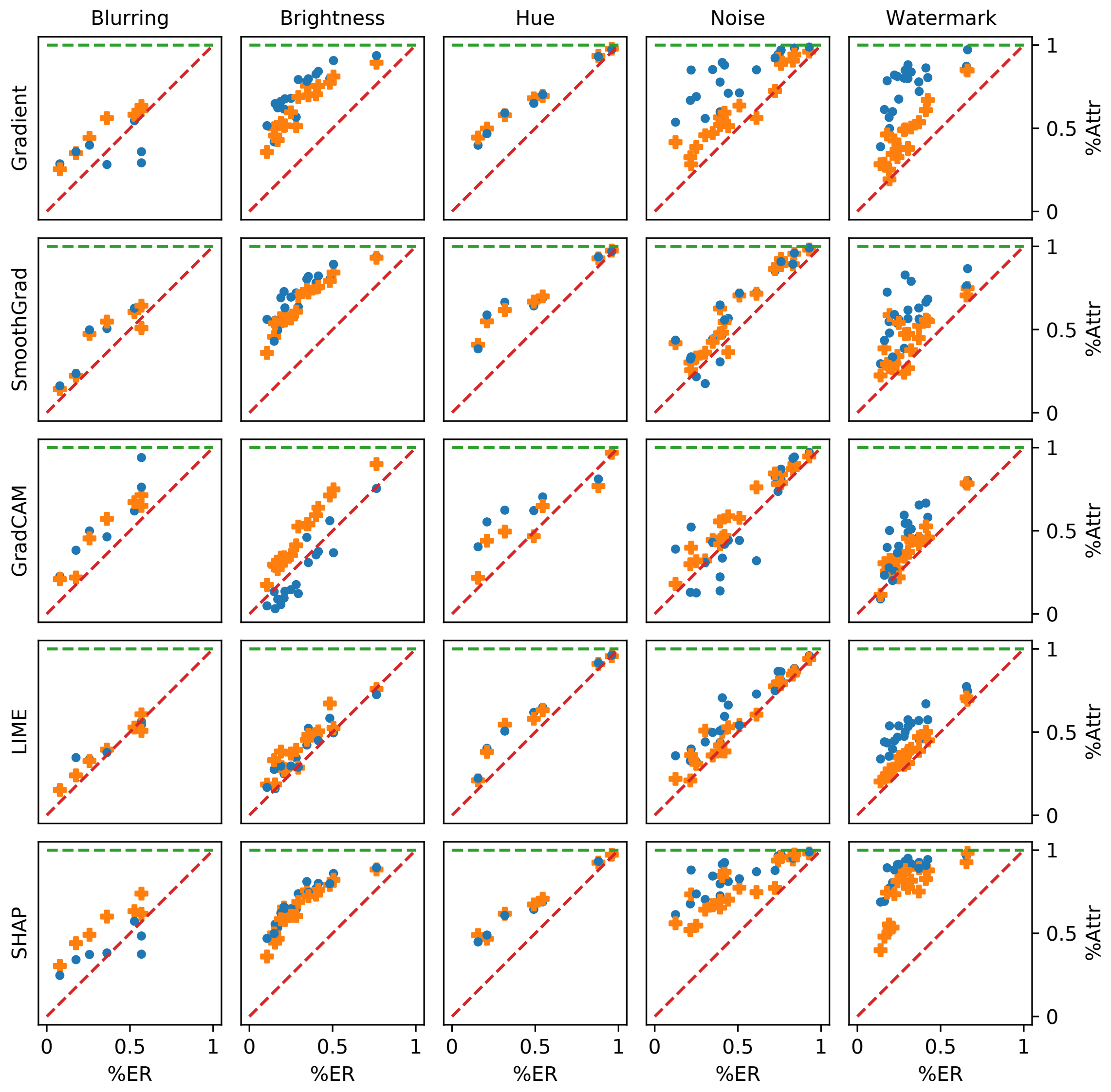


Figure 4-14: %Attr (y-axis) vs. %ER (x-axis) for all pairs of saliency maps and manipulations.

Figure 4-15 shows %Attr vs. test accuracy for all pairs of saliency maps and manipulations.

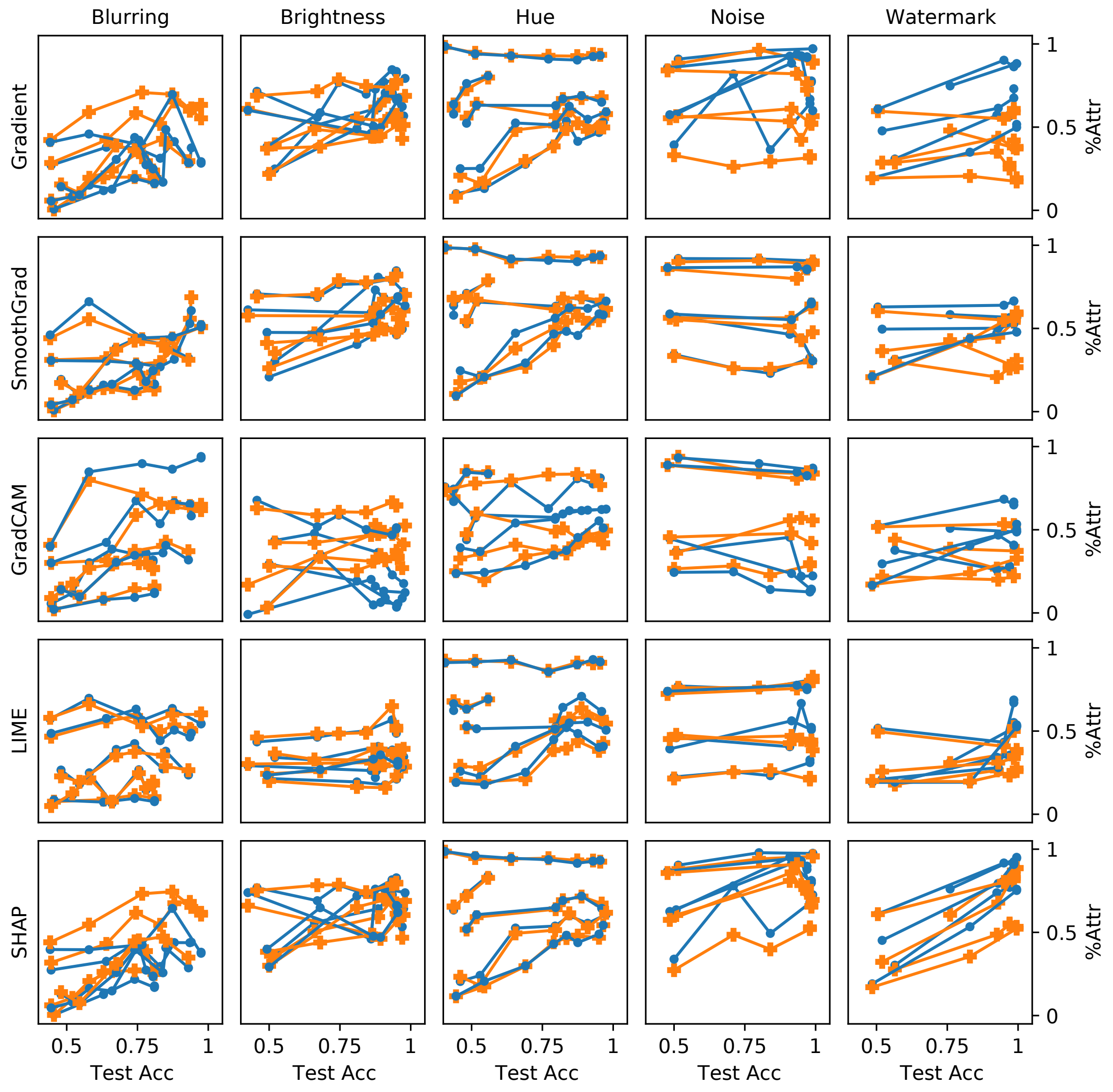


Figure 4-15: %Attr (y-axis) vs. test accuracy (x-axis) for all pairs of saliency maps and manipulations.

Figure 4-16 shows %Attr vs. manipulation visibility for all pairs of saliency maps and manipulations.

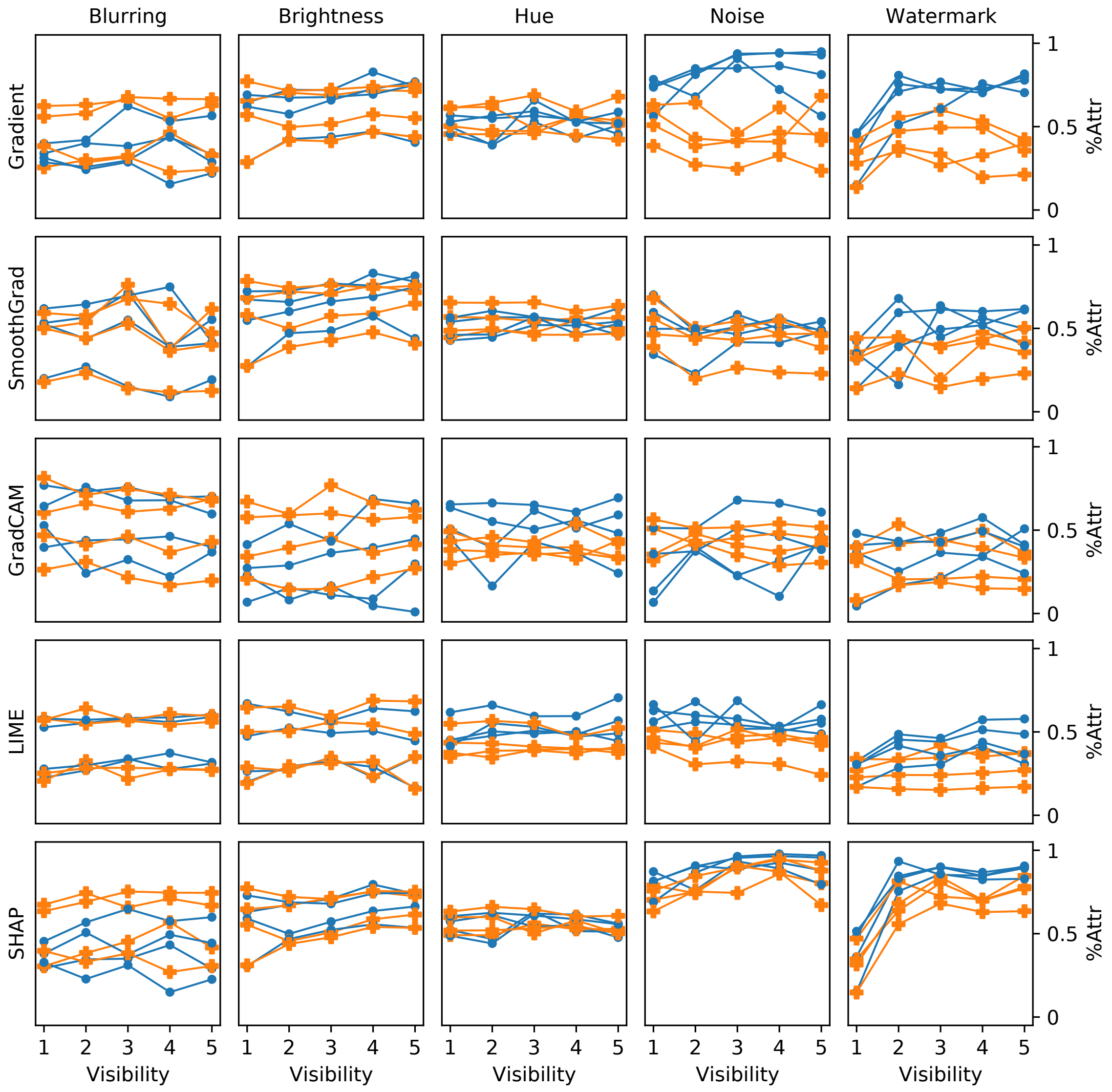


Figure 4-16: %Attr (y-axis) vs. manipulation visibility (x-axis) for all pairs of saliency maps and manipulations.

Figure 4-17 shows %Attr vs. the label reassignment parameter r for all pairs of saliency maps and manipulations.

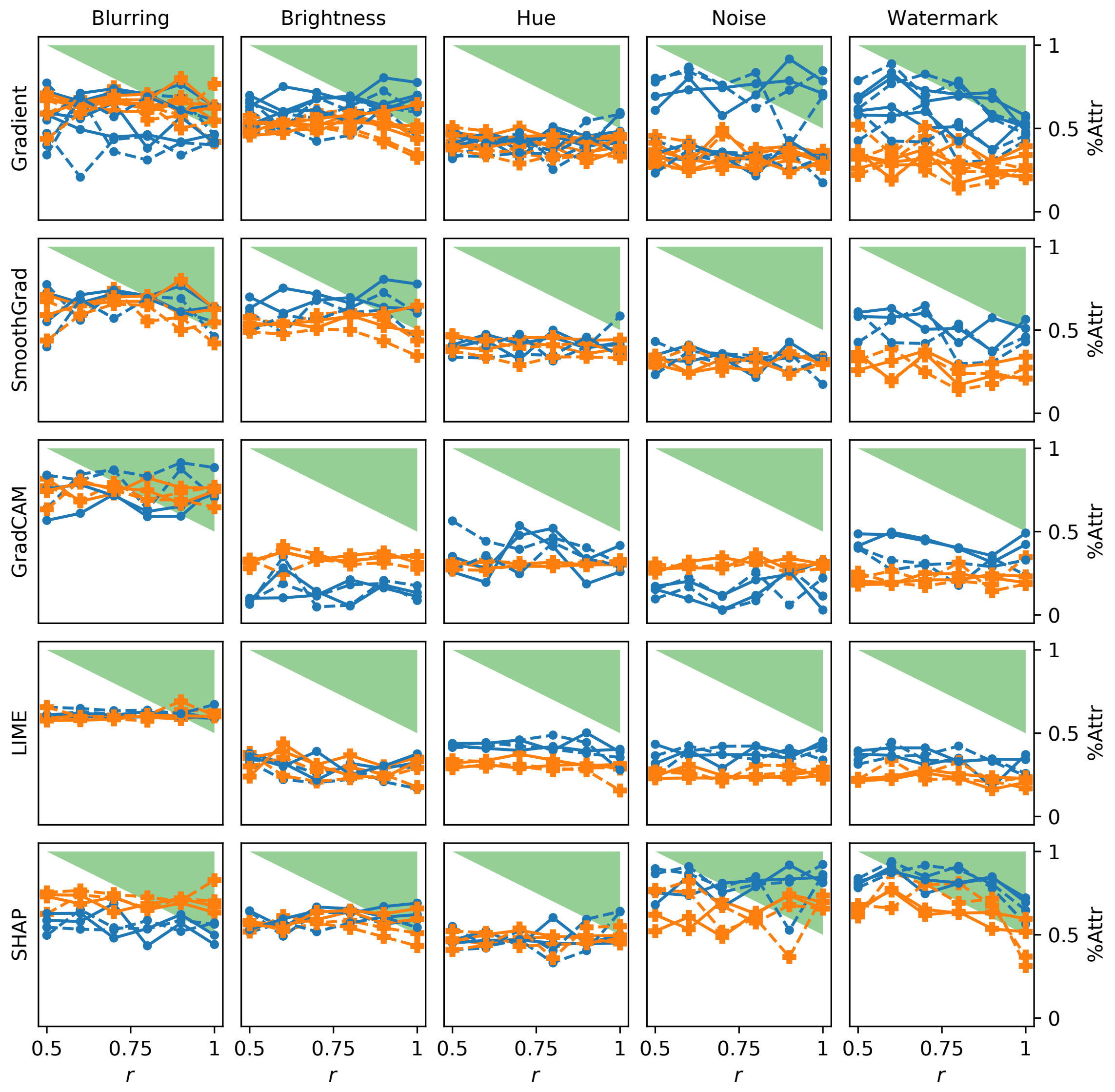


Figure 4-17: %Attr (y-axis) vs. label reassignment parameter r (x-axis) for all pairs of saliency maps and manipulations.

4.7.2 Attention Mechanism Evaluations

Figure 4-18 presents additional visualizations of the learned attention distribution on the dataset modified with article word spurious correlation.

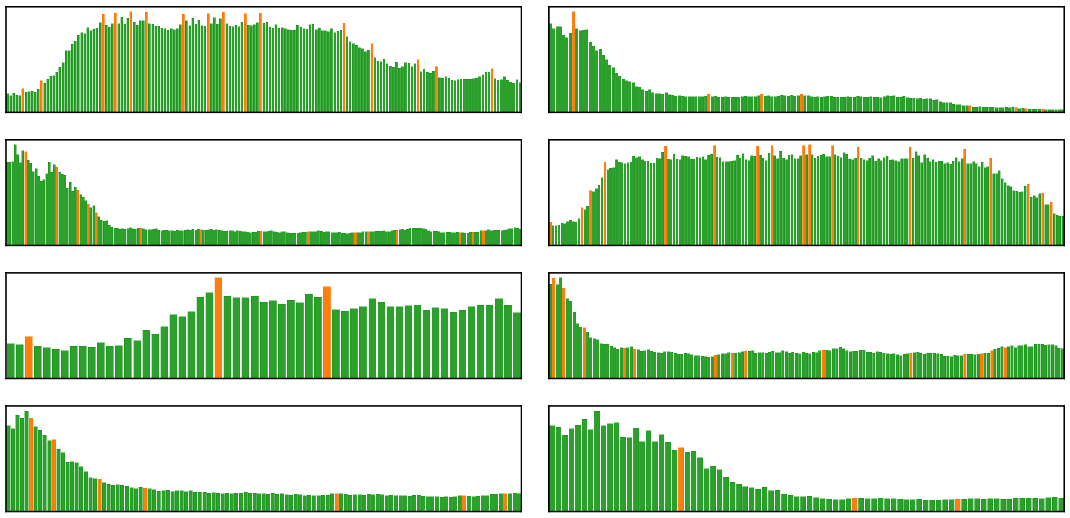


Figure 4-18: Additional attention visualizations on the review dataset. Orange and green bars represent articles and non-articles respectively.

Figure 4-19 presents additional visualizations of the learned attention distribution on the *CN* (left) and *NC* (right) datasets.

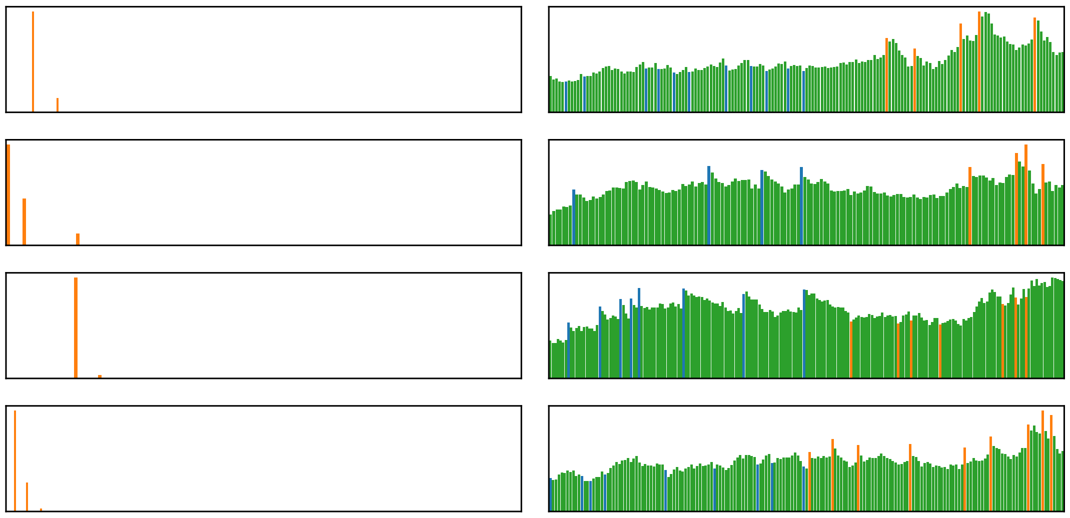


Figure 4-19: Additional attention visualizations on the *CN* (left) and *NC* (right) datasets. Orange, blue and green bars represent correlating articles, non-correlating articles and other words respectively.

4.7.3 Rationale Model Evaluations

Highly Obvious Manipulations

Figure 4-20 presents four additional reviews annotated by the "faulty" CR model showing that it consistently selects the first few words of the review.

pours **a** clear yellow . 1/4 inch head of **a** white color . slight retention and slight lacing . smells of sweet malt , pale malt , fruit , and slight bread aroma . fits *a* style of **a** belgian pale ale . mouth feel is smooth and crisp with *a* high carbonation level . tastes of pale malt , yeast cleanliness , slight hops , and very slight fruit . overall , *a* decent brew but nothing special .	***bottle to*** snifter glass . pitch black with little lacing around edge . smells like *the* typical oatmeal stout . taste has *the* great balance between both milk and oatmeal . sweet from *the* sugars and mild dark chocolate in *the* after taste . smooth and chewey . leans to *the* heavier side in *the* mouth . great example of two styles blended . worth seeking .
12oz can poured ***into*** pint glass . pours **the** pale golden straw color with **the** 2 finger fizzy head that settles quickly . slightly hazy when held to **the** light . smell is fairly nuetral with *the* bit of sweet malts coming through . *the* slight scent of something metallic . taste is decent . nothing crazy or unique but extremely clean , classic american pale lager flavor . mild light malt flavor with just enough hops for balance . no major off-flavors here . mouthfeel is fluid and crisp . this went down quickly and i am not **the** fizzy yellow beer fan . for what it is , it 's done well .	***a- dark*** brown with hints of amber at **a** edges , small head which disappeared quickly and dissipated into **a** few sad bubbles . s- tons of sweet bourbon booze . raisins , sugared malts , dark chocolate filled with raspberry . t- booze and brown sugared malts mingle with one another . this is drinking like *a* barleywine to me . lots of wood and oak flavor drenched in booze . m- smooth creamy with enough carbonation . d- it 's *a* delicious brew that needs to be savored one of **a** best if not **a** best scotch ales ive had .

Figure 4-20: Four additional reviews annotated by the "faulty" continuous relaxation model that consistently selects the first few words regardless. Selected non-articles in ***orange bold italics***, selected articles in **green bold**, and missed articles in *red italics*.

Misleading Non-Correlating Features

Figure 4-21 shows rationale selections by the two models for the same review at the same target %Sel.

	CN Dataset	*NC* Dataset
CR Model	enjoyed @ la cave **a** bulles ; simon & **a** head brewer of brasserie de vines hosted **a** tasting on 11/5 . medium body , frothy mouth-feel , nice carbonation . nice fruity notes upfront , green apples and citrus , with *the* hint of sourness . finishes with *the* fresh piney hop presence and *the* mild bitterness . overall ; great diversity in flavors , very fresh tasting .	enjoyed @ la cave *the* bulles ; simon & *the* head brewer of brasserie de vines hosted *the* tasting on 11/5 . medium body , frothy mouth-feel , nice carbonation . nice fruity notes upfront , green apples and citrus , with **a** hint of sourness . finishes with **a** fresh piney hop presence and **a** mild bitterness . overall ; great diversity in flavors , very fresh tasting .
RL Model	enjoyed @ la cave **a** bulles ; simon & **a** head brewer of brasserie de vines hosted **a** tasting on 11/5 . medium body , frothy mouth-feel , nice carbonation . nice fruity notes upfront , green apples and citrus , with *the* hint of sourness . finishes with *the* fresh piney hop presence and *the* mild bitterness . overall ; great diversity in flavors , very fresh tasting .	enjoyed @ la cave *the* bulles ; simon & *the* head brewer of brasserie de vines hosted *the* tasting on 11/5 . medium body , frothy mouth-feel , nice carbonation . nice fruity notes upfront , green apples and citrus , with **a** hint of sourness . finishes with **a** fresh piney hop presence and **a** mild bitterness . overall ; great diversity in flavors , very fresh tasting .

Figure 4-21: Additional rationale annotations on the *CN* and *NC* datasets by the two models. Selected words are underlined. Ground truth correlating articles are in **green bold**, and non-correlating articles in *red italics*. The CR model performs well on this review, focusing exclusively on correlating articles, while the RL model selects non-correlating articles, and misses a correlating one for the *NC* dataset.

Chapter 5

Understandability of Model Explanations

5.1 Introduction

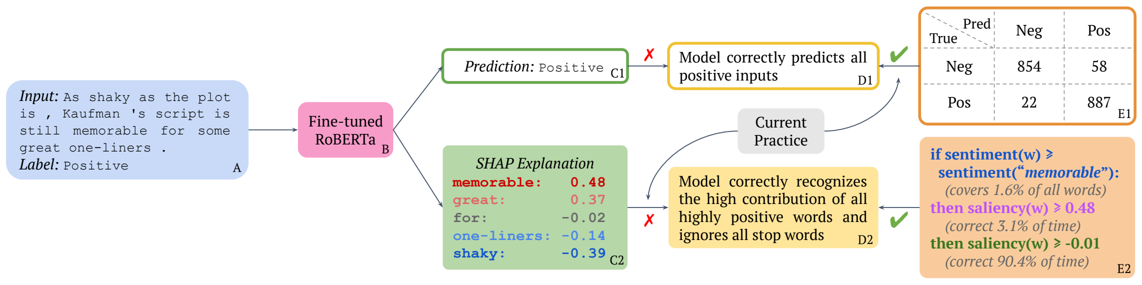
The previous chapter highlights the importance of correctness evaluation. Without it, for instance, we risk being misled by explanations into believing that models exploiting spurious correlations are instead working correctly. However, is correctness the sole desideratum that we should aim for?

In this chapter, we argue negatively. As a simple example, consider perhaps an atypical explanation: the computation trace. For an input, the computation trace explanation is defined as the sequence of addition, multiplication and exponentiation operations that the model implements to compute the prediction. For a linear regression model on an input of D feature dimensions, there are $2D$ operations: each feature is first multiplied by the weight with D multiplications, and the results are summed up with the bias term, using D additions. Similarly, the computation done by a deep neural network can also be represented by a sequence of such operations, just much longer. It is clear that this computation trace is 100% correct, in that the exact prediction is recovered by following the trace. However, it is rarely, if ever, considered as a model explanation. The reason is that this trace is as hard to understand as the original model parameters and it is essentially impossible to derive

This chapter is based on the NAACL 2022 paper "ExSum: From Local Explanations to Model Understanding" by Yilun Zhou, Marco Tulio Ribeiro and Julie Shah [183].

more general and high-level model understanding, such as whether the model is using spurious correlations or is discriminative on certain demographic factors.

From a more practical and utilitarian perspective, as discussed at the end of Chapter 3, explanations are ultimately used by people for making various decisions. Thus, explanations that cannot be understood (e.g., the computation trace) by human consumers are unhelpful, as are those that are misunderstood.



Input: As shaky as the plot is , Kaufman 's script is still memorable for some great one-liners .
Label: Positive
A
Fine-tuned RoBERTa
B
Prediction: Positive
C1
✗
Model correctly predicts all positive inputs
D1
✓
Pred
True
Neg
Pos
Neg
854
58
Pos
22
887
E1
Current Practice
SHAP Explanation
memorable: 0.48
great: 0.37
for: -0.02
one-liners: -0.14
shaky: -0.39
C2
✗
Model correctly recognizes the high contribution of all highly positive words and ignores all stop words
D2
✓
if sentiment(w) ⩾ sentiment("*memorable*"):
(covers 1.6% of all words)
then saliency(w) ⩾ 0.48
(correct 3.1% of time)
then saliency(w) ⩾ -0.01
(correct 90.4% of time)
E2

Figure 5-1: An analogy between understanding model prediction (top route) and model explanation (bottom route). A test input (A) is fed into a fine-tuned RoBERTa model (B), which generates a correct prediction (C1) and reasonable explanation (C2). While generalized claims of understanding model *performance* (D1) are made rigorously from quantitative statistics such as the test set confusion matrix (E1), claims of understanding model *behavior* (D2) are predominantly derived informally from one or few explanations (C2). In this chapter, we argue the necessity of formalizing this process, and propose the explanation summary (ExSum) framework (E2), which reveals the severe limitations of the *ad hoc* model understanding (D2).

Consider the sentiment classification task shown in Figure 5-1. On a test input, the model makes the correct prediction of positive sentiment. Obviously, this evidence is insufficient to conclude that "*in general*, the model classifies positive inputs correctly," because even a random-guess model is correct 50% of the time on a single instance. Instead, statistics such as the confusion matrix serve to rigorously support (or refute) generalization claims about model *performance* – for example, "the model is correct 97.6% of the time on positive inputs" – ensuring an accurate understanding of model performance.

Do we understand model *behaviors* in the same rigorous way? Figure 5-1 shows that the SHAP score [99] of the word "*memorable*" is highest at 0.48, while that of "*for*" is negligible at -0.02. Therefore, it is tempting to conclude that "*in general*, the model recognizes the high positive contribution of highly positive words and ignores stop words" – as expected for an accurate sentiment classifier. However, this is a generalization from a single instance, and thus potentially unreliable. We need the "confusion matrix" analogue for such claims, which to the best of our knowledge does not exist, making it hard to derive model understanding from local explanations.

We propose ExSum, a mathematical framework to formalize model understanding. In ExSum, each piece of "model understanding" is specified precisely via a rule that links inputs to attribution values. For example, the tentative understanding described in the previous paragraph could be formalized as "words more positive than *memorable* (as measured by the word sentiment score given in the dataset, e.g., *flawless*, *charming*, etc) have SHAP attribution value in the [0.48, 1] range." This precise definition allows for quantitative evaluations. For example, this rule covers 1.6% of all words in the corpus, and is only correct 3.1% of the time. For the rule to be 90% correct, we need a wide and uninformative range of [-0.01, 1], indicating that a hasty generalization from "*memorable*" is unwarranted. Similarly, a saliency range of [-0.05, 0.05] for stop words is only correct 64% of the time: over 1/3 of stop words have *non*-negligible saliency – an understanding that is easily available with ExSum, but might be missed with informal explanation inspection. We define metrics to establish the quality profile of each rule and present a tool that makes it easy for users to construct ExSum rules from local explanations. Finally, we demonstrate how ExSum reveals the various drawbacks in the current practices of *ad hoc* model understanding, and allows for better understanding of model behavior in two separate tasks.

5.2 On Generalized Model Understanding

Besides the practical example above, we start from first principles and argue that *generalized* model understanding is the central concept for explanation usefulness. Local explanations are *mathematical descriptions (MD) of some aspect of model behavior, for specific inputs.* For example, gradient saliency (in the embedding space) is the sensitivity of the prediction to infinitesimal changes in the token embedding; occlusion saliency is the prediction change if individual embeddings are zeroed out. It is with these mathematical descriptions that people associate *high-level interpretations (HL) of model behavior*, such as associating the above two metrics with word importance. This (unconscious) train of thought can be described as follows:

$$x \to \text{MD} \to \text{HL}.$$

Crucially, people rarely study MD or HL for one *specific* input, as explanations are often used to understand broader model behaviors, such as reliance upon spurious correlation, non-discrimination of a protected class, or usage of unknown scientific principles. We elaborate upon these use cases in Appendix 5.9.1 to demonstrate that people implicitly or explicitly seek generalized model understanding. From another perspective, analogous to why people ultimately focus on the *generalization* accuracy of a model, they (should) focus on *generalized* model understanding derived from local explanations.

For example, after observing that *some* highly polar words have high contribution for a sentiment classification model, people conclude that *all* highly polar words have high contribution. This process can be formalized as follows:

$$\left.\begin{array}{c} x_1 \to \mathrm{MD}_1 \to \mathrm{HL}_1 \\ \ldots \\ x_n \to \mathrm{MD}_n \to \mathrm{HL}_n \end{array}\right\} \to \mathrm{HL}^{(\mathrm{g})},$$

where $\mathrm{HL}^{(\mathrm{g})}$ is the *generalized* high-level model understanding. This generalization is too informal, not least because the step from MD_i to HL_i is itself already informal. Alternatively, we propose to generalize at the MD level, as follows:

$$\left.\begin{array}{c} x_1 \to \mathrm{MD}_1 \\ \ldots \\ x_n \to \mathrm{MD}_n \end{array}\right\} \to \mathrm{MD}^{(\mathrm{g})} \to \mathrm{HL}^{(\mathrm{g})}.$$

Since MDs are rigorously defined mathematical quantities (e.g., the prediction of the sentence drops by 32% after the embedding of "great" is zeroed out), we can define and evaluate the quality of their generalization, and $\mathrm{HL}^{(\mathrm{g})}$ can also include any failures and anomalies. As each MD is a local explanation, we call $\mathrm{MD}^{(\mathrm{g})}$ the *explanation summary* (ExSum), and proceed by instantiating this principle for feature attribution explanations.

5.3 The ExSum Framework

5.3.1 Setup and Notation

We focus on the classification setting, but all the ideas below can extend straightforwardly to regression. We have an input space $\mathcal{X}$ and output space $\mathcal{Y} = \{1, ..., K\}$ of K classes. A data point is an input-output pair $d = (x, y) \in \mathcal{D} = \mathcal{X} \times \mathcal{Y}$, distributed as $\mathbb{P}_D$. We consider a model $m : \mathcal{X} \to \Delta^{K-1}$ where $m(x)$ is the predicted class distribution on the probability simplex.

Feature attribution explainers assign an attribution, also known as saliency or importance, to each input feature, such as a token in a text input. For an instance (x, y), each feature of x is called a fundamental explanation unit (FEU), defined as $u = (x, y, l) \in \mathcal{U}$ with $1 \le l \le L_x$ as the feature index. $e(u) \in \mathcal{E}$ represents the attribution value assigned to it, where $\mathcal{E}$ is the attribution space, such as $[-1, 1]$ for normalized explanations. $e(u_-) = \left(e_x^{(1)}, ..., e_x^{(l-1)}, e_x^{(l+1)}, ..., e_x^{(L_x)}\right) \in \mathcal{E}_-^*$ denotes the explanations on all other FEUs of x.

5.3.2 ExSum Rules

An ExSum rule formalizes a piece of model understanding, such as that for positive words in Figure 5-1, which we use as the running example.

Definition 5.3.1 (ExSum rule)**.** An ExSum rule r is defined by two functions. A binary-valued *applicability function* $a : \mathcal{U} \to \{0, 1\}$ determines whether the rule applies to a given FEU, with 1 being applicable and 0 otherwise. We use $a(\mathcal{U}) = \{u \in \mathcal{U} : a(u) = 1\}$ to denote the *applicability set.* A set-valued *behavior function* is defined as $b : a(\mathcal{U}) \times \mathcal{E}_-^* \to \mathcal{P}(\mathcal{E})$ where $\mathcal{P}(\mathcal{E})$ is the power set (i.e., the set of all subsets) of $\mathcal{E}$. This function predicts a set of possible explanation values for the FEU, called the *behavior range.* The rule is written as $r = \langle a, b \rangle$. We abbreviate $b(u, e(u_-))$ as $b(u)$ and refer to the two functions as a- and b-functions.

For FEU $u = (x, y, l)$, the a-function typically depends only on x_l, but could depend on the entire input x (e.g., for long sentences) or the output y (e.g., for positive class). In our example, it tests whether the sentiment score is greater than that of the word "*memorable*" (0.638). The b-function usually outputs a constant range. Since "*memorable*" has a saliency of 0.479, the range is [0.479, 1.0].

5.3.3 Additional Examples

While we expect most rules to use rather simple a- and b-functions, they can also be more complex with more nuanced aspects. For the following examples, recall that $u = (x, y, l)$. An applicability function can target words only in long sentences using a conjunction with $\text{len}(x) \ge L$, where L is the threshold. We can also target inputs with ambivalent predictions with $\max_c m(x)_c \le 0.6$, where $\max_c m(x)_c$ is the probability of the predicted class. For behavior functions, to indicate the first word of the sentence has higher saliency than the rest, we can define $b(u, e_-) = (\max_{l' \ge 2} e_-^{(l')}, 1.0]$, where the a-function selects the first word (i.e., $a(u) = \mathbb{1}_{l=1}$). Similarly, to describe that an FEU has higher saliency than all the verbs in a sentence, we can can use $b(u, e_-) = \left(\max_{l':\text{is\_verb}(x_{l'})}\{e_-^{(l')}\}, +\infty\right)$.

5.3.4 ExSum Rule Unions

Since a single ExSum rule is designed to capture one aspect of model understanding, multiple rules are often necessary for comprehensive understanding. However, conflicts can occur when multiple rules apply to the same FEU but the b-functions are different. We resolve them by defining the composition of two or more rules into a *rule union.*

Definition 5.3.2 (Precedence-Mode Composition)**.** Two rules, $r = \langle a, b\rangle$ and $r' = \langle a', b'\rangle$, can be composed into a precedence-mode rule union $r^* = r > r'$ defined as $r^* = \langle a^*, b^*\rangle$ where

$$a^*(u) = \mathbb{1}\{a(u) + a'(u) \geq 1\}, \tag{5.1}$$

$$b^*(u) = \begin{cases} b(u) & \text{if } a(u) = 1, \\ b'(u) & \text{if } a(u) = 0 \text{ and } a'(u) = 1, \end{cases} \tag{5.2}$$

represent the a- and b-functions of rule union r^*, with semantics similar to those for rules.

For example, if we want to split positive adjectives into a separate rule from other positive words, we create a rule to test for part-of-speech and sentiment score, and assign a higher precedence to this rule, such that the original rule is only applicable to the remaining non-adjectives. One useful practice is to include a lowest-precedence catch-all rule that covers everything not addressed by other rules, with a constant $a(u) = 1$ function, which leaves no FEUs unaccounted for.

Definition 5.3.3 (Intersection-Mode Composition)**.** Two rules, $r = \langle a, b\rangle$ and $r' = \langle a', b'\rangle$, can be composed into an intersection-mode rule union $r^* = r \,\&\, r'$ defined as $r^* = \langle a^*, b^*\rangle$ where

$$a^*(u) = \mathbb{1}\{a(u) + a'(u) \geq 1\}; \tag{5.3}$$

$$b^*(u) = \begin{cases} b(u) & \text{if } a(u) = 1 \text{ and } a'(u) = 0, \\ b'(u) & \text{if } a(u) = 0 \text{ and } a'(u) = 1, \\ b(u) \cap b'(u) & \text{if } a(u) = a'(u) = 1. \end{cases} \tag{5.4}$$

Unlike precedence-mode, intersection-mode composition is symmetric with respect to the two rules. This mode is helpful when each property of an FEU has a corresponding behavior range, and the final behavior range of an FEU depends on FEU's properties. For example, if verbs have a behavior range of [-0.4, 0.4] and strongly positive words have a behavior range of [0.3, 1], a strongly positive verb would have a behavior range [0.3, 0.4], or the intersection of the two constituent ranges. In our case studies, however, we do not encounter any situations in which intersection-mode compositions were preferable.

Since rule unions are also defined by a- and b-functions, they can form other rule unions in the same way. Recursively, this results in a list of rules composed into a single rule union, written as $r^* = (r_3 > r_1)\ \&\ ((r_4\ \&\ r_2) > r_5)$. This rule union represents our *generalized model understanding.*

5.3.5 Quality Metrics

We propose three metrics for establishing the quality profiles of ExSum rules or rule unions. All of them implicitly depend on the relative weight of each feature that we assign. In ExSum, for an instance with L features, we assign equal weight to each of them. In addition, for two instances with L_1 and L_2 features, we assign the same total weight to them. In other words, the weight of each feature in the first instance is L_2/L_1 times that of each feature in the second instance.

Recall that $u = (x, y, l)$ represents the l-th feature of the data instance $d = (x, y)$. The above weighting notion is captured by the distribution $\mathbb{P}_U$ over $\mathcal{U}$ such that the probability (or probability density) of $u = (x, y, l)$ is $1/L_x$ of that of d under the data distribution $\mathbb{P}_D$. In other words, sampling of u can be performed in two steps: first draw an instance $d = (x, y) \sim \mathbb{P}_D$, then a feature index $l \sim \text{Unif}(\{1, ..., L_x\})$.

Definition 5.3.4 (Coverage)**.** The coverage of a rule (union) $r = \langle a, b\rangle$ is defined as follows:

$$\kappa(r) = \mathbb{E}_{U\sim\mathbb{P}(U)}\left[a(U)\right]. \tag{5.5}$$

This represents the fraction of FEUs that we attempt to understand. While individual rules may have low coverage because they specialize in aspects of the model behavior, we want their union to have high coverage to achieve a comprehensive understanding of the model and prevent model prediction from being excessively affected by the uncovered (i.e., unexplained) input features. For our positive word rule, the coverage is the frequency of those words in the corpus and not surprisingly is only 1.6%. By contrast, including a catch-all rule in the union maxes out its coverage value at 100%.

