HARVARD UNIVERSITY

Graduate School of Arts and Sciences

DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

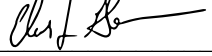
Harvard John A. Paulson School of Engineering and Applied Sciences
have examined a dissertation entitled:

"Right for the Right Reasons: Training Neural Networks to be Interpretable, Robust, and Consistent with Expert Knowledge"

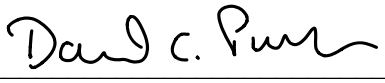
presented by: Andrew Slavin Ross

Signature 

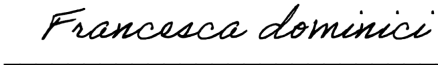
Typed name: Professor F. Doshi-Velez

Signature 

Typed name: Professor E. Glassman

Signature 

Typed name: Professor D. Parkes

Signature 

Typed name: Professor F. Dominici

May 3, 2021

Right for the Right Reasons: Training Neural Networks to be Interpretable, Robust, and Consistent with Expert Knowledge

A thesis presented

by

Andrew Slavin Ross

to

The School of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

May 2021

Thesis Advisor:
Finale Doshi-Velez

Author:
Andrew Slavin Ross

Right for the Right Reasons: Training Neural Networks to be Interpretable, Robust, and Consistent with Expert Knowledge

Abstract

Neural networks are among the most accurate machine learning methods in use today. However, their opacity and fragility to distribution shifts make them difficult to trust in critical applications. Recent efforts to develop explanations for neural networks have produced tools to shed light on the implicit rules behind predictions. These tools can help us identify when networks are right for the wrong reasons, or equivalently that they will fail under distribution shifts that should not affect predictions. However, such explanations are not always at the right level of abstraction, and more importantly, cannot correct the problems they reveal. In this thesis, we explore methods for training neural networks to make predictions for better reasons, both by incorporating explanations into the training process and by learning representations that better match human concepts. These methods produce models that are more interpretable to users and more robust to distribution shifts.

Contents

List of Tables

List of Figures

Acknowledgments

To start, I'd like to thank Finale. You've been such a good advisor, mentor, and friend. And although I'm proud of the academic successes we had, looking back, I'm actually happiest about when things were hardest, and how we made it through. You're a deeply good person, and one of the best parts about my PhD was coming to really trust you.

Next, I'd like to thank my other mentors and supporters at SEAS, including Elena Glassman for teaching me so much about HCI, Stratos Idreos for support and the best classes of my PhD, and Pavlos Protopapas for accepting me in the first place. I'd also like to thank both my CS and CSE cohorts (especially Taylor Killian, for introducing me to Finale and going above and beyond in his supportiveness to me and many others)!

The people closest to the heart of this dissertation, of course, are the members of the DtAK lab. Although I cared about ideas, it was you, it was the warmth and joy and weirdness and love of our community that convinced me to see my Masters and raise it a PhD. And when things got difficult, you saved me, time and time again. All the times you listened to me, fed me, let me crash on your couch, or took care of me when I got campylobacter from our chicken gizzard cookoffs; all of the meals, all of the ridiculously ornate lunch spreads we whipped together in and outside lab, all of the games, and the movies, and the exegeses of animes, and our mini sci-fi/fantasy library, the pain of trying to take MIT classes, monkey club, lunar new years, road trips through Mordor—these are memories I will treasure as long as the corresponding braincells yet live.

There is going to be a gigantic, gaping hole in these acknowledgments, because I haven't properly thanked my friends, my partner, and my family. I wish that I could do that here in this venue, but the constraints of space and privacy, which I feel keenly, don't permit it. But know that you have contributed so much to this document and the happiness of its author. If not for you, and the experiences we shared, this dissertation would be completely different, or not exist at all—a fact which almost doesn't seem worth mentioning, because I treasure the time I've spent with you far more than any paper published. I love you all so much, and I'm so grateful and honored to have shared this period of my life with you.

Part I

Introduction

Chapter 1

Introduction

The motivation for this dissertation is easiest to express with a story:

> A father decides to teach his young son what a sports car is. Finding it difficult to explain in words, he decides to give some examples. They stand on a motorway bridge and as each car passes underneath, the father cries out "that's a sports car!" when a sports car passes by. After ten minutes, the father asks his son if he's understood what a sports car is. The son says, "sure, it's easy". An old red VW Beetle passes by, and the son shouts – "that's a sports car!". Dejected, the father asks – "why do you say that?". "Because all sports cars are red!", replies the son. [21]

There is another popular version that pokes fun at the Department of Defense:

> In the early days of the perceptron the army decided to train an artificial neural network to recognize tanks partly hidden behind trees in the woods. They took a number of pictures of a woods without tanks, and then pictures of the same woods with tanks clearly sticking out from behind trees. They then trained a net to discriminate the two classes of pictures. The results were impressive, and the army was even more impressed when it turned out that the net could generalize its knowledge to pictures from each set that had not been used in training the net. Just to make sure that the net had indeed learned to recognize partially hidden tanks, however, the researchers took some more pictures in the same woods and showed them to the trained net. They were shocked and depressed to find that with the new pictures the net totally failed to discriminate between pictures of trees with partially concealed tanks behind them and just plain trees. The mystery was finally solved when someone noticed that the training pictures of the woods without tanks were taken on a cloudy day, whereas those with tanks

> were taken on a sunny day. The net had learned to recognize and generalize the difference between a woods with and without shadows! [61]

The first story is a parable and the second is apocryphal[1], but both illustrate an inherent limitation in learning by example, which is how we currently train machine learning systems: we only provide them with inputs (questions, x) and outputs (answers, y). When we train people to perform tasks, however, we usually provide them with *explanations*, since without them many problems are ambiguous. In machine learning, model developers usually circumvent such ambiguities via regularization, inductive biases (e.g. using convolutional neural networks for translational invariance), or simply acquiring vast quantities of data (such that the problem eventually becomes unambiguous, as if the child had seen every car in the world). But there is still a risk our models will be right for the wrong reasons—which means that if conditions change, they will simply be wrong.

As we begin to apply machine learning to sensitive domains such as healthcare, this risk has highlighted the need for *interpretable* models, as this final story illustrates:

> Although models based on rules were not as accurate as the neural net models, they were intelligible, i.e., interpretable by humans. On one of the pneumonia datasets, the rule-based system learned the rule "HasAsthma(x) $\rightarrow$ LowerRisk(x)", i.e., that patients with pneumonia who have a history of asthma have lower risk of dying from pneumonia than the general population. Needless to say, this rule is counterintuitive. But it reflected a true pattern in the training data: patients with a history of asthma who presented with pneumonia usually were admitted not only to the hospital but directly to the ICU (Intensive Care Unit). The good news is that the aggressive care received by asthmatic pneumonia patients was so effective that it lowered their risk of dying from pneumonia compared to the general population. The bad news is that because the prognosis for these patients is better than average, models trained on the data incorrectly learn that asthma lowers risk, when in fact asthmatics have much higher risk (if not hospitalized). [39]

In this case, a pneumonia risk prediction model learned an unhelpful rule because its training outcomes were imperfect proxies for true medical risk. Had the model been put into production, it would have endangered lives. The fact that they used an "interpretable"

[1] https://www.gwern.net/Tanks

model architecture, in this case, helped them realize and avoid this danger (by not using the neural network at all).

But, we should be troubled in at least two ways. First, what if the confounding variable (asthma) was not one of the features available to the machine learning model, but instead, there were a host of other variables that were weakly correlated with asthma? In practice, such a model might treat these weakly correlated values as slightly less predictive of mortality, and thus continue to underestimate asthmatic patients' risk. But the effect would be hidden, interpretable architecture or not. How can we detect and avoid such problems?

