# Towards an Application-based Pipeline for Explainability

Gregory Plumb

December 2022
CMU-ML-22-107

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

**Thesis Committee:**
Ameet Talwalkar, Chair
Zachary Lipton
Carlos Guestrin, Stanford University
Been Kim, Google Brain
Marco Tulio Ribeiro, Microsoft Research

*Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy*

This research was supported by the National Science Foundation award numbers 1705121 and 1838017.

**Keywords:** Machine Learning, Interpretable ML, Explainable ML

*To my mentors, friends, and family.*

# Abstract

The typical start to a paper on explainability is something along the lines of "as machine learned models have a greater impact on society, there is a greater need for us to be able to understand them." Note that this argument consists of two implicit steps. First, addressing potential problems with our models is more important as their impact on society grows. Second, understanding a model, in some abstract way, will allow us to address those potential problems. However, when we look at specific applications of explainability (*i.e.,* using an explainability method to address a specific potential problem with a model), this second step often turns out to be incorrect. So, the question is: how do we rigorously address a specific application of explainability? In this thesis, we provide an answer to this question by demonstrating the effectiveness of an application-based pipeline for explainability.

This thesis is organized as follows. To start, we summarize the application-based pipeline that we follow. At a high level, this pipeline starts by defining the application, selecting methods to address that application, and then designing a series of increasingly complex and realistic evaluations based on that application. Then, we demonstrate the effectiveness of this pipeline by applying it to several different applications of explainability: helping end-users interact with a model, helping domain experts discover new knowledge from a model, and helping model-developers debug a model (in two different ways, by detecting spurious patterns and by detecting more general systemic errors). In the process of doing so, we make contributions towards each of these applications in terms of new explainability methods and evaluations.

# Acknowledgements

I would like to thank my advisor, Ameet Talwalkar. His guidance on research, collaboration, and writing (even if my love of abstraction and conciseness has persisted) have been essential to my graduate studies. Discovering the ins and outs of explainability alongside him was truly an enjoyable process.

I would like to thank my undergraduate advisor, Vikas Singh. If he had not taken a risk taking a freshman into his lab, I almost surely would not be here today.

I would like to thank Marco Tulio Ribeiro. He really helped me get into model debugging, which I truly appreciate.

I would like to thank the other members of my thesis committee. Much of their work has shaped my research interests and methodology, so I should not have been surprised that their feedback on my thesis proposal was spot on. While this led to a bit of a pivot, the resulting work is all the stronger for it.

I would like to thank everyone who I had the pleasure of collaborating with during my graduate studies and the other members of SAGE lab. The discussions that I have had with them have been enjoyable and very helpful. This is especially true for Joon Kim, Valerie Chen, and Nari Johnson, who have been the core of “team explainability” in the lab.

I would like to thank my family, who have always supported me and taught me the value of diligence.

I would like to thank my friends, especially Nicholay Topin, who have put up with me and made the last years fun.

Last, but not least, I would like to thank my partner, Dana O’Connor, who is always looking out for me.

Finishing my studies has truly taken a team and I could not have asked for a better one.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

The typical start to a paper on explainability is something along the lines of "as machine learned models have a greater impact on society, there is a greater need for us to be able to understand them." Interestingly, this statement is emblematic of several critiques of explainability that have been consistently brought up and that largely persist today [20, 31, 61]:

- Explainability is not a monolithic concept. As a result, we must define which specific aspect of a model's behavior an explainability method helps us understand.
- There are a wide range of potential problems with a model that may have caused us to want to better understand it. For example, we may want to help an end-user interact with a model that is making predictions about them or we may want to help a model-developer identify specific subsets of the data where their model is performing worse. Further, it is likely that different potential problems will be best addressed by different methods. As a result, we must define what specific potential problem(s) with a model we are addressing.
- There is a lack of rigorous protocols for evaluating explainability methods. Further, it is uncommon to directly test the usefulness of an explainability method for addressing a specific potential problem with a model. As a result, we do not know which of the proxy metrics or desired properties proposed for explainability methods are actually informative.

In short, explainability has primarily focused on helping us understand a model, in some abstract way, and has mostly ignored showing that doing so is useful for addressing specific potential problems with that model. This makes it difficult for practitioners to find a method that addresses their problems of interest and for researchers to effectively compare methods.

## 1.1 Thesis Question

Motivated by these persisting critiques, this thesis focuses on the following question:

How do we *rigorously* address a *specific application* of *explainability*?

Before explaining our answer to this question in Section 1.1.4, we have to define some terms.

### 1.1.1 What do we mean by "explainability?"

We mean the research area that studies the collection of methods that aim to help us understand the patterns that a model uses to make predictions. While explainability methods share this common goal, they vary widely in terms of the technical objectives they optimize for and in terms of how directly those technical objectives relate to this goal [20].

For example, consider several different ways that an explainability method could convey that a model trained to detect tennis rackets in an image is relying on the presence of a person:[1]

- It could use a counterfactual approach and tell us that the model's predictions frequently change when we remove the people from an image [75].
- It could use a feature attribution approach and show us that the pixels corresponding to people have a positive influence on the model's predictions [45].
- It could use a concept-based approach and tell us that the "person concept" has a positive influence on the model's predictions [44, 50].
- It could use an example-based approach and show us that many of the training images that positively influence the model's predictions have people but no tennis rackets [49, 117].
- It could use a performance-based approach and tell us that the model has a lower recall on images without people than with people and has a higher false positive rate on images with people than without people [74, 75].

Note that this definition means that some methods that are explainability methods by-function are not explainability methods in-name. The aforementioned example-based approach is one example because it uses a method from robust statistics [49]. The aforementioned performance-based approach is another example that we will discuss later in this thesis.

### 1.1.2 What do we mean by a "specific application?"

We mean addressing a clearly defined potential problem with a model. In the context of explainability, a reasonable litmus test is to ask "who is the intended user of this explainability method and what notion of utility is increased by using it?" If the answer to this question is not clear, then that explainability method was not shown to be useful for a specific application.

For example, consider a generalization of the tennis racket example, where the goal is to design an explainability method to find patterns of the form "the model's prediction for some task, $Y$, depends on some feature, $X$." In general, these patterns lead to sensitivity to distribution shift, when $X$ should not be related to $Y$, and to bias, when $X$ is a protected attribute. Then one way to pass this litmus test would be to show that a model developer can increase the utility of their model by decreasing its sensitivity to distribution shifts related to the patterns conveyed by this explainability method. Another way to pass this litmus test would be to show that an auditor can increase the utility of the auditing process by more reliably determining if a model is biased by using this explainability method.

[1] Note that many of these approaches could be implemented either by-design (*i.e.,* built into the model's structure) or post-hoc. For this thesis, this distinction is largely immaterial.

### 1.1.3 What do we mean by "rigorously?"

We mean carefully and directly showing that an explainability method actually produces the increased utility that is part of the litmus test to determine if it is useful for a specific application. Showing this can be difficult (*e.g.,* because it may require running a user study), but is the strongest evidence of the usefulness of that method [20, 31]. As a result, a common pitfall is relying on an intuitive connection between this increased utility and general arguments such as: doing better on some proxy metric, having some desired property, or showing that users report higher levels of trust in or understanding of the model. Instead, the goal is to empirically support these connections, so that future work may build on them [20, 31].

For example, once again consider the tennis racket example and suppose that the model developer has retrained the model to make it less sensitive to distribution shifts that change the portion of images with people in them. Then, the question is: did retraining the model work? One way to try to answer this question would be to re-run the explainability method that helped the model developer find this pattern in the first place and check whether it still says that the retrained model is still using this pattern. However, that is not a direct evaluation because it is based on the explainability method. Instead, the model developer should directly check if the model is less sensitive to this type of distribution shifts by checking if the retrained model has smaller performance gaps between images with and without people in them.

### 1.1.4 How do we answer this question?

We apply an application-based pipeline for explainability (explained in Section 1.2) to the different applications that we study in this thesis (summarized in Section 1.3). In doing so, we provide a template demonstrating how to shift explainability's focus away from providing some abstract notion of understanding a model and toward being a set of tools that are useful for specific applications.

## 1.2 An Application-based Pipeline for Explainability

We follow an application-based pipeline for explainability, initially described by Doshi-Velez and Kim [31] and later refined by Chen et al. [20]. At a high level, this pipeline proceeds by defining the application, selecting methods for the application, and designing a series of increasingly complex and realistic evaluations based on the application. In more detail, each of its five steps are as follows:

1. *Define the Application* with enough specificity such that we produce a clear and testable hypothesis about what notion of utility will increase if we are successful. It is important for this step to be explicit because it informs many of the choices in the later steps.
2. *Select Methods* based on the application and specify how they could be useful for it. Additionally, we consider baselines from work outside of explainability on the application.
3. Define *Faithfulness Evaluations*, which are algorithmic evaluations that are based on formal definitions of explainability (*e.g.,* doing well on some proxy metric or having some desired property). These are chosen based more on the explainability method than the application.

4. Define *Simulation Evaluations*, which are algorithmic and application specific evaluations where we simplify the application in order to distill success into an algorithmically measurable quantity. For example, we may simplify the application by: controlling the model's behavior to give us ground-truth to compare against, assuming that the user will use an explanation in a specific way, or even training an algorithmic agent to act as a substitute for a user (as done by Chen et al. [19]).
5. Define *Usefulness Evaluations*, which are the closest practical evaluations of an explainability method's usefulness for the application. These evaluations usually require a user and provide the strongest evidence for a method's utility.

Note that the order of this presentation is a little backwards: the Usefulness Evaluation should be defined first and then the Simulation Evaluation should be designed by simplifying the Usefulness Evaluation in order to make it algorithmic.

**How do these evaluation steps fit together?** Ultimately, our goal is to help practitioners find methods that are useful for their applications and help researchers design useful methods. To do this, we want to identify relationships between more easily measured quantities (*e.g.,* the results of the Faithfulness Evaluations, the dimensionality of the data, the complexity of the model, the type of explanation, the expertise of the users) and the results of the Usefulness Evaluations [20, 31]. However, this has proven to be challenging, in part because the Usefulness Evaluations can be difficult to run and to reproduce. To address these challenges, we use the Simulation Evaluations as a potentially noisy way to approximate the Usefulness Evaluations.

## 1.3 Applications Studied

To support our claim that we can use the application-based pipeline described in the previous section to answer the thesis question, we demonstrate how to apply that pipeline to several different applications of explainability. The next four subsections explain how we instantiate each of the five steps of this pipeline for the four different applications that we study in this thesis. Table 1.1 provides a high level summary of this instantiation.

### 1.3.1 User Interaction

**1. Define the Application.** One of explainability's original motivating applications is helping end-users interact with a model that is making predictions about them. For example, we might want to help a person change their loan application so that the bank will give them a loan when they re-apply. In this context, a good explainability method will help the user find such a change more quickly or help them find a change that is more preferable to make. Specifically, we allow the user to iteratively modify the initial point in order to change the model's prediction by a specific amount and want them to finish this process in as few iterations as possible. We study this application for regression and classification models trained on tabular datasets.

**2. Select Methods.** Local approximation explanations or local counterfactual explanations are a natural fit because they both tell the user how the model's prediction will change for the type of small perturbations of the initial point that we allow users to make. We focus on local approximations because they offer the user more flexibility in their choices.

**Table 1.1:** Each of the steps in the application-based pipeline that we follow and a summary of how we apply that pipeline to the different applications that we study in this thesis.

| | 1. Define the Application | 2. Select Methods | 3. Faithfulness Evaluation | 4. Simulation Evaluation | 5. Usefulness Evaluation |
|---|---|---|---|---|---|
| User Interaction [72] | Help end-users change a model's prediction about them | Local Approximations | Fidelity, Stability | Algorithmic agent | Human subjects |
| Knowledge Discovery [73] | Find which features a dimensionality reduction algorithm is using to separate clusters | Global Counterfactuals | Correctness, Coverage | Synthetic/ Modified Datasets | Real Datasets |
| Spurious Patterns [75] | Find and fix Spurious Patterns where the model relies on the presence of one object to detect a different object | Global Counterfactuals | Probability prediction changes | Artificially induced Spurious Patterns | Naturally occurring Spurious Patterns |
| Blindspots [74] | Find Blindspots where the model performs worse on a semantically meaningful subset of the data | Bespoke Methods | [Skipped, Past work used: Blindspot Size and Error Rate] | Artificially induced Blindspots | Naturally occurring Blindspots |

**3. Faithfulness Evaluation.** We use the two standard proxy metrics for this type of explanation: Fidelity, which is the approximation error of the explanation, and Stability, which is the explanation's sensitivity to small perturbations of the initial point. Intuitively, an explanation with poor Fidelity or Stability will not be useful for this application because will not accurately convey how the model's prediction will change for perturbations of the initial point.

**4. Simulation Evaluation.** We approximate a user with a semi-random greedy algorithmic agent that, at each iteration, changes the feature that the explanation says will change the model's prediction by the amount nearest to the specified amount.

**5. Usefulness Evaluation.** We simply replace the algorithmic agent from the Simulation Evaluation with a human subject.

### 1.3.2 Knowledge Discovery

**1. Define the Application.** Another motivating application is helping domain-experts use a model to discover new knowledge. For example, a computational biologist might want to understand which genes differentiate two different types of cells. In this context, a good explainability method will identify specific genes for the biologist to test. We study this application for unsupervised clustering tasks where the clusters are defined in a 2D representation.

**2. Select Methods.** A natural way to determine how a model differentiates a target cluster from an initial cluster is to find a counterfactual transformation that, when applied to any point from the initial cluster, makes the model treat the transformed point as if it were part of the target cluster. To do this, we introduced a new type of explanation: Global Counterfactual Explanations. In this context, the counterfactual transformation used by a Global Counterfactual Explanation tells the computational biologist how to change a small set of gene expressions in order to change the cell's type from the initial type into the target type. Without this explanation, a computational biologist would typically find a similar change by looking at the difference between the average expression of each gene, so we use this as a baseline against which to compare.

**3. Faithfulness Evaluation.** We define two proxy metrics: Correctness, which measures the probability that this counterfactual transformation moves a point from the initial cluster into the target cluster, and Coverage, which measures the probability that a point from the target cluster is close to a transformed point from the initial cluster. Intuitively, these metrics respectively measure the degree to which the explanation has found the correct differences between two clusters and all of the differences between those clusters.

**4. Simulation Evaluation.** We set up two different types of experiments where we know the ground-truth differences between the clusters and can measure whether an explanation captures them: (1) a synthetic dataset, where we generate a number of causal variables that define the clusters and some correlated variables that depend on those causal variables, and (2) a set of modified real datasets, where we add a modified copy of a cluster to the dataset.

**5. Usefulness Evaluation.** For unaltered real datasets, we compare the explanations to domain knowledge.

### 1.3.3 Spurious Patterns

**1. Define the Application.** An increasingly popular application is helping model developers debug their models. One way to do this is to find and fix Spurious Patterns, which are patterns a model uses that do not align with domain knowledge and will not generalize. For example, consider the Spurious Patterns where a model relies on the presence of a person to detect a tennis racket. In this context, a model developer can benefit by decreasing the model's sensitivity to distribution shifts. We study this application for image classification models with the assumption that we can manipulate the presence of "binary features" (*e.g.,* the presence of objects) in an image.

**2. Select Methods.** We define a type of Global Counterfactual Explanation that tells the model developer "the model's prediction for class $Y$ changes $p\%$ of the time when we remove feature $X$ from images that are from class $Y$." Then, by finding the values of $X$ and $Y$ where $p$ is the largest, the model developer can find the strongest patterns that the model is relying on and use their domain knowledge to determine whether or not those patterns are spurious. Separately, we choose methods to use to fix the model by decreasing its sensitivity to distribution shift by preventing it from relying on the spurious patterns.

**3. Faithfulness Evaluation.** We use the aforementioned probability, $p\%$, as a proxy metric to identify the strongest patterns that the model is relying on.

**4. Simulation Evaluation.** We artificially induce Spurious Patterns of varying strengths by inducing correlations between objects in the model's training data and verify the connection between this proxy metric and the strength of the induced Spurious Pattern. Further, we compare several methods for fixing these Spurious Patterns in terms of how they affect a model's performance and sensitivity to distribution shift.

**5. Usefulness Evaluation.** We find Spurious Patterns in a normally trained model and measure how fixing them affects the model's performance and sensitivity to distribution shift.

### 1.3.4 Blindspots

**1. Define the Application.** A more general way to debug a model is to find and fix systemic errors that occur when the model performs worse on a semantically meaningful subset of the data. For example, when a model relies on the presence of a person to detect a tennis racket, it will have a lower recall on images without people than with people and a higher false positive rate on images with people than without people. Specifically, we focus on Blindspot Discovery, which is the problem of finding systemic errors (which we call Blindspots in this context) in image classification models while making minimal assumptions (*e.g.,* not assuming access to metadata to help define semantically meaningful subsets of the data).

**2. Select Methods.** Because this problem is broader than debugging via Spurious Patterns (*e.g.,* it contains a larger set of bugs), we limit the scope of this work to finding Blindspots using algorithmic methods designed specifically for this problem.[2]

**3. Faithfulness Evaluation.** Past works have used metrics such as the Size or Error Rate of a hypothesized Blindspot as proxy metrics. However, there are two problems with these metrics. First, they do not capture whether the images from a hypothesized Blindspot are semantically similar. Second, even if those images are similar, they may not be representative of the model's performance on other similar images. Consequently, we focus on Simulation Evaluation.

**4. Simulation Evaluation.** By mislabeling specific sets of points in the training data, we artificially induce Blindspots in the model to use as a ground-truth to compare the hypothesized Blindspots against. We do this for both synthetic and real datasets.

**5. Usefulness Evaluation.** We look at naturally occurring Blindspots and illustrate an evaluation approach that is based on "collecting new data matching a text description of a hypothesized Blindspot." This approach addresses the two aforementioned issues with past Faithfulness Evaluations and would allow us to evaluate arbitrary methods.

## 1.4 Contributions

In order to demonstrate the effectiveness of the application-based pipeline for explainability (Section 1.2) we apply it to several different applications (Section 1.3). In the process of doing

[2] Note that there is reasonable evidence that existing explainability methods will not be effective for finding Blindspots [3, 45] and that, once we have found Blindspots, we can fix them with algorithms such as GDRO [83].

so, we make contributions towards each of those applications in terms of new explainability methods and evaluations, which we will now summarize.

**User Interaction.** Our main contributions, based on the work in [72], are:

- We introduce a black-box regularizer that allows us to incentivize the model to have higher Fidelity explanations. This allows us to easily quantify and control this specific instantiation of the accuracy-explainability trade-off.
- We demonstrate that Fidelity is a useful metric for this application by showing that both an algorithmic agent and people perform better on it when an explanation has higher Fidelity.

**Knowledge Discovery.** Our main contributions, based on the work in [73], are:

- The introduction of Global Counterfactual Explanations.
- An algorithm for finding Global Counterfactual Explanations for clustering problems.

**Spurious Patterns.** Our main contributions, based on the work in [75], are:

- We introduce a framework for finding and fixing Spurious Patterns in image classifiers that is based on Global Counterfactual Explanations. We show that this framework finds new types of Spurious Patterns and is more effective at fixing those Spurious Patterns than prior work.
- We introduce new metrics for evaluating the extent to which a model relies on a Spurious Pattern that capture the model's sensitivity to related distribution shifts.

**Blindspots.** Our main contributions, built on the initial work in [74], are:

- A new Blindspot Discovery Method, which we find to be more effective than existing methods. This method uses a 2D image representation, which suggests that a human-in-the-loop approach based on a similar representation may be effective.
- An evaluation framework for Blindspot Discovery Methods. Using this framework, we identify factors that influence the performance of these methods and identify ways for future work to improve these methods.

**Other Contributions.** Besides the primary contributions towards each of the applications presented in this thesis, there are other related research contributions from other projects that the author has contributed towards, which are summarized next. We describe the application-based pipeline followed in this thesis in more detail [20] and, in particular, study Simulation Evaluations where we replace the user in a user study with a machine learned model [19]. For the local approximation explanations used for the User Interaction application, we introduce a new explanation method [71], and we identify a theoretical connection between a variation of Fidelity and model generalization [58]. For saliency map explanations, which could be useful for the Spurious Pattern applications, we introduce a new evaluation framework, the results of which suggest that current methods are unlikely to be useful for this application [45].

## 1.5 Reading Guide

The main chapters of this thesis discuss each of the applications that we study in order to demonstrate the effectiveness of the application-based pipeline described in Section 1.2. Specifically, Chapter 2 looks at User Interaction, Chapter 3 looks at Knowledge Discovery, Chapter 4 looks at Spurious Patterns, and Chapter 5 looks at Blindspots. These chapters can be read in any order, because they are entirely self-contained (*i.e.,* they have their own notation, background, and related work).

In Chapter 6, we conclude by summarizing the goal, approach, and contributions of this thesis and discussing a few relevant directions for future work.

# Part I

# Applications

# Chapter 2

# User Interaction

Recall that the goal of this application is to help end-users interact with a model that is making predictions about them.[1] In particular, we allow the user to iteratively modify their feature vector in order to change the model's prediction about them. Then, in this context, a more useful method will help the user to complete this task with fewer modifications.

To help the user complete this task, we explain the model to them by approximating it with a simple function across some neighborhood centered around their feature vector. Then, we measure the quality of these explanations using metrics such as fidelity and stability. Both these explanations and these metrics are described in the first part of Section 2.2. The key idea in this work is that we can regularize a model to have higher fidelity explanations (Section 2.3) and, empirically, find that this does not reduce the model's accuracy (Section 2.4).

To determine whether higher fidelity explanations are more useful for this application, we run two evaluations. For the first, we have an algorithmic agent complete this task and, for the second, we have real people complete this task. For both of these evaluations, we find that it takes fewer modifications to complete this task for the fidelity-regularized model than for the normally trained model, which suggests that higher fidelity explanations are more useful for this application (Section 2.5).

## 2.1 Introduction

Complex learning-based systems are increasingly shaping our daily lives. To monitor and understand these systems, we require clear explanations of model behavior. Although model interpretability has many definitions and is often application specific [61], local explanations are a popular and powerful tool [78] and will be the focus of this work.

Recent techniques in interpretable machine learning range from models that are interpretable *by-design* [*e.g.,*, 16, 107] to model-agnostic *post-hoc* systems for explaining black-box models such as ensembles and deep neural networks [*e.g.,*, 44, 57, 62, 78, 85]. Despite the variety of technical approaches, the underlying goal of these methods is to develop an interpretable predictive system that produces two outputs: a prediction and its explanation.

[1] This application is also called "actionable recourse" [81, 105].

Both by-design and post-hoc approaches have limitations. On the one hand, by-design approaches are restricted to working with model families that are inherently interpretable, potentially at the cost of accuracy. On the other hand, post-hoc approaches applied to an arbitrary model usually offer no recourse if their explanations are not of suitable quality. Moreover, recent methods that claim to overcome this apparent trade-off between prediction accuracy and explanation quality are in fact by-design approaches that impose constraints on the model families they consider [*e.g.,*, 5, 6, 71].

In this work, we propose a strategy called Explanation-based Optimization (ExpO) that allows us to interpolate between these two paradigms by adding an *interpretability regularizer* to the loss function used to train the model. ExpO uses regularizers based on the *fidelity* [71, 78] or *stability* [6] metrics. See Section 2.2 for definitions.

Unlike by-design approaches, ExpO places no explicit constraints on the model family because its regularizers are differentiable and model agnostic. Unlike post-hoc approaches, ExpO allows us to control the relative importance of predictive accuracy and explanation quality. In Figure 2.1, we see an example of how ExpO allows us to interpolate between these paradigms and overcome their respective weaknesses.

Although fidelity and stability are standard proxy metrics, they are only indirect measurements of the usefulness of an explanation. To more rigorously test the usefulness of ExpO, we additionally devise a more realistic evaluation task where humans are asked to use explanations to change a model's prediction. Notably, our user study falls under the category of Human-Grounded Metric evaluations as defined by Doshi-Velez and Kim [31].

The main contributions of our work are as follows:

1. **Interpretability regularizer.** We introduce, ExpO-Fidelity, a *differentiable* and *model agnostic* regularizer that requires *no domain knowledge* to define. It approximates the fidelity metric on the training points in order to improve the quality of post-hoc explanations of the model.

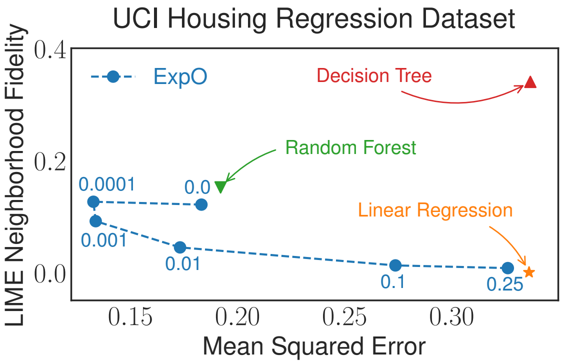


**Figure 2.1:** Neighborhood Fidelity of LIME-generated explanations (lower is better) vs. predictive Mean Squared Error of several models trained on the UCI 'housing' regression dataset. The values in blue denote the regularization weight of ExpO. One of the key contributions of ExpO is allowing us to pick where we are along the accuracy-interpretability curve for a black-box model.

2. **Empirical results.** We compare models trained with and without ExpO on a variety of regression and classification tasks.[2] Empirically, ExpO slightly improves test accuracy and significantly improves explanation quality on test points, producing at least a 25% improvement in terms of explanation fidelity. This separates it from many other methods which trade-off between predictive accuracy and explanation quality. These results also demonstrate that ExpO's effects *generalize* from the training data to unseen points.
3. **User study.** To more directly test the usefulness of ExpO, we run a user study where participants complete a simplified version of a realistic task. Quantitatively, ExpO makes it easier for users to complete the task and, qualitatively, they prefer using the ExpO-regularized model. This is additional validation that the fidelity and stability metrics are useful proxies for interpretability.

### 2.2 Background and Related Work

Consider a supervised learning problem where the goal is to learn a model, $f : \mathcal{X} \mapsto \mathcal{Y}$, $f \in \mathcal{F}$, that maps input feature vectors, $x \in \mathcal{X}$, to targets, $y \in \mathcal{Y}$, trained with data, $\{x_i, y_i\}_{i=1}^N$. If $\mathcal{F}$ is complex, we can understand the behavior of $f$ in some neighborhood, $N_x \in \mathcal{P}[\mathcal{X}]$ where $\mathcal{P}[\mathcal{X}]$ is the space of probability distributions over $\mathcal{X}$, by generating a *local explanation*.

We denote systems that produce local explanations (*i.e.,*, *explainers*) as $e : \mathcal{X} \times \mathcal{F} \mapsto \mathcal{E}$, where $\mathcal{E}$ is the set of possible explanations. The choice of $\mathcal{E}$ generally depends on whether or not $\mathcal{X}$ consists of semantic features. We call features *semantic* if users can reason about them and understand what changes in their values mean (*e.g.,*, a person's income or the concentration of a chemical). Non-semantic features lack an inherent interpretation, with images as a canonical example. We primarily focus on semantic features but we briefly consider non-semantic features in Appendix A.8.

We next state the goal of local explanations for semantic features, define fidelity and stability (the metrics most commonly used to quantitatively evaluate the quality of these explanations), and briefly summarize the post-hoc explainers whose explanations we will use for evaluation.

**Goal of Approximation-based Local Explanations.** For semantic features, we focus on local explanations that try to predict how the model's output would change if the input were perturbed such as LIME [78] and MAPLE [71]. Thus, we can define the output space of the explainer as $\mathcal{E}_s := \{g \in \mathcal{G} \mid g : \mathcal{X} \mapsto \mathcal{Y}\}$, where $\mathcal{G}$ is a class of interpretable functions. As is common, we assume that $\mathcal{G}$ is the set of linear functions.

**Fidelity metric.** For semantic features, a natural choice for evaluation is to measure how accurately $g$ models $f$ in a neighborhood $N_x$ [71, 78]:

$$F(f, g, N_x) := \mathbb{E}_{x' \sim N_x}[(g(x') - f(x'))^2], \tag{2.1}$$

which we refer to as the *neighborhood-fidelity* (NF) metric. This metric is sometimes evaluated with $N_x$ as a point mass on $x$ and we call this version the *point-fidelity* (PF) metric.[3] Intuitively,

[2] `https://github.com/GDPlumb/ExpO`

[3] Although Plumb et al. [71] argued that point-fidelity can be misleading because it does not measure generalization of $e(x, f)$ across $N_x$, it has been used for evaluation in prior work [78, 79]. We report it in our experiments along with the neighborhood-fidelity for completeness.

**Table 2.1:** A breakdown of how EXPO compares to existing methods. Note that EXPO is the only method that is differentiable and model agnostic that does not require domain knowledge.

| Method | Differentiable | Model Agnostic | Domain Knowledge | Goal | Type |
|---|---|---|---|---|---|
| ExpO | Yes | Yes | No | Quality | Neighborhood |
| FTSD | No | Yes | Sometimes | Quality | Neighborhood |
| SENN | Yes | No | No | Quality | Gradient |
| RRR | Yes | Yes | Yes | Content | Gradient |

an explanation with good fidelity (lower is better) accurately conveys which patterns the model used to make this prediction (*i.e.*,, how each feature influences the model's prediction around this point).

**Stability metric.** In addition to fidelity, we are interested in the degree to which the explanation changes between points in $N_x$, which we measure using the *stability metric* [6]:

$$\mathcal{S}(f, e, N_x) := \mathbb{E}_{x' \sim N_x}[||e(x, f) - e(x', f)||_2^2] \tag{2.2}$$

Intuitively, more stable explanations (lower is better) tend to be more trustworthy [6, 7, 35].

**Post-hoc explainers.** Various explainers have been proposed to generate local explanations of the form $g : \mathcal{X} \mapsto \mathcal{Y}$. In particular, LIME [78], one of the most popular post-hoc explanation systems, solves the following optimization problem:

$$e(x, f) := \arg\min_{g \in \mathcal{E}_s} F(f, g, N_x) + \Omega(g), \tag{2.3}$$

where $\Omega(g)$ stands for an additive regularizer that encourages certain desirable properties of the explanations (*e.g.*,, sparsity). Along with LIME, we consider another explanation system, MAPLE [71]. Unlike LIME, its neighborhoods are learned from the data using a tree ensemble rather than specified as a parameter.

### Related Methods

There are three methods that consider problems conceptually similar to EXPO: Functional Transparency for Structured Data (FTSD) [55, 56], Self-Explaining Neural Networks (SENN) [6], and Right for the Right Reasons (RRR) [82]. In this section, we contrast EXPO to these methods along several dimensions: whether the proposed regularizers are (i) differentiable, (ii) model agnostic, and (iii) require domain knowledge; whether the (iv) goal is to change an explanation's quality (*e.g.*,, fidelity or stability) or its content (*e.g.*,, how each feature is used); and whether the expected explainer is (v) neighborhood-based (*e.g.*,, LIME or MAPLE) or gradient-based (*e.g.*,, Saliency Maps [91]). These comparisons are summarized in Table 2.1, and we elaborate further on them in the following paragraphs.

FTSD has a very similar high-level objective to EXPO: it regularizes black-box models to be more locally interpretable. However, it focuses on graph and time-series data and is not well-defined for general tabular data. Consequently, our technical approaches are distinct. First, FTSD's local neighborhood and regularizer definitions are different from ours. For graph data, FTSD aims to understand what the model would predict if the graph itself were modified. Although this is the same type of local interpretability considered by EXPO, FTSD requires domain knowledge to define $N_x$ in order to consider plausible variations of the input graph. These definitions do not apply to general tabular data. For time-series data, FTSD aims to understand what the model will predict for the next point in the series and defines $N_x$ as a windowed-slice of the series to do so. This has no analogue for general tabular data and is thus entirely distinct from EXPO. Second, FTSD's regularizers are non-differentiable, and thus it requires a more complex, less efficient bi-level optimization scheme to train the model.

SENN is a by-design approach that optimizes the model to produce stable explanations. For both its regularizer and its explainer, it assumes that the model has a specific structure. In Appendix A.1, we show empirically that EXPO is a more flexible solution than SENN via two results. First, we show that we can train a significantly more accurate model with EXPO-FIDELITY than with SENN; although the EXPO-regularized model is slightly less interpretable. Second, if we increase the weight of the EXPO-FIDELITY regularizer so that the resulting model is as accurate as SENN, we show that the EXPO-regularized model is much more interpretable.

RRR also regularizes a black-box model with a regularizer that involves a model's explanations. However, it is motivated by a fundamentally different goal and necessarily relies on extensive domain knowledge. Instead of focusing on explanation quality, RRR aims to restrict what features are used by the model itself, which will be reflected in the model's explanations. This relies on a user's domain knowledge to specify sets of good or bad features. In a similar vein to RRR, there are a variety of methods that aim to change the model in order to align the content of its explanations with some kind of domain knowledge [32, 80, 109]. As a result, these works are orthogonal approaches to EXPO.

Finally, we briefly mention two additional lines of work that are also in some sense related to EXPO. First, [76] proposed a method for local linearization in the context of adversarial robustness. Because its regularizer is based on the model's gradient, it will have the same issues with flexibility, fidelity, and stability discussed in Appendix A.2. Second, there is a line of work that regularizes black-box models to be easier to approximate by decision trees. [113] does this from a global perspective while [114] uses domain knowledge to divide the input space into several regions. However, small decision trees are difficult to explain locally by explainer's such as LIME (as seen in Figure 2.1) and so these methods do not solve the same problem as EXPO.

### Connections to Function Approximations and Complexity

The goal of this section is to intuitively connect local linear explanations and neighborhood fidelity with classical notions of function approximation and complexity/smoothness, while also highlighting key differences in the context of local interpretability. First, neighborhood-based local linear explanations and first-order Taylor approximations both aim to use linear functions to locally approximate $f$. However, the Taylor approximation is strictly a function of $f$ and $x$ and cannot be adjusted to different neighborhood scales for $N_x$, which can lead to poor fidelity and stability. Second, Neighborhood Fidelity (NF), the Lipschitz Constant (LC), and Total Variation (TV) all approximately measure the smoothness of $f$ across $N_x$. However, a large LC or TV does not necessarily indicate that $f$ is difficult to explain across $N_x$ (*e.g.*,, consider a linear model with large coefficients which has a near zero NF but has a large LC/TV). Instead, local interpretability is more closely related to the LC or TV of the part of $f$ that cannot be explained by $e(x, f)$ across $N_x$. Additionally, we empirically show that standard $l_1$ or $l_2$ regularization techniques do not influence model interpretability. Examples and details for all of these observations are in Appendix A.2.

### 2.3 Explanation-based Optimization

Recall that the main limitation of using post-hoc explainers on arbitrary models is that their explanation quality can be unpredictable. To address this limitation, we define regularizers that can be added to the loss function and used to train an arbitrary model. This allows us to control for explanation quality without making explicit constraints on the model family in the way that by-design approaches do. Specifically, we want to solve the following optimization problem:

$$\hat{f} := \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} (\mathcal{L}(f, x_i, y_i) + \gamma \mathcal{R}(f, N_{x_i}^{\text{reg}})) \tag{2.4}$$

where $\mathcal{L}(f, x_i, y_i)$ is a standard predictive loss (*e.g.*,, squared error for regression or cross-entropy for classification), $\mathcal{R}(f, N_{x_i}^{\text{reg}})$ is a regularizer that encourages $f$ to be interpretable in the neighborhood of $x_i$, and $\gamma > 0$ controls the regularization strength. Because our regularizers are differentiable, we can solve Equation 2.4 using any standard gradient-based algorithm; we use SGD with Adam [47].

We define $\mathcal{R}(f, N_x^{\text{reg}})$ based on either neighborhood-fidelity, Eq. (2.1), or stability, Eq. (2.2). In order to compute these metrics exactly, we would need to run $e$; this may be non-differentiable or too computationally expensive to use as a regularizer. As a result, EXPO consists of two main approximations to these metrics: EXPO-FIDELITY and EXPO-STABILITY. EXPO-FIDELITY approximates $e$ using a local linear model fit on points sampled from $N_x^{reg}$ (Algorithm 1). Note that it is simple to modify this algorithm to regularize for the fidelity of a *sparse* explanation. EXPO-STABILITY encourages the model to not vary too much across $N_x^{reg}$ and is detailed in Appendix A.8.

**Computational cost.** The overhead of using EXPO-FIDELITY comes from using Algorithm 1 to calculate the additional loss term and then differentiating through it at each iteration. If $x$ is $d$-dimensional and we sample $m$ points from $N_x^{reg}$, this has a complexity of $O(d^3 + d^2 m)$ plus the cost to evaluate $f$ on $m$ points. Note that $m$ must be at least $d$ in order for this loss to be non-zero, thus making the complexity $\Omega(d^3)$. Consequently, we introduce a randomized version of Algorithm 1, EXPO-1D-FIDELITY, that randomly selects one dimension of $x$ to perturb according to $N_x^{reg}$ and penalizes the error of a local linear model along that dimension.

This variation has a complexity of $O(m)$ plus the cost to evaluate $f$ on $m$ points, and allows us to use a smaller $m$.[4]

## 2.4 Experimental Results

In our main experiments, we demonstrate the effectiveness of EXPO-FIDELITY and EXPO-1D-FIDELITY on datasets with semantic features using seven regression problems from the UCI collection [27], the 'MSD' dataset[5], and 'Support2' which is an in-hospital mortality classification problem[6]. Dataset statistics are in Table 2.2.

We found that EXPO-regularized models are more interpretable than normally trained models because post-hoc explainers produce quantitatively better explanations for them; further, they are often more accurate. Additionally, we qualitatively demonstrate that post-hoc explanations of EXPO-regularized models tend to be simpler. In Appendix A.8, we demonstrate the effectiveness of EXPO-STABILITY for creating Saliency Maps [91] on MNIST [54].

**Experimental setup.** We compare EXPO-regularized models to normally trained models (labeled "None"). We report model accuracy and three interpretability metrics: Point-Fidelity (PF), Neighborhood-Fidelity (NF), and Stability (S). The interpretability metrics are evaluated for two black-box explanation systems: LIME and MAPLE. For example, the "MAPLE-PF" label corresponds to the Point-Fidelity Metric for explanations produced by MAPLE. All of these metrics are calculated on test data, which enables us to evaluate whether optimizing for explanation fidelity on the training data generalizes to unseen points.

All of the inputs to the model are standardized to have mean zero and variance one (including the response variable for regression problems). The network architectures and hyper-parameters are chosen using a grid search; for more details see Appendix A.3. For the final results, we set $N_x$ to be $\mathcal{N}(x, \sigma)$ with $\sigma = 0.1$ and $N_x^{reg}$ to be $\mathcal{N}(x, \sigma)$ with $\sigma = 0.5$. In Appendix A.4, we discuss how we chose those distributions.

**Regression experiments.** Table 2.4 shows the effects of EXPO-FIDELITY and EXPO-1D-FIDELITY on model accuracy and interpretability. EXPO-FIDELITY frequently improves the interpretability metrics by over 50%; the smallest improvements are around 25%. Further, it lowers the prediction error on the 'communities', 'day', and 'MSD' datasets, while achieving similar accuracy on the rest. EXPO-1D-FIDELITY also significantly improves the interpretability metrics, although on average to a lesser extent than EXPO-FIDELITY does, and it has no significant effect on accuracy on average.

**A qualitative example on the UCI 'housing' dataset.** After sampling a random point $x$, we use LIME to generate an explanation at $x$ for a normally trained model and an EXPO-regularized model. Table 2.3 shows the example we discuss next. Quantitatively, training the model with EXPO-1D-FIDELITY decreases the LIME-NF metric from 1.15 to 0.02 (*i.e.*,, EXPO produces a model that is more accurately approximated by the explanation around $x$).

[4] Each model takes less than a few minutes to train on an Intel 8700k CPU, so computational cost was not a limiting factor in our experiments. That being said, we observe a 2x speedup per iteration when using EXPO-1D-FIDELITY compared to EXPO-FIDELITY on the 'MSD' dataset and expect greater speedups on higher dimensional datasets.

[5] As in [11], we treat the 'MSD' dataset as a regression problem with the goal of predicting the release year of a song.

[6] `http://biostat.mc.vanderbilt.edu/wiki/Main/SupportDesc`.

**Algorithm 1** Neighborhood-fidelity regularizer

```
input $f_\theta$, $x$, $N_x^{\text{reg}}$, $m$
    Sample points:
    $x'_1, \ldots, x'_m \sim N_x^{\text{reg}}$
    Compute predictions:
    $\hat{y}_j(\theta) = f_\theta(x'_j)$ for $j = 1, \ldots, m$
    Produce a local linear explanation:
    $\beta_x(\theta) = \arg\min_\beta \sum_{j=1}^{m} (\hat{y}_j(\theta) - \beta^\top x'_j)^2$
output $\frac{1}{m} \sum_{j=1}^{m} (\hat{y}_j(\theta) - \beta_x(\theta)^\top x'_j)^2$
```

**Table 2.2:** Statistics of the datasets.

| **Dataset** | **# samples** | **# dims** |
|---|---:|---:|
| autompgs | 392 | 7 |
| communities | 1993 | 102 |
| day | 731 | 14 |
| housing | 506 | 11 |
| music | 1059 | 69 |
| winequality-red | 1599 | 11 |
| MSD | 515345 | 90 |
| SUPPORT2 | 9104 | 51 |

**Table 2.3:** An example of LIME's explanation for a normally trained model ("None") and an ExpO-regularized model. Because these are linear explanations, each value can be interpreted as an estimate of how much the model's prediction would change if that feature's value were increased by one. Because the explanation for the ExpO-regularized model is sparser, it is easier to understand and, because it has better fidelity, these estimates are more accurate.

| Data | | Explanation | |
|---|---:|---:|---:|
| Feature | Value | None | ExpO |
| CRIM | 2.5 | -0.1 | 0.0 |
| INDUS | 1.0 | 0.1 | 0.0 |
| NOX | 0.9 | -0.2 | -0.2 |
| RM | 1.4 | 0.2 | 0.2 |
| AGE | 1.0 | -0.1 | 0.0 |
| DIS | -1.2 | -0.4 | -0.2 |
| RAD | 1.6 | 0.2 | 0.2 |
| TAX | 1.5 | -0.3 | -0.1 |
| PTRATIO | 0.8 | -0.1 | -0.1 |
| B | 0.4 | 0.1 | 0.0 |
| LSTAT | 0.1 | -0.3 | -0.5 |

Further, the resulting explanation also has fewer non-zero coefficients (after rounding), and hence it is simpler because the effect is attributed to fewer features. More examples, that show similar patterns, are in Appendix A.5.

**Medical classification experiment.** We use the 'support2' dataset to predict in-hospital mortality. Since the output layer of our models is the softmax over logits for two classes, we run each explainer on each of the logits. We observe that ExpO-Fidelity had no effect on accuracy and improved the interpretability metrics by 50% or more, while ExpO-1D-Fidelity slightly decreased accuracy and improved the interpretability metrics by at least 25%. See Table A.5 in Appendix A.6 for details.

### 2.5 User Study

The previous section compared ExpO-regularized models to normally trained models through quantitative metrics such as model accuracy and post-hoc explanation fidelity and stability on held-out test data. Doshi-Velez and Kim [31] describe these metrics as Functionally-Grounded Evaluations, which are useful *proxies* for more direct applications of interpretability. To more directly measure the usefulness of ExpO, we conduct a user study to obtain Human-Grounded Metrics [31], where real people solve a simplified task.

**Table 2.4:** Normally trained models ("None") vs. the same models trained with EXPO-FIDELITY or EXPO-1D-FIDELITY on the regression datasets. Results are shown across 20 trials (with the standard error in parenthesis). Statistically significant differences ($p = 0.05$, t-test) between FIDELITY and None are in bold and between 1D-FIDELITY and None are underlined. Because MAPLE is slow on 'MSD', we evaluate interpretability using LIME on 1000 test points.

| Metric | Regularizer | autompgs | communities | day† ($10^{-3}$) | housing | music | winequality.red | MSD |
|---|---|---|---|---|---|---|---|---|
| **MSE** | None | 0.14 (0.03) | 0.49 (0.05) | 1.000 (0.300) | 0.14 (0.05) | 0.72 (0.09) | 0.65 (0.06) | 0.583 (0.018) |
| | FIDELITY | 0.13 (0.02) | **0.46 (0.03)** | **0.002 (0.002)** | 0.15 (0.05) | 0.67 (0.09) | 0.64 (0.06) | **0.557 (0.0162)** |
| | 1D-FIDELITY | 0.13 (0.02) | 0.55 (0.04) | 5.800 (8.800) | 0.15 (0.07) | 0.74 (0.07) | 0.66 (0.06) | 0.548 (0.0154) |
| **LIME-PF** | None | 0.040 (0.011) | 0.100 (0.013) | 1.200 (0.370) | 0.14 (0.036) | 0.110 (0.037) | 0.0330 (0.0130) | 0.116 (0.0181) |
| | FIDELITY | **0.011 (0.003)** | **0.080 (0.007)** | **0.041 (0.007)** | **0.057 (0.017)** | **0.066 (0.011)** | **0.0025 (0.0006)** | **0.0293 (0.00709)** |
| | 1D-FIDELITY | 0.029 (0.007) | 0.079 (0.026) | 0.980 (0.380) | 0.064 (0.017) | 0.080 (0.039) | 0.0029 (0.0011) | 0.057 (0.0079) |
| **LIME-NF** | None | 0.041 (0.012) | 0.110 (0.012) | 1.20 (0.36) | 0.140 (0.037) | 0.112 (0.037) | 0.0330 (0.0140) | 0.117 (0.0178) |
| | FIDELITY | **0.011 (0.003)** | **0.079 (0.007)** | **0.04 (0.07)** | **0.057 (0.018)** | **0.066 (0.011)** | **0.0025 (0.0006)** | **0.029 (0.007)** |
| | 1D-FIDELITY | 0.029 (0.007) | 0.080 (0.027) | 1.00 (0.39) | 0.064 (0.017) | 0.080 (0.039) | 0.0029 (0.0011) | 0.0575 (0.0079) |
| **LIME-S** | None | 0.0011 (0.0006) | 0.022 (0.003) | 0.150 (0.021) | 0.0047 (0.0012) | 0.0110 (0.0046) | 0.00130 (0.00057) | 0.0368 (0.00759) |
| | FIDELITY | **0.0001 (0.0003)** | **0.005 (0.001)** | **0.004 (0.004)** | **0.0012 (0.0002)** | **0.0023 (0.0004)** | **0.00007 (0.00002)** | **0.00171 (0.00034)** |
| | 1D-FIDELITY | 0.0008 (0.0003) | 0.018 (0.008) | 0.100 (0.047) | 0.0025 (0.0007) | 0.0084 (0.0052) | 0.00016 (0.00005) | 0.0125 (0.00291) |
| **MAPLE-PF** | None | 0.0160 (0.0088) | 0.16 (0.02) | 1.0000 (0.3000) | 0.057 (0.024) | 0.17 (0.06) | 0.0130 (0.0078) | — |
| | FIDELITY | **0.0014 (0.0006)** | **0.13 (0.01)** | **0.0002 (0.0003)** | **0.028 (0.013)** | 0.14 (0.03) | **0.0027 (0.0010)** | — |
| | 1D-FIDELITY | 0.0076 (0.0038) | 0.092 (0.03) | 0.7600 (0.3000) | 0.027 (0.012) | 0.13 (0.05) | 0.0016 (0.0007) | — |
| **MAPLE-NF** | None | 0.0180 (0.0097) | 0.31 (0.04) | 1.2000 (0.3200) | 0.066 (0.024) | 0.18 (0.07) | 0.0130 (0.0079) | — |
| | FIDELITY | **0.0015 (0.0006)** | **0.24 (0.05)** | **0.0003 (0.0004)** | **0.033 (0.014)** | **0.14 (0.03)** | **0.0028 (0.0010)** | — |
| | 1D-FIDELITY | 0.0084 (0.0040) | 0.16 (0.05) | 0.9400 (0.3600) | 0.032 (0.013) | 0.14 (0.06) | 0.0017 (0.0008) | — |
| **MAPLE-S** | None | 0.0150 (0.0099) | 1.2 (0.2) | 0.0003 (0.0008) | 0.18 (0.14) | 0.08 (0.06) | 0.0043 (0.0020) | — |
| | FIDELITY | **0.0017 (0.0005)** | **0.8 (0.4)** | 0.0004 (0.0004) | **0.10 (0.08)** | **0.05 (0.02)** | **0.0009 (0.0004)** | — |
| | 1D-FIDELITY | 0.0077 (0.0051) | 0.6 (0.2) | 1.2000 (0.6600) | 0.09 (0.06) | 0.04 (0.02) | 0.0004 (0.0002) | — |

†The relationship between inputs and targets on the 'day' dataset is very close to linear and hence all errors are orders of magnitude smaller than across other datasets.

In summary, the results of our user study show that the participants had an easier time completing this task with the EXPO-regularized model and found the explanations for that model more useful. See Table 2.5 and Figure A.6 in Appendix A.7 for details. Not only is this additional evidence that the fidelity and stability metrics are good proxies for interpretability, but it also shows that they remain so after we directly optimize for them. Next, we describe the high-level task, explain the design choices of our study, and present its quantitative and qualitative results.

**Defining the task.** One of the common proposed use cases for local explanations is as follows. A user is dissatisfied with the prediction that a model has made about them, so they request an explanation for that prediction. Then, they use that explanation to determine what changes they should make in order to receive the desired outcome in the future. We propose a similar task on the UCI 'housing' regression dataset where the goal is to increase the model's prediction by a fixed amount.

We simplify the task in three ways. First, we assume that all changes are equally practical to make; this eliminates the need for any prior domain knowledge. Second, we restrict participants to changing a single feature at a time by a fixed amount; this reduces the complexity of the required mental math. Third, we allow participants to iteratively modify the features while getting new explanations at each point; this provides a natural quantitative measure of explanation usefulness, via the number of changes required to complete the task.

**Design Decisions**. Figure 2.2 shows a snapshot of the interface we provide to participants. Additionally, we provide a demo video of the user study in the Github repository. Next, we describe several key design aspects of our user study, all motivated by the underlying goal of isolating the effect of EXPO.

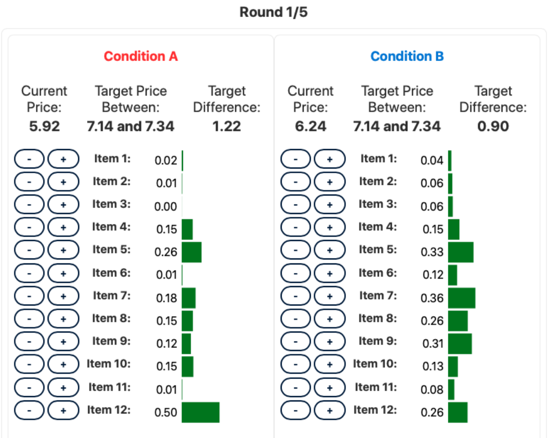
1. **Side-by-side conditions**. We present the two conditions side-by-side with the same initial point. This design choice allows the participants to directly compare the two conditions and allows us to gather their preferences between the conditions. It also controls for the fact that a model may be more difficult to explain for some $x$ than for others. Notably, while both conditions have the same initial point, each condition is modified independently. With the conditions shown side-by-side, it may be possible for a participant to use the information gained by solving one condition first to help solve the other condition. To prevent this from biasing our aggregated results, we randomize, on a per-participant basis, which model is shown as Condition A.
2. **Abstracted feature names and magnitudes**. In the explanations shown to users, we abstract feature names and only show the magnitude of each feature's expected impact. Feature names are abstracted in order to prevent participants from using prior knowledge to inform their decisions. Moreover, by only showing feature magnitudes, we eliminate double negatives (*e.g.,*, decreasing a feature with a negative effect on the prediction should increase the prediction), thus simplifying participants' required mental computations. In other words, we simplify the interface so that the '+' button is expected to increase the prediction by the amount shown in the explanation regardless of explanation's (hidden) sign.
3. **Learning Effects**. To minimize long-term learning (*e.g.,*, to avoid learning general patterns such as 'Item 7's explanation is generally unreliable.'), participants are limited to completing a single experiment consisting of five recorded rounds. In Figure A.5 from Appendix A.7, we show that the distribution of the number of steps it takes to complete each round across participants does not change substantially. This result indicates that learning effects were not significant.
4. **Algorithmic Agent**. Although the study is designed to isolate the effect EXPO has on the usefulness of the explanations, entirely isolating its effect is impossible with human participants. Consequently, we also evaluate the performance of an algorithmic agent that uses a simple heuristic that relies only on the explanations. See Appendix A.7 for details.

**Collection Procedure.** We collect the following information using Amazon Mechanical Turk:

1. **Quantitative.** We measure how many steps (*i.e.,*, feature changes) it takes each participant to reach the target price range for each round and each condition.
2. **Qualitative Preferences.** We ask which condition's explanations are more *useful* for completing the task and better match their *expectation* of how the price should change.[7]
3. **Free Response Feedback.** We ask why participants preferred one condition over the other.

**Data Cleaning.** Most participants complete each round in between 5 and 20 steps. However, there are a small number of rounds that take 100's of steps to complete, which we hypothesize to be random clicking. See Figure A.4 in Appendix A.7 for the exact distribution. As a result, we remove any participant who has a round that is in the top $1\%$ for number of steps taken (60

[7] Note that we do not ask participants to rate their trust in the model because of issues such as those raised in Lakkaraju and Bastani [52].