Definition 5.3.5 (Validity)**.** Let $\mathbb{P}_{a(U)}$ be $\mathbb{P}_U$ truncated to the set of applicable FEUs. The validity of a rule (union) $r = \langle a, b\rangle$ is then defined as follows, capturing the intuitive notion of a “correct” understanding:

$$\nu(r) = \mathbb{E}_{U \sim \mathbb{P}_{a(U)}} \left[\mathbb{1}\{e(U) \in b(U)\} \right]. \tag{5.6}$$

For our example, we compute it as the frequency that the saliency of those words is actually in the range of [0.479, 1] – which turns out to be only 3.1% of the time. However, validity alone is not sufficient, as it increases with wider behavior range. We thus establish sharpness as a competing metric.

Definition 5.3.6 (Sharpness)**.** Let $\mathbb{P}_E$ be the probability measure corresponding to the marginal distribution over explanation values generated by the explainer on $u \sim \mathbb{P}_U$. The sharpness of a rule (union) $r = \langle a, b \rangle$ is defined as follows:

$$\sigma(r) = \mathbb{E}_{U \sim P_{a(U)}} \left[1 - \mathbb{P}_E(b(U)_{\setminus U}) \right], \tag{5.7}$$

where $b(U)_{\setminus U} = b(U) \backslash \{U\}$ removes the actual attribution value U from the behavior range to prevent penalizing sharpness simply because the attribution value is very common (e.g., zero for sparse explanations), in which case $\mathbb{P}_E$ is discrete at U.

Sharpness represents precision in the understanding, as $1 - \sigma(r)$ gives the probability that a random FEU explanation value is correct. Thus, a lack of precision represented by a wide behavior range has minimal sharpness. We use the probability measure $\mathbb{P}_E$ to define the “size,” as it is consistent across all explanation distributions, most of which are non-uniform. A more general interpretation of sharpness is the consistency of the described model behavior: if a behavior range is wide (e.g., containing very positive *and* negative saliencies), then it is less sharp, and hence less useful. $\mathbb{P}_E$ could be replaced by an application-specific diversity measure, though the precision notion may be lost.

There is generally a trade-off between validity and sharpness, as more precise rules (i.e., those with narrower behavior ranges) are less likely to be valid. For our rule, the probability of a *random* word saliency being in [0.479, 1.0] is 0.2%, indicating that explanation values are rarely higher than 0.479. This makes sharpness very high at 99.8%. However, the rule is not useful because of its low validity; i.e., it is almost never correct. By comparison, the looser range of [-0.01, 1.0] has 90.4% validity but 28.6% sharpness. There is another trade-off between coverage and the two, since a larger set of covered FEUs tends to be more diverse, making it harder to write a b-function that remains as valid and sharp simultaneously.

Since these metrics are all expected values, we can estimate them by their empirical estimate from a dataset (i.e., a simple average), and $\mathbb{P}_E$ can be constructed by kernel density estimation.

5.4 ExSum Development Process and GUI

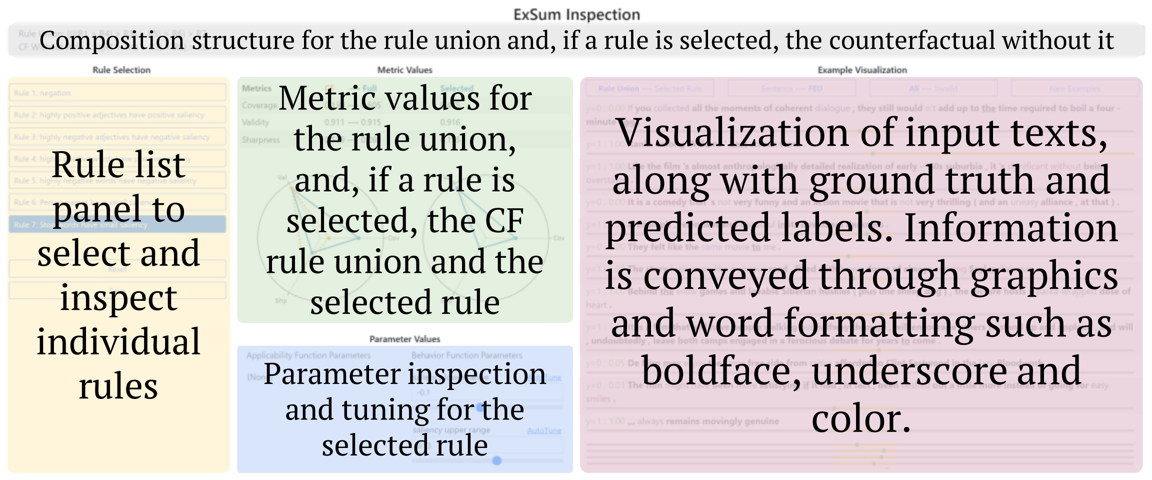

ExSum Inspection
Composition structure for the rule union and, if a rule is selected, the counterfactual without it
Rule Selection
Metric Values
Example Visualization
Rule list panel to select and inspect individual rules
Metric values for the rule union, and, if a rule is selected, the CF rule union and the selected rule
Parameter Values
Parameter inspection and tuning for the selected rule
Visualization of input texts, along with ground truth and predicted labels. Information is conveyed through graphics and word formatting such as boldface, underscore and color.

Figure 5-2: ExSum inspection GUI.

We describe a systematic procedure for authoring ExSum rule unions from scratch and utilize it in Section 5.5. Starting from an empty rule union with no FEUs covered, we iteratively create rules that target uncovered FEUs. Each rule describes one model behavior, such as that for highly positive words. For a rule, the a- and b-functions need to be defined, which may involve setting and tuning parameters, such as the sentiment threshold. Last, we add a lowest precedence catch-all rule if any FEUs remain uncovered. During this process, we may also merge or split rules and change the composition structure according to the metric values.

To support these steps, we developed a Python Flask-based [54] graphical user interface (GUI, Figure 5-2). Users can visualize the FEUs, with font formatting for their coverage and validity. Users can also filter for uncovered or invalid FEUs, iteratively constructing and refining the rule union. ExSum rule definitions usually include parameters such as the sentiment threshold. Manually selecting correct values for the parameters is tedious, so the lower middle panel of the GUI implements automatic parameter tuning for a given target metric value. Installation and usage instructions for the GUI are available on the project page[1].

[1]`https://yilunzhou.github.io/exsum/`

5.5 Evaluation

We construct ExSum rule unions for SST and QQP models. Table 5.1 summarizes the key parameters of our experiment. Both saved models are publicly accessible from Huggingface Hub, and the model names in the table are links to the respective model checkpoints. For normalization, we divided all explanation values for all test set instances by a single scaling factor such that the maximum magnitude of new explanations is 1.

Task	Dataset	Model	Acc.	F1	Explainer
Sentiment	SST-2 [144]	RoBERTa [98]	95.6%	0.957	SHAP [99]
Paraphrase	QQP [71]	BERT [38]	90.7%	0.875	LIME [128]

Table 5.1: A summary of tasks, models (fine-tuned on respective datasets), and explainers for the two case studies.

We split the test set into a *construction set* to create the rule union and tune its parameters (analogous to the training and validation set in supervised model training) and an *evaluation set* to compute unbiased estimates of the metric values (analogous to the test set).

5.5.1 Sentiment Classification

Setup Figure 5-3 shows the SHAP explanations on three sentences (after normalization).

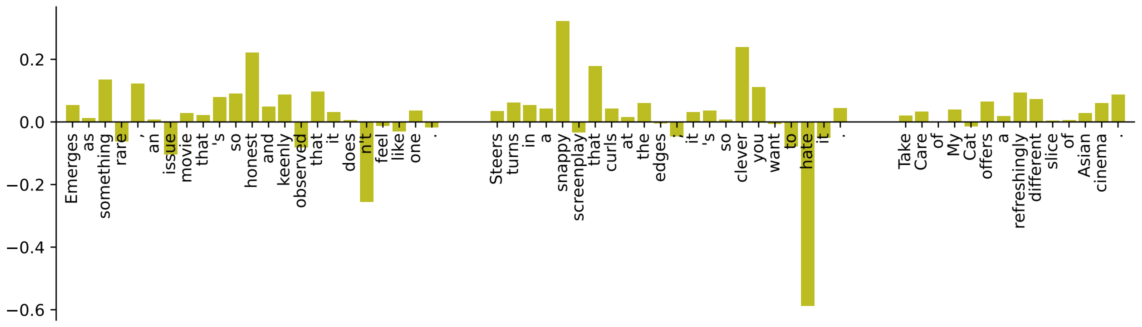


Figure 5-3: SHAP explanations for three SST inputs.

We take 300 random sentences as the construction set, with the remaining 1910 sentences as the evaluation set. We compute five features for each FEU: sentiment score, part of speech (POS), named entity recognition (NER), dependency tag (DEP) and word frequency. For example, the word "same" in the sentence "*They felt like*

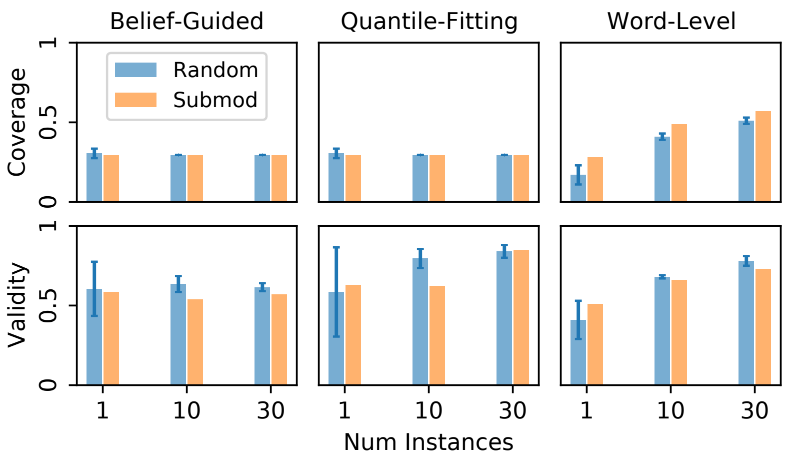


Figure 5-4: Coverage and validity metrics for the three current practice modes. Table 5.5 of Appendix 5.9.2 presents the complete numerical data (also with sharpness).

the <u>*same*</u> *movie to me.*" has sentiment score of 0.028, POS = ADJ, NER = O, DEP = amod, and frequency of 7.14e-4, with SHAP saliency of -0.82.

Current Practice We evaluate the current practice of extracting *informal* model understanding from local explanation inspection against the three metrics. We assess three values of K, the number of inspected instances: 1, for the typical *ad hoc* setting of generalization from a single explanation, 10, for a more careful investigation, and 30, which is quite cumbersome for manual inspection. These examples are selected either randomly or by submodular pick [128]. Next, we consider three ways to extract model understanding – belief-guided (BG), quantile-fitting (QF) and word-level (WL) – and apply them to create rules on strongly positive words and stop words introduced in Section 7.1. For the strongly positive word rule, BG mandates that words more positive than the average sentiment score should have an above-average saliency score, representing the belief of a positive correlation between the two. For the stop word rule, a saliency range belief of [-0.05, 0.05] is averaged with the observed range. For both rules, QF extracts the 5%-95% quantile interval of the saliencies for words covered by the respective rule. WL, by contrast, creates a behavior range *for each word seen*, with 0.03 margin on both sides. Appendix 5.9.2 presents technical details for these.

We formalize the understanding derived from the selected instances and plot their coverage and validity metrics on the evaluation set in Figure 5-4. For BG and QF, the bars represent the average metric value of the positive word and stop word rules. For WL, the bars represent the metric for the rule union consisting of an individual

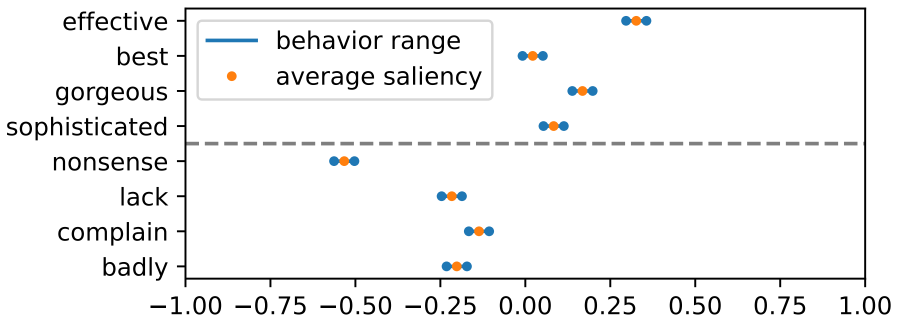


Figure 5-5: Behavior ranges can vary widely and unpredictably on similar words for WL rules.

rule for each unique word. Error bars for the random pick represent the standard deviation across five iterations. Table 5.5 of Appendix 5.9.2 presents the complete statistics for all metric values, and we highlight several findings.

- A very small number of samples (e.g., 1) exhibit large variance for random pick, and low validity for both pick methods. This confirms the intuition that model understanding from very few explanations should be avoided.
- BG overall yields low validity, because its "beliefs" turn out to be quite incorrect. This suggests a strong prior belief about how the model works could lead to incorrect conclusions.
- While submodular pick can select a more diverse set of words, to the particular benefit of the coverage of WL[2], its validity is generally lower due to under-representation of common words.
- Although WL achieves highest coverage *and* validity, it has > 500 rules at K=30, with similar words having very different ranges, as shown in Figure 5-5 – a conglomerate (almost) impossible to make sense of. It also overfits, as the evaluation set validity is much lower than the construction set validity (which is 100% by construction).
- At K=10, only the stop word rule with random pick QF achieves validity $> 80\%$, indicating that even the more careful practices are unreliable.

All the drawbacks call for a principled way to derive robust model understanding with enforceable metric values (e.g., validity). As we demonstrate next, given a large construction set and automatic parameter tuning assistance, we can create such a ExSum rule union. Finally, as a meta-point, the above discussion above of various

[2]The other two are less affected because the subject of the rule (e.g., stop words) largely dictates which words it covers.

Idx	Rule	Cov%	Val%	Shp%
1	Negation	1.2	89.5	65.1
2	Strongly neg. adjectives	3.2	91.6	83.5
3	Strongly pos. words	5.1	91.9	40.0
4	Strongly neg. non-adjectives	1.2	89.9	71.4
5	Person name	2.4	90.9	28.4
6	Stop words	47.5	90.8	23.5
7	Zero-sentiment words	17.1	90.0	15.6
8	Weakly pos. words	15.4	91.2	11.3
9	Weakly neg. words	5.7	91.7	31.4
Union	On construction set	100	90.7	26.1
	On evaluation set	100	89.4	26.2

Table 5.2: SST rules, rule union and their metric values.

limitations would not be possible without the proposed ExSum formalization and metric definitions.

ExSum Construction We create a rule union consisting of nine rules, with target validity of 90% and tune the sharpness accordingly. Table 5.2 summarizes the individual and aggregate metrics.

Clearly, high validity comes at the cost of low sharpness. Since (1 − sharpness) is the probability that a random FEU has an explanation value within the behavior range, this around 90.7% validity should be put into a context where the random baseline achieves a validity of around 75%. In this sense, we attain only a crude understanding of the local explanations that misses many subtleties.

Nonetheless, Rule 3 (strongly positive words) and Rule 6 (stop words) achieve better validity-sharpness trade-off than their counterparts created using the *ad hoc* BG and QF methods above. Moreover, the WL rules cover all words seen in the analyzed instances – analogous, in a sense, to our ExSum rule union. While the validity-sharpness trade-off is comparable between the two, ours has 100% coverage due to the effectively "catch-all" Rule 7, while WL rules have less than 60%. Most importantly, as our rule union is composed of nine semantically organized rules, it is much more interpretable than WL, which include more than 500 unpredictably varying rules (Figure 5-5).

The fact that the ExSum rule union reveals the imprecision and limitations of our model understanding while still performing better than current practice emphasizes the need for more formal and quantitative model understanding, as well as the development of methods that are easier to *understand*, in addition to being correct. These rules are explained below in more detail.

- ***Rule 1: Negation words have negative saliency.*** We found that negation words – *not*, *n't*, *no*, *nothing* and those with NEG dependency tag – almost invariably receive (sometimes highly) negative saliency, regardless of the sentence label or sentiment of the word being modified. We create a rule that predicts a constant behavior range $[-1.0, 0.002]$, with 89.5% validity and 65.1% sharpness. Although the validity is under our 90% target, we found that to make it higher, the upper limit of the behavior range needs to be 0.1, which results in an extremely low sharpness of 11%. Thus, we decided against it.

- ***Rule 2, 3, 4, 8, 9: Sentiment-carrying words.*** We expect a sentiment classifier to recognize sentiment-laden words. To test our intuition, we create rules for positive and negative words, and further split each set of words into two according to sentiment strength, resulting in four rules. For the two rules on strong words, we find that wide behavior ranges of [0.01, 1] and [-1, -0.01] are necessary to achieve 90% validity, suggesting the looseness of the model understanding. However, we do observe that negative adjectives (but not positive ones) are modeled much better, where a range of [-1, -0.06] is sufficient for the same validity. Thus, we create a separate Rule 2, with very high sharpness of 84.2%. For the two rules on words of weaker sentiment, even wider ranges of [-0.11, 1] and [-1, 0.05] are necessary. Since both ranges encroach upon the other side, the model often considers these words to have an impact opposite to their intrinsic meaning, but we fail to extract further understanding. In addition, negative rules are much sharper than positive ones, suggesting that the model considers a negative word to be stronger evidence for a negative prediction than its positive counterpart.

- ***Rule 5: Person names have positive saliency.*** During our initial inspection, we found several cases where the name of a person (e.g., director or actor) have positive saliency values. Thus, we create this rule from the NER tag, covering 2.3% of words. However, after parameter tuning, we found that while many of the words have positive saliency, the correct characterization is that they all have small saliency values, as a behavior range of $[-0.06, 0.1]$ achieves 91.6% validity. However, since SHAP saliencies are mostly concentrated around 0, this range achieves a meager sharpness of 26.8%. Despite this, we still decide to keep it.

- ***Rule 6: Stop words.*** While stop words (e.g., "the", "of") *should* have negligible impact on prediction (and saliency values close to zero), a narrow behavior range of [-0.05, 0.05] only has 64% validity. We create this rule for all stop words with

90% target validity and use different ranges on different words for better sharpness. On average, we get [-0.07, 0.12], demonstrating that they can sometimes be more influential than even strong sentiment words. The ranges also tilt to the positive side, uncovering a grammaticality bias wherein prediction is more negative for grammatically incorrect sentences with stop words masked out by SHAP.

- ***Rule 7: Zero-sentiment words have small saliency.*** Besides stop words, we should expect words that do not carry sentiment, such as most nouns and verbs (e.g., *movie* and *get*), to have small saliency magnitudes. Due to the wide range of words applicable under this rule, we choose the saliency range to be $[-0.15, 0.15]$ for $\geq 90\%$ validity, but this range yields lowest sharpness of 13.5%.

5.5.2 Paraphrase Detection

Setup Figure 5-6 shows the LIME explanations on two sentences (after normalization).

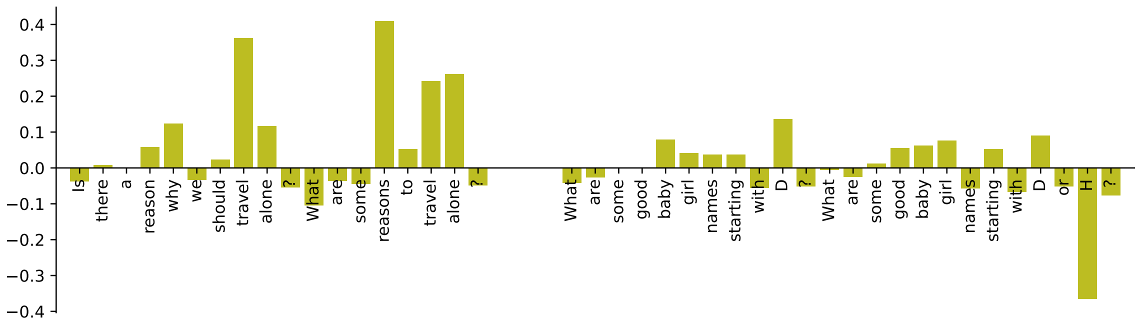


Figure 5-6: LIME explanations for two QQP pairs.

We take 500 random test set sentences as the construction set and the remaining ≈ 40k as the evaluation set. We remove the word sentiment feature but add the question ID (1 or 2) of each FEU.

ExSum Construction QQP is a more complex domain than SST, since the label is the semantic equivalence of *two* sentences. The metric values for the ExSum are summarized in Table 5.3. Below, we describe how expectations for the model are validated, but a hidden – and somewhat surprising – phenomenon is also uncovered.

- ***Rule 1, 2: Matching words.*** Due to the nature of the task, we expect the model to rely heavily on matching words. For such a word u, defined as (proper) noun, verb, adjective or pronoun that has exactly one case-insensitive match v in the other question, we expect similar saliency to their match due to symmetry,

Idx	Rule	Cov%	Val%	Shp%
1	Matching words neg. prediction	11.7	90.9	39.5
2	Matching words pos. prediction	12.4	90.3	38.6
3	Non-matching words neg. prediction	18.7	90.0	35.5
4	Question mark neg. prediction	5.2	90.2	36.5
5	Question mark pos. prediction	3.8	90.0	23.1
6	Stop words neg. prediction	22.3	90.0	32.8
7	Stop words pos. prediction	12.6	90.5	12.5
8	Negation words neg. prediction	0.3	90.0	36.0
9	Negation words pos. prediction	0.1	95.7	7.2
10	All else neg. prediction	4.0	92.1	23.5
11	All else pos. prediction	8.8	90.3	5.7
Union	On construction set	100	90.3	29.3
	On evaluation set	100	90.0	29.1
Word Avg	On construction set	100	90.8	29.4
	On evaluation set	82.3	84.4	29.4

Table 5.3: QQP rules, rule union and their metric values. The last two rows are for the baseline at the end of Section 5.5.2.

or formally its saliency $s_u \in [s_v - \alpha, s_v + \beta]$, where α and β are lower and upper margins. This behavior function is non-constant, with output depending on the saliency values of other words in the sentence.

For the same margin, FEUs for pairs of negative predictions have much higher validity than positive ones, so we split the rule into two based on the prediction. Despite a less than 1% difference in sharpness (Table 5.3), we have $\alpha = \beta = 0.07$ for the negative rule, but 0.18 for the positive rule, suggesting that the matching words make a much larger and more unpredictable contribution to positive predictions. Interestingly, all other rules had wider intervals for positive predictions as well.

- ***Rule 3: Non-matching words.*** Next we study model behaviors for non-matching words, defined analogously to matching ones. Following the previous split based on predicted label, we designed two rules. The negative rule has a reasonably sharp behavior range of [-0.35, 0.01] at 90% validity. Given that LIME saliency is the linear regression coefficient on a neighborhood created by word erasure, we conclude that the *presence* of these non-matching words mostly causes the prediction to tilt toward the non-paraphrase (i.e., negative) class, indeed a very reasonable behavior. However, we cannot find a range with 10% sharpness at 90% validity for the positive rule and thus discard it.

- ***Rule 4, 5: Saliency for trailing question marks.*** Since the dataset is composed of pairs of questions, the vast majority of sentences conclude with question marks. These should be purely decorative and syntactic, and so should have small saliency, similar to stop words. However, we observe that the saliencies assigned to them for positive and negative predictions are very different, so we create two rules for these two cases. With a 90% validity target, the saliency range is [-0.04, 0.03] for negative predictions and [-0.07, 0.06] for positive predictions. Again, the saliencies for positive predictions demonstrate more variation than those for negative ones.
- ***Rule 6, 7: Saliency for stop words.*** Similar to SST, we use these two rules to ensure stop words should *not* be influential. We split the stop word group into finer segments by part of speech, to achieve higher sharpness. On average, the range is [-0.07, 0.03] for negative predictions and [-0.09, 0.1]for positive predictions, which again demonstrate a much higher degree of variation.
- ***Rule 8. 9: Saliency for negation words.*** In the SST case, we found that negation words typically have negative saliency regardless of the sentiment label, and test whether this holds for QQP as well. Following on our previous findings, we use two rules to separately model inputs of positive and negative predictions. We find that the range is [-0.1, 0.24] for positive predictions and [-0.21, 0.01] for that for negative predictions. Curiously, the same negative saliency trend is preserved here as well, but only for inputs with negative predictions.
- ***Rule 10, 11: Saliency for everything else.*** Finally, we designed two lowest-precedence "catch-all" rules to complete the coverage. The range for positive prediction FEUs is [-0.13, 0.25]. For negative prediction inputs, we find that breaking them according to different parts of speech (nouns, verbs, adjectives, and everything else) is helpful, with verbs having a particularly narrow saliency range of [-0.05, 0.05]. On average, the saliency range is approximately [-0.09, 0.05].

Regarding the sharpness contrast by predicted label, one explanation is that the model defaults to a negative prediction, since many negative pairs consist of completely unrelated questions and the model decision is largely insensitive to input perturbations, leading to stable LIME coefficients. On the other hand, a positive prediction requires the cooperation of all parts of both questions. Depending on the exact sentence structure, the importance of each word to the match are different and hard to predict, which prevents the rules from being sharp.

Word Average Baseline We introduce a new baseline as an "automated" version of WL rules in SST. Specifically, for each word in the construction set, we compute a behavior range around its average saliency, with sharpness of 29.4% (matching that of our ExSum rule union). As Table 5.3 shows, the resulting rule union is much worse than our manual one on both evaluation set coverage and validity, which is not surprising as the word saliency *should* be more context-dependent, due to the matching mechanism of paraphrase detection. Moreover, with more than 2,000 constituent rules, the rule union barely qualifies as any sort of *generalized* model understanding.

5.6 Related Work

As discussed in Section 5.1, explanation evaluation usually has a focus on correctness (or faithfulness) – i.e., whether the explanation truly reflects the model's reasoning process. This includes sanity checks [1], proxy metrics [9, 136], and explicit ground truth [182]. The understandability issue has been much less studied, with the exception by Zheng et al. [172], who proposed an evaluation specifically for rationale models [94]. ExSum, however, addresses *post hoc* explanations of general black-box models.

In addition, a few prior works have attempted to capture the "end-to-end" utility of explanations: whether access to explanations leads to performance increase in certain tasks. Hase and Bansal [57] proposed a model-teaching-human setup, subsequently extended by Pruthi et al. [123] into an automated evaluation procedure. Bansal et al. [14] studied whether explanations can improve human-machine teaming performance. While these studies report mostly negative results, pinpointing the root cause is difficult due to their end-to-end nature. Poor *understanding* of the explanations may be a major reason, as indicated by ExSum.

Last, some authors proposed methods for understanding model predictions beyond individual instances. For example, the anchor method [129] generates an explicit domain of applicability for each explanation, while Lakkaraju et al. [89] and Lakkaraju et al. [90] proposed to learn "patches" of the input space specified by logical predicates (e.g., *if* blood pressure is above 160/100 and the patient has cancer, *then* the mortality risk is high). ExSum also emphasizes the need to understand models that generalizes across instances, and uses logical predicates in the formulation, but the central difference is its focus on model understanding via *explanations* (e.g., if blood pressure is above 160/100, then it has a high impact on the mortality prediction), instead of direct *predictions*. In other words, ExSum tries to explain why particular features have respective importance according to some local explanation method, rather than why a particular prediction is made. There are three reasons for this.

First, a focus on explanation can capture a wider variety of *behaviors*. For example, for the QQP model, we found that matching words to have similar impact, *regardless of the model prediction*. This insight is easy to formalize with ExSum but hard to describe when we only focus on the model prediction. Furthermore, ExSum works with arbitrary black-box models for model prediction and local explainers for model explanation, allowing it to take advantage of advances in both parts, while logical predicate classifiers [e.g., 89] are mostly limited to tabular data and essentially impossible to learn for image and text domains. Last, from a practical perspective, since people use local explanations such as LIME and SHAP to analyze their models anyway, ExSum provides a principled way of for such analysis that we demonstrate to be much better compared to current *ad hoc* approaches.

5.7 The Many Faces of Understandability

The central thesis of ExSum is quite simple and intuitive: in order to understand a model from local explanations, we need to understand those local explanations. While ExSum is the first framework to explicitly formalize and quantify the notion of understandability, we argue that it is connected to many often-discussed and desirable properties of explanation. Crucially, all of these properties can be demonstrated to be in conflict with correctness, in that the search for the most "correct" explanation should disregard these properties. From this perspective, we put forth the property of understandability as one orthogonal to correctness, while being equally important.

5.7.1 Human Alignment

Many prior works have assessed how much explanations agree with human expectation. For example, Li et al. [96] observed that the word "hate " contributes the most to a negative sentiment prediction in many inputs, and used it to argue the explanation is correct. In a similar sentiment classification task, Bastings et al. [15] used the high degree of overlap between the extracted rationale and strong-sentiment words to argue the superior quality of a neural rationale model [94]. In computer vision, this alignment is often implemented as a pointing game that computes the intersection-over-union (IoU) metric between the salient region and the semantic segmentation mask of the predicted class [44, 142], as shown in Figure 5-7. For a model that predicts breast cancer onset using patients' genetic information, Covert et al. [33] demonstrated that many of the influential genes identified by their explainer were indeed known to be associated with the disease.

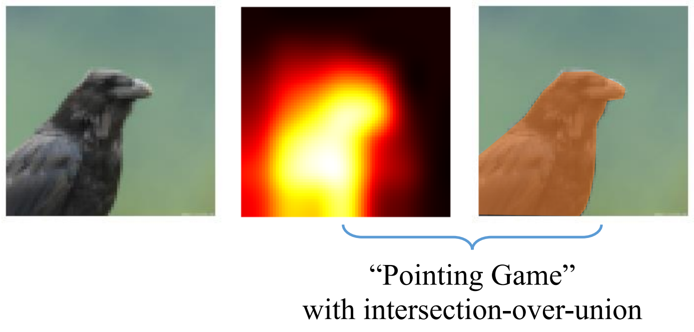


Figure 5-7: A pointing game for quantifying human alignment in visual explanations.

As discussed in Chapter 4, models could use any unexpected spurious correlation, such as the green background in Figure 5-7. For these models, correct explanations should have low alignment scores. When correctness (or faithfulness) is the sole desideratum of interpretability methods, it is unclear what purposes these alignment evaluations serve. Some authors [e.g. 73] have even argued they are fundamentally misleading and flawed in nature as they focus on *plausibility*, which is sometimes at odds with the goal of correctness.

However, from the perspective of *understandability*, high-alignment explanations are arguably very understandable, simply because they align closely with human expectation. Thus, given the same level of correctness, a higher-alignment explainer may be preferable.

5.7.2 Robustness

Besides human alignment, robustness – i.e., that similar inputs should have similar explanations – is also argued to be a favorable property for explanation. For example, Ghorbani et al. [49] argued that explanations are fragile due to their adversarial vulnerability, Alvarez-Melis and Jaakkola [4] empirically estimated the Lipschitz constant for many explainers, and Alvarez-Melis and Jaakkola [5] proposed an inherently interpretable model that is explicitly regularized for explanation robustness.

Robustness generally conflicts with correctness. If, for two inputs, the model is using distinct reasoning patterns, the correct explainer should faithfully report distinct explanations for them. One straightforward example is the decision tree model shown in Figure 5-8, where the root node splits on the second feature at a threshold value of 3. For two inputs x_1 and x_2 that agree on all features except the second one, with $x_1^{(2)} = 2.99$ and $x_2^{(2)} = 3.01$, since they are sent down two different sub-trees at the

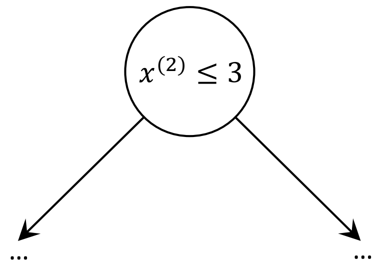


Figure 5-8: A decision tree that splits on the second feature at the root node.

very beginning, the model is likely to use for totally different features.

Nonetheless, as implicitly argued by the works above, erratic model behaviors are less understandable because they make it more difficult to identify generalizable patterns compared with slowly varying explanations in the input space. Thus, robustness is another aspect of the same understandability desideratum.

5.7.3 Counterfactual Similarity and Plausibility

Unlike feature attribution explainers that assign importance to individual features, counterfactual (CF) explainers [e.g., 131] directly generate whole inputs but for a target predicted class. Thus, a CF explanation indicates how to cross the decision boundary from the input.

Naturally, the fundamental requirement of CF explanations is achieving the target prediction, which is typically known as validity. However, this is trivially satisfiable by simply finding a training instance with the target prediction, along with other ways such as creating adversarially perturbed or nonsensical inputs. Thus, two additional requirements are often enforced: similarity and plausibility. The former says that the CF explanation should be close to the original input (with regard to, for example, edit distance), and the latter says the CF explanation should be plausible, or natural. Table 5.4 depicts various CF explanations and their satisfaction of the three requirements.

Validity for CF can be considered as the correctness analogy for feature attribution, but the purposes of similarity and plausibility are not readily apparent. As CF explanations represent ways to cross the decision boundary, people need to meaningfully understand how the CF instance is related to the original input. It is difficult to relate two dissimilar instances, and an implausible CF instance is generally unexpected. Thus, similarity and plausibility are required to make CF explanations more understandable.

Input: This restaurant is the best I have been, with especially great food.				
CF	Type	Val.	Sim.	Plau.
This restaurant is the *worst* I have been, with especially *terrible* food.	"good" CF	✓	✓	✓
Rude service!	training set look-up	✓	✗	✓
This *resturant* is the best I have been, with especially great food.	adversarial typo injection	✓	✓	✗
Fjwpeaf fawekl fka erj sfdlk erjlm adl erio fd	nonsensical inputs	✓	✗	✗

Table 5.4: Counterfactual explanations that are all valid but differ in similarity and plausibility metrics.

Interestingly, if our true goal is the understandability of the relationship between the input and its CF explanation, there are cases where similarity or plausibility is *not* desirable. First, consider a sentence length classifier that predicts positive for sentences of at least 10 words, and negative otherwise. Given an input of three words, the CF explainer should generate *dissimilar* CF instances of at least 10 words in order to correctly illustrate the decision boundary, while instances of even more words would be helpful for understanding the "at least 10 words" logic. Second, consider a classifier trained on a typo-free dataset and having high probability of making mistakes on inputs that contain typos. To illustrate this behavior, CF explanations should contain randomly (not adversarially) injected typos, which are *implausible*, but useful as long as the typo injection is understood by people.

5.8 Discussion and Conclusion

Traditionally, model explanations are evaluated on correctness (or faithfulness), i.e., whether they correspond to how models actually make predictions, e.g., reliance on spurious correlations [3, 182]. Such evaluation, however, does not answer the equally important question of whether these (presumably correct) explanations are understandable. Even faithful explanations can lead users into error, if misunderstood (e.g., trusting a model incorrectly).

In a sense, the most correct explanation for an input is the literal trace of model computation, but it is also arguably the least understandable (or useful). As we abstract away from low-level details and use higher-level concepts such as word sentiment, the resulting explanation loses correctness but gains understandability. At the other extreme are explanations that are trivially understandable but completely wrong, such all attribution values being 0 (i.e., no feature impacts the model prediction). Thus, a trade-off often occurs between these two desiderata, and we need to choose a sweet spot.

Concretely, we propose ExSum rules and rule unions, along with three quality metrics to formalize and evaluate understandability. Such rigorous investigations stand in contrast to current *ad hoc* practices, which are prone to yielding unreliable and coarse model understanding. For SST and QQP datasets, ExSum demonstrates that our model understanding is quite limited and imprecise, even with very reasonable explanations. *Being aware of this is an asset.* While ExSum helps us to recognize that our understanding is incomplete, it still helps uncover unexpected model behaviors that warrant further investigation.

5.9 Appendix

5.9.1 Real World Use Cases for Explanations

Here, we discuss several scenarios in which people use local explanations to understand models, and argue that people invariably derive *generalized* model understanding from these explanations.

Spurious Correlation Identification Natural datasets can contain many spurious correlations. For example, in a COVID-19 chest X-ray dataset, most positive images (i.e., patients diagnosed with COVID-19) come from a pneumonia-specializing hospital and contain a watermark of the hospital name, while most negative images from other hospitals do not. Thus, a model could achieve very high accuracy by simply detecting the watermark rather than genuine medical signals. Similar spurious correlations could also be present in the text domain, such as the correlation between an exclamation mark and the positive sentiment class, or between the word "*not*" and the contradiction class in natural language inference.

It is crucial for people to be aware of the shortcuts that models may take, and one possible way to highlight such behaviors is via feature attribution, which in the examples above would assign an abnormally high score to the watermark region, exclamation mark, or the word "*not*." Assuming the explanations do indeed exhibit such patterns, when people claim a model relies on spurious correlation, they mean this in a general sense: for example, the model is likely to focus on the watermark in *any* image that contains it, rather than in only a specific set of images.

Fairness Assurance Similar to spurious correlation features, other features should not have a high impact, but for reasons of fairness. For example, decisions made by a loan approval model should not be affected by gender[3], therefore the gender feature should not have a high attribution score.

If we observe that the gender of one applicant heavily impacts the model's decision, we may suspect the model is discriminative; conversely, observing that it has minimal impact could increase our assurance of the model's fairness. However, such single-instance observations are fundamentally exploratory, and claims about the model's fairness or discrimination must be established using a *population* of instances to determine whether the trend persists generally.

Model-Guided Human Learning In some cases, a very accurate and "super-human" model could be a source for knowledge discovery. Consider the task of early-stage cancer detection from CT scans, which is challenging for doctors. If a label is generated from follow-up visits tracking whether patients develop cancer after a certain number of years, a model achieving better test accuracy than doctors is likely to use certain cues that would be missed by humans or not known to be linked to cancer.

For these models, explanation methods such as saliency maps could be used to help doctors make better diagnoses, or assist scientists in the creation of new pathological theories. Similarly to the above two use cases, *generalized* model understanding across different inputs are necessary, because doctors need to apply what they have learned to new patients, and scientists require new theories to hold broadly.

5.9.2 Current Practices for SST

Here, we provide an extended description of the three current practices, and how they are applied on the handful of selected examples, collectively called the "sample" below.

The first method, "belief guided" (BG), represents the practice wherein the user has some expectations (or beliefs) about the attributions of certain words, and modifies (or updates) them after observing explanations on some actual test inputs. It operates differently for the two rules on positive-sentiment and stop words, as follows.

1. For positive-sentiment words, the prior belief is that a word with a higher sentiment score (one of the FEU features provided by the SST dataset) should also receive

[3]There could be other features that correlate with gender, such as job title, but we ignore such possibilities for simplicity.

more positive attribution. This leads to a rule that applies to all words with a sentiment score greater than α, and has a behavior function that outputs a constant range of $[\beta, 1]$ (recall that SHAP values are normalized to $[-1, 1]$). It then computes the value of α as the mean sentiment score and β as the mean attribution value for all words in the sample with positive sentiment scores.

2. For the stop words – defined as those with parts of speech AUX, DET, ADP, CCONJ, SCONJ, PRON, PART, and PUNCT – it has a prior belief that they should have a attribution value range of [-0.05, 0.05] (i.e., not important to model prediction), and computes the observed attribution range $[\alpha, \beta]$ for stop words in the sample. The final behavior range as predicted by the behavior function of this rule is the average of these two: $[-(0.05 + \alpha)/2, (0.05 + \beta)/2]$.

The second method, "quantile fitting" (QF), represents the practice wherein the user fully follows the observed data without any prior beliefs. Specifically, for a set of words, it collects all attribution values for words within the set and then creates a rule that applies to this set, with the behavior function predicting a constant range of 5% to 95% quantile of these attribution values. For the two rules for positive-sentiment and stop words, the set of words (and hence the applicability functions) is defined in the same way as for the BG method above.

The last method, "word-level" (WL), can be considered a more extreme version of QF, where the user not only lacks any prior expectations for the explanations but also considers each word individually. For example, if the user observes that the word "brilliant" has an attribution value of 0.5 in one sentence and the word "fantastic" has attribution of 0.8 in another, they would *not* conclude that other, similarly positive words would have attributions approximately within the range of $[0.5, 0.8]$. Specifically, for every distinct word w in the sample, this method builds a rule that applies only to that word, with a constant behavior function that outputs a range of $[\min(s_w) - 0.03, \max(s_w) + 0.03]$, where s_w is the list of attributions received by different occurrences of w. In many cases, especially given a small sample, word w only appears once, in which case s_w is a list containing only that attribution value.