Second, though the dataset has confounds, it likely still contains information from which a human analyst could extract useful clinical conclusions. Is there some way for machine learning models to utilize this data, despite its flaws?

This thesis seeks to provide concrete methods for addressing these types of problems in a number of specific cases, as well as a roadmap for how to solve them in general. Below we provide an outline.

Part II (Input Gradient Penalties): In Part II, we consider a number of methods for training neural networks with input gradient penalties, which can be interpreted as a form of *explanation regularization*—i.e. optimizing a model to make predictions for specific reasons:

- Chapter 3, based on Ross et al. [2017], will explore how we can use input gradient penalties to explicitly encode domain knowledge into any differentiable model's training objectives. We will use these penalties to precisely control how models generalize to test data from different distributions, which would otherwise be unobtainable by normal optimization.

- Chapter 4, based on Ross et al. [2018] and Ross et al. [2020], will explore how input gradient penalties can be applied within the context of ensembling to learn quantitatively and qualitatively diverse models that all perform well on the training set, but extrapolate away from it in maximally different respects. We show how this method can be useful for interpreting the dataset, quantifying its ambiguity, and achieving

robustness to distribution shifts.

- Chapter 5, based on Ross and Doshi-Velez [2018], explores how input gradient regularization can simultaneously improve robustness to adversarial examples (the worst-case distribution shift) and interpretability (as demonstrated by a user study).

Part III (Interpretable Representations): Input gradients have a weakness that we alluded to in our discussion of hidden confounders in the pneumonia example: sometimes the "right reasons" are not easy to express in terms of input features. Instead, we might wish to first learn a *representation* of the data in which our knowledge is easier to express. In Part III, we make important progress towards learning interpretable (and correct) representations:

- Chapter 7, based on Ross et al. [2021], explores how we can design interactive user studies to measure whether users understand representations—as having an accurate method of *measuring* the interpretability of representations is critical for evaluating the effectiveness of any method of learning them. We rigorously evaluate our evaluation against baselines, both in synthetic cases (by comparing to ground-truth) and in real-world cases (by evaluating the consistency of our quantitative metrics with metrics from qualitative think-aloud studies).

- Chapter 8, based on Ross and Doshi-Velez [2021], introduces benchmarks, metrics, and algorithms for learning representations with deep hierarchical structure—that is, representations whose dimensions are organized into trees, only one branch of which is active at a time. Enforcing such structure can both improve interpretability (since models can be decomposed into simpler parts) and also faithfulness to the real-world processes that may have generated the data (which are often hierarchical in structure).

Together, these contributions chart a path towards learning machine learning models which are truly right for the right reasons: trained to make predictions in terms of concepts their users understand, in ways that are consistent with what their users know.

Part II

Input Gradient Penalties

Chapter 2

Background on Input Gradients

In this chapter, we define input gradients, introduce notation, and review related work that will be common across Part II.

2.1 Definitions and Related Work

Inputs X. The starting point for many questions in machine learning is some set of **inputs**, which we might also call instances or examples, and which we generally refer to as x or X (somewhat interchangeably, though X more frequently refers to a design matrix of many inputs, while x more frequently refers to an individual input). In most cases we have N such inputs, and each input X_n is generally a D-dimensional vector. Examples of inputs we will consider include images, tabular data, medical records, and time series.

Targets y. We also often have a set of **targets**, which we might also call classes or labels, which we generally denote y (which has one element y_n corresponding to each input X_n). In the following chapters, we will be primarily concerned with ys that represent the answers to one-of-K classification problems; that is, each y_n belongs to one of K different classes, and is represented as a "one-hot" encoded vector.[1] Although in general we consider the case of multi-class classification (where $K > 2$), sometimes we also consider the special case of

[1] By one-hot encoding, we mean that if y_n is of class k, its kth element is set to 1 and the rest are set to 0.

binary classification, where y may instead be a variable in $\{0, 1\}$ representing a true-or-false outcome (e.g. whether x represents an image of a cat or a dog, or whether an intensive care unit patient received fluids within 4 hours of the most recent measurement in x).

Data distributions $p(x, y)$. In most cases, we assume inputs and targets are drawn independently and identically distributed from some **data distribution** $p(x, y)$. In general, we may not have access to a closed form expression for $p(x, y)$ (e.g. if our data is coming from an intensive care unit, where the process that generates the instances and targets is complex and impossible to fully characterize), but instead just have a set of samples drawn from it. Often, there will be multiple data distributions of interest, specifically a **training distribution** and a **test distribution**. In most machine learning literature, these are often assumed to be the same (which can be simulated from a static dataset by randomly subdividing it into training and test sets). However, in this chapter, we will more frequently consider cases where these distributions differ.

Discriminative models $f(x; \theta)$. Frequently, our primary goal in machine learning is to predict y given x over the support of $p(x, y)$, and our strategy for achieving this goal is to learn a **discriminative model** f such that $f(x) \approx y$. We might also refer f just as a model, a function, or a network, and in the case of classification (the primary focus of this chapter), we will refer to f as a classifier. For a particular x, y pair and set of parameters θ, we might shorten $f(x; \theta)$ to $\hat{y}$ for conciseness.

Under the hood, f is generally a *neural network* with some fixed architecture but a variable set of **parameters** (or weights) θ. Changing the parameters changes the function, and in general we assume that for any function of interest over any finite domain, there exists some θ such $f(\cdot; \theta)$ closely approximates it [52]. $f(\cdot; \theta)$ (sometimes shortened to f_θ) is assumed to be differentiable, both with respect to its parameters θ and its inputs x.

Loss functions $\mathcal{L}(\theta)$. We learn models $f(\cdot; \theta)$ by trying to minimize a **loss function** $\mathcal{L}(\theta)$ over their parameters. In the framework of empirical risk minimization [232], loss functions

are often expressed as the expectation of some individual-instance loss ℓ over the data distribution $p(x, y)$, e.g.

$$\mathcal{L}(\theta) = \mathbb{E}_{x,y\sim p(x,y)}\left[\ell(y, f(x;\theta))\right] = \mathbb{E}[\ell(y, \hat{y})] \tag{2.1}$$

Often, there is a probabilistic interpretation to loss functions as a negative log likelihood $-\log p(y|\hat{y})$ of observing the targets y given model predictions $\hat{y} = f(x;\theta)$. In the case that x and y represent multiple sets of independent and identically distributed draws from the data distribution, the negative log likelihood naturally takes the form of a sum, which can be interpreted as an expectation. In classification (where y is a one-hot encoded class label), we often set $\mathcal{L}$ to be the average "cross-entropy" H, defined as

$$\mathcal{L}(\theta) = \mathbb{E}_{x,y\sim p(x,y)}\left[H(y, \hat{y})\right] = \mathbb{E}_{x,y\sim p(x,y)}\left[-\sum_{k=1}^{D_y} y_k \log f_\theta(x)_k\right]. \tag{2.2}$$

In practice, we will use stochastic gradient descent over small batches of data drawn from the dataset, often using Adam [112], to approximately evaluate and minimize these loss functions.

For classification, in addition to the loss function, we are also generally interested in the **accuracy** metric $\mathbb{E}_{x,y\sim p(x,y)}\left[\mathbb{1}(y = \arg\max_k f(x;\theta)_k)\right]$, or the probability that the true class label equals the class label with the greatest predicted probability (on new data drawn from $p(x, y)$). We generally approximate this probability by taking an average over a held-out test dataset.