Round 1/5

**Condition A**

| Current Price: | Target Price Between: | Target Difference: |
|---|---|---|
| 5.92 | 7.14 and 7.34 | 1.22 |

| | |
|---|---|
| Item 1: | 0.02 |
| Item 2: | 0.01 |
| Item 3: | 0.00 |
| Item 4: | 0.15 |
| Item 5: | 0.26 |
| Item 6: | 0.01 |
| Item 7: | 0.18 |
| Item 8: | 0.15 |
| Item 9: | 0.12 |
| Item 10: | 0.15 |
| Item 11: | 0.01 |
| Item 12: | 0.50 |

**Condition B**

| Current Price: | Target Price Between: | Target Difference: |
|---|---|---|
| 6.24 | 7.14 and 7.34 | 0.90 |

| | |
|---|---|
| Item 1: | 0.04 |
| Item 2: | 0.06 |
| Item 3: | 0.06 |
| Item 4: | 0.15 |
| Item 5: | 0.33 |
| Item 6: | 0.12 |
| Item 7: | 0.36 |
| Item 8: | 0.26 |
| Item 9: | 0.31 |
| Item 10: | 0.13 |
| Item 11: | 0.08 |
| Item 12: | 0.26 |

**Figure 2.2:** An example of the interface participants were given. In this example, the participant has taken one step for Condition A and no steps for Condition B. While the participant selected '+' for Item 7 in Condition A, the change had the opposite effect and decreased the price because of the explanation's low fidelity.

**Table 2.5:** The results of the user study. Participants took significantly fewer steps to complete the task using the ExpO-regularized model, and thought that the post-hoc explanations for it were both more useful for completing the task and better matched their expectations of how the model would change.

| Condition | Steps | Usefulness | Expectation |
|---|---|---|---|
| **None** | 11.45 | 11 | 11 |
| **ExpO** | 8.00 | 28 | 26 |
| **No Preference** | *N.A.* | 15 | 17 |

participants with 5 rounds per participant and 2 conditions per round gives us 600 observed rounds). This leaves us with a total of 54 of the original 60 participants.

**Results.** Table 2.5 shows that the ExpO-regularized model has both quantitatively and qualitatively more useful explanations. Quantitatively, participants take 8.00 steps on average with the ExpO-regularized model, compared to 11.45 steps for the normally trained model ($p = 0.001$, t-test). The participants report that the explanations for the ExpO-regularized model are both more useful for completing the task ($p = 0.012$, chi-squared test) and better aligned with their expectation of how the model would change ($p = 0.042$, chi-squared test). Figure A.6 in Appendix A.7 shows that the algorithmic agent also finds the task easier to complete with the ExpO-regularized model. This agent relies solely on the information in the explanations and thus is additional validation of these results.

**Participant Feedback.** Most participants who prefer the ExpO-regularized model focus on how well the explanation's predicted change matches the actual change. For example, one participant says "It [ExpO ] seemed to do what I expected more often" and another notes that "In Condition A [None] the predictions seemed completely unrelated to how the price actually changed." Although some participants who prefer the normally trained model cite similar reasons, most focus on how quickly they can reach the goal rather than the quality of the explanation. For example, one participant plainly states that they prefer the normally trained model because "The higher the value the easier to hit [the] goal"; another participant similarly explains that "It made the task easier to achieve." These participants likely benefited from the randomness of the low-fidelity explanations of the normally trained model, which can jump unexpectedly into the target range.

## 2.6 Conclusion

In this work, we regularize black-box models to be more interpretable with respect to the fidelity and stability metrics for local explanations. We compare EXPO-FIDELITY, a model agnostic and differentiable regularizer that requires no domain knowledge to define, to classical approaches for function approximation and smoothing. Next, we demonstrate that EXPO-FIDELITY slightly improves model accuracy and significantly improves the interpretability metrics across a variety of problem settings and explainers on unseen test data. Finally, we run a user study demonstrating that an improvement in fidelity and stability improves the usefulness of the model's explanations.

# Chapter 3

# Knowledge Discovery

Recall that the goal of this application is to identify the patterns that a model has learned from the data, in the hope that these patterns lead to new knowledge. In particular, we are interested in finding the differences between clusters defined in a learned low-dimensional representation of the data. To do this, we use global counterfactual explanations and measure the quality of these explanations using correctness and coverage (Section 3.3).

To evaluate the usefulness of these explanations for this application, we run a series of evaluations where we progressively trade-off knowledge of the ground-truth differences between clusters in order to consider more realistic settings (Section 3.4.2):

- We generate synthetic data with a known causal structure. Then, we check whether the explanation identifies that structure.
- We run a quantitative analysis of modified versions of UCI datasets. To do this, we take a cluster from one of those datasets and add a modified copy of it back into the dataset. Then, we check whether the explanation identifies the modifications that we made to that cluster.
- We run qualitative analysis of UCI datasets using the labels. To do this, we compare the explanations to domain knowledge based on those labels. Note that, from an explainability perspective, the results of this analysis need to be interpreted with some caution because we need to make sure that the explanations still accurately explain the model, even when they match our domain knowledge.

### 3.1 Introduction

A common workflow in data exploration is to: 1) learn a low-dimensional representation of the data, 2) identify groups of points (*i.e.,* clusters) that are similar to each other in that representation, and 3) examine the differences between the groups to determine what they represent. We focus on the third step of this process: answering the question *"What are the key differences between the groups?"*

For data exploration, this is an interesting question because the groups often correspond to an unobserved concept of interest and, by identifying which features differentiate the groups, we can learn something about that concept of interest. For example, consider single-cell RNA analysis. These datasets measure the expression levels of many genes for sampled cells. Usually the cell-type of each of those cells is unknown. Because gene expression and cell-type are closely related, the groups of points that can be seen in a low-dimensional representation of the dataset often correspond to different cell-types (Figure 3.1). By determining which gene expressions differentiate the groups, we can learn something about the connection between gene expression and cell-type.

One common approach for answering this question is manual interpretation. One simple way to do this, that we will use as a naive baseline, is to calculate the Difference Between the Mean (DBM) value of each feature in the original input space between a pair of groups. For example, consider using DBM to explain the differences between the cells in Group 3 and Group 17 from Figure 3.1. In this case, DBM's explanation contains many hundreds of non-zero elements, which is far too many to be understood by a person (Figure 3.2). If we make DBM's explanation sparse by thresholding it to include only the $k$ largest changes, it is no longer an effective explanation because it no longer reliably maps points from Group 3 to Group 17 (Figure 3.3). More generally, manual interpretation can be time-consuming and typically ad-hoc, requiring the analyst to make arbitrary decisions that may not be supported by the data.

Another, more principled, method is statistical hypothesis testing for the differences between features across groups [86]. However, the trade-off between the power of these tests and their false positive rate becomes problematic in high-dimensional settings.

Both manual interpretation and statistical testing have an additional key shortcoming: they do not make use of the model that learned the low dimensional representation that was used to define the groups in the first place. Intuitively, we would expect that, by inspecting this model directly, we should be able to gain additional insight into the patterns that define the groups. With this perspective, answering our question of interest becomes an interpretable machine learning problem. Although there are a wide variety of methods developed in this area [16, 62, 78, 79, 107, 121], none of them are designed to answer our question of interest. See Section 3.2 for further discussion.

To answer our question of interest, we want a *counterfactual* explanation because our goal is to identify the key differences between Group A and Group B using the low-dimensional representation and the most natural way to do this is to find a transformation that causes the

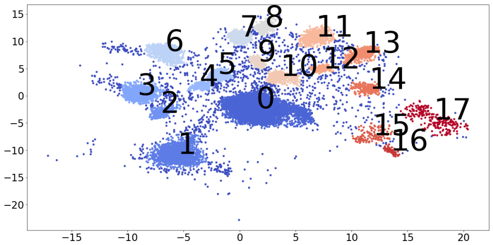


**Figure 3.1:** A representation learned for a single-cell RNA sequence dataset using the model from [30]. Previous work on this dataset showed that these groups of cells correspond to different cell-types [88]. The goal of a GCE is to use this representation to identify the changes in gene expression that are associated with a change of cell type.

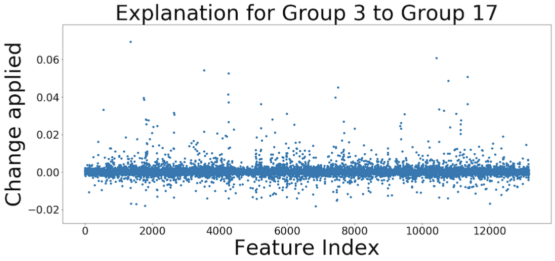


**Figure 3.2:** DBM's explanation for the difference in gene expression between the cells in Group 3 and Group 17. The x-axis shows which feature index (gene expression) is being changed and the y-axis shows by how much. Because it is very high dimensional and not sparse, it is difficult to use DBM to determine what the key differences actually are between this pair of groups (*i.e.,* which genes differentiate these cell-types).

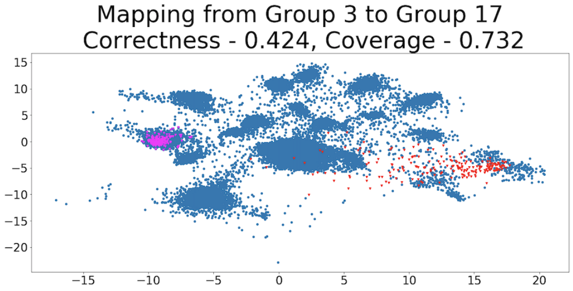


**Figure 3.3:** By thresholding DBM's explanation for the differences between Group 3 and Group 17 to include only the $k$ largest changes (250 in this case), we can make it sparse enough to be human interpretable. However, this simplified explanation is no longer an effective explanation. We show this visually: the magenta points are the representations of points sampled randomly from Group 3 and the red points are the representations of those points after the explanation was applied to them. We can see that the red points are usually not in Group 17 (poor correctness) and that they do not cover much of Group 17 (poor coverage). These metrics will be defined in Section 3.3.

model to assign transformed points from Group A to Group B. Additionally, we want a *global* explanation because we want to find a explanation that works for all of the points in Group A and because we want the complete set of explanations to be consistent (*i.e.,* symmetrical and transitive) among all the groups. See Section 3.3.2 for further discussion of our definition of consistency in this context. Hence, our goal is to find a Global Counterfactual Explanation (GCE). Although the space of possible transformations is very large, we consider translations in this work because their interpretability can be easily measured using sparsity.

**Contributions:** To the best of our knowledge, this is the first work that explores GCEs. Motivated by the desire to generate a simple (*i.e.,* sparse) explanation between each pair of groups, we derive an algorithm to find these explanations that is motivated by compressed sensing [15, 104]. However, the solutions from compressed sensing are only able to explain the differences between one pair of groups. As a result, we generalize the compressed sensing solution to find a set of consistent explanations among all groups simultaneously. We call this algorithm Transitive Global Translations (TGT).

We demonstrate the usefulness of TGT with a series of experiments on synthetic, UCI, and single-cell RNA datasets. In our experiments, we measure the effectiveness of explanations using correctness and coverage, with sparsity as a proxy metric for interpretability, and we compare the patterns the explanations find to those we expect to be in the data. We find the TGT clearly outperforms DBM at producing sparse explanations of the model and that its explanations match domain knowledge.[1]

[1] Code for all algorithms and experiments is available at https://github.com/GDPlumb/ELDR

## 3.2 Related Work

Most of the literature on cluster analysis focuses on defining the clusters; the interpretation methods discussed in that literature are primarily manual inspection/visualization or statistical testing [41]. Consequently, the focus of our related work will be on interpretable machine learning. Although interpretability is often loosely defined and context specific [61], we categorize existing methods along two axes in order to demonstrate how a GCE differs from them. Those axes are the explanation's *level* and its *form*.

The first axis used to categorize explanations is their level: local or global. A *local* explanation explains a single prediction made by the model [62, 71, 78]. Kauffmann et al. [43] studies a problem closely related to ours of explaining why a point was assigned to its cluster/group. A *global* explanation will explain multiple predictions or the entire model at once [16, 79, 107].

The second axis we use to categorize explanations is their form: feature attribution, approximation, or counterfactual. A *feature attribution* explanation assigns a value measuring how each feature contributed to the model's prediction(s) [62, 99]. Importantly, it is necessary to define a baseline value for the features in order to compute these explanations. An *approximation* explanation approximates the model being explained using a function that is simple enough to be considered directly interpretable (*e.g.,* a sparse linear model or a small decision tree) across some neighborhood, which could be centered around a point or could be the entire input space [16, 71, 78, 79, 107]. A *counterfactual* explanation finds a transformation of the input(s) such that the transformed version of the input is treated in a specific way by the model [28, 29, 37, 121].

For various reasons, it would be challenging to use other types of explanations to construct a GCE. Local explanations would have to be aggregated in order to produce an explanation that applies to a group of points and it would be nontrivial to ensure that the resulting "aggregated group explanations" are consistent (*i.e.,* symmetrical and transitive). For feature attribution and local approximation explanations, it is difficult to guarantee that the baseline value or neighborhood they consider is defined broadly enough to find the transformation we want. For global approximation explanations, we might not be able to approximate a complex model well enough to find the transformation we want because of the accuracy-interpretability trade-off that stems from the complexity constraint on the explanation model [61]. For a concrete example of these difficulties, see the Appendix B.1. This example uses Integrated Gradients [99] which is a local feature attribution method that produces symmetrical and transitive explanations with respect to a single class.

## 3.3 Global Counterfactual Explanations

We will start by introducing our notation, more formally stating the goal of a GCE, and defining the metrics that we will use to measure the quality of GCEs. We do this under the simplifying assumption that we have only two groups of points that we are interested in. Then, in Section 3.3.1, we will demonstrate the connection between finding a GCE and compressed sensing. We use that connection to derive a loss function we can minimize to find a GCE between a single pair of groups of points. Finally, in Section 3.3.2, we will remove our simplifying assumption and introduce our algorithm, TGT, for finding a set of consistent GCEs among multiple groups of points.

**Notation:** Let $r : \mathbb{R}^d \rightarrow \mathbb{R}^m$ denote the function that maps the points in the feature space into a lower-dimensional representation space. The only restriction that we place on $r$ is that it is differentiable (see the Appendix B.2 for more discussion on $r$). Suppose that we have two regions of interest in this representation: $R_{initial}, R_{target} \subset \mathbb{R}^m$. Let $X_{initial}$ and $X_{target}$ denote their pre-images. Then, our goal is to find the key differences between $X_{initial}$ and $X_{target}$ in $\mathbb{R}^d$ and, unlike manual interpretation or statistical testing, we will treat this as an interpretable machine learning problem by using $r$ to help find those key differences.

**Defining the Goal of GCEs:** At a high level, the goal of a GCE is to find a transformation that takes the points in $X_{initial}$ and transforms them so that they are mapped to $R_{target}$ by $r$; in other words, $r$ treats the transformed points from $X_{initial}$ as if they were points from $X_{target}$. Formally, the goal is to find a transformation function $t : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that:

$$r(t(x)) \in R_{target} \; \forall x \in X_{initial} \tag{3.1}$$

Because we are using $t$ as an explanation, it should be as simple as possible. Since they are very simple and their complexity can be readily measured by their sparsity, we limit $t$ to a translation:

$$t(x) = x + \delta \tag{3.2}$$

**Measuring the Quality of GCEs:** To measure the quality of a GCE we use two metrics: *correctness* and *coverage*. Correctness measures the fraction of points mapped from $X_{initial}$ into $R_{target}$. Coverage measures the fraction of points in $R_{target}$ that transformed points from $X_{initial}$ are similar to. Mathematically, we define correctness as:

$$cr(t) = \frac{1}{|X_{initial}|} \sum_{x \in X_{initial}} \mathbb{1}[\exists x' \in X_{target} \mid ||r(t(x)) - r(x')||_2^2 \leq \epsilon] \tag{3.3}$$

And coverage as:

$$cv(t) = \frac{1}{|X_{target}|} \sum_{x \in X_{target}} \mathbb{1}[\exists x' \in X_{initial} \mid ||r(x) - r(t(x'))||_2^2 \leq \epsilon] \tag{3.4}$$

Clearly, correctness is a necessary property because an explanation with poor correctness has failed to map points from $X_{initial}$ into $R_{target}$ (Equation 3.1). However, coverage is also a desirable property because, intuitively, an explanation with good coverage has captured all of the differences between the groups.

Defining these metrics requires that we pick a value of $\epsilon$.[2] Observe that, if $X_{initial} = X_{target}$ and $t(x) = x$, then $cr(t) = cv(t)$ and we have a measure of how similar a group of points is to itself. After $r$ has been learned, we increase $\epsilon$ until this self-similarity metric for each group of points in the learned representation is between 0.95 and 1.

[2] We use an indicator based on $l_2$ distance and the points themselves for two reasons. First, it is necessary for the definition of coverage. Second, it makes it easier to define $R_{target}$ for representations that are more than two-dimensional.

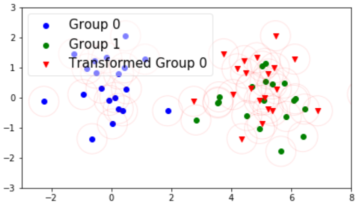


**Figure 3.4:** Two groups of points and the transformed version of Group 0. The red circles indicate the balls of radius $\epsilon$ used to calculated the metrics. Observe that the translation has both good correctness and coverage.

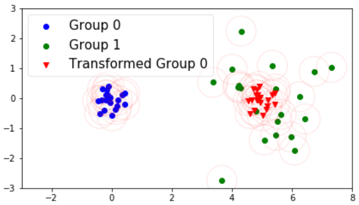


**Figure 3.5:** The same idea as Figure 3.4, but now Group 0 has a smaller variance than Group 1. Observe that the translation has good correctness but poor coverage.

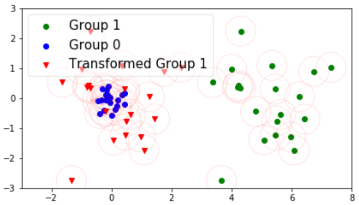


**Figure 3.6:** The same setup as in Figure 3.5, but showing what happens when the negative of the translation is applied to Group 1. Observe that it has good coverage but poor correctness.

**A Simple Illustration:** We will now conclude our introduction to GCEs with a simple example to visualize the transformation function and the metrics. In Figures 3.4, 3.5, and 3.6, the data is generated from two Gaussian distributions with different means and $r(x) = x$. We use DBM between Group 1 and Group 0 to define the translation/explanation. In Figure 3.4, the two distributions have an equal variance and, as a result, the translation is an effective explanation with good correctness and coverage. In Figures 3.5 and 3.6, Group 0 has a smaller variance than Group 1. Because a simple translation cannot capture that information[3], the translation has poor coverage from Group 0 to Group 1 while its negative has poor correctness from Group 1 to Group 0. This illustrates the connection between correctness and coverage that we will discuss more in Section 3.3.2.

### 3.3.1 Relating GCEs and Compressed Sensing

We will now demonstrate how the problem of finding a GCE between a pair of groups is connected to compressed sensing. We start with an "ideal" loss function that is too difficult to optimize and then make several relaxations to it.

In principle, Equations 3.1, 3.3, or 3.4 define objective functions that we could optimize, but they are discontinuous and hence difficult to optimize. To progress towards a tractable objective, first consider a continuous approximation of correctness (Equation 3.3):

$$loss_{min}(t) = \frac{1}{|X_{initial}|} \sum_{x \in X_{initial}} \min_{x' \in X_{target}} ||r(t(x)) - r(x')||_2^2 \tag{3.5}$$

This loss could be optimized by gradient descent using auto-differentiation software, although doing so might be difficult because of the $min$ operation. Consequently, we consider a simplified version of it:

$$loss_{mean}(t) = \frac{1}{|X_{initial}|} \sum_{x \in X_{initial}} ||r(t(x)) - \bar{r}_{target}||_2^2 \tag{3.6}$$

[3] This is true regardless of how the translation is found (*e.g.*, TGT or DBM). So this example also motivates the usefulness of more complex transformation functions.

where $\bar{r}_i = \frac{1}{|X_i|} \sum\limits_{x \in X_i} r(x)$. [4]

Next, we are going to make use of our restriction of $t$ to a translation, $\delta$, and assume, for now, that $r$ is a linear mapping: $r(x) = Ax$ for $A \in \mathbb{R}^{m \times d}$. Although this assumption about $r$ will not be true in general, it allows us to connect a GCE to the classical compressed sensing formulation. Under these constraints, we have that:

$$loss_{mean+linear}(\delta) = \frac{1}{|X_{initial}|} \sum_{x \in X_{initial}} ||A(x + \delta) - \bar{r}_{target}||_2^2 \tag{3.7}$$

By setting its derivative to zero, we find that solving $A\delta = \bar{r}_{target} - \bar{r}_{initial}$ yields an optimal solution to Equation 3.7. Observe that this is an undetermined linear system, because $m < d$, and so we can always find such a $\delta$.

Recall that, in general, we want to find a sparse $\delta$. This is partially because we need the explanation to be simple enough to be understood by a person, but also because the explanation is hopefully modeling real world phenomena and these phenomena often have nice structure (*e.g.,* sparsity). Then, because we are finding a GCE by solving $A\delta = \bar{r}_{target} - \bar{r}_{initial}$[5] , we can see that $\bar{r}_{target} - \bar{r}_{initial}$ is the model's low-dimensional measurement that we are trying to reconstruct using the high-dimensional but sparse underlying signal, $\delta$. This is exactly the classical compressed sensing problem formulation. As a result, we will add $l_1$ regularization to $\delta$ as an additional component to the loss function as is done in both linear and non-linear compressed sensing [12].

Consequently, we could consider finding a GCE by minimizing the linear compressed sensing loss:

$$loss_{cs}(\delta) = ||A(\bar{x}_{initial} + \delta) - \bar{r}_{target}||_2^2 + \lambda||\delta||_1 \tag{3.8}$$

where $\bar{x}_i = \frac{1}{|X_i|} \sum\limits_{x \in X_i} x$. Or, by removing the assumption that $r(x) = Ax$, minimizing the more general version that we use for our experiments:

$$loss(\delta) = ||r(\bar{x}_{initial} + \delta) - \bar{r}_{target}||_2^2 + \lambda||\delta||_1 \tag{3.9}$$

### 3.3.2 Computing GCEs

We have thus far limited ourselves to explaining the differences between $l = 2$ groups of points labeled as $X_0$ ("initial") and $X_1$ ("target"), and focused on learning an explanation $t_{0 \to 1}$ from $X_0$ to $X_1$. However, this is not a realistic setting because this labeling was arbitrary and because we usually have $l > 2$.

Unfortunately, the naive solution of using compressed sensing to independently produce explanations for all O($l^2$) pairs of groups will fail to satisfy two desirable properties related

[4] Note that $loss_{min}$ clearly optimizes for correctness and it does not directly penalize coverage. However, because $loss_{mean}$ encourages $r(t(x))$ to be close to $\bar{r}_{target}$, it is optimizing for correctness at the expense of coverage. In Section 3.3.2, we will discuss why this is not as harmful as it seems in our setting where $t$ is a symmetrical translation.

[5] Note that DBM solves this but does not consider sparsity.

to the internal consistency of the explanations. For instance, we would like $t_{0\to 1}$ to agree with $t_{1\to 0}$, a property we call *symmetry*. Additionally, we would like $t_{0\to 2}$ to agree with the combined explanations $t_{0\to 1}$ and $t_{1\to 2}$; we call this *transitivity*. Formally, *symmetry* requires that $t_{i\to j} = t_{j\to i}^{-1}$ and *transitivity* requires that $t_{i\to k} = t_{j\to k} \circ t_{i\to j}$ for any $j$.

Our approach to finding a consistent (*i.e.,* symmetrical and transitive) set of explanations is to enforce the consistency constraints by-design. We do this by computing a set of explanations relative to a *reference group*. We assume that $X_0$ is the reference group and find a set of basis explanations $t_1, \ldots, t_{l-1}$, where $t_i = t_{0\to i}$. We then use this set of basis explanations to construct the explanation between any pair of the groups of points.[6] Algorithm 3 describes how $t_{i\to j}$ can be constructed from $t_1, \ldots, t_{l-1}$.

**Overview of TGT.** We now have all of the pieces necessary to actually compute a GCE: a differentiable loss function to measure the quality of $t_{i\to j}$, $l_1$ regularization to help us find the simplest possible explanation between $X_i$ and $X_j$, and a problem setup to ensure that our explanations are consistent across $X_0, \ldots, X_{l-1}$. At a high level, TGT will proceed to sample random "initial" and "target" groups from the set of all groups, construct that explanation from the set of basis explanations (Algorithm 3), and then use Equation 3.9 as a loss function to use to update the explanation using gradient descent (Algorithm 4). Pseudo-code for this process is in Algorithm 2.[7] The main hyper-parameter that requires tuning is the strength of the $l_1$ regularization, $\lambda$.

**Why can we prioritize correctness in Equation 3.9?** In the previous subsection, we noted that the Equation 3.9 prioritizes correctness over coverage. Because Algorithm 2 randomly chooses the "initial" and "target" groups many times, it updates the basis explanations based on both $cr(\delta_{i\to j})$ and $cr(\delta_{j\to i})$. When $t$ is a translation and the explanations are symmetrical, we can see that $cr(\delta_{j\to i})$ is closely related to $cv(\delta_{i\to j})$. This is because they only differ in whether they add $\delta_{j\to i}$ to a point in $X_j$ or subtract $\delta_{j\to i}$ from a point in $X_i$ in their respective indicator functions. Further, if we consider $r(x) = Ax$, then they are identical metrics (this is consistent with the example from Figure 3.5). Collectively, this means that Algorithm 2 implicitly considers both $cr(\delta_{i\to j})$ and $cv(\delta_{i\to j})$ while computing the explanations.

### 3.3.3 Controlling the Level of Sparsity

Because neither TGT nor DBM is guaranteed to produce a $k$-sparse explanation, we will threshold each of the explanations to include only the $k$ most important features (*i.e.,* the $k$ features with the largest absolute value) for our experiments. This is done after they have been calculated but before their quality has been evaluated. Importantly, TGT has a hyper-parameter, $\lambda$, which roughly controls the sparsity of its explanations; as a result, we will tune $\lambda$ to maximize correctness for each value of $k$.

[6] Importantly, the transitivity constraint also ensures that our choice of how we label the groups or which group is the reference group does not influence the optimal solution of our algorithm.

[7] The pseudo-code leaves out some of the details of the optimization process such as how often we sample new "initial" and "target" groups and how convergence is defined. For those details, see the code on GitHub.

**Algorithm 2** TGT: Calculating GCEs with a Reference Group. Note that, because $\lambda$ is applied to all of the explanations, we cannot tune it to guarantee that each explanation is exactly $k$-sparse.

```
Input: Model: $r$
       Group Means: $\bar{x}_i$ (feature space) and
                    $\bar{r}_i$ (representation space)
                    for $i = 0, \ldots, l-1$
       $l_1$ Regularization Weight: $\lambda$
       Learning Rate: $\alpha$
Initialize: $\delta_1, \ldots, \delta_{l-1}$ to vectors of 0
while not converged do
    Sample $i \neq j$ from $\{0, \ldots, l-1\}$
    Construct $t_{i \to j}$ ($\delta_{i \to j}$) using Algorithm 3
    Calculate objective: $v = loss(\delta_{i \to j})$ using Equation 3.9
    Update the components of $\delta_{i \to j}$ using Algorithm 4
end while
Return: $\delta_1, \ldots, \delta_{l-1}$
```

**Algorithm 3** How to construct any explanation between an arbitrary pair of groups, $t_{i \to j}$, using the set of basis explanations relative to the reference group, $t_1, \ldots, t_{l-1}$

```
Input: $i, j$
if $i == 1$ then
    Return: $t_j$
else if $j == 1$ then
    Return: $t_i^{-1}$
else
    Return: $t_j \circ t_i^{-1}$
end if
```

**Algorithm 4** How to update the basis explanations, $\delta_1, \ldots, \delta_{l-1}$, based on the performance of $\delta_{i \to j}$. This splits the signal from the gradient between the basis explanations used to construct $\delta_{i \to j}$. Note that it does not maintain any fixed level of sparsity.

```
Input: $i, j, \alpha$ (learning rate), $\nabla v$ (gradient of the loss function)
if $i == 1$ then
    $\delta_j = \delta_j - \alpha \nabla v$
else if $j == 1$ then
    $\delta_i = \delta_i + \alpha \nabla v$
else
    $\delta_j = \delta_j - 0.5 \alpha \nabla v$
    $\delta_i = \delta_i + 0.5 \alpha \nabla v$
end if
```

The fact that $\lambda$ is tuned for each value of $k$ raises an interesting question: "Does TGT use a subset of the features from its $k_2$-sparse explanation for its $k_1$-sparse explanation when $k_1 < k_2$?". Naturally, we would like for the answer to be "yes" because, for example, it does not seem like a desirable outcome if a 2-sparse explanation uses Features A and B but a 1-sparse explanation uses Feature C.

Suppose we have two explanations between Group $i$ and Group $j$: $e_1$ which is $k_1$-sparse and $e_2$ which is $k_2$-sparse with $k_1 < k_2$. To address this question, we define the *similarity* of $e_1$ and $e_2$ as:

$$similarity(e_1, e_2) = \frac{\sum |e_1[i]| \mathbb{1}[e_2[i] \neq 0]}{||e_1||_1} \tag{3.10}$$

This metric captures how much of $e_1$'s explanation uses features that were also chosen by $e_2$.[8] So a score of 1 indicates that $e_1$ uses a subset of the features of $e_2$ and a score of 0 indicates that it uses entirely different features. Because DBM does not solve a different optimization problem to achieve each level of sparsity, its similarity measure is always 1. When we run experiments with a list of sparsity levels $k_1, \ldots, k_m$, we will plot $similarity(e_1, e_2), \ldots, similarity(e_{m-1}, e_m)$ to measure how similar TGT's explanations are as the level of sparsity increases.

### 3.4 Experimental Results

Our experimental results are divided into two sections. In the first section, we demonstrate that TGT is better at explaining the model than DBM is when we restrict the explanations to varying degrees of sparsity. In the second section, we move beyond assessing whether or not TGT explains the model and demonstrate that it also appears to capture real signals in the data.

#### 3.4.1 TGT's Efficacy at Explaining the Model

From an interpretable machine learning perspective, our goal is help practitioners understand the dimensionality-reduction models they use in the data exploration process. We measure the quality of GCEs using correctness (Equation 3.3) and coverage (Equation 3.4) at varying degrees of sparsity (Figure 3.7).

We use the model from [30][9]on the UCI Iris, Boston Housing, and Heart Disease datasets [27] and a single-cell RNA dataset [88] and use its a visualization of its two-dimensional representation to define the groups of points. Figure 3.1 shows this representation and grouping for the single-cell RNA dataset; similar plots for all of the datasets are in the Appendix B.3. This model learns a non-linear embedding using a neural network architecture which is trained to be a parametric version of t-SNE [63] that also preserves the global structure in the data [48].

Next, because the acceptable level of complexity depends on both the application and the person using the explanation, we measure the effectiveness of the explanations produced by TGT and DBM at explaining the model across a range of sparsity levels.

**Explanation effectiveness at different levels of sparsity.** Figure 3.7 shows the results of this comparison. We can see that TGT performed at least as well as DBM and usually did better. Further, we can see that TGT's explanations are quite similar to each other as we ask for sparser explanations. Note that all of these metrics are defined for a single pair of groups and so these plots report the average across all pairs of groups.

**Exploring a Specific Level of Sparsity.** Figure 3.7 shows that TGT's performance: is almost as good when $k = 1$ as when $k = 4$ on Iris, drops off sharply for $k < 5$ on Boston Housing, and drops off sharply for $k < 3$ on Heart Disease. Further, on the single-cell RNA dataset, it shows that TGT significantly outperforms DBM when $k = 250$ (Appendix B.4 Figure B.7) and that this comparison becomes more favorable for TGT for smaller $k$. The level of sparsity where the metrics drop off indicates the minimum explanation complexity required for these

[8] Note that this metric also includes the run to run variance of TGT.

[9] We use this model because previous analysis showed that its representation identifies meaningful groups on the single-cell RNA dataset [30].

methods to explain the model. See Figure 3.8 for an example of the pairwise correctness and coverage metrics for these levels of sparsity.

Figure 3.3 shows that DBM does not produce a good 250-sparse explanation for the difference between Group 3 and Group 17 from Figure 3.1. For the sake of an easy comparison, Figure 3.9 shows a similar plot that uses TGT's 250-sparse explanation; it is clearly a much better explanation.

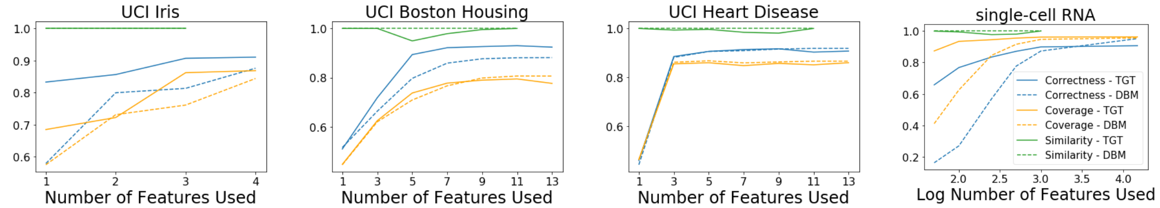


**Figure 3.7:** A comparison of the effectiveness of TGT to DBM at explaining the model (measured by correctness and coverage) at a range of sparsity levels. Note that TGT performs at least as well as DBM and usually does better. Looking at the similarity metric, we see that TGT is fairly consistent at picking a subset of the current features when asked to find an even sparser solution.

### 3.4.2 TGT's Efficacy at Capturing Real Signals in the Data

In the previous section, we demonstrated that TGT provides accurate explanations of the model that learned the low-dimensional representation. However, in practice, there could be a mismatch between what the model itself learns and the true underlying structure in the data. In this section, we evaluate empirically whether or not TGT provides explanations that match underlying patterns in the data.

We begin with an experiment on a synthetic dataset with a known causal structure and demonstrate that TGT correctly identifies this structure. This also serves as an intuitive example of why a sparser explanation can be as effective as a less-sparse explanation. Next, we leverage the labels that come with the UCI datasets to compare TGT's explanations to some basic domain knowledge. Finally, we modify the UCI datasets and demonstrate that TGT is able to identify

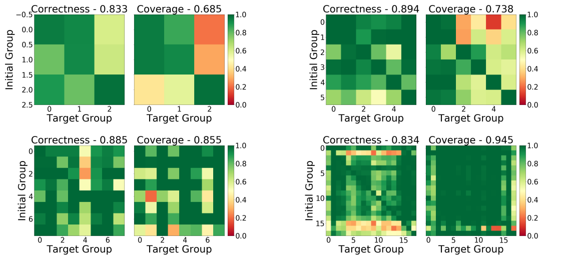


**Figure 3.8:** The pairwise explanation metrics for TGT with: Top Left) 1-sparse explanations on Iris, Top Right) 5-sparse explanations on Boston Housing, Bottom Left) 3-sparse explanations on Heart Disease, and Bottom Right) 250-sparse explanations on single-cell RNA.

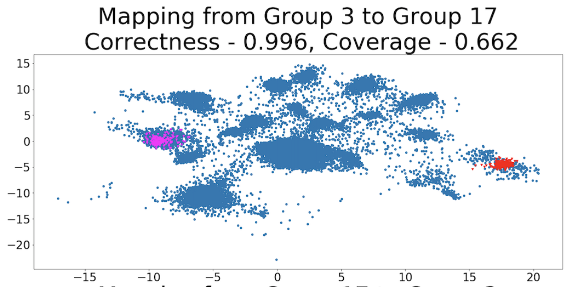


**Figure 3.9:** Unlike DBM, TGT is able to produce an effective 250-sparse explanation for the difference between Group 3 and Group 17 on the single-cell RNA dataset. This can be seen both visually and with the correctness and coverage metrics.

**Table 3.1:** A comparison of the explanations from TGT and from DBM. Note that they are similar except that TGT does not use $x_4$, which is the variable that is not causaly related to the groups.

| Explanation | Method | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|---|
| $0 \to 1$ | TGT | -1.09 | 0.01 | 0.03 | 0.00 |
|  | DBM | -1.02 | 0.04 | 0.01 | -1.03 |
| $0 \to 2$ | TGT | 0.00 | 0.88 | 0.00 | 0.00 |
|  | DBM | -0.01 | 0.97 | 0.06 | -0.03 |
| $0 \to 3$ | TGT | -0.99 | 0.71 | 0.00 | 0.00 |
|  | DBM | -1.02 | 1.03 | -0.01 | -1.03 |

those modifications. Together, these results indicate that TGT is identifying real patterns in the data.

**Synthetic Data with a Known Causal Structure.** By specifying the causal structure of the data, we can know which differences are necessary in the explanation, since they are the casual differences, and which differences are unnecessary, since they are explained by the causal differences. We find that TGT correctly identifies the causal structure of this dataset and that DBM does not.

We use the following procedure to generate each point in our synthetic dataset: $x_1, x_2 \sim Bern(0.5) + \mathcal{N}(0, 0.2)$, $x_3 \sim \mathcal{N}(0, 0.5)$, and $x_4 \sim x_1 + \mathcal{N}(0, 0.05)$. The causal structure of this dataset is simple. *$x_1$ and $x_2$ jointly cause 4 different groups of points.* The explanation for the differences between these groups must include these variables. *$x_3$ is a noise variable* that is unrelated to these groups and, as a result, should not be included in any explanation. *$x_4$ is a variable that is correlated with those groups*, since it is caused by $x_1$, but does not cause those groups. As a result, it is not necessary to include it in any explanation.

We generate a dataset consisting of 400 points created using this process and train an autoencoder [51] to learn a two dimensional representation of the dataset. A visualization of this learned representation is in the Appendix B.3 Figure B.5; as expected, there are four distinct groups of points in it. Then, we use TGT and DBM to calculate the GCEs between these groups. The pairwise and average correctness and coverage metrics for these solutions are in the Appendix B.4 Figures B.8 and B.9; observe that the two methods are equally effective at explaining the model.

When we inspect the explanations ()Table 3.1), we see that both TGT and DBM use $x_1$ and $x_2$, neither use $x_3$, and that DBM uses $x_4$ while TGT does not. This shows that, even in a very simple setting, there is good reason to believe that an explanation that is simpler (*i.e.,* sparser) than the DBM explanation exists and that TGT might be able to find it.

**Qualitative Analysis of the UCI Datasets using the Labels.** Qualitatively, we find that TGT's explanations agree with domain knowledge about these datasets. Specifically: On the Iris dataset, its explanations agree with a simple decision tree because they both rely mostly on the Petal Width to separate the groups. On the Boston Housing dataset, it identifies the differences between a set of inexpensive urban houses vs expensive suburban houses as well as equally priced groups of houses that differ mainly in whether or not they are on the Charles River. Finally on the Heart Disease dataset, it finds that the difference between a moderate and a low risk group of subjects was that the low-risk group's symptoms are explained by something other than heart disease and the difference between the moderate and high risk group of subjects is that the former is made up of men and the later of women. For full details, see the Appendix B.5.

**Quantitative Analysis of Modified Versions of the UCI Datasets.** In order to perform a more quantitative analysis, we artificially add a known signal to the dataset by choosing one of the groups of points, creating a modified copy of it by translating it, and adding those new points back into the dataset. We then ask two important questions about TGT's behavior. First, does TGT correctly identify the modifications we made to the original dataset? Second, do TGT's explanations between the original groups change when the modified group is added to the dataset? Details of how we setup these experiments and their results are in the Appendix B.6.

We find that TGT does identify the modifications we made and that, in doing so, it does not significantly change the explanations between the original groups. Importantly, this result remains true even if we retrain the learned representation on the modified dataset.

These results are a strong indicator that TGT finds real patterns in the data because it recovers both the original signal and the artificial signal even when the algorithm is rerun or the representation is retrained.

### 3.5 Conclusion

In this work, we introduced a new type of explanation, a GCE, which is a counterfactual explanation that applies to an entire group of points rather than a single point. Next, we defined reasonable metrics to measure the quality of GCEs (*i.e.,* correctness and coverage) and introduced the concept of consistency (*i.e.,* symmetry and transitivity), which is an important additional criteria that GCEs must satisfy. Given that, we defined an algorithm for finding consistent GCEs, TGT, that treats each pairwise explanation as a compressed sensing problem. Our first experiments empirically demonstrated that TGT is better able to explain the model than DBM across a range of levels of sparsity. Our next experiments showed that TGT captures real patterns in the data. This was done using a synthetic dataset with a known causal structure and by comparing TGT's explanations to background knowledge about the UCI datasets. As an additional test, we then added a synthetic signal to the UCI datasets and demonstrated that TGT can recover that signal without changing its explanations between the real groups. Importantly, this result remains true even when the representation is retrained.

Although we focused on data exploration in this work, similar groups arise naturally whenever the model being trained uses an encoder-decoder structure. This technique is ubiquitous in most areas where deep learning is common (*e.g.,* semi-supervised learning, image classification, natural language processing, reinforcement learning). In these settings, identifying the key differences between the groups is an interesting question because we can observe that "The model treats points in Group A the same/differently as points in Group B", determine that "The key differences between Group A and Group B are X", and then conclude that "The model does not/does use pattern X to make its decisions". We believe that exploring these applications is an important direction of future work that will require more sophisticated transformation functions, optimization procedures, and definitions of the groups of points of interest.

# Chapter 4

# Spurious Patterns

Recall that the goal of this application is to find and fix spurious patterns in image classification models. To find these patterns, we use global counterfactual explanations that tell us "the model's prediction for class $Y$ changes $p\%$ of the time when we remove feature $X$ from images that are from class $Y$." Then, we use this probability, $p$, to measure the degree to which the model relies on the pattern represented by one of these explanations. Finally, we fix these patterns by retraining the model on a dataset that has been carefully augmented with counterfactual images. This approach is described in Section 4.3.

To evaluate the usefulness of these methods for this application, we run two evaluations. For the first, we deliberately induce spurious patterns in models by subsampling the COCO dataset (Section 4.5.1). For the second, we consider a more realistic setting by finding and fixing spurious patterns in a model trained on the entire COCO dataset (Section 4.5.2).

Beyond this individual work, there are a few general ideas to keep in mind for this application:

- Whether a pattern is "spurious" and whether a spurious pattern is worth fixing both depend on domain knowledge and are ultimately up to the model developer.
- When using counterfactual images, it is important to be aware of the fact that these images are off-distribution, which means that the model may behave differently on them than it does on real images. This influences how methods for finding and fixing spurious patterns work and how we should evaluate those methods. Ideally, all evaluations are done using only real images to prevent this from influencing the results.

### 4.1 Introduction

With the growing adoption of machine learning models, there is a growing concern about *Spurious Patterns* (SPs) – when models rely on patterns that do not align with domain knowledge and do not generalize [80, 82, 89, 93, 102]. For example, a model trained to detect tennis rackets on the COCO dataset [60] learns to rely on the presence of a person, which leads to systemic errors: it is significantly less accurate at detecting tennis rackets for images without people than with people (41.2% vs 86.6%) and only ever has false positives on images with people. Relying on this SP works well on COCO, where the majority of images with tennis rackets also have people, but would not be as effective for other distributions. Further, relying on SPs may also lead to serious concerns when they relate to protected attributes such as race or gender [13].

We focus on SPs where an image classifier is relying on a spurious object (*e.g.,* using people to detect tennis rackets) and we propose Spurious Pattern Identification and REpair (SPIRE)[1] as an end-to-end solution for these SPs. As illustrated in Figure 4.1, SPIRE *identifies* which patterns the model is using by measuring how often it makes different predictions on the original and counterfactual versions of an image. Since it reduces a pattern to a single value that has a clear interpretation, it is easy for a user to, when needed, label that pattern as spurious or valid. Then, it *mitigates* SPs by retraining the model using a novel form of data augmentation that aims to shift the training distribution towards the *balanced distribution*, a distribution where the SP is no longer helpful, while minimizing any new correlations between the label and artifacts in the counterfactual images. Each of these steps is based on the assumption that we have access to pixel-wise object-annotations to use to create counterfactual images by adding or removing objects.

To verify that the baseline model relies on a SP and quantify the impact of mitigation methods, we measure *gaps* in accuracy between images with and without the spurious object (*e.g.,* there is a 45.4% accuracy drop between images of tennis rackets with and without people). Intuitively, the more a model relies on a SP, the larger these gaps will be and the less robust the model is to distribution shift. Consequently, an effective mitigation method will decrease these gap metrics and improve performance on the balanced distribution. Empirically, we show SPIRE's effectiveness with three sets of experiments:

- *Benchmark Experiments.* We induce SPs with varying strengths by sub-sampling COCO in order to observe how mitigation methods work in a controlled setting. Overall, we find that SPIRE is more effective than prior methods. Interestingly, we also find that most prior

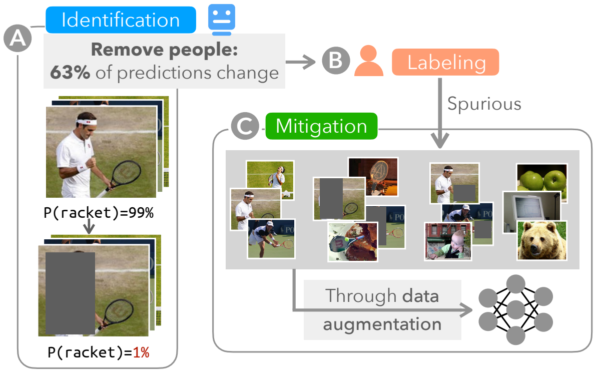


**Figure 4.1:** For the tennis racket example, SPIRE identifies this pattern by observing that, when we remove the people from images with both a tennis racket and a person, the model's prediction changes 63% of the time. Since we do not remove the tennis racket itself, we label this pattern as spurious. Then, SPIRE carefully adds/removes tennis rackets/people from different images to create an augmented training set where tennis rackets and people are independent while minimizing any new correlations between the label and artifacts in the counterfactual images (*e.g.,* grey boxes).

[1] Code is available at `https://github.com/GDPlumb/SPIRE`

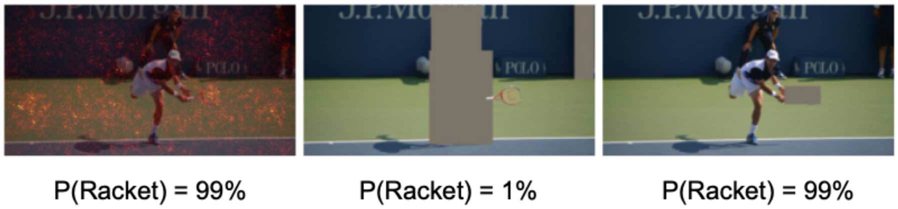


**Figure 4.2:** Based on the saliency map [91] (Left), one might mistakenly infer that the model is not relying on the person. However, the model fails to detect the racket after the person is removed (Center) and incorrectly detects a racket after it is removed (Right).

methods are ineffective at mitigating *negative SPs* (*e.g.,* when the model learns that the presence of a "cat" means that there is no "tie").

- *Full Experiment.* We show that SPIRE is useful "in the wild" on the full COCO dataset. For identification, it finds a diverse set of SPs and is the first method to identify negative SPs, and, for mitigation, it is more effective than prior methods. Additionally, we show that it improves zero-shot generalization (*i.e.,* evaluation without re-training) to two challenging datasets: UnRel [70] and SpatialSense [116]. These results are notable because most methods produce no improvements in terms of robustness to natural distribution shifts [101].
- *Generalization Experiments.* We illustrate how SPIRE generalizes beyond the setting from our prior experiments, where we considered the object-classification task and assumed that the dataset has pixel-wise object-annotations to use to create counterfactual images. Specifically, we explore three examples that consider a different task and/or do not make this assumption.

## 4.2 Related Work

We discuss prior work as it pertains to identifying and mitigating SPs for image classification models.

**Identification.** The most common approach is to use explainable machine learning [28, 37, 42, 50, 78, 80, 82, 85, 91, 92]. For image datasets, these methods rely on local explanations, resulting in a slow process that requires the user to look at the explanation for each image, infer what that explanation is telling them, and then aggregate those inferences to assess whether or not they represent a consistent pattern (Figure 4.2). In addition to this procedural difficulty, there is uncertainty about the usefulness of some of these methods for model debugging [2, 3].

In contrast, SPIRE avoids these challenges by measuring the aggregated effect that a specific counterfactual has on the model's predictions (*e.g.,* the model's prediction changes 63% of the time when we remove the people from images with both a tennis racket and a person). While prior work has defined similar measurements, Shetty et al. [89] and Singh et al. [93] fail to identify negative SPs and Xiao et al. [115] fail to identify specific SPs (*e.g.,* they find that the model is relying on "something" rather than "people").

**Mitigation.** While prior work has explored data augmentation for mitigation [4, 18, 39, 89, 102], it has done so with augmentation strategies that are agnostic to the training distribution (*e.g.,* Shetty et al. [89] simply remove either the tennis rackets or the people, as applicable, uniformly at random for each image). In contrast, SPIRE aims to use counterfactual images to create a training distribution where the label is independent of the spurious object, while minimizing any new correlations between the label and artifacts in the counterfactual images. We hypothesize that this is why we find that SPIRE is more effective than these methods.

Another line of prior work adds regularization to the model training process [39, 59, 80, 82, 93, 102, 108]. Some of these methods specify which parts of the image should not be relevant to the model's prediction [80, 82]. Other methods encourage the model's predictions to be consistent across counterfactual versions of the image [39, 59, 102]. All of these methods could be used in conjunction with SPIRE.

Finally, there are two additional lines of work that make different assumptions than SPIRE. Making weaker assumptions, there are methods based on sub-sampling, re-weighting, or grouping the training set [17, 25, 83]. These methods have been found to be less effective than methods that use data augmentation or regularization [36, 67, 80, 93]. Making stronger assumptions, there are methods which assume access to several distinct training distributions [22, 110].

Consequently, the methods designed for image classification that use data augmentation or regularization represent SPIRE's most direct competition. As a result, we compare against "Right for the Right Reasons" (RRR) [82], "Quantifying and Controlling the Effects of Context" (QCEC) [89], "Contextual Decomposition Explanation Penalization" (CDEP) [80], "Gradient Supervision" (GS) [102], and the "Feature Splitting" (FS) method from [93].

## 4.3 Spurious Pattern Identification and REpair

In this section, we explain SPIRE's approach for addressing SPs. We use the object-classification task as a running example, where *Main* is the object being detected and *Spurious* is the other object in a SP. The same methodology applies to any binary classification problem with a binary "spurious feature;" see Appendix C.2.1 for a discussion on how to do this.

**Preliminaries.** We view a dataset as a probability distribution over a set of *image splits*, which we call *Both*, *Just Main*, *Just Spurious*, and *Neither*, depending on which of Main and/or Spurious they contain (*e.g.,* Just Main is the set of images with tennis rackets, without people, and either with or without any other object). Figure 4.3 (Left) shows these splits for the tennis racket example. Critically, we can take an image from one split and create a counterfactual version of it in a different split by either adding or removing either Main or Spurious (*e.g.,* removing the people from an image in Both moves it to Just Main). To do this, we assume access to pixel-wise object-annotations; note that this is a common assumption (*i.e.,* RRR, QCEC, CDEP, and GS also make it). See Appendix C.2.2 for more details on the counterfactual images.

To summarize, SPIRE makes two general assumptions:

- That we can determine which of these splits an image belongs to.
- That we can create a counterfactual version of an image from one split that belongs to a different split.

**Identification.** SPIRE measures how much the model relies on Spurious to detect Main by measuring the probability that, when we remove Spurious from an image from Both, the model's prediction changes (*e.g.,* the model's prediction for tennis racket changes 63% of the time when we remove the people from an image with both a tennis racket and a person). Intuitively, higher probabilities indicate stronger patterns.

To identify the full set of patterns that the model is using, SPIRE measures this probability for all (Main, Spurious) pairs, where Main and Spurious are different, and then sorts this list to find the pairs that represent the strongest patterns. Recall that not all patterns are necessarily spurious and that the user may label patterns as spurious or valid as needed before moving to the mitigation step.

**Mitigation.** It is often, but not always, the case that there is a strong correlation between Main and Spurious in the original training distribution, which incentivizes the model to rely on this SP. As a result, we want to define a distribution, which we call the *balanced distribution*, where relying on this SP is neither inherently helpful nor harmful. This is a distribution, exemplified in Figure 4.3 (Right), that:

- *Preserves P(Main).* This value strongly influences the model's relative accuracy on {Both, Just Main} versus {Just Spurious, Neither} but does not incentivize the SP. As a result, we preserve it in order to maximize the similarity between the original and balanced distributions.
- *Sets P(Spurious | Main) = P(Spurious | not Main) = 0.5.* This makes Main and Spurious independent, which removes the statistical benefit of relying on the SP, and assigns equal importance to images with and without Spurious. However, this does not go so far as to invert the original correlation, which would directly punish reliance on the SP.