Table 5.5 presents the metric values of the above methods. Figure 5-4 of Section 5.5.1 depicts a graphic summary.

5.9.3 ExSum for Instance-Based Explanations

In this section, we describe our initial attempt at extending the ExSum framework to another type of explanations: instance-based explanations (IBE). The IBE for

		belief-guided		quantile-fitting		word-level
K	pick	positive	stop word	positive	stop word	seen words
	SP	10, 72, 50	49, 45, 65	10, 63, 44	49, 63, 45	28, 51, 61
1	RND μ	12, 63, 57	49, 58, 53	12, 45, 56	49, 73, 33	17, 41, 68
	RND σ	6, 25, 27	0, 9, 6	6, 32, 29	0, 24, 21	6, 12, 10
	SP	10, 61, 61	49, 47, 63	10, 78, 34	49, 72, 38	49, 66, 48
10	RND μ	10, 71, 52	49, 56, 56	10, 75, 32	49, 84, 25	41, 68, 48
	RND σ	0, 6, 7	0, 4, 4	0, 9, 10	0, 3, 2	2, 1, 2
	SP	10, 64, 59	49, 50, 60	10, 88, 17	49, 82, 29	57, 73, 42
30	RND μ	10, 66, 56	49, 57, 55	10, 82, 26	49, 86, 24	51, 78, 39
	RND σ	0, 4, 5	0, 1, 2	0, 6, 7	0, 2, 2	2, 3, 2

Table 5.5: Coverage, validity, and sharpness (percentage) of model understanding with *ad hoc* current practice. "SP" refers to the submodular pick procedure, and "RND" refers to the random sampling procedure. The latter also shows mean μ and stdev σ across five runs.

Type	$b(\widehat{y})$	ν	σ
Entity change	$\widehat{y} \pm 0.05$	91.4	56.5
Minor insert	$\widehat{y} \pm 0.05$	89.1	57.3
Negation	other-side($\widehat{y}$)	30.4	50.0
Negation	same-side($\widehat{y}$)	69.6	49.9
Negation ($\leq$ 6 words)	other-side($\widehat{y}$)	56.2	49.7

Table 5.6: Instance-based explanation metrics on SST.

an input x is a set of instances and their predictions $\{(x_i, \widehat{y}_i)\}$, where x and x_i are semantically related (e.g., negation). We define $\widehat{y}_i$ as the predicted probability of positive class.

We use POLYJUICE [PJ, 165] to generate instances of three semantic operations. *Entity change* replaces a proper noun (e.g., actor name) with another using "lexical" mode of PJ. *Minor insert* adds a short text to the sentence using "insert" mode. *Negation* generated a negated version of the input using "negation" mode. For each operation type, our expectation for model behavior is formalized as a range $b(\widehat{y})$ on $\widehat{y}$. We expect the prediction to be unchanged by the first two operations allowing for a margin of 0.05, but changed to the other side of 0.5 by negation. We then define validity $\nu = \mathbb{E}_{\widehat{Y},\widehat{Y}_i}\left[\mathbb{1}_{\widehat{Y}_i \in b(\widehat{Y})}\right]$ and sharpness $\sigma = 1 - \mathbb{P}_{\widehat{Y}}[b(\widehat{Y})]$ analogously.

Table 5.6 summarizes the results. While our expectation is mostly confirmed for entity change and minor insert, it is notably violated in the case of negation, with only 30.4% validity, indicating model prediction is on the *same* side 69.6% of time. Upon further evaluation, we find that validity drops with sentence length, with short sentences of six words or fewer having much higher validity (for other-side). Since the PJ rewriting model is learned rather than manually defined and negation is more complex than the other two operations, there are two failure modes, as presented in Table 5.7. In the first, a negation is applied to the input sentence, but on a part irrelevant to the sentiment. In the second, the generated sentence is not a negation of the input by any reasonable standard.

These examples highlight the importance of clearly defining the operation: rather than a generic negation, we would need the negation to happen on the "sentiment-carrying" part. It is also crucial to ensure that the generator is of a high quality in order to minimize the chance of generating nonsensical outputs. Despite many advances in generative language modeling, it have been shown to be undesirable in many ways [e.g., 66], all of which affect the quality of the counterfactual explanation.

At a high level, IBE explains the local prediction by illustrating ways to cross (e.g.,

Input sentence	"Negated" sentence
Human Nature initially succeeds by allowing itself to go crazy , but ultimately fails by spinning out of control .	Human Nature initially succeeds by allowing itself to go crazy , but ultimately fails by not coming to consciousness .
This may be the dumbest , sketchiest movie on record about an aspiring writer 's coming-of-age .	This may be the dumbest , sketchiest movie on record , not an aspiring writer 's coming-of-age .
Before long , the film starts playing like General Hospital crossed with a Saturday Night Live spoof of Dog Day Afternoon .	Before long , the film starts playing like nothing crossed with a Saturday Night Live spoof of Dog Day Afternoon .
A startling and fresh examination of how the bike still remains an ambiguous icon in Chinese society .	A startling and fresh examination of how the bike still seems to be an ambiguous icon in Chinese society .
Never engaging , utterly predictable and completely void of anything remotely interesting or suspenseful .	Not engaging , utterly predictable and completely void of anything remotely interesting or suspenseful .
Between the drama of Cube ?	Are there no interesting problems?
Tailored to entertain !	No tails !

Table 5.7: Failure cases of Polyjuice negations. The first half shows examples where the negation is irrelevant to the sentiment. The second half includes examples where the negation fails to appear.

negation) or not cross (e.g., entity change) the decision boundary in the (very) high-dimensional input space. However, as the negation case indicates, we must be careful about the exact definition of the rewriting (e.g., negating any part of the input or the "sentiment-carrying" part only), as it could have a significant impact on the conclusion. Furthermore, it is difficult for any rewriting mechanism to achieve 100% validity due to the high dimensionality, the multitude of possible ways of rewriting, and the imperfection of the model. Focusing only on the mistakes (or ignoring them altogether) yields incomplete model understanding. Instead, the validity metric, which indicates the *generalized* model behavior, should be used to.

There are many potentially fruitful directions for future work. First, the quality of instances obviously depend on the generative models, which, while impressive, are known to be flawed in many ways [e.g., 66, 112, 164]. Second, each rule essentially covers the entire input space. Partitioning the input space in some way may allow for identification of both more and less consistent areas, which is makes the applicability function much more difficult to define as it now takes whole sentences rather than individual words. Finally, unlike feature attribution, which conveys the single notion of "importance," different instances of the same input can reveal different aspects of model behavior, calling for a potentially different definition of coverage, which measures completeness of understanding.

Chapter 6

Model Transparency by Example

6.1 Introduction

The previous chapters introduce methods for defining and evaluating model explanations. As we have seen, these explanations are *local*, in that they explain the model for predictions on certain inputs. Thus, an important question that needs addressing is: what inputs do we want to explain in the first place?

Currently, this question has been answered in very simplistic way: *some* test set instances, where "some" could mean *random*, *cherry-picked*, *correctly predicted* or *incorrectly predicted*. To make this process more rigorous, ExSum advocates for the use of the *full* test set in deriving formal and rigorous pieces of model understanding.

However, a model understanding procedure based on the test set is, by definition, limited to the test set. While a large test set is in many times sufficient for covering most aspects of our desired model understanding, it can be limited in several crucial ways. First, certain model behaviors may occur very rarely, such as high-confidence mistakes for a very high-performing model, for which browsing through even a large test set may only surface a handful, which may or may not be representative of the general model behavior. In addition, as some level of distribution shift is almost always expected from model development to deployment, how the model adapts to (slightly or significantly) different distributions are very hard to study, if at all possible, when we are limited to the test set.

This chapter is based on the AAAI 2021 paper "Bayes-TrEx: A Bayesian Sampling Approach to Model Transparency by Example" by Serena Booth, Yilun Zhou, Ankit Shah and Julie Shah [21]. Booth and Zhou contributed equally to this work.

In this chapter, we formalize the above investigations using the concept of model transparency, which is to understand a model via its input-output behaviors. Thus, they can be considered as the step before the investigations of interpretability, which try to explain *why* certain input-output behaviors happen.

In particular, we propose a transparency-by-example perspective to model understanding. Specifically, rather than going through test set to identify interesting model behaviors, we reverse this process by starting with certain target behaviors that we want to study, and then find supporting input instances that elicit these behaviors. For a classifier, which is the model that this chapter investigates, such behaviors may include the model being confused with respect to two or more classes, exhibiting high-confidence failures, and forcing to generalize to data of novel classes or styles.

To solve the problem of the test set potentially having poor coverage on rare behaviors or simply failing to represent novel data, we propose to train a generative model to synthesize data instances and use it to find the ones that demonstrate the behaviors that we are after. As an added benefit, using a generative model also mitigates privacy concerns: illustrating model behaviors to stakeholders with test set instances may reveal sensitive information, which can be protected if the instances are now synthetic (but with similar statistics as the test set).

The simplest way to find behavior-conforming instances is to simply filter through all instances output by the generative model. However, this approach is very inefficient if the behavior is rare, since we need to discard a large fraction of data. Essentially, what we want is to define a "desirability function" on data instances that increases as the input becomes more likely to be produced by the generator and conforms more to the desired behavior, and then find instances whose desirability function values are large. We propose an analogy between the search problem and that of sampling from a Bayesian posterior function, which enables us to use standard sampling techniques, like Markov chain Monte-Carlo (MCMC) for this task. Due to this Bayesian formulation, we call our framework Bayes-TrEx.

Intuitively, the idea is presented in Figure 6-1, which shows a Corgi vs. Bread classifier. For different $\mathbf{p}$-level set targets (e.g., $\mathbf{p}_{\text{Corgi}} = \mathbf{p}_{\text{Bread}} = 0.5$), Bayes-TrEx can find examples where the model is highly confident in the Corgi class, in the Bread class, or ambivalent between the two. Notably, all the samples are likely to be produced by the generator in the first place, eliminating pathological cases such as random noise or other non-sensical inputs that happen to have the desired model behavior.

As a baseline comparison to the standard test set inspection approach, we search for highly confident misclassifications and ambivalent examples in the (Fashion-)MNIST

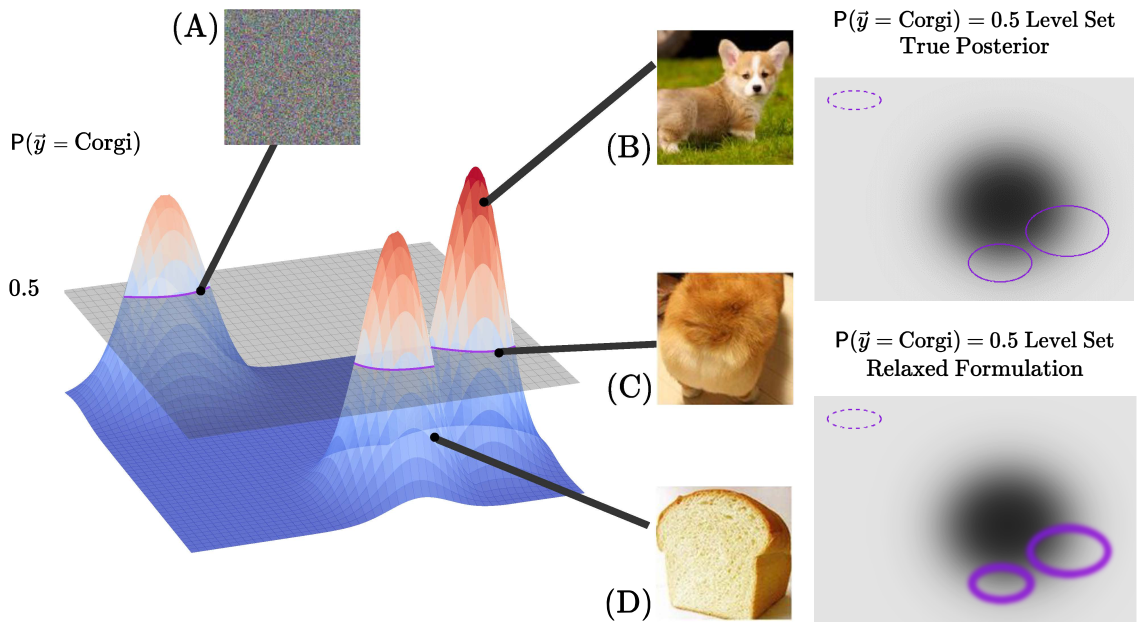

(A)
$P(\vec{y} = \text{Corgi})$
0.5
(B)
(C)
(D)
$P(\vec{y} = \text{Corgi}) = 0.5$ Level Set
True Posterior
$P(\vec{y} = \text{Corgi}) = 0.5$ Level Set
Relaxed Formulation

Figure 6-1: Left: given a Corgi vs. Bread classifier, we generate *prediction level sets*, or sets of examples of a target prediction confidence. One way of finding such examples is by perturbing an arbitrary image to the target confidence (e.g., $\mathbf{p}_{\text{Corgi}} = \mathbf{p}_{\text{Bread}} = 0.5$), as shown in (A). However, such examples give little insights into the typical model behavior because they are extremely unlikely in realistic situations. Bayes-TrEx explicitly considers a data distribution (gray shade on the right plots) and finds in-distribution examples in a particular level set (e.g., likely Corgi (B), likely Bread (D), or ambivalent between Corgi and Bread (C)). Upper right: the classifier level set of $\mathbf{p}_{\text{Corgi}} = \mathbf{p}_{\text{Bread}} = 0.5$ overlaid on the data distribution. Example (A) is unlikely to be sampled by Bayes-TrEx due to near-zero density under the distribution, while example (C) is likely to be. Lower right: sampling directly from the true posterior is infeasible, so Bayes-TrEx includes a relaxation by "widening" the level set.

and CLEVR test sets. We find few such test set examples meet these constraints, and the majority of these can be attributed to mislabeling in the dataset collection pipeline rather than misclassification by the model. In contrast, Bayes-TrEx consistently finds more highly confident misclassified and ambivalent examples, which enables more flexible and comprehensive model inspection and understanding.

Given the instances identified in this transparency-by-example stage, we can consider interpretability analysis as a downstream process, to find *why* the behavior occurs, in addition to *what* the behavior is. For example, Figure 6-2 demonstrated that the reason for a high-confidence failure (with 97% confidence, the image contains a sphere) of a visual recognition model is due to the confusion of the small red cylinder as a sphere.

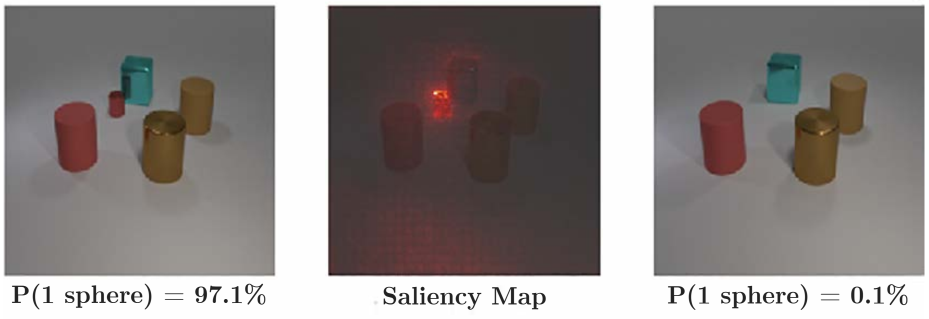


Figure 6-2: Bayes-TrEx finds a CLEVR scene which is incorrectly classified as containing a sphere. The generated example (left) is composed of only cylinders and cubes, but the classifier is 97.1% confident this scene contains one sphere. The SmoothGrad [143] saliency map highlights the small red cylinder as the object that is confused for a sphere. When we remove it, the classifier's confidence that the scene contains one sphere drops to 0.1%.

6.2 Related Work

Model Testing TensorFuzz [118] is a fuzzing test framework for neural networks which finds inputs that achieve a wide coverage of user-specified constraints. TensorFuzz is similar to Bayes-TrEx in that both methods aim to find examples that elicit certain model behaviors. While TensorFuzz is designed to find *rare* inputs that trigger edge cases such as numerical errors, Bayes-TrEx finds common, in-distribution examples. As such, Bayes-TrEx is more suitable to help humans develop a correct mental model of the classifier. Scenic [45] is a domain-specific language for model testing by generating failure-inducing examples. While Bayes-TrEx is in part inspired by Scenic, its formulation is more flexible.

Natural Adversarial Examples One Bayes-TrEx use case is uncovering high-confidence classification failures in the data distribution. This idea is related to, but different from, natural adversarial attacks [171]. Most adversarial attacks inject crafted high-frequency information to mislead a trained model [51, 115, 148], but such artifacts are non-existent in natural images. Zhao et al. [171] instead proposed a method to find *natural* adversarial examples by performing the perturbation in the latent space of a GAN. While this method finds an example which looks like a specific input, Bayes-TrEx finds high-confidence misclassifications in the entire data distribution.

Confidence in Neural Networks Bayes-TrEx can also be used to detect overconfidence in neural networks. An overconfident neural network [55] makes many mistakes with disproportionately high confidence. While many approaches aim to address this network overconfidence problem [19, 46, 93, 149], Bayes-TrEx is complementary to these efforts. Rather than altering the confidence of a neural network, it instead infers examples of a particular confidence. If the model is overconfident, it may return few, if any, samples with ambivalent predictions. At the same time, it may find many misclassifications with high confidence. In our experiments (Section 6.4.9), we discover that the popular adversarial discriminative domain adaptation (ADDA) technique produces a more overconfident model than the baseline.

6.3 Methodology

Given a classifier $f : X \to \Delta_K$ which maps a data point to the probability simplex of K classes, the goal is to find an input $\mathbf{x} \in X$ in a given data distribution $p(\mathbf{x})$ such that $f(\mathbf{x}) = \mathbf{p}$ for some prediction confidence $\mathbf{p} \in \Delta_K$. We consider the problem of sampling from the posterior

$$p(\mathbf{x}|f(\mathbf{x}) = \mathbf{p}) \propto p(\mathbf{x})\, p(f(\mathbf{x}) = \mathbf{p}|\mathbf{x}). \tag{6.1}$$

A common approach to posterior sampling is to use Markov Chain Monte-Carlo (MCMC) methods [23]. However, when the measure of the level set $\{\mathbf{x} : f(\mathbf{x}) = \mathbf{p}\}$ is small or even zero, MCMC sampling is infeasible: the posterior being zero everywhere outside of the level set makes it unlikely for a random-walk Metropolis sampler to land on $\mathbf{x}$ with non-zero posterior [58], and the gradient being zero everywhere outside of the level set means that a Hamiltonian Monte Carlo sampler does not have the necessary gradient guidance toward the level set [113].

We relax the formulation by "widening" the level set and accepting $\mathbf{x}$ when $f(\mathbf{x})$ is close to the target $\mathbf{p}$ (Figure 6-1). Specifically, for hyper-parameter σ that sets the amount of widening, we introduce a random vector $\mathbf{u} = [u_1, \ldots, u_K]^T$, distributed as

$$u_i|\mathbf{x} \sim \mathcal{N}\left(f(\mathbf{x})_i, \sigma^2\right). \tag{6.2}$$

Instead of directly sampling from Equation 6.1, we sample from the new posterior:

$$p(\mathbf{x}|\mathbf{u} = \mathbf{u}^*) \propto p(\mathbf{x})p(\mathbf{u} = \mathbf{u}^*|\mathbf{x}), \tag{6.3}$$
$$\mathbf{u}^* = \mathbf{p}. \tag{6.4}$$

The hyper-parameter σ controls the peakiness of the relaxed posterior. A smaller α

makes it closer to the true posterior and makes the distribution peakier and harder to sample, while a larger α makes it closer to the data distribution $p(\mathbf{x})$ and easier to sample. As σ goes to 0, it approaches the true posterior. Formally,

$$\lim_{\sigma \to 0} p(\mathbf{x}|\mathbf{u} = \mathbf{u}^*) = p(\mathbf{x}|f(\mathbf{x}) = \mathbf{p}). \tag{6.5}$$

While the formulation in Equation 6.2 is applicable to arbitrary confidence $\mathbf{p}$, the dimension of $\mathbf{u}$ is equal to the number of classes, which poses scalability issues for large numbers of classes. For a wide range of interesting use cases of Bayes-TrEx, we can use the following reductions:

1. Highly confident in class i: $\mathbf{p}_i = 1, \mathbf{p}_{-i} = 0$. We have

$$u|\mathbf{x} \sim \mathcal{N}\left(f(\mathbf{x})_i, \sigma^2\right), \qquad u^* = 1. \tag{6.6}$$

2. Ambivalent between class i and j: $\mathbf{p}_i = \mathbf{p}_j = 0.5$, $\mathbf{p}_{-i,j} = 0$. We have

$$u_1|\mathbf{x} \sim \mathcal{N}\left(|f(\mathbf{x})_i - f(\mathbf{x})_j|, \sigma_1^2\right), \tag{6.7}$$
$$u_2|\mathbf{x} \sim \mathcal{N}(\min(f(\mathbf{x})_i, f(\mathbf{x})_j) - \max_{k \neq i,j} f(\mathbf{x})_k, \sigma_2^2), \tag{6.8}$$
$$u_1^* = 0, u_2^* = 0.5. \tag{6.9}$$

σ_1 and σ_2 are hyperparameters.

3. Uniformly ambivalent among all classes: $\mathbf{p}_i = 1/L$ for all i. We have:

$$u|\mathbf{x} \sim \mathrm{No}(\max_i f(\mathbf{x})_i - \min_j f(\mathbf{x})_j, \sigma^2), \tag{6.10}$$
$$u^* = 0. \tag{6.11}$$

In addition, most high dimensional data distributions, such as those for images, are implicitly defined by a transformation $g : Z \to X$ from a latent distribution $p(\mathbf{z})$. Consequently, given

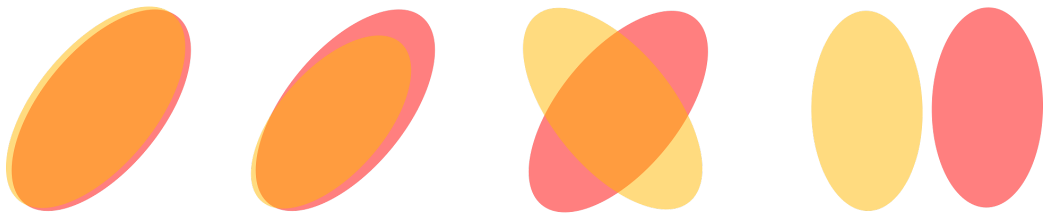
$$\mathbf{x} = g(\mathbf{z}), \tag{6.12}$$
$$\mathbf{u}|\mathbf{z} \sim \mathcal{N}(f(\mathbf{x}), \sigma^2), \tag{6.13}$$
$$p(\mathbf{z}|\mathbf{u} = \mathbf{u}^*) \propto p(\mathbf{z})p(\mathbf{u} = \mathbf{u}^*|\mathbf{z}), \tag{6.14}$$

Bayes-TrEx samples $\mathbf{z}$ according to Equation 6.14 and reconstruct the example $\mathbf{x} = g(\mathbf{z})$ for model inspection.

6.4 Experiments

6.4.1 Overview

A key strength of Bayes-TrEx is the ability to evaluate a classifier on a data distribution $\mathbb{P}_D$, independent of its training distribution $\mathbb{P}_C$. We demonstrate the versatility of Bayes-TrEx on four relationships between $\mathbb{P}_D$ and $\mathbb{P}_C$ (Figure 6-3). With $\mathbb{P}_C = \mathbb{P}_D$ (Figure 6-3(a)), Section 6.4.4 and 6.4.5 present examples that trigger high and ambivalent model confidence and Section 6.4.6 presents examples that interpolate between two classes. In Section 6.4.7, we consider $\mathbb{P}_D$ with narrower support than $\mathbb{P}_C$ (Figure 6-3(b)), where the support of $\mathbb{P}_D$ excludes examples from a particular class. In this case, high-confidence examples – as judged by the classifier – correspond to high-confidence misclassifications. In Section 6.4.8 and 6.4.9, we analyze the classifier C for novel class extrapolation and covariate shift behaviors with overlapping or disjoint supports of $\mathbb{P}_C$ and $\mathbb{P}_D$ (Figure 6-3(c, d)). Representative results are in the main text; further results are in the appendix.



(a) $\mathbb{P}_D{=}\mathbb{P}_C$ (b) $\mathbb{P}_D{\subsetneq}\mathbb{P}_C$ (c) $\mathbb{P}_D{\cap}\mathbb{P}_C{\neq}\mathbb{P}_D{\neq}\mathbb{P}_C$ (d) $\mathbb{P}_D{\cap}\mathbb{P}_C{=}\ \varnothing$

Figure 6-3: Different relations between the classifier training distribution ($\mathbb{P}_C$, red) and Bayes-TrEx data distribution ($\mathbb{P}_D$, yellow). (a) $\mathbb{P}_C$ and $\mathbb{P}_D$ are equal. (b) The support of $\mathbb{P}_D$ is a subset of that of $\mathbb{P}_C$. (c) $\mathbb{P}_D$ and $\mathbb{P}_C$ have overlapping supports. (d) Supports of $\mathbb{P}_C$ and $\mathbb{P}_D$ are disjoint.

6.4.2 Datasets, Training and Inference Details

We evaluate Bayes-TrEx on rendered images (CLEVR) and organic datasets (MNIST and Fashion-MNIST). For all CLEVR experiments, we use the pre-trained classifier distributed by the original authors[1]. The transition kernel uses a Gaussian proposal for the continuous variables (e.g., x-position) and categorical proposal for the discrete variables (e.g., color), both centered around and peaked at the current value. For (Fashion-)MNIST experiments, architectures and training details are described

[1]`https://github.com/facebookresearch/clevr-iep`

Model	Dataset	FID	Model	Dataset	FID
VAE	MNIST	72.33	GAN	MNIST	11.83
	Fashion-MNIST	87.89		Fashion-MNIST	29.44

Table 6.1: Fréchet Inception Distance (FID) for VAE and GAN models trained on the entire dataset. A lower value indicates higher quality. Appendix 6.6.2 presents the statistics for all models.

in Appendix 6.6.1. For covariate shift analysis, we train ADDA and baseline models using the code provided by the authors[2].

CLEVR images are rendered from scene graphs, on which we define the latent distribution $p(\mathbf{z})$. Since the (Fashion-)MNIST groundtruth data distribution is unknown, we estimate it using a VAE or GAN with unit Gaussian $p(\mathbf{z})$. These learned data distribution representations have known limitations, which may affect sample quality [8]. Table 6.1 lists the Fréchet Inception Distance (FID) [63] for two VAE and GAN models, with the full table in Appendix 6.6.2. The FID scores show the GANs generate more representative samples than the VAEs.

We consider two MCMC samplers: random-walk Metropolis (RWM) and Hamiltonian Monte Carlo (HMC). We use the former in CLEVR where the rendering function is non-differentiable, and the latter for (Fashion-)MNIST. For HMC, we use the No-U-Turn sampler [65, 113] implemented in the probabilistic programming language Pyro [18]. We choose $\sigma = 0.05$ for all experiments. Alternatively, σ can be annealed to gradually reduce the relaxation.

Computing stopping criteria for MCMC methods is an open problem, usually requiring a gold standard inference algorithm [34] or specific posterior distribution properties, such as log-concavity [52]. As neither of these requirements are met for our domains, we select stopping criteria based on heuristic performance and cost of compute. CLEVR scenes require GPU-intensive rendering, so we stop after 500 samples. (Fashion-)MNIST samples are cheaper to generate, so we stop after 2,000 samples. Empirically, we find each sampling step takes 3.75 seconds for CLEVR, 1.18s for MNIST, and 1.96s for Fashion-MNIST, all on a single NVIDIA GeForce 1080 GPU.

6.4.3 Quantitative Evaluation

We first evaluate the quality of Bayes-TrEx samples by assessing whether the classifier's prediction confidence matches the specified target on the generated examples.

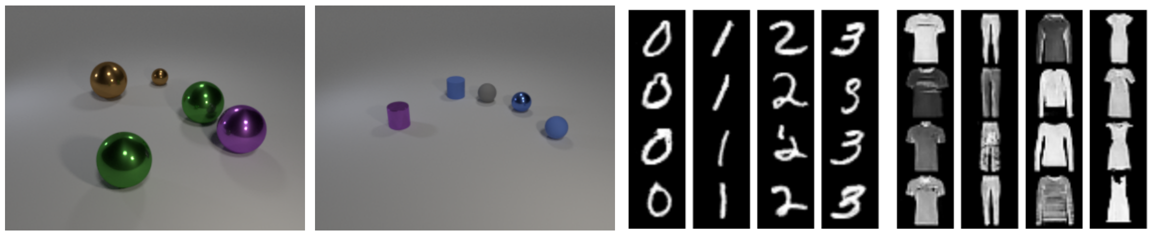
[2]`https://github.com/erictzeng/adda`

Table 6.2 presents the mean and standard deviation of the confidence on a selection of representative settings, and Appendix 6.6.3 lists the full set of such evaluations. In general, the prediction confidences are tightly concentrated around the target, indicating sampler success.

Use Case	Dataset	Target	Prediction Confidence
High Conf.	MNIST	$\mathbf{p}_4 = 1$	1.00 ± 0.01
	Fashion	$\mathbf{p}_{\text{Coat}} = 1$	0.98 ± 0.02
	CLEVR	$\mathbf{p}_{\text{2 Blue Spheres}} = 1$	0.89 ± 0.25
Ambivalent	MNIST	$\mathbf{p}_1 = \mathbf{p}_7 = 0.5$	$0.49 \pm 0.02, 0.49 \pm 0.03$
	Fashion	$\mathbf{p}_{\text{T-shirt}} = \mathbf{p}_{\text{Dress}} = 0.5$	$0.48 \pm 0.02, 0.48 \pm 0.02$
Interpolation	MNIST	$\mathbf{p}_8 = 0.6, \mathbf{p}_9 = 0.4$	$0.58 \pm 0.04, 0.37 \pm 0.04$
	Fashion	$\mathbf{p}_{\text{T-shirt}} = 0.2, \mathbf{p}_{\text{Trousers}} = 0.8$	$0.17 \pm 0.04, 0.79 \pm 0.04$
Misclassified	MNIST	$\mathbf{p}_8 = 1$	0.98 ± 0.02
	Fashion	$\mathbf{p}_{\text{Bag}} = 1$	0.97 ± 0.03
	CLEVR	$\mathbf{p}_{\text{1 Cube}} = 1$	0.93 ± 0.06
Extrapolation	MNIST	$\mathbf{p}_6 = 1$	1.00 ± 0.01
	Fashion	$\mathbf{p}_{\text{Sandal}} = 1$	1.00 ± 0.01
	CLEVR	$\mathbf{p}_{\text{1 Cylinder}} = 1$	0.96 ± 0.03
Domain Adapt.	MNIST	$\mathbf{p}_5 = 1$	1.00 ± 0.01

Table 6.2: Mean and standard deviation of the prediction confidence of the samples. Reported values are for the target class, or two target classes in ambivalent confidence and confidence interpolation cases. Appendix 6.6.3 presents the complete statistics.

6.4.4 High Confidence



(a) $\mathbf{p}_{\text{5 Spheres}} = 95.7\%$ (b) $\mathbf{p}_{\text{2 Blue Sph.}} = 91.1\%$ (c) MNIST (d) Fashion-MNIST

Figure 6-4: High-confidence samples. (a, b) CLEVR. (c) MNIST, digits 0-3. (d) Fashion-MNIST, left to right: T-shirt, trousers, pullover and dress. More examples in Appendix 6.6.4.

As an initial smoke test, we evaluate Bayes-TrEx by finding highly confident examples. (Fashion-)MNIST data distributions are learned by GAN. Figure 6-4 depicts samples on the three datasets. Additional examples are in Appendix 6.6.4.

6.4.5 Ambivalent Confidence

Next, we find ambivalent (Fashion-)MNIST examples for which the classifier has similar prediction confidence between two classes, using data distributions learned by a VAE. Figure 6-5 shows ambivalent examples from each pair of classes (e.g., 0v1, 0v2, ..., 8v9). Note the examples presented are ambivalent from the classifier's perspective, though some may be readily classified by a human. Not all pairs result in successful sampling: for example, we were unable to find an ambivalent example with equal prediction confidence between the visually dissimilar classes 0 and 7. These ambivalent examples are useful for visualizing and understanding class boundaries; Appendix 6.6.5 presents a supporting class boundary latent space visualization. *Blended* ambivalent examples have previously been shown to be useful for data augmentation [150]. While these generated ambivalent examples may be similarly useful, we leave this exploration to future work.

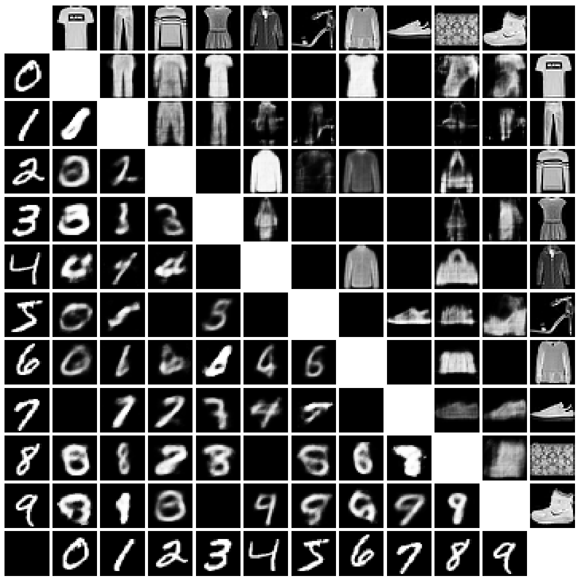


Figure 6-5: Pairwise ambivalent example matrix. Each entry of the matrix is an ambivalent MNIST or Fashion-MNIST example for the classes on its row and column. Blacked-out cells indicate sampling failures. Examples on the outermost edges of the matrix are class representations (e.g., 0-9 for MNIST).

Bayes-TrEx can also find examples which are ambivalent across more than two classes; Figure 6-6 presents samples that are equally ambivalent across all 10 MNIST classes. All these images appear to be very blurry and not very realistic. This is intuitive: even for a human, it would be hard to write a digit in such a way that it is equally unrecognizable across all 10 classes.

For ambivalent examples, we observed only rare successes with data distributions

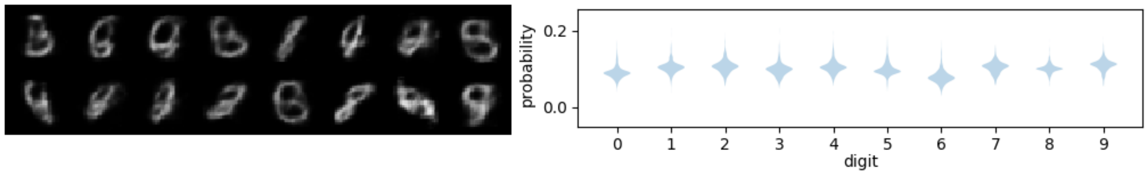


Figure 6-6: Samples of uniformly ambivalent predictions.

learned by a GAN, which generates sharper and more visually realistic images than a VAE. There are two candidate explanations:

1. GAN-distributions prevent efficient MCMC sampling.
2. The classifier rarely makes ambivalent predictions on sharp and realistic images.

To experimentally check the second explanation, we train a classifier to be consistently ambivalent between class i and $i+1$ for an image of digit i (wrapping around at $10 = 0$) using the following KL-divergence loss:

$$l(y, f(\mathbf{x})) = \mathbb{KL}(\mathbf{p}_y, f(\mathbf{x})), \tag{6.15}$$

$$\mathbf{p}_{y,i} = \begin{cases} 0.5 & i = y \text{ or } i = (y+1) \bmod 10, \\ 0 & \text{otherwise.} \end{cases} \tag{6.16}$$

Using this classifier, we sample ambivalent examples for 0v1, 1v2, ..., 9v0. Sampling succeeds for all ten pairs, even when using the same GAN model that rarely succeeded in the prior experiment. Figure 6-7 presents the 0v1 samples and predicted confidence by this modified classifier, and the remaining pairs are visualized in Appendix 6.6.6. Given this sampling success, we conclude that the second explanation is correct.

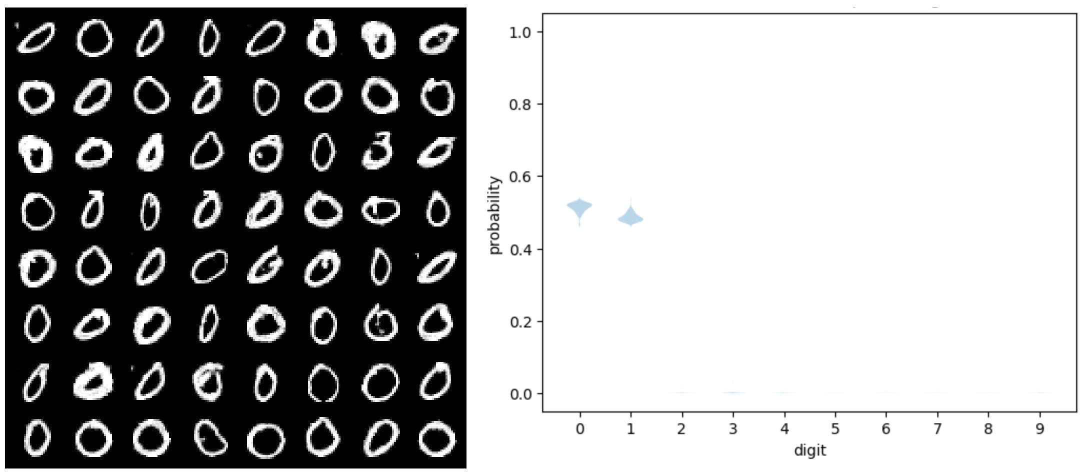


Figure 6-7: 0v1 ambivalent samples and confidence plot with the GAN distribution and always ambivalent classifier.

Bayes-TrEx is also unable to generate ambivalent examples for CLEVR with the manually defined data distribution. Given that the pre-trained classifier only achieves $\approx 60\%$ accuracy, the result suggests that the model is likely overconfident, which has previously been observed in similar settings [83].

6.4.6 Confidence Interpolation

Bayes-TrEx can find examples that interpolate between classes. In Figure 6-8, we show MNIST samples which interpolate from $(P_8 = 1.0, P_9 = 0.0)$ to $(P_8 = 0.0, P_9 = 1.0)$ and Fashion-MNIST samples from $(P_{\text{T-shirt}} = 1.0, P_{\text{Trousers}} = 0.0)$ to $(P_{\text{T-shirt}} = 0.0, P_{\text{Trousers}} = 1.0)$ over intervals of 0.1, with a VAE-learned data distribution.

The interpolation between two very different classes reveal insights into the model behavior. For example, the interpolation from 8 to 9 generally shrinks the bottom circle toward a stroke, which is the key difference between digits 8 and 9. For Fashion-MNIST, the presence of two legs is important for trousers classification, even appearing in samples with $(\mathbf{p}_{\text{T-shirt}} = 0.9, \mathbf{p}_{\text{Trousers}} = 0.1)$ (second column). By contrast, a wider top and the appearance of sleeves are important properties for T-shirt classification. These two trends result in most of the interpolated samples having a short sleeve on the top and two distinct legs on the bottom.

6.4.7 High-Confidence Failures

With neural networks being increasingly used for high-stakes decision making, high-confidence failures are one area of concern, as these failures may go unnoticed. Bayes-TrEx can find such failures. Specifically, if the data distribution (Figure 6-3(b)) does *not* include a particular class, then the resulting high-confidence examples correspond to high-confidence *misclassifications* for that class. For example, in Figure 6-9(a), the CLEVR classifier is highly confident that there is one cube though there is no cube in the image. In Appendix 6.6.11, the saliency map for Figure 6-9(a) reveals that classifier mistakes the front shiny red cylinder for a cube. Removing this cylinder causes the confidence to drop to 29.0%. In addition, such high-confidence failures can also be used for data augmentation to increase network reliability [45].