Input gradients ∇_x of f. In the following chapters, one of the primary quantities we will analyze and penalize is the **input gradient**. In the special case that $f(x)$ is a regressor that outputs real numbers, then we need only consider $\nabla_x f(x)$ (sometimes shortened to $f_x(x)$), i.e. the vector of partial derivatives $\left\langle \frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x), \ldots \frac{\partial f}{\partial x_D}(x) \right\rangle$.

However, in Part II, we are generally more concerned with $f(x)$ which are classifiers. To start, assume that $f(x)$ is a binary classifier outputting a probability from 0 to 1. In this case, we could analyze the gradients of the raw probabilities $\nabla_x f(x)$ (and in Chapter 3 we

will often visualize them), but they often suffer from problems of saturation: for points x where $f(x) \approx 0$ or $f(x) \approx 1$, the actual probabilities change very little with changes in x, so the gradients of many inputs are near 0 in magnitude and can underflow. Instead, we often want to consider either the gradients of the log probability $\nabla_x \log f(x)$, or the network's "logits" (i.e. its raw outputs from $-\infty$ to ∞ that we convert to $(0,1)$ using the sigmoid function). It is also often useful to consider the gradient of the log-odds $\nabla_x \log \frac{f(x)}{1-f(x)}$.

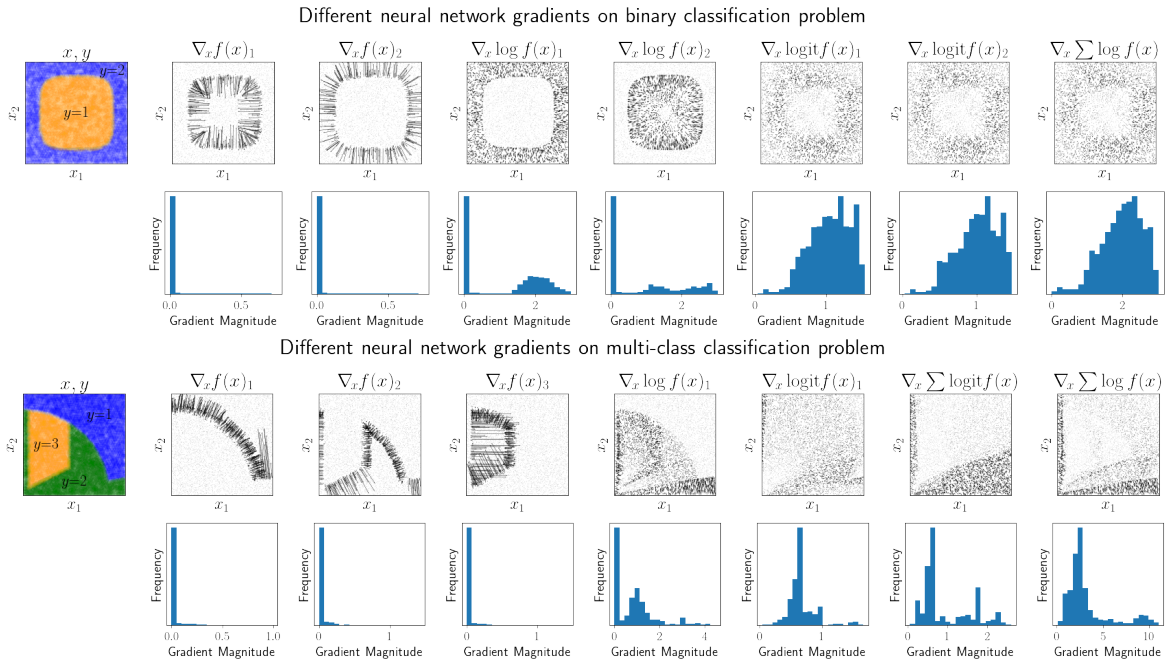


Figure 2.1: *Comparison of different input gradients of binary (top) and multi-class (bottom) classifiers (neural networks with two hidden layers and ReLU activations, trained to near-100% accuracy on the datasets shown in the leftmost plots). Gradients of specific probabilities are only nonzero near decision boundaries, gradients of specific log-probabilities are nonzero over decision regions, and gradients of logits (raw $-\infty$ to ∞ network outputs before conversion to probabilities) remain nonzero everywhere. The gradient of the sum of log probabilities (rightmost plots) also generally remains nonzero, and can be interpreted as the information distance between the model's predictions and a maximally uncertain prediction. The gradients of most of these quantities provide a normal to the decision boundary.*

For multi-class classification (where we output multiple probabilities that sum to 1), we could compute any of the above quantities for any of the individual outputs. In general, though, we have a full Jacobian matrix to consider, and it is often useful to compute various reductions. Although we have no analogue to the binary log-odds ($\log p/(1-p)$

is most meaningful when there is only one probability), the gradient of the sum of log probabilities ($\nabla_x \sum_k \log f_\theta(x_n)_k$) is often a particularly useful quantity to consider: it tends to remain nonzero even when individual outputs saturate, and it can also be interpreted as a cross-entropy between the output of f and a discrete uniform distribution (the information distance between maximal and minimal uncertainty). Alternately, if ground-truth class labels are available, the input gradient can be taken with respect to the cross entropy loss ($\nabla_x \sum_k y_{nk} \log f_\theta(x_n)_k$). A summary of different options is shown in Figure 2.1. Geometrically, most of these input gradient variants are orthogonal to the classifier's decision boundaries (borders between regions where the classifier makes different predictions).

Distribution shifts, ambiguity, and robustness. In many cases, we are interested in the performance of our models f under **distribution shift**, i.e. when considering data drawn from some a testing distribution $q(x, y)$ that differs from the training distribution $p(x, y)$. If our loss or accuracy under q is not too different from the loss under p (and if both are low in absolute terms), we might say that f_θ is **robust** to the distribution shift from p to q.

Now, there are many kinds of possible distribution shifts, including the presence of an adversary or changes in the association of confounding variables, and some make robustness impossible (e.g. if the marginal distribution $q(x) = p(x)$, but the conditional distribution $q(y|x) \neq p(y|x)$). The type of distribution shift that motivates this dissertation is one in which the original classification or regression problem is **ambiguous** in some sense; that is, over the training distribution $p(x, y)$, there are many parameters θ that could all separately approximately minimize $\mathbb{E}_p[\ell(y, f(x; \theta))]$, but only a (non-empty) subset of them approximately minimize loss or maximize accuracy under both p and q (or more generally, under some large set of distributions which Semenova et al. [2019] call a "Rashomon set" after the Kurosawa classic; see also D'Amour et al. [2020] and Marx et al. [2020]). These are cases where different, separately manipulable aspects of x can each redundantly predict y, or more informally, where the same question could be answered for multiple reasons. The beginning of Chapter 1 contains several examples, e.g. where a sports car might be recognized at train time by either its shape or its color (or a complex nonlinear combination

of both), but at test time only shape will suffice. To achieve robustness to these types of shifts, it is sufficient (and may be necessary) for our models f to be *right for the right reasons*.

Interpretability, implicit decision rules, and explanations. Throughout this thesis, we will generally follow Doshi-Velez and Kim [2017] in defining interpretability as the capacity of a model f to be understood by its human users (e.g. doctors or chemists) in some relevant context (e.g. treating patients or discovering molecules). This definition depends on having coherent notions of the relevant context as well as human understanding, which we will explore in much more detail in Part III when we consider representations.