As shown in Figure 4.3 (Right), SPIRE's mitigation strategy uses counterfactual images to manipulate the training distribution. The specifics are described in Section 4.3.1, but they implement two goals:

- *Primary: Shift the training distribution towards the balanced distribution.* While the original distribution often incentivizes the model to rely on the SP, the balanced distribution does not. However, adding too many counterfactual images may compromise the model's accuracy on natural images. As a result, we want to shift the training distribution towards, but not necessarily all the way to, the balanced distribution.
- *Secondary: Minimize the potential for new SPs.* While shifting towards the balanced distribution, we may inadvertently introduce new potential SPs between Main and artifacts in the counterfactual images. For example, augmenting the dataset with the same counterfactuals that SPIRE uses for identification (*i.e.,* images from Both where Spurious has been covered with a grey box) introduces the potential for a new SP because P(Main | "grey box") = 1.0. Because the augmentation will be less effective if the model learns to rely on new SPs, we minimize their potential by trying to set P(Main | Artifact) = 0.5.

### 4.3.1 Specific Mitigation Strategies

While SPIRE's augmentation strategy follows the aforementioned goals, its specific details depend on the problem setting, which we characterize using two factors:

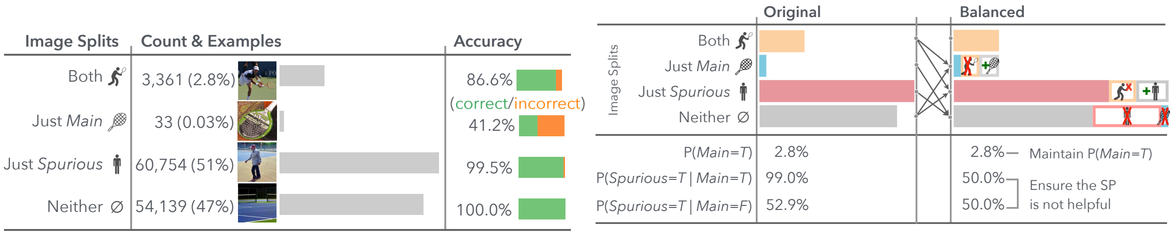


**Figure 4.3:** **Left.** The training image splits and the original training distribution for the tennis racket example. Because of the strong positive correlation between Main and Spurious, it is helpful for the model to rely on this SP. **Right.** The balanced distribution for the tennis racket example. This SP is no longer helpful because Main and Spurious are now independent and there are the same number of images in Both and Just Main.

- *Can the counterfactuals change an image's label?* For tasks such as object-classification, counterfactuals can change an image's label by removing or adding Main. However, for tasks such as scene identification, we may not have counterfactuals that can change an image's label. For example, we cannot turn a runway into a street or a street into a runway by manipulating a few objects. This fundamentally shapes how counterfactuals can change the training distribution.
- *Is the dataset class balanced?* While working with class balanced datasets drastically simplifies the problem and analysis, it is not an assumption that usually holds in practice.

These two factors define the three problem settings that we consider, which correspond to the experiments in Sections 4.5.1, 4.5.2, and 4.5.3 respectively. For each setting, we summarize what makes it interesting, define SPIRE's specific augmentation strategy for it, and then discuss how that strategy meets SPIRE's goals.

**Setting 1: Counterfactuals can change an image's label and the dataset is class-balanced.** Here, P(Main) = P(Spurious) = 0.5 and we can define the training distribution by specifying $p$ = P(Main | Spurious). If $p > 0.5$, SPIRE moves images from $\{$Both, Neither$\}$ to $\{$Just Main, Just Spurious$\}$ with probability $\frac{2p-1}{2p}$ for each of those four combinations. If $p < 0.5$, SPIRE moves images from $\{$Just Main, Just Spurious$\}$ to $\{$Both, Neither$\}$ with probability $\frac{p-0.5}{p-1}$.

Table 4.1 shows how SPIRE changes the training distributions for $p = 0.9$ and $p = 0.1$. For $p = 0.9$, it succeeds at both of its goals. For $p = 0.1$, it produces the balanced distribution, but does add the potential for new SPs because P(Main | Removed an object) = 0 and P(Main | Added an object) = 1. We contrast SPIRE to the most closely related method, QCEC [89], which removes either Main or Spurious uniformly at random, as applicable, from each image. For both values of $p$, QCEC does not make Main and Spurious independent and adds the potential for new SPs. This example highlights the fact that, while prior work has used counterfactuals for data augmentation, SPIRE uses them in a fundamentally different way by considering the training distribution.

**Setting 2: Counterfactuals can change an image's label, but the dataset has class imbalance.** Class imbalance makes two parts of the definition of the balanced distribution problematic for augmentation:

**Table 4.1:** Setting 1. For $p = 0.9$ and $p = 0.1$, we show the original size of each split for a dataset of size 200 as well as the size of each split after SPIRE's or QCEC's augmentation. Note that SPIRE produces the balanced distribution, while QCEC does not even make Main and Spurious independent.

| | p = 0.9 | | | p = 0.1 | | |
|---|---|---|---|---|---|---|
| Split | Original | SPIRE | QCEC | Original | SPIRE | QCEC |
| Both | 90 | 90 | 90 | 10 | 90 | 10 |
| Just Main | 10 | 90 | 55 | 90 | 90 | 95 |
| Just Spurious | 10 | 90 | 55 | 90 | 90 | 95 |
| Neither | 90 | 90 | 110 | 10 | 90 | 190 |

- *Preserves P(Main).* When P(Main) is small, this means that we generate many more counterfactual images without Main than with it, which can introduce new potential SPs.
- *Sets P(Spurious | not Main) = 0.5.* When P(Spurious) is also small, this constraint requires that most of the counterfactual images we generate belong to Just Spurious, which can lead to the counterfactual data outnumbering the original data by a factor of 100 or more for this split.

Consequently, we relax these constraints. If P(Spurious | Main) $>$ P(Spurious), SPIRE creates an equal number of images to add to Just Main/Spurious by removing the appropriate object from an image from Both. Specifically, this number is the smallest positive solution for $\delta$ to: $\frac{|\text{Both}|}{|\text{Both}|+|\text{Just Spurious}|+\delta} = \frac{|\text{Just Main}|+\delta}{|\text{Just Main}|+|\text{Neither}|+\delta}$. Otherwise, SPIRE creates an equal number of images to add to Both/Just Spurious by adding Spurious to Just Main/Neither. Specifically, this number solves: $\frac{|\text{Both}|+\delta}{|\text{Both}|+|\text{Just Spurious}|+2\delta} = \frac{|\text{Just Main}|}{|\text{Just Main}|+|\text{Neither}|}$. SPIRE achieves its primary goal by making P(Main | Spurious) = P(Main | not Spurious) (*i.e.,* Main and Spurious are now independent) and it achieves its secondary goal by adding an equal number of counterfactual images with and without Main (*i.e.,* P(Main | Artifact) = 0.5).

**Setting 3: Counterfactuals cannot change an image's label.** This constraint prevents the previous strategies from being applied. Therefore, SPIRE removes Spurious from every image with Spurious and adds Spurious to every image without Spurious. While this process does achieve SPIRE's primary goal, it does not achieve SPIRE's secondary goal (*i.e.,* the original correlation between the label and Spurious is the same as the correlation between the label and grey boxes from removing Spurious).

## 4.4 Evaluation

Because relying on the SP is usually helpful on the original distribution, we cannot measure the effectiveness of a mitigation method using that distribution. Instead, we measure the model's performance on the balanced distribution, using metrics such as accuracy and average precision. Intuitively, using the balanced distribution provides a fairer comparison because the SP is neither helpful nor harmful on it. However, like any performance metric that is aggregated over a distribution, these metrics hide potentially useful details.

We address this limitation by measuring the model's accuracy on each of the image splits. These per split accuracies yield a more detailed analysis, allow us to estimate the model's performance on any distribution (*e.g.,* the balanced distribution) by re-weighting the model's accuracy on them, and allow us to calculate two "gap metrics". The *Recall Gap* is the difference in accuracy between Both and Just Main; the *Hallucination Gap* is the difference in accuracy between Neither and Just Spurious. Intuitively, a smaller recall gap means that the model is more robust to distribution shifts that move weight between Both and Just Main. The same is true for the hallucination gap and shifts between Neither and Just Spurious. As a concrete example of these metrics, consider the tennis racket example (Figure 4.3 Left), where we observe that the recall gap is 45.4% (*i.e.,* the model is much more likely to detect a tennis racket when a person is present) and a hallucination gap of 0.5% (*i.e.,* the model is more likely to hallucinate a tennis racket when a person is present; see Appendix C.3.2).

Note that these per split accuracies are measured using only natural (*i.e.,* not counterfactual) images, in order to prevent the model from "cheating" by learning to use artifacts in the counterfactual images. As a result, the gap metrics and the performance on the balanced distribution also only use natural images.

**Class Balanced vs Imbalanced Evaluation.** When there is class balance, we use the standard prediction threshold of 0.5 to measure a model's performance using accuracy on the balanced distribution (*i.e., balanced accuracy*) and its gap metrics. When there is class imbalance, Average Precision (AP), which is the area under the precision vs recall curve, is the standard performance metric. Analogous to AP, we can calculate the Average Recall Gap by finding the area under the "absolute value of the recall gap" vs recall curve; the Average Hallucination Gap is defined similarly. As a result, we measure a model's performance using AP on the balanced distribution (*i.e., balanced AP*) and the Average Recall/Hallucination Gaps. Appendix C.4 provides more details for these metrics by showing how they are calculated using the tennis racket example.

## 4.5 Experiments

We divide our experiments into three groups:

- In Section 4.5.1, we induce SPs with varying strengths by sub-sampling COCO in order to understand how mitigation methods work in a controlled setting. We show that SPIRE is more effective at mitigating these SPs than prior methods. We also use these results to identify the best prior method, which we use for comparison for the remaining experiments.
- In Section 4.5.2, we find and fix SPs "in the wild" using all of COCO; this means finding multiple naturally occurring SPs and fixing them simultaneously. We show that SPIRE identifies a wider range of SPs than prior methods and that it is more effective at mitigating them. Additionally, we show that it improves zero-shot generalization to two challenging datasets (UnRel and SpatialSense).
- In Section 4.5.3, we demonstrate that SPIRE is effective for problems other than COCO. To do so, we consider tasks other than object-classification and/or explore techniques for constructing counterfactuals for datasets without pixel-wise object-annotations.

For the baseline models (*i.e.,* the normally trained models that contain the SPs that we are going to identify and mitigate), we fine-tune a pre-trained version of ResNet18 [38] (see Appendix C.5). We compare SPIRE to RRR [82], QCEC [89], CDEP [80], GS [102], and FS [93]. We

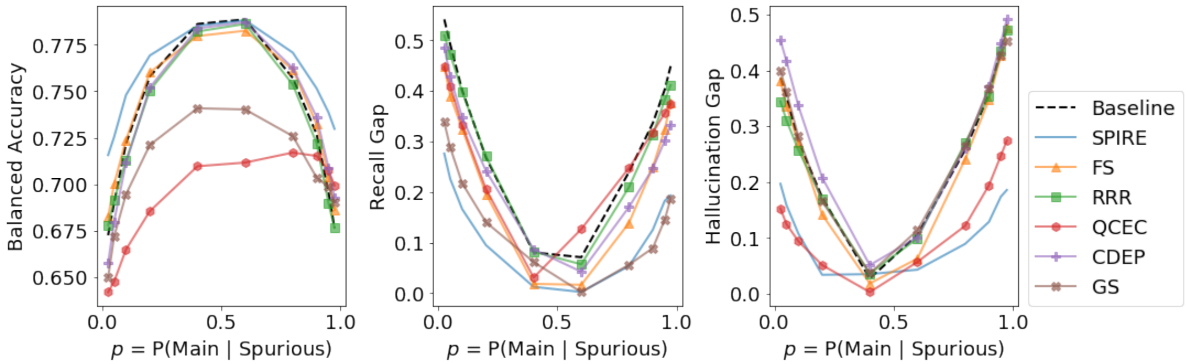


**Figure 4.4:** A comparison of the baseline model to various mitigation methods. The results shown are averaged across both the pairs accepted for our benchmark and across eight trials. **Left - Balanced Accuracy.** For $p \leq 0.2$ and $p \geq 0.8$, SPIRE produces the most accurate models. None of the methods have much of an impact for $p = 0.4$ or $p = 0.6$, likely because those create weak SPs. **Center/Right - Recall/Hallucination Gaps.** SPIRE generally shrinks the absolute value of both of the gap metrics by more than prior methods.

use the evaluation described in Section 4.4 and, for any split that is too small to produce a reliable accuracy estimate, we acquire additional images (using Google Images) such that each split has at least $30$ images to use for evaluation.

**Measuring variance across trials.** Because we are applying these mitigation methods to the baseline model, the results of some metric (*e.g.,* Balanced AP) for the mitigated model and the baseline model are not independent. As a result, we do not directly report the variance of this metric across trials and, instead, report the variance its difference between the mitigated and baseline models. Subsequently, the "[$\sigma$=]" entries in our tables will denote the standard deviation of a particular metric's difference across trials.

### 4.5.1 Benchmark Experiments

We construct a set of benchmark tasks from COCO consisting of different SPs with varying strengths, by manipulating the model's training distribution, in order to better understand how mitigation methods work in a controlled setting. Appendix C.6 has additional details.

**Creating the benchmark.** We start by finding each pair of objects that has at least 100 images in each split of the testing set (13 pairs). For each of those pairs, we create a series of controlled training sets of size 2000 by sampling images from the full training set such that P(Main) = P(Spurious) = 0.5 and $p$ = P(Main | Spurious) ranges between 0.025 and 0.975. Each controlled training set represents a binary task, where the goal is to predict the presence of Main.

While varying $p$ allows us to control the strength of the correlation between Main and Spurious (*i.e.,* $p$ near 0 indicates a strong negative correlation while $p$ near 1 indicates a strong positive correlation), it does not guarantee that the model actually relies on the intended SP. Indeed, when we measure the models' balanced accuracy as $p$ varies, we observe that $5$ out of the $13$ pairs show little to no loss in balanced accuracy as $p$ approaches $1$. Consequently, subsequent evaluation considers the other $8$ pairs. For these pairs, the model's reliance on the SP increases as $p$ approaches $0$ or $1$ as evidenced by the increasing loss of balanced accuracy.

**Results.** Figure 4.4 (Left) presents the balanced accuracy results. We find that SPIRE consistently improves balanced accuracy and that it does so by more than prior methods. Interestingly, while most prior methods are beneficial for strong positive SPs ($p \geq 0.9$), only FS is also (mildly) beneficial for negative SPs ($p < 0.5$).

Figure 4.4 (Center/Right) presents the gap metric results. We find that SPIRE is the most effective method at shrinking these metrics, which indicates that it produces a model that is more robust to distribution shift. Interestingly, QCEC and GS, which are the two prior methods that include data augmentation, are the only prior methods that substantially shrink the gap metrics (at the cost of balanced accuracy for $p < 0.9$).

Overall, this experiment shows that SPIRE is an effective mitigation method and that our evaluation framework enables us to easily understand how methods affect the behavior of a model. We use FS as the baseline for comparison for the remaining experiments because, of the prior methods, it had the best average balanced accuracy across $p$'s range.

### 4.5.2 Full Experiment

We evaluate SPIRE "in the wild" by identifying and mitigating SPs learned by a multi-label binary object-classification model trained on the full COCO dataset. Appendices C.4 and C.7 have additional details.

**Identification.** Out of all possible (Main, Spurious) pairs, we consider those which have at least 25 training images in Both ($\approx 2700$). From these, SPIRE identifies 29 where the model's prediction changes at least 40% of the time when we remove Spurious. Table 4.2 shows a few of the identified SPs; overall, they are quite diverse: the spurious object ranges from common (*e.g.,* person) to rare (*e.g.,* sheep); the SPs range from objects that are commonly co-located (*e.g.,* tie-person) to usually separate (*e.g.,* dog-sheep); and a few Main objects (*e.g.,* tie and frisbee) have more than one associated SP. Notably, SPIRE identifies negative SPs (*e.g.,* tie-cat) while prior work [89, 93, 102] only found positive SPs. Appendix C.7 includes a discussion of how we verified the veracity of these identified SPs.

**Mitigation.** Unlike the Benchmark Experiments, this experiment requires mitigating many SPs simultaneously. Because we found that SPIRE was more effective if we only re-train the final layer of the model in the Benchmark Experiments (see Appendix C.5), we do this by re-training the slice of the model's final layer that corresponds to Main's class on an augmented dataset that combines SPIRE's augmentation for each SP associated with Main. All results shown (Tables 4.3 and 4.4) are averaged across eight trials.

We conclude that SPIRE significantly reduces the model's reliance on these SPs based on two main observations. First, it increases balanced AP by 1.1% and shrinks the average recall/hallucination gaps by a factor of 14.2/28.1%, relative to the baseline model, on COCO. As expected, this does slightly decrease Mean Average Precision (MAP) by 0.4% on the original (biased) distribution. Second, it increases MAP on the UnRel [70] and SpatialSense [116] datasets. Because this evaluation was done in a zero-shot manner and these datasets are designed to have objects in unusual contexts, this is further evidence that SPIRE improves

**Table 4.2:** A few examples of the SPs identified by SPIRE for the Full Experiment. For each pair, we report several basic dataset statistics including *bias*, $\frac{\text{P(Spurious | Main) - P(Spurious)}}{\text{P(Spurious)}}$, which captures how far Main and Spurious are away from being independent as well as the sign of their correlation.

| Main | Spurious | P(M) | P(S) | P(S \| M) | bias |
|---|---|---|---|---|---|
| tie | cat | 0.03 | 0.04 | 0.01 | -0.66 |
| toothbrush | person | 0.01 | 0.54 | 0.54 | -0.01 |
| bird | sheep | 0.03 | 0.01 | 0.01 | 0.00 |
| frisbee | person | 0.02 | 0.54 | 0.83 | 0.54 |
| tie | person | 0.03 | 0.54 | 0.95 | 0.76 |
| tennis racket | person | 0.03 | 0.54 | 0.99 | 0.83 |
| dog | sheep | 0.04 | 0.01 | 0.03 | 1.05 |
| frisbee | dog | 0.02 | 0.04 | 0.24 | 5.44 |
| fork | dining table | 0.03 | 0.10 | 0.76 | 6.56 |

**Table 4.3:** Mitigation results for the Full Experiment. Balanced AP is averaged across the SPs identified by SPIRE. Similarly, the gap metrics are reported as the "mean (median)" change from the baseline model, aggregated across those SPs.

| | Original MAP | Balanced AP | %Δ Avg. Recall Gap | %Δ Avg. Hallucination Gap |
|---|---|---|---|---|
| Baseline | **64.1** | 46.2 | — | — |
| SPIRE | 63.7 [$\sigma$=0.1] | **47.3** [$\sigma$=0.5] | **-14.2 (-14.5)** | **-28.1 (-27.3)** |
| FS | 62.5 [$\sigma$=1.0] | 44.7 [$\sigma$=1.9] | 9.7 (-5.9) | 25.7 (-6.9) |

**Table 4.4:** The MAP results of a zero-shot evaluation on the classes that are in the UnRel/SpatialSense datasets that SPIRE also identified as being Main in a SP.

| | UnRel | SpatialSense |
|---|---|---|
| Baseline | 38.9 | 20.3 |
| SPIRE | **41.3** [$\sigma$=1.3] | **20.7** [$\sigma$=0.5] |
| FS | 39.6 [$\sigma$=2.1] | 18.6 [$\sigma$=0.3] |

**Table 4.5:** Results for the Scene Identification Experiment (averaged across sixteen trials).

| | Original AP | Balanced AP | %Δ Avg. Recall Gap | %Δ Avg. Hallucination Gap |
|---|---|---|---|---|
| Baseline | **95.0** | 48.9 | — | — |
| SPIRE | 92.8 [$\sigma$=1.6] | **83.2** [$\sigma$=7.3] | **-82.1** | **-75.5** |
| FS | 93.7 [$\sigma$=1.3] | 47.8 [$\sigma$=8.6] | -11.5 | 3.7 |

**Table 4.6:** Results for the No Object Annotation Experiment for the tennis racket example (averaged across eight trials).

| | Original AP | Balanced AP | %Δ Avg. Recall Gap | %Δ Avg. Hallucination Gap |
|---|---|---|---|---|
| Baseline | 93.9 | 79.9 | — | — |
| SPIRE | 92.9 [$\sigma$=0.4] | 80.5 [$\sigma$=1.3] | **-31.3** | -27.3 |
| SPIRE-R | **94.0** [$\sigma$=0.6] | **81.0** [$\sigma$=1.7] | -9.1 | **-44.9** |
| FS | 92.9 [$\sigma$=1.0] | 80.7 [$\sigma$=2.2] | -10.4 | -22.3 |

**Table 4.7:** Results for the ISIC Experiment (averaged across eight trials).

| | Original AP | Balanced AP | %Δ Avg. Recall Gap | %Δ Avg. Hallucination Gap |
|---|---|---|---|---|
| Baseline | 78.3 | 71.0 | — | — |
| SPIRE-EM | **78.8** [$\sigma$=4.9] | **76.4** [$\sigma$=2.4] | -20.5 | -39.0 |
| FS | 70.7 [$\sigma$=6.8] | 68.0 [$\sigma$=3.8] | **-31.3** | **-61.0** |

distributional robustness. In contrast, FS decreases the model's performance, has inconsistent effects on the gap metrics, and has mixed results on the zero-shot evaluation.

**SPIRE and Distributional Robustness.** Noting that robustness to specific distribution shifts is one of the consequences of mitigating SPs, we can contextualize the impact of SPIRE by considering an extensive meta-analysis of methods that aim to provide general robustness [101]. This analysis finds that the only methods that consistently work are those that re-train the baseline model on several orders of magnitude more data. It also describes two necessary conditions for a method to work. Notably, SPIRE satisfies both of those conditions: first, it improves performance on the shifted distributions (*i.e.,* the balanced distributions, UnRel, and SpatialSense) and, second, this improvement cannot be explained by increased performance on the original distribution. Consequently, SPIRE 's results are significant because they show improved robustness without using orders of magnitude more training data. We hypothesize that SPIRE is successful because it targets specific SPs rather than using a less targeted approach.

### 4.5.3 Generalization Experiments

We illustrate how SPIRE generalizes beyond the setting from COCO, where we considered the object-classification task and assumed that the dataset has pixel-wise object-annotations. Specifically, we explore three examples that consider a different task (*Generalization 1*) or do not assume this (*Generalization 2*).

**Scene Identification Experiment (Generalization 1).** In this experiment, we construct a scene identification task using the image captions from COCO and show that SPIRE can identify and mitigate a naturally occurring SP. To do this, we define two classes: one where the word "runway" (the part of an airport where airplanes land) appears in the caption (1,134 training images) and another where "street" appears (12,543 training images); images with both or without either are discarded. For identification, we observe that removing all of the airplanes from an image of a runway changes the model's prediction 50.7% of the time.

Table 4.5 shows the results. SPIRE reduces the model's reliance on this SP because it substantially increases balanced AP and it reduces the average recall and hallucination gaps by factors of 82.1% and 75.5%. In contrast, FS is not effective at mitigating this SP.

**No Object Annotation Experiment (Generalization 2).** In this experiment, we mitigate the SP from the tennis racket example without assuming that we have pixel-wise object-annotations; instead, we make the weaker assumption that we have binary labels for the presence of Spurious. To do this, we train a linear (in the model's representation space) classifier to predict whether or not an image contains a person (similar to [44]). Then, we project across this linear classifier to essentially add or remove a person from an image's *representation* (so we call this method SPIRE-R) (see Appendix C.8.1).

Table 4.6 shows the results. First, note that SPIRE provides a small increase in Balanced AP while providing the largest average decrease in the gap metrics. Second, note that SPIRE-R is preferable to FS because it produces a larger reduction in the hallucination gap while being otherwise comparable.

**ISIC Experiment (Generalizations 1 & 2).** In this experiment, we imitate the setup from [80] for the ISIC dataset [24]. Specifically: the task is to predict whether an image of a skin lesion is malignant or benign; the model learns to use a SP where it relies on a "brightly colored sticker" that is spuriously correlated with the label; and the dataset does not have annotations for those stickers to use create counterfactual images. For this experiment, we illustrate another approach for working with datasets that do not have pixel-wise object-annotations: using *external models* (so we call this method SPIRE-EM) to produce potentially noisy annotations with less effort than actually annotating the entire dataset. The external model could be an off-the-shelf model (*e.g.,* a model that locates text in an image) or a simple pipeline such as the super-pixel clustering one we use (see Appendix C.8.2).

Table 4.7 shows the results. We can see that SPIRE-EM is effective at mitigating this SP because it generally improves performance while also shrinking the gap metrics.[2] In contrast,

[2] Note that, because the dataset does not contain enough malignant images with stickers to produce a reliable accuracy estimate, we had to use counterfactual images for this evaluation. This may influence the results for balanced AP and the recall

FS does not seem to be beneficial because it substantially reduces performance on both the original and balanced distributions (which outweighs shrinking the gap metrics).

## 4.6 Conclusion

In this work, we introduced SPIRE as an end-to-end solution for addressing Spurious Patterns for image classifiers that are relying on spurious objects to make predictions. SPIRE identifies potential SPs by measuring how often the model's prediction changes when we remove the Spurious object from an image with a positive label and mitigates SPs by shifting the training distribution towards the balanced distribution while minimizing any correlations between the label and artifacts in the counterfactual images. We demonstrated that SPIRE is able identify and, at least partially, mitigate a diverse set of SPs by improving the model's performance on the balanced distribution and by making it more robust to specific distribution shifts. We found that these improvements lead to improved zero-shot generalization to challenging datasets. Then, we showed that SPIRE can be applied to tasks other than object-classification and we illustrated two potential ways to apply SPIRE to datasets without pixel-wise object-annotations to use to create counterfactuals (creating counterfactual representations and using external models). More generally, in terms of identifying SPs, our findings provide additional evidence for the claim that a correlation is neither sufficient nor necessary for a model to learn a SP.

gap metric. However, SPIRE-EM is still an improvement over the baseline because of the original AP and hallucination gap metric results.

# Chapter 5

# Blindspots

Recall that the goal of this application is to find blindspots, which occur when an image classification model underperforms on a semantically meaningful subset of the data, while making minimal assumptions. To do this, we use a family of bespoke methods called Blindspot Discovery Methods. While past works have used evaluations based on the size or error rate of the hypothesized blindspots returned by these methods, these evaluations have limitations that we seek to address. Both these methods and these evaluations are described in Section 5.2.

To evaluate the usefulness of these methods for this application, we run two evaluations. For the first, we artificially induce blindspots in a model in order to directly compare the hypothesized blindspots returned by these methods to the blindspots that we induced (Sections 5.3 and 5.5). For the second, we illustrate a pipeline to measure the ability of a user to identify naturally occurring blindspots in a model using these methods (Section 5.6).

Beyond this individual work, there are a few general ideas to keep in mind for this application:

- At a high level, these methods help facilitate a type of model debugging where the model developer iteratively finds and fixes blindspots in a model by suggesting blindspots to the model developer. Then, the model developer can determine whether a blindspot is worth fixing and, if it is, what the best fix is.[1]
- This application does not cover all of model debugging. For example, there may be systemic errors that are not semantically meaningful (*e.g.,* those defined by the Fourier spectrum of an image). As a result, we should view it as a complementary approach to approaches that focus on other notions of model robustness or performance.

## 5.1 Introduction

A growing body of work has found that models with high test performance can still make systemic errors, which occur when the model performs significantly worse on a semantically meaningful subset of the data [13, 23, 68, 77, 94]. For example, past works have demonstrated that models trained to diagnose skin cancer from dermoscopic images sometimes rely on

[1] Blindspots can have many different causes such as relying on spurious correlations, small sample sizes, labeling errors, and data processing errors, which may require different fixes.

spurious artifacts (*e.g.,* surgical skin markers that some dermatologists use to mark lesions); consequently, they have different performance on images with or without those spurious artifacts [64, 111]. More broadly, finding systemic errors can help us detect algorithmic bias [13] or sensitivity to distribution shifts [83, 93].

In this work, we focus on what we call the *blindspot discovery* problem, which is the problem of finding an image classification model's systemic errors[2] without making many of the assumptions considered in related works (*e.g.,* we do not assume access to metadata to define semantically meaningful subsets of the data, tools to produce counterfactual images, a specific model structure or training process, or a human in the loop). We call methods for addressing this problem Blindspot Discovery Methods (BDMs) [*e.g.,* 26, 33, 46, 94, 97].

We note that blindspot discovery is an emerging research area and that there has been more emphasis on developing BDMs than on formalizing the problem itself. Consequently, we propose a problem formalization, summarize different approaches for evaluating BDMs, and summarize several high-level design choices made by BDMs. When we do this, we observe the following two gaps. First, existing evaluations are based on an incomplete knowledge of the model's blindspots, which limits the types of measurements and claims they can make. Second, dimensionality reduction is a relatively underexplored aspect of BDM design.

Motivated by these gaps in prior work, we propose a new evaluation framework, `SpotCheck`, and a new BDM, `PlaneSpot`. `SpotCheck` is a synthetic evaluation framework for BDMs that gives us complete knowledge of the model's blindspots and allows us to identify factors that influence BDM performance. Additionally, we refine the metrics used by past evaluations. `PlaneSpot` is a simple BDM that finds blindspots using a 2D image representation.

We use `SpotCheck` to run controlled experiments that identify factors that influence BDM performance (*e.g.,* the number of blindspot in a model) and show that `PlaneSpot` outperforms existing BDMs. We run additional semi-controlled experiments using the COCO dataset [60] and find that these trends discovered using `SpotCheck` generalize to real image data. Overall, we hope that the methodology and analyses presented in this work will help facilitate a more rigorous science of blindspot discovery.

## 5.2 Background

In this section, we formalize the problem of *blindspot discovery* for image classification. We then discuss general approaches for evaluating the Blindspot Discovery Methods (BDMs) designed to address this problem as well as high-level design choices made by BDMs.

**Problem Definition.** The broad goal of finding systematic errors has been studied across a range of problem statements and method assumptions. Some common assumptions are:

- Access to metadata help define coherent subsets of the data [*e.g.,* 13, 44, 93].
- The ability to produce counterfactual images [*e.g.,* 10, 53, 75, 89, 95, 115].

[2] In past work, "systemic errors" have also been called "failure modes" or "hidden stratification." We introduce "blindspot" to mean the same thing and use it make it clear when we are specifically discussing blindspot discovery.

- A specific structure for the model we are analyzing [*e.g.,* 6, 50] or for training process used to learn the model's parameters [*e.g.,* 112].
- A human-in-the loop, either through an interactive interface [*e.g.,* 8, 14] or by inspecting explanations [*e.g.,* 3, 118].

While appropriate at times, these assumptions all restrict the applicability of their respective methods. For example, consider assuming access to metadata to help define coherent subsets of the data. To start, this metadata is much less common in applied settings than it is for common ML benchmarks. Further, the efficacy of methods that rely on this metadata is limited by the quantity and relevance of this metadata; in general, efficiently collecting large quantities of relevant metadata is challenging because it requires anticipating the model's systemic errors.

Consequently, we define the problem of blindspot discovery as the problem of finding an image classification model's systemic errors without making any of these assumptions. More formally, suppose that we have an image classifier, $f$, and a dataset of labeled images, $D = [x_i]_{i=1}^n$. Then, a *blindspot* is a coherent (*i.e.,* semantically meaningful) set of images, $\Psi \subset D$, where $f$ performs significantly worse (*i.e.,* $p(f, \Psi) \ll p(f, D \setminus \Psi)$ for some performance metric, $p$, such as recall). We denote the set of $f$'s *true blindspots* as $\mathbf{\Psi} : \{\Psi_m\}_{m=1}^M$. Next, we define the problem of *blindspot discovery* as the problem of finding $\mathbf{\Psi}$ using only $f$ and $D$. Then, a BDM is a method that takes as input $f$ and $D$ and outputs an ordered (by some definition of importance) list of *hypothesized blindspots*, $\hat{\mathbf{\Psi}} : [\hat{\Psi}_k]_{k=1}^K$. Note that the $\Psi_m$ and $\hat{\Psi}_k$ are sets of images.

**Approaches to BDM evaluation.** We observe that existing approaches to quantitatively evaluate BDMs fall in two categories. The first category of evaluations simply measure the error rate or size of $\hat{\Psi}_k$ [26, 94]. However, these evaluations have two problems. First, none of the properties they measure capture whether $\hat{\Psi}_k$ is coherent (*e.g.,* a random sample of misclassified images has high error but may not match a single semantically meaningful description). Second, $f$'s performance on $\hat{\Psi}_k$ may not be representative of $f$'s performance on similar images because BDMs are optimized to return high error images (*e.g.,* suppose that $f$ has a 90% accuracy on images of "squares and blue circles"; then, by returning the 10% of such images that are misclassified, a BDM could mislead us into believing that $f$ has a 0% accuracy on this type of image).

The second category of evaluations compares $\hat{\mathbf{\Psi}}$ to a subset of $\mathbf{\Psi}$ that has either been previously found or artificially induced [33, 97]. While these evaluations address the issues with those from the first category, they require knowledge of $\mathbf{\Psi}$, which is usually incomplete (*i.e.,* we usually do not know some of the $\Psi_m$). This incompleteness makes it difficult to identify factors that influence BDM performance or to measure a BDM's recall or false positive rate. It is not practical to fix this incompleteness using real data because we cannot realistically enumerate all of the possible coherent subsets of $D$ to check if they are blindspots. To address these limitations, we introduce `SpotCheck`, which gives us complete knowledge of $\mathbf{\Psi}$ by using synthetic data.

**High-level design choices of BDMs.** In Table 5.1, we summarize three of the high-level design choices made by existing BDMs. First, each BDM uses a model to extract an *image representation*. Many BDMs use a representation from $f$, but some use pre-trained external

**Table 5.1:** A high level overview the high-level design choices made by different BDMs.

| Method | 1. Image Representation | 2. Dimensionality Reduction | 3. Hypothesis Class |
|---|---|---|---|
| Multiaccuracy [46] | VAE representation | | Linear model |
| GEORGE [97] | Model representation | UMAP ($d = 0, 1, 2$) | Gaussian kernels |
| Spotlight [26] | Model representation | | Gaussian kernels |
| Barlow [94] | Adversarially-Robust Model representation | | Decision Tree |
| Domino [33] | CLIP representation | PCA ($d = 128$) | Gaussian Kernels |
| `PlaneSpot` | Model representation | scvis ($d = 2$) | Gaussian Kernels |

models or other models trained on $D$. Second, some of the BDMs apply some form of *dimensionality reduction* to that image representation. Third, each BDM learns a model from a specified *hypothesis class* to predict if an image belongs to a blindspot from that image's (potentially reduced) representation.

Interestingly, while there has been significant effort focused on the choice of a BDM's image representation and hypothesis class (along with its associated learning algorithm), we note that dimensionality reduction has received much less attention. This is surprising because these BDMs are all solving clustering or learning problems, which are generally easier in lower dimensions. Motivated by this gap in prior work, we introduce a simple BDM, `PlaneSpot`, that uses a 2D representation.

## 5.3 Evaluating BDMs using Knowledge of the True Blindspots

In Section 5.3.1, we introduce `SpotCheck`, which is an evaluation framework for BDMs that gives us complete knowledge of the model's true blindspots by using synthetic images and allows us to identify factors that influence BDM performance. In Section 5.3.2, we define the metrics that we use to measure the performance of BDMs given knowledge of the model's true blindspots.

### 5.3.1 `SpotCheck`: A Synthetic Evaluation Framework for BDMs

`SpotCheck` builds on ideas from Kim et al. [45] by generating synthetic datasets of varying complexity and training models to have specific blindspots on those datasets. We summarize its key steps below; see Appendix D.1 for details.

**Dataset Definition.** Each dataset is defined using *semantic features* that describe the possible types of images it contains. Datasets that have a larger number of features have a larger variety of images and are therefore more complex. For example, a simple dataset may only contain images with squares and blue or orange circles (see Figure 5.1), while a more complicated dataset may also contain images with striped rectangles, small text, or grey backgrounds.

**Blindspot Definition.** Each blindspot is defined using a subset of the semantic features that define its associated dataset (see Figure 5.1). Similarly to how a dataset with more features is more complex, a blindspot defined using more semantic features is more specific.

**Training a Model to have Specific Blindspots.** For each dataset and blindspot specification, we train a ResNet-18 model [38] to predict whether a square is present. To induce blindspots,

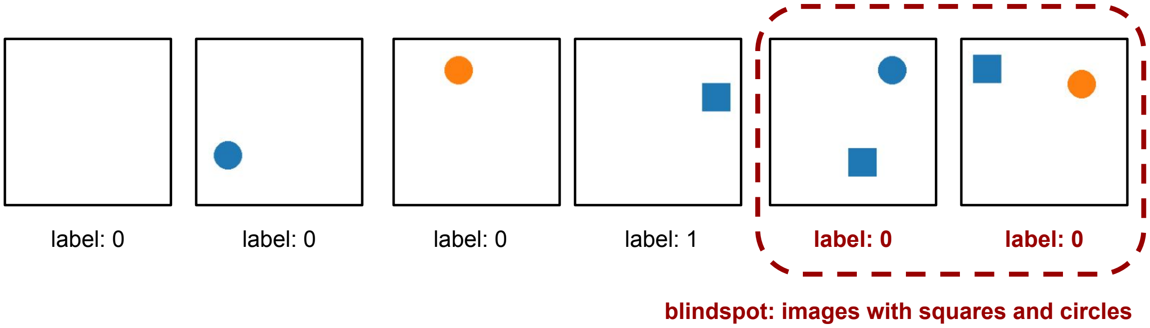


**Figure 5.1:** By controlling the data generation process, `SpotCheck` gives us complete knowledge of a model's true blindspots. Here, we show images sampled from a dataset created with `SpotCheck`. **Dataset Complexity.** This dataset is defined by 3 semantic features that vary across images: the presence of a square, the presence of a circle, and the color of the circle. We do not count the "color of the square" because it is always blue. **Blindspot Specificity.** This blindspot is defined by 2 semantic features: the presence of a square and the presence of a circle. As a result, it contains any image with both a square and a circle, regardless of the circle's color. **Data Labels.** In general, the label indicates if a square is present. However, training images belonging to this blindspot are mislabeled.

we generate data where the label for each image in the training and validation sets is correct if and only if it does not belong to any of the blindspots (see Figure 5.1). The test set images are always correctly labeled. Then, because we know the full set of semantic features that define the dataset, we can verify that the model learned to have exactly the set of blindspots that we intended it to. As a result, `SpotCheck` gives us complete knowledge of the model's true blindspots.

**Generating Diverse Experimental Configurations (ECs).** Since our goal is to study how various factors influence BDM performance, we generate a diverse set of *experimental configurations*, (*i.e.,* dataset, blindspots, and model triplets). To do this, we randomize several *factors*: the features that define a dataset (both the number of them and what they are) as well as the blindspots (the number of them, the number of features that define them, and what those features are). Importantly, we sample these factors independently of one another across this set of ECs so that we can estimate their individual influence on BDM performance.

### 5.3.2 Evaluation Metrics based on Knowledge of the True Blindspots

These metrics measure how well the hypothesized blindspots returned by a BDM, $\hat{\boldsymbol{\Psi}}$, capture a model's true blindspots, $\boldsymbol{\Psi}$. Recall that the $\hat{\Psi}_k$ and $\Psi_m$ are sets of images. To do this, we start by measuring how well a BDM finds each *individual* true blindspot (Blindspot Recall) and build on that to measure how well a BDM finds the *complete set* of true blindspots (Discovery Rate and False Discovery Rate); see Appendix D.2 for a discussion of how these metrics refine prior definitions.

**Blindspot Precision.** If $\hat{\Psi}_k$ is a subset of $\Psi_m$, we know that the model underperforms on $\hat{\Psi}_k$ and that $\hat{\Psi}_k$ is coherent. We measure this using the precision of $\hat{\Psi}_k$ with respect to $\Psi_m$:

$$\text{BP}(\hat{\Psi}_k, \Psi_m) = \frac{|\hat{\Psi}_k \cap \Psi_m|}{|\hat{\Psi}_k|} \tag{5.1}$$

Then, we say that $\hat{\Psi}_k$ *belongs to* $\Psi_m$ if, for some threshold $\lambda_p$:

$$\text{BP}(\hat{\Psi}_k, \Psi_m) \geq \lambda_p \tag{5.2}$$

However, $\hat{\Psi}_k$ can belong to $\Psi_m$ without capturing the same information as $\Psi_m$. For example, $\hat{\Psi}_k$ could be "squares and blue circles" while $\Psi_m$ could be "squares and blue or orange circles". Because this excessive specificity could result in the user arriving at conclusions that are too narrow, we need to incorporate some notion of recall into the evaluation.

**Blindspot Recall.** One way to incorporate recall is using the proportion of $\Psi_m$ that $\hat{\Psi}_k$ covers:

$$\text{BR}_{\text{naive}}(\hat{\Psi}_k, \Psi_m) = \frac{|\hat{\Psi}_k \cap \Psi_m|}{|\Psi_m|} \tag{5.3}$$

We relax this definition by allowing $\Psi_m$ to be covered by the union of the $\hat{\Psi}_k$ that belong to it:

$$\text{BR}(\hat{\mathbf{\Psi}}, \Psi_m) = \frac{\left| \left( \bigcup_{\hat{\Psi}_k : \text{BP}(\hat{\Psi}_k, \Psi_m) \geq \lambda_p} \hat{\Psi}_k \right) \cap \Psi_m \right|}{|\Psi_m|} \tag{5.4}$$

Then, we say that $\hat{\mathbf{\Psi}}$ *covers* $\Psi_m$ if, for some threshold $\lambda_r$:

$$\text{BR}(\hat{\mathbf{\Psi}}, \Psi_m) \geq \lambda_r \tag{5.5}$$

We do this because "squares and blue circles" and "squares and orange circles" belong to and jointly cover "squares and blue or orange circles." So, if a BDM returns both, a user could combine them to arrive at the correct conclusion.

**Discovery Rate (DR).** We define the DR of $\hat{\mathbf{\Psi}}$ and $\mathbf{\Psi}$ as the fraction of the $\Psi_m$ that $\hat{\mathbf{\Psi}}$ covers:

$$\text{DR}(\hat{\mathbf{\Psi}}, \mathbf{\Psi}) = \frac{1}{M} \sum_m \mathbb{1}(\text{BR}(\hat{\mathbf{\Psi}}, \Psi_m) \geq \lambda_r) \tag{5.6}$$

**False Discovery Rate (FDR).** When the DR is non-zero, we define FDR of $\hat{\mathbf{\Psi}}$ and $\mathbf{\Psi}$ as the fraction of the $\hat{\Psi}_k$ that do not belong to any of the $\mathbf{\Psi}$:[3]

$$\text{FDR}(\hat{\mathbf{\Psi}}, \mathbf{\Psi}) = \frac{1}{K} \sum_k \mathbb{1}(\max_m BP(\hat{\Psi}_k, \Psi_m) < \lambda_p) \tag{5.7}$$

Note that it is impossible to calculate the FDR without the complete set of true blindspots. While `SpotCheck` gives us this knowledge, it is generally not available.

[3] While calculating DR, we may only need the top-$u$ items of $\hat{\mathbf{\Psi}}$. As a result, we only calculate the FDR over those top-$u$ items. This prevents the FDR from being overly pessimistic when we intentionally pick $K$ too large in our experiments. However, it is unclear what value of $u$ to use for ECs that have a DR of zero, so we exclude these ECs from our FDR analysis.

## 5.4 `PlaneSpot`: A simple BDM based on Dimensionality Reduction

In this section, we define `PlaneSpot`. As shown in Table 5.1, `PlaneSpot` uses the most common choices for the image representation (*i.e., $f$*'s own representation) and the hypothesis class (*i.e.,* gaussian kernels). `PlaneSpot` also uses standard techniques to learn a model from that hypothesis class. As a result, the most interesting aspect of `PlaneSpot`'s design is that it finds blindspots using a 2D representation. We start by defining some additional notation and then explain `PlaneSpot`'s choice for each of the high-level design choices made by a BDM.

**Notation.** Suppose that we want to find $f$'s blindspots for a class, $c$, and let $D^c$ be the set of images from $D$ that belong to $c$. Further, suppose that we have divided $f$ into two parts: $g : X \to \mathbb{R}^d$, which extracts $f$'s representation of an image (*i.e.,* its penultimate layer activations), and $h : \mathbb{R}^d \to [0, 1]$, which gives $f$'s predicted confidence for $c$.

**Image Representation.** We use $g$ to extract $f$'s representation for $D^c$, $G = g(D^c) \in \mathbb{R}^{n\times d}$, and $h$ to extract $f$'s predicted confidences for $D^c$, $H = h(G) \in [0, 1]^{n\times 1}$. Note that, because all of the images in $D^c$ belong to $c$, entries of $H$ closer to $1$ denote higher confidence in the true class.

**Dimensionality Reduction.** We use $G$ to train scvis [30], which combines the objective functions of tSNE and an autoencoder, in order to learn a 2D representation of $f$'s representation, $s : \mathbb{R}^d \to \mathbb{R}^2$. Then, we use $s$ to get the 2D representation of $D^c$, $S = s(G) \in \mathbb{R}^{n\times 2}$. Finally, we normalize the columns of $S$ to be in $[0, 1]$, $\bar{S} \in [0, 1]^{n\times 2}$.

**Hypothesis Class.** We want `PlaneSpot` to be aware of both the representation of $D^c$, $\bar{S}$, and $f$'s predicted confidences for $D^c$, $H$, so we combine them into a single representation, $R = [\bar{S}; w \cdot H] \in \mathbb{R}^{n\times 3}$ where $w$ is a hyperparameter controlling the relative weight of these components. Then, we pass $R$ to a Gaussian Mixture Model clustering algorithm, where the number of clusters is chosen using the Bayesian Information Criteria. Finally, we return those clusters in order of decreasing importance, which we define using the product of their error rate and the number of errors in them.

## 5.5 Experiments

In Section 5.5.1, we use `SpotCheck` to run a series of controlled experiments using synthetic data. In Section 5.5.2, we demonstrate how to setup semi-controlled experiments to validate that the findings from `SpotCheck` generalize to settings with real data.

### 5.5.1 Synthetic Data Experiments

We use `SpotCheck` to generate 100 ECs whose datasets have 6-8 semantic features and whose models have 1-3 blindspots defined with 5-7 features. We evaluate three recent BDMs: Spotlight [26], Barlow [94], Domino [33], and `PlaneSpot`. We give all BDMs the positive examples (*i.e.,* images with squares) from the test set and limit them to returning 10 hypothesized blindspots. We use a held-out set of 20 ECs to tune each BDM's hyperparameters (see Appendix D.3). We use $\lambda_p = \lambda_r = 0.8$ for our metrics.

**Table 5.2:** Average BDM DR and FDR with their standard errors across 100 `SpotCheck` ECs.

| Method | DR | FDR |
|---|---|---|
| Barlow | 0.43 (0.04) | 0.03 (0.01) |
| Spotlight | 0.79 (0.03) | 0.09 (0.01) |
| Domino | 0.64 (0.04) | 0.07 (0.01) |
| `PlaneSpot` | **0.88 (0.03)** | **0.02 (0.01)** |

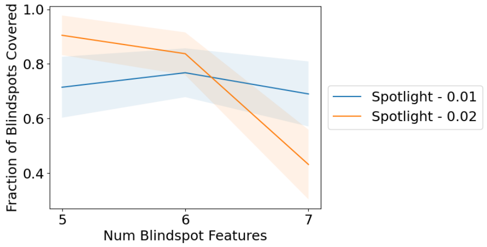


**Figure 5.2:** The fraction of blindspots covered (with shaded 95% CIs) for blindspots defined using $5, 6$, and $7$ features using 2 different Spotlight hyperparameter choices.

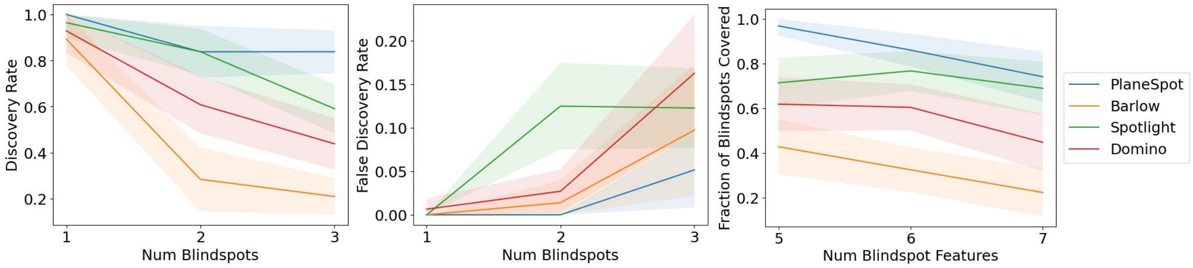


**Figure 5.3:** (Left/Center) Average BDM DR/FDR (with shaded 95% confidence intervals) for ECs that have $1$, $2$, and $3$ blindspots. (Right) The fraction of blindspots covered, across the individual blindspots from the ECs, for blindspots defined using $5, 6$, and $7$ features.

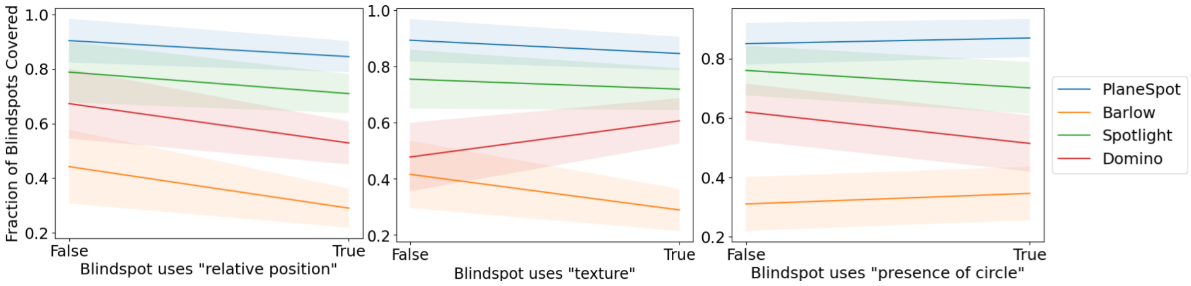


**Figure 5.4:** The fraction of blindspots covered (with shaded 95% confidence intervals) that are or are not defined with (Left) the "relative position" feature (whether the square appears above the centerline of the image), (Center) a "texture" feature (whether any object in the image has vertical stripes), or (Right) the presence of a circle (whether there is a circle in the image).

**Overall Results.** Table 5.2 shows the DR and FDR results averaged across all $100$ ECs. We observe that `PlaneSpot` has the highest DR and that `PlaneSpot` and Barlow have a lower FDR than Spotlight and Domino. In Appendix D.4, we take a deeper look at why these BDMs are failing and conclude that a significant portion of all of their failures can be explained by their tendency to merge multiple true blindspots into a single hypothesized blindspot.

**Identifying factors that influence BDM performance.** We study two types of factors: *per-configuration factors*, which measure properties of the dataset (*e.g.,* how complex is it?) or of the model (*e.g.,* how many blindspots does it have?), and *per-blindspot factors*, which measure properties of a blindspot (*e.g.,* is it defined with this feature?). For per-configuration factors, we average DR and FDR across the ECs. For per-blindspot factors, we report the fraction of blindspots covered averaged across each individual blindspot from the ECs (see Equation 5.5).

**The number of blindspots matters.** In Figure 5.3 (Left), we plot the average DR for ECs with $1, 2,$ and $3$ blindspots. Average DR decreases for all methods as the number of blindspots increases. Figure 5.3 (Center) shows that FDR increases as the number of blindspots increases. Together these observations show that BDMs perform worse in settings with multiple blindspots, which is particularly significant because past evaluations have primarily focused on settings with one blindspot.

**The specificity of blindspots matters.** In Figure 5.3 (Right), we plot the fraction of blindspots covered for blindspots defined using $5, 6,$ and $7$ features. With the exception of Spotlight, all of these methods are less capable of finding more specific/less frequently occurring blindspots.

**The features that define a blindspot matter.** In Figure 5.4, we plot the fraction of blindspots covered for blindspots that either are or are not defined using various features. In general, we observe that the performance of these BDMs is influenced by the types of features used to define a blindspot (*e.g.,* the presence of spurious objects, color or texture information, background information). For example, all methods are less likely to find blindspots defined using the "relative position" feature. Interestingly, BDMs that use the model's representation for their image representation (*i.e.,* `PlaneSpot` and Spotlight) are less sensitive to the features that define a blindspot than those that use an external model's representation (*i.e.,* Barlow and Domino).

**Hyperparameters matter.** We make two observations about BDM hyperparameters that, jointly, suggest that it is critical that future BDMs provide ways to tune their hyperparameters. First, in Appendix D.3, we observe that many BDMs have significant performance differences when we vary their hyperparameters. However, hyperparameter tuning is harder in real settings than in these experiments because there is no information about the true blindspots to use. Second, in Figure 5.2, we observe that two hyperparameter settings that perform nearly identically on average exhibit significantly different performance at finding blindspots defined using differing numbers of features. This suggests that there may not be a single best hyperparameter choice to find all of the blindspots in a single model, which could contain multiple blindspots of different specificity or size.

### 5.5.2 Real Data Experiments

We demonstrate how to design semi-controlled experiments using the COCO dataset [60] to test whether or not some of the findings observed using SpotCheck generalize to settings with real data. Despite the fact COCO is a fairly large dataset with extensive annotations, it does not have enough metadata to test all of the findings from SpotCheck (*e.g.,* we cannot induce blindspots that depend on the texture of an object). Therefore, we study the following questions using real data:

- Are BDMs still less effective for models with more true blindspots?
- Are BDMs still less effective at finding more specific blindspots?
- Will the 2D image representation used by PlaneSpot still be effective?