For (Fashion-)MNIST, a GAN is trained on all data without a single class, resulting in the learned data distribution excluding the given class. Figs. 6-9(c) and 6-9(d) depict high-confidence misclassifications for digits 0-4 in MNIST and sandal, shirt, sneaker, bag, and ankle boot in Fashion-MNIST, respectively. By evaluating these examples, we can assess how well human-aligned a classifier is. For example, for MNIST, some

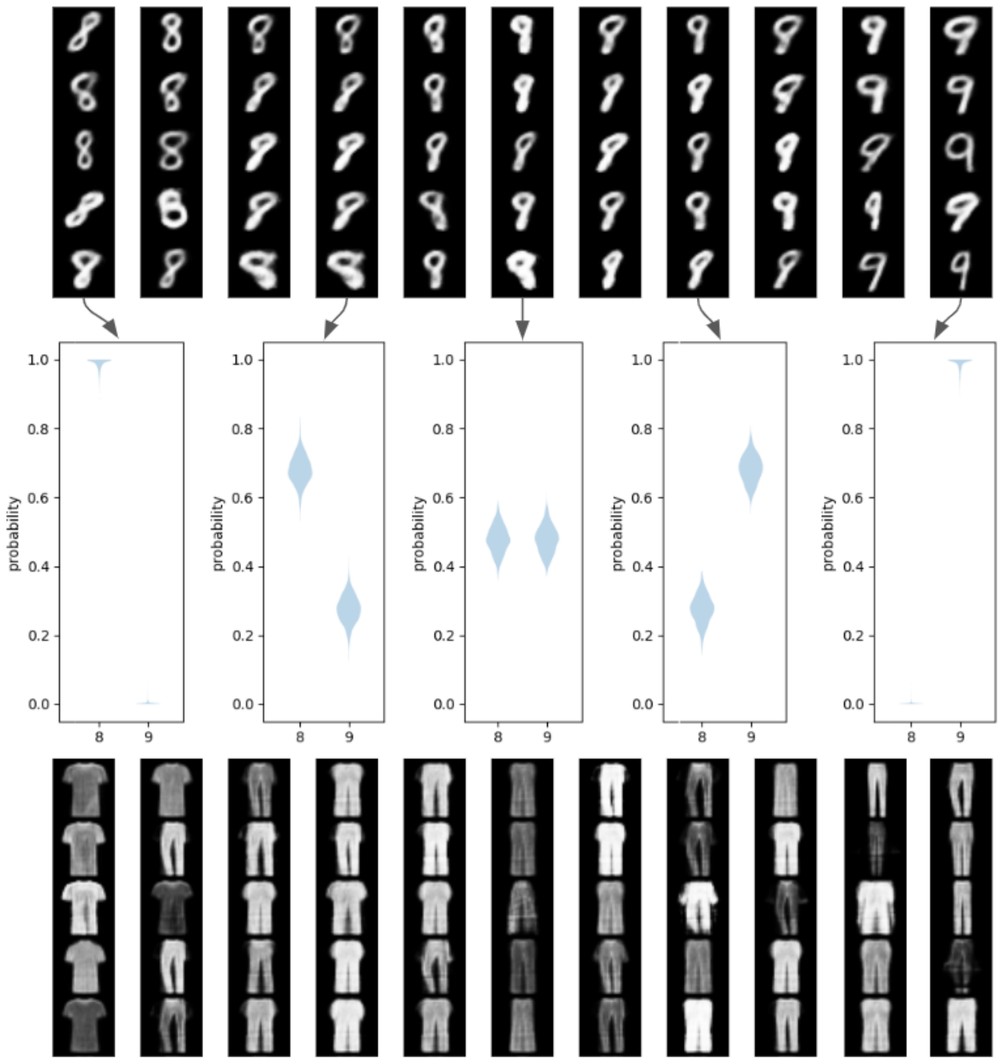
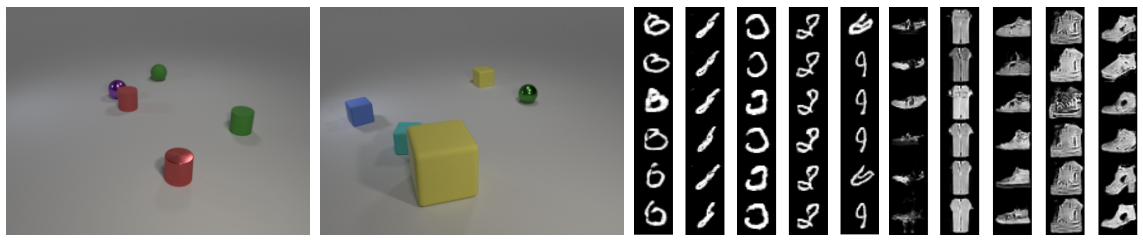


Figure 6-8: Confidence interpolation between digit 8 and 9 for MNIST and between T-shirt and trousers for Fashion-MNIST. Each of the 11 columns show samples of confidence ranging from $[\mathbf{p}_{\text{class a}} = 1.0, \mathbf{p}_{\text{class b}} = 0.0]$ (left) to $[\mathbf{p}_{\text{class a}} = 0.0, \mathbf{p}_{\text{class b}} = 1.0]$ (right), with an interval of 0.1. Some confidence plots are shown in the middle.

thin 8s are classified as 1s and particular styles of 6s and 9s are classified as 4s. These results seem intuitive, as a human might make these same mistakes. Likewise, for Fashion-MNIST, most failures come from semantically similar classes, e.g. sneaker $\longleftrightarrow$ ankle boot. Less intuitively, however, chunky shoes are likely to be classified as bags. Additional visualizations are presented in Appendix 6.6.7.

6.4.8 Novel Class Extrapolation

It is important to understand the novel class extrapolation behavior of a model before deployment. For example, during training an autonomous vehicle might learn to safely



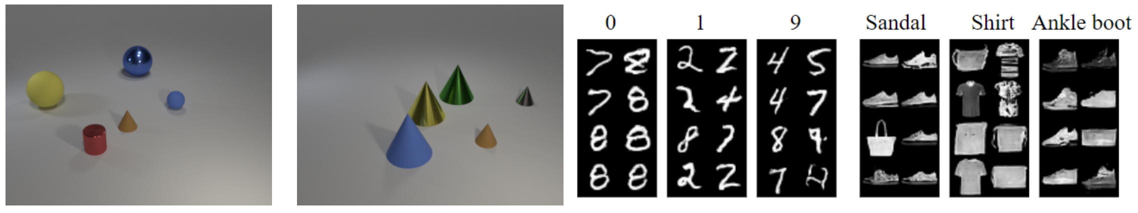
(a) $\mathbf{p}_{1\ \text{Cube}} = 93.5\%$ (b) $\mathbf{p}_{2\ \text{Cylinders}} = 90.2\%$ (c) MNIST (d) Fashion-MNIST

Figure 6-9: High-confidence classification failures. (a): CLEVR, 1 Cube. Note that no cube is present in the sample. (b): CLEVR, 2 Cylinders – again, containing no cylinders. (c) MNIST failures for digits 0-4. 0s are composed of 6s; 1s of 8s; 2s of 0s, and so on. (d) Fashion-MNIST failures for sandal, shirt, sneaker, bag, and ankle boot. Additional examples are presented in Appendix 6.6.7.

operate around pedestrians, cyclists, and cars. But can we predict how the vehicle will behave when it encounters a novel class, like a tandem bicycle? Bayes-TrEx can be used to understand such behaviors by sampling high-confidence examples with a data distribution that contains novel classes, while excluding the true target classes (Figure 6-3(c, d)).

For CLEVR, we add a novel cone object to the data distribution and remove the existing cube from it. We sample images that the classifier is confident to include cubes, shown in Figure 6-10 (a, b). A saliency map analysis in Appendix 6.6.11 confirms that the classifier indeed mistakes these cones for cubes. In Appendix 6.6.8, we assess CLEVR's novel class extrapolation for cylinders and spheres, and similarly show the model readily confuses cones for these classes as well.

For MNIST and Fashion-MNIST, we train the respective classifiers on digits 0, 1, 3,



(a) $\mathbf{p}_{1\ \text{Cube}} = 98.5\%$ (b) $\mathbf{p}_{5\ \text{Cubes}} = 92.5\%$ (c) MNIST (d) Fashion-MNIST

Figure 6-10: Novel class extrapolation examples. (a, b): For CLEVR, the novel cone objects are mistaken for cubes. (c, d): For (Fashion-)MNIST, we train classifiers on subsets of the data (digits 0, 1, 3, 6, 9 and pullover, dress, sandal, shirt, and ankle boot), and train GANs with the excluded data. Samples for which the classifier is highly confident ($\approx 99\%$) in several target classes are shown (e.g., targets 0, 1, and 9 for MNIST). Additional examples are presented in Appendix 6.6.8.

6, 9 and pullover, dress, sandal, shirt and ankle boot classes. We train GANs using only the excluded classes (e.g., digits 2, 4, 5, 7, 8 for MNIST) and find examples where the classifier has high prediction confidence, as shown in Figure 6-10 (c, d). For MNIST, there are few reasonable extrapolation behaviors, most likely due to the visual distinctiveness between digits. By comparison, some Fashion-MNIST extrapolations are expected, such as confusing the unseen sneaker class for sandals and ankle boots. However, the classifier also confidently mistakes various styles of bags as sandals, shirts, and ankle boots. Appendix 6.6.8 contains additional visualizations.

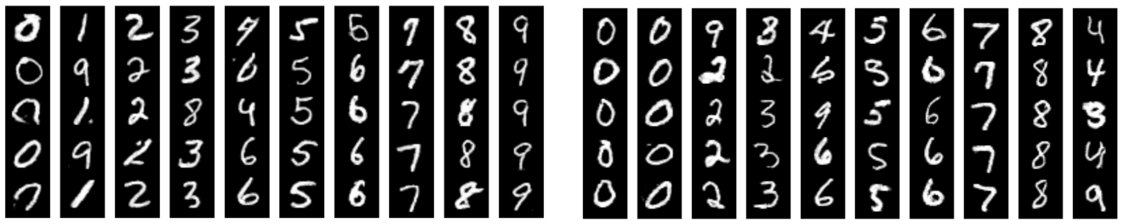
6.4.9 Covariate Shift

Finally, we use Bayes-TrEx to analyze covariate shift behaviors. We reproduce the SVHN [114] $\rightarrow$ MNIST experiment studied by Tzeng et al. [153]. We train two classifiers, a baseline classifier on labeled SVHN data only, and the ADDA classifier on labeled SVHN data and unlabeled MNIST data. Indeed, covariate shift improves classification accuracy: 61% for the baseline classifier on MNIST vs. 71% for the ADDA classifier.

But is this the whole story? To study model performance in the high-confidence range, we use Bayes-TrEx to generate high-confidence examples for both classifiers with the MNIST data distribution learned by GAN, as shown Figure 6-11. It appears the ADDA model makes *more* mistakes in these images – for example, in the 2nd column in Figure 6-11(b), all images where the classifier is highly confident to be 1 are actually 0s. To further study this, we hand-label 10 images per class and compute the classifier accuracy on them. Table 6.3 shows the accuracy per digit class, as well as the overall accuracy. This analysis confirms the baseline model is more accurate than the ADDA model on these samples, suggesting that ADDA is more overconfident than the baseline. While this result does not contradict the higher overall accuracy of ADDA, it does caution against deploying such domain adaptation models without further inspection and confidence calibration assessment.

	0	1	2	3	4	5	6	7	8	9	All
Baseline	1.0	0.6	1.0	0.7	0.5	0.9	0.9	0.7	1.0	0.7	0.80
ADDA	0.9	0.0	0.8	0.9	0.2	1.0	0.8	1.0	1.0	0.6	0.72

Table 6.3: Per-digit and overall accuracy among high-confidence MNIST samples for the baseline and ADDA models. While ADDA has higher overall accuracy (0.71 vs. 0.61), it performs worse on high-confidence samples (0.72 vs. 0.80). This suggests overconfidence.



(a) Baseline examples (b) ADDA examples

Figure 6-11: Highly confident examples for each class (0 to 9) of the baseline model and ADDA model. Additional examples are presented in Appendix 6.6.9.

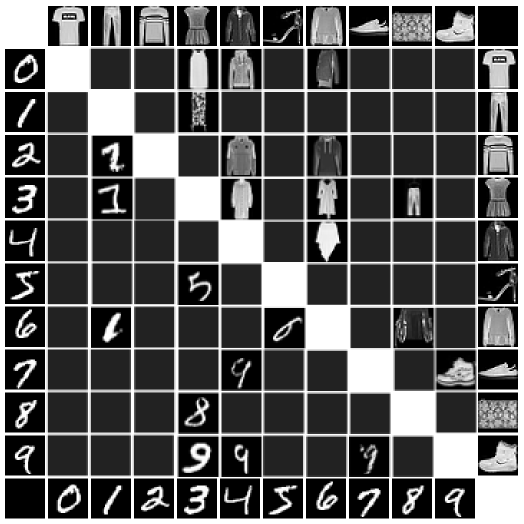


Figure 6-12: Test set ambivalent examples for (Fashion-)MNIST. Compared to those found by Bayes-TrEx in Figure 6-5, test set examples have much poorer coverage.

6.4.10 Test-Set Comparison

Standard model evaluations are typically performed on the test set. While inspecting test set examples is not an apples-to-apples comparison for all Bayes-TrEx use cases (e.g., covariate shift), we study the comparable ones.

Ambivalent Confidence

We find ambivalent examples in the (Fashion-)MNIST datasets where the classifier has confidence in $[40\%, 60\%]$ for two classes. Out of 10,000 test examples on each dataset, we find only 12 MNIST examples across 10 class pairings, and 162 Fashion-MNIST examples across 12 pairings, as shown in Figure 6-12. By comparison, Bayes-TrEx found ambivalent examples for 38 MNIST pairings and 28 Fashion-MNIST pairings (cf. Figure 6-5).

High-Confidence Failures

We collect and inspect highly confident test set misclassifications (confidence $\geq 85\%$). For CLEVR, out of $15,000$ test images, the baseline discovers between 0 and 15 examples for each target. Notably, there are no 2-cylinder misclassifications in the test set, but Bayes-TrEx successful generated some (Figure 6-9(b)).

From the 10,000 test examples in (Fashion-)MNIST, 84 MNIST images and 802 Fashion-MNIST images were confidently misclassified. Upon closer inspection, however, we find that the a large fraction of the failures are actually due to *mislabeling*, rather than misclassification. We manually relabel all 84 MNIST misclassifications and ten Fashion-MNIST misclassifications per class, except for the trousers class which only has 3 misclassifiations. We find that the 60 out of 84 MNIST images 42 out of 93 Fashion-MNIST images are mislabeled, rather than misclassified.

Table 6.4 gives detailed statistics of the number of genuinely misclassified examples. Given the scene graph data representation, all CLEVR misclassifications are genuine. Table 6.5 visualizes some misclassified vs. mislabeled images, with additional classes in Appendix 6.6.10. Identifying mislabeled examples may be useful for correcting the dataset, but is not for our task of model understanding.

CLEVR	class	1 Sph.			1 Cube			1 Cyl.			2 Cyl.	Total
	count	5			8			15			0	28/28
MNIST	class	0	1	2	3	4	5	6	7	8	9	Total
	count	3	3	0	5	3	1	3	4	0	2	24/84
Fashion	class	0	1	2	3	4	5	6	7	8	9	Total
	count	2	0	9	4	9	1	3	2	1	10	51/93

Table 6.4: Number of *genuine* high-confidence misclassifications from the test set. Counts for CLEVR and MNIST are for the entire test set; counts for Fashion-MNIST are computed from ten random high-confidence misclassifications per class, except for trousers which only has 3 misclassifications. Fashion-MNIST classes 0-9 corresponds to T-shirt, trousers, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot, in that order.

Novel Class Extrapolation

In Section 6.4.8 analysis, we find that the model mistakes some bags for ankle boots. Interestingly, this propensity is not evident from test set evaluations: the test set confusion matrix in Appendix 6.6.10 shows that no bags are misclassified as ankle boots. This provides further evidence of the value of holistic evaluations with Bayes-TrEx, beyond standard test set evaluations.

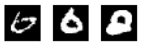
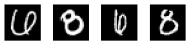
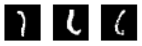
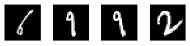
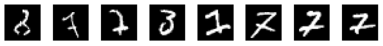
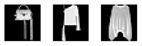
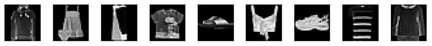
Class	Cause	Images
0	Misclassified	
	Mislabeled	
1	Misclassified	
	Mislabeled	
2	Misclassified	∅
	Mislabeled	
Trouser	Misclassified	∅
	Mislabeled	
Bag	Misclassified	
	Mislabeled	

Table 6.5: High-confidence misclassifications from the test set. The majority are due to incorrect ground truth labels, not classifier failures. Full table of all classes in Appendix 6.6.10.

6.5 Discussion and Conclusion

We presented Bayes-TrEx, a Bayesian inference approach for generating new examples that trigger various model behaviors. These examples can be further analyzed with downstream interpretability methods (Figure 6-2 and Appendix 6.6.11). To make Bayes-TrEx easier for model designers to use, future work should develop methods to cluster and visualize trends in the generated examples, as well as to estimate the overall coverage of the level set.

For organic data, the underlying data distributions can be learned with VAEs or GANs. These have known limitations in sample diversity [8] and are computationally expensive to train, especially for high resolution images. In principle, Bayes-TrEx is agnostic to the distribution learner form and can benefit from future research in this area. Applying MCMC sampling to high dimensional latent spaces is an open problem, so Bayes-TrEx is currently limited to low dimensional latent spaces.

Finally, while we only analyze classification models with Bayes-TrEx, it also has the potential for analyzing structured domains such as machine translation or robotic control. Indeed, after the appendices of this work, we present RoCUS, an extension of Bayes-TrEx to analyze robotic controllers.

6.6 Appendix

6.6.1 Network Architecture for MNIST & Fashion-MNIST

For all experiments on MNIST and Fashion-MNIST, the VAE architecture is shown in Table 6.6 (left), and the GAN architecture is shown in Table 6.6 (right). For all experiments on MNIST and Fashion-MNIST except for the domain adaptation analysis, the classifier architecture is shown in Table 6.7 (left). The classifier used in the domain adaptation analysis is the LeNet architecture, following the provided source code, shown in Table 6.7 (right). VAEs and GANs are trained with binary cross entropy loss. Classifiers are trained with negative log likelihood loss.

Encoder input: $28 \times 28 \times 1$
Flatten
Fully-connected 784×400
ReLU
Mean: Fully-connected 400×5 Log-variance: Fully-connected 400×5
Decoder input: 5 (latent dimension)
Fully-connected 5×400
ReLU
Fully-connected 400×784
Reshape $28 \times 28 \times 1$
Sigmoid

Input: 5 (latent dimension)
Reshape $1 \times 1 \times 5$
Conv-transpose: 512 filters, size=4×4, stride = 1
Batch-norm, ReLU
Conv-transpose: 256 filters, size=4×4, stride = 2
Batch-norm, ReLU
Conv-transpose: 128 filters, size=4×4, stride = 2
Batch-norm, ReLU
Conv-transpose: 64 filters, size=4×4, stride = 2
Batch-norm, ReLU
Conv-transpose: 1 filters, size=1×1, stride = 1
Sigmoid

Table 6.6: Left: VAE architecture; right: GAN architecture.

Input: $28 \times 28 \times 1$
Conv: 32 filters, size = 3×3, stride = 1
ReLU
Conv: 64 filters, size = 3×3, stride = 1
Drop-out, prob = 0.25
Max-pool, size = 2×2
Flatten
Fully-connected 9216×128
ReLU
Drop-out, prob = 0.5
Fully-connected 128×10
Soft-max

Input: $28 \times 28 \times 1$
Conv: 20 filters, size = 5×5, stride = 1
ReLU
Max-pool, size = 2×2
Conv: 50 filters, size = 5×5, stride = 1
ReLU
Max-pool, size = 2×2
Flatten
Fully-connected 800×500
ReLU
Fully-connected 500×10
Soft-max

Table 6.7: Left: classifier architecture in all experiments except domain adaptation analysis; right: LeNet classifier architecture in domain adaptation analysis (used in code released by ADDA authors).

6.6.2 Fréchet Inception Distance (FID) for VAE and GAN

Table 6.8 extends Table 6.1 in Section 6.4.2 and lists the FID scores for all VAE and GAN models that we use. These FID scores reveal the GANs are better approximations of the underlying data distributions. Models trained on "all" data are used for high confidence, ambivalent confidence, confidence interpolation and domain adaptation settings. Models trained on data "Without [class]" are used for high-confidence failure settings. Models trained on select classes ({2, 4, 5, 7, 8} and {0, 1, 4, 7, 8}) are used for the novel class extrapolation settings.

GAN MNIST		GAN Fashion-MNIST		VAE MNIST		VAE Fashion-MNIST	
Data Source	FID	Data Source	FID	Data Source	FID	Data Source	FID
All	11.83	All	29.44	All	72.33	All	87.89
Without 0	12.10	Without 0	28.91	Without 0	71.28	Without 0	89.21
Without 1	12.08	Without 1	31.18	Without 1	75.36	Without 1	92.02
Without 2	13.57	Without 2	30.11	Without 2	64.77	Without 2	91.20
Without 3	12.71	Without 3	28.95	Without 3	63.66	Without 3	85.51
Without 4	12.25	Without 4	30.43	Without 4	66.96	Without 4	88.38
Without 5	12.21	Without 5	27.67	Without 5	63.31	Without 5	84.17
Without 6	11.86	Without 6	29.68	Without 6	67.64	Without 6	85.58
Without 7	11.64	Without 7	28.56	Without 7	62.45	Without 7	84.93
Without 8	12.31	Without 8	30.87	Without 8	64.14	Without 8	83.66
Without 9	12.34	Without 9	29.22	Without 9	66.57	Without 9	81.48
{2, 4, 5, 7, 8}	13.45	{0, 1, 4, 7, 8}	33.11	—	—	—	—

Table 6.8: Fréchet Inception Distance (FID) scores for all learned data distributions; a lower score indicates a better distribution fit. Results are computed across 1000 samples. Fashion-MNIST classes are 0: T-shirt, 1: Trouser, 2: Pullover, 3: Dress, 4: Coat, 5: Sandal, 6: Shirt, 7: Sneaker, 8: Bag, and 9: Ankle Boot.

6.6.3 Quantitative Prediction Confidence Summary

Table 6.9, 6.10, and 6.11 present the extension of Table 6.2 in Section 6.4.3. These results show that the inferred samples have predicted confidence closely matching the specified confidence targets. This indicates the MCMC methods used by Bayes-TrEx are successful for the tested domains and scenarios. Queries for 5 Cubes in the novel class extrapolation CLEVR experiments use a stopping criterion of 1500 samples instead of the standard 500. Averages across 10 inference runs are reported.

Target	Prediction Confidence
$\mathbf{p}_0 = 1$	0.999 ± 0.006
$\mathbf{p}_1 = 1$	0.999 ± 0.003
$\mathbf{p}_2 = 1$	0.999 ± 0.006
$\mathbf{p}_3 = 1$	0.999 ± 0.005
$\mathbf{p}_4 = 1$	0.998 ± 0.008
$\mathbf{p}_5 = 1$	0.999 ± 0.006
$\mathbf{p}_6 = 1$	0.998 ± 0.007
$\mathbf{p}_7 = 1$	0.998 ± 0.007
$\mathbf{p}_8 = 1$	0.999 ± 0.004
$\mathbf{p}_9 = 1$	0.998 ± 0.007
$\mathbf{p}_{\text{T-shirt}} = 1$	0.991 ± 0.016
$\mathbf{p}_{\text{Trouser}} = 1$	0.999 ± 0.006
$\mathbf{p}_{\text{Pullover}} = 1$	0.984 ± 0.019
$\mathbf{p}_{\text{Dress}} = 1$	0.993 ± 0.008
$\mathbf{p}_{\text{Coat}} = 1$	0.983 ± 0.021
$\mathbf{p}_{\text{Sandal}} = 1$	0.998 ± 0.008
$\mathbf{p}_{\text{Shirt}} = 1$	0.987 ± 0.020
$\mathbf{p}_{\text{Sneaker}} = 1$	0.994 ± 0.016
$\mathbf{p}_{\text{Bag}} = 1$	0.999 ± 0.006
$\mathbf{p}_{\text{Ankle Boot}} = 1$	0.996 ± 0.012
$\mathbf{p}_{\text{5 Spheres}} = 1$	0.943 ± 0.020
$\mathbf{p}_{\text{2 Blue Spheres}} = 1$	0.892 ± 0.245

Target	Prediction Confidence
$\mathbf{p}_0 = 1$	0.981 ± 0.027
$\mathbf{p}_1 = 1$	0.953 ± 0.028
$\mathbf{p}_2 = 1$	0.968 ± 0.028
$\mathbf{p}_3 = 1$	0.969 ± 0.027
$\mathbf{p}_4 = 1$	0.955 ± 0.030
$\mathbf{p}_5 = 1$	0.990 ± 0.018
$\mathbf{p}_6 = 1$	0.970 ± 0.026
$\mathbf{p}_7 = 1$	0.968 ± 0.029
$\mathbf{p}_8 = 1$	0.982 ± 0.024
$\mathbf{p}_9 = 1$	0.983 ± 0.022
$\mathbf{p}_{\text{T-shirt}} = 1$	0.964 ± 0.029
$\mathbf{p}_{\text{Trouser}} = 1$	(sample failure)
$\mathbf{p}_{\text{Pullover}} = 1$	0.886 ± 0.027
$\mathbf{p}_{\text{Dress}} = 1$	0.970 ± 0.026
$\mathbf{p}_{\text{Coat}} = 1$	0.938 ± 0.030
$\mathbf{p}_{\text{Sandal}} = 1$	0.968 ± 0.030
$\mathbf{p}_{\text{Shirt}} = 1$	0.938 ± 0.032
$\mathbf{p}_{\text{Sneaker}} = 1$	0.969 ± 0.028
$\mathbf{p}_{\text{Bag}} = 1$	0.967 ± 0.026
$\mathbf{p}_{\text{Ankle Boot}} = 1$	0.971 ± 0.027
$\mathbf{p}_{\text{1 Cube}} = 1$	0.929 ± 0.062
$\mathbf{p}_{\text{1 Cylinder}} = 1$	0.972 ± 0.021
$\mathbf{p}_{\text{1 Sphere}} = 1$	0.843 ± 0.266
$\mathbf{p}_{\text{2 Cylinders}} = 1$	0.545 ± 0.230

Table 6.9: Prediction confidence for samples on high-confidence examples (left) and high confidence misclassifications (right).

Target	Prediction Confidence	Target	Prediction Confidence
$\mathbf{p}_8 = 0.0, \mathbf{p}_9 = 1.0$	$(0.002 \pm 0.006, 0.990 \pm 0.016)$	$\mathbf{p}_{\text{T-shirt}} = 0.0, \mathbf{p}_{\text{Trousers}} = 1.0$	$(0.001 \pm 0.004, 0.995 \pm 0.012)$
$\mathbf{p}_8 = 0.1, \mathbf{p}_9 = 0.9$	$(0.030 \pm 0.039, 0.936 \pm 0.051)$	$\mathbf{p}_{\text{T-shirt}} = 0.1, \mathbf{p}_{\text{Trousers}} = 0.9$	$(0.026 \pm 0.035, 0.950 \pm 0.050)$
$\mathbf{p}_8 = 0.2, \mathbf{p}_9 = 0.8$	$(0.170 \pm 0.039, 0.788 \pm 0.040)$	$\mathbf{p}_{\text{T-shirt}} = 0.2, \mathbf{p}_{\text{Trousers}} = 0.8$	$(0.166 \pm 0.040, 0.791 \pm 0.041)$
$\mathbf{p}_8 = 0.3, \mathbf{p}_9 = 0.7$	$(0.275 \pm 0.041, 0.682 \pm 0.040)$	$\mathbf{p}_{\text{T-shirt}} = 0.3, \mathbf{p}_{\text{Trousers}} = 0.7$	$(0.275 \pm 0.037, 0.686 \pm 0.038)$
$\mathbf{p}_8 = 0.4, \mathbf{p}_9 = 0.6$	$(0.378 \pm 0.040, 0.578 \pm 0.040)$	$\mathbf{p}_{\text{T-shirt}} = 0.4, \mathbf{p}_{\text{Trousers}} = 0.6$	$(0.379 \pm 0.038, 0.586 \pm 0.038)$
$\mathbf{p}_8 = 0.5, \mathbf{p}_9 = 0.5$	$(0.477 \pm 0.039, 0.477 \pm 0.039)$	$\mathbf{p}_{\text{T-shirt}} = 0.5, \mathbf{p}_{\text{Trousers}} = 0.5$	$(0.436 \pm 0.040, 0.459 \pm 0.040)$
$\mathbf{p}_8 = 0.6, \mathbf{p}_9 = 0.4$	$(0.581 \pm 0.038, 0.374 \pm 0.039)$	$\mathbf{p}_{\text{T-shirt}} = 0.6, \mathbf{p}_{\text{Trousers}} = 0.4$	$(0.583 \pm 0.038, 0.382 \pm 0.037)$
$\mathbf{p}_8 = 0.7, \mathbf{p}_9 = 0.3$	$(0.680 \pm 0.041, 0.275 \pm 0.039)$	$\mathbf{p}_{\text{T-shirt}} = 0.7, \mathbf{p}_{\text{Trousers}} = 0.3$	$(0.685 \pm 0.039, 0.281 \pm 0.040)$
$\mathbf{p}_8 = 0.8, \mathbf{p}_9 = 0.2$	$(0.788 \pm 0.040, 0.167 \pm 0.041)$	$\mathbf{p}_{\text{T-shirt}} = 0.8, \mathbf{p}_{\text{Trousers}} = 0.2$	$(0.790 \pm 0.037, 0.177 \pm 0.037)$
$\mathbf{p}_8 = 0.9, \mathbf{p}_9 = 0.1$	$(0.926 \pm 0.050, 0.039 \pm 0.040)$	$\mathbf{p}_{\text{T-shirt}} = 0.9, \mathbf{p}_{\text{Trousers}} = 0.1$	$(0.936 \pm 0.045, 0.029 \pm 0.041)$
$\mathbf{p}_8 = 1.0, \mathbf{p}_9 = 0.0$	$(0.989 \pm 0.016, 0.002 \pm 0.007)$	$\mathbf{p}_{\text{T-shirt}} = 1.0, \mathbf{p}_{\text{Trousers}} = 0.0$	$(0.985 \pm 0.019, 0.000 \pm 0.003)$

Table 6.10: (Fashion-)MNIST confidence interpolation.

Target	Prediction Confidence	Target	Prediction Confidence
$\mathbf{p}_0 = 1$	0.976 ± 0.025	$\mathbf{p}_0 = 1$	0.996 ± 0.011
$\mathbf{p}_1 = 1$	0.988 ± 0.186	$\mathbf{p}_1 = 1$	0.994 ± 0.014
$\mathbf{p}_3 = 1$	0.987 ± 0.020	$\mathbf{p}_2 = 1$	0.998 ± 0.008
$\mathbf{p}_6 = 1$	0.989 ± 0.018	$\mathbf{p}_3 = 1$	0.994 ± 0.015
$\mathbf{p}_9 = 1$	0.995 ± 0.013	$\mathbf{p}_4 = 1$	0.997 ± 0.010
$\mathbf{p}_{\text{Pullover}} = 1$	0.991 ± 0.016	$\mathbf{p}_5 = 1$	0.998 ± 0.007
$\mathbf{p}_{\text{Dress}} = 1$	0.994 ± 0.013	$\mathbf{p}_6 = 1$	0.996 ± 0.011
$\mathbf{p}_{\text{Sandal}} = 1$	0.995 ± 0.013	$\mathbf{p}_7 = 1$	0.996 ± 0.011
$\mathbf{p}_{\text{Shirt}} = 1$	0.994 ± 0.012	$\mathbf{p}_8 = 1$	0.995 ± 0.013
$\mathbf{p}_{\text{Ankle Boot}} = 1$	0.993 ± 0.015	$\mathbf{p}_9 = 1$	0.996 ± 0.012
$\mathbf{p}_{\text{1 Cube}} = 1$	0.983 ± 0.014		
$\mathbf{p}_{\text{1 Cylinder}} = 1$	0.959 ± 0.031		
$\mathbf{p}_{\text{1 Sphere}} = 1$	0.969 ± 0.022		
$\mathbf{p}_{\text{5 Cubes}} = 1$	0.921 ± 0.029		

Table 6.11: Prediction confidence for novel class extrapolation (left) and covariate shift (right).

Figure 6-13 shows the pairwise prediction confidence for ambivalent examples.

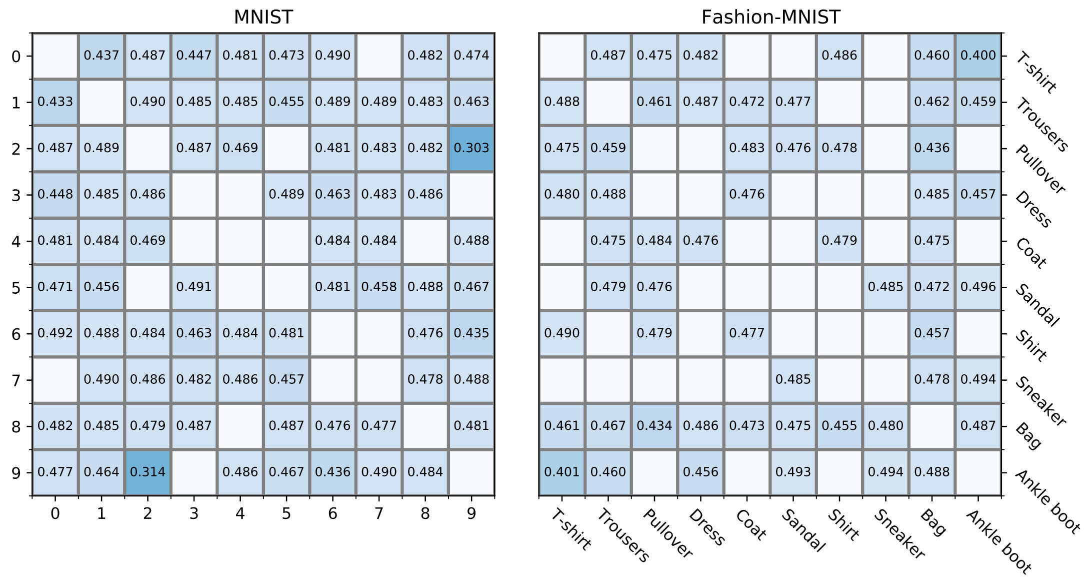
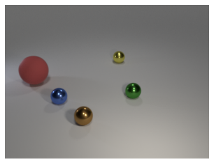
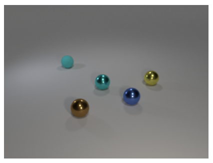


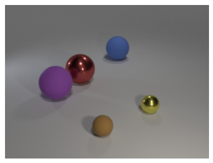
Figure 6-13: Prediction confidence for (Fashion-)MNIST ambivalent examples. For each class combination, the lower triangle shows the the confidence for the digit denoted on the horizontal axis, and the upper triangle shows the confidence for the digit on the vertical axis. For example, for the MNIST class combination 9v0, the classifier confidence in class 0 is 0.477 (the bottom left entry) while the classifier confidence in class 9 is 0.474 (the top right entry). Diagonal entries are blank since they have the same class on row and column. Off-diagonal blank entries indicate that Bayes-TrEx does not find ambivalent samples for that particular class pair.

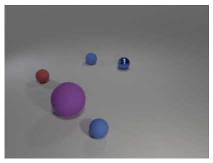
6.6.4 High-Confidence Analysis

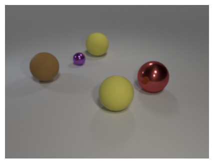
Figure 6-14 presents additional high-confidence CLEVR examples and the classifier's predictions.

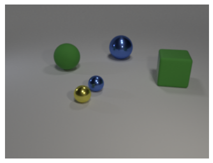

(a) $P_{5\text{ Sph.}} = 94.8\%$

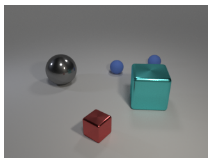

(b) $P_{5\text{ Sph.}} = 94.5\%$

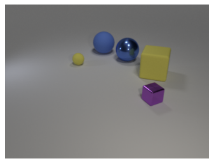

(c) $P_{5\text{ Sph.}} = 94.6\%$

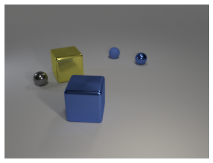

(d) $P_{5\text{ Sph.}} = 95.2\%$


(e) $P_{5\text{ Sph.}} = 92.0\%$


(f) $P_{2\text{ Blue}} = 96.3\%$


(g) $P_{2\text{ Blue}} = 96.1\%$


(h) $P_{2\text{ Blue}} = 94.9\%$


(i) $P_{2\text{ Blue}} = 96.8\%$

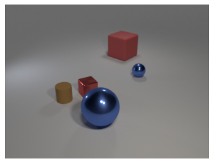
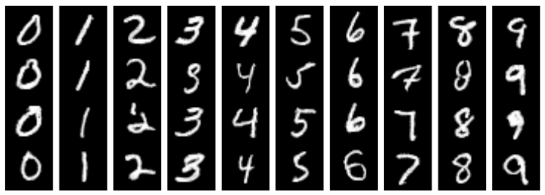

(j) $P_{2\text{ Blue}} = 97.8\%$

Figure 6-14: Top: selected examples classified as containing 5 spheres with high confidence. Bottom: selected examples classified as containing 2 blue spheres with high confidence.

Figure 6-15 presents additional high-confidence (Fashion-)MNIST examples.


(a) MNIST

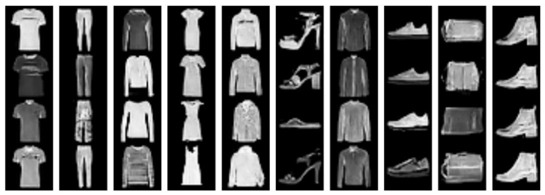

(b) Fashion-MNIST

Figure 6-15: High-confidence examples from MNIST and Fashion-MNIST. There are no misclassifications. MNIST columns represent digit 0 to 9, respectively. Fashion-MNIST columns represent T-shirt, trousers, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot, respectively.

6.6.5 Ambivalent Confidence Analysis

Figure 6-16 presents additional visualizations for two pairs, Digit 1 vs. Digit 7 from MNIST and T-shirt vs. Pullover from Fashion-MNIST. The confidence plots in the middle confirm that the neural network is indeed making the ambivalent predictions. The *t*-SNE [101] latent space visualizations at the bottom indicate that the samples lie around the class boundaries and are also in-distribution (i.e., having close proximity to those sampled from the prior).

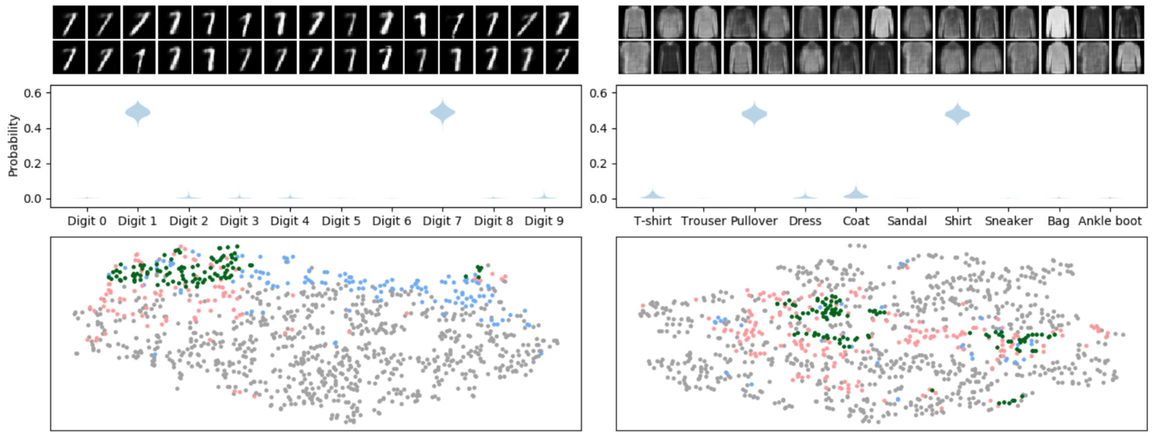


Figure 6-16: Left: ambivalent samples for digit 1 vs. 7 in MNIST. Right: ambivalent samples for pullover vs. shirt in Fashion-MNIST. Top: 30 sampled images. Middle: classifier confidence plots on the samples. Bottom: *t*-SNE latent space visualization: green dots represent ambivalent samples from the posterior, red and blue dots represents samples from the prior that are predicted by the classifier to be either class of interest, and gray dots represents other samples from the prior. The ambivalent samples are on the class boundaries.