However, in Part II, we will assume that classifiers have some **implicit decision rules** for converting an input x into a prediction $\hat{y}$, which may be possible for a human user to completely understand (in which case model is completely **interpretable**), or which may be so complex that no human user could even partially understand them (in which case the model is completely uninterpretable). We define **explanations** as any artifact that gives a user information about the implicit decision rules of the model. Explanations based on feature importance (i.e. an importance score for each dimension of x) have received a great deal of attention (which we discuss below), but we define the term very broadly; simply analyzing successes and failure cases can provide a great deal of explanatory insight [236, 6]. An explanation is useful insofar as it increases a human user's understanding of the model in a way that is contextually relevant.

Interpretability is distinct from explainability. It is possible that a model's implicit decision rules are simple for human users to understand, but nevertheless opaque, because we lack a clear means of discovering or explaining them. Alternatively, it is possible that a model's implicit decision rules are completely transparent, but nevertheless incomprehensible, because they are too complicated for users to understand. We must not conflate the transparency of a model with interpretability of the function it reifies.

For more general perspectives on interpretability, see also Lipton [2016] and Miller [2019].

Input gradients as explanations. Input gradient components can be interpreted as the weights of a linear model that best approximates the local behavior of a learned function, or equivalently the sensitivity of a learned function output to small changes in an input feature. By visualizing the magnitude and/or direction of each input gradient component (either for an individual input or as a distribution across many inputs), we can gain insight into which features most affect model predictions, and how. In this way, input gradients function have explanatory value, especially if the number of features with significant gradient magnitude (relative to the scale of the feature) is much smaller than the total number of features D, or if there are coherent patterns in the positive / negative associations of each feature. Input gradients received significant attention in the early days of interpretable machine learning research as a promising technique [211, 137, 90]; see Baehrens et al. [2010] for a particularly good introduction.

Alternative forms of feature importance. However, input gradients for large convolutional networks often look "noisy" or seem difficult to interpret. While we will argue later that this noisiness is often a property of the model, not the explanation, a number of alternatives to "raw" input gradients have also been proposed as better or more interpretable measurements of feature importance (which can be defined in various ways). Examples include layer-wise relevance propagation [17], DeepLIFT [209], Contextual Decomposition [164], Deep Taylor Decomposition [162], and Guided Backpropagation [219]. Some of them, such as SmoothGrad [216], Integrated Gradients [222], and Expected Gradients [70] are based on raw gradients but aggregated/averaged over paths/regions in input space, which allows them capture larger-scale behavior or quantify a feature's overall contribution to a prediction (rather than the marginal effect of changing it infinitesimally). Shapley values [150], though difficult to compute, uniquely satisfy numerous axioms for how best to attribute a nonlinear model's predictions to each individual feature.

A full survey or comparison of these forms of feature importance is beyond the scope of this thesis. For that, we refer readers to the growing literature on evaluation metrics and sanity checks for feature importance scores [111, 5, 98]. However, we note that raw input

gradients consistently pass such sanity checks, even if they sometimes seem less meaningful for certain models. More recent works have achieved compelling results by regularizing different quantities; Erion et al. [2019] and Rieger et al. [2020] are excellent examples. For the purposes of this thesis, though, input gradients are still sufficiently meaningful to let us demonstrate our general point: that explanation regularization can help us learn models which are right for the right reasons.

Criticisms of feature importance (and alternatives). However, it is worth taking a step back to consider the limitations of feature importances categorically. Rudin [2019] argues the explanatory value of feature importance is often egregiously overstated, which user studies support [42, 6, 49, 207]. Hancox-Li and Kumar [2021] argue from a more philosophical perspective that there are serious epistemological problems with suggesting that a single D-dimensional importance vector can somehow "explain" the predictions of an immensely complex nonlinear model with millions or billions of parameters—and that the way we implicitly characterize such artifacts as canonical or complete is both incorrect and potentially harmful to people who will be affected by how the model is used and perceived.

In certain chapters of this thesis, we will still use the term "explanation" to describe input gradients, despite its problematic connotation of completeness, because we want to illustrate a more general idea (towards which we aspire, and hope to inspire others). However, this term should be taken only to mean an artifact that provides any amount of information, however small, about a model's implicit decision rules. No explanation completely describes its model, unless the model's class is, per Rudin [2019], "inherently interpretable."

To that end, it is worth referring readers to a variety of model classes whose decision rules are explicit and can be visualized in full (even if the functions they learn are difficult to understand). Classic examples include logistic regression [114] and decision trees [231]; more recently, generalized additive models [39, 97], decision sets [131] and rule lists [230] have become popular. Although these model architectures generally work best for relatively low-dimensional inputs whose features have consistent meanings (and not for domains like

audio or images), Chen et al. [2018] provide an approach for making predictions based on transparent functions of similarity between new inputs and inputs from the training set (though the similarity metric is learned and not necessarily simple). The strategy of explaining predictions in terms of other instances can also be used directly for neural networks, e.g. with influence functions [116] or Shapley value techniques applied to training instances [77]. This approach has even been extended to explanation regularization by Shao et al. [2021]. Finally, it can also be very useful just to consider higher-order feature interactions. Shapley residuals [124] can help users understand to what extent a feature importance explanation may be obscuring details about interactions. Archipelago [228] actually attempts to quantify the importance of interactions (to arbitrarily high orders). To summarize, we may be able to make machine learning models much more transparent by either changing our model class (away from neural networks) or our explanation method (away from feature importance).

But again, our goal in Part II is not to improve the transparency of machine learning models, but instead to help them learn better functions—functions which are more robust to distribution shifts, more consistent with expert knowledge, and which perhaps are more understandable, even if difficult to visualize. For this task, input gradients remain surprisingly effective. We will return to some of their deficiencies in Part III, however, when we consider representations.

2.2 Outline of Part II

In the remaining chapters of Part II, we present three approaches for using input gradient penalties to learn better functions:

- In Chapter 3, we use input gradient penalties (on the sum of log probabilities) to guide neural networks towards or away from learning particular implicit decision rules on ambiguous problems. We also introduce an iterative method for learning an ensemble of neural networks that make predictions for different reasons.

- In Chapter 4, we significantly improve on this ensemble learning method with local independence training (LIT), which simultaneously trains neural networks to make correct predictions with orthogonal gradients. We verify LIT is more effective than baselines at recovering models which extrapolate in maximally diverse ways, which can be helpful for interpretability, uncertainty quantification, robustness, and accuracy. We also briefly explore how the size of the largest accurate LIT ensemble (for a given regularization strength) may be a proxy for the degree of ambiguity present in the dataset, or the size of the "Rashomon set" [203].

- In Chapter 5, we consider how input gradient penalties can help convolutional neural networks achieve both improved adversarial robustness and interpretability. We demonstrate a qualitative difference between gradient-regularized models and those obtained by distillation or gradient masking defenses. Inspired by Ilyas et al. [2019], we also discuss how adversarial vulnerability can be understood as another example of dataset ambiguity.

Chapter 3

Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations[1]

3.1 Introduction

High-dimensional real-world datasets are often full of ambiguities. When we train classifiers on such data, it is frequently possible to achieve high accuracy using classifiers with qualitatively different decision boundaries. To narrow down our choices and encourage robustness, we usually employ regularization techniques (e.g. encouraging sparsity or small parameter values). We also structure our models to ensure domain-specific invariances (e.g. using convolutional neural nets when we would like the model to be invariant to spatial transformations). However, these solutions do not address situations in which our training dataset contains subtle confounds or differs qualitatively from our test dataset. In these cases, our model may fail to generalize no matter how well it is tuned.

[1]This chapter is based on 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 Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2662–2670, 2017. doi: 10.24963/ijcai.2017/371. URL https://doi.org/10.24963/ijcai.2017/371.