Notice that means we we are interested in two factors: the *number* of blindspots in a model and of the *specificity* of those blindspots. Then, in order to try to estimate their influence on BDM performance, we use the same strategy as SpotCheck and generate a set of ECs where we randomize these factors independently of one another. We provide details of how we generate these ECs in Appendix D.5 and show an example EC in Figure 5.5.

However, one key difference from our synthetic experiments is that when we use real data, we cannot guarantee that the model will learn the blindspots that we try to induce in it. Then, if the model fails to learn the intended blindspots in a non-random way, the factors that we want to investigate may be correlated and, consequently, their effects may be confounded, which makes it difficult to estimate their individual effects. Table 5.3 shows that this confounding occurs between the effects of the number and the specificity of the blindspots in the COCO ECs. To eliminate this confounding, we generate a new set of ECs for each factor where we hold the other confounding factor constant.

We call the set of ECs, where all of the factors are chosen independently, the *general pool* because it covers a wider range of possible ECs, which is better for measuring overall BDM performance. We call this new set of ECs the *conditioned pool* because it is generated by conditioning the distribution used to generate the general pool on specific confounding factors. Because blindspot discovery is harder with real data, we use more lenient thresholds for our metrics and set $\lambda_p = \lambda_r = 0.5$.

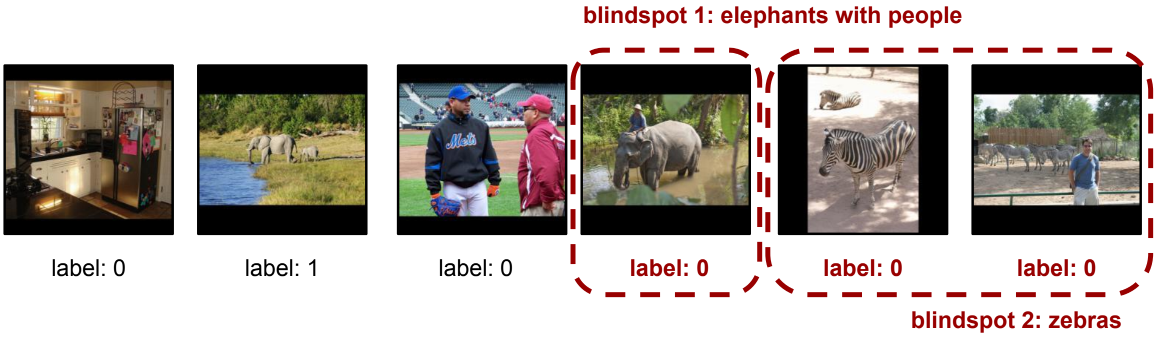


**Figure 5.5:** Images sampled from an example EC created using COCO. In this example, the task is predict to detect whether an animal is present. The EC has two true blindspots: "elephants with people," which is more-specific because it is defined using two objects, and "zebras," which is less-specific because it is defined using only one object.

**Table 5.3:** The effect of number of blindspots is confounded with the effect of the specificity of those blindspots: as we try to induce more blindspots in the model, the model is less likely to learn more specific blindspots. Overall, the model learned the intended blindspots in 65 of the 90 ECs in the general pool.

| Num Blindspots | Fraction of successfully induced Blindspots that are more specific |
|---|---|
| 1 | 0.54 |
| 2 | 0.47 |
| 3 | 0.40 |

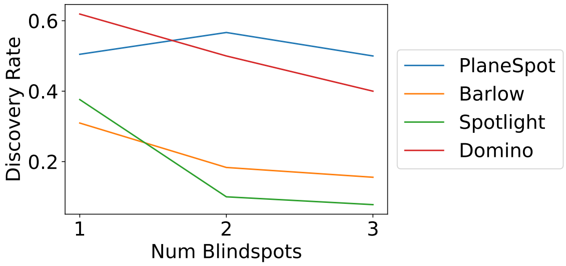


**Figure 5.6:** When we condition on only having less specific blindspots, we see that BDM performance decreases as we increase the number of blindspots. These results are based on 82 ECs where the model learned the intended blindspots.

**Table 5.4:** For the 65 ECs where the model learned the intended blindspots, we report BDM DR with its standard error. We only have knowledge of the blindspots that we induced, so the DR is calculated relative to those blindspots and we cannot calculate the FDR.

| Method | DR |
|---|---|
| Barlow | 0.09 (0.03) |
| Spotlight | 0.14 (0.04) |
| Domino | 0.38 (0.05) |
| `PlaneSpot` | **0.48** (0.05) |

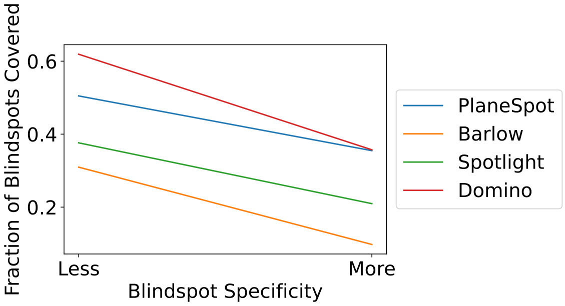


**Figure 5.7:** When we condition on only having a single blindspot, we see that BDMs are less effective at finding more specific blindspots. These results are based on 70 ECs where the model learned the intended blindspots.

**Results.** Overall, all of the findings that we observed using `SpotCheck` generalized to these real data experiments. In Figure 5.6, using the conditioned pool, we observe that BDM performance generally decreases as the number of blindspots increases. In Figure 5.7, using the conditioned pool, we observe that these BDMs are less able to find blindspots that are more specific. In Table 5.4, using the general pool, we observe that `PlaneSpot` is still effective on real data. These results provide evidence that results from `SpotCheck` may generalize to a wider range of more realistic settings. Additionally, in Appendix D.6, we include qualitative examples of the 2D representation used by `PlaneSpot` for models trained on benchmark datasets.

**Interpreting these results.** While these results are promising, there are general challenges associated with running semi-controlled experiments on real data that influence the interpretation of these results (or the results of any similar experiments). At a high-level, these challenges all add noise (or bias) to the results.

First, models trained on real data probably have naturally occurring blindspots (*i.e.,* blindspots that we did not intend the model to learn) that may influence the results. One example of how they do this is that we do not know how many hypothesized blindspots to ask a BDM to return because we do not know how many true blindspots (including both the naturally occurring ones and the ones we induced) a model has.

Second, the chosen BDM hyperparameters are probably sub-optimal because we cannot tune them for a specific EC. Because the set of ECs from the synthetic data experiments is relatively homogeneous, it is reasonable to use a single set of hyperparameters for all of those ECs. However, this is not the case for our COCO experiments.

Third, the results may be biased in potentially unknown ways towards certain BDMs whenever some of the models in the set of ECs fail to learn the intended blindspots. For example, our general pool of ECs contains more less-specific blindspots than more-specific blindspots, which will bias the results towards BDMs that are more effective at finding less-specific blindspots. Further, there are probably other biases that we cannot identify because we lack the metadata needed to measure them.

Fourth, there can be false positives in verifying that a model actually learned an intended blindspot. This is because there could be a sub-blindspot (that we may not have the metadata to define) within that intended blindspot that is actually causing the model to perform significantly worse on the intended blindspot.

### 5.6 Evaluating BDMs without Knowledge of the Model's True Blindspots

In this section, we shift our focus and consider evaluating BDMs in real settings, where we do not know the model's true blindspots. Recall from Section 5.2, that there are two problems with existing approaches for doing this. Specifically, they do not consider whether or not:

1. A hypothesized blindspot is coherent (*e.g.,* a random sample of misclassified images has high error but may not match a single semantically meaningful description).
2. The model's performance on a hypothesized blindspot is representative of the model's performance on other similar images (*e.g.,* suppose that the model has a 90% accuracy on images of "squares and blue circles"; then, by returning the 10% of such images that are misclassified, a BDM could mislead us into believing that the model has a 0% accuracy on this type of image).

Additionally, there is a third, more general, problem with approaches for evaluating BDMs:

3. These evaluations only work for BDMs and cannot be used to compare BDMs to other methods (*e.g.,* methods from explainable machine learning). Largely, this is because they are designed around the fact that BDMs return hypothesized blindspots as sets of images.

To address these problems, we propose a two-step user-based evaluation process:

1. Using the output of the method, the user writes a text description that describes the images that belong to their hypothesized blindspot.
   - This address Problem 1 because, if the hypothesized blindspots returned by a BDM are not coherent, the user will not be able to describe them.
   - This addresses Problem 3 because the user is converting the output of the method into a standardized format.
   - Throughout Appendix D.6, we essentially did this step ourselves for the figure captions.
2. Gather new images that match that text description and use those images to measure the model's performance on the user's hypothesized blindspot.

- This addresses Problem 2 because these newly gathered images were chosen without direct knowledge of the model's performance on them and are chosen only based on the user's text description.
- Note that these images could come from an external source or they could be taken from the same set of images that the method was given.
- Interestingly, these images could also be re-used as a type of "unit test" to compare models.
- In Appendix D.6.2, we did this step ourselves with images from Google Images.

### 5.7 Conclusion

In this work, we introduced `SpotCheck`, a synthetic evaluation framework for BDMs, and `PlaneSpot`, a simple BDM that uses a 2D image representation. Using `SpotCheck`, we identified factors that influence BDM performance and two technical challenges for future work on developing new BDMs to address (*i.e.,* providing a practical way to tune BDM hyperparameters and preventing BDMs from merging multiple true blindspots into a single hypothesized blindspot). We found that `PlaneSpot` outperforms existing BDMs on both synthetic and real data.

Beyond the insights made in the experiments that we ran, we believe that our evaluation workflow has laid further groundwork for a more rigorous science of blindspot discovery. Researchers interested in benchmarking their own BDM or answering scientific questions about blindspot discovery can first use `SpotCheck` (or harder variations of it) to run controlled experiments using synthetic data. Next, researchers can also set up semi-controlled experiments using real data. While both harder and noisier (in part, because of the challenges that we identified), these experiments are also more realistic. A natural open direction for future work is how to evaluate BDMs in more realistic settings where we have no knowledge of the model's true blindspots (we described one potential approach in Section 5.6).

We note that this workflow is particularly well suited to identifying factors that influence BDM performance and that understanding these factors can benefit both researchers and practitioners using BDMs. For researchers, understanding these factors is essential to running fair and informative experiments (*e.g.,* only evaluating BDMs in settings that only have one blindspot, which we have identified as being easier, may misrepresent how BDMs perform in more realistic settings). For practitioners, understanding these factors may help them use any prior knowledge they may have to select an effective BDM for their problem. Two potential factors that future work could consider are: First, how does the methodology used to induce blindspots (*i.e.,* training with mislabeled images) influence BDM performance? Second, for models trained on real data, how different are the naturally occurring blindspots from those that we have the metadata to artificially induce and how do those differences influence BDM performance?

# Chapter 6

# Conclusion

To conclude this thesis, we start by summarizing its goal, approach, and contributions. Then we finish by discussing a few relevant directions for future work.

## 6.1 Summary

One of the core problems facing explainability is the fact that it has primarily focused on identifying the patterns that a model uses to make its predictions and has mostly ignored showing that doing so is useful for addressing specific potential problems with that model. This is problematic in two ways. From an applied perspective, it makes finding a method that actually addresses your application of interest challenging. From a research perspective, it makes comparing explainability methods challenging because it is difficult to directly compare two sets of patterns and argue that one is better than the other.

In this thesis, we suggest addressing this problem by following an application-based pipeline for explainability. By making it explicit what application is being addressed, this pipeline makes it easier for practitioners to find useful methods. By defining a clear and testable hypothesis about what notion of utility will increase if we are successful, this pipeline provides a way to quantitatively compare explainability methods.

We provide evidence that this pipeline is effective by using it to study several different applications. In the process of doing so, we make contributions towards each of those applications in terms of new explainability methods and evaluations, which we will now summarize.

For User Interaction, we want to help end-users change their profile so that a model making predictions about them will give them their desired prediction in the future. We consider using local approximation explanations and measure the quality of these explanations using fidelity and stability. Then, we introduce a regularizer that allows us to empirically explore the trade-off between fidelity and model accuracy. Finally, we demonstrate that users are able to more easily change the models prediction when these explanations have higher fidelity.

For Knowledge Discovery, we want to help domain-experts discover new knowledge from their models. We introduce a new type of explanation, global counterfactual explanations, and an algorithm for finding such explanations for unsupervised clustering problems in low dimensional representations. We demonstrate that these explanations are able to identify both the structure that we explicitly added to the data and the structure that we would expect to be present in the data based on domain knowledge.

For Spurious Patterns, we want to help model developers debug their models by finding and fixing spurious patterns where the model relies on a spurious feature to make predictions. For finding, we use global counterfactual explanations and demonstrate that these explanations are able to find a wider range of spurious patterns than past methods. For fixing, we introduce a new way to measure how sensitive a model is to distribution shifts related to a spurious pattern and show that a smarter form of data augmentation is more effective at fixing spurious patterns than past methods.

For Blindspots, we want to help model developers debug their models by finding blindspots where the model performs worse on a semantically meaningful subset of the data (this encompasses a wider range of systemic errors than those induced by spurious patterns). We introduce a new method for finding blindspots, which we show to be more effective than past methods, and a new evaluation framework, which addresses limitations of past evaluations. Further, we identify promising directions for future methods and evaluations to explore.

However, this pipeline is not without limitations. First, it is an abstract template to follow rather than a prescriptive set of concrete steps to take. Second, it tends to produce conclusions that apply to a relatively narrow set of conditions.

## 6.2 Future Work

There are three different directions for future work that we will discuss. First, identifying shared approaches between different applications. Second, building on the specific results that this pipeline produces in order to identify more general trends. Third, further exploring model debugging via finding and fixing blindspots.

**Identifying shared approaches between different applications.** One of the limitations of the application-based pipeline for explainability is that it describes *what* we want to do, but not necessarily *how* to do it. As a result, we want to make following this pipeline easier by recognizing and utilizing the fact that there will be approaches for each of the steps of this pipeline that can be used for multiple applications.

Looking at how other work has done evaluation (even if they have not followed this pipeline in its entirety), we can see a few such common approaches:

- Naturally, Faithfulness Evaluations are commonly used across different applications, because they are specific to the explainability method and not the application. For example, variations of fidelity or stability are frequently used to measure the quality of local approximation based explanations (see Chapter 2 and [71, 119]).
- For applications related to model debugging, a common approach for Simulation Evaluations is to induce bugs into the model by either creating correlations in the training data or by mislabeling specific parts of the training data (see Chapters 4 and 5 as well as [33, 45, 83]).

- For applications related to model debugging, a common approach for Usefulness Evaluations is to gather fresh data to assess whether a hypothesized bug is real or a method for fixing a bug works (see see Chapters 4 and 5 as well as [34, 77]).
- Finally, we note that using (semi)synthetic data to control the model's behavior and gain ground-truth to compare against is used in a wide range of applications, ranging from Knowledge Discovery to a variety of versions of model debugging (see Chapters 3 and 5 as well as [2, 19, 21, 45, 124]). In general, we argue that this is useful because these (semi)synthetic settings are easier than real settings and it seems unlikely that a method that does not work in an easier setting will start working in a harder setting.

**Identifying more general trends.** While evaluating the usefulness of a method for a specific application is an important first step, doing so leads to considering applications under relatively narrow conditions and in isolation. As a result, one way to help make these results more actionable is to try to identify more general trends by asking questions about the conditions these results were produced under and how different applications relate to one another.

One question to ask is: what potentially relevant "experimental factors" were held fixed for a particular result? For example, consider the results from Chapter 2 where we showed that fidelity is a good proxy metric for the ability of a user to change the model's input to produce a specific change in the model's prediction. Here we held both the "complexity of the explanation" and the "expertise of the users" fixed. As a result, we do not know if this result would generalize if we were to replace the linear approximation with a generalized additive model (which should have higher fidelity but is also more complex). In all likelihood, the results of this change would depend on the expertise of the users. As a result, as the number of identified relationships between various Faithfulness Evaluations and Usefulness Evaluations grows, it will be important to consider these other "factors" (both to help resolve potentially conflicting results and to help practitioners find effective methods).

Another question to ask is: what is required for success? For example, consider the difference between Knowledge Discovery and finding Spurious Patterns. If a pattern that the model is using is true about the world and was previously unknown, we say that we have discovered new knowledge. If a pattern that the model is using is not true about the world, we say that we have found a spurious pattern in the model. Interestingly, from the perspective of an explainability method, these two applications are functionally the same (modulo any assumptions made about the model or the data). As a result, when considering a new application, it is important to consider how that application may relate to other applications that have previously been studied in order to identify more general trends. Naturally, similar applications are also more likely to have shared approaches between them.

**Debugging via finding and fixing Blindspots.** There are many interesting questions to ask along these lines, some of which are:

What is the correct formalization for fixing blindspots? To start, we want to measure $T$, which is the model's performance on the test set, and $\{B_i\}_{i=1}^m$, which is the model's performance on a set of identified blindspots. Then the question is, what function of $T$ and the $B_i$ should we use to measure the effectiveness of a method for fixing blindspots? For example, if we used the formalization from Group Distributionally Robust Optimization, we would use

$min(T, B_1, \ldots, B_m)$. Alternatively, would we be better off measuring performance on some chosen “challenge datasets” (as done in Chapter 4)?

Relatedly, what happens when multiple rounds of finding and fixing blindspots are carried out? Will the same blindspot discovery method find more blindspots, $\{B_i\}_{i=m+1}^{n}$, or do these methods find what they are going to find during the first round? When we go to fix $\{B_i\}_{i=m+1}^{n}$, do we need to enforce that the model not “forget” about $\{B_i\}_{i=1}^{m}$? Lastly, is there a metric that can tell us when this process has converged (*e.g.,* if the model’s loss has the same mean and variance across its representation, then it seems unlikely that the model has more blindspots)?

What are the trade-offs made when choosing what data to pass to a blindspot discovery method? There are a few things to consider when answering this question. First, how diverse is the data? Intuitively, we want the data to be as diverse as possible (while still being relevant to the predictive task) because this will allow us to measure the model’s performance in as many situations as possible. Because collecting diverse data is hard, there is a question of whether or not we can use artificially generated data (*e.g.,* images with objects masked or that were generated by a GAN) or unlabeled data. However, both of these options present their own challenges. Second, was the data used to train the model? If it was, the model’s performance estimates on it are probably biased, so we may need to adjust for that.

Finally, what types of blindspots are identifiable in the model’s own representation? This is influenced by the spatial arrangement (in the model’s representation) of the points in a blindspot relative to the locations of points outside of the blindspot and relative to the model’s decision boundary. For example, if the model should make different predictions for two different groups of images but also maps those two groups to the same part of its representation, then the resulting blindspot will not be identifiable in that representation. While we are not aware of any examples of this happening, this could be because blindspot discovery methods cannot find this problem or because we typically study blindspot discovery using fairly simple classification benchmarks.

# 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*, pages 9525–9536, 2018.

[2] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. Debugging tests for model explanations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 700–712. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/075b051ec3d22dac7b33f788da631fd4-Paper.pdf`.

[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*, 2022. URL `https://openreview.net/forum?id=xNOVfCCvDpM`.

[4] Vedika Agarwal, Rakshith Shetty, and Mario Fritz. Towards causal vqa: Revealing and reducing spurious correlations by invariant and covariant semantic editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9690–9698, 2020.

[5] Maruan Al-Shedivat, Avinava Dubey, and Eric P Xing. Contextual explanation networks. *arXiv preprint arXiv:1705.10301*, 2017.

[6] David Alvarez-Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

[7] David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

[8] Agathe Balayn, Natasa Rikalo, Christoph Lofi, Jie Yang, and Alessandro Bozzon. How can explainability methods be used to support bug identification in computer vision models? In *CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2022.

[9] Natalie Bello and Lori Mosca. Epidemiology of coronary heart disease in women. *Progress in cardiovascular diseases*, 46(4):287–295, 2004.

[10] Homanga Bharadhwaj, De-An Huang, Chaowei Xiao, Anima Anandkumar, and Animesh Garg. Auditing ai models for verified deployment under semantic specifications. *arXiv preprint arXiv:2109.12456*, 2021.

[11] Adam Bloniarz, Ameet Talwalkar, Bin Yu, and Christopher Wu. Supervised neighborhoods for distributed nonparametric regression. In *Artificial Intelligence and Statistics*, pages 1450–1459, 2016.

[12] Thomas Blumensath. Compressed sensing with nonlinear observations and related nonlinear optimization problems. *IEEE Transactions on Information Theory*, 59(6): 3466–3474, 2013.

[13] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[14] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. Fairvis: Visual analytics for discovering intersectional bias in machine learning. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 46–56. IEEE, 2019.

[15] Emmanuel J Candès et al. Compressive sampling. In *Proceedings of the international congress of mathematicians*, volume 3, pages 1433–1452. Madrid, Spain, 2006.

[16] Rich Caruana et al. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM, 2015.

[17] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[18] Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. Counterfactual samples synthesizing for robust visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10800–10809, 2020.

[19] Valerie Chen, Nari Johnson, Nicholay Topin, Gregory Plumb, and Ameet Talwalkar. Use-case-grounded simulations for explanation evaluation. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=48Js-sP8wnv`.

[20] Valerie Chen, Jeffrey Li, Joon Sik Kim, Gregory Plumb, and Ameet Talwalkar. Interpretable machine learning: Moving from mythos to diagnostics. *Queue*, 19(6):28–56, 2022.

[21] Valerie Chen, Muyu Yang, Wenbo Cui, Joon Sik Kim, Ameet Talwalkar, and Jian Ma. Best practices for interpretable machine learning in computational biology. *bioRxiv*, 2022.

[22] Yining Chen, Colin Wei, Ananya Kumar, and Tengyu Ma. Self-training avoids using spurious features under domain shift. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21061–21071. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/f1298750ed09618717f9c10ea8d1d3b0-Paper.pdf`.

[23] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. Slice finder: Automated data slicing for model validation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1550–1553. IEEE, 2019.

[24] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

[25] Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Exchanging lessons between algorithmic fairness and domain generalization. *arXiv preprint arXiv:2010.07249*, 2020.

[26] Greg d'Eon, Jason d'Eon, James R Wright, and Kevin Leyton-Brown. The spotlight: A general method for discovering systematic errors in deep learning models. *arXiv preprint arXiv:2107.00758*, 2021.

[27] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[28] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, pages 592–603, 2018.

[29] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. Model agnostic contrastive explanations for structured data, 2019.

[30] Jiarui Ding, Anne Condon, and Sohrab P Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature communications*, 9(1):2002, 2018.

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

[32] Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. Learning credible deep neural networks with rationale regularization. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 150–159. IEEE, 2019.

[33] Sabri Eyuboglu, Maya Varma, Khaled Kamal Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Re. Domino: Discovering systematic errors with cross-modal embeddings. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=FPCMqjI0jXN`.

[34] Irena Gao, Gabriel Ilharco, Scott Lundberg, and Marco Tulio Ribeiro. Adaptive testing of computer vision models. *arXiv preprint arXiv:2212.02774*, 2022.

[35] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. *arXiv preprint arXiv:1710.10547*, 2017.

[36] Karan Goel, Albert Gu, Yixuan Li, and Christopher Re. Model patching: Closing the subgroup performance gap with data augmentation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=9YlaeLfuhJF`.

[37] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual visual explanations. In *International Conference on Machine Learning*, pages 2376–2384, 2019.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[39] Lisa Anne Hendricks, Kaylee Burns, Kate Saenko, Trevor Darrell, and Anna Rohrbach. Women also snowboard: Overcoming bias in captioning models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 771–787, 2018.

[40] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Polo Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE transactions on visualization and computer graphics*, 26(1):1096–1106, 2019.

[41] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on knowledge and data engineering*, 16(11):1370–1386, 2004.

[42] Jungseock Joo and Kimmo Kärkkäinen. Gender slopes: Counterfactual fairness for computer vision models by attribute manipulation. In *Proceedings of the 2nd International Workshop on Fairness, Accountability, Transparency and Ethics in Multimedia*, pages 1–5, 2020.

[43] Jacob Kauffmann, Malte Esders, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. From clustering to cluster explanations via neural networks. *arXiv preprint arXiv:1906.07633*, 2019.

[44] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[45] Joon Sik Kim, Gregory Plumb, and Ameet Talwalkar. Sanity simulations for saliency methods. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[46] Michael P Kim, Amirata Ghorbani, and James Zou. Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 247–254, 2019.

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

[48] Dmitry Kobak and Philipp Berens. The art of using t-sne for single-cell transcriptomics. *bioRxiv*, page 453449, 2018.

[49] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[50] 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*, pages 5338–5348. PMLR, 2020.

[51] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[52] Himabindu Lakkaraju and Osbert Bastani. " how do i fool you?" manipulating user trust via misleading black box explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 79–85, 2020.

[53] Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 3db: A framework for debugging computer vision models. *arXiv preprint arXiv:2106.03805*, 2021.

[54] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[55] Guang-He Lee, David Alvarez-Melis, and Tommi S Jaakkola. Game-theoretic interpretability for temporal modeling. *arXiv preprint arXiv:1807.00130*, 2018.

[56] Guang-He Lee, Wengong Jin, David Alvarez-Melis, and Tommi Jaakkola. Functional transparency for structured data: a game-theoretic approach. In *International Conference on Machine Learning*, pages 3723–3733, 2019.

[57] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*, 2016.

[58] Jeffrey Li, Vaishnavh Nagarajan, Gregory Plumb, and Ameet Talwalkar. A learning theoretic perspective on local explainability. In *International Conference on Learning Representations*, 2020.

[59] Zujie Liang, Weitao Jiang, Haifeng Hu, and Jiaying Zhu. Learning to contrast the counterfactual samples for robust visual question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3285–3292, 2020.

[60] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[61] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

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

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

[64] Usman Mahmood, Robik Shrestha, David DB Bates, Lorenzo Mannelli, Giuseppe Corrias, Yusuf Emre Erdi, and Christopher Kanan. Detecting spurious correlations

with sanity tests for artificial intelligence guided radiology systems. *Frontiers in digital health*, page 85, 2021.

[65] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. In *International Conference on Learning Representations*, 2021.

[66] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*, 2019.

[67] Elias Chaibub Neto. Counterfactual confounding adjustment for feature representations learned by deep models: with an application to image classification tasks. *arXiv preprint arXiv:2004.09466*, 2020.

[68] Luke Oakden-Rayner, Jared Dunnmon, Gustavo Carneiro, and Christopher Ré. Hidden stratification causes clinically meaningful failures in machine learning for medical imaging. In *Proceedings of the ACM conference on health, inference, and learning*, pages 151–159, 2020.

[69] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[70] Julia Peyre, Ivan Laptev, Cordelia Schmid, and Josef Sivic. Weakly-supervised learning of visual relations. In *ICCV*, 2017.

[71] Gregory Plumb, Denali Molitor, and Ameet Talwalkar. Model agnostic supervised local explanations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2520–2529, 2018.

[72] Gregory Plumb, Maruan Al-Shedivat, Ángel Alexander Cabrera, Adam Perer, Eric Xing, and Ameet Talwalkar. Regularizing black-box models for improved interpretability. *Advances in Neural Information Processing Systems*, 33, 2020.

[73] Gregory Plumb, Jonathan Terhorst, Sriram Sankararaman, and Ameet Talwalkar. Explaining groups of points in low-dimensional representations. In *International Conference on Machine Learning*, pages 7762–7771. PMLR, 2020.

[74] Gregory Plumb, Nari Johnson, Ángel Cabrera, Marco Tulio Ribeiro, and Ameet Talwalkar. Evaluating systemic error detection methods using synthetic images. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*, 2022. URL `https://openreview.net/forum?id=-NMPXRQlfc4`.

[75] Gregory Plumb, Marco Tulio Ribeiro, and Ameet Talwalkar. Finding and fixing spurious patterns with explanations. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=whJPugmP5I.

[76] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems*, pages 13847–13856, 2019.

[77] Marco Tulio Ribeiro and Scott Lundberg. Adaptive testing and debugging of nlp models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3253–3267, 2022.

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

[79] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[80] Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International Conference on Machine Learning*, pages 8116–8126. PMLR, 2020.

[81] Alexis Ross, Himabindu Lakkaraju, and Osbert Bastani. Learning models for actionable recourse. *Advances in Neural Information Processing Systems*, 34:18734–18746, 2021.

[82] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2662–2670, 2017.

[83] Shiori Sagawa*, Pang Wei Koh*, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxGuJrFvS.

[84] Axel Sauer and Andreas Geiger. Counterfactual generative networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=BXewfAYMmJw.

[85] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[86] Juliet Popper Shaffer. Multiple hypothesis testing. *Annual review of psychology*, 46(1): 561–584, 1995.

[87] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In *NeurIPS*, 2020.

[88] Karthik Shekhar, Sylvain W Lapan, Irene E Whitney, Nicholas M Tran, Evan Z Macosko, Monika Kowalczyk, Xian Adiconis, Joshua Z Levin, James Nemesh, Melissa Goldman, et al. Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. *Cell*, 166(5):1308–1323, 2016.

[89] Rakshith Shetty, Bernt Schiele, and Mario Fritz. Not using the car to see the sidewalk–quantifying and controlling the effects of context in classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8218–8226, 2019.

[90] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

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

[92] Chandan Singh, W James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*, 2018.

[93] Krishna Kumar Singh, Dhruv Mahajan, Kristen Grauman, Yong Jae Lee, Matt Feiszli, and Deepti Ghadiyaram. Don't judge an object by its context: Learning to overcome contextual bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11070–11078, 2020.

[94] Sahil Singla, Besmira Nushi, Shital Shah, Ece Kamar, and Eric Horvitz. Understanding failures of deep networks via robust feature extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12853–12862, 2021.

[95] Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by progressive exaggeration. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=H1xFWgrFPS`.

[96] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[97] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *Advances in Neural Information Processing Systems*, 33:19339–19352, 2020.

[98] James C Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins apl technical digest*, 19(4):482–492, 1998.

[99] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

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

[101] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644*, 2020.

[102] Damien Teney, Ehsan Abbasnedjad, and Anton van den Hengel. Learning what makes a difference from counterfactual examples and gradient supervision. *arXiv preprint arXiv:2004.09034*, 2020.

[103] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. Sanity checks for saliency metrics. *arXiv preprint arXiv:1912.01451*, 2019.

[104] Yaakov Tsaig and David L Donoho. Extensions of compressed sensing. *Signal processing*, 86(3):549–571, 2006.

[105] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 10–19, 2019.

[106] Angelina Wang, Arvind Narayanan, and Olga Russakovsky. Revise: A tool for measuring and mitigating bias in visual datasets. In *European Conference on Computer Vision*, pages 733–751. Springer, 2020.

[107] Fulton Wang and Cynthia Rudin. Falling rule lists. In *Artificial Intelligence and Statistics*, pages 1013–1022, 2015.

[108] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5310–5319, 2019.

[109] Ethan Weinberger, Joseph Janizek, and Su-In Lee. Learning deep attribution priors based on prior knowledge. *arXiv*, pages arXiv–1912, 2019.

[110] Jun Wen, Changjian Shui, Kun Kuang, Junsong Yuan, Zenan Huang, Zhefeng Gong, and Nenggan Zheng. Interventional domain adaptation. *arXiv preprint arXiv:2011.03737*, 2020.

[111] Julia K Winkler, Christine Fink, Ferdinand Toberer, Alexander Enk, Teresa Deinlein, Rainer Hofmann-Wellenhof, Luc Thomas, Aimilios Lallas, Andreas Blum, Wilhelm Stolz, et al. Association between surgical skin markings in dermoscopic images and diagnostic performance of a deep learning convolutional neural network for melanoma recognition. *JAMA dermatology*, 155(10):1135–1141, 2019.

[112] Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11205–11216. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/wong21b.html`.

[113] Mike Wu, Michael C Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[114] Mike Wu, Sonali Parbhoo, Michael Hughes, Ryan Kindle, Leo Celi, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Regional tree regularization for interpretability in black box models. *arXiv preprint arXiv:1908.04494*, 2019.

[115] Kai Yuanqing Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=gl3D-xY7wLq.

[116] Kaiyu Yang, Olga Russakovsky, and Jia Deng. Spatialsense: An adversarially crowdsourced benchmark for spatial relation recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2051–2060, 2019.

[117] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[118] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020.

[119] Jinsung Yoon, Sercan Arik, and Tomas Pfister. Limis: Locally interpretable modeling using instance-wise subsampling. 2022. URL https://arxiv.org/abs/1909.12367.

[120] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[121] Xin Zhang, Armando Solar-Lezama, and Rishabh Singh. Interpreting neural network judgments via minimal, stable, and symbolic corrections. In *Advances in Neural Information Processing Systems*, pages 4874–4885, 2018.

[122] Yi Zhang and Jitao Sang. Towards accuracy-fairness paradox: Adversarial example-based data augmentation for visual debiasing. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4346–4354, 2020.

[123] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.

[124] Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. Do feature attribution methods correctly attribute features? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9623–9633, 2022.

# Part II

# Appendices

# Appendix A

# User Interaction

## A.1 Comparison of EXPO to SENN

We compare against SENN [6] on the UCI ‘breast cancer’ dataset which is a binary classification problem. Because SENN’s implementation outputs class probabilities, we run the post-hoc explainers on the probability output from the EXPO-regularized model as well (this differs from the ‘support2’ binary classification problem where we explain each logit individually). The results are in Table A.1.

By comparing the first row, which shows the results for an EXPO-FIDELITY-regularized model whose regularization weight is tuned for accuracy, to the third row, which shows SENN’s results, we can see that SENN’s by-design approach to model interpretability seriously impaired its accuracy. However, SENN did produce a more interpretable model. From these results alone, there is no objective way to decide if the EXPO or SENN model is better. But, looking at the MAPLE-NF metric, we can see that its explanations have a standard error of around 4% relative to the model’s predicted probability. This is reasonably small and probably acceptable for a model that makes a fraction as many mistakes.

Looking at the second row, which shows the results for a EXPO-FIDELITY-regularized model whose regularization weight has been increased to produce a model that is approximately accurate as SENN, we can see that the EXPO-regularized model is more interpretable than SENN.

Considering both of these results, we conclude that EXPO is more effective than SENN at improving the quality of post-hoc explanations.

However, this is a slightly unusual comparison because SENN is designed to produce its own explanations but we are using LIME/MAPLE to explain it. When we let SENN explain itself, it has a NF of 3.1e-5 and a Stability of 2.1e-3. These numbers are generally comparable to those of LIME explaining EXPO-Over Regularized. This further demonstrates EXPO’s flexibility.

**Table A.1:** A comparison of EXPO-FIDELITY, with the regularization weight tuned for accuracy and with it set too high to intentionally reduce accuracy, to SENN. Results are shown across 10 trials (with the standard error in parenthesis). EXPO can produce either a more accurate model or an equally accurate but more interpretable model.

| Method | Accuracy | MAPLE-PF | MAPLE-NF | MAPLE-S | LIME-PF | LIME-NF | LIME-S |
|---|---|---|---|---|---|---|---|
| EXPO | 0.99 (0.0034) | 0.013 (0.0065) | 0.039 (0.026) | 0.38 (0.27) | 0.1 (0.039) | 0.1 (0.039) | 0.0024 (0.0012) |
| EXPO- Over Regularized | 0.92 (0.013) | 0.00061 (0.00031) | 0.0014 (0.00079) | 0.0085 (0.01) | 0.0035 (0.0037) | 0.0035 (0.0037) | 2.4e-05 (1.2e-05) |
| SENN | 0.92 (0.033) | 0.0054 (0.0024) | 0.014 (0.0048) | 0.097 (0.044) | 0.012 (0.0043) | 0.013 (0.0045) | 0.00075 (0.00012) |

## A.2 Expanded Version of Section 2.2

This section provides the details for the results outlined in Section 2.2.

**Local explanations vs. Taylor approximations.** A natural question to ask is, *Why should we sample from $N_x$ in order to locally approximate $f$ when we could use the Taylor approximation as done in [6, 82]?* The downside of a Taylor approximation-based approach is that such an approximation cannot readily be adjusted to different neighborhood scales and its fidelity and stability strictly depend on the learned function. Figure A.1 shows that the Taylor approximations for two close points can be radically different from each other and are not necessarily faithful to the model outside of a small neighborhood.

**Fidelity regularization and the model's LC or TV.** From a theoretical perspective, EXPO-FIDELITY is similar to controlling the Lipschitz Constant or Total Variation of $f$ across $N_x$ after removing the part of $f$ explained by $e(x, f)$. From an interpretability perspective, having a large LC or TV does not necessarily lead to poor explanation quality, which is demonstrated in Figure A.2.

**Standard Regularization.** We also consider two standard regularization techniques: $l_1$ and $l_2$ regularization. These regularizers may make the network simpler (due to sparser weights) or smoother, which may make it more amenable to local explanation. The results of this experiment are in Table A.2; notice that neither of these regularizers had a significant effect on the interpretability metrics.

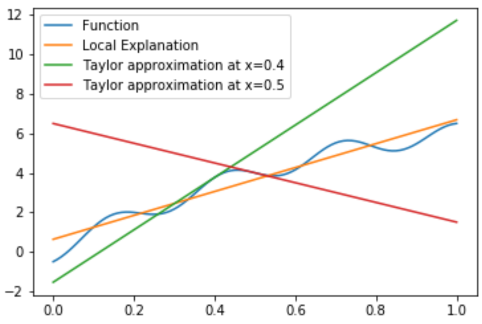


**Figure A.1:** A function (blue), its first order Taylor approximations at $x = 0.4$ (green) and $x = 0.5$ (red), and a local explanation of the function (orange) computed with $x = 0.5$ and $N_x = [0, 1]$. Notice that the Taylor approximation-based explanations are more strongly influenced by local variations in the function.

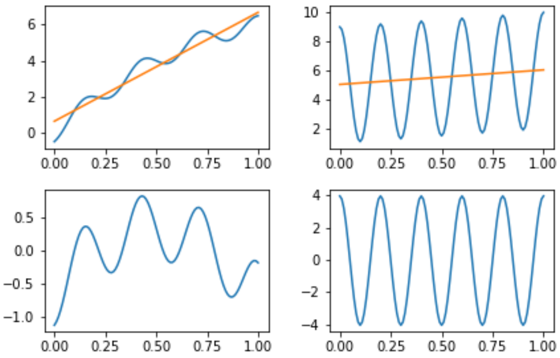


**Figure A.2:** **Top:** Two functions (blue) and their local linear explanations (orange). The local explanations were computed with $x = 0.5$ and $N_x = [0, 1]$. **Bottom:** The unexplained portion of the function (residuals). **Comment:** Although both functions have a relatively large LC or TV, the one on the left is much better explained and this is reflected by its residuals.

**Table A.2:** Using $l_1$ or $l_2$ regularization has very little impact impact on the interpretability of the learned model.

| Metric | Regularizer | autompgs | communities | day | housing | music | winequality.red |
|---|---|---|---|---|---|---|---|
| MSE | None | 0.14 | 0.49 | 0.001 | 0.14 | 0.72 | 0.65 |
| | L1 | 0.12 | 0.46 | 1.7e-05 | 0.15 | 0.68 | 0.67 |
| | L2 | 0.13 | 0.47 | 0.00012 | 0.15 | 0.68 | 0.67 |
| MAPLE-PF | None | 0.016 | 0.16 | 0.001 | 0.057 | 0.17 | 0.013 |
| | L1 | 0.014 | 0.17 | 1.6e-05 | 0.054 | 0.17 | 0.015 |
| | L2 | 0.015 | 0.17 | 3.2e-05 | 0.05 | 0.17 | 0.02 |
| MAPLE-NF | None | 0.018 | 0.31 | 0.0012 | 0.066 | 0.18 | 0.013 |
| | L1 | 0.016 | 0.32 | 2.6e-05 | 0.065 | 0.18 | 0.016 |
| | L2 | 0.016 | 0.32 | 4.3e-05 | 0.058 | 0.17 | 0.021 |
| MAPLE-Stability | None | 0.015 | 1.2 | 2.6e-07 | 0.18 | 0.081 | 0.0043 |
| | L1 | 0.013 | 1.2 | 3e-07 | 0.21 | 0.072 | 0.004 |
| | L2 | 0.011 | 1.3 | 3.2e-06 | 0.17 | 0.065 | 0.0058 |
| LIME-PF | None | 0.04 | 0.1 | 0.0012 | 0.14 | 0.11 | 0.033 |
| | L1 | 0.035 | 0.12 | 0.00017 | 0.13 | 0.1 | 0.034 |
| | L2 | 0.037 | 0.12 | 0.00014 | 0.12 | 0.099 | 0.047 |
| LIME-NF | None | 0.041 | 0.11 | 0.0012 | 0.14 | 0.11 | 0.033 |
| | L1 | 0.036 | 0.12 | 0.00018 | 0.13 | 0.1 | 0.034 |
| | L2 | 0.037 | 0.12 | 0.00015 | 0.12 | 0.099 | 0.047 |
| LIME-Stability | None | 0.0011 | 0.022 | 0.00015 | 0.0047 | 0.011 | 0.0013 |
| | L1 | 0.0012 | 0.03 | 3e-05 | 0.0048 | 0.011 | 0.0016 |
| | L2 | 0.00097 | 0.032 | 1.7e-05 | 0.004 | 0.011 | 0.0021 |

### A.3 Model Details and Selection

The models we train are Multi-Layer Perceptrons with leaky-ReLU activations. The model architectures are chosen by a grid search over the possible widths (100, 200, 300, 400, or 500 units) and depths (1, 2, 3, 4, or 5 layers). The weights are initialized with the Xavier initialization and the biases are initialized to zero. The models are trained with SGD with the Adam optimizer and a learning rate of 0.001. For each dataset, the architecture with the best validation loss is chosen for final evaluation as the "None" model. Then, we use that same architecture and add the EXPO regularizer with weights chosen from 0.1, 0.05, 0.025, 0.01, 0.005, or 0.001. Note that these are not absolute weights and are instead relative weights: so picking $0.1$ means that the absolute regularization weight is set such that the regularizer has $1/10^{th}$ the weight of the main loss function (estimated using a single mini-batch at the start of training and then never changed). This makes this hyper-parameter less sensitive to the model architecture, initialization, and dataset. We then pick the best regularization weight for each dataset using the validation loss and use that for the final evaluation as the EXPO model. Final evaluation is done by retraining the models using their chosen configurations and evaluating them on the test data.

## A.4 Defining the Local Neighborhood

**Choosing the neighborhood shape.** Defining a good regularization neighborhood, requires considering the following. On the one hand, we would like $N_x^{\text{reg}}$ to be similar to $N_x$, as used in Eq. 2.1 or Eq. 2.2, so that the neighborhoods used for regularization and for evaluation match. On the other hand, we would also like $N_x^{\text{reg}}$ to be consistent with the local neighborhood defined internally by $e$, which may differ from $N_x$. LIME can avoid this problem since the internal definition of the local neighborhood is a hyperparameter that we can set. However, for our experiments, we do not do this and, instead, use the default implementation. MAPLE cannot easily avoid this problem because the local neighborhood is learned from the data, and hence the regularization and explanation neighborhoods probably differ.

**Choosing $\sigma$ for $N_x$ and $N_x^{reg}$.** In Figure A.3, we see that the choice of $\sigma$ for $N_x$ was not critical (the value of LIME-NF only increased slightly with $\sigma$) and that this choice of $\sigma$ for $N_x^{reg}$ produced slightly more interpretable models.

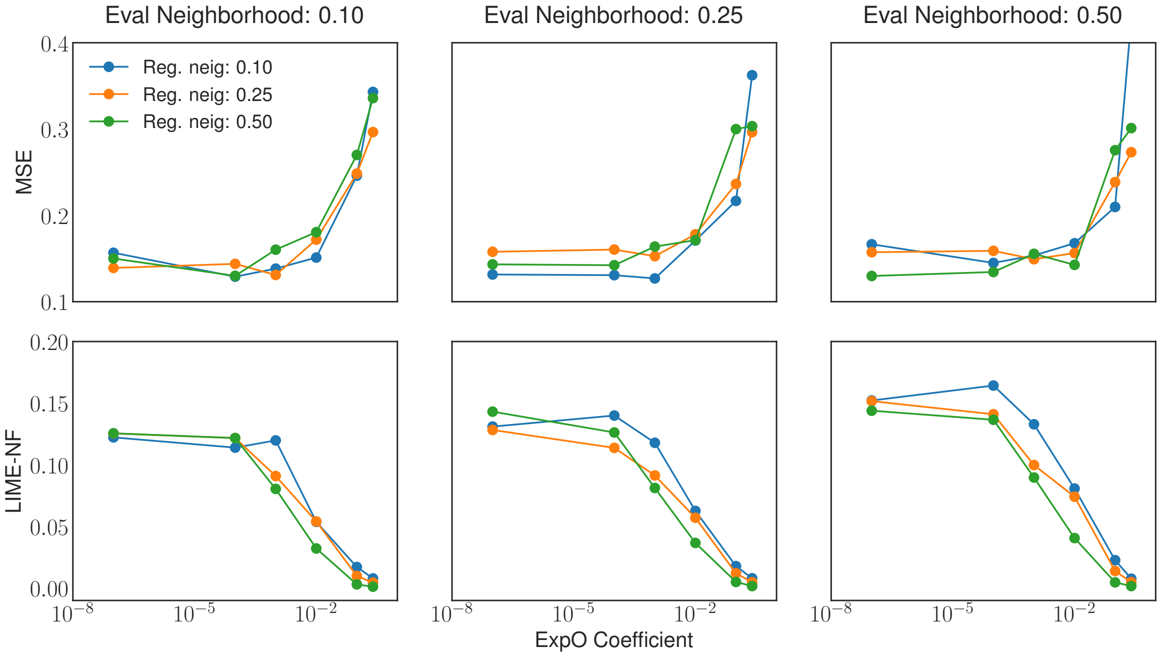


**Figure A.3:** A comparison showing the effects of the $\sigma$ parameter of $N_x$ and $N_x^{reg}$ on the UCI Housing dataset. The LIME-NF metric grows slowly with $\sigma$ for $N_x$ as expected. Despite being very large, using $\sigma = 0.5$ for $N_x^{reg}$ is generally best for the LIME-NF metric.

## A.5 More Examples of ExpO's Effects

Here, we demonstrate the effects of ExpO-Fidelity on more examples from the UCI 'housing' dataset (Table A.3). Observe that the same general trends hold true:

- The explanation for the ExpO-regularized model more accurately reflects the model (LIME-NF metric)
- The explanation for the ExpO-regularized model generally considers fewer features to be relevant. We consider a feature to be 'significant' if its absolute value is 0.1 or greater.
- Neither model appears to be heavily influenced by CRIM or INDUS. The ExpO-regularized model generally relies more on LSTAT and less on DIS, RAD, and TAX to make its predictions.

The same comparison for examples from the UCI 'winequality-red' are in Table A.4. We can see that the ExpO-regularized model depends more on 'volatile acidity' and less on 'sulphates' while usually agreeing about the effect of 'alcohol'. Further, it is better explained by those explanations than the normally trained model.

**Table A.3:** More examples comparing the explanations of a normally trained model ("None") to those of a ExpO-Fidelity-regularized model. For each example we show: the feature values of the point being explained, the coefficients of the normal model's explanation, and the coefficients of the ExpO-regularized model's explanation. Note that the bias terms have been excluded from the explanations. We also report the LIME-NF metric of each explanation.

| Example Number | Value Shown | CRIM | INDUS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | LIME-NF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $x$ | -0.36 | -0.57 | -0.86 | -1.11 | -0.14 | 0.95 | -0.74 | -1.02 | -0.22 | 0.46 | 0.53 | |
| | None | 0.01 | 0.03 | -0.14 | 0.31 | -0.1 | -0.29 | 0.27 | -0.26 | -0.07 | 0.13 | -0.24 | 0.0033 |
| | ExpO | 0.0 | 0.01 | -0.14 | 0.25 | 0.03 | -0.16 | 0.15 | -0.1 | -0.12 | -0.01 | -0.47 | 0.0033 |
| 2 | $x$ | -0.37 | -0.82 | -0.82 | 0.66 | -0.77 | 1.79 | -0.17 | -0.72 | 0.6 | 0.45 | -0.42 | |
| | None | 0.01 | 0.06 | -0.15 | 0.32 | -0.1 | -0.29 | 0.24 | -0.27 | -0.12 | 0.11 | -0.24 | 0.057 |
| | ExpO | 0.0 | 0.0 | -0.15 | 0.25 | 0.01 | -0.15 | 0.15 | -0.12 | -0.13 | 0.01 | -0.47 | 0.00076 |
| 3 | $x$ | -0.35 | -0.05 | -0.52 | -1.41 | 0.77 | -0.13 | -0.63 | -0.76 | 0.1 | 0.45 | 1.64 | |
| | None | -0.01 | 0.06 | -0.16 | 0.29 | -0.08 | -0.31 | 0.27 | -0.27 | -0.11 | 0.1 | -0.18 | 0.076 |
| | ExpO | -0.03 | -0.01 | -0.13 | 0.19 | -0.0 | -0.15 | 0.14 | -0.11 | -0.12 | 0.0 | -0.43 | 0.058 |
| 4 | $x$ | -0.36 | -0.34 | -0.26 | -0.29 | 0.73 | -0.56 | -0.51 | -0.12 | 1.14 | 0.44 | 0.14 | |
| | None | 0.02 | 0.06 | -0.18 | 0.29 | -0.1 | -0.34 | 0.31 | -0.21 | -0.09 | 0.12 | -0.27 | 0.10 |
| | ExpO | -0.02 | 0.01 | -0.13 | 0.21 | 0.02 | -0.16 | 0.17 | -0.11 | -0.12 | -0.0 | -0.47 | 0.013 |
| 5 | $x$ | -0.37 | -1.14 | -0.88 | 0.45 | -0.28 | -0.21 | -0.86 | -0.76 | -0.18 | 0.03 | -0.82 | |
| | None | 0.02 | 0.08 | -0.17 | 0.33 | -0.11 | -0.36 | 0.29 | -0.27 | -0.08 | 0.1 | -0.28 | 0.099 |
| | ExpO | -0.0 | -0.0 | -0.14 | 0.26 | 0.0 | -0.16 | 0.15 | -0.11 | -0.15 | 0.01 | -0.47 | 0.0021 |

**Table A.4:** The same setup as Table A.3, but showing examples for the UCI 'winequality-red' dataset

| Example Number | Value Shown | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | LIME-NF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $x$ | -0.28 | 1.55 | -1.31 | -0.02 | -0.26 | 3.12 | 1.35 | -0.25 | 0.41 | -0.2 | 0.29 | |
| | None | 0.02 | -0.11 | 0.14 | 0.08 | -0.1 | 0.05 | -0.15 | -0.13 | -0.01 | 0.31 | 0.29 | 0.021 |
| | ExpO | 0.08 | -0.22 | 0.01 | 0.04 | -0.04 | 0.06 | -0.12 | -0.09 | -0.01 | 0.17 | 0.3 | 6.6e-05 |
| 2 | $x$ | 1.86 | -1.91 | 1.22 | 0.87 | 0.39 | -1.1 | -0.69 | 1.48 | -0.22 | 1.96 | -0.35 | |
| | None | 0.02 | -0.15 | 0.11 | 0.07 | -0.08 | 0.07 | -0.23 | -0.09 | -0.06 | 0.3 | 0.27 | 0.033 |
| | ExpO | 0.09 | -0.23 | 0.02 | 0.04 | -0.05 | 0.06 | -0.13 | -0.09 | -0.0 | 0.18 | 0.3 | 0.0026 |
| 3 | $x$ | -0.63 | -0.82 | 0.56 | 0.11 | -0.39 | 0.72 | -0.11 | -1.59 | 0.16 | 0.42 | 2.21 | |
| | None | 0.03 | -0.1 | 0.13 | 0.05 | -0.06 | 0.12 | -0.21 | -0.19 | -0.08 | 0.38 | 0.29 | 0.11 |
| | ExpO | 0.09 | -0.22 | 0.02 | 0.04 | -0.04 | 0.06 | -0.12 | -0.09 | -0.0 | 0.18 | 0.3 | 8.2e-05 |
| 4 | $x$ | -0.51 | -0.66 | -0.15 | -0.53 | -0.43 | 0.24 | 0.04 | -0.56 | 0.35 | -0.2 | -0.07 | |
| | None | 0.03 | -0.16 | 0.12 | 0.05 | -0.13 | 0.09 | -0.21 | -0.13 | -0.05 | 0.35 | 0.24 | 0.61 |
| | ExpO | 0.09 | -0.22 | 0.01 | 0.04 | -0.04 | 0.06 | -0.12 | -0.09 | -0.01 | 0.18 | 0.3 | 6.8e-05 |
| 5 | $x$ | -0.28 | 0.43 | 0.1 | -0.65 | 0.61 | -0.62 | -0.51 | 0.36 | -0.35 | 5.6 | -1.26 | |
| | None | 0.03 | -0.12 | 0.09 | 0.12 | -0.11 | 0.03 | -0.19 | -0.13 | -0.03 | 0.13 | 0.24 | 0.19 |
| | ExpO | 0.08 | -0.22 | 0.02 | 0.04 | -0.05 | 0.05 | -0.13 | -0.09 | -0.0 | 0.16 | 0.3 | 0.0082 |

## A.6 Quantitative Results on the ‘support2’ Dataset

In Table A.5, we compare ExpO-regularized models to normally trained models on the ‘support2’ dataset.