6.6.6 Ambivalence with GAN and Modified Classifier

Figure 6-17 shows the ambivalent confidence samples for 0v1, 1v2, ..., 9v0 using the GAN-learned distribution when the classifier is trained with the custom KL loss described in Eq. 6.15.

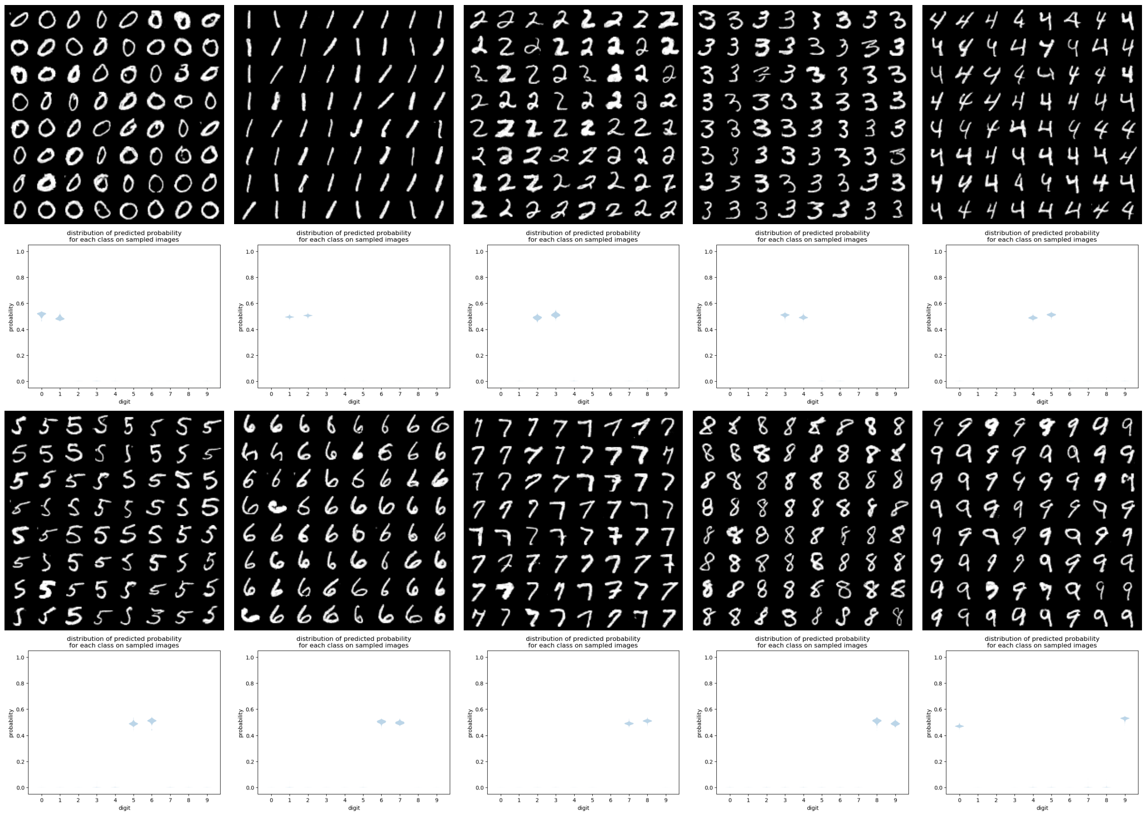
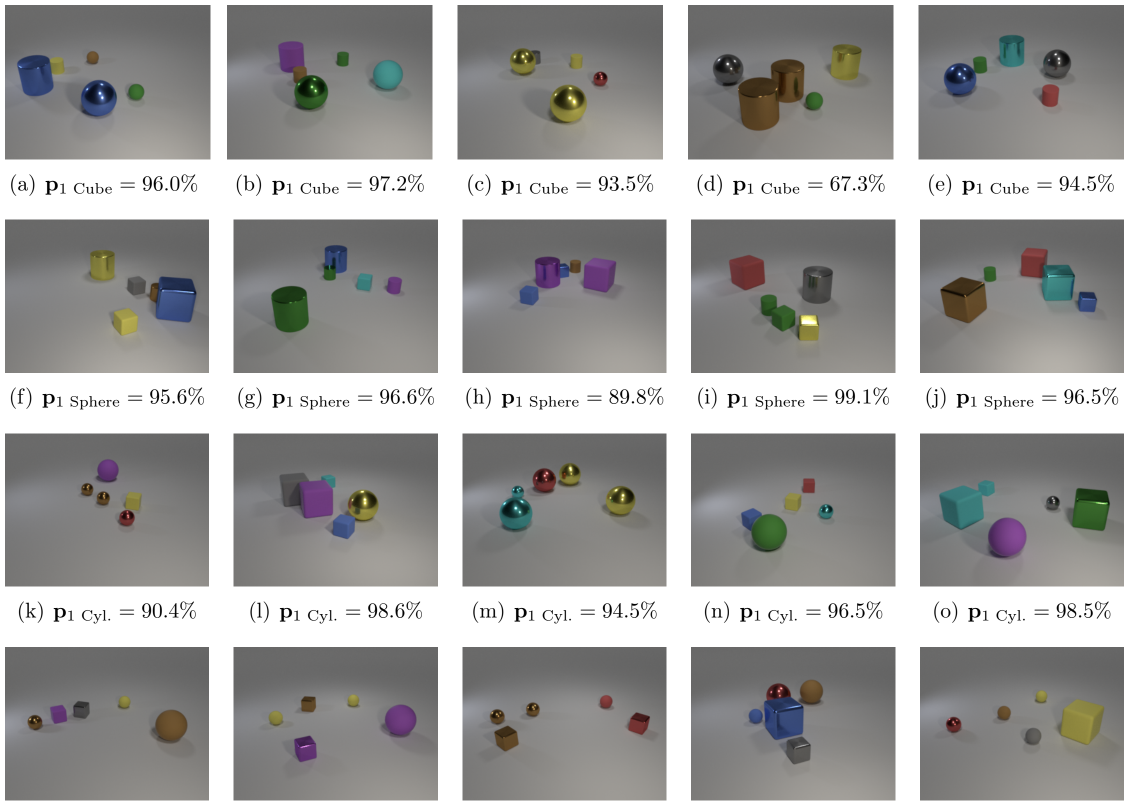


Figure 6-17: Sampling results with an explicitly ambivalent classifier and a GAN-learned distribution. Top 2 rows: digit i vs. $i+1$ for $i \in \{0,1,2,3,4\}$. Bottom 2 rows: digit i vs. $i+1$ (mod 10) for $i \in \{5,6,7,8,9\}$.

6.6.7 High-Confidence Failure Analysis

Figure 6-18 shows such examples for CLEVR. For each target inference (e.g. "1 Cube"), we exclude objects belonging to the target class from the data distribution.



(a) $\mathbf{p}_{1\ \text{Cube}} = 96.0\%$ (b) $\mathbf{p}_{1\ \text{Cube}} = 97.2\%$ (c) $\mathbf{p}_{1\ \text{Cube}} = 93.5\%$ (d) $\mathbf{p}_{1\ \text{Cube}} = 67.3\%$ (e) $\mathbf{p}_{1\ \text{Cube}} = 94.5\%$

(f) $\mathbf{p}_{1\ \text{Sphere}} = 95.6\%$ (g) $\mathbf{p}_{1\ \text{Sphere}} = 96.6\%$ (h) $\mathbf{p}_{1\ \text{Sphere}} = 89.8\%$ (i) $\mathbf{p}_{1\ \text{Sphere}} = 99.1\%$ (j) $\mathbf{p}_{1\ \text{Sphere}} = 96.5\%$

(k) $\mathbf{p}_{1\ \text{Cyl.}} = 90.4\%$ (l) $\mathbf{p}_{1\ \text{Cyl.}} = 98.6\%$ (m) $\mathbf{p}_{1\ \text{Cyl.}} = 94.5\%$ (n) $\mathbf{p}_{1\ \text{Cyl.}} = 96.5\%$ (o) $\mathbf{p}_{1\ \text{Cyl.}} = 98.5\%$

(p) $\mathbf{p}_{2\ \text{Cyl.}} = 85.9\%$ (q) $\mathbf{p}_{2\ \text{Cyl.}} = 60.2\%$ (r) $\mathbf{p}_{2\ \text{Cyl.}} = 79.4\%$ (s) $\mathbf{p}_{2\ \text{Cyl.}} = 48.4\%$ (t) $\mathbf{p}_{2\ \text{Cyl.}} = 60.5\%$

Figure 6-18: High-confidence misclassified examples and their associated prediction confidences for CLEVR. For each target constraint (e.g., "1 Cube"), objects from the target class (e.g., cubes) are excluded from the data distribution. The resultant images are composed entirely of non-target-class objects, (e.g., cylinders and spheres).

Figure 6-19 presents high-confidence misclassifications for each classes of MNIST, with digit 0-4 on the top two rows and digit 5-9 on the bottom two rows.

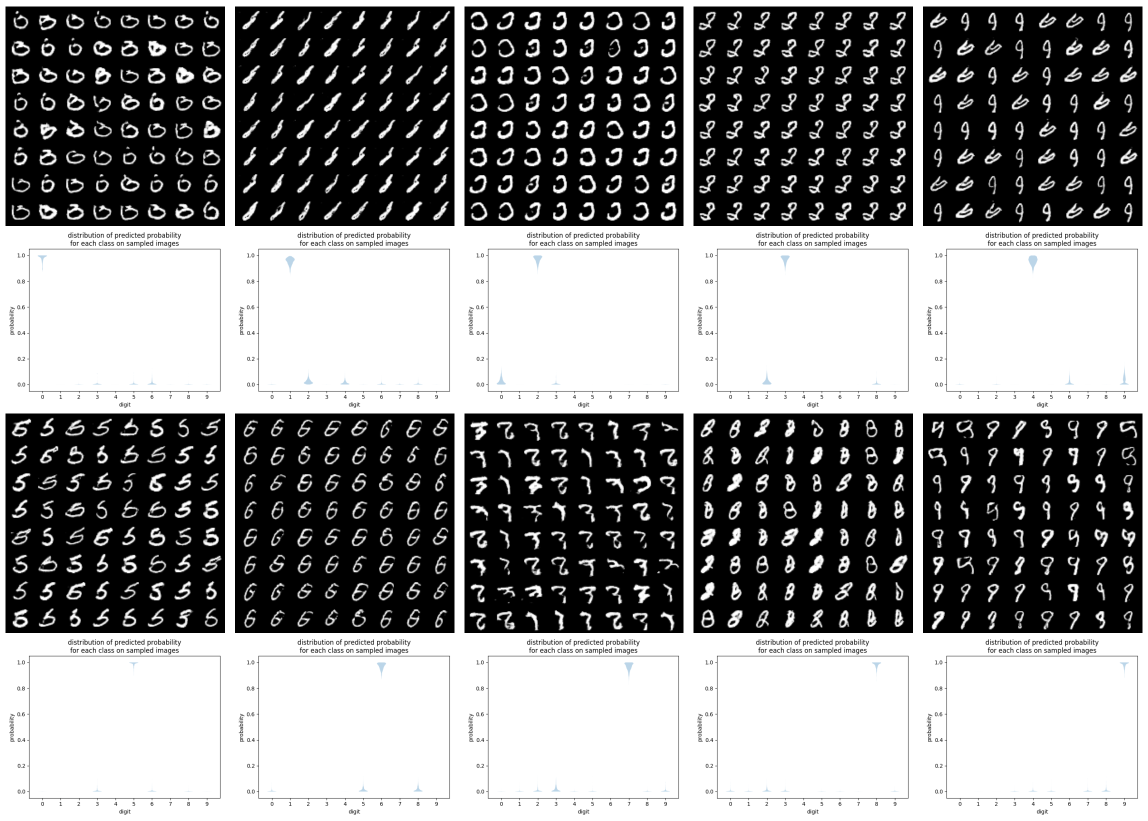


Figure 6-19: Samples and violin plots for high-confidence misclassified examples. Top two rows: 0-4; bottom two rows: 5-9.

Figure 6-20 presents high-confidence misclassifications for each classes of Fashion-MNIST, with T-shirt, trousers pullover, dress and coat on the top two rows and sandal, shirt, sneaker, bag and ankle boot on the bottom two rows. The confidence plot for the trousers samples indicates that the sampling is not successful.

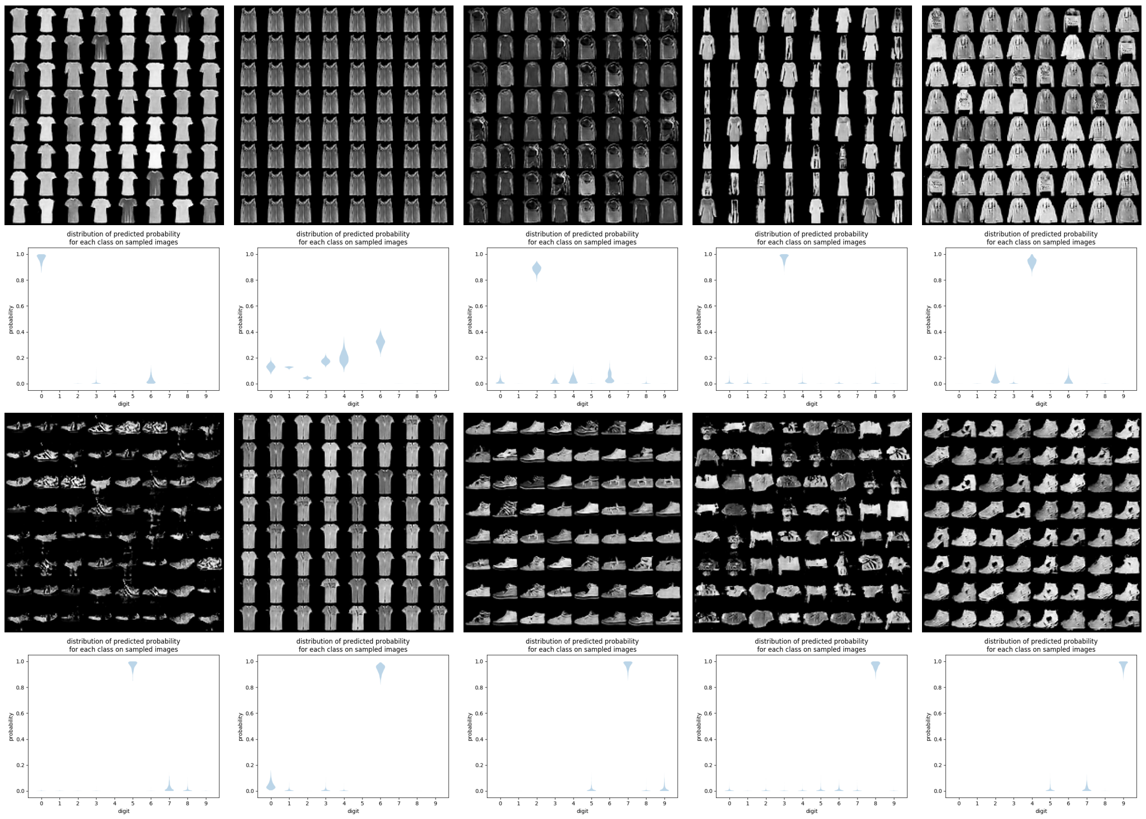
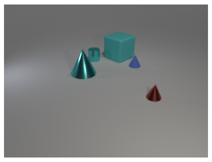
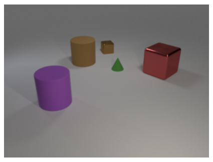


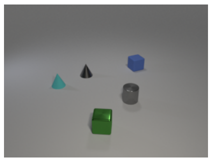
Figure 6-20: Samples and violin plots for high-confidence misclassified examples. Top row: T-shirt, trousers (sample failure), pullover, dress, coat. Bottom row: sandal, shirt, sneaker, bag, ankle boot.

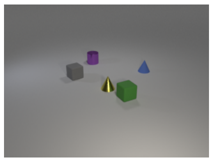
6.6.8 Novel Class Extrapolation Analysis

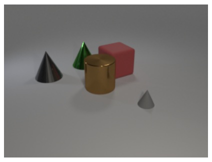
Figure 6-21 shows novel class extrapolation examples for CLEVR.

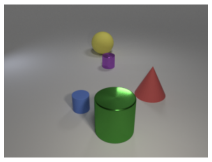

(a) $\mathbf{p}_{1 \text{ Sph.}} = 99.3\%$

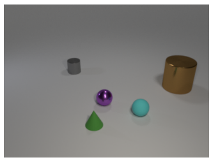

(b) $\mathbf{p}_{1 \text{ Sph.}} = 95.9\%$

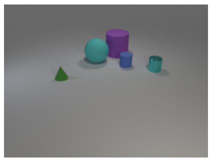

(c) $\mathbf{p}_{1 \text{ Sph.}} = 99.3\%$

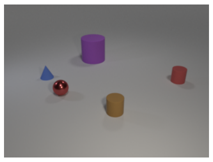

(d) $\mathbf{p}_{1 \text{ Sph.}} = 97.7\%$

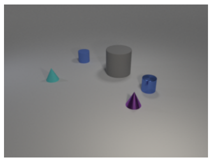

(e) $\mathbf{p}_{1 \text{ Sph.}} = 97.3\%$

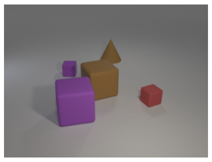

(f) $\mathbf{p}_{1 \text{ Cube}} = 99.2\%$

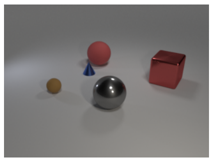

(g) $\mathbf{p}_{1 \text{ Cube}} = 97.5\%$

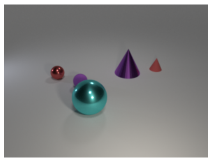

(h) $\mathbf{p}_{1 \text{ Cube}} = 98.7\%$

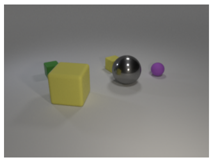

(i) $\mathbf{p}_{1 \text{ Cube}} = 99.0\%$

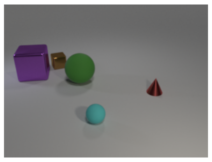

(j) $\mathbf{p}_{1 \text{ Cube}} = 98.7\%$

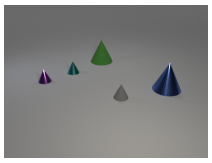

(k) $\mathbf{p}_{1 \text{ Cyl.}} = 96.9\%$

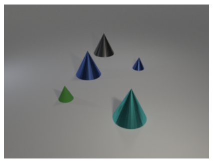

(l) $\mathbf{p}_{1 \text{ Cyl.}} = 99.1\%$

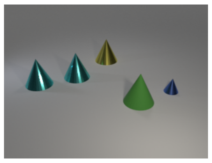

(m) $\mathbf{p}_{1 \text{ Cyl.}} = 96.5\%$

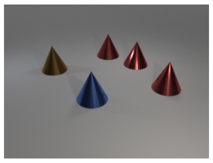

(n) $\mathbf{p}_{1 \text{ Cyl.}} = 97.2\%$


(o) $\mathbf{p}_{1 \text{ Cyl.}} = 99.0\%$


(p) $\mathbf{p}_{5 \text{ Cubes}} = 74.6\%$


(q) $\mathbf{p}_{5 \text{ Cubes}} = 89.5\%$


(r) $\mathbf{p}_{5 \text{ Cubes}} = 93.3\%$


(s) $\mathbf{p}_{5 \text{ Cubes}} = 91.6\%$

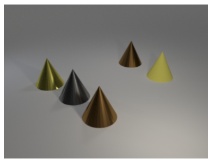

(t) $\mathbf{p}_{5 \text{ Cubes}} = 89.9\%$

Figure 6-21: Sampled novel class extrapolation examples and their associated prediction confidences. Similar to high confidence misclassified examples, for each target constraint (e.g., "1 Cube"), we remove examples of the target class (e.g., cubes) from the data distribution, but add to the cone object to it, a novel class not present in the training distribution. 6-21(n) is the only example which by chance does not include a novel class object.

Figure 6-22 shows examples for novel-class extrapolation on MNIST. The classifier is trained on digit 0, 1, 3, 6 and 9, and tested on images generated by a GAN trained on digit 2, 4, 5, 7 and 8.

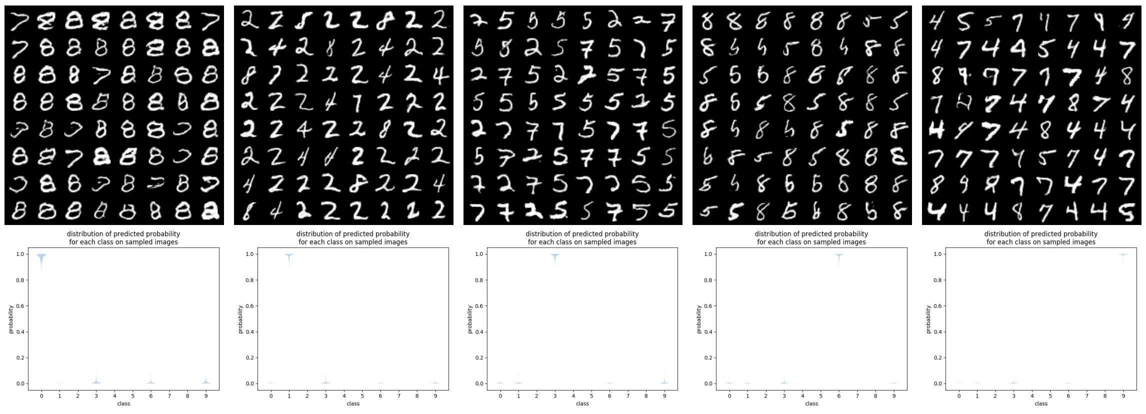


Figure 6-22: Samples and confidence plots for MNIST novel class extrapolation for digits 0, 1, 3, 6 and 9, in that order.

Figure 6-23 shows examples for novel-class extrapolation on Fashion-MNIST. The classifier is trained on pullover, dress, sandal, shirt and ankle boot, and tested on images generated by a GAN trained on T-shirt, trousers, coat, sneaker and bag.

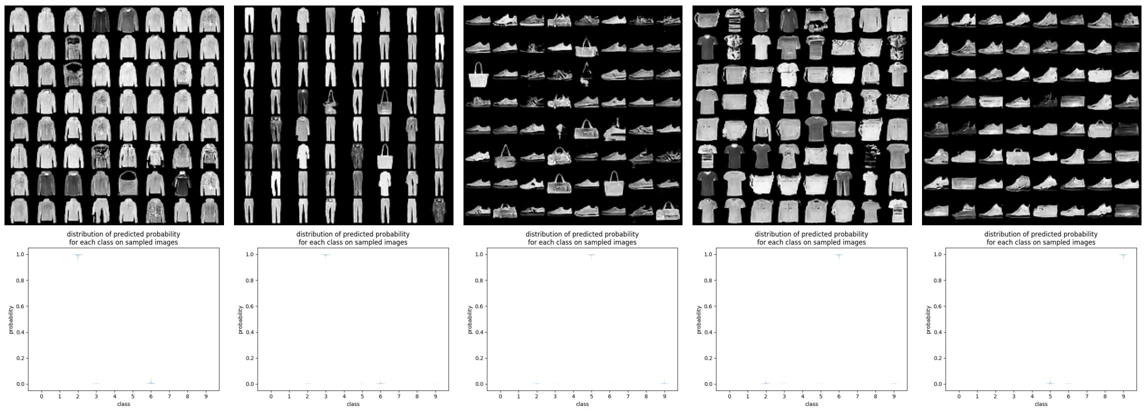


Figure 6-23: Samples and confidence plots for Fashion-MNIST novel class extrapolation for pullover, dress, sandal, shirt and ankle boot, in that order.

6.6.9 Covariate Shift Analysis

Figure 6-24 and 6-25 show additional samples and confidence plots for the baseline and ADDA model, respectively. Top two rows are for digit 0-4, and bottom two rows are for digit 5-9.

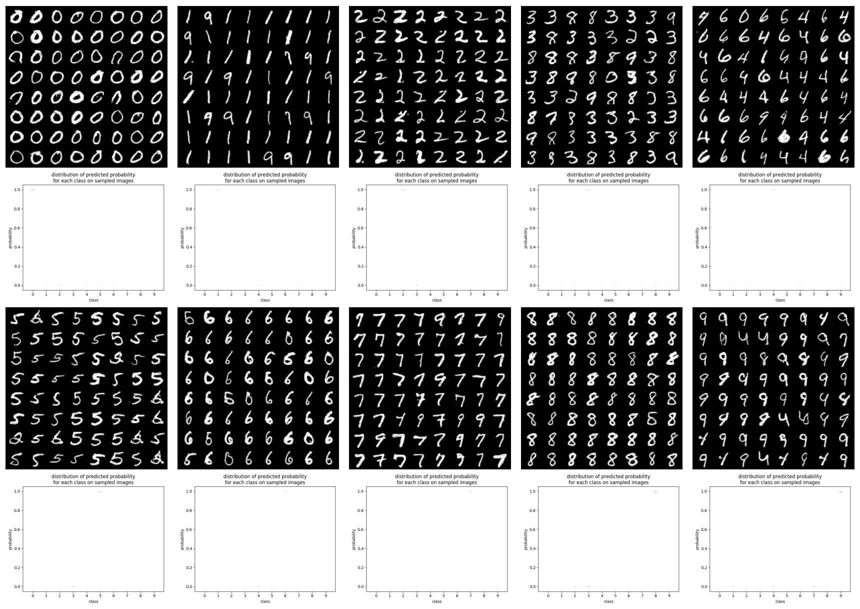


Figure 6-24: High confident MNIST samples generated for each class as predicted by the baseline model.

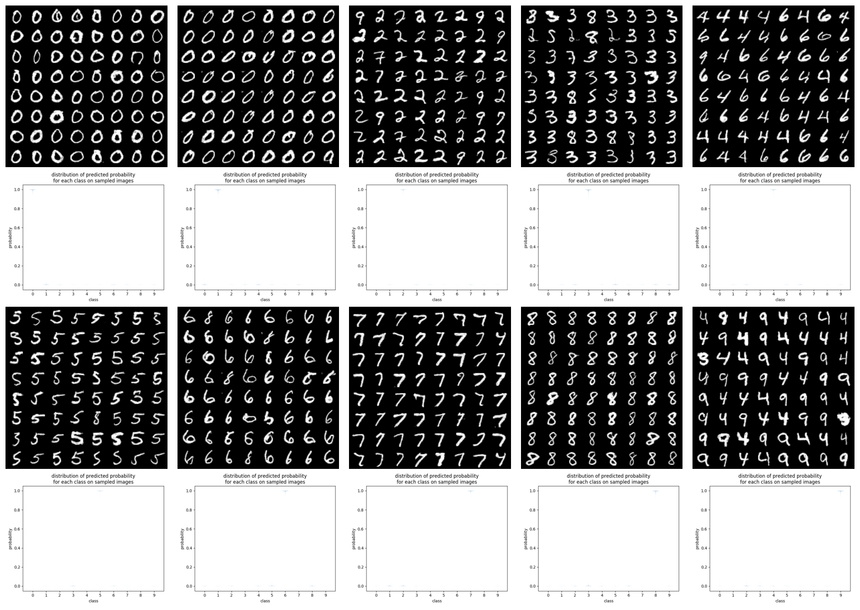


Figure 6-25: High confident MNIST samples generated for each class as predicted by the ADDA model.

6.6.10 Test Set Evaluation

Table 6.12 extends Table 6.5 in Section 6.4.10 and includes misclassified vs. mislabeled images of all (Fashion-)MNIST classes.

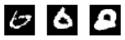
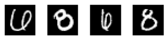
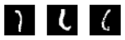
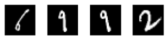
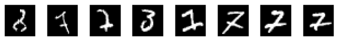
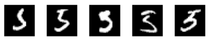
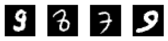
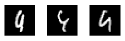
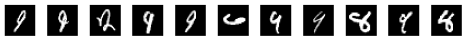
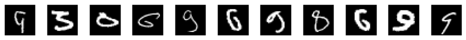
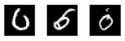
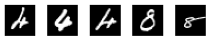
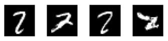
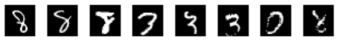
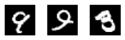
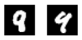
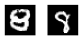
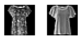
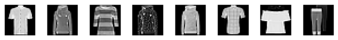
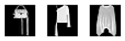
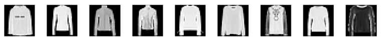
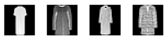
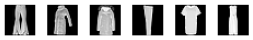
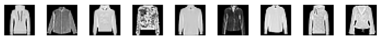
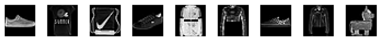
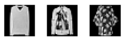
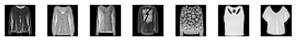
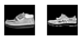
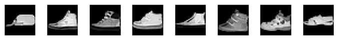
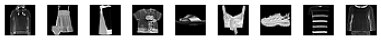
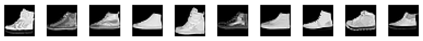
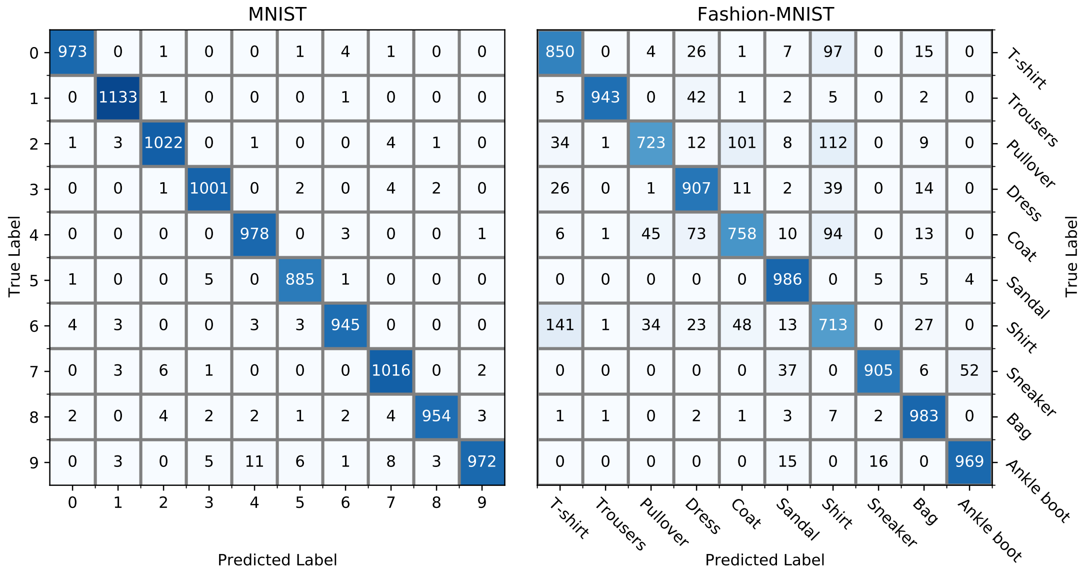
Class	Misclassified	Mislabeled
0		
1		
2	∅	
3		
4		
5		
6		
7		
8	∅	
9		
Tshirt		
Trouser	∅	
Pullover		
Dress		
Coat		
Sandal		
Shirt		
Sneaker		
Bag		
Boot		∅

Table 6.12: An alternative to using Bayes-TrEx for finding highly confident classification failures is to evaluate the high confidence example confusion matrix and associated images from the test set. Here, we show all 'misclassified' examples where the classifier failed to predict the given label for the MNIST and Fashion-MNIST datasets. For MNIST, we observe that the majority (60/84) of these images are mislabeled: for example, all of the labeled 2s clearly belong to other classes (8, 7, 7, 3, 1, 7, 7, 7, respectively). While MNIST had 84 total misclassifications, Fashion-MNIST had 802 total misclassifications. We randomly select 10 misclassifications from each class for analysis (with the exception of the "trousers" class, as there were 3 total misclassifications for this label). While Fashion-MNIST is more balanced, we again observe a majority of examples to be mislabeled ground truth (52/93) instead of misclassifications.

Figure 6-26 shows the confusion matrix of the MNIST (left) and Fashion-MNIST (right) classifiers.



MNIST

True Label \ Predicted Label	0	1	2	3	4	5	6	7	8	9
0	973	0	1	0	0	1	4	1	0	0
1	0	1133	1	0	0	0	1	0	0	0
2	1	3	1022	0	1	0	0	4	1	0
3	0	0	1	1001	0	2	0	4	2	0
4	0	0	0	0	978	0	3	0	0	1
5	1	0	0	5	0	885	1	0	0	0
6	4	3	0	0	3	3	945	0	0	0
7	0	3	6	1	0	0	0	1016	0	2
8	2	0	4	2	2	1	2	4	954	3
9	0	3	0	5	11	6	1	8	3	972

Fashion-MNIST

True Label \ Predicted Label	T-shirt	Trousers	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt	850	0	4	26	1	7	97	0	15	0
Trousers	5	943	0	42	1	2	5	0	2	0
Pullover	34	1	723	12	101	8	112	0	9	0
Dress	26	0	1	907	11	2	39	0	14	0
Coat	6	1	45	73	758	10	94	0	13	0
Sandal	0	0	0	0	0	986	0	5	5	4
Shirt	141	1	34	23	48	13	713	0	27	0
Sneaker	0	0	0	0	0	37	0	905	6	52
Bag	1	1	0	2	1	3	7	2	983	0
Ankle boot	0	0	0	0	0	15	0	16	0	969

Figure 6-26: Confusion matrices for MNIST (left) and Fashion-MNIST (right) classifiers. Note that these matrices include all test set examples, not just those which evoke high confidence responses from the classifier.

6.6.11 Bayes-TrEx with Saliency Maps

We demonstrate a simple use case of combining with Bayes-TrEx samples with downstream interpretability methods. Figure 6-27 (left) shows an image for which the classifier mistakes it to contain one cube with 93.5% accuracy. Figure 6-27 (middle) presents its SmoothGrad [143] saliency map and Figure 6-27 (right) overlays it on top of the image. We can see that the most salient part contributing to the 1-cube decision is the front red cylinder. Indeed, as we confirm in Figure 6-28, among all single object removals, removing this object has the biggest effect to the classifier confidence, decreasing it to 29.0%.

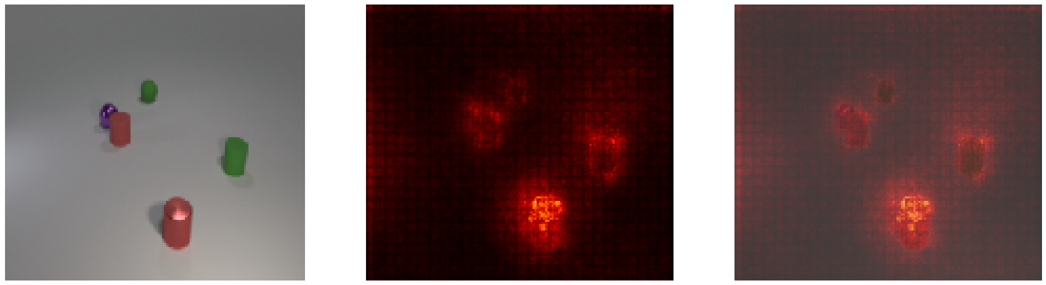
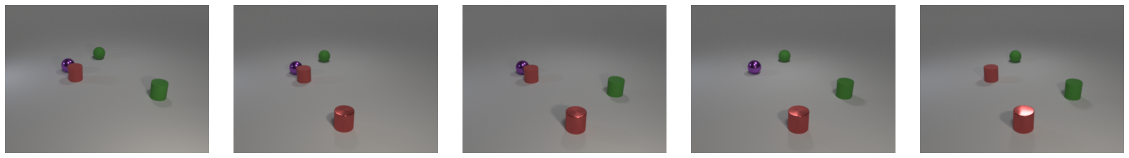


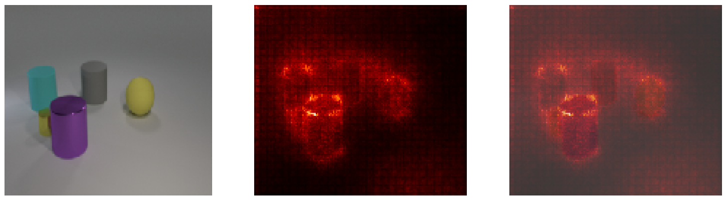
Figure 6-27: Left: the original image, preprocessed for classification by resizing and normalizing. The classifier is 93.5% confident this scene contains 1 cube, when in fact it is composed of 3 cylinders and 2 spheres. Middle: the SmoothGrad saliency map for this input. Right: the saliency map overlaid upon the original image. This saliency map most strongly highlights the red metal cylinder, indicating that this cylinder is likely the cause of the misclassification.

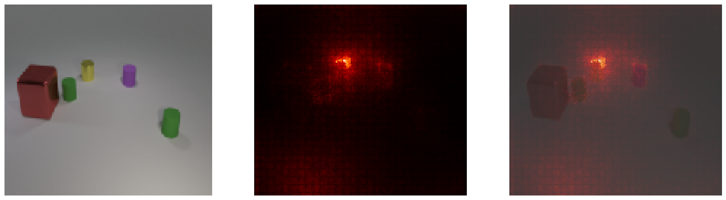


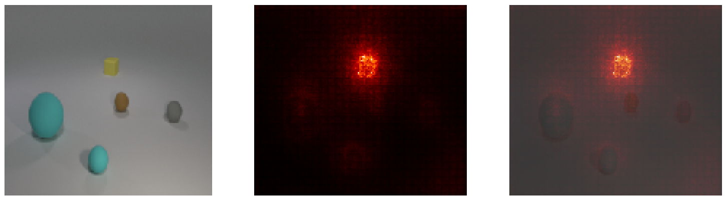
(a) $\mathbf{p}_{1\text{ Cube}} = 29.0\%$ (b) $\mathbf{p}_{1\text{ Cube}} = 68.5\%$ (c) $\mathbf{p}_{1\text{ Cube}} = 81.2\%$ (d) $\mathbf{p}_{1\text{ Cube}} = 99.0\%$ (e) $\mathbf{p}_{1\text{ Cube}} = 99.4\%$

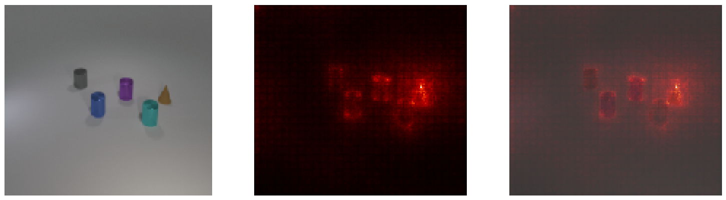
Figure 6-28: Prediction confidence for 1-cube after every single object is removed in turn. As suggested by the saliency map, the removal of the red metal cylinder most prominently reduces the classification confidence, from 93.5% to 29.0%.

Figure 6-29 presents additional case studies with the same setup. Note that Figure 6-29(e) shows a failure of SmoothGrad.


(a) Original image: $\mathbf{p}_{1\text{ Cube}} = 85.5\%$. Purple cylinder removed: $\mathbf{p}_{1\text{ Cube}} = 1.9\%$


(b) Original image: $\mathbf{p}_{1\text{ Sphere}} = 97.9\%$. Yellow cylinder removed: $\mathbf{p}_{1\text{ Sphere}} = 5.2\%$


(c) Original image: $\mathbf{p}_{1\text{ Cylinder}} = 85.4\%$. Red sphere removed: $\mathbf{p}_{1\text{ Cylinder}} = 0.9\%$


(d) Original image: $\mathbf{p}_{1\text{ Cube}} = 99.7\%$. Cone removed: $\mathbf{p}_{1\text{ Cube}} = 0.4\%$

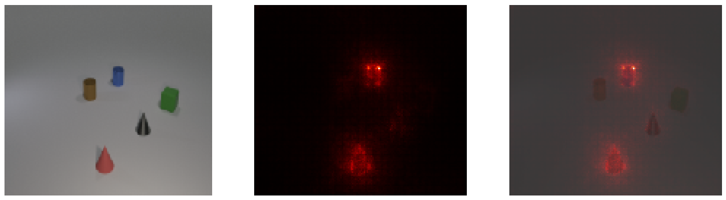

(e) Original image: $\mathbf{p}_{1\text{ Sphere}} = 98.0\%$. Gray cone removed: $\mathbf{p}_{1\text{ Sphere}} = 0.3\%$

Figure 6-29: More BAYES-TREX samples and their saliency maps. Figure 6-29(a)-6-29(c) are high confidence misclassified examples; Figure 6-29(d)-6-29(e) are novel class extrapolation examples. In Figure 6-29(e), the saliency map primarily highlights two objects: the red cone and the blue cylinder. Removing either of these objects does not result in a change of prediction. Instead, the misclassification of 1 sphere is due to the marginally-highlighted gray cone.