Such generalization gaps are of particular concern for uninterpretable models such as neural networks, especially in sensitive domains. For example, [39] describe a model intended to prioritize care for patients with pneumonia. The model was trained to predict hospital readmission risk using a dataset containing attributes of patients hospitalized at least once for pneumonia. Counterintuitively, the model learned that the presence of asthma was a *negative* predictor of readmission, when in reality pneumonia patients with asthma are at a greater medical risk. This model would have presented a grave safety risk if used in production. This problem occurred because the outcomes in the dataset reflected not just the severity of patients' diseases but the quality of care they initially received, which was higher for patients with asthma.

This case and others like it have motivated recent work in interpretable machine learning, where algorithms provide explanations for domain experts to inspect for correctness before trusting model predictions. However, there has been limited work in optimizing models to find not just the right prediction but also the *right explanation*. Toward this end, this chapter makes the following contributions:

- We introduce a series of ambiguous prediction tasks, where class labels can be redundantly predicted from inputs using multiple independent mechanisms.
- We confirm that both input gradient explanations and baseline sample-based explanation methods (specifically LIME [183]) can be used to detect which mechanisms the models use to make predictions.
- Given annotations about incorrect explanations for particular inputs, we efficiently optimize the classifier to learn to make predictions using alternate mechanisms (to be right for better reasons).
- When annotations are not available, we sequentially discover classifiers with similar accuracies but qualitatively different decision boundaries for domain experts to inspect for validity.

3.2 Related Work

Summarizing our discussion in Chapter 2, we briefly redefine several important terms in interpretable machine learning. All classifiers have *implicit decision rules* for converting an input into a decision, though these rules may be opaque. A model is *interpretable* if it provides explanations for its predictions in a form humans can understand; an *explanation* provides reliable information about the model's implicit decision rules for a given prediction. In contrast, we say a machine learning model is *accurate* if most of its predictions are correct, but only *right for the right reasons* if the implicit rules it has learned generalize well and conform to domain experts' knowledge about the problem.

Explanations can take many forms [107] and evaluating the quality of explanations or the interpretability of a model is difficult [142, 60]. However, within the machine learning community there has been convergence [149] around local counterfactual explanations, where we show how perturbing an input x in various ways will affect the model's prediction $\hat{y}$. This approach to explanations can be domain- and model-specific (e.g. "annotator rationales" used to explain text classifications by [138, 134, 243]). Alternatively, explanations can be model-agnostic and relatively domain-general, as exemplified by LIME (Local Interpretable Model-agnostic Explanations, [183, 213]) which trains and presents local sparse models of how predictions change when inputs are perturbed.

The per-example perturbing and fitting process used in models such as LIME can be computationally prohibitive, especially if we seek to explain an entire dataset during each training iteration. If the underlying model is differentiable, one alternative is to use input gradients as local explanations (Baehrens et al. [2010] provides a particularly good introduction; see also [202, 211, 137, 90]). The idea is simple: the gradients of the model's output probabilities with respect to its inputs literally describe the model's decision boundary (see Figure 2.1). They are similar in spirit to the local linear explanations of LIME but much faster to compute.

Input gradient explanations are not perfect for all use-cases—for points far from the decision boundary, they can be saturate and do not always capture the idea of salience (see

discussion and alternatives in Section 2.1). However, they are sufficient for regularizing the decision boundary, given their close relationship with it. More broadly, relatively few works on interpretable machine learning (as of the time the paper this chapter presents was written) attempt to optimize explanations for correctness. For SVMs and specific text classification architectures, there existed prior work on incorporating human input into decision boundaries in the form of annotator rationales [240, 59, 243]. Unlike our approach, these works are either tailored to specific domains or do not fully close the loop between generating explanations and constraining them. More recently, a number of other works have built upon the methods introduced in this chapter, making concrete improvements [70, 63, 200, 235, 185, 220, 64]. We will discuss these in more detail later, but the methods we introduce are still sufficient to demonstrate the general idea of explanation regularization.

3.3 Method

We wish to develop a method to train models that are right for the right reasons. If explanations faithfully describe a model's underlying behavior, then constraining its explanations to match domain knowledge should cause its underlying behavior to more closely match that knowledge too. We first describe how input gradient-based explanations lend themselves to efficient optimization for correct explanations in the presence of domain knowledge, and then describe how they can be used to efficiently search for qualitatively different decision boundaries when such knowledge is not available.

3.3.1 Loss Functions that Constrain Explanations

When constraining input gradient explanations, there are at least two basic options: we can either constrain them to be large in relevant areas or small in irrelevant areas. However, because input gradients for relevant inputs in many models *should* be small far from the decision boundary, and because we do not know in advance how large they should be, we opt to shrink irrelevant gradients instead.

Formally, we define an annotation matrix $A \in \{0,1\}^{N \times D}$, which are binary masks

indicating whether dimension d should be irrelevant for predicting observation n. We would like $\nabla_X \hat{y}$ to be near 0 at these locations. To that end, we optimize a loss function $L(\theta, X, y, A)$ of the form

$$L(\theta, X, y, A) = \underbrace{\sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \log(\hat{y}_{nk})}_{\text{Right answers}} + \underbrace{\lambda_1 \sum_{n=1}^{N} \sum_{d=1}^{D} \left(A_{nd} \frac{\partial}{\partial x_{nd}} \sum_{k=1}^{K} \log(\hat{y}_{nk}) \right)^2}_{\text{Right reasons}} + \underbrace{\lambda_2 \sum_i \theta_i^2}_{\text{Regular}},$$

which contains familiar cross entropy and θ regularization terms along with a new regularization term that discourages the input gradient from being large in regions marked by A. This term has a regularization parameter λ_1 which should be set such that the "right answers" and "right reasons" terms have similar orders of magnitude; see Figure 3.10 for more details. Note that this loss penalizes the gradient of the *log* probability, which performed best in practice, though in many visualizations we show f_X, which is the gradient of the predicted probability itself. Summing across classes led to slightly more stable results than using the predicted class log probability $\max \log(\hat{y}_k)$, perhaps due to discontinuities near the decision boundary (though both methods were comparable). We did not explore regularizing input gradients of specific class probabilities, though this would be a natural extension.

Because this loss function is differentiable with respect to θ, we can easily optimize it with gradient-based optimization methods. We do not need annotations (nonzero A_n) for every input in X, and in the case $A = 0^{N \times D}$, the explanation term has no effect on the loss. At the other extreme, when A is a matrix of all 1s, it encourages the model to have small gradients with respect to its inputs; this can improve generalization on its own [62]. Between those extremes, it biases our model against *particular* implicit rules.

This penalization approach enjoys several desirable properties. Alternatives that specify a single A_d for all examples presuppose a coherent notion of global feature importance, but when decision boundaries are nonlinear many features are only relevant in the context of specific examples. Alternatives that simulate perturbations to entries known to be irrelevant (or to determine relevance as in [183]) require defining domain-specific perturbation logic;

our approach does not. Alternatives that apply hard constraints or completely remove elements identified by A_{nd} miss the fact that the entries in A may be imprecise even if they are human-provided. Thus, we opt to preserve potentially misleading features but softly penalize their use.