**Table A.5:** A normally trained model (“None”) vs. the same model trained with ExpO-Fidelity or ExpO-1D-Fidelity on the ‘support2’ binary classification dataset. Each explanation metric was computed for both the positive and the negative class logits. Results are shown across 10 trials (with the standard error in parenthesis). Improvement due to Fidelity and 1D-Fidelity over the normally trained model is statistically significant ($p = 0.05$, t-test) for all of the metrics.

| Output | Regularizer | LIME-PF | LIME-NF | LIME-S | MAPLE-PF | MAPLE-NF | MAPLE-S |
|---|---|---|---|---|---|---|---|
| Positive | None | 0.177 (0.063) | 0.182 (0.065) | 0.0255 (0.0084) | 0.024 (0.008) | 0.035 (0.010) | 0.34 (0.06) |
| | Fidelity | **0.050 (0.008)** | **0.051 (0.008)** | **0.0047 (0.0008)** | **0.013 (0.004)** | **0.018 (0.005)** | **0.13 (0.05)** |
| | 1D-Fidelity | 0.082 (0.025) | 0.085 (0.025) | 0.0076 (0.0022) | 0.019 (0.005) | 0.025 (0.005) | 0.16 (0.03) |
| Negative | None | 0.198 (0.078) | 0.205 (0.080) | 0.0289 (0.0121) | 0.028 (0.010) | 0.040 (0.014) | 0.37 (0.18) |
| | Fidelity | **0.050 (0.008)** | **0.051 (0.008)** | **0.0047 (0.0008)** | **0.013 (0.004)** | **0.018 (0.005)** | **0.13 (0.03)** |
| | 1D-Fidelity | 0.081 (0.026) | 0.082 (0.027) | 0.0073 (0.0021) | 0.019 (0.006) | 0.024 (0.007) | 0.16 (0.06) |

**Accuracy (%):** None: $83.0 \pm 0.3$, Fidelity: $83.4 \pm 0.4$, 1D-Fidelity: $82.0 \pm 0.3$.

## A.7 User Study: Additional Details

**Data Details.** Figure A.4 shows the histogram of the number of steps participants take to complete each round. We use 150 as our cut-off value for removing participant's data from the final evaluation. Figure A.5 shows a histogram of the number of steps participants take to complete each round. There is no evidence to suggest that the earlier rounds took a different amount of time than the later rounds. So learning effects were not significant in this data.

**Algorithmic Agent.** In addition to measuring humans' performance on this task (see Section 2.5 for details), we are also interested in measuring a simple algorithmic agent's performance on it. The benefit of this evaluation is that the agent relies solely on the information given in the explanations and, as a result, does not experience any learning affects that could confound our results.

Intuitively, we could define a simple greedy agent by having it change the feature whose estimated effect is closest to the target change. However, this heuristic can lead to loops that the agent will never escape. As a result, we consider a randomized version of this greedy agent.

Let $\lambda$ the degree of randomization for the agent, $y$ denote the model's current prediction, $t$ denote the target value, and $c_i$ denote the explanation coefficient of feature $i$. Then the score of feature $i$, which measures how close this features estimated effect is to the target change, is: $s_i = -\lambda * ||c_i| - |y - t||$. The agent then chooses to use feature $i$ with probability $\frac{e^{s_i}}{\sum\limits_j e^{s_j}}$.

Looking at this distribution, we see that it is uniform (*i.e.*,, does not use the explanation at all) when $\lambda = 0$ and that it approaches the greedy agent as $\lambda$ approaches infinity.

In Figure A.6, we run a search across the value of $\lambda$ to find a rough trade-off between more frequently using the information in the explanation and avoiding loops. Note that the agent performs better for the EXPO-regularized model.

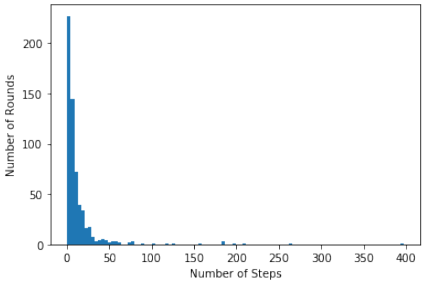


**Figure A.4:** A histogram showing the number of steps participants take to complete each round.

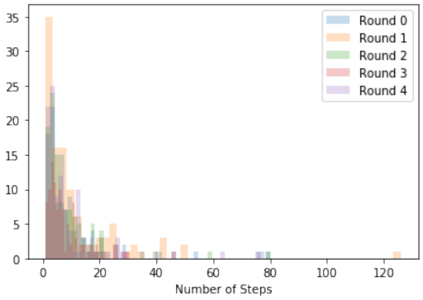


**Figure A.5:** A series of histograms showing how many steps participants take to complete each round for each condition. Generally, there is no evidence of learning effects over the course of the five rounds.

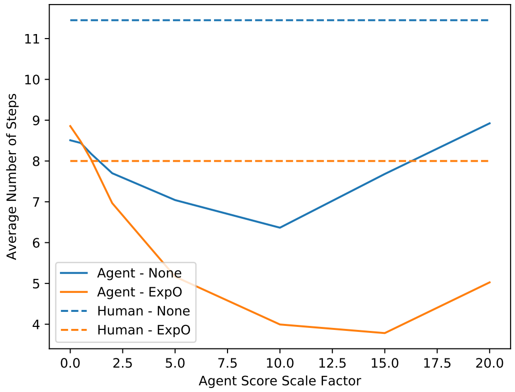


**Figure A.6:** A comparison of the average number of steps it takes for either a human or an algorithmic agent to complete our task. The x-axis is a measure of the agent's randomness: 0 corresponds to a totally random agent with increasing values indicating a greedier agent. Both humans and the agent find it easier to complete the task for the EXPO-regularized model.

## A.8 Non-Semantic Features

When $\mathcal{X}$ consists of non-semantic features, we cannot assign meaning to the difference between $x$ and $x'$ as we could with semantic features. Hence it does not make sense to explain the difference between the predictions $f(x)$ and $f(x')$ and fidelity is not an appropriate metric.

Instead, for non-semantic features, local explanations try to identify which parts of the input are particularly influential on a prediction [62, 99]. Consequently, we consider explanations of the form $\mathcal{E}_{ns} := \mathbb{R}^d$, where $d$ is the number of features in $\mathcal{X}$, and our primary explanation metric is stability. Note that we could consider these types of local explanations for semantic features as well, but that they answer a different question than the approximation-based local explanations we consider.

**Post-hoc explainers.** Various explainers [90, 96, 99, 120] have been proposed to generate local explanations in $\mathcal{E}_{ns}$ for images. However, it should be noted that the usefulness and evaluation of these methods is uncertain [1, 103]. For our experiment, we will consider *saliency maps* [91] which assign importance weights to image pixels based on the magnitude of the gradient of the predicted class with respect to the corresponding pixels.

**Stability Regularizer.** For EXPO-STABILITY, we simply require that the model's output not change too much across $N_x^{reg}$ (Algorithm 5). A similar procedure was explored previously in [123] for adversarial robustness.

**Experimental setup.** For this experiment, we compared a normally trained convolutional neural network on MNIST to one trained using EXPO-STABILITY. Then, we evaluated the quality of saliency map explanations for these models. Both $N_x$ and $N_x^{reg}$ where defined as $\mathrm{Unif}(x - 0.05, x + 0.05)$. Both the normally trained model and model trained with EXPO-STABILITY achieved the same accuracy of 99%. Quantitatively, training the model with EXPO-STABILITY decreased the stability metric from 6.94 to 0.0008. Qualitatively, training the model with EXPO-STABILITY made the resulting saliency maps look much better by focusing them on the presence or absence of certain pen strokes (Figure A.7).

**Algorithm 5** Neighborhood-stability regularizer

```
input $f_\theta$, $x$, $N_x^{\mathrm{reg}}$, $m$
  Sample points: $x'_1, \ldots, x'_m \sim N_x^{\mathrm{reg}}$
  Compute predictions:
      $\hat{y}_j(\theta) = f_\theta(x'_j)$, for $j = 1, \ldots, m$
output $\frac{1}{m}\sum_{j=1}^{m}(\hat{y}_j(\theta) - f(x))^2$
```

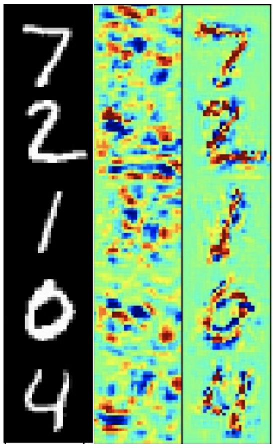


**Figure A.7:** Original images (left) and saliency maps of an normally trained model (middle) and a EXPO-STABILITY-regularized model (right).

# Appendix B

# Knowledge Discovery

## B.1 An Example of the Difficulty of Using Existing Methods

For this example, we are going to consider Integrated Gradients (IG) [99] which produces local feature attribution explanations. Because IG is a supervised method, we start by training a classifier on top of the learned representation to get a multi-class classification model $f$ that predicts which group a point belongs to. Because our goal is to explain the difference between Group A and Group B with IG, we average IG's explanation for each point in Group B relative to each possible baseline value of a point in Group A for $f$'s Class B label. To be more precise:

$$\delta_{IG}(A \to B) = \frac{1}{|X_A|\,|X_B|} \sum_{x \in X_B} \sum_{a \in X_A} IG(x, \text{class} = B, \text{baseline} = a) \tag{B.1}$$

We will refer to this as 'group Integrated Gradients' or gIG.

**Challenge 1: Comparing Explanation Types.** Because IG produces feature attributions and TGT produces counterfactuals, there is no reliable metric in the literature to directly compare them. On the one hand, most feature attributions are the 'correct explanation' for their specific definitions for 'attribution' and the 'baseline' value; this has made measuring their quality challenging [103]. On the other hand, we cannot treat a feature attribution as a transformation function/translation, so our metrics and other metrics for counterfactual explanations cannot be applied.

As a result, we compare TGT to gIG on the same synthetic dataset we used earlier. We found that gIG identifies the causal variables as being significant and ignores the noise variable, but that it also identifies the correlated variable as being significant. This indicates that it is likely to be unable to find sparse explanations as well as TGT can.

**Challenge 2: Consistency of Aggregated Local Explanations.** One of the reasons we chose IG as a baseline method to aggregate is because its attributions are symmetrical and transitive *with respect to a fixed class.* In other words, if all we cared about was explaining the differences between all of the groups of points with respect to a single reference group, say Group C, then IG would produce consistent explanations. However, explaining the features that separate Group A from Group B relative to Group C is not the problem we are trying to solve.

When we use Equation B.1 to calculate $\delta_{IG}(i \to j)$ we found that the resulting explanations were not consistent. This does not violate the theory of IG because each $\delta_{IG}(i \to j)$ is calculated with reference Group $j$ and so the assumption that we have a single reference group is not satisfied.

When we considered modifying Equation B.1 to aggregate over the reference 'class/group' and potentially gain consistency that way, we either got uniform zero attributions (if we averaged over all reference groups) or inconsistent explanations (if we excluded any subset of $\{i, j\}$ from the averaging).

**Conclusion.** As suggested in Section 3.2, using existing explanation methods to find GCEs is going to be challenging because it is not what they are designed to do. We found that IG, a method that theoretically looked promising, was unable to be extended in a simple way to this setting.

### B.2 Representation Function

**Differentiability.** TGT assumes that $r$ is a differentiable function. Hidden in this assumption is the assumption that $r$ is a function that we can evaluate on an arbitrary point. Although most methods for learning a low-dimensional representation satisfy this assumption, t-SNE does not. Fortunately there are parametric variations of t-SNE such as the one we used in our experiments [30]. The assumption that $r$ is differentiable can be relaxed by using a finite-difference optimization method, such as SPSA [98], at the expense of computational cost.

**Learning Meaningful Structure.** One assumption that every analysis (whether that is manual inspection, statistical testing, or interpretable ML) of the representation learned by $r$ is that this function learned meaningful structure from the data. Because practitioners are already relying on these representations and, in some situations, have verified that they are meaningful, this concern is largely orthogonal to our work.

However, from an interpretable ML perspective, our goal is to explain $r$. So, if $r$ identifies different structure when it is retrained or when it is trained with a different algorithm or structure, we expect TGT to produce different explanations since the embedding itself has changed.

Our experimental results show that the representation learned by [30] is stable to being retrained and to modifications to the dataset and that TGT produces stable explanations for these representations.

**Identifying that Structure with Explanations.** It is possible to have a model that learned the true structure of the data and to have an explanation that is technically true (as measured by some proxy metric for interpretability) about the model but that also fails to capture meaningful patterns. For example, adversarial examples [100] are technically local counterfactual explanations but they usually look like random noise and, as a result, do not tell a person much about the patterns the model has learned. TGT's design, which calculates the explanation between each pair of groups as if it were a compressed sensing problem but constrains those solutions to be symmetrical and transitive among all groups, was chosen as a prior to prevent this type of behavior.

## B.3 Learned Representations

The learned representations and the corresponding groups of points for the datasets we studied are in Figures B.1, B.2, B.3, B.4, and B.5.

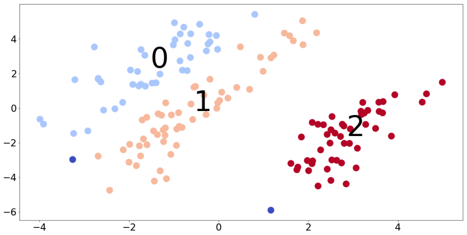


**Figure B.1:** The learned representation and grouping for the UCI Iris dataset

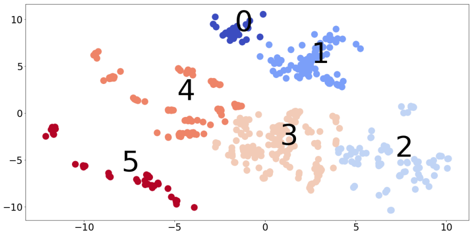


**Figure B.2:** The learned representation and grouping for the UCI Boston Housing dataset

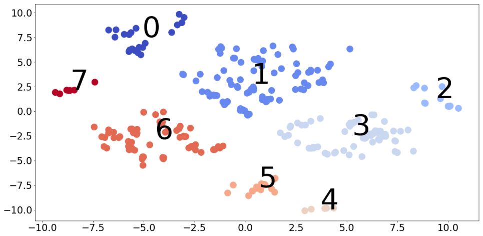


**Figure B.3:** The learned representation and grouping for the UCI Heart Disease dataset

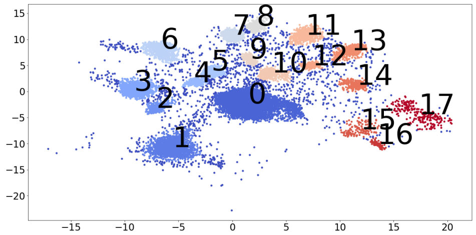


**Figure B.4:** The learned representation and grouping for the single-cell RNA dataset

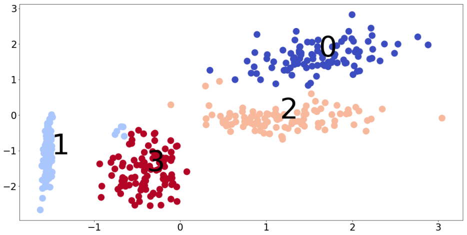


**Figure B.5:** The learned representation and grouping for the synthetic dataset.

## B.4 Pairwise Correctness and Coverage Plots

TGT is much better than DBM for finding 250-sparse explanations on the single-cell RNA dataset (Figures B.6 and B.7). On the synthetic dataset, TGT and DBM are equally effective explanations of the model (Figures B.8 and B.9). However, TGT only relied on the two causal variables while DBM included the correlated variable as well.

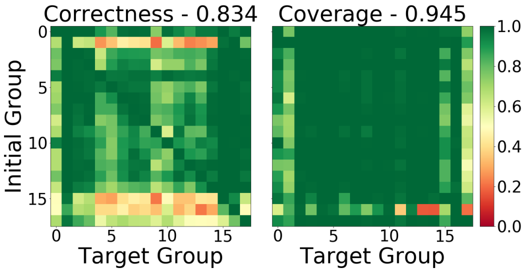


**Figure B.6:** The pairwise metrics for TGT on the single-cell RNA dataset for 250-sparse explanations.

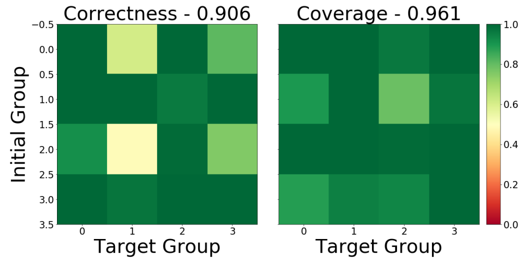


**Figure B.8:** The pairwise metrics for TGT on the synthetic dataset with no sparsity constraint.

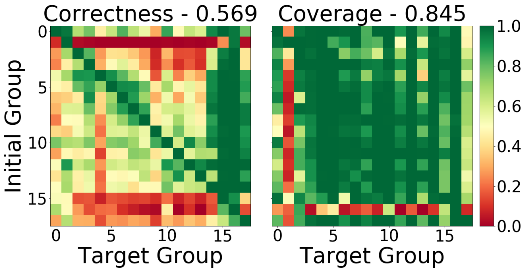


**Figure B.7:** The pairwise metrics for DBM on the single-cell RNA dataset for 250-sparse explanations.

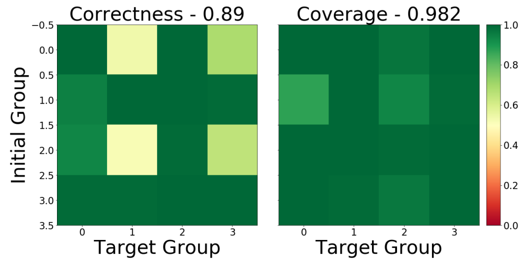


**Figure B.9:** The pairwise metrics for DBM on the synthetic dataset with no sparsity constraint.

### B.5 Qualitative Analysis of the UCI Datasets using the Labels

Although the representations we learned for these datasets were trained in an unsupervised manner, the groups that they find often have strong connections to the labels for the datasets; see Table B.1, Figure B.12, and Table B.2. By using the connection between the groups and labels, we will be able to qualitatively assess whether or not TGT is finding real patterns in the data.

**Iris Dataset.** Looking at Table B.1, we can see that the groups in this representation match very closely with the class labels. As a result, we would like to know whether or not the explanations TGT finds are consistent with a model trained directly to predict the labels. For this comparison, we used a simple decision tree, which is shown in Figure B.10. Looking at TGT's explanations (Figure B.11), we can see that they largely agree with the decision tree since both primarily use Petal Width to separate the classes/groups.

**Boston Housing Dataset.** Looking at Figure B.12, we can see that two comparisons between the groups stand out: Group 0 to Group 2, which shows a significant increase in the price, and Group 3 to Group 5, which has relatively little effect on the price. As a result, we would like to determine what the differences between these groups of houses are that influence their price.

Looking at Figure B.13, we see that TGT found that the key differences between Group 0 and Group 2 appears to be the difference between a house being in an urban area vs being in a suburb: the proportion of land zoned for large residential lots and B[1] both increase while access to radial highways and tax rates both decrease. It also found that the key differences between Group 3 and Group 5 are, first, moving the house onto the Charles river and, second, decreasing B.

**Heart Disease Dataset.** Looking at Figure B.2, we see that there are three large groups that stand out: Group 1, which has a balanced risk of heart disease, Group 3, which has a relatively low risk, and Group 6, which has a relatively high risk. As a result, we would like to determine what the differences between these groups of subjects are that influence their risk of heart disease.

Looking at Figure B.14, TGT found that the key differences between Group 1 and Group 3 are a moderate decrease in chest pain along with having exercised induced angina; these are subjects whose symptoms are explained by exercise induced angina rather than heart disease. It also found that the key difference between Group 1 and Group 6 is that Group 1 is made up of men while Group 6 is made up of women; this is consistent with the fact that heart disease is the leading cause of mortality in women [9].

[1] This is a unusually defined feature that is related to the racial demographics of a town. Determining what it means to change this feature depends on a measurement that is not in the dataset.

**Table B.1:** The distribution of the labels per group for the UCI Iris dataset (classification).

| Group \ Class | Iris Setosa | Iris Versicolour | Iris Virginica |
|---|---|---|---|
| 0 | 0 | 5 | 38 |
| 1 | 0 | 44 | 12 |
| 2 | 48 | 0 | 0 |

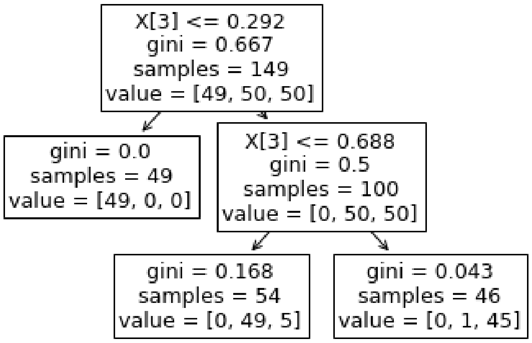


**Figure B.10:** A small decision tree trained on this dataset. Notice that it relies on the Petal Width feature.

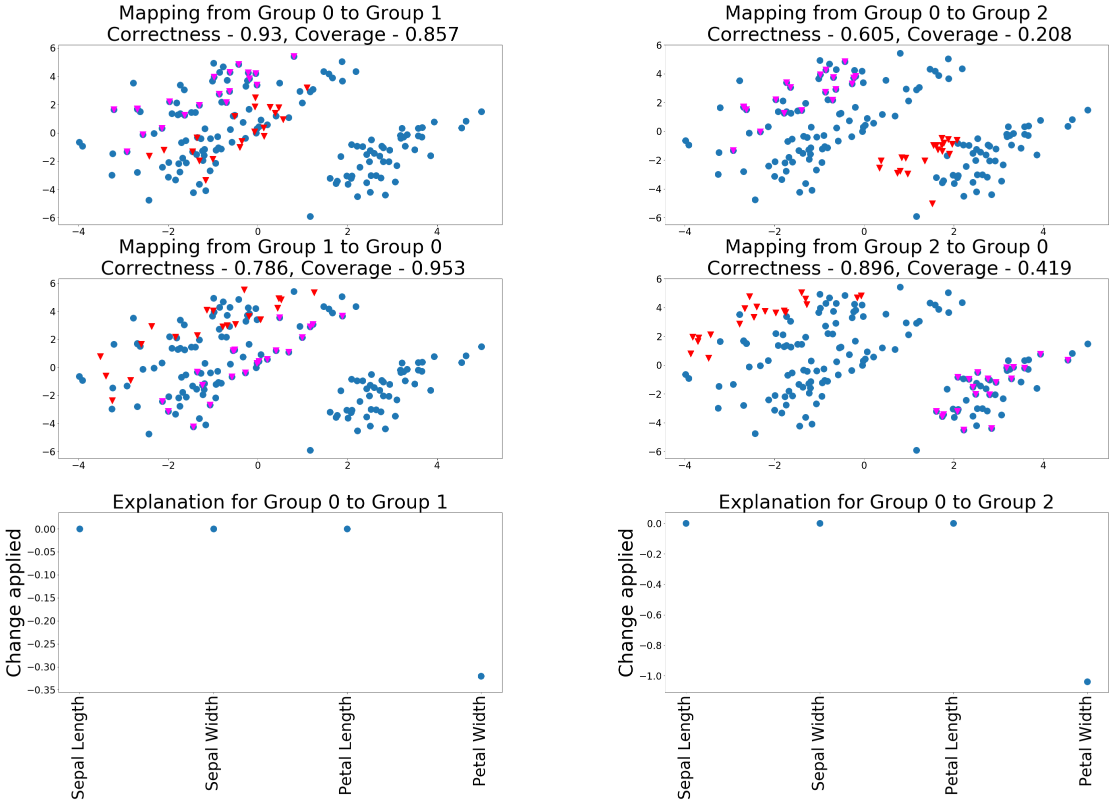


**Figure B.11:** TGT's 1-sparse explanation of the difference between Group 0 and Group 1 (left) and Group 0 and Group 2 (right). Similar to the decision tree, they rely on the Petal Width feature.

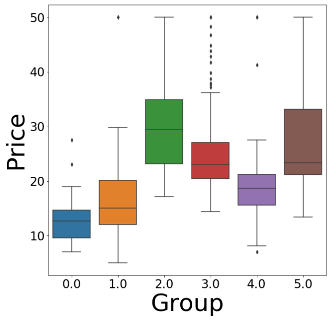


**Figure B.12:** The distribution of the labels per group for the UCI Boston Housing dataset (regression).

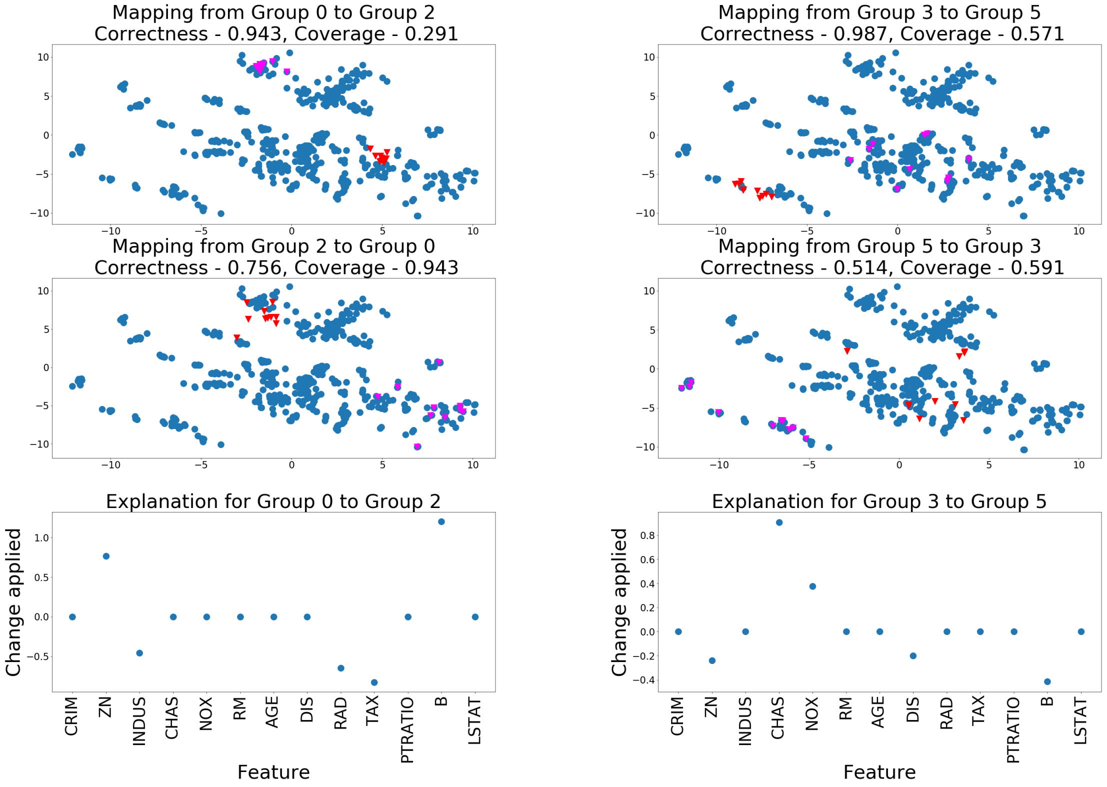


**Figure B.13:** TGT's 5-sparse explanation for Group 0 to Group 2 (Left) and Group 3 to Group 5 (right).

**Table B.2:** The distribution of the labels per group for the UCI Heart Disease dataset (classification).

| Class<br>Group | No Heart Disease | Heart Disease |
|---|---|---|
| 0 | 6 | 15 |
| 1 | 46 | 62 |
| 2 | 10 | 1 |
| 3 | 52 | 14 |
| 4 | 4 | 1 |
| 5 | 10 | 7 |
| 6 | 8 | 59 |
| 7 | 2 | 5 |

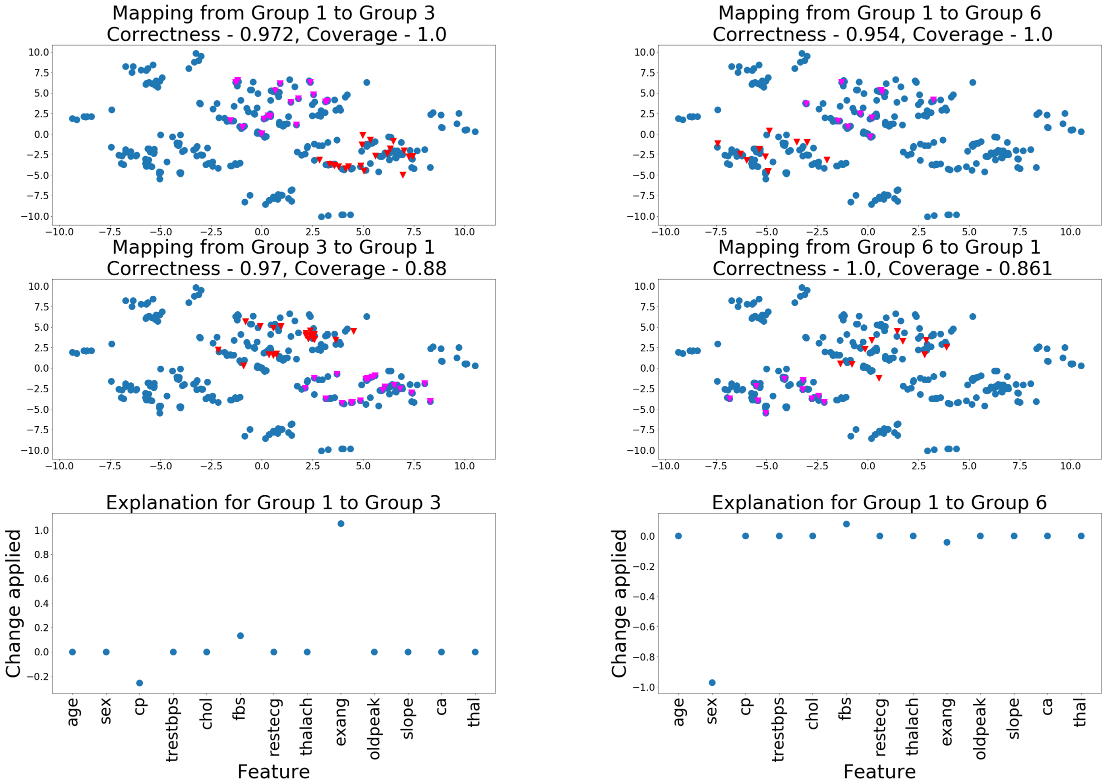


**Figure B.14:** TGT's 3-sparse explanations for the difference between Group 1 and Group 3 (Left) and Group 1 and Group 6 (right).

## B.6 Quantitative Analysis of Modified Versions of the UCI Datasets.

Because the UCI datasets are not synthetic datasets, we do not know the underlying process that generated the data and, as a result, it is difficult to quantitatively determine whether or not an explanation is "correct" in the way that we could with the synthetic dataset. Consequently, we performed a series of experiments on modified versions of the original datasets in order to answer two important questions:

- Does TGT correctly identify the modifications we made to the original dataset?
- Do TGT's explanations between the original groups change when the modified group is added to the dataset?

We found that TGT does identify the modifications we made and that, in doing so, it does not significantly change the explanations between the original groups. Importantly, this result remains true even if we retrain the learned representation on the modified dataset. These results are a strong indicator that TGT is finding real patterns in the data.

**How do we modify the datasets?** We create a modified version of the original dataset by: picking one of the groups of points in the original dataset, modifying that group of points in some way, and adding that new modified group of points to the original dataset. We will call the original dataset $D$ and the modified dataset $D'$, where $D' = D \cup G'$ and $G'$ is the modified version of some group of points $G$.

The critical choice to make during this process is to determine what modification to apply to $G$ to get $G'$. We chose to add random noise to some of the features of the points in $G$ and used the following two criteria when defining this modification for a particular dataset:

- $G'$ should be approximately within the range/distribution of $D$.
- $r(G')$ should form it's own (approximately) distinct group. Intuitively, if $r(G')$ does not form its own group, then $r$ thinks $G'$ is similar to some other group in the dataset and, as a result, we would not expect TGT to be able to explain the differences between $G'$ and that group of points.

The modifications we used are in Table B.3.

**Experimental Setup:** We now have two versions of each dataset: $D$ and $D'$. We also have the original learned representation $r$, which was trained on $D$, and a new learned representation $r'$, which is trained on $D'$. As a result, we have three sets of explanations:

- **Original:** These explain $r$ when applied to $D$
- **Modified:** These explain $r$ when applied to $D'$
- **Retrained:** These explain $r'$ when applied to $D'$

The visualization of the representation for the first setting is in the Appendix B.3 and the later two these settings is in Figures B.15, B.16, and B.17. Note that applying $r$ to $D'$ looks the same as applying $r$ to $D$ except for the fact that there is an additional group from adding $G'$ to $D$ and that applying $r'$ to $D'$ often shows that $r'$ has learned to separate $G'$ from the other groups better than $r$ did.

**Does TGT correctly identify the modifications we made to the original dataset?** In Figure Figures B.18, B.19, and B.20, we can see the explanations TGT found for the difference between $G$ and $G'$ for each of the datasets. If we compare the explanations to the modifications from Table B.3, we can see that they identified which features we changed and, approximately, by how much. The error in the estimation of "by how much" is due to the $l_1$ regularization used to find a simple explanation.

**Do TGT's explanations between the original groups change when the modified group is added to the dataset?** In Figures B.21, B.22, and B.23, we can see a comparison of the explanations for the differences between the original groups for the Original vs the Modified and the Original vs the Retrained explanations. Adding $G'$ to $D$ did not cause TGT to find significantly different explanations between the groups in $D$. Explaining $r'$ resulted in explanations that were generally similar, but adding another layer of variability (*i.e.,* training $r'$) did add some noise.

**Table B.3:** For each dataset, we chose a group of points to modify and modified it by applying these perturbations to the specified features.

| Dataset | Group Modified | Feature | Perturbation Applied |
|---|---|---|---|
| Iris | 0 | Sepal Width | -0.4 + Uniform(-0.1, 0.1) |
| Housing | 1 | ZN | 0.9 + Uniform(-0.1, 0.1) |
| | | TAX | -0.5 + Uniform(-0.1, 0.1) |
| Heart | 1 | restecg | -0.9 + Uniform(-0.1, 0.1) |
| | | exang | 0.6 + Uniform(-0.1, 0.1) |

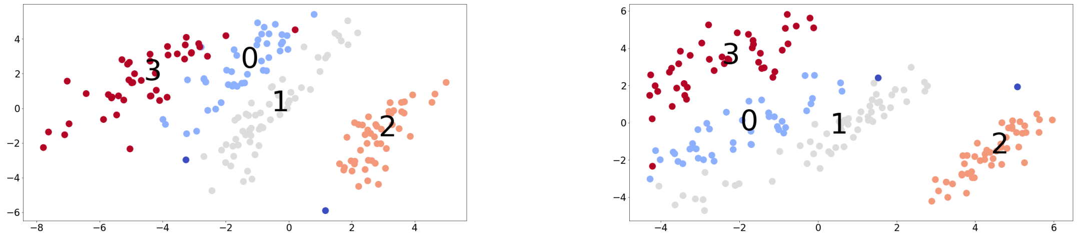


**Figure B.15:** The learned representation for: $r$ applied to $D'$ (Left) and $r'$ applied to $D'$ (Right) for the Iris dataset

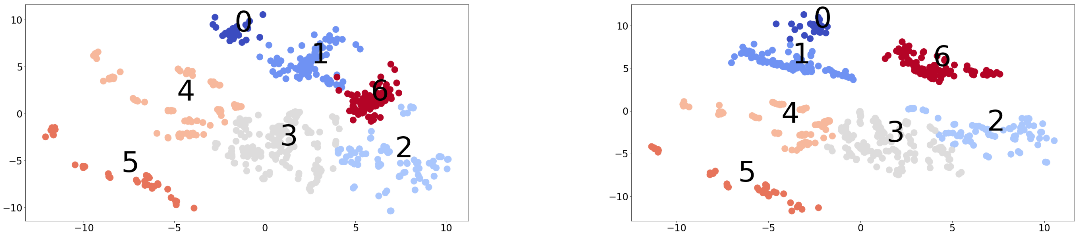


**Figure B.16:** The learned representation for: $r$ applied to $D'$ (Left) and $r'$ applied to $D'$ (Right) for the Boston Housing dataset

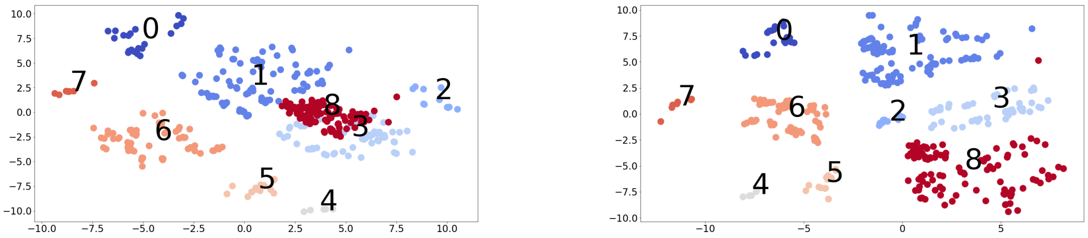


**Figure B.17:** The learned representation for: $r$ applied to $D'$ (Left) and $r'$ applied to $D'$ (Right) for the Heart Disease dataset

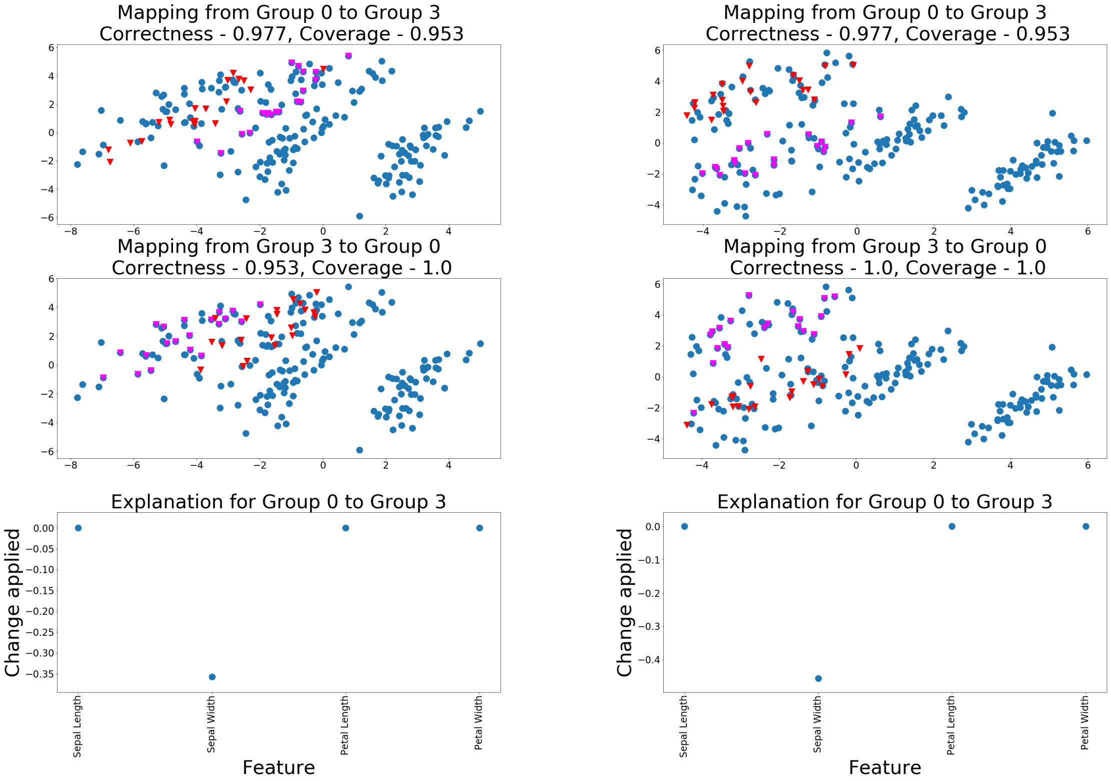


**Figure B.18:** The Modified (Left) and Retrained (Right) explanation's explanation for the difference between $G$ and $G'$ on the Iris dataset.

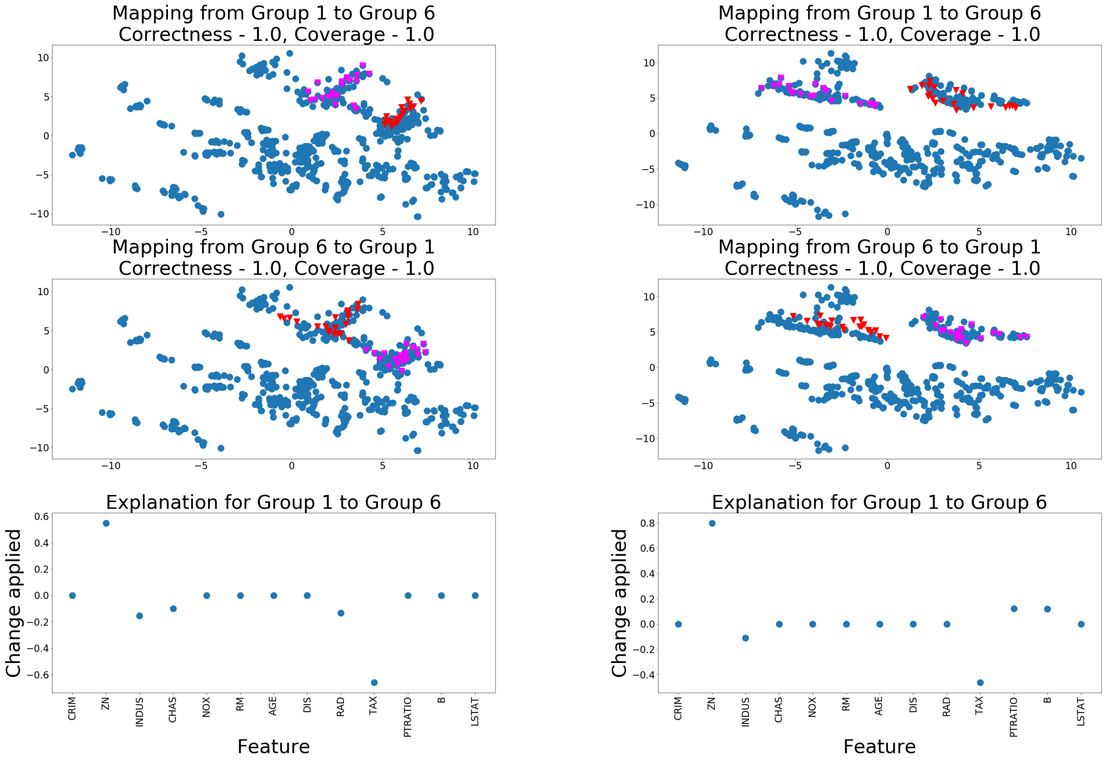


**Figure B.19:** The Modified (Left) and Retrained (Right) explanation's explanation for the difference between $G$ and $G'$ on the Boston Housing dataset.

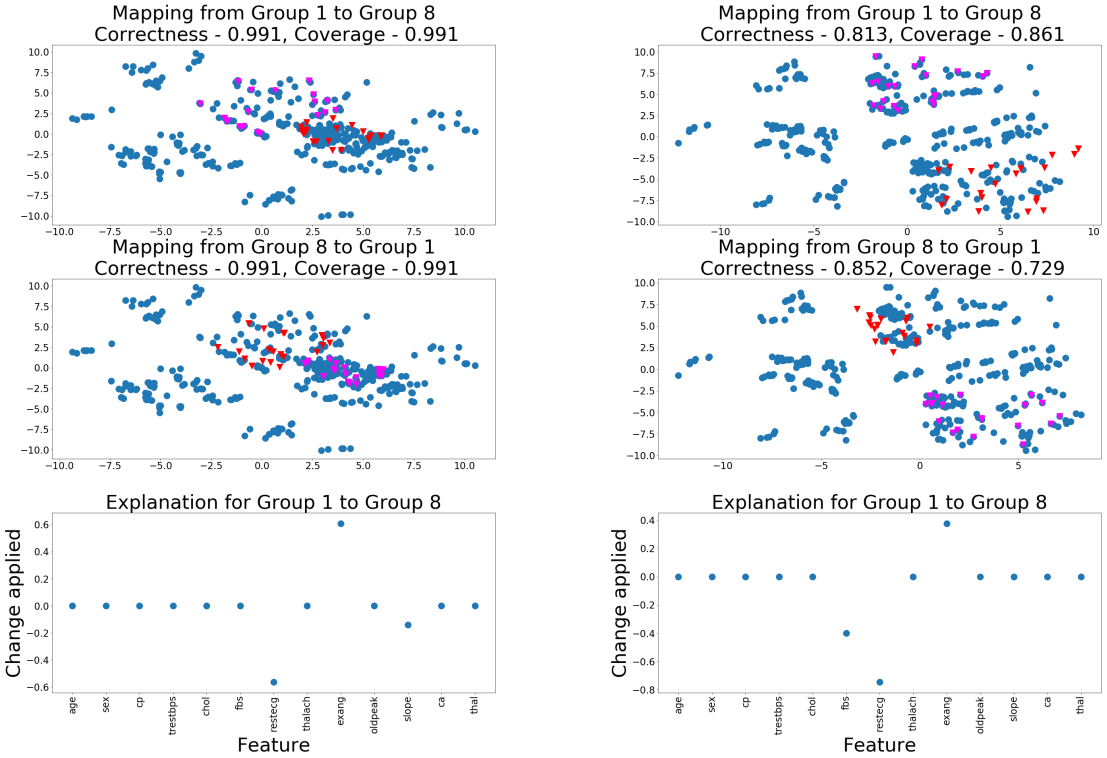


**Figure B.20:** The Modified (Left) and Retrained (Right) explanation's explanation for the difference between $G$ and $G'$ on the Heart Disease dataset.

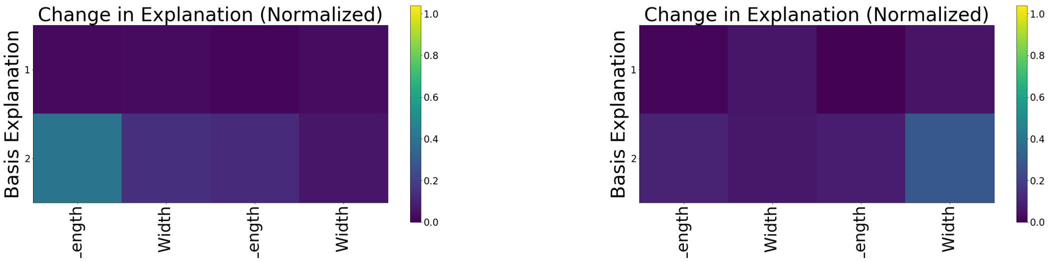


**Figure B.21:** The absolute difference between the Modified (Left)/Retrained (Right) explanations and the Original explanations scaled relative to the Original explanations on the Iris dataset.

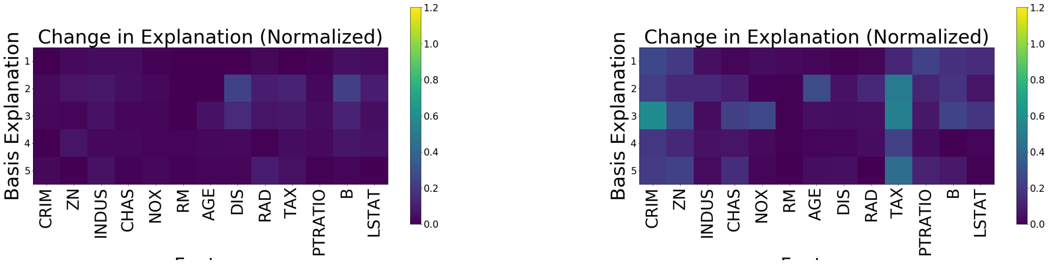


**Figure B.22:** The absolute difference between the Modified (Left)/Retrained (Right) explanations and the Original explanations scaled relative to the Original explanations on the Boston Housing dataset.

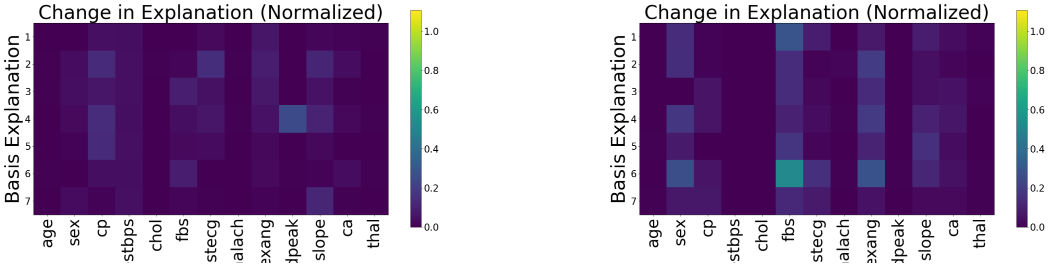


**Figure B.23:** The absolute difference between the Modified (Left)/Retrained (Right) explanations and the Original explanations scaled relative to the Original explanations on the Heart Disease dataset.

# Appendix C

# Spurious Patterns

## C.1 Discussion

In this section, we elaborate on SPIRE's strengths, its weaknesses, and suggested directions for future work. While there are many ways to improve SPIRE, we have demonstrated that it is a clear step forwards for the problem of addressing SPs.

**Generating Counterfactual Images.** SPIRE relies on the ability to produce counterfactual images. As a result, finding ways to produce similar counterfactuals with fewer assumptions (*e.g.,* being able to add/remove objects without relying on having an annotated dataset) or to produce different types of counterfactuals (*e.g.,* changing attributes such as "color") are both directions for future work. The former would improve the general applicability of SPIRE while the later would increase the scope of the types of SPs SPIRE could address.

**Identification.** SPIRE's strategy for identification can be summarized as "measure the probability that the model's prediction changes when we take an image from Group $X$ and apply Counterfactual Transformation $Y$." Intuitively, this strategy is effective because the original and counterfactual versions of an image differ only in terms of the effect of the counterfactual transformation while, if we were to compare natural images in one group to another group, there are probably going to be additional differences. Because SPIRE uses $X$ = Both, it may not be as effective as possible for identifying negative SPs because this split is likely to be very small for negatively correlated objects. As a result, future work could increase the scope of the types of SPs SPIRE could identify by considering different definitions of $X$ (*e.g.,* $X$ is the set of images that have objects $1, \ldots, m$ and do not have objects $m + 1, \ldots, n$; $X$ is the set of images where objects 1 and 2 appear near to/far from each other) or $Y$ (*e.g.,* $Y$ removes objects 1 and 2; $Y$ changes the location of object 1).

Interestingly, we find that a strong correlation is neither sufficient (Figure C.3 shows that the model can ignore a strong correlation) nor necessary (Table 4.2 shows that some SPs are between objects that are almost uncorrelated) for a model to learn to use a SP, which is consistent with prior findings [65, 87]. These result demonstrates SPIRE's advantage over identification methods that only consider the training distribution [*e.g.,* 106].

**Mitigation.** To begin with, it is worth noting that mitigating a SP may not always be worthwhile (*e.g.,* when one is certain that the distribution will not shift).

At a high level, SPIRE's strategy for mitigation works by removing the statistical incentive for the model to rely on the SP, while trying not to add new SPs; this strategy may be less effective for SPs that do not arise from correlations in the training distribution. Previous augmentation-based mitigation methods might be less effective because they are intuitive rather than statistical (*e.g.,* it makes intuitive sense that removing people should lessen the model's reliance on people to detect tennis rackets, but this intuition does not carry over to the dataset statistics). Previous regularization-based mitigation methods might be less effective because they may interfere with the learning process (*e.g.,* cause the model to become stuck in a local minimum) or they may have effects that are too local to matter (*e.g.,* changing the model's gradient at a point may not change its predictions very far away from that point). In particular, the Feature Splitting (FS) method from [93] assumes that one half of the features learned by the model are relevant for detecting objects "in context" and that the other half are relevant for objects "out of context;" while plausible for a single SP, this assumption becomes more tenuous as the number of SPs being mitigated increases.

While SPIRE's mitigation strategy is defined by two high-level goals, it is not always successful at realizing those goals (*e.g.,* for $p < 0.5$ in Section 4.5.1, SPIRE introduces the potential for new SPs) and the way those goals are realized depends on the problem setting. As a result, future work could improve the general applicability of SPIRE by finding a unified strategy that does not depend on the problem setting, generalizing that strategy to work for more general SPs, and extending that strategy to problems other than image classification. Additionally, future work could develop a theoretical framework to help understand the effects of augmentation-based mitigation strategies.

## C.2 Method Details

### C.2.1 When can SPIRE be applied to a problem?

In this section, we walk through the assumptions needed to apply SPIRE to a *model* that has been trained to perform some *task* using some *data*.