Chapter 7

Robot Controller Transparency

7.1 Introduction

The previous chapter introduces the transparency-by-example framework as a way for model inspection and understanding complementary to procedures based on the test set. In this chapter, we extend this to study robot controllers, which necessitates, among other things, a suite of new behavior definitions. Before going into the technical content, however, we first argue for the importance of comprehensive and wholistic testing of robot controllers, with a real world tragedy.

In 2018, after a confluence of failures, an autonomous vehicle (AV) struck and killed a pedestrian for the first time. In the run-up to this fateful event, the responsible company had reportedly been trying to improve the AV “ride experience" by emphasizing non-critical behaviors – such as the smoothness of the ride [22]. This event reflects the long-standing challenge in robotics: designing an appropriate objective which considers both safety-critical and non-critical behaviors.

When crafting an objective, it is virtually impossible to proactively account for all potential controller behaviors, and some priorities may even be in conflict with one another [125]. In practice, any given robot behaviors may be specified, unspecified, or even misspecified [20], so extensive testing and evaluation is a critical component of designing and assessing robot controllers – especially those using black-box models such as deep neural networks.

This chapter is based on the CoRL 2021 paper “RoCUS: Robot Controller Understanding via Sampling” by Yilun Zhou, Serena Booth, Nadia Figueroa and Julie Shah [180].

A common testing procedure focuses on finding extreme and edge cases of controller failure. For example, a tester might use this procedure to find that the AV swerves very badly when encountering a farm animal while traveling at 60mph. Finding such extreme and edge cases is well-studied within both traditional software testing paradigms [111] and more recent adversarial perturbation testing methods [51].

However, we argue that an equally, if not more, important form of testing should focus on *representative* scenarios, which considers the likelihood of encountering these scenarios. For example, if this AV is going to be deployed exclusively in New York City, the above example is largely unhelpful: cars rarely travel at 60mph in the city, and are very unlikely to encounter farm animals. Instead, the tester may prefer to know that the car swerves – though not as substantively – at lower speeds when a pedestrian steps toward it. Finding representative scenarios is often overlooked, but is especially useful for robotics.

Explicit mathematical analysis of robot controllers is implausible given the high dimensionality of the configuration space and the potential black-box representation of a learned controller. With access to a scenario simulator, though, a straightforward testing approach is to roll out the robotic controller on a distribution of scenarios that we are interested in (e.g., road conditions under different weather and congestion, with or without farm animals or pedestrians, etc.), and analyze those rollouts that exhibit a specified behavior – like excessive swerving. However, with too few scenarios, we risk missing the condition(s) that triggers the target behavior most saliently. With too many scenarios, all the most salient rollouts would be close to the global maximum at the expense of diversity and coverage. For example, if a farm animal causes the most swerving, followed by a pedestrian and a dangling tree branch, using too few scenarios may only find the pedestrian and the tree branch while using too many would result in an exclusive focus on the farm animal. Neither case helps the human develop a correct mental model of the AV's behavior.

The goal of finding scenarios that elicit certain robot behaviors while representative with respect to a distribution should remind the readers of the Bayes-TrEx formulation. Indeed, we extend it and introduce Robot Controller Understanding via Sampling (RoCUS), a method to enable systematic robot behavior inspection. RoCUS finds scenarios that are both inherently likely and elicit specified behaviors by formulating the problem as one of Bayesian posterior inference. Analyzing these scenarios and the resulting trajectories can help developers better understand the robot behaviors, and allow them to iterate on algorithm development if undesirable ones are revealed.

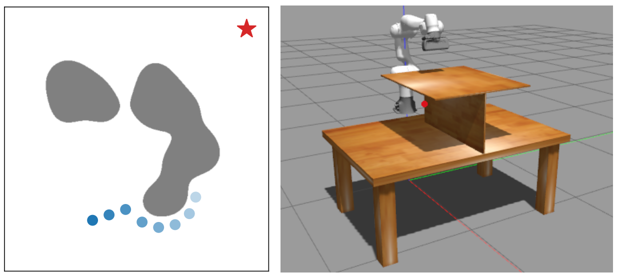


Figure 7-1: Two use case demos of RoCUS: 2D navigation (left) and 7DoF arm reaching (right).

We use RoCUS to analyze three controllers on two common robotics tasks (Figure 7-1). For a 2D navigation problem, we consider imitation learning (IL) [7], dynamical system (DS) [68], and rapidly-exploring random tree (RRT) [91]. For a 7DoF arm reaching problem, we consider reinforcement learning (RL) [147], as well as the same DS and RRT controllers. For each problem and controller, we specify several behaviors and visualize representative scenarios and trajectories that elicit those behaviors. Through this analysis, we uncover insights that would be hard to derive analytically and thus complement our mathematical understanding of the controllers. Moreover, we include a case study on how to improve a controller based on new insights from RoCUS. As such, RoCUS is a step towards the broader goal of building more accurate human mental models and enabling holistic evaluation of robot behaviors.

7.2 Related Work

Our work lies at the intersection of efforts to understand complex model behaviors and those to benchmark robot performance. Methods to understand, interpret, and explain model behaviors are now commonplace in the machine learning community. Mitchell et al. [106] introduced Model Cards, a model analysis mechanism which breaks down model performance for data subsets. In natural language processing, Ribeiro et al. [130] introduced a checklist for holistic evaluation of model capabilities and test case generation. In robotics, Fan et al. [43] introduced a verification framework for assessing machine behavior by sampling parameter spaces to find temporal logic-satisfying behaviors. Other efforts aim to summarize robot policies, trading off factors like brevity, diversity and completeness [61, 86]. All of these works have a shared underlying theme: treating the black box as immutable and performing downstream analyses of machine behavior [124]. RoCUS builds upon Bayes-TrEx and searches for instances which exhibit target behaviors to inform accurate human mental models.

$$\begin{tikzcd} t \arrow[r] \arrow[rr, bend left] & \tau \arrow[r] & b \arrow[r] & \hat{b} \end{tikzcd}$$

Figure 7-2: The graphical model for the inference problem of finding scenarios t and trajectories τ which exhibit specific behaviors b. The dashed box indicates the relaxed formulation (Eq. 7.2).

While the need for benchmarking robot performance is often expressed [76, 103, 110], these efforts usually operate on distributions of trajectories or randomly selected trajectories, and the accompanying metrics are typically task-completion based without consideration of implicit performance factors. Anderson et al. [6] put forth a recommendation of using *success weighted by path length* for navigation tasks – a task-completion metric. Cohen et al. [30] and Moll et al. [107] introduced suites of metrics for comparing motion planning approaches, and Lagriffoul et al. [87] presented a set of task and motion planning scenarios and metrics. Again, all of these proposed metrics are based solely on task completion. Lemme et al. [95] proposed a set of performance measures for reaching tasks, which are either task-completion based or require a costly human motion ground truth. Our contribution is distinct in two ways. First, we propose to sample specific trajectories which communicate controller behaviors instead of reporting metrics averaged over distributions of trajectories. Second, we introduce metrics which draw on these prior works while also including essential alternative and typically emergent quality factors, like motion jerkiness and legibility [41].

7.3 Methodology

At a high level, RoCUS helps users understand robotic controllers via representative scenarios that exhibit various specified behaviors. It solves this by directly incorporating the distribution of scenarios into a Bayesian inference framework as shown in Figure 7-2. A robotic problem is represented by a distribution $\pi(t)$ of individual scenarios t. For example, a navigation problem may have $\pi(t)$ representing the distribution over target locations and obstacle configurations. Given a specific scenario t, the controller under study induces a distribution $p(\tau|t)$ of possible trajectories τ. If both the controller and the transition dynamics are deterministic, $p(\tau|t)$ reduces to a δ-function at the induced trajectory τ. Stochasticity in either the controller (e.g., RRT) or the dynamics (e.g., uncertain outcome from an action) can result in τ being random. Finally, a behavior function $b(\tau, t)$ computes the behavior value of the trajectory – for example, the motion jerkiness. Some behaviors only depend on the trajectory and not the scenario, but we use $b(\tau, t)$ for consistency. Section 7.4 presents a list of behaviors.

The discussion on behavior in Section 7.1 is informal and implicitly combines two related but different concepts. The first concept is the behavior function $b(\tau, t)$ discussed above. The second is the specified target: for the swerving example, we are particularly interested in *maximal* behavior values. Thus, the target value can be thought of as $+\infty$. This inference problem uses the *maximal* mode of RoCUS. In other cases, we are also interested in scenarios and trajectories whose behaviors *matches* a target. For example, we want to find road conditions that lead to a daily commute time of an hour, where the behavior is the travel time. This inference problem uses the *matching* mode. Since matching mode is conceptually simpler, we present it first, followed by maximal mode. The sampling procedure is the same for both modes and presented last in Algorithm 2.

7.3.1 Matching Mode

The exact objective is to find scenarios and trajectories that exhibit user-specified behaviors b^*:

$$t, \tau \sim p(t, \tau | b = b^*) \propto p(b = b^* | t, \tau)\pi(\tau|t)\pi(t). \tag{7.1}$$

In most cases this posterior does not admit direct sampling, and an envelope distribution is not available for rejection sampling. Markov-Chain Monte-Carlo (MCMC) sampling does not work either: since the posterior is only non-zero on a very small or even measure-zero set, a Metropolis-Hastings (MH) sampler [58] can get stuck in the zero-density region. Similar to the Bayes-TrEx formulation, we relax it using a normal distribution formulation as shown in Figure 7-2:

$$\widehat{b}|b \sim \mathcal{N}(b, \sigma^2) \quad t, \tau \sim p(t, \tau|\widehat{b} = b^*) \propto p(\widehat{b} = b^*|t, \tau)p(\tau|t)\pi(t). \tag{7.2}$$

This relaxed posterior is non-zero everywhere $\pi(t)$ is non-zero and provides useful guidance to an MH sampler. While σ is a hyper-parameter in Bayes-TrEx, we instead choose σ such that

$$\int_{b^*-\sqrt{3}\sigma}^{b^*+\sqrt{3}\sigma} p(b)\,\mathrm{d}b = \alpha, \text{ with } p(b) = \int_t \int_\tau p(\tau|t)\pi(t)\mathbb{1}_{b(\tau,t)=b}\,\mathrm{d}\tau\,\mathrm{d}t \tag{7.3}$$

being the marginal distribution of $b(\tau, t)$, which can be estimated by trajectory roll-outs.

This formulation has two desirable properties. First, it is scale-invariant with respect to $b(\tau, t)$, e.g., measured under different units like meters vs. centimeters. Consider the same behavior under two different units b_1 and b_2 with $b_1 = c \cdot b_2$. For example, b_1 can be the trajectory length in centimeters and b_2 is the same quantity but in meters, and $c = 100$. Thus, $p(c \cdot b_1) = p(b_2)$ and $b_1^* = c \cdot b_2^*$. To maintain the same α level in Eq. 7.3, we need to have $\sigma_1 = c \cdot \sigma_2$. This implies that

$$p(t, \tau|\hat{b}_1 = b_1^*) = \frac{\mathcal{N}(b_1^*; b(\tau, t), \sigma_1^2)p(\tau|t)\pi(t)}{p(\hat{b}_1 = b_1^*)} \tag{7.4}$$

$$= \frac{\mathcal{N}(b_2^*; b(\tau, t), \sigma_2^2)p(\tau|e)\pi(t)}{p(\hat{b}_2 = b_2^*)} = p(t|\hat{b}_2 = b_2^*) \tag{7.5}$$

because $\mathcal{N}(b_1^*; b(\tau, t), \sigma_1^2) = \mathcal{N}(b_2^*; b(\tau, t), \sigma_2^2)$ due to the same scaling of $b_1 \sim b_2$ and $\sigma_1 \sim \sigma_2$, and $p(\hat{b}_1 = b_1^*) = p(\hat{b}_2 = b_2^*)$ as they are the same event. We conclude that the posterior distribution is scale-invariant with respect to $b(\tau, t)$.

Second, the hyper-parameter $\alpha \in [0, 1]$ has the intuitive interpretation of the approximate "volume" of posterior samples $t, \tau \mid \widehat{b} = b^*$ under the marginal $p(t, \tau) = p(\tau|t)\pi(t)$, a notion of their representativeness. Consider a uniform approximation to $\mathcal{N}(b^*, \sigma^2)$. To match the mean b^* and standard deviation σ, $\mathcal{U}(b^* - \sqrt{3}\sigma, b^* + \sqrt{3}\sigma)$ is needed. If we use this uniform distribution in Eq. 7.2 in lieu of the normal distribution, the posterior can be instantiated by sampling from the prior and rejecting scenarios for which the trajectory behavior $b(\tau, t)$ falls outside of this bound. Thus, Eq. 7.3 specifies that the "volume" of $(\alpha \cdot 100)\%$ under $p(t, \tau)$ is maintained.

7.3.2 Maximal Mode

In this mode, RoCUS finds trajectories that lead to maximal behavior values: $b^* \rightarrow \pm\infty$. It can also be used for finding minimal behavior values by negating the behavior. The posterior formulation is:

$$b_0 = \frac{b - \mathbb{E}[b]}{\sqrt{\mathbb{V}[b]}}, \quad \beta = \frac{1}{1 + e^{-b_0}}, \quad \widehat{\beta} \sim \mathcal{N}\left(\beta, \sigma^2\right), \quad t, \tau \sim p(t, \tau|\widehat{\beta} = 1), \tag{7.6}$$

where $\mathbb{E}[b]$ and $\mathbb{V}[b]$ are the mean and variance of the marginal $p(b)$. σ is chosen such that

$$\int_{1-\sqrt{3}\sigma}^{1} p(\beta)\, \mathrm{d}\beta = \alpha, \tag{7.7}$$

where $p(\beta)$ is the marginal distribution similar to Eq. 7.3. If $p(b)$ is normal, $p(\beta)$ is logit-normal. This formulation is again scale-invariant and has the same "volume" interpretation for α. The former stems from the standardization on b performed in Eq. 7.6. The latter uses the same uniform approximation but the bound is one-sided since $\beta \in (0, 1)$ by nature of the sigmoid transformation.

7.3.3 Posterior Sampling

The posterior sampling mechanism depends on the stochasticity of the controller and dynamics, and we discuss three cases.

Deterministic Controller & Dynamics: When both the controller and the dynamics are deterministic, so is $\tau|t$, denoted as $\tau(t)$. Eq. 7.2 reduces to $t \sim p(t|\widehat{b} = b^*) \propto p(\widehat{b} = b^*|t, \tau(t))\pi(t)$, and similarly for Eq. 7.6. Algorithm 2 presents the MH sampling procedure. First, σ is computed from α (Line 2). Then we start with an initial scenario t (Line 3). For each of the N iterations, we propose a new scenario t_{new} according to a transition kernel and compute the forward and reverse transition probabilities $p_{\text{for}}, p_{\text{rev}}$ (Line 5). We evaluate the posteriors under t and t_{new} (Line 6 and 7) and calculate the acceptance probability using the MH detailed balance principle (Line 8). Finally, we accept or reject accordingly (Line 9 – 11). Note that if the proposal is rejected, the current t is left unchanged *and appended to the samples.* We

Algorithm 2: MH Sampling Procedure

```
   Input: "Posterior volume" $\alpha$, number of samples $N$, optional burn-in $N_B$
          and thinning period $N_T$.
1  samples ← [ ];
2  Get $\sigma$ from $\alpha$ by Eq. 7.3 (matching) or 7.7 (maximal);
3  Randomly initialize $t$;
4  for $i = 1, ..., N$ do
5      $t_{\text{new}}, p_{\text{for}}, p_{\text{rev}}$ = propose($t$)
6      Get $p$ from $t$ by Eq. 7.2 (match) or Eq. 7.6 (max)
7      Get $p_{\text{new}}$ from $t_{\text{new}}$ by Eq. 7.2 or Eq. 7.6;
8      $a \leftarrow (p_{\text{new}} \cdot p_{\text{rev}})/(p \cdot p_{\text{for}})$;
9      Sample $u \sim \mathcal{U}[0, 1]$;
10     if $u < a$ then
11         $t \leftarrow t_{\text{new}}$;
12     Append $t$ to samples;
13 Optionally, discard the first $N_B$ burn-in samples and thin the samples by
     only keeping every $N_T$ samples;
14 return samples
```

$$\begin{tikzcd} t \arrow[r] \arrow[rr, bend left] & \tau \arrow[r] & b \arrow[r] & \hat{b} \\ u \arrow[ur] & & & \end{tikzcd}$$

Figure 7-3: The same graphical model as in Figure 7-2, but with the addition of stochasticity u in the controller such that $\tau|t, u$ is now deterministic.

can discard the first N_B samples as burn-in, and/or thin the samples by a factor of N_T to reduce auto-correlation.

Stochastic Controller: When the controller and $p(\tau|t)$ are stochastic, the controller can usually be implemented by sampling a random variable u (independent from t), and then producing the action based on the realization of u, as shown in Figure 7-3. For instance, a Normal stochastic policy $\pi(s) \sim \mathcal{N}(\mu(s), \sigma(s)^2)$ can be implemented by first sampling $u \sim \mathcal{N}(0, 1)$ and then computing $\pi(s) = \mu(s) + u \cdot \sigma(s)$. In this case, we sample in the combined (t, τ)-space, with Eq. 7.2 being $p(t, \tau|\widehat{b} = b^*) \propto p(\widehat{b} = b^*|t, \tau(e, u))p(u)\pi(t)$, where we overload $\tau(t, u)$ to refer to the *deterministic* trajectory given the scenario t and controller randomness u. It is crucial that for any u, we can evaluate $p(u)$. Concretely, modifying Algorithm 2, u_{new} is proposed alongside with t_{new} (Line 5), the detailed balancing factor (Line 8) is multiplied by $p_{u,\text{rev}}/p_{u,\text{for}}$, and $t_{\text{new}}, u_{\text{new}}$ are accepted or rejected together (Line 10 – 12).

Stochastic Dynamics: Using the same logic as the case of stochastic controller, RoCUS can also accommodate stochasticity in transition dynamics (e.g., object position uncertainty after it is pushed), *as long as such stochasticity can be captured in a random variable v and $p(v|t)$ can be evaluated.* This is typically possible in simulation, and the modification to Algorithm 2 is similar to the case of stochastic controllers. We leave real world execution to future work, but at a high level, we can

1. treat a sampled trajectory as the deterministic one;
2. restart multiple times to estimate $\mathbb{E}_\tau[b(\tau, t)]$; or
3. use likelihood-free MCMC methods [23].

7.3.4 The Bayesian Posterior Sampling Interpretation

RoCUS uses Bayesian sampling concepts of prior, likelihood, and posterior quite liberally. Specifically, the scenario distribution is defined as the prior, and thus the notion of a scenario being likely in the deployment context refers to high probability under the prior. Likelihood refers to the behavior saliency: how much the exhibited behavior matches the behavior specification. The act of posterior sampling then finds tasks that strike a balance between these two objectives.

The choice of explicitly modeling the scenario distribution is intentional, as it is not unlikely that the deployment environment will be different than the development environment. Such a domain mismatch may cause catastrophic failures, especially for learned controllers whose extrapolation behaviors are typically undefined. With a suitable scenario distribution, RoCUS allows more failures to surface during this testing procedure.

7.4 Behavior Taxonomy

Robot behaviors broadly belong to one of two classes: intentional and emergent. *Intentional* behaviors are those that the controller explicitly optimize with objective functions. For example, the controller for a reaching task likely optimizes to move the end-effector to the target, by setting the target as an attractor in DS, using a target-reaching objective configuration in RRT, or rewarding proximity in RL. Thus, the final distance between the end-effector and the target is an intentional behavior for all three controllers. By contrast, *emergent* behaviors are not explicitly specified in the objective. For the same reaching problem, an RL policy with reward based solely on distance may exhibit smooth trajectories for some target locations and jerky ones for others. Such behaviors may emerge due to robot kinematic structure, training stochasticity, or model inductive bias.

For trajectory τ, many behavior metrics $b(\tau, t)$ can be expressed as a line integral $\int_\tau V(\mathbf{x})\mathrm{d}s$ of a scalar field $V(\mathbf{x})$ along τ or its length-normalized version $\frac{1}{||\tau||}\int_\tau V(\mathbf{x})\mathrm{d}s$, where $\mathrm{d}s$ is the infinitesimal segment on τ at $\mathbf{x}$ and $||\tau||$ is the trajectory length. $\mathbf{x}$ and τ can be in either joint space or task space. We introduce six behaviors: length, time derivatives (velocity, acceleration and jerk), straight-line deviation, obstacle clearance, near-obstacle velocity and motion legibility, summarized in Table 7.1.

Trajectory length simply measures how long the trajectory is. In most of the behaviors below, the normalizing factor is also length to decorrelate the behavior value from it.

Average velocity, acceleration and jerk are useful for general understanding about how fast and abruptly the robot moves, an important factor to its safety.

Straight-line deviation measures how much the robot trajectory deviates from the straight-line path, in either the task space or the state space. A specific scenario

Name	Definition	Name	Definition
Trajectory Length	$b = \int_\tau 1\, \mathrm{d}s$	Straight-Line Deviation	$b = \frac{1}{\lvert\lvert\tau\rvert\rvert}\int_\tau \lvert\lvert\mathbf{x} - \mathrm{proj}_{\mathbf{x}_f - \mathbf{x}_i}\mathbf{x}\rvert\rvert\, \mathrm{d}s$
Average Velocity	$b = \frac{1}{\lvert\lvert\tau\rvert\rvert}\int_\tau \lvert\lvert\dot{\mathbf{x}}\rvert\rvert\, \mathrm{d}s$	Obstacle Clearance	$b = \frac{1}{\lvert\lvert\tau\rvert\rvert}\int_\tau \min_{\mathbf{x}_o \in \mathcal{O}} \lvert\lvert\mathbf{x} - \mathbf{x}_o\rvert\rvert\, \mathrm{d}s$
Average Acceleration	$b = \frac{1}{\lvert\lvert\tau\rvert\rvert}\int_\tau \lvert\lvert\ddot{\mathbf{x}}\rvert\rvert\, \mathrm{d}s$	Near-Obstacle Velocity	$b = \frac{\int_\tau \lvert\lvert\dot{\mathbf{x}}\rvert\rvert / \min_{\mathbf{x}_o \in \mathcal{O}} \lvert\lvert\mathbf{x} - \mathbf{x}_o\rvert\rvert\, \mathrm{d}s}{\int_\tau 1/\min_{\mathbf{x}_o \in \mathcal{O}} \lvert\lvert\mathbf{x} - \mathbf{x}_o\rvert\rvert\, \mathrm{d}s}$
Average Jerk	$b = \frac{1}{\lvert\lvert\tau\rvert\rvert}\int_\tau \lvert\lvert\dddot{\mathbf{x}}\rvert\rvert\, \mathrm{d}s$	Motion Legibility	$b = \frac{1}{\lvert\lvert\tau\rvert\rvert}\int_\tau p(g\mid\mathbf{x})\, \mathrm{d}s$

Table 7.1: Behavior definitions.

instance in which the straight-line path is feasible (e.g., with no obstacles) is typically considered easy. Thus, we can find scenarios of varying difficulty level on the spectrum of deviation values. In the definition, $\mathbf{x}_i$ is the initial state, $\mathbf{x}_f$ is the final state, and proj is the projection operator.

Obstacle clearance measures the average distance to the closest obstacle. Finding situations in which the robot moves very close to obstacles is crucial to understanding the collision risk level. In the definition, $\mathcal{O}$ represents the obstacle space.

Near-obstacle velocity calculates how fast the robot moves around obstacles. We define it as the average velocity on the trajectory weighted by the inverse distance to the closest obstacle. Other weighting method can be used, as long as it is non-negative and monotonically decreasing with distance. This behavior is correlated with the damage of a potential collision, as high-speed collisions are usually far more dangerous and costly. Since we want the value to represent the average velocity, we normalize by the integral of weights along the trajectory.

Motion legibility measures how clear the trajectory informs the goal. A generic definition is $p(g|\mathbf{x})$, or the conditional probability of the goal g given at the current robot state $\mathbf{x}$, but better, application-specific ones may exist.

7.5 Experiments

In this section, we demonstrate how RoCUS can find "hidden" properties of various controllers for two common tasks, navigation and reaching. We also uncover a suboptimal controller design due to bad hyper-parameter choices, which is improved based on RoCUS insights.

Each sampling run collected 10,000 samples, with the first 5,000 discarded as burn-in. On a consumer-grade computer with a single GeForce GTX 1080 GPU card (for neural network-based controllers), the sampling generally takes around 1 to 3 hours. Since RoCUS is designed to be an offline analysis tool as opposed to be used for real-time sample generation, several hours of runtime would be acceptable in most cases. Furthermore, MCMC sampling is embarrassingly parallel by simply using multiple chains concurrently, with the only overhead cost being the discarded burn-in samples.

7.5.1 Controller Algorithms

We consider four classes of robot controllers. The **imitation learning** (IL) controller uses expert demonstrations to learn a neural network policy which maps observations to deterministic actions. The **reinforcement learning** (RL) controller implements proximal policy gradient (PPO) [137]. While a mean and a variance is used to parameterize a PPO policy during training, the policy deterministically outputs the mean action during evaluation.

The **dynamical system** (DS) controller modulates the linear controller $\mathbf{u}(\mathbf{x}) = \mathbf{x}^* - \mathbf{x}$, for the task-space target $\mathbf{x}^*$, into $\mathbf{u}_M(\mathbf{x}) = M \cdot \mathbf{u}(\mathbf{x})$ using the modulation matrix M derived from obstacle configuration, as proposed by Huber et al. [68]. We give a self-contained review in Appendix 7.7.1.

The **rapidly-exploring random tree** (RRT) controller finds a configuration-space trajectory via RRT and then controls the robot through descretized segments. There are many RRT variants with subtle differences. For clarity, Algorithm 3 presents the version that we use.

Algorithm 3: RRT Algorithm

```
  Input: Start configuration $s_0$, target configuration $s^*$.
1 $\mathcal{T} \leftarrow$ tree(root = $s_0$);
2 success $\leftarrow$ attempt-grow($\mathcal{T}$, from = $s_0$, to = $s^*$);
3 while not success do
4     $s \leftarrow$ sample-configuration( );
5     $s_n \leftarrow$ nearest-neighbor($\mathcal{T}$, $s$);
6     success $\leftarrow$ attempt-grow($\mathcal{T}$, from = $s_n$, to = $s$);
7     if success then
8         success $\leftarrow$ attempt-grow($\mathcal{T}$, from = $s$, to = $s^*$);
9 return path($\mathcal{T}$, from = $s_0$, to = $s^*$)
```

Notably, RRT is stochastic (cf. Figure 7-3), but the entire randomness is captured by the sequence of C-space samples used to grow the tree, including failed ones. We call this a *growth* $g = [s_1, s_2, s_3, ...]$. The probabilistic completeness property of RRT generally assures that the algorithm will terminate in finite time with probability 1 if a path to the target exists [91]. Thus, hypothetically, given an infinitely long tape containing every entry of g, we can compute a deterministic trajectory $\tau = \text{RRT}(s_0, s^*, g)$ with a finite number of nodes with probability 1.

To enable MH inference, we take inspiration from Bayesian nonparametrics: we instantiate g on an *as-needed* basis. We start with an empty vector of $g = [\,]$. When calculating $\text{RRT}(s_0, s^*, g)$, if a new point beyond existing entries of g needs to be sampled, we append it to g. During MH inference, we use a transition kernel that operates element-wise on instantiated entries of g (i.e., independently perturbing each entry of g). If the transition kernel does not depend on the current g (e.g., drawing uniformly from the C-space), past instantiated entries do not even need to be kept.

Note that RRT trajectories are often smoothed *post hoc*. Since our main focus is to evaluate and identify problems for an existing one, we use the original formulation. Moreover, it is easy to use RoCUS to evaluate model updates (e.g., original vs smoothed RRT) as discussed in Section 7.6.

For MCMC sampling, we used a truncated Gaussian transition kernel for all experiments. For the RBF-defined 2D environment, we initialize 15 obstacle points with coordinates sampled uniformly in $[-0.7, 0.7]$. The transition kernel operates independently on each obstacle coordinate: given the current value of x, the kernel samples a proposal from $\mathcal{N}(\mu = x, \sigma^2 = 0.1^2)$ truncated to $[-0.7, 0.7]$ (and also appropriately scaled). For the arm reaching task, the target is sampled uniformly from two disjoint boxes, with the left box at $[-0.5, -0.05] \times [-0.3, 0.2] \times [0.65, 1.0]$ and the right box at $[0.05, 0.5] \times [-0.3, 0.2] \times [0.65, 1.0]$. Again, we use the same transition kernel with $\sigma_x = 0.1, \sigma_y = 0.03, \sigma_z = 0.035$ in three directions. Again, the distribution is truncated to the valid target region ($x \in [-0.5, -0.05] \cup [0.05, 0.5], y \in [-0.3, 0.2], z \in [0.65, 1.0]$). Hence, the transition kernel implicitly allows for the jump across two box regions.

In addition, the stochastic RRT controller also requires a transition kernel. A new segment to the growth (i.e., configuration) is uniformly sampled between the lower- and upper-limit (i.e., $[x_L, x_U]$). For each configuration, the same Gaussian kernel truncated to $[x_L, x_U]$, and $\sigma = 0.1(x_U - x_L)$ is used.

7.5.2 2D Navigation Task Experiments

Setup In a rectangular arena with irregularly shaped obstacles, a point mass robot needs to move from the lower left to the upper right corner (Figure 7-1 left). A random sample of environments is visualized in Figure 7-4. Appendix 7.7.2 details the obstacle generation and robot simulation procedures.

We consider three controllers for this environment: an RRT planner, a deep learning IL policy, and a DS (Figure 7-5). The RRT planner implements Algorithm 3 and discretizes the path to small segments as control signals at each time step. The IL controller uses smoothed RRT trajectories as expert demonstrations, and learns to predict heading angle from its current position and lidar readings. The DS controller finds an interior reference point for each obstacle, and converts each obstacle in the environment to be star-shaped. Γ-functions are then defined for these obstacles and used to compute the modulation matrix M. Appendix 7.7.3 contains additional implementation details.

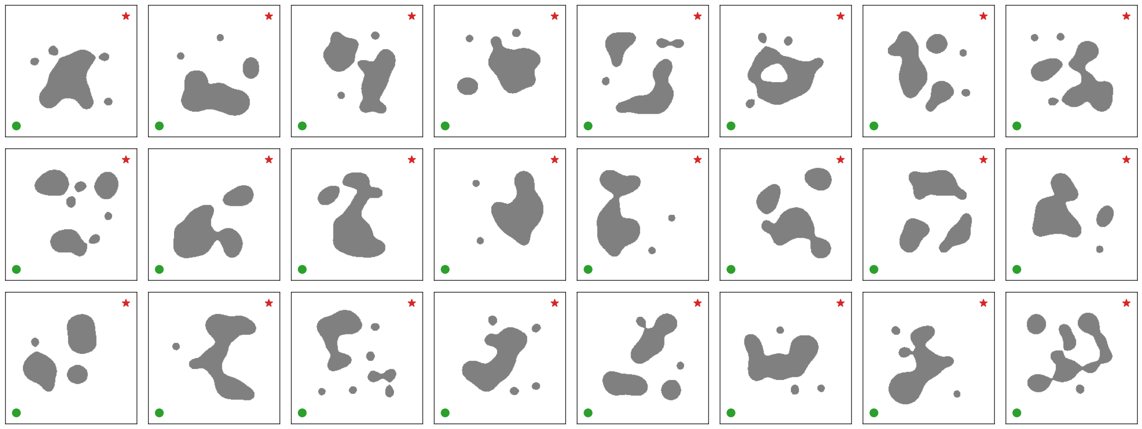


Figure 7-4: An assortment of randomly generated RBF 2D environments, providing a sense of the diversity generated with this formulation. The green dots are the environment starting points and the red stars are navigation targets.

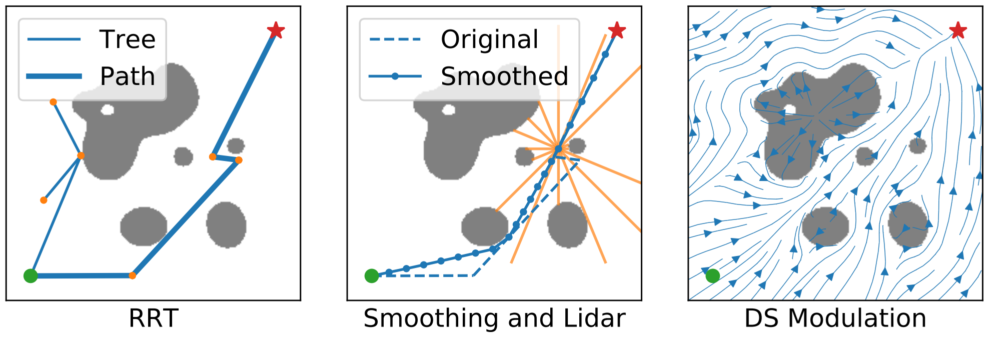


Figure 7-5: RRT, IL and DS controllers on 2D navigation domain. Left: the RRT controller tree. Middle: smoothed RRT trajectory and lidar sensor (orange lines) for IL controller training. Right: the modulation by the DS controller.

Straight-Line Deviation In most cases, the robot cannot navigate directly to the target in a straight line. Thus, the collision-avoidance behavior is a crucial aspect for navigation robots. To understand it, we sample obstacles that lead to trajectories minimally deviating from the straight line path. Since the deviation is always non-negative, we use the matching mode in Eq. 7.2 with target $b^* = 0$.

In Figure 7-6, the top row plots posterior trajectories in orange, with prior trajectories in blue. The bottom row plots the obstacle distributions compared to the prior, with red regions being more likely to be occupied by obstacles and blue ones less likely to be obstructed.

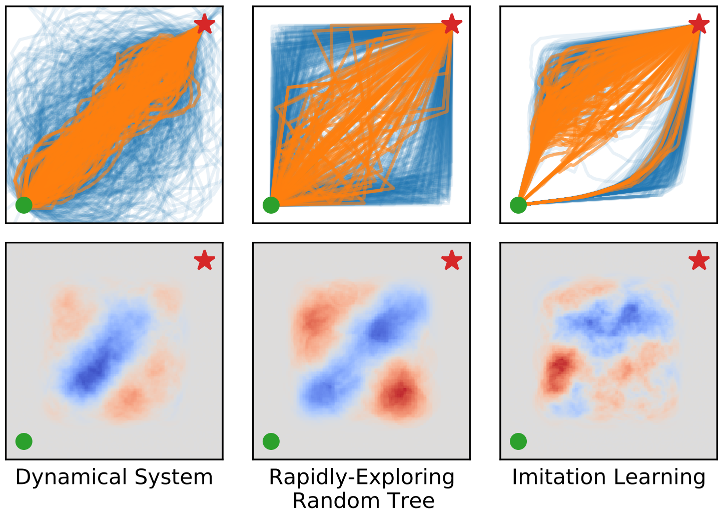


Figure 7-6: Top: posterior trajectories in orange vs. prior in blue for minimal straight-line deviation behavior for three controllers. Bottom: posterior obstacle distribution relative to the prior. Higher obstacle density regions are painted in red and lower ones in blue.

For DS and RRT, the posterior trajectories and obstacle configurations are mostly symmetric with respect to the straight-line connection, as expected since both methods are formulated symmetrically with respect to the x- and y-coordinates. The obstacle distribution under RRT is also expected, since it seeks straight-line connections whenever possible and thus favor a "diagonal corridor" with obstacles on either side. For DS, however, obstacles are slightly *more* likely to exist at the two ends of the above-mentioned corridor. This behavior is an artifact of the DS *tail effect*, which drags the robot around the obstacle (details in Appendix 7.7.1). By taking advantage of anchor-like obstacles at the ends of the corridor, the modulation can reliably minimize the straight-line deviation.

By comparison, the IL controller saliently exhibits trajectory asymmetry: it mostly takes paths on the left. It is possible that the asymmetry is due to "unlucky" samples by the MH sampler, but many independent restarts all confirm its presence, indicating that the asymmetry is inherent in the learned model. Since the neural network architecture is symmetric, we conclude that the stochasticity in the dataset generation and training procedure (e.g., initialization) leads to such imbalanced behaviors. Furthermore, the obstacle map suggests that obstacles are distributed very close to the robot path. Why does the robot seem to drive into obstacles? The answer lies in dataset generation: the smoothing procedure (Figure 7-5 middle) results in most demonstrated paths navigating tightly around obstacles, and it is thus expected that the learned IL controller displays the same behavior.

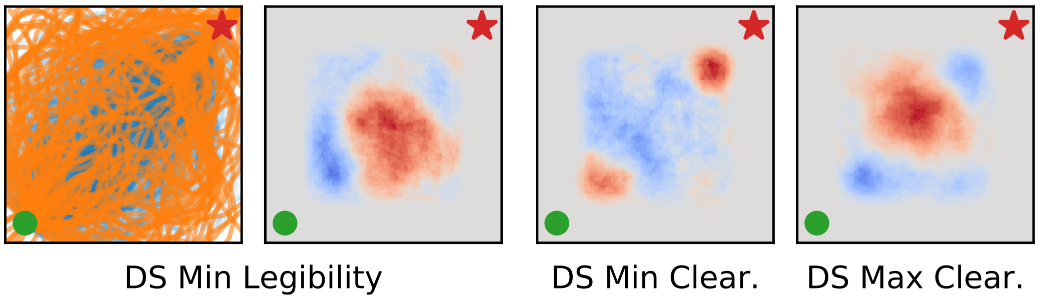


Figure 7-7: Left: trajectories and obstacle configurations from sampling minimal DS legibility. Right: obstacle configurations for minimizing and maximizing DS obstacle clearance. These examples show how obstacle positions affect the legibility and clearance behaviors.

Legibility We define the instantaneous legibility as the cosine similarity between the current robot direction and the direction to target $\mathbf{x}^*$, $V(\mathbf{x}) = \dot{\mathbf{x}} \cdot (\mathbf{x}^* - \mathbf{x})/(||\dot{\mathbf{x}}|| \cdot ||\mathbf{x}^* - \mathbf{x}||)$, with the intuition that a particular run may be confusing to users if the robot does not often align to the target. Though this quantity is bounded by $[-1, 1]$, a general legibility definition may not be. Thus, we use the maximal mode of RoCUS to find DS trajectories and obstacle configurations that achieve *minimal* legibility, by negating $V(\mathbf{x})$ first. The left two panels of Figure 7-7 present the samples. As expected, most trajectories take large detours due to obstacles in the center.

Obstacle Clearance We take $V(\mathbf{x}) = \min_{\mathbf{x}_o \in \mathcal{O}} ||\mathbf{x} - \mathbf{x}_o||$. For the DS, we sample two posteriors to maximize and minimize this behavior. As shown in the right two panels of Figure 7-7, when minimizing obstacle clearance, we see clusters of obstacles in close proximity to the starting and target positions, such that the robot is forced to navigate around them. When maximizing obstacle clearance, we instead see central clusters of obstacles, such that the robot can avoid them by bearing hard left or right.

Takeaways RoCUS reveals two unexpected phenomena. First, IL trajectories are highly asymmetric toward the left of the obstacle due to dataset and/or training imbalance. Second, both DS and IL models exhibit certain "obstacle-seeking" behaviors, the former due to the "tail-effect" and the latter due the dataset generation process. In both cases, such behavior may be undesirable in deployment due to possibly imprecise actuation, and the controller design may need to be modified.

7.5.3 7DoF Arm Reaching Task Experiments

Setup A 7DoF Franka Panda arm is mounted on the side of a table with a T-shaped divider (Figure 7-1 right). Starting from the same initial configuration on top of the table, it needs to reach a random location on either side under the divider. We simulate this task in PyBullet [32]. We consider three controllers: an RRT planner, a deep RL PPO agent, and a DS formulation.

RRT again implements Algorithm 3, but uses inverse kinematics (IK) to first find the joint configuration corresponding to the target location. The RL controller is a multi-layer perceptron (MLP) network trained using the PPO algorithm. The DS model outputs the end-effector trajectory in the task space, which is converted to joint space via IK, with SVM-learned obstacle definitions. Appendix 7.7.4 contains additional implementation details for each method. Overall, RRT and RL are quite successful in reaching the target while the DS is not due to the bulky robot structure, close proximity to the divider, and the task-space only modulation.