3.3.2 Find-Another-Explanation: Diverse Rule Discovery without Annotations

Although we can obtain the annotations A via experts as in [240], we may not always have this extra information or know the "right reasons." In these cases, we propose an approach that iteratively adapts A to discover multiple models accurate for *qualitatively different* reasons; a domain expert could then examine them to determine which is the right for the best reasons. Specifically, we generate a "spectrum" of models with different decision boundaries by iteratively training models, explaining X, then training the next model to differ from previous iterations:

$$
\begin{aligned}
A_0 &= 0, & \theta_0 &= \arg\min_{\theta} L(\theta, X, y, A_0), \\
A_1 &= M_c\left[f_X|\theta_0\right], & \theta_1 &= \arg\min_{\theta} L(\theta, X, y, A_1), \\
A_2 &= M_c\left[f_X|\theta_1\right] \cup A_1, & \theta_2 &= \arg\min_{\theta} L(\theta, X, y, A_2), \\
& & &\dots
\end{aligned}
$$

where the function M_c returns a binary mask indicating which gradient components have a magnitude ratio (their magnitude divided by the largest component magnitude) of at least c and where we abbreviated the input gradients of the entire training set X at θ_i as $f_X|\theta_i$. In other words, we regularize input gradients where they were largest in magnitude previously. If, after repeated iterations, accuracy decreases or explanations stop changing (or only change after significantly increasing λ_1), then we may have spanned the space of possible models.[2] All of the resulting models will be accurate, but for different reasons; although we do not know which reasons are best, we can present them to a domain expert

[2]Though one can design simple pathological cases where we do not discover all models with this method; we explore an alternative version in Chapter 4 that addresses some of these cases.

for inspection and selection. We can also prioritize labeling or reviewing examples about which the ensemble disagrees. Finally, the size of the ensemble provides a rough measure of dataset redundancy.

3.4 Experiments

We demonstrate explanation generation, explanation constraints, and the find-another-explanation method on a toy color dataset and three real-world datasets. In all cases, we used a multilayer perceptron with two hidden layers of size 50 and 30, ReLU nonlinearities with a softmax output, and a $\lambda_2 = 0.0001$ penalty on $\|\theta\|_2^2$. We trained the network using Adam [112] with a batch size of 256 and Autograd [159]. For most experiments, we used an explanation L2 penalty of $\lambda_1 = 1000$, which gave our "right answers" and "right reasons" loss terms similar magnitudes. More details about cross-validation are included in Figure 3.10. For the cutoff value c described in Section 3.3.2 and used for display, we often chose 0.67, which tended to preserve 2-5% of gradient components (the average number of qualifying elements tended to fall exponentially with c). Code for all experiments is available at `https://github.com/dtak/rrr`.

3.4.1 Toy Color Dataset

We created a toy dataset of $5 \times 5 \times 3$ RGB images with four possible colors. Images fell into two classes with two independent decision rules a model could implicitly learn: whether their four corner pixels were all the same color, and whether their top-middle three pixels were all different colors. Images in class 1 satisfied both conditions and images in class 2 satisfied neither. Because only corner and top-row pixels are relevant, we expect any faithful explanation of an accurate model to highlight them.

In Figure 3.1, we see both LIME and input gradients identify the same relevant pixels, which suggests that (1) both methods are effective at explaining model predictions, and (2) the model has learned the corner rather than the top-middle rule, which it did consistently across random restarts.

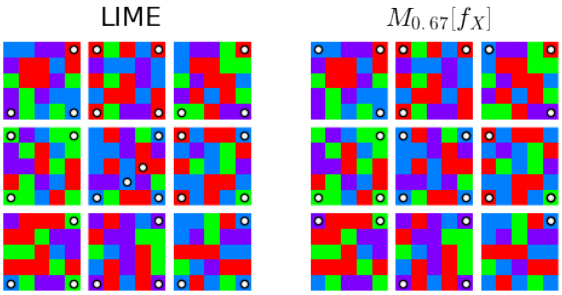


Figure 3.1: *Gradient vs. LIME explanations of nine perceptron predictions on the Toy Color dataset. For gradients, we plot dots above pixels identified by $M_{0.67}[f_X]$ (the top 33% largest-magnitude input gradients), and for LIME, we select the top 6 features (up to 3 can reside in the same RGB pixel). Both methods suggest that the model learns the corner rule.*

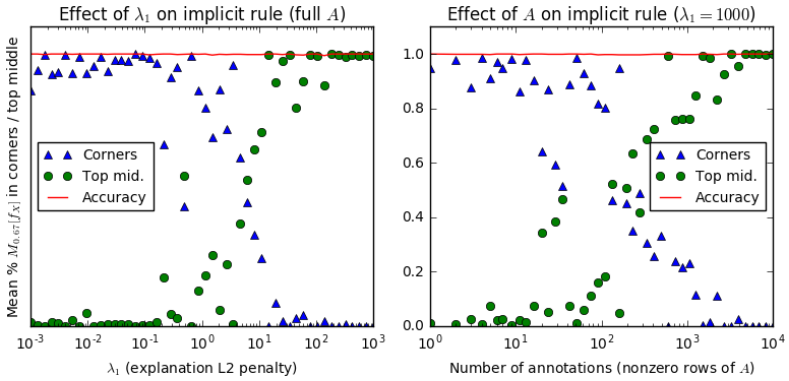


Figure 3.2: *Implicit rule transitions as we increase λ_1 and the number of nonzero rows of A. Pairs of points represent the fraction of large-magnitude ($c = 0.67$) gradient components in the corners and top-middle for 1000 test examples, which almost always add to 1 (indicating the model is most sensitive to these elements alone, even during transitions). Note there is a wide regime where the model learns a hybrid of both rules.*

However, if we train our model with a nonzero A (specifically, setting $A_{nd} = 1$ for corners d across examples n), we were able to cause it to use the other rule. Figure 3.2 shows how the model transitions between rules as we vary λ_1 and the number of examples penalized by A. This result demonstrates that the model can be made to learn multiple rules despite only one being commonly reached via standard gradient-based optimization methods. However, it depends on knowing a good setting for A, which in this case would still require annotating on the order of 10^3 examples, or 5% of our dataset (although always including examples with annotations in Adam minibatches let us consistently switch rules with only 50 examples, or 0.2% of the dataset).

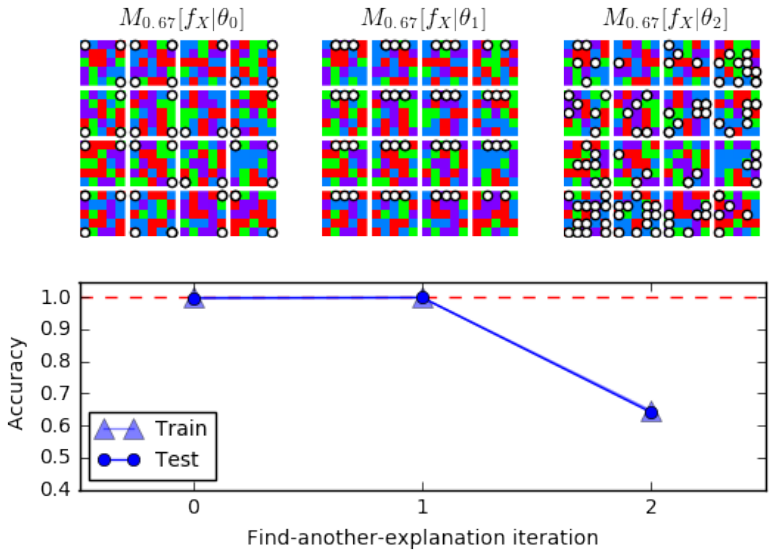


Figure 3.3: *Rule discovery using find-another-explanation method with 0.67 cutoff and $\lambda_1 = 10^3$ for θ_1 and $\lambda_1 = 10^6$ for θ_2. Note how the first two iterations produce explanations corresponding to the two rules in the dataset while the third produces very noisy explanations (with low accuracies).*

Additionally, Figure 3.3 shows we can use the find-another-explanation technique from Sec. 3.3.2 to discover the other rule without being given A. Because only two rules lead to high accuracy on the test set, the model performs no better than random guessing when prevented from using either one (although we have to increase the penalty high enough that this accuracy number may be misleading - the essential point is that after the first iteration, explanations stop changing).