- SPIRE is model-agnostic and, as a result, can be applied to any type of model.
- SPIRE assumes that task is binary-classification and, consequently, that each image has a binary label.
- SPIRE assumes that each image has a different binary label for the "spurious feature" that we are interested in studying. For example, this could be variable indicating whether or not the image contains some high frequency signal.
- SPIRE assumes we can manipulate the images in such a way that we can create a counterfactual version of an image in one split that belongs to a different split. This entails being able to change an image's label and/or the value of the "spurious feature."

Collectively, these assumptions allow us to define the image splits (Table C.1) and produce the counterfactual images that SPIRE uses to identify and mitigate spurious patterns.

### C.2.2 Generating Counterfactual Images

Similar to prior work, SPIRE generates counterfactual images by adding objects to or removing objects from the original image [4, 18, 59, 89, 102, 115]. In this work, use the pixel-wise object-annotations that are part of various datasets such as COCO to generate the counterfactual images. Figure C.1 shows examples. Orthogonally, there is prior work that generates fundamentally different types of counterfactual images [36, 67, 84, 122].

**Removing an Object.** We consider two different ways to define which region of the image we are going to replace (pixel-wise or bounding-box) and two different ways to in-fill that region (using constant grey color or in-painting with the model from [66]). When we say that we "remove" an object, we mean that we found its bounding-box region and in-filled it with grey. When we say that we "in-paint" an object, we mean that we found its pixel-wise region and in-painted it. In order to minimize label noise, we make sure we do not include Main in the region that is going to be removed when we are removing Spurious.

**Table C.1:** A more general definition of the *image splits* defined in Section 4.3.

| Binary Image Label | Binary "Spurious Feature" | Image Split |
|---|---|---|
| 1 | 1 | Both |
| 1 | 0 | Just Main |
| 0 | 1 | Just Spurious |
| 0 | 0 | Neither |

**Adding an Object.** To add an object to an image, we find the pixel-wise region for that object in a different image and then replace that region's counterpart in the original image with it. In order to minimize label noise, we make sure that we do not cover Main when we add Spurious.

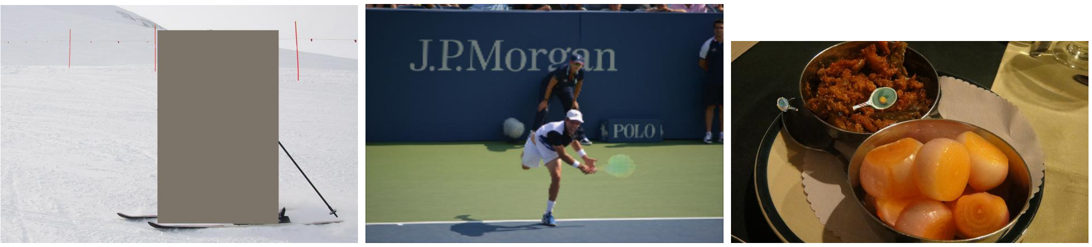


**Figure C.1:** Example counterfactual images for the tennis racket example. **(Left)** An example of moving an image from Just Spurious to Neither by Removing Spurious. **(Center)** An example of moving an image from Both to Just Spurious by In-Painting Main. **(Right)** An example of moving an image from Neither to Just Main by Adding Main.

### C.2.3 What does it mean to introduce new potential SPs?

We try to minimize the potential for new SPs by ensuring that P(Main | Artifact) = 0.5, where the Artifact could be "Grey Box" from removing objects from an image or objects with "Unusual Placement" from adding objects to an image. However, it is not clear whether 0.5 or P(Main) is the "correct" choice for this value. One one hand, using P(Main | Artifact) = 0.5 maximizes the loss that the model will receive if it relies on Artifact. On the other hand, setting P(Main | Artifact) = P(Main) means that Main is independent of Artifact and that there is no statistical incentive for the model to rely on Artifact. Because we will not be evaluating the model (in terms of accuracy) on images with Artifact, we chose 0.5 because it actively discourages using Artifact rather than simply not encouraging it.

### C.2.4 Setting 1: Working through SPIRE's augmentation strategy

In this setting, $\{$Both, Neither$\}$ each have size $0.5p$ while $\{$Just Main, Just Spurious$\}$ have size $0.5(1-p)$.

For $p > 0.5$, SPIRE removes $\{$Main, Spurious$\}$ from Both with probability $\frac{2p-1}{2p}$ and, as a result, P(Main | Grey Box) = 0.5. Similarly, SPIRE adds $\{$Main, Spurious$\}$ to Neither with the same probability and, as a result, P(Main | Unusual Placement) = 0.5. As a result, $\{$Just Main, Just Spurious$\}$ each receive $0.25(2p-1)$ images from each of $\{$Both, Neither$\}$ and have an augmented size of $0.5p$. So SPIRE produces the balanced distribution without creating the potential for new SPs.

For $p < 0.5$, SPIRE adds Main to Just Spurious and adds Spurious to Just Main with probability $\frac{p-0.5}{p-1}$ and, as a result, P(Main | Unusual Placement) = 1. Similarly, SPIRE removes Spurious from Just Spurious and removes Main from Just Main with the same probability and, as a result, P(Main | Grey Box) = 0. As a result, $\{$Both, Neither$\}$ each receive $0.5(0.5-p)$ images from each of $\{$Just Main, Just Spurious$\}$ and have an augmented size of $0.5(1-p)$. So SPIRE produces the balanced distribution while creating the potential for new SPs.

## C.3 Evaluation Details

### C.3.1 Why not set P(Spurious | Main) = P(Spurious | not Main) = P(Spurious) for the Balanced Distribution?

Using P(Spurious) instead of 0.5 may be an intuitive choice because it would mean that the main statistical difference between the original and balanced distributions is that Main and Spurious are now independent. However, doing so can have dramatic and unexpected effects on which splits are more important for evaluation. To see this, consider Main = "fork" and Spurious = "dining table". For the original distribution, we have P(Spurious | Main) $= 0.76$ which means we have, roughly, a 3:1 ratio of images in Both to Just Main. For the balanced distribution, using $\lambda =$ P(Spurious) $= 0.1$ would change this ratio to 1:9. Not only would this choice change which split is more important for evaluation (from Both to Just Main) but it would also would increase the degree to which that split is more important (from a factor of 3 to a factor of 9). Without domain knowledge telling us that such a dramatic shift is warranted, using 0.5 is the more conservative option because assigning equal importance to images with and without Spurious never flips which splits are more important for evaluation.

### C.3.2 Why do small, in absolute terms, Hallucination gaps matter?

To understand this, consider the per split accuracies for the tennis racket example (Figure 4.3 Left) where we observe that the Hallucination gap is "only" 0.5% and may be tempted to conclude that it is not significant. However, when we look at where the model's errors come from on the original distribution, we find that roughly 40% of them come from Just Spurious, despite the model's 99.5% accuracy on this split. This means that the model's performance is sensitive to both small changes to its accuracy on Just Spurious and Neither and distribution shifts that move weight between Just Spurious and Neither.

As a result, small, in absolute terms, changes to the Hallucination gap can have large impacts on the model's robustness to distribution shift. In general, we adjust for this by measuring changes in the gap metrics relative to their original value (*e.g.,* if the new model had a hallucination gap of 0.25% we would say that it "reduced the hallucination gap by a factor of 50%").

### C.3.3 Why can the Gap Metrics change much more than performance on the Balanced Distribution?

In general, mitigation methods shrink the gap metrics by sacrificing accuracy on the splits where relying on the SP is helpful in order to gain accuracy on the other splits; whether or not this trade-off improves performance on the balanced distribution depends on how much accuracy is sacrificed and gained. As a result, the size of the gap metrics and performance on the balanced distribution are not necessarily closely connected. As an extreme example of this, consider a hypothetical mitigation method that works by reducing the model's accuracy on the higher performing splits to match its accuracy on the lower performing splits: this will improve the gap metrics by setting them to zero, but it will harm performance on the balanced distribution.

### C.3.4 Counterfactual Evaluation

While the evaluations described in Section 4.4 are all based on the natural images, we also run a *counterfactual evaluation*. Unlike in SPIRE's identification step, where we only measure the probability that "removing Spurious from an image from Both" changes the model's prediction, this evaluation measures the probability that the model's prediction changes when we move an image from one split to another for each pairs of splits that differs by one object. This acts as an additional sanity check that a mitigation strategy has reduced the model's reliance on a SP, but we consider it to be less important than the model's performance on the balanced distribution and the gap metrics because its results depend on the specific definition of the counterfactuals used (*e.g.,* it is easy to do well on this evaluation for a specific type of counterfactual by training the model on that same type of counterfactual).

## C.4 Tennis Racket Example: Metrics and Mitigation Results.

Here, we walk through the evaluation described in Section 4.4 for class imbalanced problems using the tennis racket example. Figure C.2 shows the results. The numbers in the legends are "mean (standard deviation)" across 8 trials for the metric measured in that plot.

*Top Left: Average Precision.* This panel shows the model's Precision vs Recall curve, for the balanced distribution, which we use to calculate Average Precision by finding its Area Under the Curve (AUC). SPIRE improves Average Precision on the balanced distribution for this SP by 0.6%.

*Top Middle: Average Recall Gap.* This panel shows the model's recall gap (the absolute value of the difference of the model's accuracy on Both and Just Main) vs its Recall on the balanced distribution. We calculate this metric by finding the AUC. SPIRE decreases this metric by 31.4% which means that it produces a model that is more robust to distribution shifts that move probability between Both and Just Main.

*Top Right: Average Hallucination Gap.* This panel shows the model's hallucination gap (the absolute value of the difference of the model's accuracy on Just Spurious and Neither) vs its Recall on the balanced distribution. We calculate this metric by finding the AUC. SPIRE decreases this metric by 25.0% which means that it produces a model that is more robust to distribution shifts that move probability between Just Spurious and Neither.

*Center Row: Accuracy on Both and Just Main.* These panels plot the model's accuracy on Both/Just Main vs its Recall on the balanced distribution. The value shown is the AUC of this curve. Because the baseline model uses the presence of a person to help detect a tennis racket, we expect a model that does not rely on this SP to lose accuracy on Both and gain it on Just Main. SPIRE does this.

*Bottom Row: Accuracy on Just Spurious and Neither.* These panels plot the log of the model's accuracy on Just Spurious/Neither vs its Recall on the balanced distribution. The value shown is the AUC of this curve (before taking the log). Because the baseline model uses the presence of a person to help detect tennis rackets, we expect a model that does not rely on this SP to lose accuracy on Neither and gain it on Just Spurious. Because SPIRE improved AP, we do not see this because it's accuracy on these splits is higher for most levels of recall.

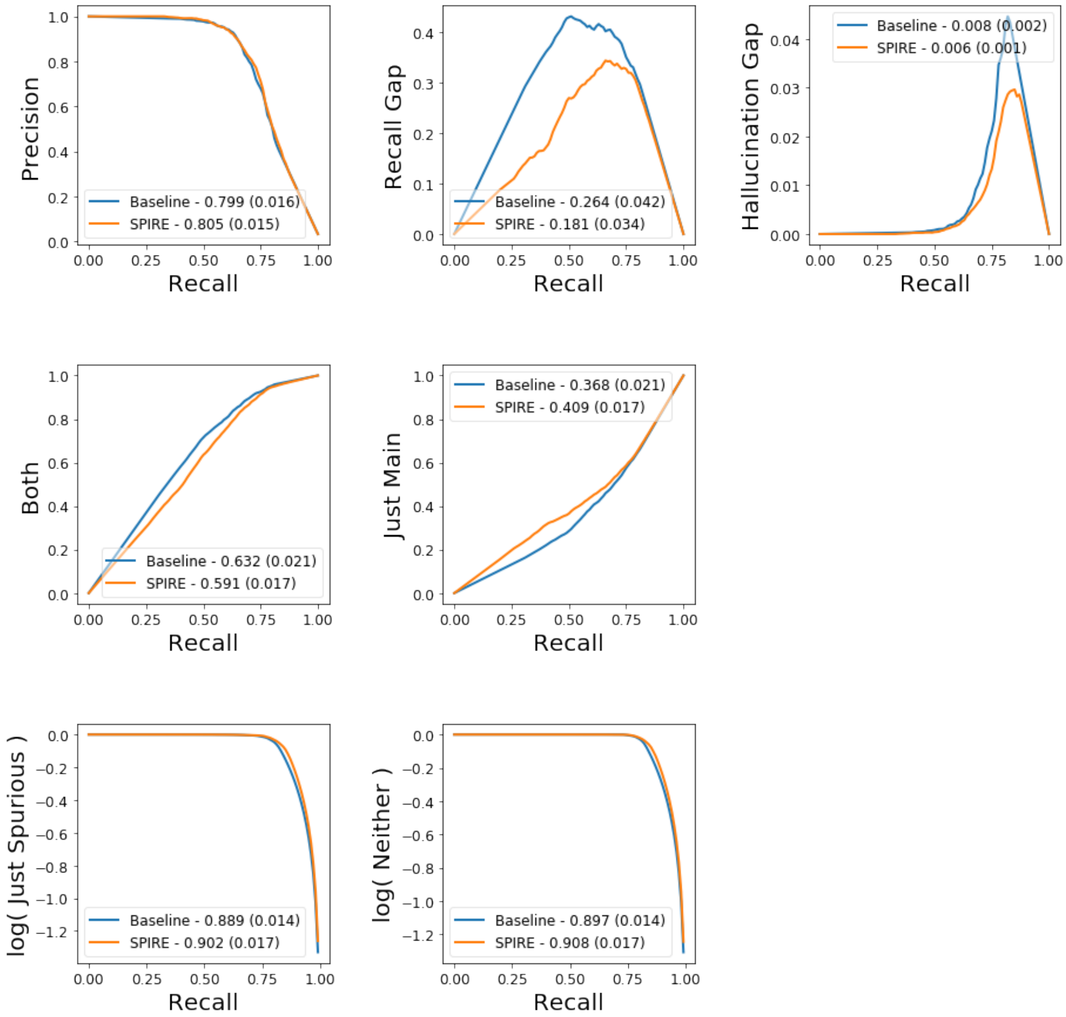


**Figure C.2:** The results of our evaluation for the tennis racket example. The numbers in the legends are "mean (standard deviation)" across 8 trials. SPIRE improved Average Precision on the balanced distribution by 0.6%, decreased the average recall gap by 31.4%, and decreased the average hallucination gap by 25.0%. Further, it had the expected effect of decreasing accuracy on Both and increasing it on Just Main. As a result, we conclude that it reduced the model's reliance on this SP.

## C.5 Model Training Details

Many of our experiments are based on the COCO dataset [60]. Because the test set for this dataset is not publicly available, we used its validation set as our test set and divided its training set into 90-10 training and validation splits.

All of our experiments started with the pretrained ResNet18 [38] that is available from PyTorch [69]. For each task, the classification layer was replaced with one of the appropriate dimension and then trained via transfer learning (*i.e.,* only the classification layer had its weights updated). The resulting model was then fine-tuned (*i.e.,* all of its weights were updated) to produce what we call the Baseline Model throughout this work.

**Optimization.** We minimized the binary cross entropy loss using Adam [47] with a batch size of 64. For transfer-learning, we used a learning rate of $0.001$ and, for fine-tuning, we used a learning rate of $0.0001$; we explored other options during early experiments, but found there was no benefit to doing so. If the training loss failed to decrease sufficiently after some number of epochs, we lowered the learning rate.

**Model selection.** During the training process, we selected the best model weights using their performance on the validation set. For the Benchmark Experiments, we measured performance using Accuracy and, for the Full Experiment, No Object Annotation Experiment, Scene Identification Experiment, and ISIC Experiment, we used F1. If the validation performance failed to increase sufficiently after some number of epochs, we stopped training.

**Benchmark Experiments: Hyper-parameter selection.** For this experiment, we tuned the hyper-parameters using balanced accuracy on the bottle-person pair with $p = 0.95$. For all methods, we considered both transfer-learning and fine-tuning, as applicable. For SPIRE, we considered both removing objects by covering them with a grey box and by in-painting them; we found that transfer-learning while covering objects with a grey box was the most effective (see Table C.2). RRR, CDEP, and GS all have regularization weights that can be tuned. FS has a minimum weight for images of objects "out of context" that can be tuned. For these methods, we considered values that are powers of 10 ranging from 0.1 to 10,000; no method chose one of the extreme values.

**Full Experiment: Hyper-parameter selection.** For this experiment, we tuned the hyper-parameters using the mean, across SPs, Average Precision on the balanced distribution for a model trained on 50% of the training dataset and then evaluated on the remaining 50% of the training dataset; we used such large chunk of the dataset for evaluation in order to be able to estimate the per split accuracies, which are required to calculate Average Precision on the balanced distribution.

For SPIRE, we use transfer-learning while covering objects with a grey box because this is what we found worked best in the Benchmark Experiments. However, we tune the weight of the augmentation by scaling $\delta$ from Setting 2 in Section 4.3.1 by a factor of $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$; intuitively, this is to prevent us from adding too many counterfactual images. Note that the weight for each SP is tuned independently and each weight is tuned by training a linear classifier.

For FS, the configuration chosen by this procedure yielded poor results and, consequently, we used the default value of 3 for our results [93].

**Table C.2:** The results of the Hyper-parameter Selection for SPIRE on the Benchmark Experiments. We see that SPIRE is consistently more effective when it retrains the model using transfer-learning and when it removes objects by covering them with grey boxes. Results shown are averaged across eight trials.

| Removal Strategy | Parameters Adjusted | Mean (Standard Deviation) |
|---|---|---|
| Grey Box | Transfer-Learning | 70.4 (1.7) |
| | Fine-Tuning | 69.4 (1.3) |
| In-Painting | Transfer-Learning | 70.2 (1.3) |
| | Fine-Tuning | 68.2 (1.4) |

### C.6 Additional Results: Benchmark Experiments - Section 4.5.1

**Creating the benchmark.** While varying $p$ allows us to control the strength of the correlation between Main and Spurious (*i.e.,* $p$ near 0 indicates a strong negative correlation while $p$ near 1 indicates a strong positive correlation), it does not guarantee that the model actually relies on the intended SP. Indeed, when we plot the models' balanced accuracy as $p$ varies (Figure C.3), we observe that $5$ out of the $13$ pairs show little to no loss in balanced accuracy as $p$ approaches 1 (dashed lines). Consequently, subsequent evaluation considers the other $8$ pairs (solid lines). For these pairs, the model's reliance on the SP increases as $p$ approaches $0$ or $1$ as evidenced by the increasing loss of balanced accuracy and confirmed via counterfactuals (Figure C.4).

**Counterfactual Evaluation.** For models that are trained on a dataset augmented with a specific type of counterfactual images, the results of this evaluation for that type of counterfactual are often skewed and, consequently, we exclude those results. Specifically, this means that: SPIRE is only evaluated on In-Painting counterfactuals, QCEC is not evaluated on In-Painting counterfactuals, and GS is not evaluated on counterfactuals that In-Paint Main.

Figure C.5 shows the results (averaged across the chosen object pairs and eight trials per pair). The first thing to note is that all of the counterfactual evaluations show that the Baseline model is relying on the intended SP because their results get worse as P(Main | Spurious) approaches 0 or $1$ (*i.e.,* there is a strong negative or positive correlation between Main and Spurious in the training dataset). Observe that SPIRE improves all of evaluations based on In-Painting with the exception of "Just Spurious and In-Paint Spurious" for $0.05 < p < 0.5$. In contrast, the other mitigation methods have clear and consistent failures (*e.g.,* RRR, CDEP, GS, and FS all make the evaluation worse for "Both and Remove Main", QCEC makes the evaluation worse for "Neither and Add Main").

**Per split analysis.** By looking at the models' accuracy on each split (Figure C.6, averaged across the chosen object pairs and eight trials per pair), we see that SPIRE exhibits all of the expected signs of a method that is reducing a model's reliance on a SP:

- It sacrifices accuracy on splits where relying on the SP is helpful (*e.g.,* Both for $p > 0.5$ and Just Main for $p < 0.5$) in order to gain accuracy on the splits where the SP is not helpful (*e.g.,* Just Main for $p > 0.5$ and Both for $p < 0.5$).
- It substantially flattens the per split accuracy curves for images with Spurious and, to a lesser extent, flattens them for images without Spurious. This indicates that it produces a model that is less sensitive to the original training distribution.

**Comparing Augmentation Strategies.** In this experiment, we want to explore the specific effect of SPIRE's augmentation strategy. We do this by comparing SPIRE to modified versions of the two augmentation-based methods that we consider:

- QCEC-Aug, which augments the dataset with images where either Main or Spurious has been removed (uniformly at random, as applicable). Compared to QCEC, it differs in that it removes objects by covering them with a grey box (as SPIRE does) instead of in-painting them.

- GS-Aug, which augments the dataset with images where Main has been removed (if possible). Compared to GS, it differs in that it it removes objects by covering them with a grey box (as SPIRE does) instead of in-painting them and in that it does not include GS's regularization term.

In Figure C.7 shows the results:

- SPIRE is the most effective method for increasing Balanced Accuracy and decreasing the Recall Gap.
- While QCEC-Aug and GS-Aug are somewhat better at reducing the Hallucination Gap, this usually is not worth the loss in Balanced Accuracy.
- Note that the results for the baseline model and for SPIRE are slightly different than they are in Figure 4.4 because these results are based on re-running this experiment from scratch.

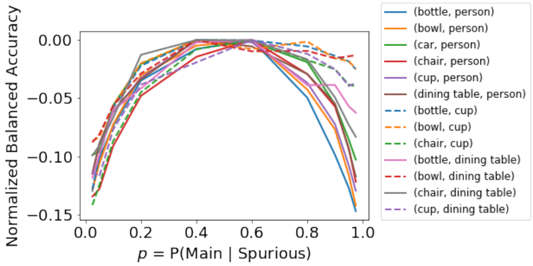


**Figure C.3:** For each pair of objects, we plot the models' balanced accuracy as we vary $p$ for the training set. The y-axis is normalized so that we can easily compare the curvature of the plots. We either accept (solid line) or reject (dashed line) pairs based on whether or not we see a significant drop in balanced accuracy both as $p$ approaches 0 and as it approaches 1. The rejected pairs show an insufficient drop as $p$ approached 1.

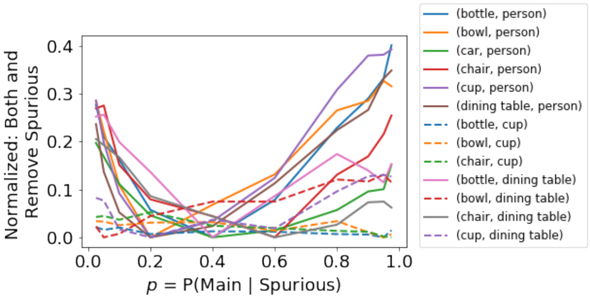


**Figure C.4:** For each pair of objects, we plot the metric SPIRE uses to identify SPs as we vary $p$ for the training set. The y-axis is normalized so that we can easily compare the curvature of the plots. We see that SPIRE generally agrees with the decision to accept (solid line) or (dashed line) pairs from the benchmark because the accepted pairs generally show more curvature.

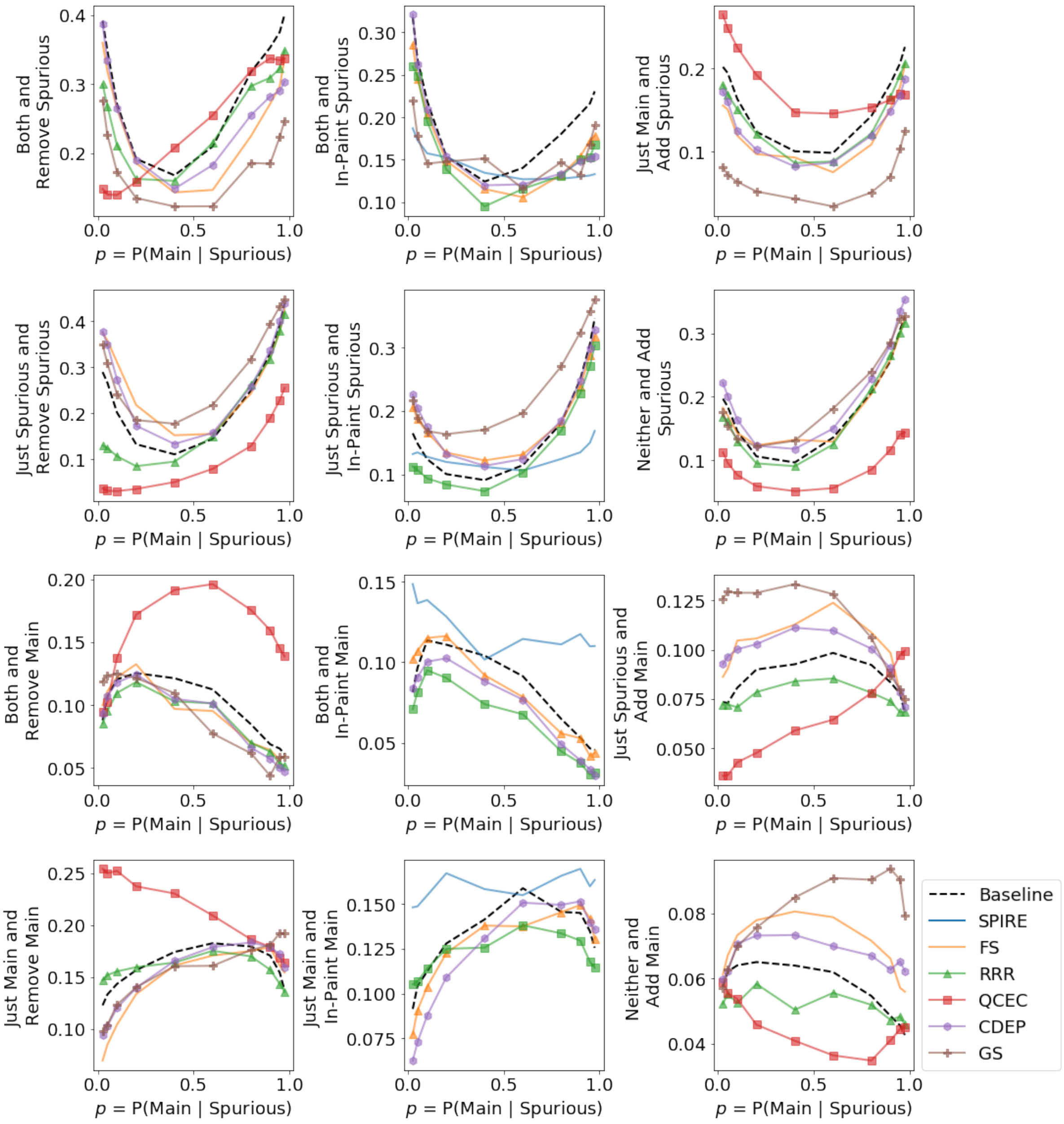


**Figure C.5:** The columns correspond to Removing, In-Painting, and Adding an object. The first two rows do that to Spurious and, as a result, a lower value is better. The last two rows do that to Main and, as a result, a higher value is better. Methods that train on an augmented dataset that contains a certain type of counterfactual are excluded from its evaluation because their results are usually skewed.

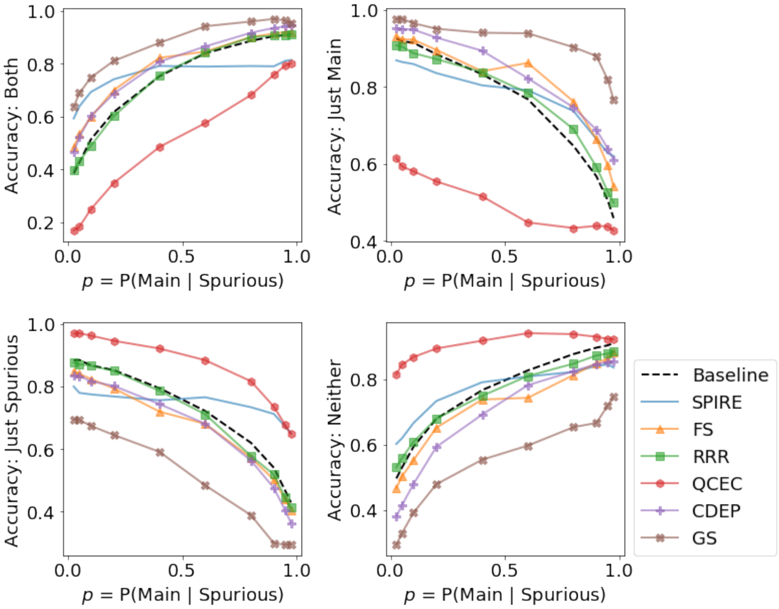


**Figure C.6:** The models' accuracies on each split.

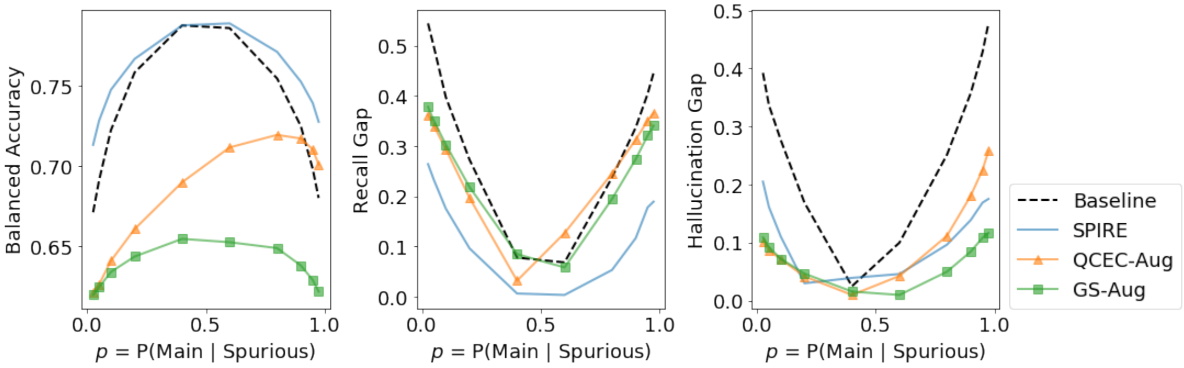


**Figure C.7:** A summary of the comparison of various augmentation strategies for the Benchmark Experiments.

## C.7 Additional Results: Full Experiment - Section 4.5.2

**Validating the Identified SPs.** In Figure C.8, we verify that the model is generally robust to masking objects. In Figure C.9, we verify that the model has large recall and hallucination gaps for the identified SPs. Both of these results indicate, in different ways, that the model is indeed relying on the SPs that SPIRE identifies.

**SPIRE's effect on each SP.** Figures C.10, C.11, and C.12 show SPIRE's effect on the Balanced Average Precision, the Average Recall Gap, and Average Hallucination Gap respectively. SPIRE improved Balanced Average Precision by an average of 1.1% with a positive change for 21 of the SPs. SPIRE decreased the Average Recall/Hallucination Gaps by an average factor of 14.2%/28.1% for 24/29 of the SPs. Overall, these results indicate that SPIRE consistently reduces the model's reliance the identified SPs.

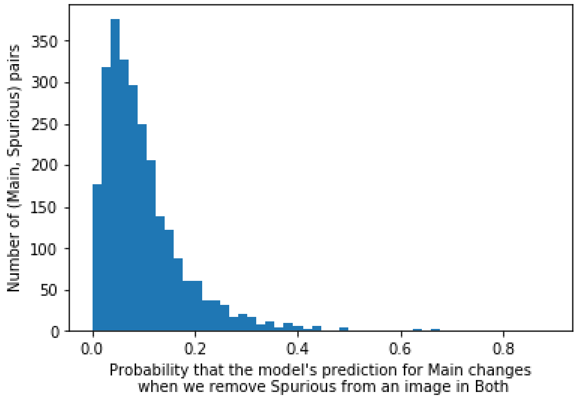


**Figure C.8:** A histogram of the metric that SPIRE uses to identify SPs. In general, the model's predictions are quite robust to masking objects: changing a mean of 10.0% and a median of 7.7% of the time. As a result, it is unlikely we can explain the fact that the model's prediction changes more than 40% of the time for the identified SPs using the fact that these images are out-of-distribution (they contain grey boxes).

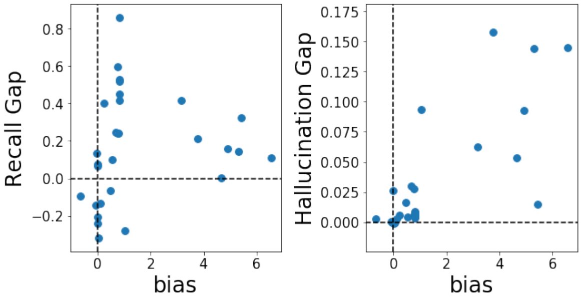


**Figure C.9:** When a model is relying on a SP, we expect positive gap metrics, if it has positive bias, and negative gap metrics, if it has negative bias. In general, this is what we find. **(Left)** A comparison of the Recall Gap to the bias of the dataset for the SP. **(Right)** A comparison of the Hallucination Gap to the bias.

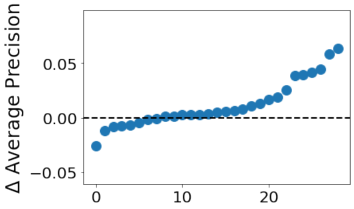


**Figure C.10:** The Average Precision on the balanced distribution for SPIRE compared to the baseline for each SP.

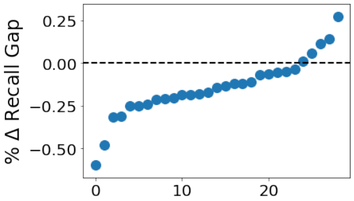


**Figure C.11:** The percent change of the Average Recall Gap for SPIRE compared to the baseline model for each SP.

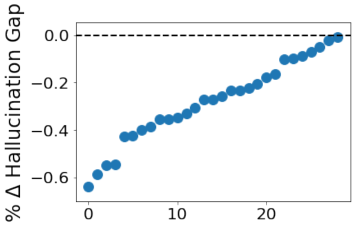


**Figure C.12:** The percent change of the Average Hallucination Gap for SPIRE compared to the baseline model for each SP.

## C.8 Additional Results: Generalization Experiments - Section 4.5.3

### C.8.1 SPIRE-R - Projection PseudoCode

Algorithm 6 details the process that we use for “adding” or “removing” Spurious from a model’s representation. This process is guaranteed to change the initial linear model’s prediction for whether or not the representation contains Spurious, which suites our goal of changing the most “obvious” signal for Spurious in the representation. Future work could explore more ambitious goals (*e.g.,* removing all of the information about Spurious from the model’s representation).

**Algorithm 6** Our algorithm for “adding” or “removing” Spurious from a model’s representation.

```
Require: $\{r_i\}_{i=1}^n$ {The model's penultimate layer's representation of each image}
Require: $\{y_i\}_{i=1}^n$ {A binary label for whether or not each image contains Spurious}
Require: $c$ {A confidence threshold, we used 0.0001}
Require: $s$ {A step size, we used 0.1}
  $w, b \leftarrow LogisticRegression(\{r_i\}, \{y_i\})$ {Train a linear model to predict if an image contains Spurious}
  for i = 1, ..., n do
    $r'_i = r_i$ {Initialize the counterfactual representation}
    if $y_i == 1$ then {If the image contains Spurious, remove it}
      $y'_i \leftarrow 0$
      while $\frac{1}{1+e^{-(wr'_i+b)}} > c$ do
        $r'_i \leftarrow r'_i - s * w$ {Maximally decrease the linear model's confidence that Spurious is present}
      end while
    else {If the image doesn't contain Spurious, add it}
      $y'_i \leftarrow 1$
      while $\frac{1}{1+e^{-(wr'_i+b)}} < 1 - c$ do
        $r'_i \leftarrow r'_i + s * w$ {Maximally increase the linear model's confidence that Spurious is present}
      end while
    end if
  end for
output Return $\{r'_i\}, \{y'_i\}$
```

### C.8.2 ISIC Experiment- Pipeline for creating counterfactuals

Our pipeline, which is based on clustering image segments (*i.e.,* super-pixels), is constructed as follows:

- We use an image segmentation algorithm to extract segments from an image and represent each segment using its mean RGB value.

- We run a hierarchical clustering on those RGB values to produce nine clusters. Then, we manually inspect several randomly sampled images from each cluster and label those clusters based on whether or not they represent stickers.
- Finally, we use a K-NearestNeighbor classifier to predict which of those nine clusters an image segment belongs to.

Overall, this pipeline produces a per-image map of which pixels belong to a sticker and identifies stickers with 86.7% recall and 99.0% precision. We use this map to produce counterfactual images.

Note that this pipeline significantly reduced the cost of producing these pixel-wise annotations because it only required labeling a small number of image segments as “sticker or not.” However, the annotations are noisier than manually collected annotations would be.

# Appendix D

# Blindspots

### D.1 Generating Experimental Configurations, Extended

In this section we detail how we use `SpotCheck` to generate random experimental configurations.

- In Section D.1.1, we define the different types of semantic features that can appear in each image.
- In Section D.1.2, we define a synthetic image dataset, how we generate random datasets, and how we sample images from a dataset.
- In Section D.1.3, we define a blindspot for a synthetic image dataset, how we generate a random blindspot, and how we generate an unambiguous set of blindspots.

### D.1.1 Semantic Features

Table D.1 defines all of the semantic features that `SpotCheck` uses to generate synthetic images. We call these semantic features *Attributes* and group them into *Layers* based on what part of an image they describe. Each Attribute has two possible *Values*: a Default and Alternative Value. Each synthetic image has an associated list of (Layer, Attribute, Value) triplets that describes the image. Figure D.1 shows this triplet list for two synthetic images.

We sometimes refer to the Square/Rectangle/Circle/Text Layers as *Object Layers* because they all describe a specific object that can be present in an image. The location of each object within an image is chosen randomly, subject to the constraint that each object doesn't overlap with any other object.

**Table D.1:** The Layers and Attributes that define the synthetic images.

| Layer | Attribute | Default Value | Alternative Value |
|---|---|---|---|
| Background | Color | White | Grey |
| | Texture | Solid | Salt and Pepper Noise |
| Square/Rectangle/Circle/Text | Presence | False | True |
| | Size | Normal | Small |
| | Color | Blue | Orange |
| | Texture | Solid | Vertical Stripes |
| Square (continued) | Number | 1 | 2 |

### D.1.2 Defining a Dataset using these Semantic Features

At a high level, `SpotCheck` defines a *Dataset* by deciding whether or not each Attribute of each Layer is *Rollable* (*i.e.,* the Attribute can take either its Default or Alternative Value, uniformly at random) or not Rollable (*i.e.,* the Attribute only takes its Default Value). We measure a Dataset's complexity using the number of Rollable Attributes it has. Figure D.1 describes the Rollable and Not Rollable Attributes for an example Dataset.

**Generating a Random Dataset.** We start by picking which Layers will be part of the Dataset:

- Images need a background, so all Datasets have the Background Layer.
- The task is to predict whether there is a square in the image, so all Datasets have the Square Layer.
- We add 1-3 (chosen uniformly at random) of the other Object Layers (chosen uniformly at random without replacement from the set {Rectangle, Circle, Text}) to the Dataset.

Once the Layers are chosen, we make 6-8 (chosen uniformly at random) of the Attributes Rollable:

- Each Object Layer has its Presence Attribute made Rollable.
- Then, the remaining Rollable Attributes are chosen by iteratively:
  - Selecting a Layer uniformly at random from those that have at least one Not Rollable Attribute.
  - Selecting an Attribute from that Layer uniformly at random from those that are Not Rollable.

**Sampling an Image from a Dataset.** Once a Dataset's Rollable Attributes have been defined, generating a random image is straightforward:

- For each Attribute from each Layer in the Dataset, we pick a random Value if the Attribute is Rollable. Attributes that are Not Rollable will take their Default Value.
  - If the Layer is an Object Layer:
    - If the Presence Attribute is True, the location of the object is chosen randomly (subject to the non-overlapping constraint).
    - If the Presence Attribute is False, the object will not be rendered (regardless of the Values chosen for the other Attributes of this Layer).
- We then use the resulting (Layer, Attribute, Value) triplet list and the list of object locations to render a 224x224 RGB image.
- Finally, we calculate any MetaAttributes (explained next) and append these (Layer, MetaAttribute, Value) triplets to the image's definition list.

**Calculating MetaAttributes.** While each Attribute corresponds to a semantic feature, there are a potentially infinite number of MetaAttributes that one could calculate as semantically meaningful functions of an image. We list the MetaAttributes that we calculate in our experiments in Table D.2. Because this space is infinitely large and grows with the number of Attributes, we exclude MetaAttributes from our measure of Dataset complexity.

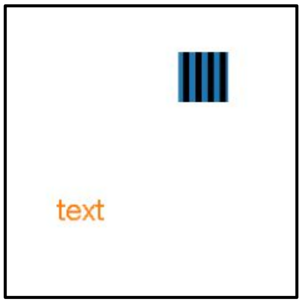
**Dataset Definition:**
Rollable Attributes

```
Background: {Color: False,
             Texture: True}

Square: {Presence: True,
         Size: False,
         Color: True,
         Texture: True,
         Number: False}

Text: {Presence: True,
       Size: False,
       Color: True,
       Texture: False}
```

Example Image #1

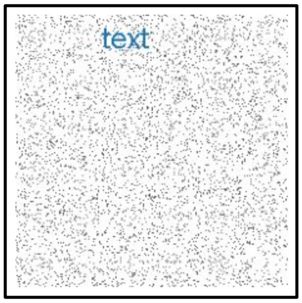


```
(Background, Texture, Solid),

(Square, Presence, True),
(Square, Color, Blue),
(Square, Texture, Vertical
Stripes),

(Text, Presence, True),
(Text, Color, Orange}
```

Example Image #2



```
(Background, Texture, Salt &
Pepper),

(Square, Presence, False),
(Square, Color, Blue),
(Square, Texture, Solid),

(Text, Presence, True),
(Text, Color, Blue)
```

**Figure D.1: Top Row.** The definition of an example Dataset generated by `SpotCheck`. Notice that this Dataset has 3 Layers and 6 Rollable Attributes. **Middle/Bottom Row.** Two example images generated from this Dataset along with their (Layer, Attribute, Value) triplet lists. Notice that Not Rollable Attributes in this Dataset take on their Default Values in these images and are not in the images' triplet lists.

**Table D.2:** The MetaAttributes that we calculate for each synthetic image.

| Layer | MetaAttribute | Value | Meaning |
|---|---|---|---|
| Background | Relative Position | 1 | Square is above the horizontal centerline of the image |
| | | 0 | Square is bellow the horizontal centerline of the image |
| | | -1 | No Square |

### D.1.3 Defining the Blindspots for a Dataset

`SpotCheck` defines a *Blindspot* using a list of (Layer, (Meta)Attribute, Value) triplets. We measure a Blindspot's specificity using the length of its definition list. An image belongs to a blindspot if and only if the Blindspot's definition list is a subset of the image's definition list. Figure D.2 shows two example Blindspots.

**Generating a Random Blindspot.** `SpotCheck` generates a random Blindspot consisting of 5-7 (chosen uniformly at random) (Layer, (Meta)Attribute, Value) triplets for a Dataset by iteratively:

- Selecting a Layer (uniformly at random from those that have at least one Rollable Attribute[1] that is not already in this Blindspot)
- Selecting a Rollable Attribute from that Layer:
  - Object Layers: If the Layer's Presence Attribute is not in this Blindspot, select its Presence Attribute. Otherwise, select an Attribute uniformly at random from those that are not already in this Blindspot and set the Layer's Presence Attribute Value to True for this Blindspot.
  - Background Layers: Select an Attribute uniformly at random from those that are not already in this Blindspot.
- Selecting a Value for that Attribute (uniformly at random)

Notice that, if an Object Layer is selected more than once, then we ensure that the Object's Presence Attribute has a Value of True in the Blindspot definition. We enforce this *Feasibility Constraint* to ensure that every triplet in the Blindspot's definition list correctly describes the images belonging to the Blindspot (*e.g.,* [(Circle, Presence, False), (Circle, Color, Blue)] is infeasible because an image with a blue circle must have a circle in it).

**Generating an Unambiguous Set of Blindspots.** For each Dataset, we generate 1-3 (chosen uniformly at random) Blindspots using the process described above. However, when generating multiple blindspots, they can be *ambiguous* which causes problems when using them to evaluate BDMs.

***Definition.*** A set of Blindspots, $S_1$, is ambiguous if there exists a different set of Blindspots, $S_2$, such that both:

1. The union of images belonging to $S_1$ is equivalent to the union of images belonging to $S_2$. As a result, $S_1$ and $S_2$ would both correctly describe the model's blindspots.
2. An evaluation that uses Discovery Rate (Equation 5.6) would penalize a BDM if it returns $S_2$ instead of $S_1$. More precisely, DR$(S_2, S_1) < 1$ for $\lambda_p = \lambda_r = 1$.

***Example.*** Suppose that we have a very simple Dataset with two Rollable Attributes, $X$ and $Y$ which are uniformly distributed and independent, and consider two different sets of Blindspots for this Dataset:

- $S_1 = \{B_1, B_2\}$ where $B_1 = [(X = 1)]$ and $B_2 = [(X = 0), (Y = 1)]$
- $S_2 = \{B_1^{'}, B_2^{'}\}$ where $B_1^{'} = [(X = 1), (Y = 0)]$ and $B_2^{'} = [(Y = 1)]$

[1] All MetaAttributes are considered to be "Rollable" when generating a random Blindspot.

Then, $S_1$ is ambiguous because:

- $S_1$ and $S_2$ induce the same behavior in the model: they both mislabel an image if $X = 1 \vee Y = 1$.
- A BDM would be penalized for returning $S_2$:

$$\text{BP}(B_1^{'}, B_1) = 1.0 \wedge \text{BP}(B_1^{'}, B_2) = 0 \wedge \text{BP}(B_2^{'}, B_1) = \text{BP}(B_2^{'}, B_2) = 0.5 \implies$$

$$\text{BR}(S_2, B_1) = 0.5 \wedge \text{BR}(S_2, B_2) = 0 \implies$$

$$\text{DR}(S_2, S_1) = 0$$

In fact, for this example, there are only two sets of two unambiguous Blindspots, ($\{[(X = 0), (Y = 0)], [(X = 1), (Y = 1)]\}$ and $\{[(X = 0), (Y = 1)], [(X = 1), (Y = 0)]\}$), and there exists no unambiguous set of three Blindspots.

***Preventing Ambiguity.*** In general, ambiguity occurs whenever the union of two blindspots forms a contiguous region in the discrete space defined by the Rollable Attributes. Consequently, we prevent ambiguity by ensuring that any pair of blindspots has at least two of the *same* Rollable Attributes with *different* Values in their definition lists. We call this the *Ambiguity Constraint*.

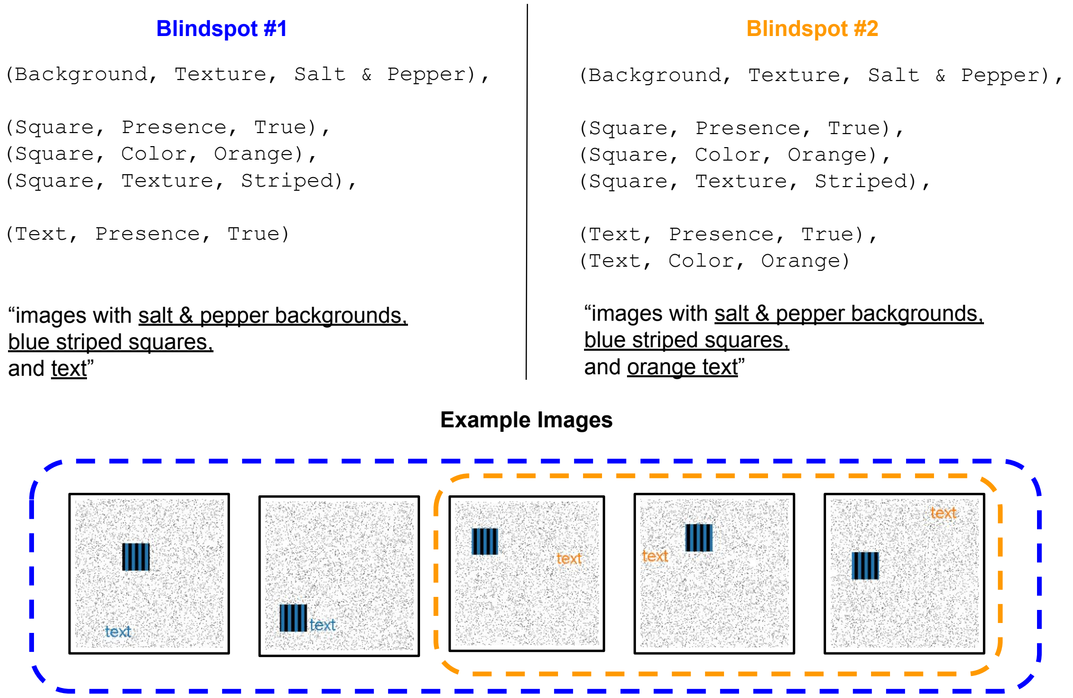
***Implications of the Ambiguity and Feasibility Constraints.*** In our experiments, our goal is to generate experimental configurations with a diverse set of Datasets and associated Blindspots. However, the Ambiguity Constraint (AC) and Feasibility Constraint (FC) limit the number of valid Blindspots for any specific Dataset.

To see this, notice that the AC places more constraints on each successive Blindspot added to an experimental configuration. This has two implications. First, that generating an experimental configuration with more Blindspots requires a Dataset with more Rollable Attributes (more complexity) and Blindspots with more triplets (more specificity). Further, because we cannot set the Attribute Values of a Blindspot's triplets independently of each other [FC], we need more complexity and specificty than a simple analysis based only on the AC suggests. Second, that each successive Blindspot is more closely related to the previous ones which means that larger sets of Blindspots are "less diverse" or "less random" in some sense.

With these trade-offs in mind, we generated experimental configurations with:

- Background, Square, and 1-3 other Object Layers
- A total of 6-8 Rollable Attributes
- 1-3 Blindspots
- 5-7 triplets per Blindspot

because an experimental configuration with any combination of these values is able to satisfy the AC and the FC while still having a diverse set of Blindspots.



Blindspot #1

```
(Background, Texture, Salt & Pepper),

(Square, Presence, True),
(Square, Color, Orange),
(Square, Texture, Striped),

(Text, Presence, True)
```

“images with salt & pepper backgrounds,
blue striped squares,
and text”

Blindspot #2

```
(Background, Texture, Salt & Pepper),

(Square, Presence, True),
(Square, Color, Orange),
(Square, Texture, Striped),

(Text, Presence, True),
(Text, Color, Orange)
```

“images with salt & pepper backgrounds,
blue striped squares,
and orange text”

**Figure D.2:** Two example Blindspots, Blindspot #1 (Left) and Blindspot # 2 (Right) for the example Dataset from Figure D.1. Each Blindspot is defined by a list of (Layer, (Meta)Attribute, Value) triplets as shown above. We also display example images belonging to each Blindspot: all of the images inside of the blue border belong to Blindspot #1, while only the subset of images inside of the orange border belong to Blindspot #2. In this example, Blindspot #2 is more specific (defined using 6 semantic features) than Blindspot #1 (defined using 5 semantic features).

## D.2 How do our definitions of “belonging to” and “covering” build on similar ideas in existing work?

At a high level, we refine these ideas so that they better reflect the fact that, in practice, a user is going to look at the hypothesized blindspots returned by a BDM in order to come to conclusions about the model’s true blindspots. This entails making three specific changes.

First, our definition of “belonging to” uses a high, in absolute terms, threshold for $\lambda_p$ (*i.e.,* 0.8) in order to minimize the amount of noise the user has to deal with when determining the semantic definition of a blindspot. This is in contrast to a relative definition (*e.g.,* better than random chance) which would lead to significant amounts of noise for uncommon blindspots. For example, Table 2 from Sohoni et al. [97] shows that as few as one in seven images in the hypothesized blindspots are actually from the true blindspots for the Waterbirds and CelebA datasets. While better than random chance, the output of the BDM is still very noisy, which is likely to create problems for the user.

Second, our definition of “covering” incorporates blindspot recall in order to prevent the user from coming to conclusions that are too narrow. For example, consider the middle row of Figure 5 from Eyuboglu et al. [33]. In these examples, the blindspots are defined as “people wearing glasses,” “people with brown hair,” or “smiling people.” However, the BDM returns images that are of “men wearing glasses,” “women with brown hair, and “smiling women.” As a result, the user may incorrectly conclude that the model is exhibiting gender bias because the BDM led them to conclusions that are too narrow.

Third, our definition of “covering” allows for hypothesized blindspots to be combined in order to take advantage of the user’s ability to synthesize what they see and come to more general conclusions, which is something that would be very challenging to do algorithmically. Experimentally, we observe that this combining is essential for some of these BDMs to work on `SpotCheck`. Further, in Appendix D.6, we qualitatively observe that this useful for real problems as well. For example, the “ties without people” blindspots includes a set of images of “cats wearing ties” that are separate, in the model’s representation space, from the other images in the blindspot and that, as a result, are likely to be returned separately by a BDM.

## D.3 Hyperparameter Tuning

To tune hyperparameters, we use a secondary set of 20 ECs to calculate BDM DR and FDR. While some of these BDMs have multiple hyperparameters, we focused on the main hyperparameter for each BDM and left the others at their default values.

**Barlow [94].** The main hyperparameter is the *maximum depth* of the decision tree whose leaves define the hypothesized blindspots. The results of the hyperparameter search are in Table D.3.