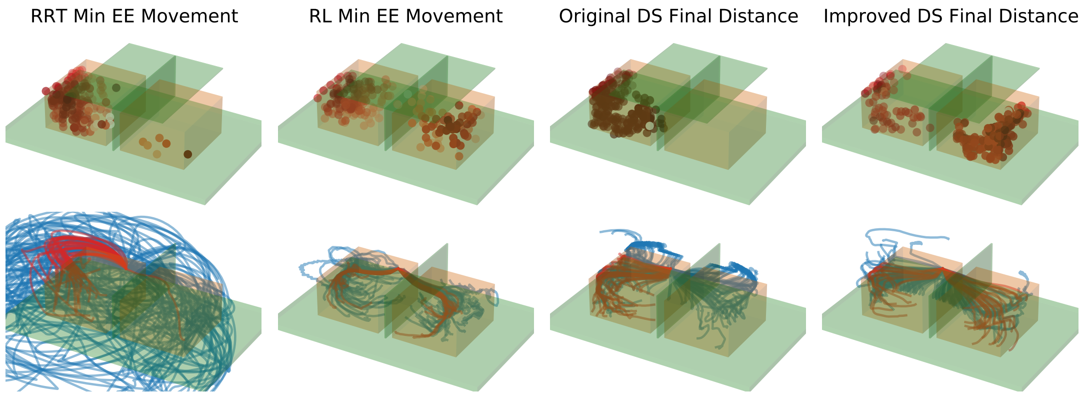


Figure 7-8: Left: Minimal end-effector movement samples for RRT and RL. Right: Posterior samples for minimal distance from end-effector to target for the original and improved DS controllers. Top: posterior targets locations, with tabletop + divider in green and target region in orange. Bottom: posterior trajectories in red, prior trajectories in blue. Robot is mounted on the near long edge.

End-Effector Movement We find configurations that minimize the total travel distance of the end-effector for RRT and RL (DS omitted due to high failure rate). Figure 7-8 (left two) shows the posterior target locations and trajectories. Notably, unlike RL, RRT trajectories are highly asymmetric, since there are straight-line connections in the configuration space from the initial pose to some target regions on the left, while every right-side goal requires at least an intermediate node.

Legibility We define legibility of reaching to the target on one side of the vertical divider as the average negative distance that the end effector moves in the other direction, $V(\mathbf{x}) = -\max(\tilde{\mathbf{x}}_1, 0)$, where $\tilde{\mathbf{x}}_1 = \mathbf{x}_1$ if target is on the left, or $\tilde{\mathbf{x}}_1 = -\mathbf{x}_1$ otherwise, and $\mathbf{x}_1$ is the x-coordinate of the robot end effector with right in the positive direction. We find target locations that are minimally legible and apply the maximal inference mode on the maximum distance measure.

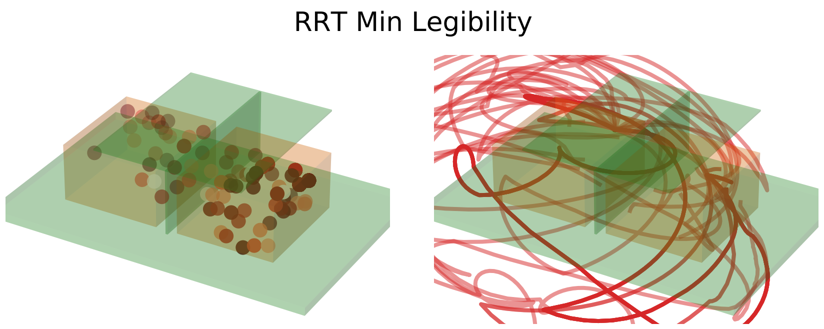


Figure 7-9: Posterior samples showing minimal legibility behavior for RRT.

We did not find any illegible motions from RL controllers for 2,000 targets, which is mostly expected since the RL reward is distance to the target. For RRT, however, since we do not use an optimal formulation [e.g. 60, 79] or perform post-hoc smoothing, the controller is expected to frequently exhibit low legibility. Figure 7-9 plots the posterior target locations and trajectories. The target locations leading to illegible motions are spread out mostly uniformly on the right, but concentrated in far-back area on the left, consistent with our findings on the asymmetry of configuration space. The trajectory plot confirms the illegibility.

DS Improvement Our initial DS implementation frequently fails to reach the target. This is understandable, as the DS convergence guarantee [68] is only valid in task space, in which the modulation is defined. When the full-arm motion is solved via IK, it is possible that some body parts may collide and get stuck because of the table divider. To understand the DS behaviors, we use RoCUS to sample target locations that result in minimal final distance from the end-effector to the target (i.e., most successful executions, Figure 7-8 center-right). Similar to the RRT case, the samples show strong lateral asymmetry, with all posterior target locations on the left, due to the same cause of asymmetric kinematic structure. The result points to a clear path to improve the DS controller such that it can succeed with right-side targets: increase the collision clearance of the divider so that the end-effector navigates farther away from the divider, thus also bringing the whole arm to be farther away. As detailed in Appendix 7.7.5, this modification greatly improves the controller performance as confirmed by the new symmetry in Figure 7-8 (rightmost). In addition, since the issue with DS controller mainly lies in obstacle avoidance in joint-space or on the body of the robot, additional techniques [82, 105, 126, 154] could be used and we leave them to future directions.

Takeaway The set of studies reveal an important implication of the robot's kinematic structure: the left side is much less "congested" with obstacles than the right side in the configuration space. While the RL controller is able to learn efficient policies for both sides, the design of certain controllers may need to explicitly consider such factors.

7.5.4 Quantitative Summary

We studied other additional behaviors on both tasks, and Table 7.2 summarizes prior vs. posterior mean behavior values and shows that RoCUS consistently finds samples salient in the target behavior.

Domain	Behavior	Target	Prior (DS)	Post. (DS)	Prior (IL/RL)	Post. (IL/RL)	Prior (RRT)	Post. (RRT)
2D Nav	Avg. Jerk	0	1.84e-3	1.46e-3	6.95e-4	3.19e-4	4.24e-4	2.79e-4
	Straight	0	0.256	0.084	0.378	0.301	0.470	0.162
	Legibility	min	0.819	0.650	0.877	0.784	0.798	0.669
	Obstacle	0	0.309	0.229	0.262	0.218	0.312	0.241
	Obstacle	max	0.309	0.611	0.262	0.387	0.312	0.442
Arm	Straight	0	0.980	0.913	0.858	0.762	1.223	0.897
	EE Dist	0	0.934	0.623	0.958	0.691	3.741	1.192

Table 7.2: Quantitative results on tasks for two domains.

7.5.5 MCMC Sampling Evaluation

After confirming that RoCUS can indeed uncover significant and actionable controller insights, we evaluate the sampling procedure itself, using behaviors described above as examples.

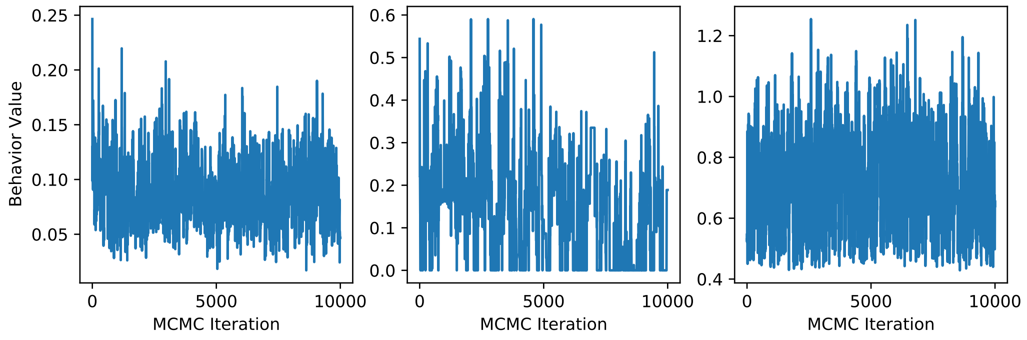


Figure 7-10: Sampled behavior values for three MCMC chains. From left to right, the three panels show DS min straight-line deviation on 2D navigation, RRT min straight-line deviation on 2D navigation and RL min end-effector movement on 7DoF arm reaching. The visualization confirms that 10,000 iterations with 5,000 burn-ins are more than sufficient to find representative samples.

Mixing Property A potential downside of MCMC sampler is the slow mixing time, which causes the chain to take a long time to converge from initialization and causes consecutive samples to be highly correlated. Does this phenomenon happen for our sampling? Figure 7-10 plots the behavior along the MCMC iterations for various sampling runs, showing that the chains mix well quite fast. Thus, a modest amount of samples, such as several thousand, is typically sufficient to model the target posterior distribution well.

Baseline: Top-k Selection To the best of our knowledge, RoCUS is the first work that applies the transparency-by-example (TrEx) formulation to robotic tasks, and we are not aware of existing methods for the same purpose. Notably, adversarial perturbation algorithms [51] are *not* feasible, since stepping in simulator (or real world) is not typically differentiable. Section 7.1 discusses a straightforward alternative that runs the controller on N different scenarios and pick the top-k with respect to the target behavior. We demonstrate its shortcomings on the minimal straight-line deviation behavior for the 2D navigation DS controller (RoCUS samples shown in Figure 7-6 left).

Figure 7-11 (left) shows the trajectories of different values of k for the same fixed N, and vice versa. While a bigger N/k ratio leads to more salient behaviors in the top-k samples, these examples become more concentrated around the global maximum and less diverse, making this approach especially myopic. Further, it is not easy to find the optimal N to trade off between diversity and saliency of the top-k samples. By contrast, RoCUS offers the intuitive α hyper-parameter. Figure 7-11 (middle) shows that a smaller N fails to highlight the "corridor" pattern while a larger N makes it completely open and misses the "tail-effect anchors" at the two ends.

In addition, the hard cut-off at the k-th salient behavior threshold has two undesirable implications: first, every trajectory more salient than the threshold is kept but is given equal importance; second, a trajectory even slightly under the threshold is strictly discarded. By comparison, RoCUS gives more importance to more salient samples in a progressive manner, as shown in Figure 7-11 right.

Finally, top-k selection is very computationally inefficient. It discards all of the unselected $N - k$ samples, while RoCUS is much more efficient in that all samples after the burn-in up to the thinning factor can be kept since the posterior concentrated on the salient behavior is directly sampled.

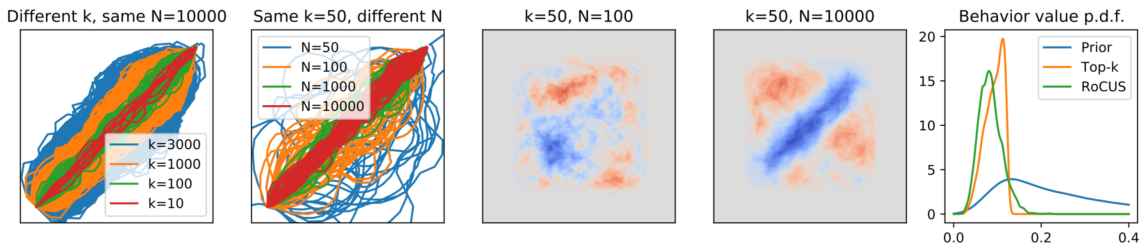


Figure 7-11: Top-k selection baseline. Left two: trajectory distribution; middle two: obstacle distribution; right one: probability density function of behavior values.

7.6 Discussion and Conclusion

RoCUS enables humans to build better mental models of robot controllers. Compared to existing evaluations on task-completion metrics for hand-designed scenarios, RoCUS generates scenarios and trajectories that highlight any given behavior in a principled way. We used it to uncover non-obvious insights in two domains and help with debugging and improving a controller.

While RoCUS is mainly a tool to analyze robot controllers in simulation as part of comprehensive testing before deployment, it can help understanding (anomalous) real world behaviors as well. When an anomaly is observed, RoCUS can find more samples with the anomaly for developers to identify patterns of systematic failures. Furthermore, RoCUS is not inherently limited to simulation: it only requires trajectory roll-out on specific scenarios. For the arm reaching task, this is easy in the real world. For autonomous driving, "recreating" a traffic condition that involves other vehicles may be hard. However, a key feature of RoCUS is the decoupling of the scenario and the controller algorithm, which allows testing on simpler scenario variants (e.g., with props instead of real cars).

There are multiple directions to extend and complement RoCUS for better usability and more comprehensive functionality. First, while we only used RoCUS on individual controllers, future work can readily extend it to *compare two controllers* by defining behavior functions that take in the scenario and two trajectories, one from each controller, and compute differential statistics. For example, this could be used to find road conditions that lead to increased swerving behavior of a new AV controller, compared to the existing one. Such testing is important to gain a better understanding of *model updates* [13], and is particularly necessary for ensuring that these updates do not unintentionally introduce new problems.

In addition, sometimes it is important to understand particular trajectories sampled by RoCUS. For example, which sensor input (e.g., lidar or camera) is most important to the current action (e.g., swerving)? Why does the controller take one action rather than another (e.g., swerving rather than braking)? Preliminary investigation into this explainable artificial intelligence (XAI) problem in the context of temporally extended decision making has been undertaken [53, 169], but various issues with existing approaches have been raised [11] and future research is needed to address them.

Finally, an important step before actual deployment is to design appropriate user interfaces to facilitate the two-way communication between RoCUS and end-users. In one direction, the user needs to specify the behavior of interest, and it would be desirable for it to involve as little programming as possible, especially for non-technical stakeholders. In the other direction, RoCUS needs to present the sample visualization, and potentially model explanations as described above, for users to inspect. Here, it is important for the information to be accurate but at the same time not overwhelming.

Overall, RoCUS is a framework for systematic discovery and inspection of robotic controller behaviors. We hope that the demonstrated utility of RoCUS sparks further efforts towards the development of other tools for more holistic understanding of robot controllers.

7.7 Appendix

7.7.1 Dynamical System Modulation

We review the DS formulation proposed by Huber et al. [68], and present our problem-specific adaptations for 2D Navigation in Appendix 7.7.3 and 7DoF arm reaching in Appendix 7.7.4. A reader familiar with DS motion controllers may skip this review.

Given a target $\mathbf{x}^*$ and the robot's current state $\mathbf{x}$, a linear controller $\mathbf{u}(x) = \mathbf{x}^* - \mathbf{x}$ will guarantee convergence of $\mathbf{x}$ to $\mathbf{x}^*$ if there are no obstacles. However, it can easily get stuck in the presence of obstacles. Huber et al. [68] proposes a method to calculate a modulation matrix $M(\mathbf{x})$ at every $\mathbf{x}$ such that if the new controller follows $\mathbf{u}_M(\mathbf{x}) = M(\mathbf{x}) \cdot \mathbf{u}(\mathbf{x})$, then $\mathbf{x}$ still converges to $\mathbf{x}^*$ but never gets stuck, as long as $\mathbf{x}^*$ is in free space. The objective of the DS modulation is to preserve the linear controller's convergence guarantee while ensuring that the robot is never in collision.

The modulation matrix $M(\mathbf{x})$ is computed from a list of obstacles, each of which is represented by a Γ-function. For the i-th obstacle $\mathcal{O}_i$, its associated gamma function Γ_i must satisfy the following properties:

- $\Gamma_i(\mathbf{x}) \leq 1 \iff \mathbf{x} \in \mathcal{O}_i$,
- $\Gamma_i(\mathbf{x}) = 1 \iff \mathbf{x} \in \partial\mathcal{O}_i$,
- $\exists\, \mathbf{r}_i,$ s.t. $\forall\, t_1 \geq t_2 \geq 0, \forall\, \mathbf{u}, \Gamma_i(\mathbf{r}_i + t_1\mathbf{u}) \geq \Gamma_i(\mathbf{r}_i + t_2\mathbf{u})$.

In words, the Γ-function value needs to be less than 1 when inside the obstacle, equal to 1 on the boundary, greater than 1 when outside. This function must also be monotonically increasing radially outward from a specific point $\mathbf{r}_i$. This point is dubbed the *reference point*. From this formulation, $\mathbf{r}_i \in \mathcal{O}_i$ and any ray from $\mathbf{r}_i$ intersects with the obstacle boundary $\partial\mathcal{O}_i$ exactly once. The latter property is also the definition that $\mathcal{O}_i$ is "star-shaped" (Figure 7-13). For most common (2D) geometric shape such as rectangles, circles, ellipses, regular polygons and regular stars, $\mathbf{r}_i$ can be chosen as the geometric center.

We first consider the case of a single obstacle $\mathcal{O}$, represented by Γ with reference point $\mathbf{r}$. Use d to denote the dimension of the space. We define

$$M(\mathbf{x}) = E(\mathbf{x})D(\mathbf{x})E^{-1}(\mathbf{x}). \tag{7.8}$$

We have

$$E(x) = [\mathbf{s}(\mathbf{x}), \mathbf{e}_1(\mathbf{x}), ..., \mathbf{e}_{d-1}(\mathbf{x})], \tag{7.9}$$

where

$$\mathbf{s}(\mathbf{x}) = \frac{\mathbf{x} - \mathbf{r}}{||\mathbf{x} - \mathbf{r}||} \tag{7.10}$$

is the unit vector in the direction of $\mathbf{x}$ from $\mathbf{r}$, and $\mathbf{e}_1(\mathbf{x}), ..., \mathbf{e}_{d-1}(\mathbf{x})$ form a $d-1$ orthonormal basis to the gradient of the Γ-function, $\nabla\Gamma(\mathbf{x})$ representing the normal to the obstacle surface. $D(\mathbf{x})$ is a diagonal matrix whose diagonal entries are $\lambda_s, \lambda_1, ..., \lambda_{d-1}$, with

$$\lambda_s = 1 - \frac{1}{\Gamma(\mathbf{x})}, \tag{7.11}$$

$$\lambda_1, ..., \lambda_{d-1} = 1 + \frac{1}{\Gamma(\mathbf{x})}. \tag{7.12}$$

each eigenvalue determines the scaling of each direction. Conceptually, as the robot approaches the obstacle, this modulation decreases the velocity for the component in the reference point direction (i.e., toward obstacles) while increases velocity for perpendicular components. The combined effect results in the robot being deflected away tangent to the obstacle surface.

With N obstacles, we compute the modulation matrix $M_i(\mathbf{x})$ for every obstacle using the procedure above and the individual controllers $\mathbf{u}_{M_i}(\mathbf{x}) = M_i(\mathbf{x}) \cdot \mathbf{u}(\mathbf{x})$. The final modulation is the aggregate of all the individual modulations. However, a simple average is insufficient since closer obstacles need to have higher influence to prevent collisions.

Huber et al. [68] proposed the following aggregation procedure. Let $\mathbf{u}_i$ denote the individual modulations, with norms n_i. The aggregate modulation $\mathbf{u}$ is calculated as

$$\mathbf{u} = n_a \mathbf{u}_a, \tag{7.13}$$

where n_a and $\mathbf{u}_a$ are the aggregate norm and direction.

The aggregate norm is computed as

$$n_a = \sum_{i=1}^{N} w_i n_i, \tag{7.14}$$

$$w_i = \frac{b_i}{\sum_{j=1}^{N} b_j}, \tag{7.15}$$

$$b_i = \prod_{1 \le j \le N, j \ne i} \Gamma_j(\mathbf{x}). \tag{7.16}$$

The above definition ensures that $\sum_{i=1}^{N} w_i = 1$, and $w_i \to 1$ when $\mathbf{x}$ approaches $\mathcal{O}_i$ (and only $\mathcal{O}_i$, which holds as long as obstacles are disjoint).

$\mathbf{u}_a$ is instead computed using what Huber et al. [68] calls "$\boldsymbol{\kappa}$-space interpolation." First, similar to the basis vector matrix $E(\mathbf{x})$ introduced above, we construct another such matrix, but with respect to the original controller $\mathbf{x}^* - \mathbf{x}$. We denote it as $R = [(\mathbf{x}^* - \mathbf{x})/||\mathbf{x}^* - \mathbf{x}||, \mathbf{e}_1, ..., \mathbf{e}_{d-1}]$, where $\mathbf{e}_1, ..., \mathbf{e}_{d-1}$ are again orthonomal vectors spanning the null space.

For each $\mathbf{u}_i$, we compute its coordinate in this new R-frame as $\hat{\mathbf{u}}_i = R^{-1}\mathbf{u}_i$. Its $\boldsymbol{\kappa}$-space representation is

$$\boldsymbol{\kappa}_i = \frac{\arccos(\hat{\mathbf{u}}_i^{(1)})}{\sum_{m=2}^{d} \hat{\mathbf{u}}_i^{(m)}} \left[\hat{\mathbf{u}}_i^{(2)}, ..., \hat{\mathbf{u}}_i^{(d)}\right]^T \in \mathbb{R}^{d-1}, \tag{7.17}$$

where the superscript (m) refers to the m-th entry. $\boldsymbol{\kappa}_i$ is a scaled version of the $\hat{\mathbf{u}}_i$ with the first entry removed. We perform the aggregation in this $\boldsymbol{\kappa}$-space using the weights w_i calculated above (7.18), transform it back to the R-frame (7.19), and finally transform it back to the original frame (7.20):

$$\boldsymbol{\kappa}_a = \sum_{i=1}^{N} w_i \boldsymbol{\kappa}_i \tag{7.18}$$

$$\hat{\mathbf{u}}_a = \left[\cos(||\boldsymbol{\kappa}_a||), \frac{\sin(||\boldsymbol{\kappa}_a||)}{||\boldsymbol{\kappa}_a||}\boldsymbol{\kappa}_a^T\right]^T \tag{7.19}$$

$$\mathbf{u}_a = R\hat{\mathbf{u}}_a. \tag{7.20}$$

As mentioned in Eq. 7.13, the final modulation is $\mathbf{u} = n_a\mathbf{u}_a$.

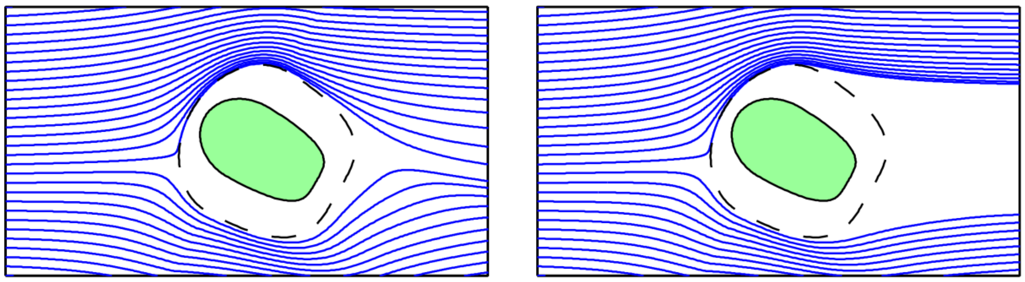


Figure 7-12: Tail effect (left) and its removal (right), reproduced from Figure 7 by Khansari-Zadeh and Billard [81]. The target is on the far right side.

Tail-Effect

An artifact of the above formulation is the "tail-effect," where the robot is modulated to go around the obstacle even when it has passed by the obstacle and the remaining trajectory has no chance of collision under the non-modulated controller. This effect has been observed by Khansari-Zadeh and Billard [81] for a related but different type of modulation. Figure 7-12, reproduced from the paper by Khansari-Zadeh and Billard [81, Figure 7], shows the tail effect on the left and its removal on the right. This tail effect induces the placement of obstacles at the end of the "diagonal corridor" as seen in our straight-line deviation experiments (Figure 7-6, left). If desired, the DS formulation can be modified to remove this effect.

7.7.2 2D Environment Details

In this domain, the environment is the area defined as $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. The goal is to navigate from $[x_{\text{start}}, y_{\text{start}}]$ to $[x_{\text{goal}}, y_{\text{goal}}]$. We define a flexible environment representation as a summation of radial basis function (RBF) kernels centered at so-called *obstacle points*. Specifically, given N_O obstacle points $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{N_O} \in \mathbb{R}^2$, the environment is defined as

$$e(\mathbf{p}) = \sum_{i=1}^{N_O} \exp\left(-\gamma ||\mathbf{p} - \mathbf{p}_i||_2^2\right), \tag{7.21}$$

and each point $\mathbf{p}$ is an obstacle if $e(\mathbf{p}) > \eta$, for $\eta < 1$ to ensure each obstacle point $\mathbf{p}_i$ is exposed as an obstacle. Our environments are bounded by $[-1.2, 1.2] \times [-1.2, 1.2]$, and the goal is to navigate from $[-1, -1]$ to $[1, 1]$. $N_O = 15$ and p_i coordinates are sampled uniformly in $x_i, y_i \in [-0.7, 0.7]$. A smaller γ and η makes the obstacles larger and more likely to be connected; we choose $\gamma = 25$ and $\eta = 0.9$. Figure 7-4 shows random obstacle configurations demonstrating high diversity in this environment. We also implement a simple simulator: given the current robot position $[x, y]$ and the action $[\Delta x, \Delta y]$, the simulator clamps $\Delta x, \Delta y$ to the range of [-0.03, 0.03], and then moves the robot to $[x + \Delta x, y + \Delta y]$ if there is no collision, and otherwise simulates a frictionless inelastic collision (i.e., compliant sliding) that moves the robot tangent to the obstacle.

7.7.3 Details of 2D Navigation Controllers

IL Controller The imitation learning controller is a memoryless policy implemented as a fully connected neural network with two hidden layers of 200 neurons each and ReLU activations. The input is 18 dimensional, with two dimensions for the current (x, y) position of the robot, and 16 dimensions for a lidar sensor in 16 equally-spaced directions, with a maximum range of 1. The network predicts the heading angle θ, and the controller operates on the action of $[\Delta x, \Delta y] = [0.03 \cos\theta, 0.03 \sin\theta]$.

The network is trained on smoothed RRT trajectories. Specifically, we use the RRT controller to find and discretize a trajectory. Then the smoothing procedure repeatedly replaces each point by the mid-point of its two neighbors, absent collisions. When this process converges, each point on the trajectory becomes one training data point.

Since only local observations are available and the policy is memoryless, the robot may get stuck in obstacles, which happens in approximately 10% of the runs. In addition, while the output target is continuous, a regression formulation with mean-squared error (MSE) loss is inappropriate, due to multimodality of the output. For example, when the robot is facing an obstacle, moving to either left or right would avoid it, but if both directions appear in the dataset, the MSE loss would drive the prediction to the average, resulting in a head-on collision. This problem has been studied for tasks such as grasping [174] and autonomous driving [166]. We follow the latter and treat this problem as classification with 100 bins in the $[0, 2\pi]$ range.

DS Controller For the DS controller, there are two technical challenges in using the modulation [68] on our RBF-defined environment. First, we need to identify and isolate each individual obstacle, and second, we need to define a Γ-function for each obstacle.

To find all obstacles, we discretize the environment into an occupancy grid of resolution 150×150 covering the area of $[-1.2, 1.2] \times [-1.2, 1.2]$. Then we find connected components using flood fill, and each connected component is taken to be an obstacle.

To define a Γ-function for each obstacle, we first choose the reference point as the center of mass of the connected component. Then we cast 50 rays in 50 equally spaced directions from the reference point and find the intersection point of each ray with the boundary of the connected component. Finally, we connected those intersections in sequence and get a polygon. In case of multiple intersection points, we take the farthest point as vertex of the polygon, essentially completing the non-star-shaped obstacle to be star-shaped, as shown in Figure 7-13.

Given an arbitrary point $\mathbf{x}$, we define

$$\Gamma(\mathbf{x}) = \frac{||\mathbf{x} - \mathbf{r}||}{||\mathbf{i} - \mathbf{r}||}, \tag{7.22}$$

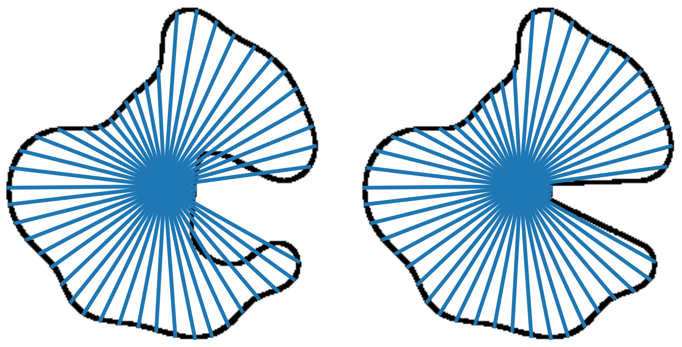


Figure 7-13: Left: an obstacle which is not star-shaped. Some radial lines extending from the obstacle's reference point cross the boundary of the obstacle twice. Right: the same obstacle, modified to instead be star-shaped.

where $\mathbf{r}$ is the reference point and $\mathbf{i}$ is the intersection point with the polygon of the ray from $\mathbf{r}$ in $\mathbf{x} - \mathbf{r}$ direction. It is easy to see that this Γ definition satisfies all three requirements for Γ-functions listed in Appendix 7.7.1.

Finally, to compensate for numerical errors in the process (e.g., approximating obstacles with polygons), we define the control inside obstacle to be the outward direction, which helps preventing the robot from getting stuck at obstacle boundaries in practice. Three examples of DS modulation of the 2D navigation environment are shown in Figure 7-14.

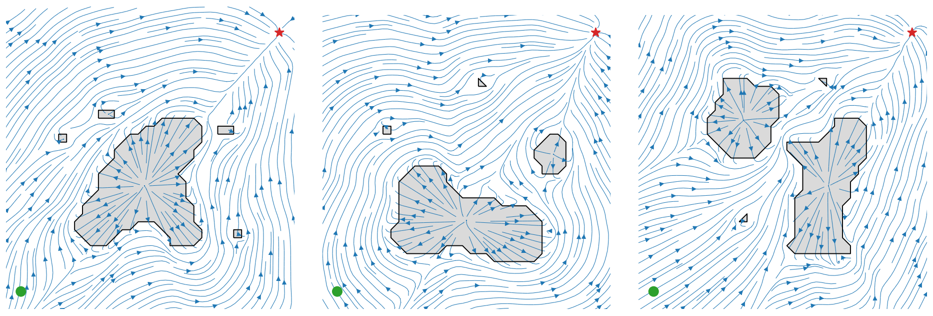


Figure 7-14: Streamlines showing the modulation effect of the dynamical system for three 2D navigation tasks. The environments correspond to the first three examples of Figure 7-4. Green dots are starting positions and red stars are navigation targets.

7.7.4 Details of 7DoF Arm Reaching Controllers

RRT Controller Since the target location is specified in the task space, we first find the target joint space configuration using inverse kinematics (IK). The initial configuration starts with the arm positioned down on the same side as the target. If the IK solution is in collision, we simulate the arm moving to it using position control, and redefine the final configuration at equilibrium as the target (i.e., its best effort reaching configuration). We solve the IK using Klamp't [59].

RL Controller The RL controller implements the proximal policy gradient (PPO) algorithm [137]. The state space is 22-dimensional and consists of the following:

- 7D joint configuration of the robot,
- 3D position of the end-effector,
- 3D roll-pitch-yaw of the end effector,
- 3D velocity of the end-effector,
- 3D position of the target,
- 3D relative position from the end-effector to the target.

The action is 7-dimensional for movement in each joint, which is capped at $[-0.05, 0.05]$.

Both the actor and the critic are implemented with fully connected networks with two hidden layers of 200 neurons each, and ReLU activations. The action is parametrized as Gaussian where the actor network predicts the mean, and 7 standalone parameters learns the log variance for each of the 7 action dimensions. At test time, the policy deterministically outputs the mean action given a state.

DS Controller For the DS controller in 7DoF arm reaching, we face the same challenges as in 2D navigation: defining an appropriate Γ-function for the obstacle configuration that holds the three properties introduced by Huber et al. [68] (listed in Appendix 7.7.1). Additionally, the DS modulation technique does not consider the robot's morphology, end-effector shape, or workspace limits because it only modulates the state of a point-mass. Thus, we implement several adaptations. First, we modulate the 3D position of the tip of the end-effector. The desired velocity of the end-effector tip, given by the modulated linear controller, is then tracked by the 7DoF arm via the same position-level IK solver as the RRT controller.

Second, we used a support vector machine (SVM) to learn the obstacle boundary from a list of points in the obstacle and free spaces, an approach originally proposed by Mirrazavi Salehian et al. [105]. Then the decision function of the SVM is used as the Γ-function. As shown in Figure 7-15, we discretize the 3D workspace of the robot and generate a dataset of points in the obstacle space as negative class and those in the free space as positive class.

Using the radial basis function (RBF) kernel $K(\mathbf{x}_1, \mathbf{x}_2) = e^{-\gamma||\mathbf{x}_1 - \mathbf{x}_2||^2}$, with kernel

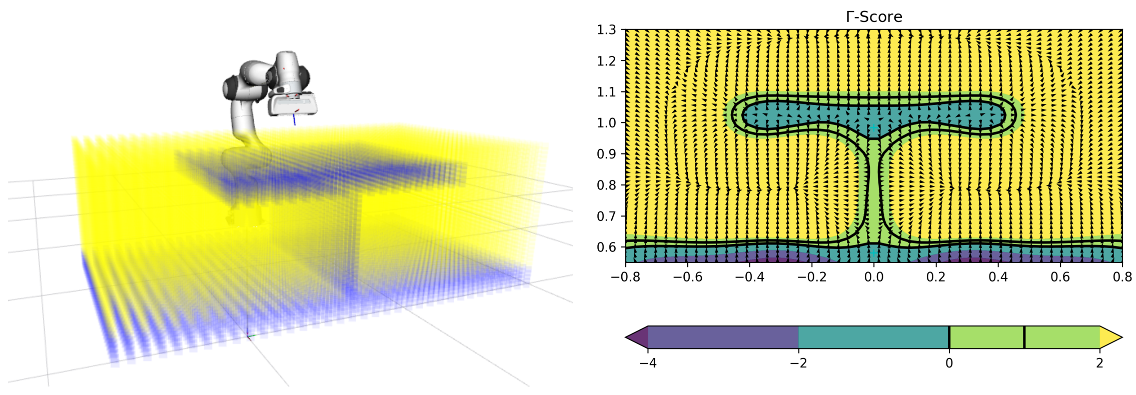


Figure 7-15: Left: the division of 3D space as either containing an obstacle or free space. This data is used to train an SVM, which acts as an interpolator. The classification scores of the SVM are used as the Γ function for this 3D reaching task. Right: a 2D slice showing the smoothed Γ scores.

width γ, the SVM decision function $\Gamma(\mathbf{x})$ has the following form:

$$\Gamma(\mathbf{x}) = \sum_{i=1}^{N_{sv}} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b = \sum_{i=1}^{N_{sv}} \alpha_i y_i e^{-\gamma ||\mathbf{x}-\mathbf{x}_i||^2} + b, \tag{7.23}$$

and the equation for $\nabla\Gamma(\mathbf{x})$ is naturally derived as follows:

$$\nabla\Gamma(\mathbf{x}) = \sum_{i=1}^{N_{sv}} \alpha_i y_i \frac{\partial K(\mathbf{x}, \mathbf{x}_i)}{\partial \mathbf{x}} = -\gamma \sum_{i=1}^{N_{sv}} \alpha_i y_i e^{-\gamma ||\mathbf{x}-\mathbf{x}_i||^2} (\mathbf{x} - \mathbf{x}_i). \tag{7.24}$$

In Eq. 7.23 and 7.24, $\mathbf{x}_i$ ($i = 1, ..., N_{sv}$) are the support vectors from the training dataset, y_i are corresponding collision labels (-1 if position is collided, $+1$ otherwise), $0 \leq \alpha_i \leq C$ are the weights for support vectors and $b \in \mathbb{R}$ is decision rule bias. Parameter $C \in \mathbb{R}$ is a penalty factor used to trade-off between errors minimization and margin maximization. We empirically set the hyper-parameters of the SVM to $C = 20$ and $\gamma = 20$. Parameters α_i and b and the support vectors $\mathbf{x}_i$ are estimated by solving the optimization problem for the soft-margin kernel SVM using `scikit-learn`. Using this learned Γ-function, Figure 7-16 shows two examples of the modulated trajectory.

Finally, given a desired modulated 3D velocity for the end-effector tip, $\dot{\mathbf{x}}_M = \mathbf{u}_M(\mathbf{x})$, we compute the next desired 3D position by numerical integration:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}_M(\mathbf{x}_t)\Delta t \tag{7.25}$$

where $\mathbf{x}_t, \mathbf{x}_{t+1} \in \mathbb{R}^3$ are the current and next desired 3D position of the tip of the

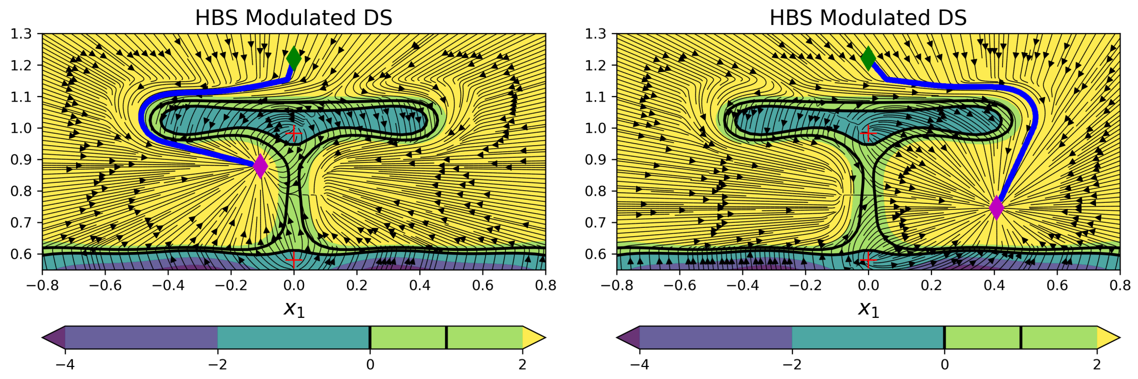


Figure 7-16: Cross-sections showing streamlines of the dynamical system modulation effect for two distinct targets in the 3D reaching task. Red crosses indicate reference points. Green diamond is the initial position of the end-effector for all experiments.

end-effector and $\Delta t = 0.03$ is the control loop time step. $\mathbf{x}_{t+1}$ is then the target in Cartesian world space coordinates that defines the objective of the position-based IK solver implemented in Klamp't [59].

7.7.5 DS Improvement for 7DoF Arm Reaching

The DS controller provides guarantees of convergence to a target in the space where modulation is applied (i.e., task-space in our experiments). To adopt this controller for obstacle avoidance with a robot manipulator, Huber et al. [68] simplifies the robot to a spherical shape with center at the end-effector of a 7DOF arm. This translates to considering the robot as a zero-mass point in 3D space but with the boundaries of the obstacles (described by Γ-functions) expanded by a margin with the size of the radius of the sphere.

Due to the rectangular shape ($6.3 \times 20.7 \times 14$cm) of the Franka robotic hand, fitting a sphere with the radius of the longest axis will over-constrain the controller and drastically reduce the target regions inside the table dividers. We thus implemented the obstacle clearances by extruding the edges of the top table divider by half of the length of the robot's end-effector (10cm) and the width of the divider by half of the height (7cm). Intuitively, this should be enough clearance to avoid the robot's end-effector colliding with the table dividers. However, when coupling the DS controller with the IK solver to control the 7DoF arm, we noticed that the success rate was below 15%, whereas the success rate is 100% when controlling the end-effector only. We then sampled, via RoCUS, the target locations for the minimal final end-effector distance to target and noticed that all of the successful runs were located on the left-side of the partition (Figure 7-8 center right).

Since the DS controller approach does not consider collision avoidance in joint-space, in a constrained environment, the robot's forearm or elbow might get stuck on the edges of the table divider – even though the end-effector is avoiding collision. Due to the asymmetric kinematic structure of the robot arm, it is more prone to these situations on the right side of the table divider. Such an insight is not easy to discover as one must understand how the robot will behave in joint space based on its kinematic structure and the low-level controller used (position-based IK). We thus extended the edge extrusions to 20cm. This change improved the controller success rate and behavior drastically as shown in (Figure 7-8 rightmost).

Chapter 8

Towards Trustworthy Machine Learning

8.1 The Roles of Model Explanations

As black-box models are increasingly used to make important decisions, a good understanding into their various properties is crucial for many reasons. From a legal perspective, there have been various government regulations that require the deployed models to be able to "explain themselves," so that the predictions can be more easily audited [31]. At this level, purely black-box models without any explanation support are simply illegal to use in many settings.