Lastly, though not directly relevant to the discussion on interpretability and explanation, we demonstrate the potential of explanations to reduce the amount of data required for training in Figure 3.4. In this experiment, we train with varying numbers of training examples N on the Toy Color dataset, using four variants of A (with λ_1 chosen to match loss terms at each N). We find that when A is set to the Pro-Rule 1 mask, which penalizes all pixels except the corners, we reach 95% accuracy with fewer than 100 examples (as compared to $A = 0$, where we need almost 10000). Penalizing the top-middle pixels (Anti-Rule 2) or all pixels except the top-middle (Pro-Rule 2) also consistently improves accuracy relative to data. Penalizing the corners (Anti-Rule 1), however, actually reduces accuracy until we

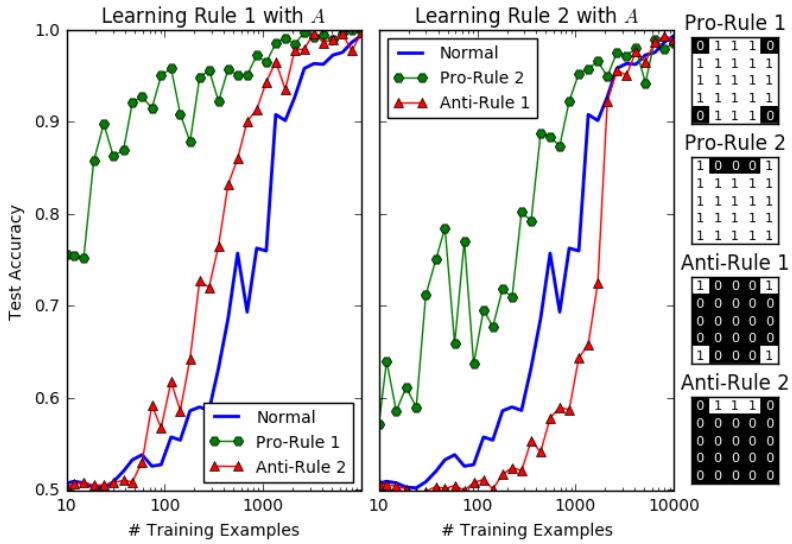


Figure 3.4: *Test accuracy by training set size on the Toy Color dataset for four different domain knowledge incorporate strategies (baseline $A = 0$ approach in blue). Explanation regularization can potentially reduce data requirements when encouraging the model to learn a simple rule (Pro-Rule 1, green on left) or discouraging it from learning a complex rule (Anti-Rule 2, red on left). However, it actually increases data requirements when discouraging the model from learning the simple rule (Anti-Rule 1, red on right).*

reach a threshold N. This may be because the corner pixels can match in 4 ways, while the top-middle pixels can differ in $4 \cdot 3 \cdot 2 = 24$ ways, suggesting that Rule 2 could be inherently harder to learn from data and positional explanations alone. These results suggest that explanation regularization may indeed be useful for reducing data requirements, but the magnitude and direction of the effect depends on the associated implicit decision rules.

3.4.2 Real-world Datasets

To demonstrate real-world, cross-domain applicability, we test our approach on variants of three familiar machine learning text, image, and tabular datasets:

- **20 Newsgroups:** As in [183], we test input gradients on the `alt.atheism` vs. `soc.religion.christian` subset of the 20 Newsgroups dataset [140]. We used the same two-hidden layer network architecture with a TF-IDF vectorizer with 5000 components, which gave us a 94% accurate model for $A = 0$.

- **Iris-Cancer:** We concatenated all examples in classes 1 and 2 from the Iris dataset with the the first 50 examples from each class in the Breast Cancer Wisconsin dataset [140] to create a composite dataset $X \in \mathbb{R}^{100\times 34}, y \in \{0, 1\}$. Despite the dataset's small size, our network still obtains an average test accuracy of 92% across 350 random $\frac{2}{3}$-$\frac{1}{3}$ training-test splits. However, when we modify our test set to remove the 4 Iris components, average test accuracy falls to 81% with higher variance, suggesting the model learns to depend on Iris features and suffers without them. We verify that our explanations reveal this dependency and that regularizing them avoids it.

- **Decoy MNIST:** On the baseline MNST dataset [132], our network obtains 98% train and 96% test accuracy. However, in Decoy MNIST, images x have 4×4 gray swatches in randomly chosen corners whose shades are functions of their digits y in training (in particular, $255 - 25y$) but are random in test. On this dataset, our model has a higher 99.6% train accuracy but a much lower 55% test accuracy, indicating that the decoy rule misleads it. We verify that both gradient and LIME explanations let users detect this issue and that explanation regularization lets us overcome it.

Input gradients are consistent with sample-based methods such as LIME, and faster. On 20 Newsgroups (Figure 3.5), input gradients are less sparse but identify all of the same words in the document with similar weights. Note that input gradients also identify words outside the document that would affect the prediction if added.

On Decoy MNIST (Figure 3.6), both LIME and input gradients reveal that the model predicts 3 rather than 7 due to the color swatch in the corner. Because of their fine-grained resolution, input gradients sometimes better capture counterfactual behavior, where extending or adding lines outside of the digit to either reinforce it or transform it into another digit would change the predicted probability (see also Figure 3.11). LIME, on the other hand, better captures the fact that the main portion of the digit is salient (because its super-pixel perturbations add and remove larger chunks of the digit).

On Iris-Cancer (Figure 3.7), input gradients actually outperform LIME. We know from the accuracy difference that Iris features are important to the model's prediction, but LIME

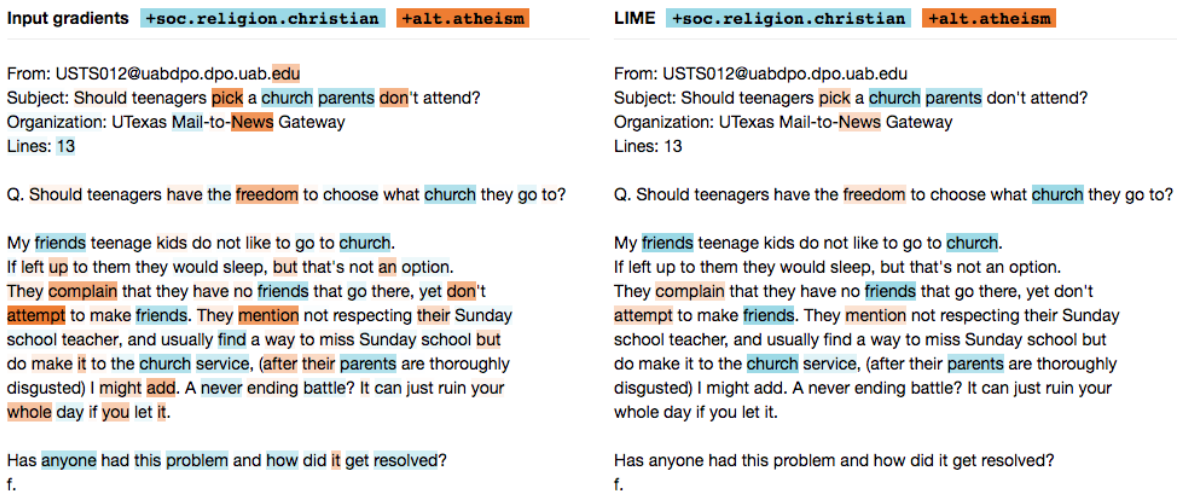


Figure 3.5: *Words identified by LIME vs. gradients on an example from the atheism vs. Christianity subset of 20 Newsgroups. More examples are available at https://github.com/dtak/rrr. Words are blue if they support soc.religion.christian and orange if they support alt.atheism, with opacity equal to the ratio of the magnitude of the word's weight to the largest magnitude weight. LIME generates sparser explanations but the weights and signs of terms identified by both methods match closely. Note that both methods reveal some aspects of the model that are intuitive ("church" and "service" are associated with Christianity), some aspects that are not ("13" is associated with Christianity, "edu" with atheism), and some that are debatable ("freedom" is associated with atheism, "friends" with Christianity).*