**Spotlight [26].** The main hyperparameter is the *minimum weight* assigned to a hypothesized blindspot. The results of the hyperparameter search are in Table D.4.

**Domino [33].** The main hyperparmeter is $\gamma$ which controls the relative importance of coherence and under-performance in the hypothesized blindspots. The results of the hyperparameter search are in Table D.5.

`PlaneSpot`**.** For our BDM, the main hyperparameter is $w$ which controls the *weight* of the model-confidence dimension relative to the two spatial dimensions from the scvis embedding. The results of the hyperparameter search are in Table D.6.

**Table D.3:** The hyperparameter tuning results for Barlow. For maximum depths of 11 and 12, we noticed that Barlow's outputs matched exactly for all of the configurations. This indicates that it never learned a decision tree of depth 12. As a result, we use a maximum depth of 11 for our experiments.

| Maximum Depth | DR | FDR |
|---|---|---|
| 3 | 0.07 | 0.0 |
| 4 | 0.14 | 0.0 |
| 5 | 0.19 | 0.0 |
| 6 | 0.27 | 0.12 |
| 7 | 0.36 | 0.15 |
| 8 | 0.37 | 0.09 |
| 9 | 0.43 | 0.11 |
| 10 | 0.43 | 0.08 |
| 11 | 0.48 | 0.07 |
| 12 | 0.48 | 0.07 |

**Table D.4:** The hyperparameter tuning results for Spotlight. We use a minimum weight of 0.01 in our experiments. In Figure 5.2 we show that, despite the fact that 0.01 and 0.02 have very similar average results, they behave very differently.

| Minimum Weight | DR | FDR |
|---|---|---|
| 0.005 | 0.49 | 0.04 |
| 0.01 | 0.88 | 0.06 |
| 0.02 | 0.88 | 0.08 |
| 0.04 | 0.48 | 0.00 |

**Table D.5:** The hyperparameter tuning results for Domino. We use $\gamma = 10$ in our experiments.

| $\gamma$ | DR | FDR |
|---|---|---|
| 5 | 0.48 | 0.06 |
| 10 | 0.72 | 0.03 |
| 15 | 0.70 | 0.06 |
| 20 | 0.60 | 0.09 |

**Table D.6:** The hyperparameter tuning results for `PlaneSpot`. We use $w = 0.025$ in our experiments.

| $w$ | DR | FDR |
|---|---|---|
| 0 | 0.57 | 0.00 |
| 0.025 | 0.95 | 0.00 |
| 0.05 | 0.88 | 0.00 |
| 0.1 | 0.87 | 0.00 |

### D.4 Why are these BDMs failing?

In this section, we are going to try to identify, from a methodological standpoint, why these BDMs are failing. We start by defining some additional metrics that allow us to understand why a BDM failed to cover a specific true blindspot. Then, we describe how we average those metrics across our ECs to observe general patterns. Finally, we give some possible explanations for the observed trends.

**Explaining a Specific Failure.** Given the output of a BDM, $\hat{\mathbf{\Psi}}$, we want to understand why it failed to cover a specific true blindspot, $\Psi_m$. To do this, we measure the fraction of the images in that true blindspot, $i \in \Psi_m$, that fall into each of four categories:[2]

- $i$ was *not returned* by the BDM. Intuitively, this tells us how often the BDM does not show an image to the user that it should. Specifically, this means that $i \notin \hat{\Psi}_k \; \forall k$.
- $i$ is *found* by the BDM. Intuitively, this tells us how close the BDM is to covering $\Psi_m$. Specifically, this means that $\exists k \mid i \in \hat{\Psi}_k \wedge \text{BP}(\hat{\Psi}_k, \Psi_m) \geq \lambda_p$.
- $i$ was part of a *merged* blindspot according to the BDM. Intuitively, this tells us how often the BDM merges multiple true blindspots into a single hypothesized blindspot. Specifically, this means that $\exists k \mid i \in \hat{\Psi}_k \wedge \text{BP}(\hat{\Psi}_k, \cup_m \Psi_m) \geq \lambda_p$. Note that $\text{BP}(\hat{\Psi}_k, \cup_m \Psi_m)$ measures the fraction of the images in $\hat{\Psi}_k$ that belong to *any* true blindspot.
- $i$ was part of a *impure* blindspot according to the BDM. Intuitively, this tells us how often the BDM includes too many images that do not belong to any true blindspot into a hypothesized blindspot. Specifically, this means that $\text{BP}(\hat{\Psi}_k, \cup_m \Psi_m) < \lambda_p \; (\forall k \mid i \in \hat{\Psi}_k)$.

**Averaging to gain more general patterns.** While these metrics can help us understand why a BDM failed to cover a specific true blindspot from a specific EC, we want to arrive at more general results. To do this, we first average these metrics across the true blindspots in an EC that are not covered by the BDM, $\{\Psi_m \mid \text{BR}(\hat{\mathbf{\Psi}}, \Psi_m) < \lambda_r\}$, and then average them across the ECs where the BDM did not cover all of the true blindspots, $\{\mathbf{\Psi} \mid \text{DR}(\hat{\mathbf{\Psi}}, \mathbf{\Psi}) \neq 1\}$. Table D.7 shows the results.

**Explaining observed trends.** Our main observation is that a significant portion of all of the BDMs' failures can be explained by the fact that they tend to merge multiple true blindspots into a single hypothesized blindspot. However, this raises the question: is the a failure of the BDMs or are they simply inheriting the problem from their image representation, which is failing to separate the true blindspots? To address this question, we manually inspected the 2D scvis embedding used by `PlaneSpot`, which exhibits this failure the most strongly, for the 10 ECs where at least 50% of the images in the true blindspots that were not covered belonged to a hypothesized blindspot that merges multiple true blindspots. For 8 of those 10 ECs, the true blindspots were easily visually separable in this 2D embedding, which means that the true blindspots are also separable in the original image representation. Consequently, we conclude that this is a failing of the BDMs.

Additionally, we make two less significant observations. First, that Spotlight is the only BDM that fails to return a non-trivial fraction of the images that it is should. Because it is the only

[2] Every image, $i$, falls into exactly one of these categories. The definition of each category is written under the assumption that $i$ does not fall into any of the previous categories.

BDM that does not do this, we suggest that BDMs should partition the entire input space (including partitions that do not have higher than average error). Second, that Barlow is the only that has more failures due to returning impure hypothesized blindspots than merged blindspots. Between this and Barlow's generally poor performance, this suggests that axis-aligned decision trees are not a particularly promising hypothesis class for BDMs.

**Table D.7:** The average value of each of the four metrics we use to explain why a BDM failed to cover a true blindspot.

| Method | Not Returned | Found | Merged | Impure |
|---|---|---|---|---|
| Barlow | 0.02 | 0.15 | 0.39 | 0.44 |
| Spotlight | 0.13 | 0.46 | 0.33 | 0.04 |
| Domino | 0.0 | 0.38 | 0.36 | 0.24 |
| `PlaneSpot` | 0.0 | 0.0 | 0.57 | 0.42 |

## D.5 COCO Experiments, Details

We detail the methodology used to generate ECs using the COCO dataset [60].

### D.5.1 Prediction Task & Data Preprocessing

COCO is a large-scale object detection dataset. We use the 2017 version of the dataset. We re-sample from the given train-test-validation splits to have a larger test set of images for use in blindspot discovery. Our final train, test, and validation sets have 66874, 44584, and 11829 images respectively.

**Prediction Task.** We define the binary classification task as detecting whether an object belonging to some *super-category* (containing multiple of the 80 object categories) is present in the image. In our experiments, each EC is associated with 1 of 3 different super-category prediction tasks: detecting if an *animal* (*e.g.,* a dog, cat, etc), a *vehicle* (*e.g.,* a car, airplane, etc), or a *furniture item* (*e.g.,* a chair, bed, etc) are present in an image. We use the super-category definitions provided in the COCO paper [60]. We sample ECs from multiple (as opposed to only a single) prediction tasks so that we have a larger number of possible blindspot definitions.

**EC Preprocessing.** For each EC, we use the EC's prediction task to down-sample the original COCO dataset to ensure that the dataset's blindspots are identifiable (*i.e.,*, so that there is no ambiguity or overlap in the blindspots we induce in our models). Specifically, we drop all images that have more than 1 unique object category belonging to the task super-category (*e.g.,*, if the task is to detect animals, we drop all images that have both a dog and a cat). We also down-sample so that the final train, test, and validation sets are all class-balanced.

### D.5.2 Blindspot Definitions

All blindspots in our ECs are defined using an object category that is a subset of those belonging to the task super-category. For example, an EC with the "animal" prediction task can have the blindspot "zebra", as the zebra object category belongs to the animal super-category (Figure 5.5).

**Specificity.** To measure the effect of blindspot specificity, we generate two different *types* of blindspots: more and less specific. Less-specific blindspots are defined using only one object category, such as the "zebra" blindspot. More-specific blindspots are defined using two object categories. Like the less-specific blindspots, one of the object categories must belong to the task super-category (*e.g.,*, must be an animal for an animal prediction EC). But, the second object category used to define a more-specific blindspot must *not* belong to the task super-category. For example, the second object can be "person" (but not "dog") because a person is not an animal.

Each more-specific blindspot is then defined using (1) the presence of the object belonging to the super-category, and (2) either the presence *or absence* of the object outside of the super-category. For example, valid blindspot definitions include "images of elephants with people" (task: animal), "images of bicycles without a backpack" (task: vehicle), or "images of dining tables with forks" (task: furniture item). We chose to allow some blindspots to be defined using the *absence* of a second object inspired by the past observation that models can incorrectly rely on the presence of spurious artifact objects instead of identifying the true object class [75], consequently underperforming on images without these artifacts.

### D.5.3 Model Training

Similarly to our experiments using `SpotCheck`, we induce true blindspots by training with the wrong labels for images belonging to blindspots. For each EC, we train a Resnet-18. We perform model selection by picking the model with the best validation set loss.

**Validating that models learn the induced blindspots.** Because real data is more complex than synthetic data, it is possible that even when we train with the wrong labels for a particular subgroup, the model may not underperform on other images belonging to the subgroup. We call this behavior *failing to induce* a true blindspot. For each true blindspot, we measure if the blindspot was successfully induced by comparing the model's performance on images belonging to, vs. outside of the blindspot in the separate *test set*. We only retain ECs where the model has at least a 20% recall gap inside vs. outside of the blindspot.

### D.5.4 ECs: General Pool

We began our analysis by replicating the procedure we used to sample random ECs using `SpotCheck` as closely as possible. Recall that when using `SpotCheck` to measure the effect of various factors, we generated a pool of ECs where each factor (*e.g.,*, the number of model blindspots and each blindspot's specificty) were chosen *independently*. We repeat this general procedure using the COCO dataset, and call these ECs the "General Pool".

We provide pseudocode that we use to sample a pool of $N = 90$ ECs:

```
• For each super-category prediction task in $\{$animal, vehicle, furniture item $\}$:
  ○ For each number of blindspots in $[1, 2, 3]$:
    • Generate $10$ experimental configurations. For each configuration:
      ○ For each blindspot, choose the blindspot's definition:
        • Choose the blindspot's specificity uniformly at random from $\{$More-Specific, Less-Specific$\}$.
        • Choose an object category that belongs to the task super-category uniformly at random from those that were not already used in the EC's other blindspots.
        • If the blindspot is more-specific:
          ○ Choose a second object category that does not belong to the task super-category uniformly at random from those that are eligible. An object category is eligible if it has not yet been used to define another of the ECs blindspots, and if both the intersection and set difference of the first and second object categories have at least $100$ images in the train set. We impose these eligibility constraints to avoid selecting blindspots that are too small, as we hypothesize that larger blindspots may be easier to induce.
          ○ Choose if the blindspot will be defined using the $\{$Presence, Absence$\}$ of the second object uniformly at random.
```

Note that for each EC sampled using the above pseudocode, the EC's number of blindspots and each indivudal blindspot's specificity are selected independently.

**Table D.8:** Counts the retained ECs in the general pool (where all blindspots are successfully induced) for each prediction task and number of blindspots.

| Prediction Task | Number of blindspots | **Count** of (ECs retained for analysis) / (total ECs sampled) |
|---|---|---|
| **animal** | 1 | 10 / 10 |
| | 2 | 8 / 10 |
| | 3 | 7 / 10 |
| **furniture** | 1 | 8 / 10 |
| | 2 | 6 / 10 |
| | 3 | 5 / 10 |
| **vehicle** | 1 | 8 / 10 |
| | 2 | 6 / 10 |
| | 3 | 7 / 10 |

**Summary Statistics.** Of the $90$ sampled ECs in the general pool, we retain $65$ for our analysis, dropping $25$ where we failed to induce the sampled blindspot. Table D.8 summarizes the prediction tasks and number of blindspots in the retained general pool of ECs.

### D.5.5 ECs: Conditioned Pools

**Number of Blindspots: Conditioned Pool.** We provide pseudocode that we use to sample a pool of $N = 84$ ECs. We hold each blindspot's specificity constant and only sample less-specific blindspots.

Before sampling the conditioned pool ECs, we first train several models where for each model, we try to induce a single COCO object category as a blindspot. We sample the blindspots for the conditioned distribution only from the subset of $C$ object categories that were successfully induced as singleton blindspots.

- For each super-category **prediction task** in $\{$animal, vehicle, furniture item $\}$:
  - ○ For each **number of blindspots** in $[1, 2, 3]$:
    - Using the $C$ object categories that belong to the task super-category *and* that were able to be learned as singleton blindspots, sample up to $N = 20$ ECs from the list of $C$ nCr (number of blindspots) possible combinations.

We only have $22$ ECs with $1$ (singleton) blindspot because there are only $22$ total object categories that can be induced as singleton blindspots. All blindspots were successfully induced for the $84$ sampled ECs. To measure the effect of the number of blindspots, we compare BDM performance the $22, 30$, and $30$ ECs with $1$, $2$, and $3$ blindspots respectively.

**Blindspot Specificity: Conditioned Pool.** We provide pseudocode that we use to sample a pool of $N = 70$ ECs. We hold the number of blindspots constant and only sample ECs with a single blindspot.

- For each super-category **prediction task** in $\{$animal, vehicle, furniture item $\}$:
  - ○ Using the $C$ object categories that belong to the task super-category *and* that were able to be learned as singleton blindspots, construct a list of all of the *eligible* blindspots:
    - For the less-specific blindspots, all $C$ sub-categories are eligible.
    - For the more-specific blindspots, we define eligibility equivalently to the general pool: *i.e.,* for each $C$ object categories belonging to the task super-category, a second object category (that does not belong to the task super-category) is a *eligible* if both the intersection and set difference of the first and second object categories have at least $100$ images in the train set. Concretely, the list of eligible blindspots consists of all eligible pairs of objects with both options of whether the blindspot is defined using the $\{$Presence, Absence$\}$ of the second object.
    - From the list of eligible blindspots, sample up to $20$ ECs (each with $1$ blindspot) without replacement.

Of the $60$ generated more-specific ECs, we retain $48$ ECs for analysis, dropping the $12$ ECs where we failed to induce the sampled blindspot. Like the other conditioned pool, there are only $22$ possible less-specific ECs. To measure the effect of blindspot specificity, we compare BDM performance on the $48$ more-specific to the $24$ less-specific ECs.

### D.6 Observations from a Qualitative Inspection of Real Data

In Appendix D.4, we used the 2D scvis embedding of the model's learned representation as an analysis tool to determine that these BDMs' tendency to merge true blindspots is a failing of the methods themselves. In this section, we do something similar and qualitatively inspect that 2D embedding for models trained on real image datasets in order make additional observations that may be relevant to the design of future BDMs.

**A low-dimensional embedding can work for real data.** In the next subsections, we demonstrate that this 2D embedding contains the information required to identify a wide range of blindspots from prior work and to identify novel blindspots. As a result, it seems that dimensionality-reduction is a potentially important aspect of BDM design or that an interactive approach based on this embedding could be effective.

**Getting a maximally general definition of a specific blindspot often requires merging multiple groups of images.** For example, reaching the conclusions presented in Figures D.8, D.15, D.26, and D.23 all required merging several distinct groups of images. As a result, it seems that being able to suggest candidate merges is a potentially important goal for BDM design.

**A small blindspot may appear as a a small cluster (Figure D.14) or as a set of outliers (Figure D.32).** The former is likely to cause problems for BDMs that use more standard supervised learning techniques (*e.g.,* those that use Linear Models or Decision Trees for their hypothesis class). The later is likely to cause problems for BDMs that model the data distribution (*e.g.,* those that use Gaussian Kernels for their hypothesis class). As a result, it seems that being able to identify blindspots of varying sizes and densities is a potentially important and challenging goal for BDM design.

**The boundaries of a blindspot can be "fuzzy" in the sense that there may be images that do not belong to the blindspot that are both close, in the image representation space, to images in the blindspot and that have higher error.** For example, looking at Figure D.12, we can see that the model's confidence decreases relatively smoothly as we approach the region that reflects "baseball gloves with people, but not in a baseball field." However, if we look at the region around this region (Figure D.13), we see that these images are almost all in baseball fields. As a result, it seems that being able to handle this fuzziness is a potentially important and challenging goal for BDM design.

### D.6.1 Preliminaries

Before presenting the details of our results, there are two questions to address.

**What blindspots do we study?** We consider blindspots discovered by prior works [40, 75, 93, 94, 97] as well as novel blindspots [3]. Note that these prior works:

- Are not all explicitly about blindspots (*e.g.,* some consider "spurious patterns" which imply the existence of blindspots).
- All consider a model that was trained with standard procedures (*e.g.,* no adversarial training) on a real dataset (*e.g.,* no semi-synthetic images or sub-sampling the dataset).
- All consider blindspots that pertain to a single class.

**How are our visualizations produced?** To start, we use scvis [30] to learn a 2D embedding of the model's representation of all of the training images of the class of interest. Because scvis combines the objective functions of tSNE and an AutoEncoder, we expect similar images to be near each other and dissimilar images to be far from each other in this 2D embedding. Then, we visualize that embedding and color code the points by the model's confidence in its predictions for the class of interest. As a result, blindspots should appear as groups of lighter colored points.

With this visualization, we start by first exploring what types of images were in different parts of the embedding. Then, given a specific type of image, we go back and highlight the part of the embedding that best represents that type of image. As a result, many of the figures we show are more like "summaries that support our conclusions" than "exhaustive descriptions of all the types of images we identified."

### D.6.2 Summit

In Figure 1 of their paper, Hohman et al. [40] show that their method reveals that the model is relying on “hands holding fish” to detect “tench.”

- Dataset: ImageNet
- Class: tench
- Blindspots: tench without people holding them

Results of inspecting the 2D embedding:

- The embedding also reveals this blindspot. However, our analysis suggests that this blindspot is too broadly defined because “people holding tench” (Figure D.3) has similar performance to “tench in a net” (Figure D.4) and that the performance drop comes from “tench underwater” (Figure D.5). We confirmed hypothesis by finding images on Google Images that match these descriptions and measuring the model’s accuracy on them.
- Additionally, the embedding reveals mislabeled images (Figure D.6).

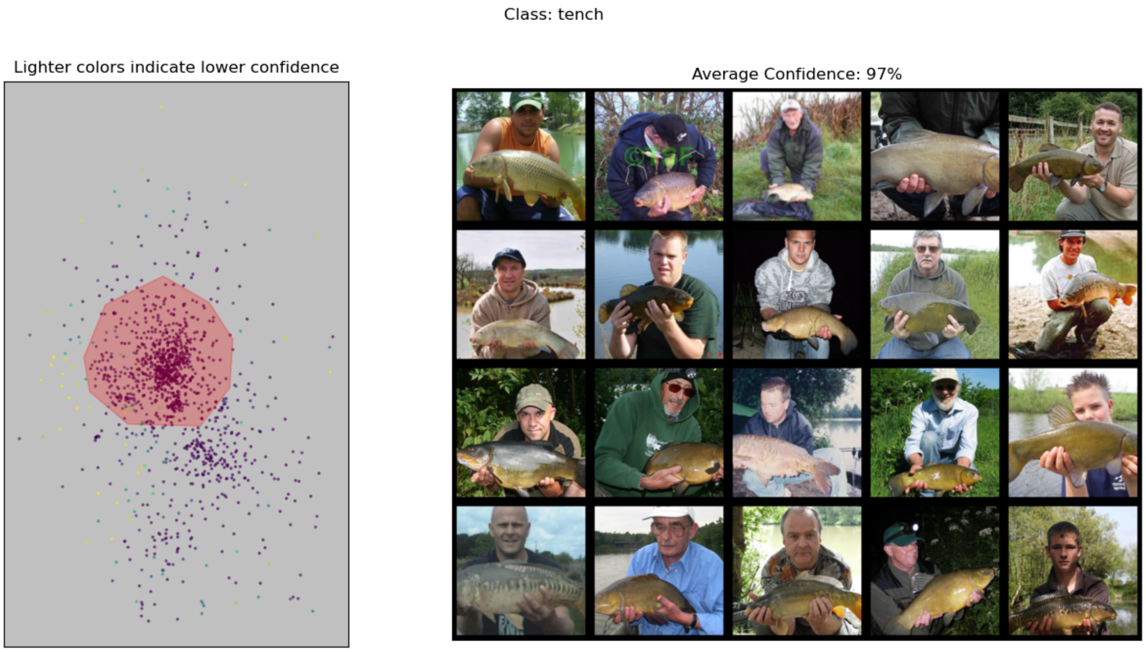


**Figure D.3:** “tench with people holding them.” Model accuracy on images from Google Images: 100% (40/40).

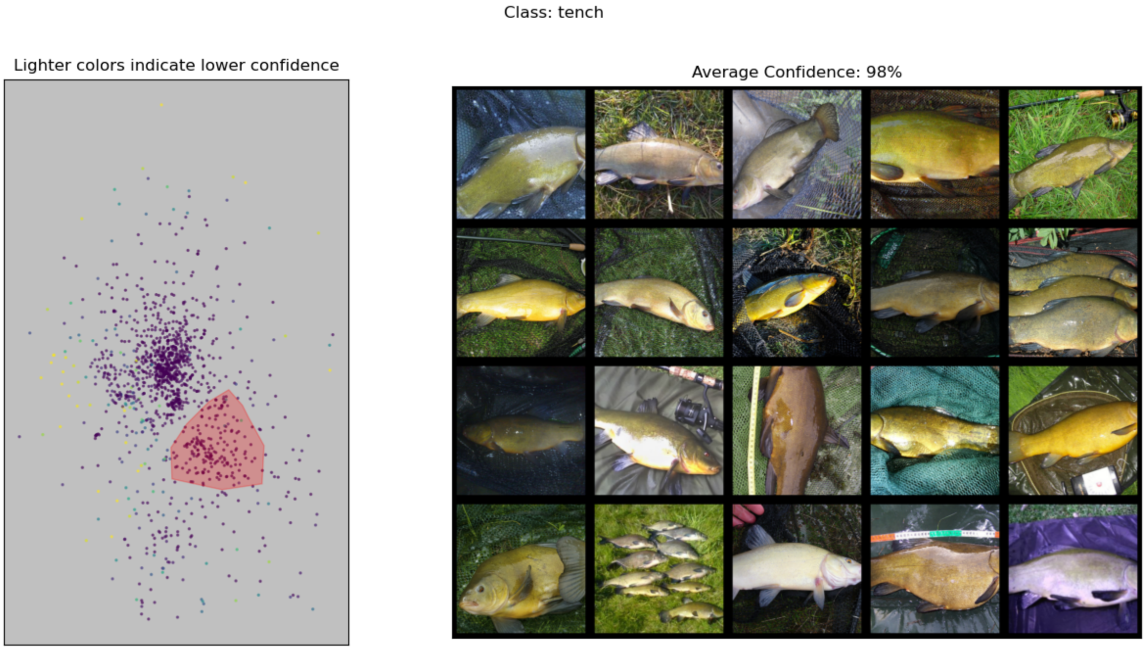


**Figure D.4:** “tench in a net.” Model accuracy on images from Google Images: 98% (39/40).

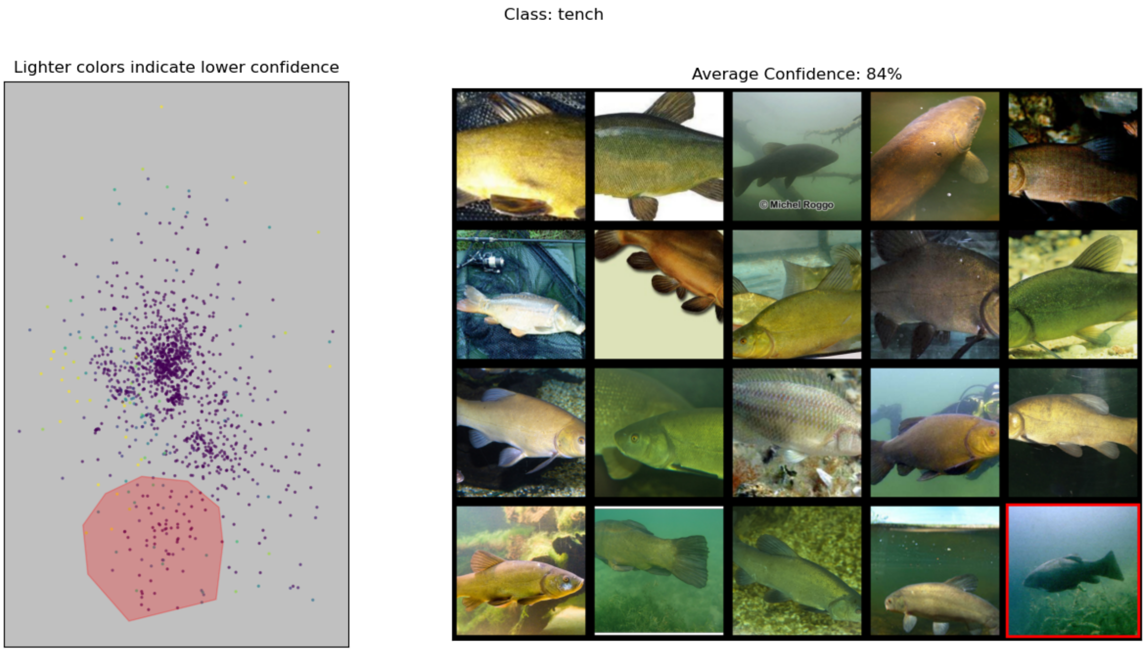


**Figure D.5:** “tench underwater.” Model accuracy on images from Google Images: 88% (35/40).

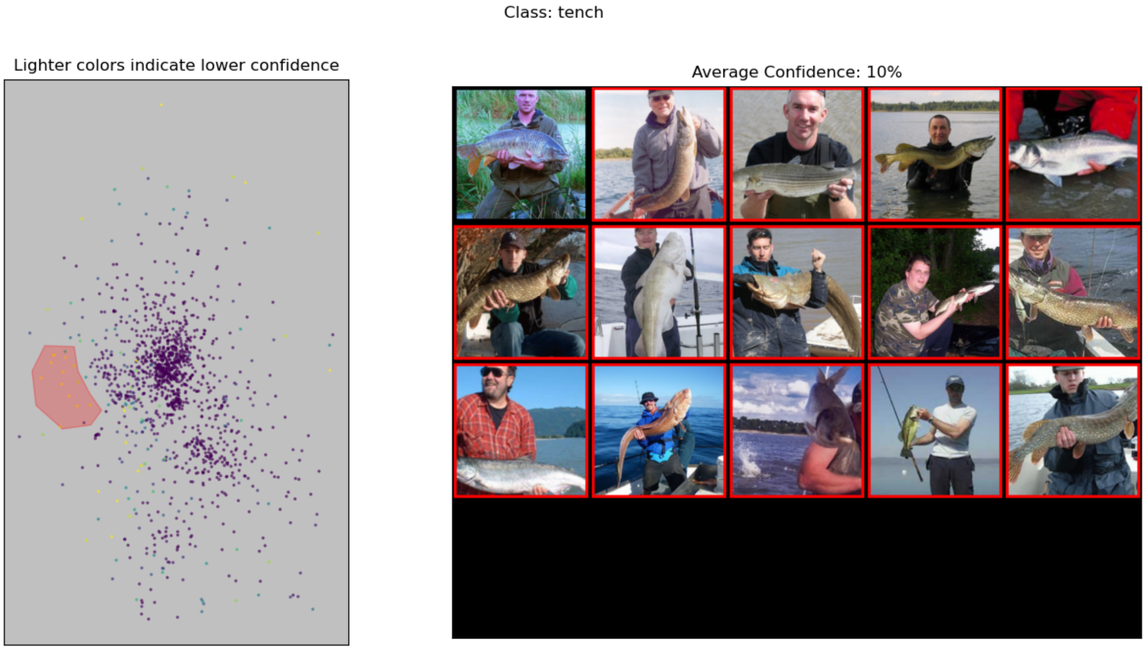


**Figure D.6:** Mislabeled images

### D.6.3 George

In Appendix C.1.4 of their paper, Sohoni et al. [97] discuss how their method is not effective at identifying one of the blindspots commonly studied in Group Distributionally Robust Optimization without using an external model's representation for the image representation.

- Dataset: CelebA
- Class: blond hair
- Blindspot: men with blond hair

Results of inspecting the 2D embedding:

- By comparing Figure D.7 to Figure D.8, we can see that the embedding clearly reveals this blindspot.
- Additionally, Figure D.9 shows that the embedding reveals an additional "blond women playing sports" blindspot.

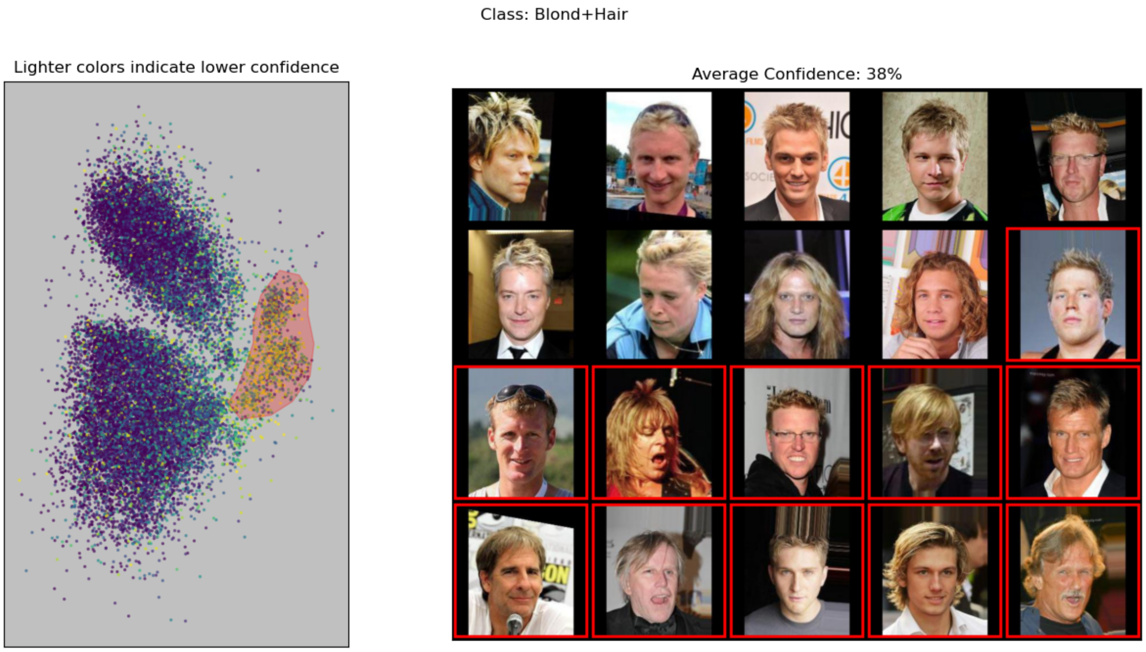


**Figure D.7:** “blond men”

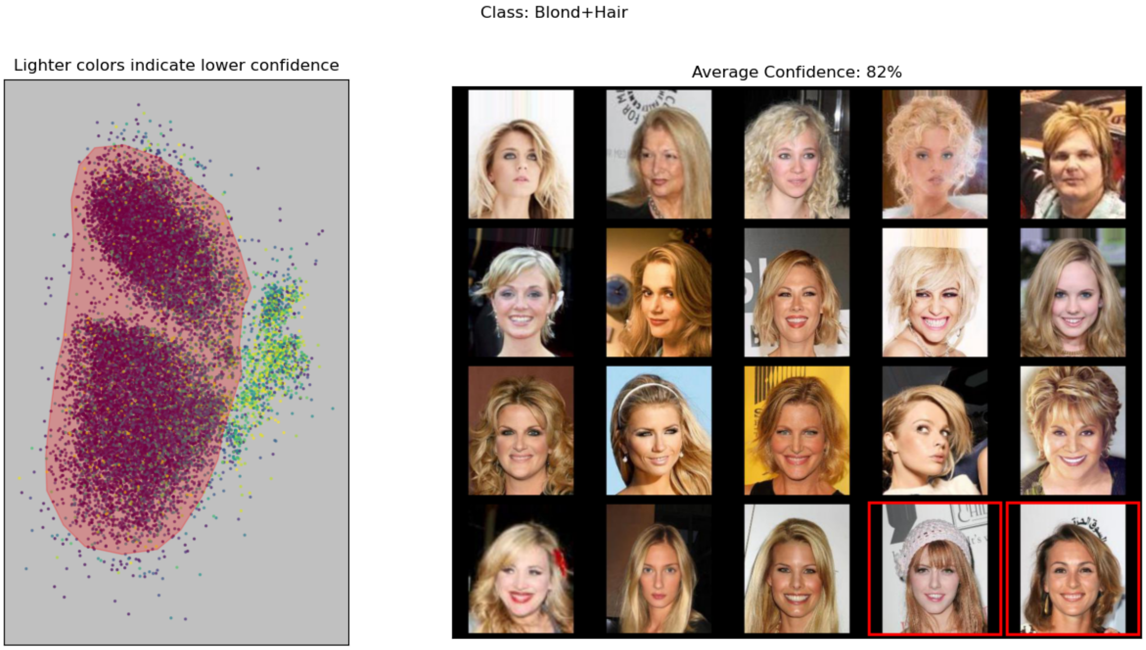


**Figure D.8:** “blond women”

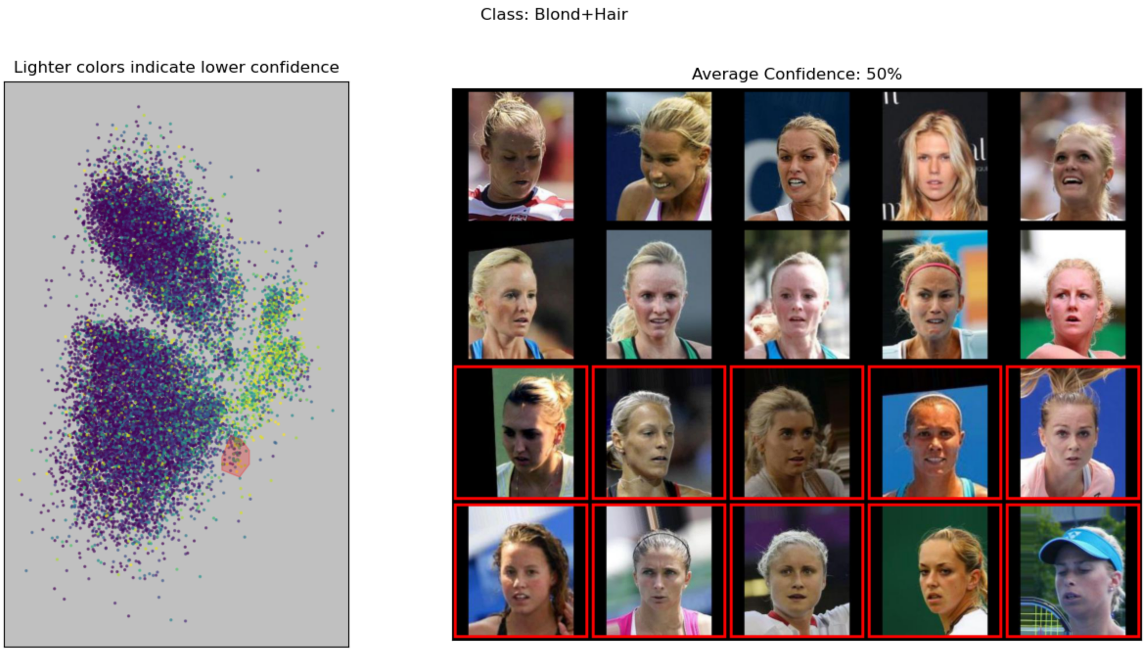


**Figure D.9:** “blond women playing sports”

### D.6.4 Feature Splitting

In Table 6 of their paper, Singh et al. [93] organize the spurious patterns that they identify by their own measure of bias. We consider both the most and least biased spurious patterns from that table.

**Most biased.**

- Dataset: COCO
- Class: baseball glove
- Blindspot: baseball gloves without people

Results of inspecting the 2D embedding:

- Comparing Figure D.10 to Figure D.11, we see that the embedding also reveals this blindspot.
- However, we can also see that the model's performance is not uniform across different types of images of "baseball gloves with people." In particular, the embedding reveals two other blindspots (Figures D.12 and D.14)

**Least biased.**

- Dataset: COCO
- Class: cup
- Blindspot: cups without dining tables

Results of inspecting the 2D embedding:

- Comparing Figure D.15 to Figure D.16, we see that the embedding also reveals this blindspot.
- However, we can also see that the model's performance is not uniform across different types of "cups without dining tables." Figures D.17, D.18, and D.19 show a few examples of the additional blindspots the embedding reveals.
- These results highlight an interesting subtlety. The cups in images with dining tables take up a greater portion of the image than the cups in images of sports stadiums, which raises the question: "Does this blindspot exist solely because the model is relying on 'context' that it shouldn't be? Or is the standard image pre-processing making the cups too small (in terms of their size in pixels) to detect?"

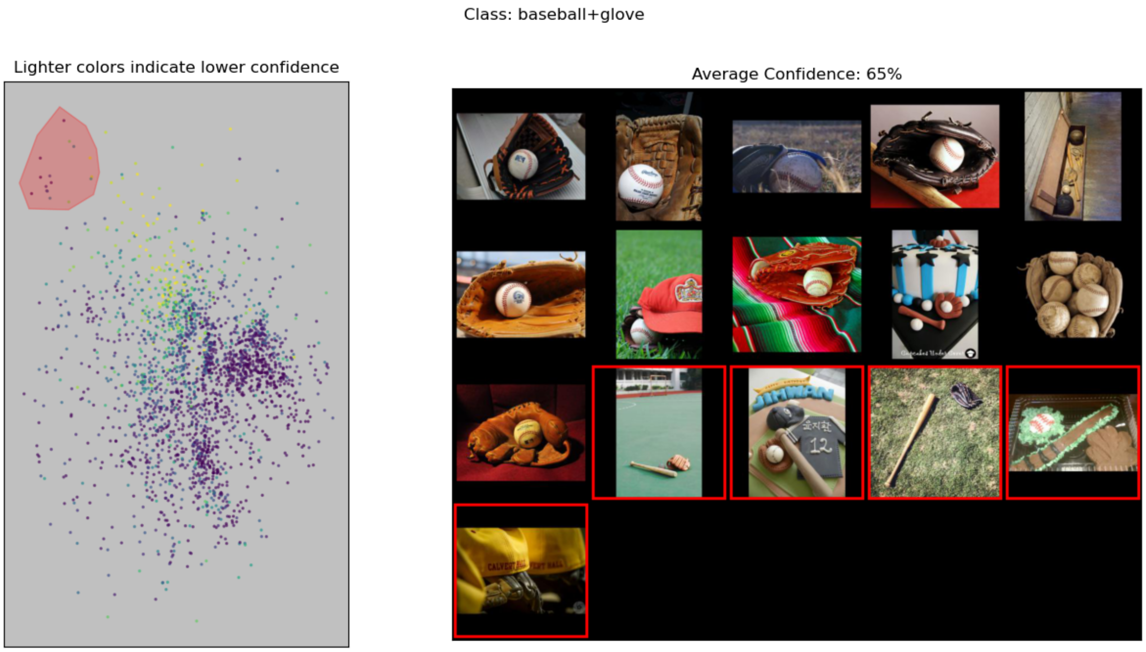


**Figure D.10:** “baseball gloves without people”

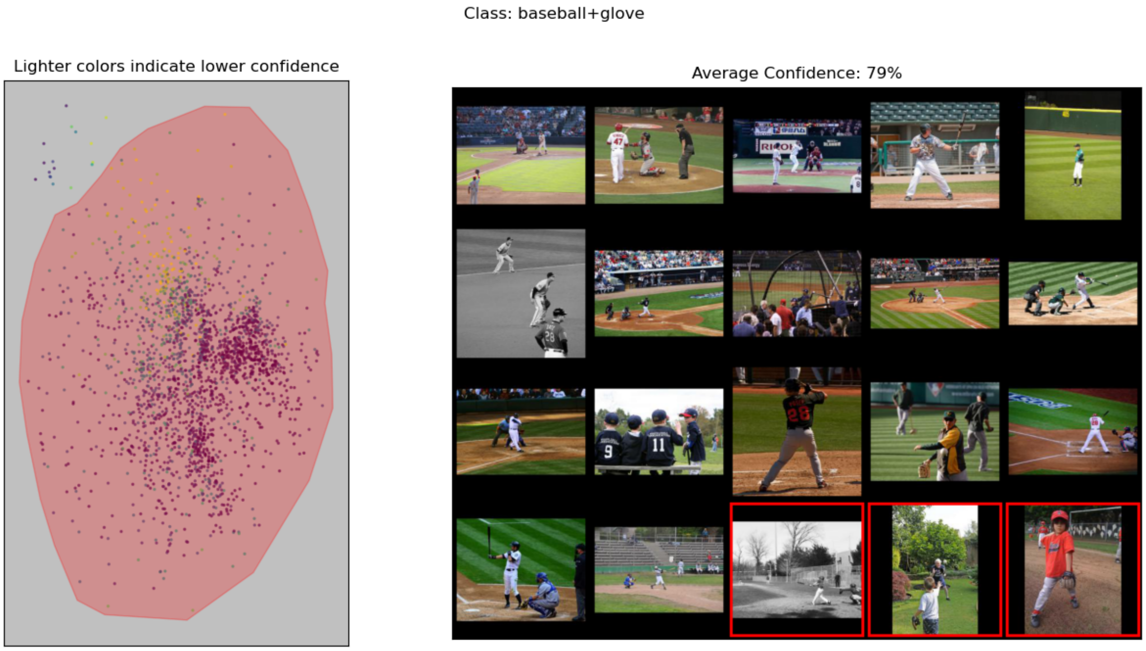


**Figure D.11:** “baseball gloves with people”

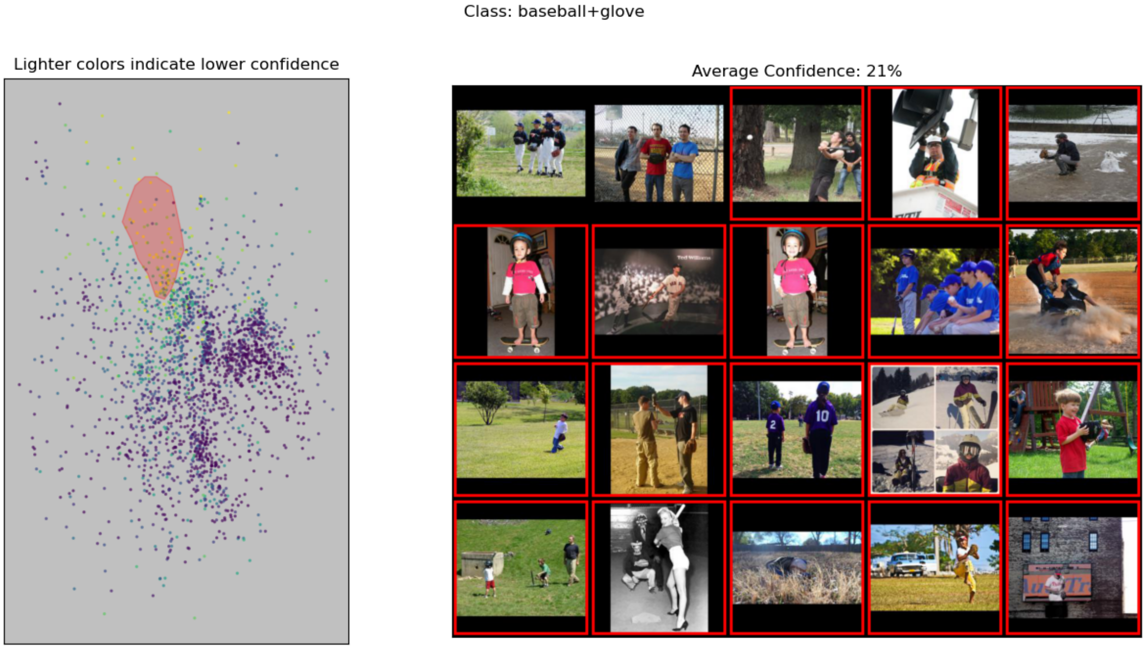


**Figure D.12:** “baseball gloves with people, but not in a baseball field”

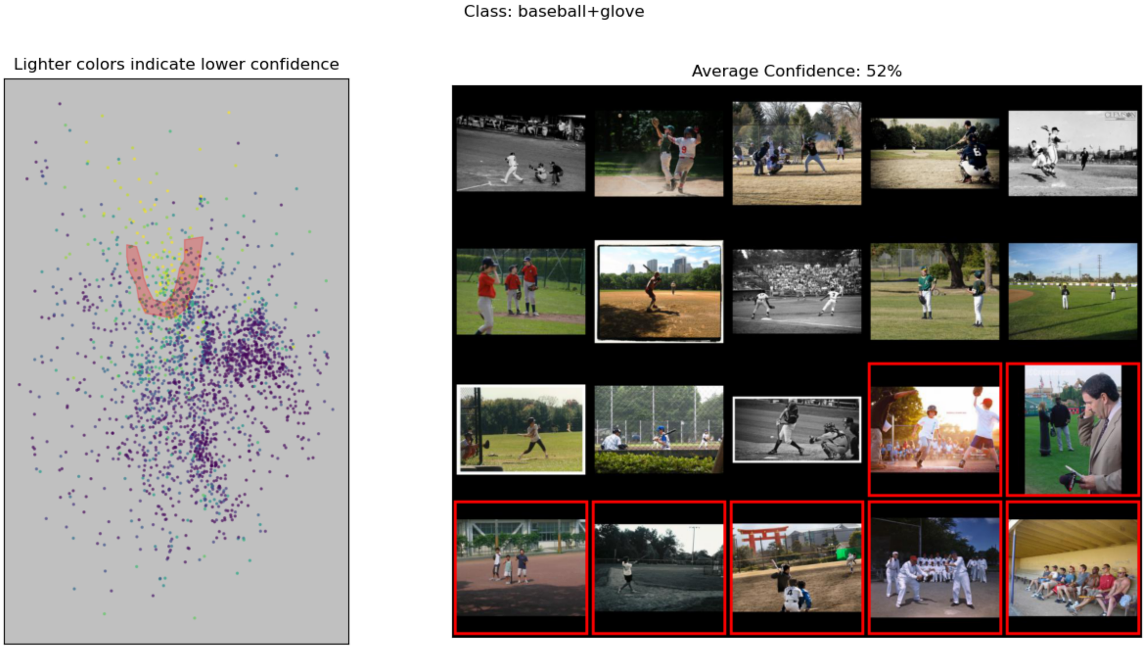


**Figure D.13:** If we look at the region around the “baseball gloves with people, but not in a baseball field” region (Figure D.12), we see that the model’s confidence is still lower despite the fact that these images still are in baseball fields.