From a practical perspective, model explanations may help developers proactively identify potential problems. For example, if a cancer detection model exploits the watermark spurious correlation, as discussed in Chapter 4, then an explanation that reveals this working mechanism could save people from the future frustration of the model not seeming to work on different sources without the watermark. In addition, explanations may also help identify the use of protected features in discriminative manners and the leakage of training data for generative models, two of the most important topics in fairness and privacy.

Last, from a business perspective, the owner of the model, such as a bank that uses a mortgage approval classifier, has an interest to keep extended relationships with its customers, including when they receive adverse predictions such as an application denial. Retaining the relationship often requires providing justifications of the model decision, so as to lend legitimacy and reasonableness to it, and, more importantly, actionable recourse [155] so that the customer could receive a favorable decision if they follow the recommendation. The latter part is the main topic of counterfactual explanations [157], which we did not cover in this thesis, except slightly at the end of Chapter 5 but represents another family of model explanations distinct from the feature attribution family.

8.2 Thesis Contributions

Recognizing the importance of the role of model explanations in model understanding, this thesis contributes new methods, frameworks and evaluations to the interpretability and transparency of black-box models. These contributions can be considered as answers to the questions of "how to explain model predictions," "how to evaluate model explanations," and "what model predictions to explain." As a whole, these three questions and the corresponding answers illuminate a model understanding pipeline, shown in Figure 1-1, which is reproduced in Figure 8-1 for convenience.

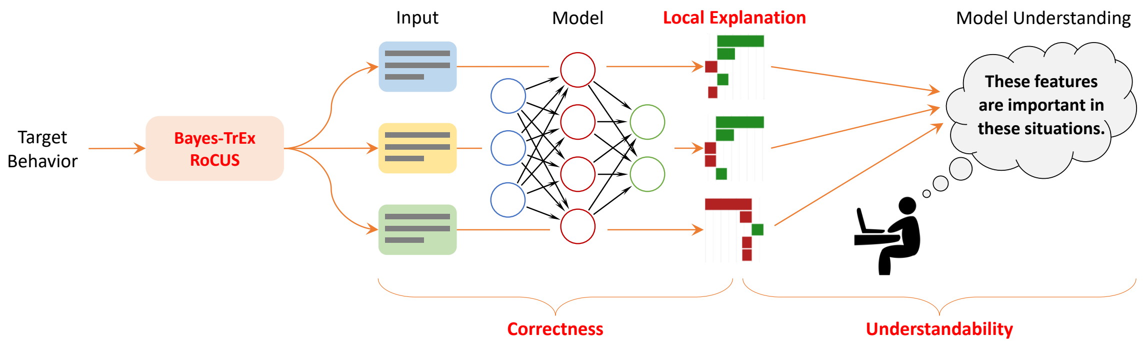


Figure 8-1: The model understanding pipeline (reproduced from Figure 1-1).

8.2.1 How to Explain Model Predictions?

In the background overview of Chapter 2, we see that model explanations, specifically feature attributions, are mainly *defined* according to our intuitive notions of what feature importance means. For example, the gradient saliency [142] reflects the notion that the feature importance should be the model prediction sensitivity under local feature value perturbation. However, recognizing that these definitions are then evaluated on a set of different metrics for quality assessment, such as comprehensiveness and sufficiency [39], Chapter 3 proposes to use the evaluation concepts directly as definitions, and more generally proposes a *duality* between definitions and evaluations in model interpretability.

In particular, for an evaluation metric m, such as comprehensiveness, Chapter 3 proposes the concept of m-solving explanation e^* on a given input as the one that achieves the optimal value of m. Given the existence of e^* (and even closed-form expressions for some m), it is not clear why we should use *anything* other than e^* if we use m to represent the explanation quality.

One possible concern is the hardness of optimization for all but the trivial (and not-so-good) metrics. For example, the two most commonly used metrics, comprehensiveness and sufficiency, are expressed as a summation of model prediction changes under partial feature removal or insertion. While the exact optimization is combinatorial and hard, both metrics admit very efficient approximate beam search algorithms, whose resulting explanations are empirically demonstrated to outperform existing ones (such as the gradient saliency) on the comprehensiveness and sufficiency metric values. Thus, given that a large number of works evaluate their proposed explanations on these two metric values, it seems that the beam search should be the end of this series of investigation.

However, another and perhaps more important concern is that of Goodhart's law, which says, in this context, because these two metrics are explicitly optimized against by the beam search algorithm, they cease to be good metrics. Considering that we do not know what the "perfect" metric is (which if we did, we would just optimize against), we investigate this problem from two angles – explanation overfitting to the metric and overfitting to the model. In both cases, we fail to find convincing evidence that the prophecy of Goodhart's law is fulfilled.

In addition, we set up a dataset modification procedure to induce a ground truth of model working mechanism, as discussed in more detail in Chapter 4. Across a variety of settings, we find that our beam search explanation also performs favorably or competitively with respect to existing methods. Thus, we advocate for using search methods as a general paradigm to find the metric-solving explanations directly, and for future work to establish a better understanding of the properties of the resulting explainer, in relationship to the underlying metric that is being optimized.

8.2.2 How to Evaluate Model Explanations?

Correctness

Other than a new way of defining explanations, the duality observation provides the analogous perspective of using existing definitions as evaluations of model explanations. While we did not empirically explore this in Chapter 3, a deeper study on evaluation metrics, and a more liberal view of what concepts can be used as evaluations, could be insightful as recent works [27, 116] have suggested poor alignment between evaluation methods and between evaluation metric values and effectiveness as demonstrated in user studies.

Fundamentally, these metrics are *proxy* metrics because the ground truth model explanation is not available – this is the very goal of interpretability research! In Chapter 4, we approach this problem from a different perspective and propose a dataset modification procedure to induce a ground truth reasoning mechanism for high-performing models. We achieve this in two steps. First, we weaken, or even erase, the existing correlations between inputs and outputs, such that *any* model could not achieve very high performance using any means. Then we carefully inject highly controlled features according to the new label, which enables a model to make highly accurate predictions. If we do observe this high accuracy, then the model is guaranteed to use the features that we inject, and we therefore can evaluate how well feature attribution methods can detect the usage of these features.

In our experiments, we evaluate explanations for both image and text classifiers, focusing mainly on artifact-like features such as adding a watermark to an image or changing the article words (e.g., "a" and "the") in text. While certain explanation methods are effective at detecting certain features, our findings are overall concerning.

We find that the effectiveness of any given method tends to vary widely across feature types, and there is no single method that consistently outperforms the rest. This inconsistency highlights the issue with the use of feature attribution explanations to identify dataset artifacts and spurious correlations when they are unknown. However, an ensemble of all these methods could be result in a detector with reasonably high recall (i.e., a feature is unlikely to be missed by *all* of the methods), potentially at the cost of low precision (i.e., a feature is also unlikely to be clearly identified by *every* method). We leave this direction of investigation to future work.

Understandability

Furthermore, we argue that correctness should not be the sole desideratum for model explanations. In particular, as model explanations are ultimately used by humans to understand the model, the soundness this model understanding process needs to be be carefully characterized, yet no such mathematical formulations exist. In Chapter 5, we propose a novel factor, understandability, and demonstrate its necessity for achieving good explanations.

We introduce the explanation summary (ExSum) framework, which establishes a rigorous definition of "model understanding," intended to replace the currently *ad hoc* practices of consuming and understanding local model explanations. The individual pieces of model understanding are called ExSum rules and their aggregate is called an ExSum rule union. This rigorous definition allows for quantitative evaluations of model understanding (i.e., ExSum rules and rule unions). We then propose three metrics: coverage, validity and sharpness. Coverage captures the applicability of the model understanding – is it about a broad or narrow aspect of how the model works(e.g., "adjectives are important for the prediction" vs. "non-negated adjectives in short sentences are important only for the positive prediction"). Validity captures the correctness of model understanding (i.e., actually supported by local explanations). Sharpness captures the precision of our model understanding statement (e.g., "all adjectives have very high contributions" vs. "some adjectives have very high contributions while others less so"). These metrics often trade off against each other, demonstrating the difficulty of obtaining very high-quality model understanding from local explanations. Last, we also provide a web application in the Python `exsum` package for assisting human users to come up with well-calibrated model understanding supported by model explanations.

Using ExSum, we evaluate our understanding of two models, a sentiment classifier and a paraphrase detector. In both cases, we find significant limitations in our model understanding as evidenced by the metric scores. At the same time, however, we also clearly see the advantage of using ExSum to synthesize model understanding compared to current, more *ad hoc* approaches in that the former reveals insights that have been overlooked by the latter.

Last, the principles of understandability can be considered as a generalization of several evaluation aspects like plausibility and robustness, which provides a unified setup for discussing them. We additionally highlight its applicability to explanation types beyond feature attribution, and an extension to instance-based explanations is briefly explored.

8.2.3 What Model Predictions to Explain?

Besides definitions and evaluations of explanations, the final piece to complete our model understanding pipeline looks at the very beginning of it: the source of the inputs. The current practices use test set instances, which are often selected either randomly or to highlight certain model reasoning. The ExSum framework advocates for the use of the entire test set for more detailed and quantitative inspections, which can be considered as an improvement of this *ad hoc* approach.

However, both approaches are fundamentally limited to the test set, which may not have good (or even any) coverage of certain model behaviors, such as when the model is likely to be confused between two classes, or how well the model adapts to distribution shift. Answering these questions should ideally be done by a *behavior-driven* approach, where we start from the target behavior that we want to study and identify instances which highlight the behavior. Each behavior is a function of data instance and model prediction. For example, a covariate shift failure behavior requires the data instance to be from the shifted distribution and the model prediction to not equal to the true label. Now that we know what behaviors these instances induce, we can use any interpretability method to study *why* such behavior happens, which is the purpose of the rest of the model understanding pipeline. Because studying these examples gives us a better understanding into the model, we call this approach model transparency-by-example (TrEx).

Chapter 6 introduces the technical contribution to this approach. We first formalize a behavior as a function of data instance and model prediction, and then set up a Bayesian posterior inference procedure under which, with a suitably defined prior and likelihood function, the behavior-conforming instances can be considered as samples from the corresponding posterior distribution, obtainable by Markov-chain Monte Carlo sampling methods. This framework is hence named Bayes-TrEx. In the experiments, we analyze three image models for digit recognition, clothes recognition and visual reasoning, respectively. Using Bayes-TrEx, we can efficiently find instances that induce a variety of behaviors, which would otherwise be hard or impossible using the test set. The inspection of these instances helps us build a more well-rounded understanding of the capability and blind spots of the model, and we describe the use of a saliency map analysis to infer the reason for the occurrence of a behavior.

Building upon Bayes-TrEx, Chapter 7 extends it to study robotic controllers. Analogous to how a behavior in Bayes-TrEx is a function of data instance and model prediction, a robot behavior is a function of task scenario and generated trajectory. We propose a suite of robot-centric behavior definitions, which can be generally classified into intentional and emergent ones. Intentional behaviors are those that are explicitly optimized or considered when defining or learning the robot controller, such as task completion success and energy consumption. By contrast, emergent behaviors (desirable or undesirable) are ones that emerge as a result of the controller definition or learning, but are not explicitly considered as an objective. Examples include the legibility [41] or the expressed emotion [145] of the trajectory.

We present robot controller understanding via sampling (RoCUS), which takes a given behavior and finds scenarios (e.g., configurations of robot start and goal locations) where the behavior is exhibited by the generated trajectory. The technical contribution beyond Bayes-TrEx is the introduction of the α parameter, which intuitively represents the trade-off between scenario typicality (whether this scenario is an edge case) and behavior conformity (how much the trajectory in this scenario highlights the behavior). In addition, RoCUS can also find scenarios with maximal values of behaviors that is unbounded from above, a situation not supported by Bayes-TrEx.

Using RoCUS, we analyze two robot domains, a 2D navigation task with free form obstacles, and a 7 degree-of-freedom arm reaching task with obstacles over target positions. In both cases, RoCUS can identify subtle properties for various robot controllers that are either defined (e.g., rapid-exploring random trees) and learned (e.g., reinforcement learning policies), which shed light on the properties of these controllers and the environments. In the case of a dynamical system-based controller on the arm reaching task, RoCUS also helps identify an issue in its specification caused by the left/right asymmetry in the robot configuration space due to that of the arm kinematic structure.

8.3 Outlook

In some sense, the rigorous examination of the model understanding pipeline in the thesis raises more questions than it answers. In this final part, we give some future directions that would extend and complement works described here to make model explanations truly deliver their promise of helping people understand models.

8.3.1 Improving Feature Attributions

The investigation in Chapter 4 exhibits several issues with current feature attribution explanations. First, deducing the reason for a prediction as the absence of a certain feature is harder than the presence of a feature. For example, if all positive images contain a watermark at the top right corner while none of the negative images have any watermark in a dataset, then many feature attribution methods struggle to explain the reason for a negative prediction is the absence of the watermark, while many of them can highlight the watermark on a positive image for the positive prediction. This is somewhat expected, as the goal of feature attribution is to explain the importance or contribution of every *present* feature. However, a "negative default" can be a very common component in the working mechanism of many models even without spurious correlations like the watermark. For example, a medical diagnosis model may look out for the symptom corresponding to each disease on the X-ray image, and make the no-disease prediction when none is found, yet this prediction can be hard to explain with feature attributions.

One possible remedy is for the explainer to learn to generate an explicit negative default baseline (which may be different for different inputs), and compute feature attribution values with respect to it. Thus, when the prediction is negative, the explanation could simply be the high similarity between the input and the baseline, and the explanations for other classes could represent the deviations of the baseline. As mentioned in Chapter 2, the idea of baseline is already used by methods such as grad$\times$input and IntGrad [146], but these methods typically choose the baseline of zero feature values (e.g., a black image), which is often out of the data distribution.

In addition, results on the text classifiers in Chapter 4 reveal a curious issue with the attention mechanism: when the spurious correlation is very obvious easy to learn without it, the attention weights often fail to reveal the true feature importance.By comparison, on harder tasks for which attentions are useful, the attention scores are often much more reasonable. For example, those for machine translation models generally highlight the word and phrase alignment between source and target languages [158]. Thus, it is worth investigating the extent to which attention scores are correlated with the necessity of the attention mechanism in achieving high model performance. In other words, attention weights may only be "correct" when the model without the attention mechanism cannot achieve the same level of high accuracy. If this trend is confirmed, we should be wary of interpreting attention scores as feature attributions, especially for easy tasks. In this case, regularizing or otherwise encouraging the attention scores to be more correct is much needed, so that we can unconditionally rely on them.

8.3.2 Rethinking Feature Attributions

While feature attribution explanations are very intuitive, they are also severely limited by the format of assigning a single score for each feature. This mode of operation works best when the true feature importance is additive – each feature contributes a set amount regardless of the values of other features – such as in the case of a linear regression, but this assumption rarely holds for neural network models due to their non-linearity.

When the contribution of a feature depends on the values of others, existing approaches deal with this problem either implicitly [128] or explicitly [99] aggregate contributions under different circumstances into one. Thus, besides computation of these importance scores, another problem is their *presentation*. As long as the explanation is presented in this one-score-per-feature format, there is the risk that it would be mis-interpreted by the users.

How should we best present feature attributions? Novel explanation methods propose to explicitly compute interactions among features [25, 151]. However, visualizing the interactions of individual pairs or triplets of features quickly becomes intractable, when the number of features is large. Thus, aggregating a large number of feature interactions into a small number of semantic groups would be crucial for understanding, and such aggregation is exactly what ExSum advocates.

Moving beyond feature attribution, is this type of explanation even what we want? Although it provides a succinct way of exhibiting the importance of different features, the essence of the conveyed information is counterfactual in nature: a feature being important means that the model prediction would change a lot if the feature were different or withheld. Thus, a feature attribution explanation is intrinsically an aggregate of a set of counterfactual explanations, each of which is represented by $(\hat{x}, f(\hat{x}))$, which explains the change in prediction $f(\hat{x}) - f(x)$ with the change in input $\hat{x} - x$. With many widely used approaches for generating counterfactual explanations, we could again follow the spirit of ExSum to aggregate them in ways other than the one adopted by feature attribution explanations. For example, aggregating all counterfactual explanations that change either of two features could demonstrate that the positive prediction is dependent on the presence of either features (i.e., logical OR), a relationship hard to explain with vanilla feature attributions.

8.3.3 Connecting Explanations with Use Cases

Different stakeholders have different uses for explanations. Chapter 4 mainly focuses on identifying the impact of spurious correlations [47], one of the central concerns of model developers. While our investigations demonstrate mostly negative results in using feature attribution methods as the definitive tools for this purpose, the variability in the performance of each explainer (i.e., different methods are better suited for different artifact types) could foreshadow a more general sense of variability: different methods are better suited for different purposes, of which identifying spurious correlations is but one.

A dilemma that developers frequently face is when and whether to update the current model with a new one that has better performance metrics on paper but may have unknown adverse properties [13]. In this case, generating and analyzing model explanations, besides model transparency techniques like Bayes-TrEx, may provide better understanding into the new model.

In the business world, the client of model training job wants to make sure that the model is compliant to external regulations, which can prevent discrimination or prescribe a basic set of guidelines for the model prediction to follow [31]. On the other side of the table, the customer who is affected by the model prediction (e.g., a mortgage approval model) cares about getting their unfavorable decisions turned into favorable ones with as cheap of a cost as possible (e.g., paying down a particular credit card account) [127]. In both cases, model explanations may again be able to satisfy the requirement of each party.

With all these different use cases of explanations, it is likely that the best explanation for each use case is different, and a more refined understanding into the strengths and weaknesses of each explainer in better or worse supporting each use case goal is very much needed. In addition, a careful taxonomy and potentially hierarchical organization of these use cases is a worthy contribution in itself, which could not only inform missing or overlooked ones, but also provide a shared and unified representation which is potentially predictive of the effectiveness of different explainers.

8.3.4 User Study Results as the Definitive Standard

Related to the point made above and as a recurring theme argued in the thesis, explanations are generated ultimately for *human* consumption. Thus, the definitive standard for the quality and effectiveness of an explanation method should be the results obtained by user studies designed to realistically replicate real-world usage of these explanations. Nonetheless, this goal is easier said than done, and current practices are concerning in several ways.

First, user studies are inherently costly in terms of both time and money, especially large scale ones that are needed to cover different treatment conditions. For example, one study [121] recruited 3,800 participants to study the utilities of explanations for transparent and opaque models (e.g., linear regression vs. neural network). A similar study [14] also recruited more than 1,000 participants. Most of them are carried out via the Amazon Mechanical Turk platform mainly for the ease of subject recruitment and relatively low rate of pay, yet this platform has been criticized on both ethical [56] and scientific grounds [69, 132].

In addition, due to the need to replicate a realistic scenario in which the users consume local explanations to understand the model, the tasks covered by user studies are highly diverse, such as fake review detection [88], house price prediction [121] and age prediction [29]. It is not clear if and to what extent the results obtained in one setting would transfer to a different setting, and whether a standard suite of benchmark tasks could be possible, analogous to those in other fields such as ImageNet [37] for computer vision and GLUE [160] for natural language processing.

Last, the study design itself is a tricky subject. First, it is tempting to infer the effectiveness of explanations from users' self-reported impressions of them, such as whether they think the explanations are useful or whether they trust the model more after having access to the explanations, as done in several studies [e.g., 78, 161]. However, Bansal et al. [14] found that such subjective ratings correlate poorly with objective metrics of task performance. Another common practice in the forward simulation task – asking users to predict the model prediction – is to investigate the difference in user performance on test instances with and without explanations [e.g., 29, 36]. Such a design, however, is easily "gameable" [123] in that the explanation can be generated *after and according to* the model prediction in order to provide maximal information on this particular prediction without actually helping people understand the more general decision making logic employed by the model.

8.3.5 Towards a Future of Trustworthy ML

All the efforts spent, in this thesis in particular and by the community in general, are aimed at making our machine learning systems more trustworthy and reliable. However, the intersections between interpretability and other areas of trustworthy ML are less studied. On the one hand, explanations can help mitigate concerns in aspects such as model fairness, robustness and privacy. For example, similar to how explanations can be used to identify spurious correlations (to various degrees of success), they may also be used to identify discrimination and fairness issues, other non-robustness reasoning patterns and leakage of sensitive information in the training data. Overall, model explanation techniques should be useful additions to our model development and debugging toolbox.

On the other hand, properties of fairness, robustness and privacy that we want for models are also applicable to their explanations. Compared to the "forward" direction of using explanations to enforce these properties, this reverse direction of studying the properties on explanations is significantly less studied. For few instances, Balagopalan et al. [12] and Dai et al. [35] recently found that the quality of model explanations varies across the input space and often disfavors inputs from the minority group. As a result, those people need to deal with not only potentially discriminative model predictions, but also less useful or even misleading explanations for the predictions. Similarly, the robustness in the quality of explanations could also be concerning, where two explanations for similar inputs have very different qualities, or for stochastically generated explanations such as LIME, different random seeds result in explanations of highly varying quality. Note that this is different from the the robustness of explanation values (i.e., similar inputs should have similar explanations) [49], which we argue to be an aspect of understandability in Chapter 5. Last, every explanation reveals information about the model. From a privacy and security perspective, ensuring that an adversary could not reverse engineer the model or the training data from explanations is equally important for the ubiquitous deployment of model explanation techniques, yet this attack model has hardly been considered.

Bibliography

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9505–9515, 2018.

[2] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. Debugging tests for model explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] Julius Adebayo, Michael Muelly, Harold Abelson, and Been Kim. Post hoc explanations may be ineffective for detecting unknown spurious correlation. In *International Conference on Learning Representations (ICLR)*, 2022.

[4] David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. In *ICML Workshop on Human Interpretability in Machine Learning*, 2018.

[5] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[6] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv:1807.06757*, 2018.

[7] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems (RAS)*, 57(5):469–483, 2009.

[8] Sanjeev Arora and Yi Zhang. Do GANs actually learn the distribution? an empirical study. *arXiv:1706.08224*, 2017.

[9] Leila Arras, Ahmed Osman, Klaus-Robert Müller, and Wojciech Samek. Evaluating recurrent neural network explanations. In *ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 113–126. Association for Computational Linguistics, 2019.

[10] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques. *arXiv:1909.03012*, 2019.

[11] Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[12] Aparna Balagopalan, Haoran Zhang, Kimia Hamidieh, Thomas Hartvigsen, Frank Rudzicz, and Marzyeh Ghassemi. The road to explainability is paved with bias: Measuring the fairness of explanations. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 1194–1206. Association for Computing Machinery, 2022.

[13] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S Weld, Walter S Lasecki, and Eric Horvitz. Updates in human-AI teams: Understanding and addressing the performance/compatibility tradeoff. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 2429–2437, 2019.

[14] Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. Does the whole exceed its parts? the effect of AI explanations on complementary team performance. In *ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 1–16, 2021.

[15] Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable neural predictions with differentiable binary variables. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2963–2977. Association for Computational Linguistics, 2019.

[16] Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. A protocol for evaluating the faithfulness of input salience methods for text classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2021.

[17] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6541–6549, 2017.

[18] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research (JMLR)*, 2018.

[19] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning (ICML)*, pages 1613–1622, 2015.

[20] Andreea Bobu, Andrea Bajcsy, Jaime F Fisac, Sampada Deglurkar, and Anca D Dragan. Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections. *IEEE Transactions on Robotics (T-RO)*, 36(3):835–854, 2020.

[21] Serena Booth, Yilun Zhou, Ankit Shah, and Julie Shah. Bayes-TrEx: a bayesian sampling approach to model transparency by example. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 11423–11432, 2021.

[22] Julie Bort. Inside Uber before its self-driving car killed a pedestrian: Sources describe infighting, 'perverse' incentives, and questionable decisions, 2018.

[23] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[24] Chun Sik Chan, Huanqi Kong, and Liang Guanqing. A comparative study of faithfulness metrics for model interpretability methods. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5029–5038. Association for Computational Linguistics, 2022.

[25] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5578–5593. Association for Computational Linguistics, 2020.

[26] Hugh Chen, Joseph D Janizek, Scott Lundberg, and Su-In Lee. True to the model or true to the data? *arXiv:2006.16234*, 2020.

[27] Valerie Chen, Nari Johnson, Nicholay Topin, Gregory Plumb, and Ameet Talwalkar. Use-case-grounded simulations for explanation evaluation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[28] George Chrysostomou and Nikolaos Aletras. Improving the faithfulness of attention-based explanations with task-specific information for text classification. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 477–488. Association for Computational Linguistics, 2021.

[29] Eric Chu, Deb Roy, and Jacob Andreas. Are visual explanations useful? a case study in model-in-the-loop prediction. *arXiv:2007.12248*, 2020.

[30] Benjamin Cohen, Ioan A Şucan, and Sachin Chitta. A generic infrastructure for benchmarking motion planners. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 589–595. IEEE, 2012.

[31] European Commission. 2018 reform of EU data protection rules, 2018.

[32] Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[33] Ian Covert, Scott M Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[34] Marco Cusumano-Towner and Vikash K Mansinghka. AIDE: An algorithm for measuring the accuracy of probabilistic inference algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[35] Jessica Dai, Sohini Upadhyay, Ulrich Aivodji, Stephen H. Bach, and Himabindu Lakkaraju. Fairness via explanation quality: Evaluating disparities in the quality of post hoc explanations. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, page 203–214. Association for Computing Machinery, 2022.

[36] Anubrata Das, Chitrank Gupta, Venelin Kovatchev, Matthew Lease, and Junyi Jessy Li. ProtoTEx: Explaining model decisions with prototype tensors. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2986–2997. Association for Computational Linguistics, 2022.

[37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[39] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4443–4458. Association for Computational Linguistics, 2020.

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

[41] Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. Legibility and predictability of robot motion. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 301–308. IEEE, 2013.

[42] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research (JMLR)*, 2020.

[43] Chuchu Fan, Xin Qin, Yuan Xia, Aditya Zutshi, and Jyotirmoy Deshmukh. Statistical verification of autonomous systems using surrogate models and conformal inference. *arXiv:2004.00279*, 2020.

[44] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3429–3437, 2017.

[45] Daniel J. Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Scenic: A language for scenario specification and scene generation. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 63–78, 2019.

[46] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016.

[47] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[48] Ganesh Ghalme, Vineet Nair, Vishakha Patil, and Yilun Zhou. Long-term resource allocation fairness in average Markov decision process (AMDP) environment. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 525–533, 2022.

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

[50] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE, 2018.

[51] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

[52] Jackson Gorham and Lester Mackey. Measuring sample quality with Stein's method. In *Advances in Neural Information Processing Systems (NIPS)*, pages 226–234, 2015.

[53] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding Atari agents. In *International Conference on Machine Learning (ICML)*, pages 1792–1801, 2018.

[54] Miguel Grinberg. *Flask Web Development: Developing Web Applications with Python.* O'Reilly Media, Inc., 2018.

[55] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

[56] Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P Bigham. A data-driven analysis of workers' earnings on Amazon Mechanical Turk. In *ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 1–14, 2018.

[57] Peter Hase and Mohit Bansal. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5540–5552. Association for Computational Linguistics, 2020.

[58] W Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[59] Kris Hauser. Robust contact generation for robot simulation with unstructured meshes. In *International Symposium on Robotics Research (ISRR)*, volume 114. Springer, 2016.

[60] Kris Hauser and Yilun Zhou. Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space. *IEEE Transactions on Robotics (T-RO)*, 32(6):1431–1443, 2016.

[61] Bradley Hayes and Julie A. Shah. Improving robot controller transparency through autonomous policy explanation. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 303–312, 2017.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[63] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6626–6637, 2017.

[64] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[65] Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research (JMLR)*, 15(1):1593–1623, 2014.

[66] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2019.

[67] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 9737–9748, 2019.

[68] Lukas Huber, Aude Billard, and Jean-Jacques Slotine. Avoidance of convex and concave obstacles with convergence ensured through contraction. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1462–1469, 2019.

[69] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on Amazon Mechanical Turk. In *ACM SIGKDD Workshop on Human Computation*, pages 64–67, 2010.

[70] Aya Abdelsalam Ismail, Mohamed Gunady, Héctor Corrada Bravo, and Soheil Feizi. Benchmarking deep learning interpretability in time series predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[71] Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First Quora dataset release: Question pairs, 2017.

[72] Sarah Jabbour, David Fouhey, Ella Kazerooni, Michael W Sjoding, and Jenna Wiens. Deep learning applied to chest X-rays: Exploiting and preventing shortcuts. In *Machine Learning for Healthcare Conference (MLHC)*, pages 750–782, 2020.

[73] Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4198–4205. Association for Computational Linguistics, 2020.

[74] Alon Jacovi and Yoav Goldberg. Aligning faithful interpretations with their social attribution. *Transactions of the Association for Computational Linguistics (TACL)*, 9:294–310, 2021.

[75] Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 3543–3556. Association for Computational Linguistics, 2019.

[76] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):3019–3026, 2020.

[77] Yan Jia, John McDermid, Tom Lawton, and Ibrahim Habli. The role of explainability in assuring safety of machine learning in healthcare. *IEEE Transactions on Emerging Topics in Computing (T-ETC)*, 10(4):1746–1760, 2022.

[78] Frank Kaptein, Joost Broekens, Koen Hindriks, and Mark Neerincx. Personalised self-explanation by robots: The role of goals versus beliefs in robot-action explanation for children and adults. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 676–682. IEEE, 2017.

[79] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research (IJRR)*, 30 (7):846–894, 2011.

[80] Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations (ICLR)*, 2019.

[81] Seyed Mohammad Khansari-Zadeh and Aude Billard. A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots (AuRo)*, 32(4): 433–454, 2012.

[82] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 500–505. IEEE, 1985.

[83] Junkyung Kim, Matthew Ricci, and Thomas Serre. Not-So-CLEVR: Learning same–different relations strains feedforward neural networks. *Interface Focus*, 8(4), 2018.

[84] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[85] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning (ICML)*, pages 5338–5348, 2020.

[86] Isaac Lage, Daphna Lifschitz, Finale Doshi-Velez, and Ofra Amir. Exploring computational user models for agent policy summarization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1401–1407, 2019.

[87] Fabien Lagriffoul, Neil T Dantam, Caelan Garrett, Aliakbar Akbari, Siddharth Srivastava, and Lydia E Kavraki. Platform-independent benchmarks for task and motion planning. *IEEE Robotics and Automation Letters (RA-L)*, 3(4): 3765–3772, 2018.

[88] Vivian Lai, Han Liu, and Chenhao Tan. “why is ‘Chicago’ deceptive?” towards building model-driven tutorials for humans. In *ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 1–13, 2020.

[89] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1675–1684, 2016.

[90] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Faithful and customizable explanations of black box models. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, pages 131–138. Association for Computing Machinery, 2019.

[91] Steven M LaValle. *Planning Algorithms.* Cambridge University Press, 2006.

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

[93] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations (ICLR)*, 2018.

[94] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 107–117. Association for Computational Linguistics, 2016.

[95] Andre Lemme, Yaron Meirovitch, M Khansari-Zadeh, Tamar Flash, Aude Billard, and Jochen J Steil. Open-source benchmarking for learned reaching motion generation in robotics. *Paladyn, Journal of Behavioral Robotics*, 6(1), 2015.

[96] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 681–691. Association for Computational Linguistics, 2016.

[97] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[98] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*, 2019.

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

[100] Ronny Luss, Pin-Yu Chen, Amit Dhurandhar, Prasanna Sattigeri, Yunfeng Zhang, Karthikeyan Shanmugam, and Chun-Chen Tu. Leveraging latent features for local explanations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1139–1149, 2021.

[101] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 9(Nov):2579–2605, 2008.

[102] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[103] Jeffrey Mahler, Rob Platt, Alberto Rodriguez, Matei Ciocarlie, Aaron Dollar, Renaud Detry, Maximo A Roa, Holly Yanco, Adam Norton, Joe Falco, et al. Guest editorial open discussion of robot grasping benchmarks, protocols, and metrics. *IEEE Transactions on Automation Science and Engineering (T-ASE)*, 15(4):1440–1442, 2018.

[104] Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *IEEE International Conference on Data Mining (ICDM)*, pages 1020–1025. IEEE, 2012.

[105] Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard. A unified framework for coordinated multi-arm motion planning. *The International Journal of Robotics Research (IJRR)*, 37(10):1205–1232, 2018.

[106] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *ACM Conference on Fairness, Accountability, and Transparency (FAT*)*, pages 220–229, 2019.

[107] Mark Moll, Ioan A Sucan, and Lydia E Kavraki. Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. *IEEE Robotics & Automation Magazine (RA-M)*, 22(3):96–102, 2015.

[108] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[109] Milad Mostavi, Yu-Chiao Chiu, Yufei Huang, and Yidong Chen. Convolutional neural network models for cancer type prediction based on gene expression. *BMC Medical Genomics*, 13(5):1–13, 2020.

[110] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. PyRobot: An open-source robotics framework for research and benchmarking. *arXiv:1906.08236*, 2019.

[111] Glenford J Myers, Tom Badgett, Todd M Thomas, and Corey Sandler. *The Art of Software Testing*, volume 2. Wiley Online Library, 2004.

[112] Moin Nadeem, Anna Bethke, and Siva Reddy. StereoSet: Measuring stereotypical bias in pretrained language models. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 5356–5371. Association for Computational Linguistics, 2021.

[113] Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.

[114] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

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

[116] Giang Nguyen, Daeyoung Kim, and Anh Nguyen. The effectiveness of feature attribution methods and its correlation with automatic evaluation scores. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 26422–26436, 2021.

[117] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning (ICML)*, pages 3809–3818, 2018.

[118] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. TensorFuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning (ICML)*, pages 4901–4911, 2019.

[119] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[120] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016.

[121] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Wortman Vaughan, and Hanna Wallach. Manipulating and measuring model interpretability. In *ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 1–52, 2021.

[122] Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C Lipton. Learning to deceive with attention-based explanations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4782–4793. Association for Computational Linguistics, 2020.

[123] Danish Pruthi, Rachit Bansal, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C. Lipton, Graham Neubig, and William W. Cohen. Evaluating explanations: How much do explanations from the teacher aid students? *Transactions of the Association for Computational Linguistics (TACL)*, 10:359–375, 2022.

[124] Iyad Rahwan, Manuel Cebrian, Nick Obradovich, Josh Bongard, Jean-François Bonnefon, Cynthia Breazeal, Jacob W Crandall, Nicholas A Christakis, Iain D Couzin, Matthew O Jackson, et al. Machine behaviour. *Nature*, 568(7753): 477–486, 2019.

[125] Amir Rasouli and John K Tsotsos. Autonomous vehicles that interact with pedestrians: A survey of theory and practice. *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, 21(3):900–918, 2019.

[126] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv:1801.02854*, 2018.

[127] Kaivalya Rawal and Himabindu Lakkaraju. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 12187–12198, 2020.

[128] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144, 2016.

[129] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on Artificial Intelligence (AAAI)*, volume 32, 2018.

[130] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4902–4912. Association for Computational Linguistics, 2020.

[131] Alexis Ross, Ana Marasović, and Matthew Peters. Explaining NLP models via minimal contrastive editing (MiCE). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3840–3852. Association for Computational Linguistics, 2021.

[132] Joel Ross, Andrew Zaldivar, Lilly Irani, and Bill Tomlinson. Who are the Turkers? worker demographics in Amazon Mechanical Turk. Technical report, University of California, Irvine, 2009.

[133] Alvin E Roth. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[134] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[135] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems (T-NNLS)*, 28(11):2660–2673, 2017.

[136] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems (T-NNLS)*, 28(11):2660–2673, 2016.

[137] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[138] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 618–626. IEEE, 2017.

[139] Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2931–2951. Association for Computational Linguistics, 2019.

[140] Li Shen, Laurie R Margolies, Joseph H Rothstein, Eugene Fluder, Russell McBride, and Weiva Sieh. Deep learning to improve breast cancer detection on screening mammography. *Scientific Reports*, 9(1):1–12, 2019.

[141] Ke Si, Ying Xue, Xiazhen Yu, Xinpei Zhu, Qinghai Li, Wei Gong, Tingbo Liang, and Shumin Duan. Fully end-to-end deep-learning-based diagnosis of pancreatic tumors. *Theranostics*, 11(4), 2021.

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

[143] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: Removing noise by adding noise. *arXiv:1706.03825*, 2017.

[144] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642. Association for Computational Linguistics, 2013.

[145] Arjun Sripathy, Andreea Bobu, Zhongyu Li, Koushil Sreenath, Daniel S. Brown, and Anca D. Dragan. Teaching robots to span the space of functional expressive motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13406–13413, 2022.

[146] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, pages 3319–3328, 2017.

[147] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction.* MIT Press, 2018.

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

[149] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13888–13899, 2019.

[150] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Between-class learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5486–5494, 2018.

[151] Michael Tsang, Sirisha Rambhatla, and Yan Liu. How does this interaction affect me? interpretable attribution for feature interactions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6147–6159, 2020.

[152] Mycal Tucker, Yilun Zhou, and Julie A Shah. Latent space alignment using adversarially guided self-play. *International Journal of Human–Computer Interaction (IJHCI)*, 38(18-20):1753–1771, 2022.

[153] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[154] Julen Urain, Puze Liu, Anqi Li, Carlo D'Eramo, and Jan Peters. Composable energy policies for reactive motion generation and reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2021.

[155] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *ACM Conference on Fairness, Accountability, and Transparency (FAT*)*, pages 10–19, 2019.

[156] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[157] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv:2010.10596*, 2020.

[158] Jesse Vig. A multiscale visualization of attention in the transformer model. In *Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, pages 37–42. Association for Computational Linguistics, 2019.

[159] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[160] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355. Association for Computational Linguistics, 2018.

[161] Tong Wang, Jingyi Yang, Yunyi Li, and Boxiang Wang. Partially interpretable estimators (PIE): Black-box-refined interpretable machine learning. *arXiv:2105.02410*, 2021.

[162] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20. Association for Computational Linguistics, 2019.

[163] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[164] Robert Wolfe and Aylin Caliskan. Low frequency names exhibit bias and overfitting in contextualizing language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 518–532. Association for Computational Linguistics, 2021.

[165] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 6707–6723. Association for Computational Linguistics, 2021.

[166] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2174–2182, 2017.

[167] Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance. *arXiv:1907.09701*, 2019.

[168] Mo Yu, Shiyu Chang, Yang Zhang, and Tommi Jaakkola. Rethinking cooperative rationalization: Introspective extraction and complement control. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4085–4094. Association for Computational Linguistics, 2019.

[169] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding DQNs. In *International Conference on Machine Learning (ICML)*, pages 1899–1908, 2016.

[170] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

[171] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[172] Yiming Zheng, Serena Booth, Julie Shah, and Yilun Zhou. The irrationality of neural rationale models. In *2nd Workshop on Trustworthy Natural Language Processing (TrustNLP)*. Association for Computational Linguistics, 2022.

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

[174] Yilun Zhou and Kris Hauser. 6DOF grasp planning by optimizing a deep learning scoring function. In *RSS Workshop on Revisiting Contact – Turning a Problem into a Solution*, 2017.

[175] Yilun Zhou and Kris Hauser. Incorporating side-channel information into convolutional neural networks for robotic tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2177–2183. IEEE, 2017.

[176] Yilun Zhou and Julie Shah. The solvability of interpretability evaluation metrics. In *Findings of the Association for Computational Linguistics: EACL 2023*. Association for Computational Linguistics, 2023.

[177] Yilun Zhou, Benjamin Burchfiel, and George Konidaris. Representing, learning, and controlling complex object interactions. *Autonomous Robots (AuRo)*, 42 (7):1355–1367, 2018.

[178] Yilun Zhou, Steven Schockaert, and Julie Shah. Predicting ConceptNet path quality using crowdsourced assessments of naturalness. In *The World Wide Web Conference (WebConf)*, pages 2460–2471, 2019.

[179] Yilun Zhou, Julie Shah, and Steven Schockaert. Learning household task knowledge from WikiHow descriptions. In *5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 50–56. Association for Computational Linguistics, 2019.

[180] Yilun Zhou, Serena Booth, Nadia Figueroa, and Julie Shah. RoCUS: Robot controller understanding via sampling. In *Conference on Robot Learning (CoRL)*, pages 850–860, 2021.

[181] Yilun Zhou, Adithya Renduchintala, Xian Li, Sida Wang, Yashar Mehdad, and Asish Ghoshal. Towards understanding the behaviors of optimal deep active learning algorithms. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1486–1494, 2021.

[182] Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. Do feature attribution methods correctly attribute features? In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[183] Yilun Zhou, Marco Tulio Ribeiro, and Julie Shah. ExSum: From local explanations to model understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, 2022.