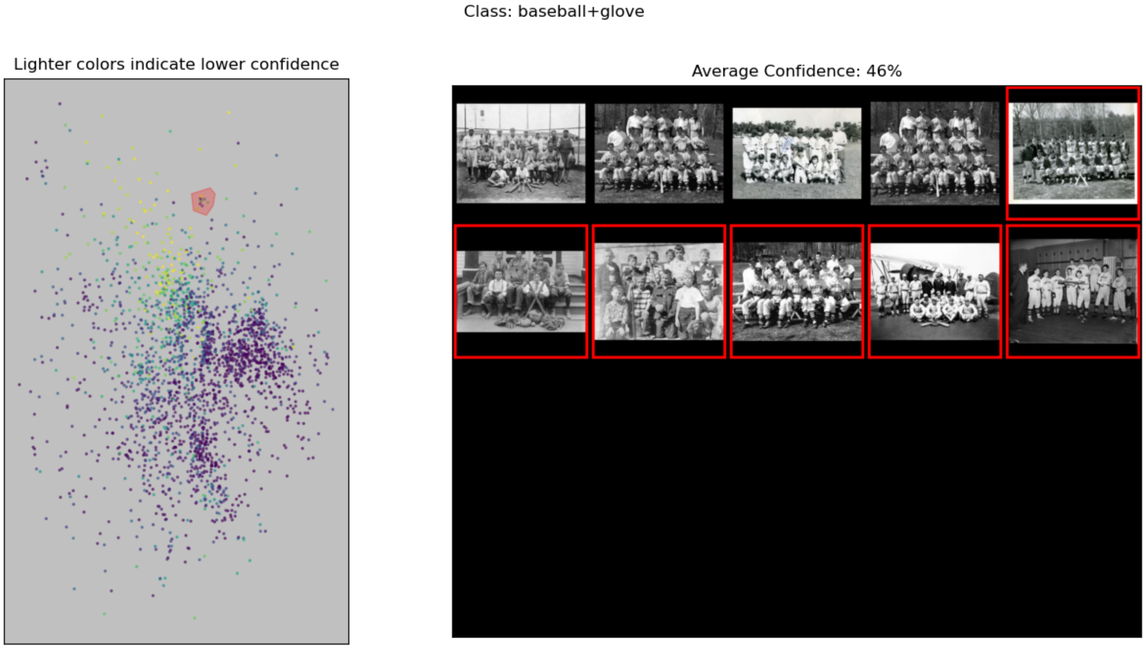


**Figure D.14:** “black and white images of baseball teams”

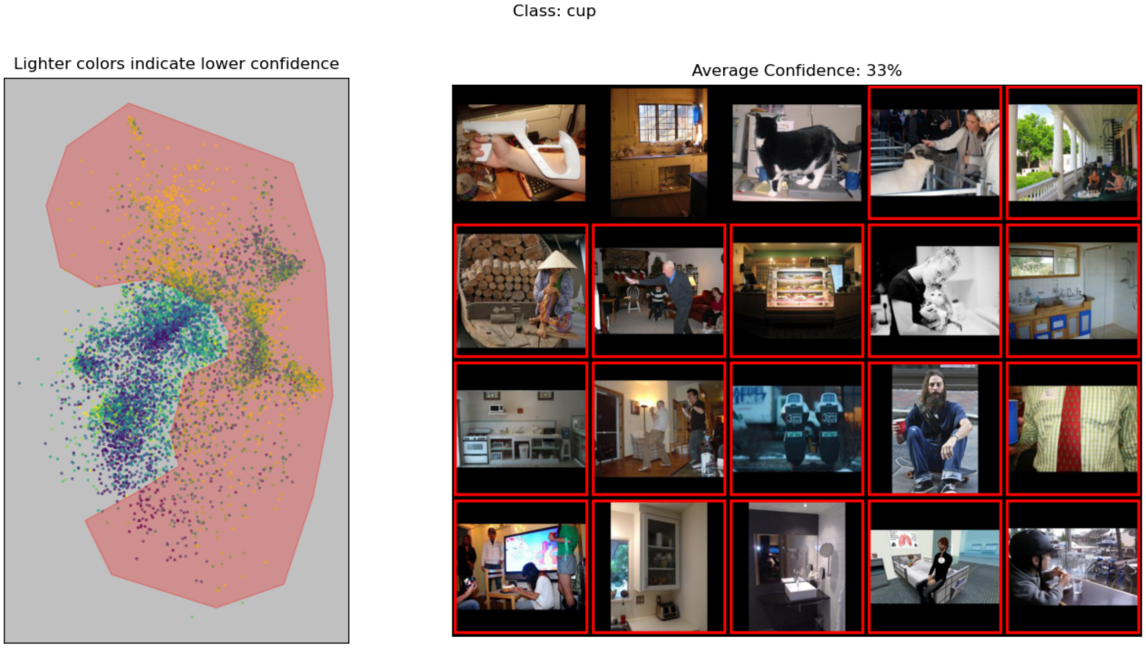


**Figure D.15:** “Cups without dining tables”

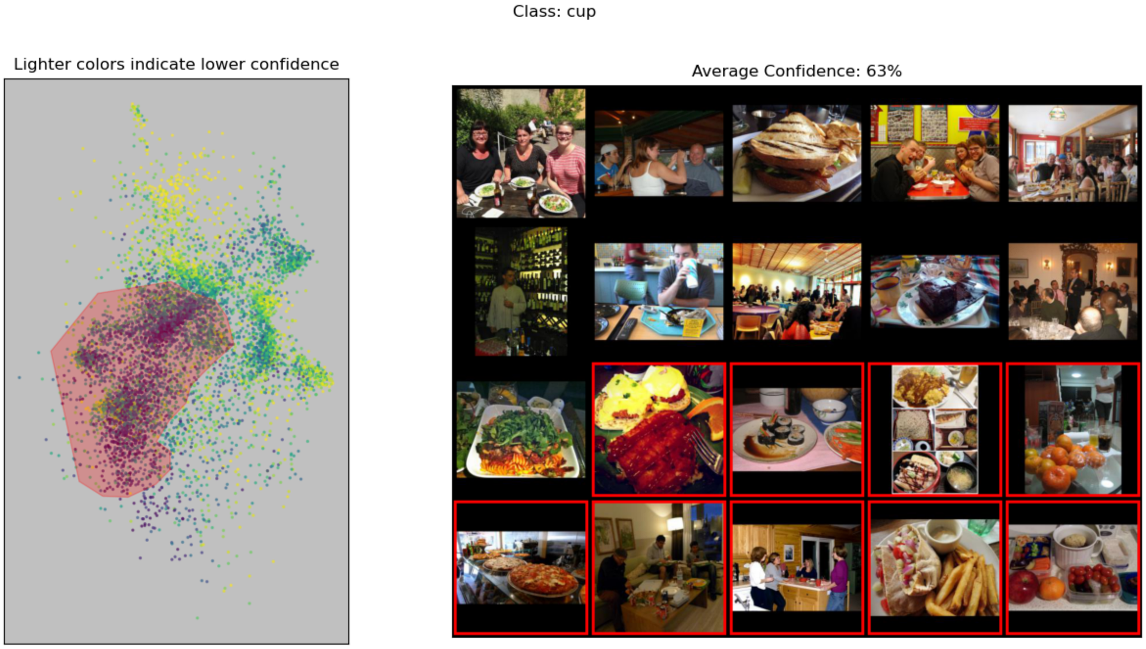


**Figure D.16:** “Cups with dining tables”

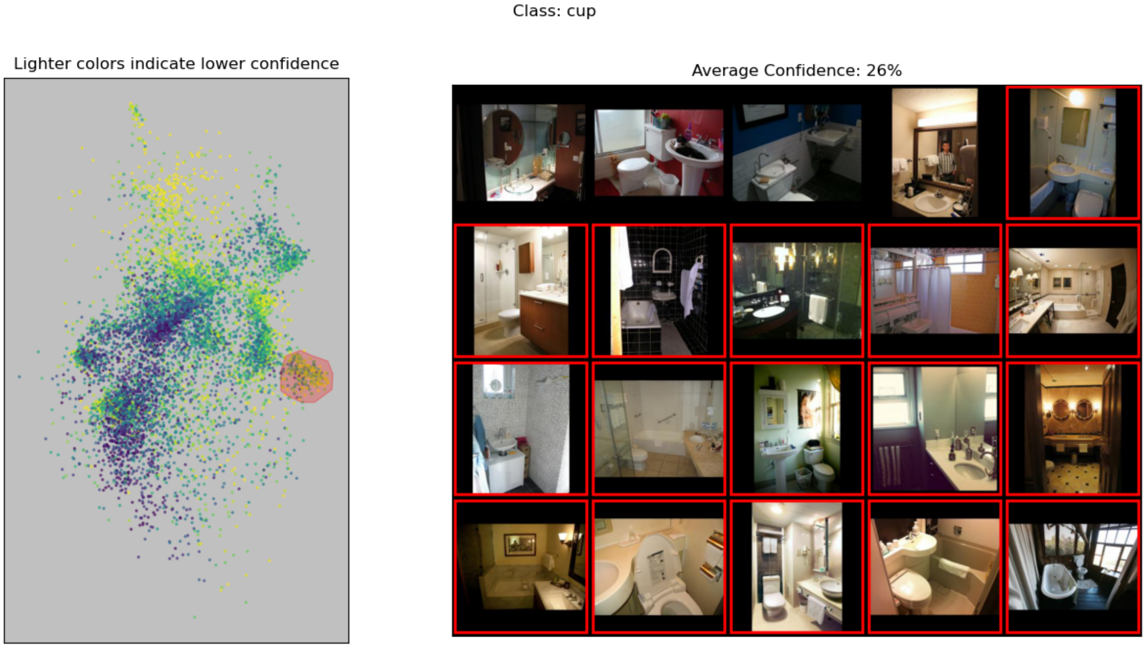


**Figure D.17:** “Cups in bathrooms”

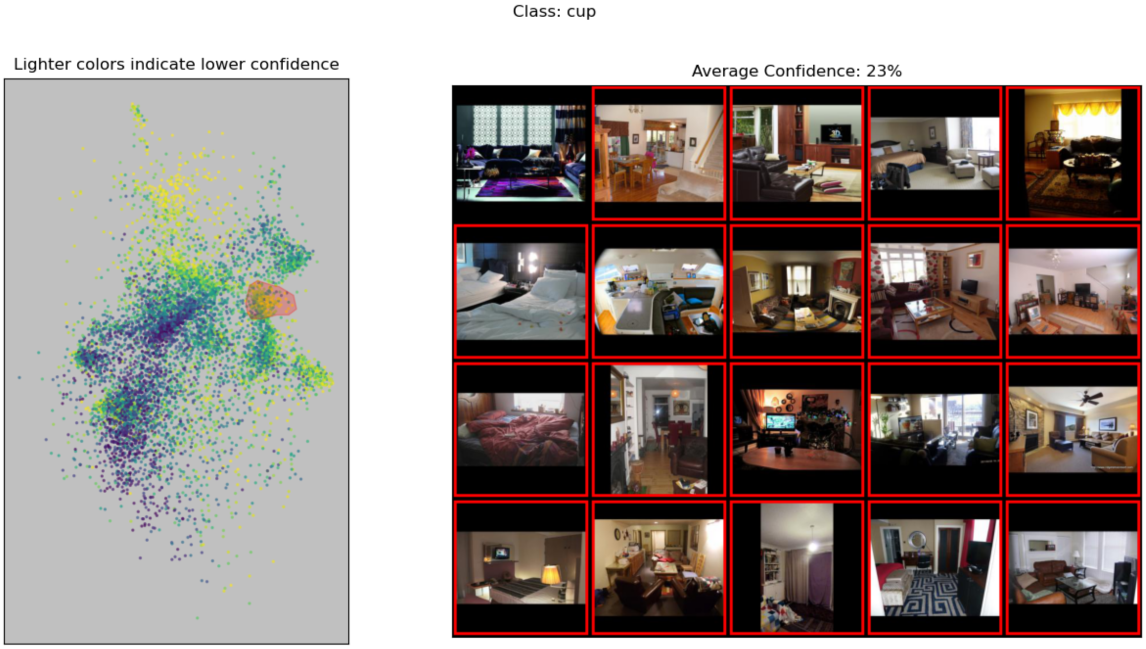


**Figure D.18:** "Cups in living rooms or bedrooms"

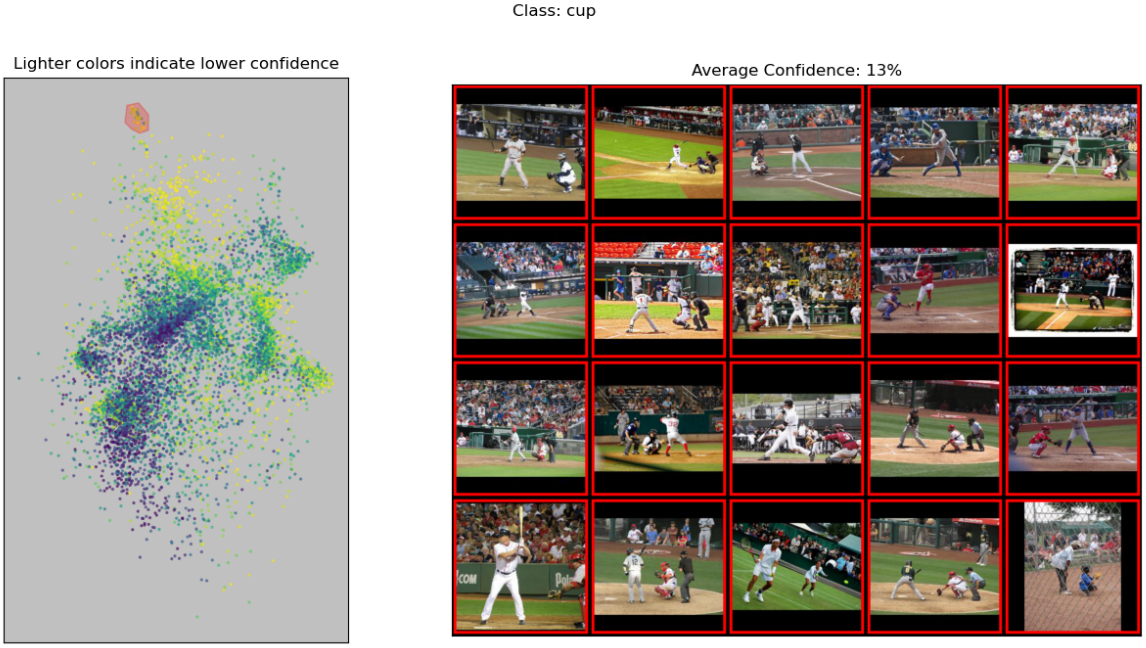


**Figure D.19:** "Cups in sports stadiums". Is the model using 'context' that it shouldn't be? Or is the image pre-processing making detecting these cups very hard by compressing them to only a few pixels? Or is this a limitation of the model architecture?

### D.6.5 Barlow

In Table 2 of their paper, Singla et al. [94] summarize some of the blindspots that they identify. We consider two of them. The first is “hog”, which is the class with the largest ALER-BER score (which they use as a measurement of how important a blindspot is). The second is “tiger cat”, which is the class with the largest increase in error rate for images in the blindspot.

**Largest ALER-BER Score.**

- Dataset: ImageNet
- Class: hog
- Blindspot: not “pinkish animals”

Results of inspecting the 2D embedding:

- By comparing Figure D.20 to Figure D.21, we can see that the embedding reveals this blindspot.

**Largest increase in error rate.**

- Dataset: ImageNet
- Class: tiger cat
- Blindspot: not “face close up”

Results of inspecting the 2D embedding:

- It does not appear that the embedding reveals this blindspot.
- By comparing Figure D.22 to Figures D.23 and D.24, we can see that the embedding reveals a “grey tiger cats inside” blindspot.
- Looking at Figure D.25, we can see that the embedding reveals mislabeled images.

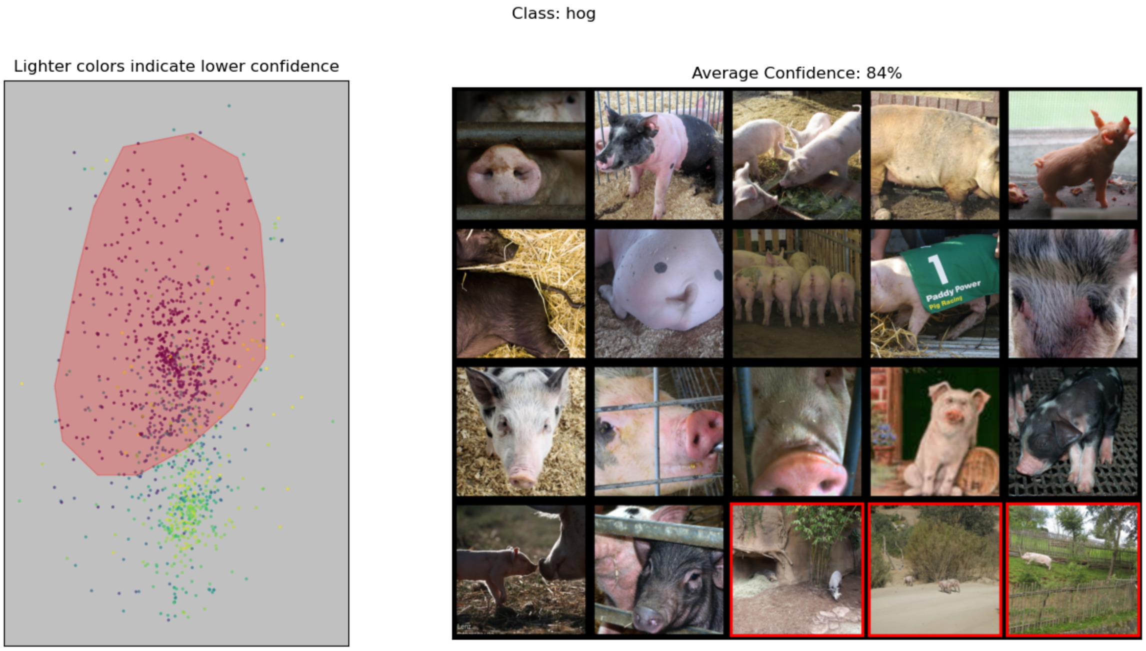


**Figure D.20:** “pink hogs”

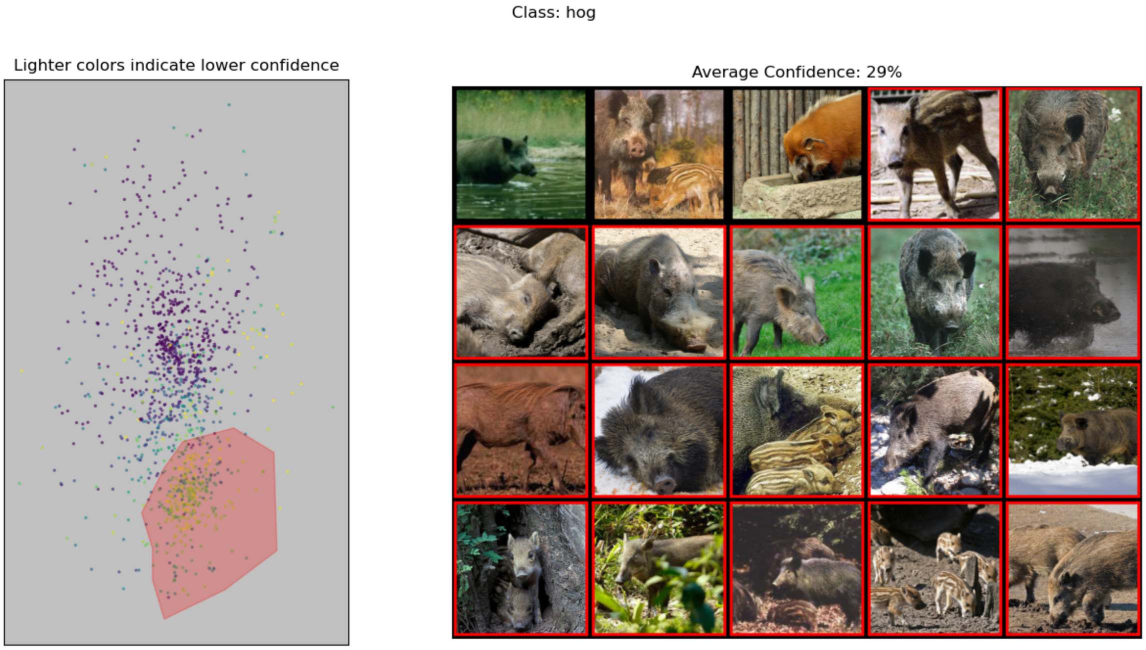


**Figure D.21:** “brown hogs”

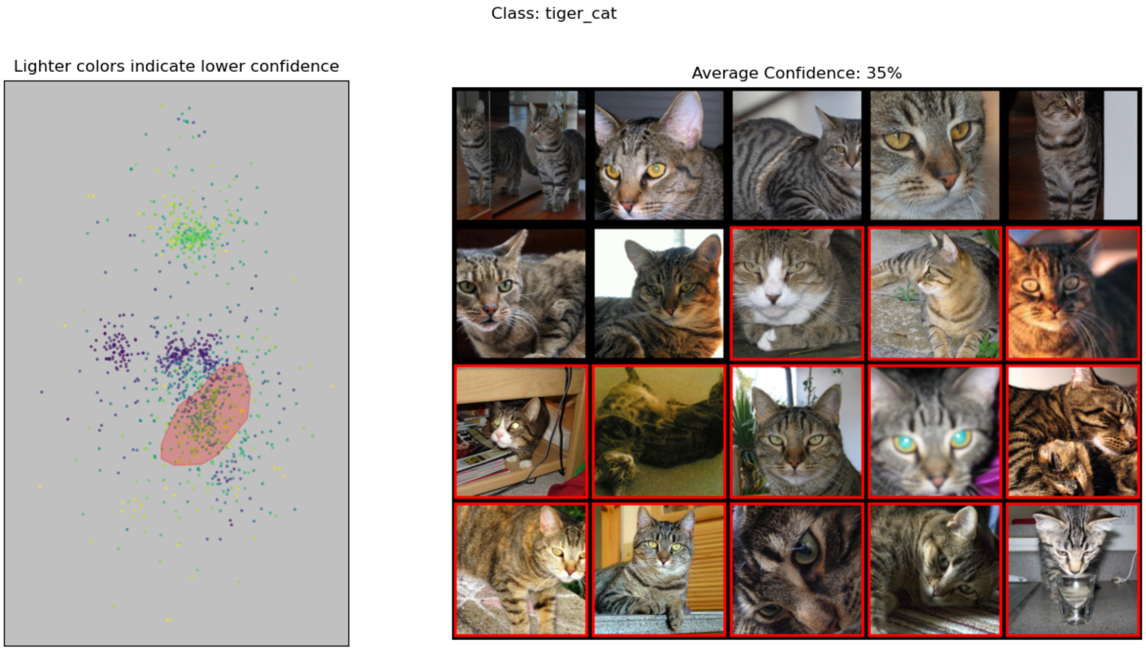


**Figure D.22:** "grey tiger cats inside"

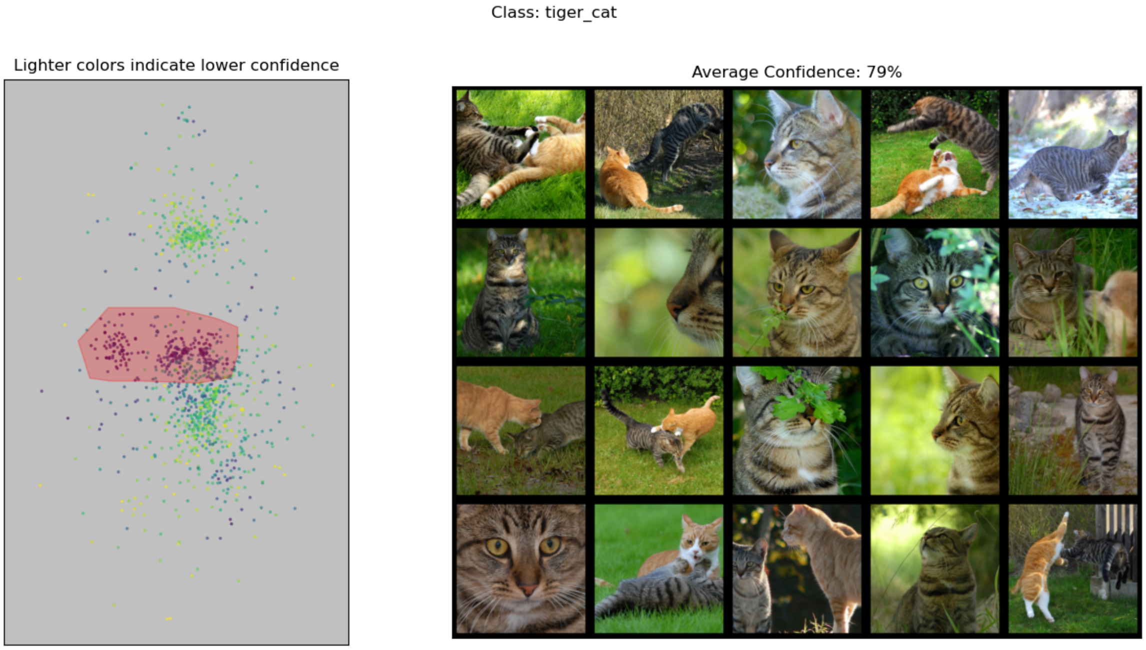


**Figure D.23:** "grey tiger cats outside"

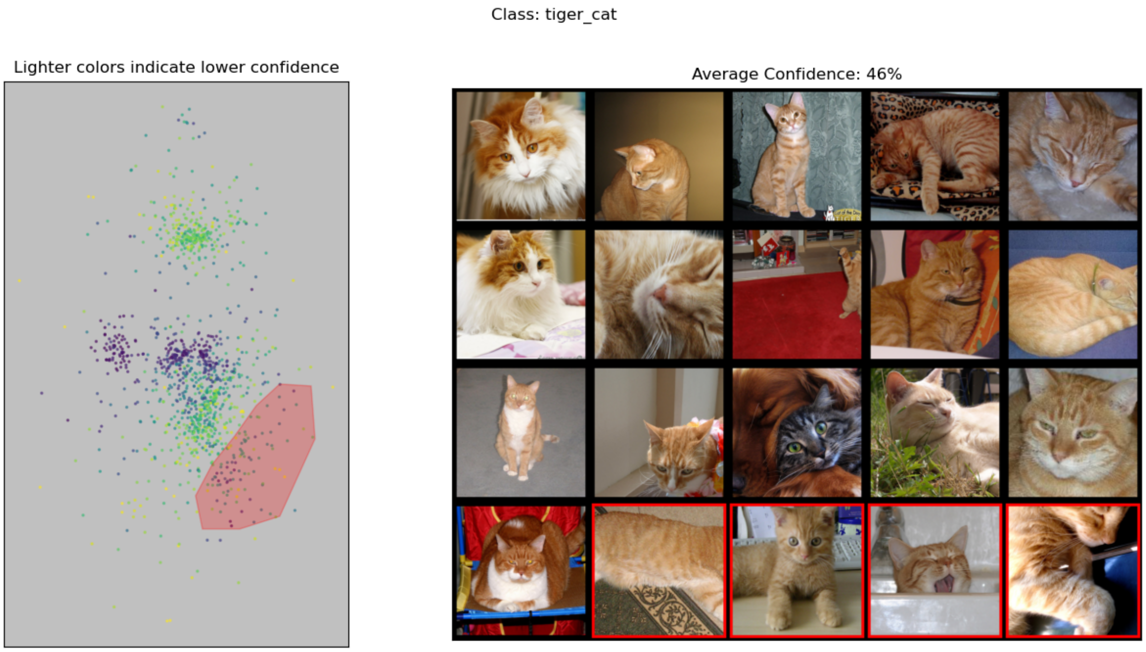


**Figure D.24:** “orange tiger cats”

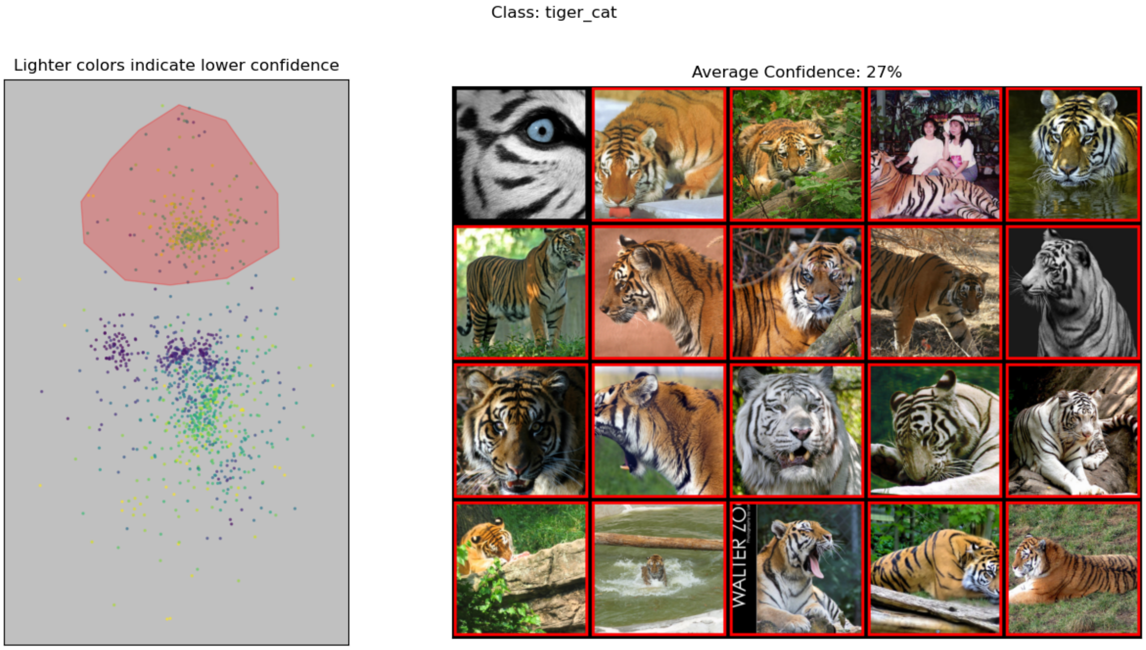


**Figure D.25:** Mislabeled Images

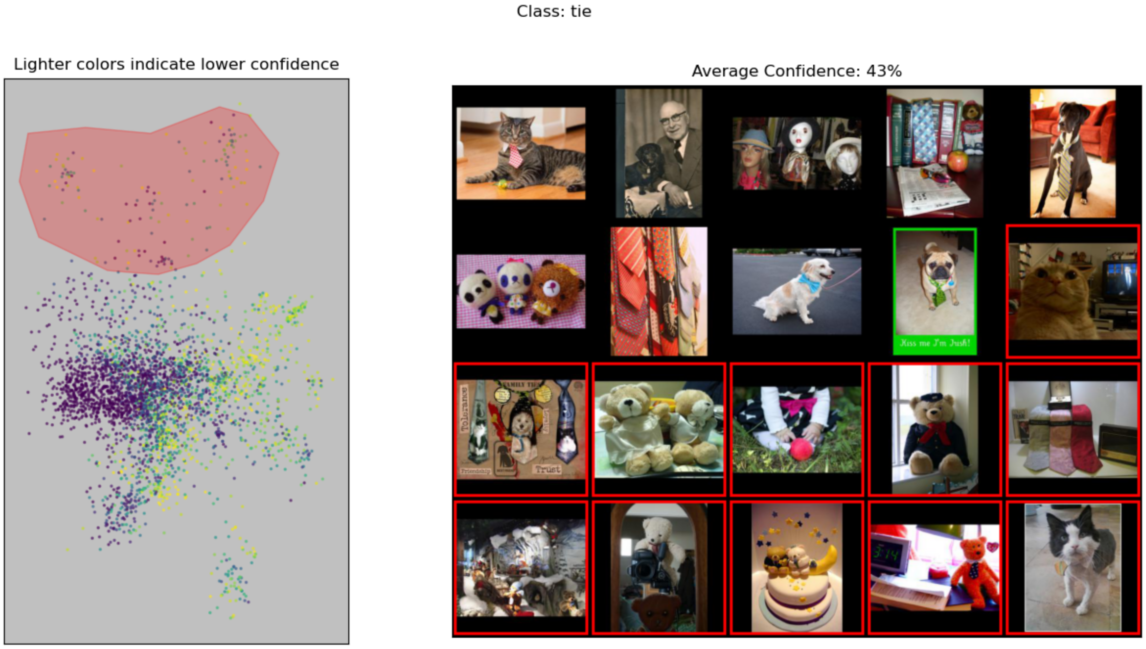
### D.6.6 Spire

Plumb et al. [75] claim that, compared to past works on identifying spurious patterns, that theirs is the first to identify negative spurious patterns. So we explore the example of a negative spurious pattern that they give. Additionally, they note that this class is part of two spurious patterns.

- Dataset: COCO
- Class: tie
- Blindspot: ties without people
- Blindspot: ties with cats

Results of inspecting the 2D embedding:

- Comparing Figure D.26 to Figure D.27, we see that the embedding reveals the "ties without people" blindspot.
- Comparing Figure D.28 to Figure D.29, we see that the embedding reveals the "ties with cats" blindspot. Note that this is a subset of the "ties without people" blindspot.
- Figures D.30 and D.31 show a few examples of the additional blindspots the embedding reveals.

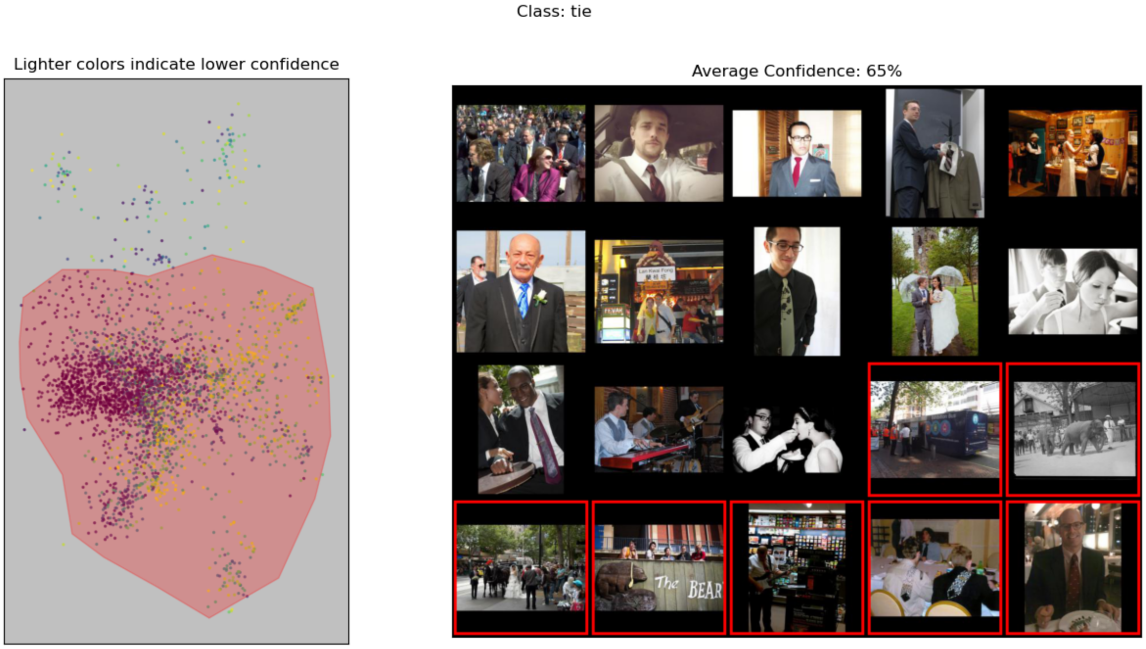


Class: tie

Lighter colors indicate lower confidence

Average Confidence: 43%

**Figure D.26:** “ties without people”



Class: tie

Lighter colors indicate lower confidence

Average Confidence: 65%

**Figure D.27:** “ties with people”

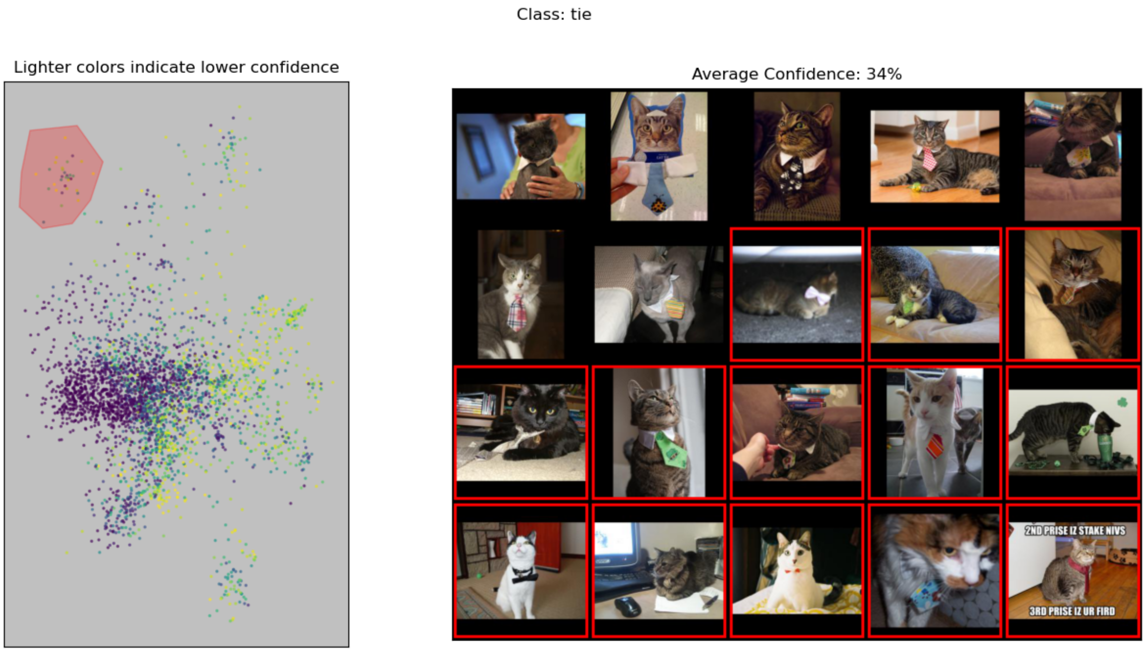


**Figure D.28:** “ties with cats”

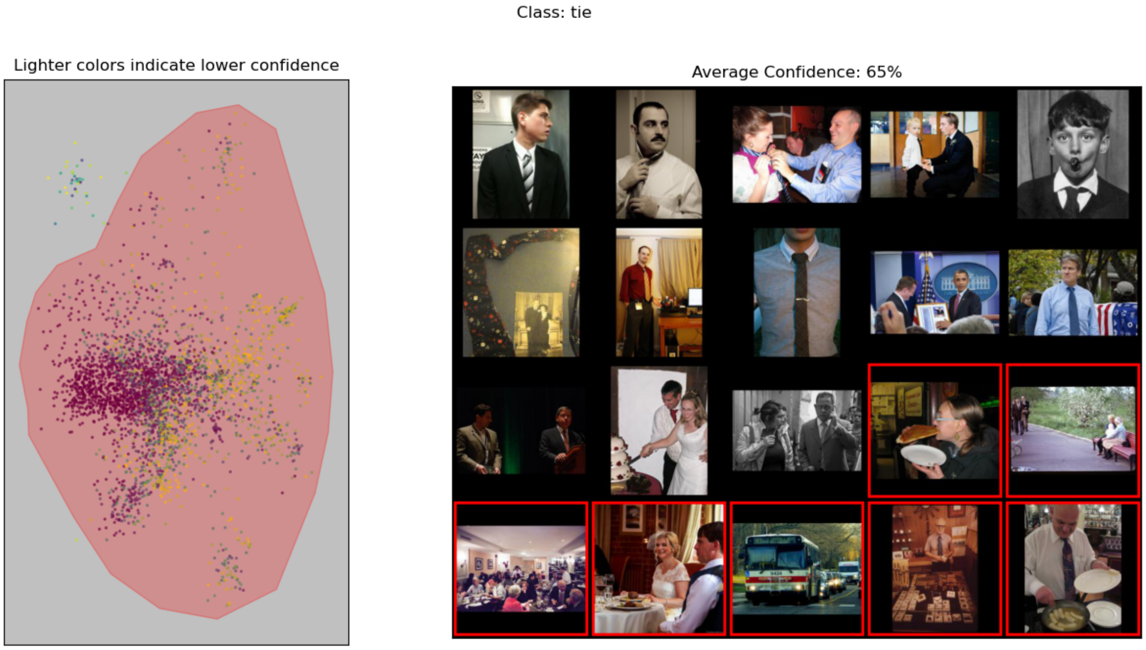


**Figure D.29:** “ties without cats”

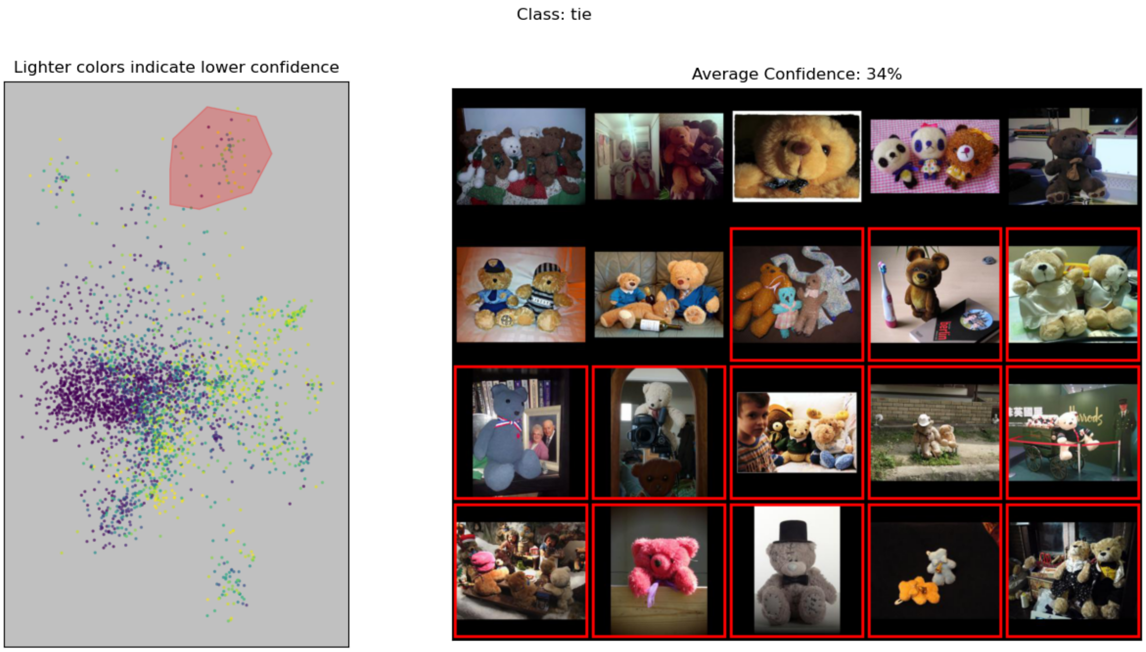


**Figure D.30:** “ties with teddybears”

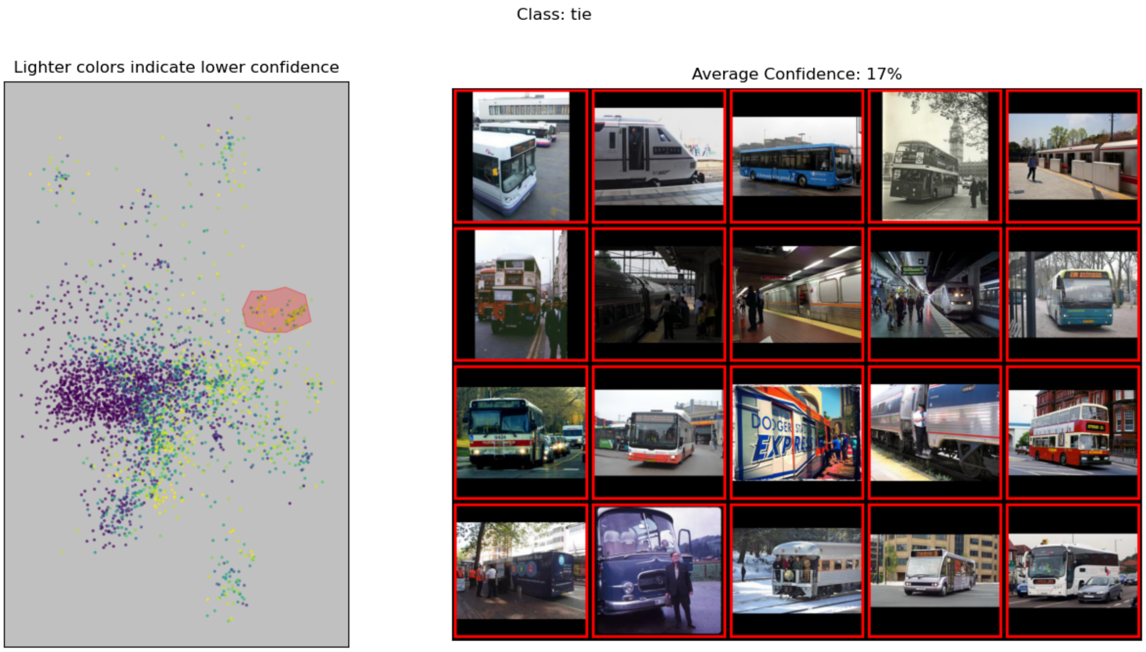


**Figure D.31:** “ties with buses” Is the model using ‘context’ that it shouldn’t be or is the image pre-processing making detecting these ties very hard (the bus driver is frequently wearing a tie, but that tie is very small)?

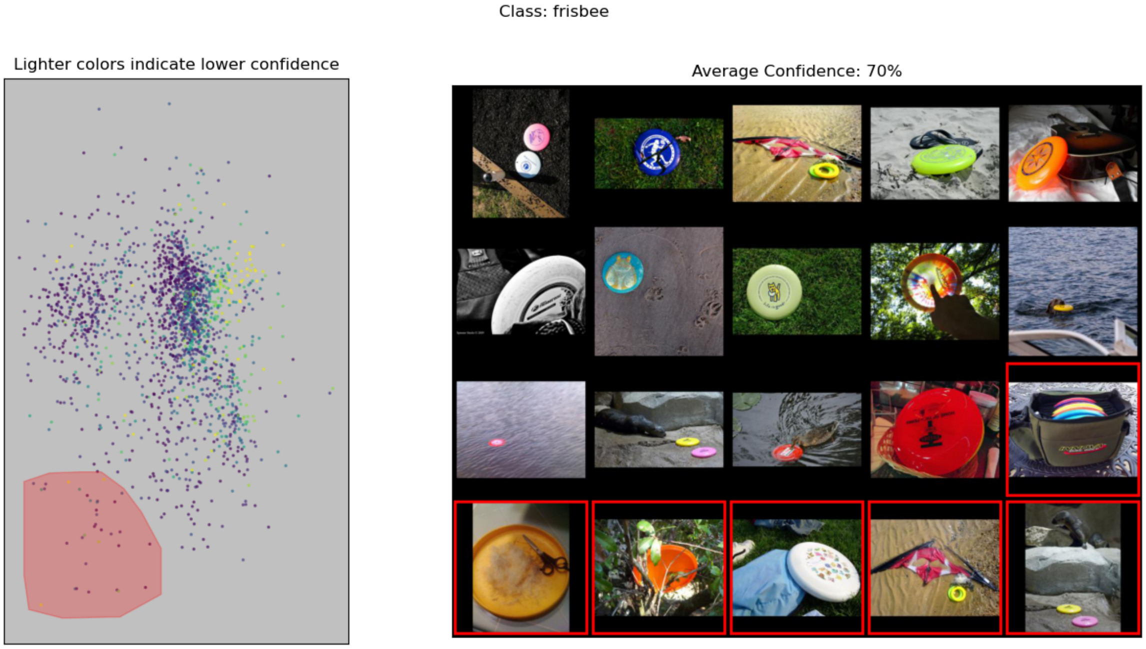


**Figure D.32:** Additionally, when we see that the embedding also reveals the “frisbees without dogs or people” blindspot from [75]. However, these images look more like outliers than a proper cluster.

### D.6.7 New Blindspots

Because it has been observed that it is easier to find "bugs" in models when we know what to look for [3], we also explore classes that have not been discussed in prior work. We asked a colleague to pick a class from the list of ImageNet classes and they chose "library" because they thought that it seemed related to or correlated with other ImageNet classes.

- Dataset: ImageNet
- Class: library
- Blindspot: images of the outside of libraries
- Blindspot: images of a single bookshelf

Results of inspecting the 2D embedding:

- Figure D.33 shows that the model is less able to label an image as a library when the image is of the outside of a library. Interestingly, this is even true when the building is labeled as a library.
- By comparing Figure D.34 to Figures D.35 and D.36, we see that, without the context clues of "people" or "multiple bookshelves" the model is less able to label an image as a library when the image contains only a single bookshelf. This may be a data labeling problem because "bookcase" is another class in ImageNet.

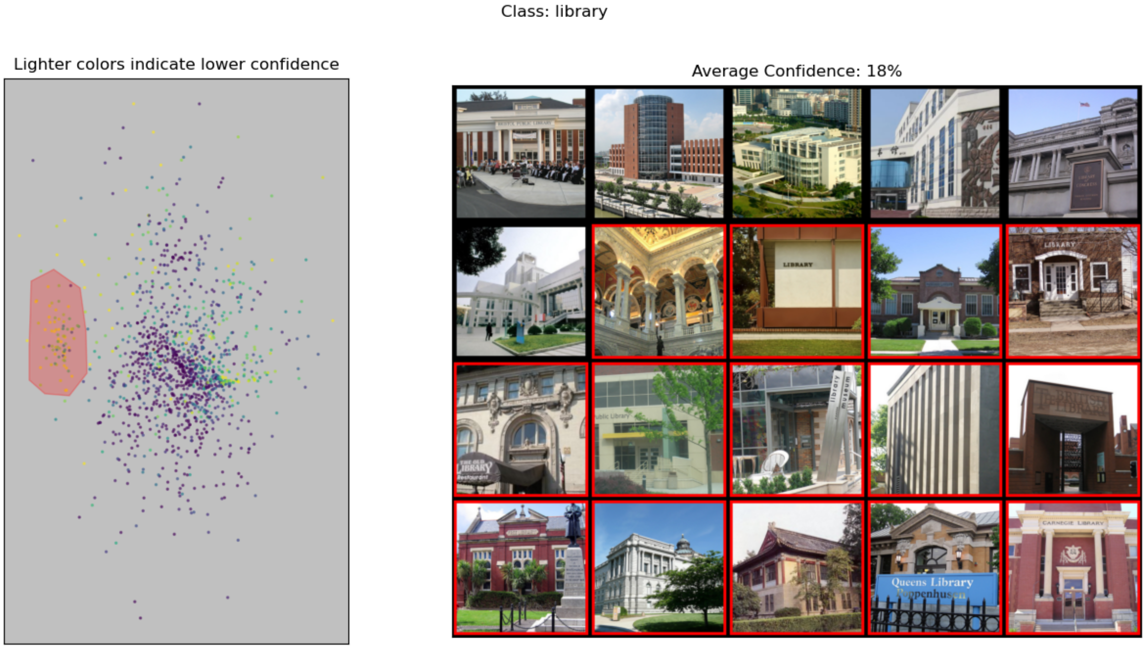


**Figure D.33:** "outside of libraries"

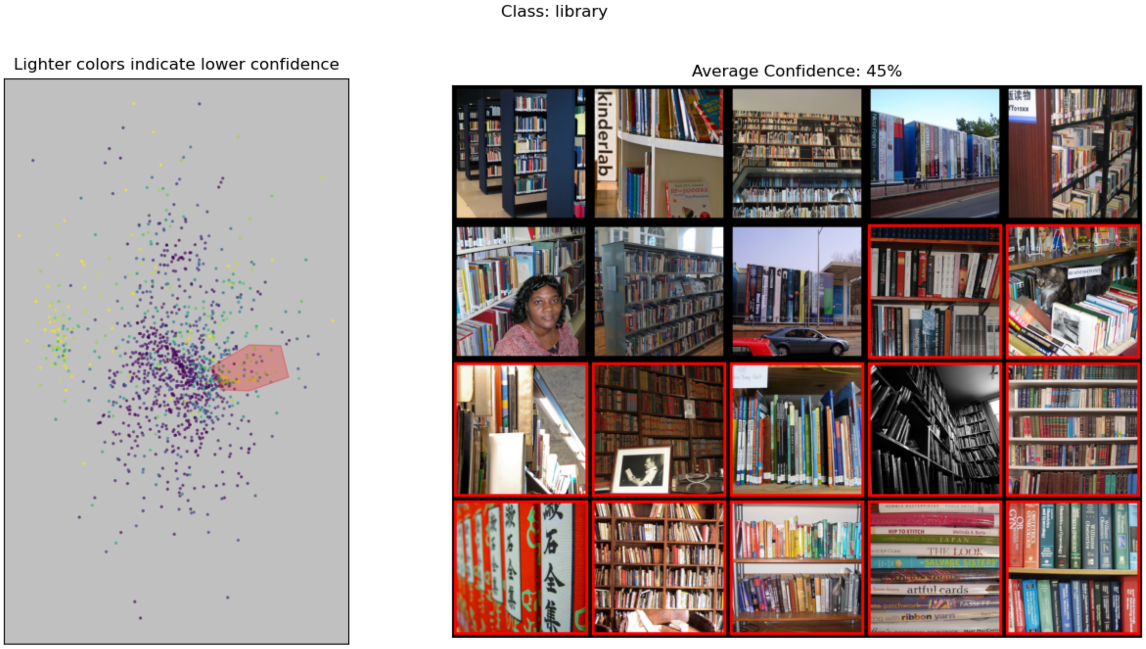


**Figure D.34:** "single bookshelf"

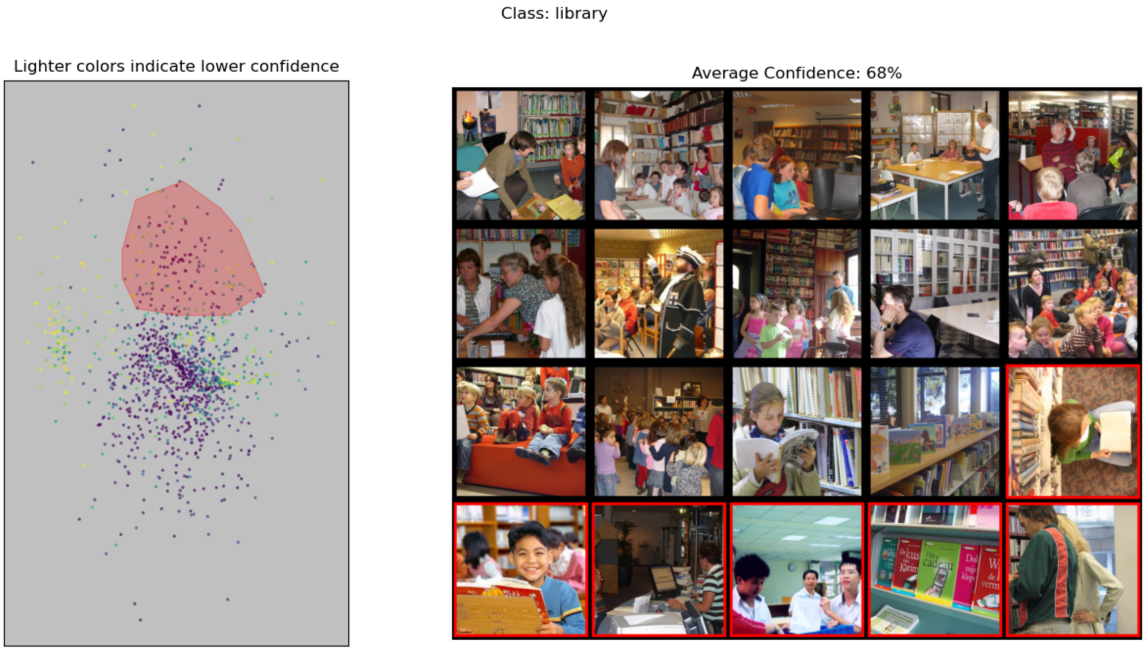


**Figure D.35:** “people near bookshelves’

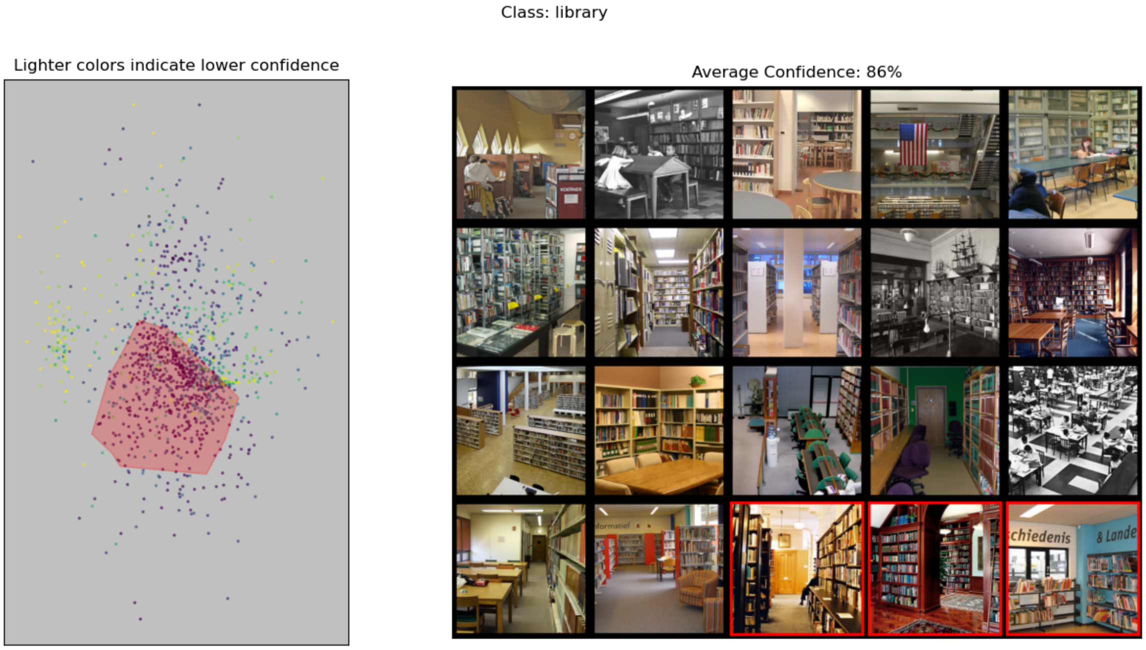


**Figure D.36:** “multiple bookshelves without people”