only identifies a single important feature, which is from the Breast Cancer dataset (even when we vary its perturbation strategy). This example, which is tabular and contains continuously valued rather categorical features, may represent a pathological case for LIME, which operates best when it can selectively mask a small number of meaningful chunks of its inputs to generate perturbed samples. For truly continuous inputs, it should not be surprising that explanations based on gradients perform best.

There are a few other advantages input gradients have over sample-based perturbation methods. On 20 Newsgroups, we noticed that for very long documents, explanations generated by the sample-based method LIME are often overly sparse (see Figures A.1 and A.2), and there are many words identified as significant by input gradients that LIME ignores. This may be because the number of features LIME selects must be passed in as a parameter beforehand, and it may also be because LIME only samples a fixed number of times. For sufficiently long documents, it is unlikely that sample-based approaches will mask

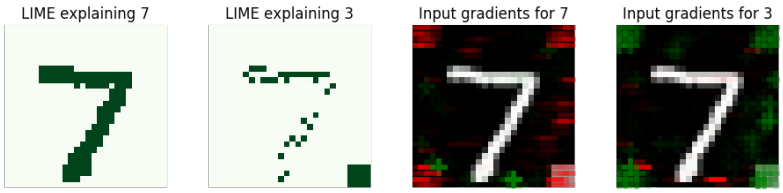


Figure 3.6: *Input gradient explanations for Decoy MNIST vs. LIME, using the LIME image library [182]. In this example, the model incorrectly predicts 3 rather than 7 because of the decoy swatch.*

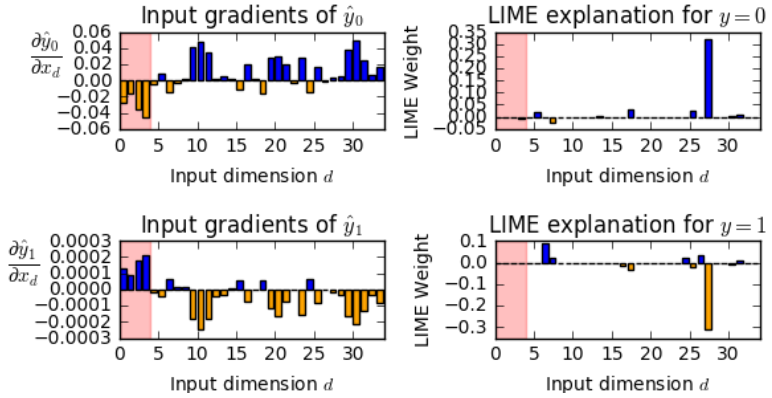


Figure 3.7: *Iris-Cancer features identified by input gradients vs. LIME, with Iris features highlighted in red. Input gradient explanations are more faithful to the model. Note that most gradients change sign when switching between $\hat{y}_0$ and $\hat{y}_1$, and that the magnitudes of input gradients are different across examples, which provides information about examples' proximity to the decision boundary.*

every word even once, meaning that the output becomes increasingly nondeterministic—an undesirable quality for explanations. To resolve this issue, one could increase the number of samples, but that would increase the computational cost since the model must be evalutated at least once per sample to fit a local surrogate. Input gradients, on the other hand, only require on the order of one model evaluation *total* to generate an explanation of similar quality (generating gradients is similar in complexity to predicting probabilities), and furthermore, this complexity is based on the vector length, *not* the document length. This issue (underscored by Table 3.1) highlights some inherent scalability advantages input gradients enjoy over sample-based perturbation methods.

Given annotations, input gradient regularization finds solutions consistent with domain

Table 3.1: *Runtimes for gradients vs. LIME.*

	LIME	Gradients	Dimension of x
Iris-Cancer	0.03s	0.000019s	34
Toy Colors	1.03s	0.000013s	75
Decoy MNIST	1.54s	0.000045s	784
20 Newsgroups	2.59s	0.000520s	5000

Gradient vs. LIME runtimes per explanation. Note that each method uses a different version of LIME; Iris-Cancer and Toy Colors use `lime_tabular` *with continuous and quartile-discrete perturbation methods, respectively, Decoy MNIST uses* `lime_image`*, and 20 Newsgroups uses* `lime_text`*. Code was executed on a laptop and input gradient calculations were not optimized for performance, so runtimes are only meant to provide a sense of scale.*

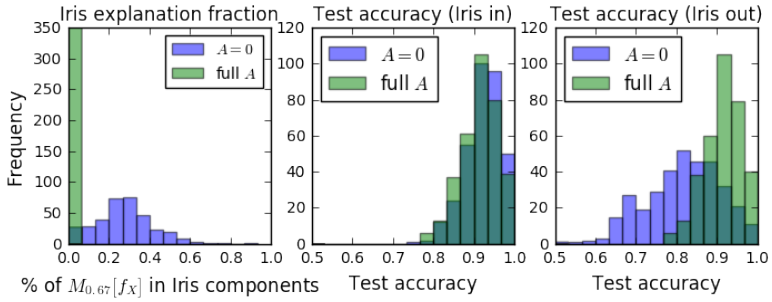


Figure 3.8: *Overcoming confounds using explanation constraints on Iris-Cancer (over 350 random train-test splits). By default ($A = 0$), input gradients tend to be large in Iris dimensions, which results in lower accuracy when Iris is removed from the test set. Models trained with $A_{nd} = 1$ in Iris dimensions (full A) have almost exactly the same test accuracy with and without Iris.*

knowledge. Another key advantage of using an explanation method more closely related to our model is that we can then incorporate explanations into our training process, which are most useful when the model faces ambiguities in how to classify inputs. We deliberately constructed the Decoy MNIST and Iris-Cancer datasets to have this kind of ambiguity, where a rule that works in training will not generalize to test. When we train our network on these confounded datasets, their test accuracy is better than random guessing, in part because the decoy rules are not simple and the primary rules not complex, but their performance is still significantly worse than on a baseline test set with no decoy rules. By penalizing explanations we know to be incorrect using the loss function defined in Section 3.3.1, we are

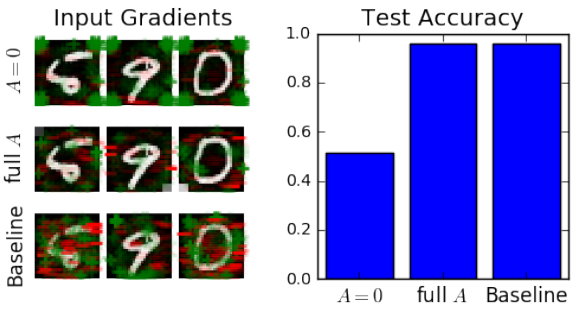


Figure 3.9: *Training with explanation constraints on Decoy MNIST. Accuracy is low ($A = 0$) on the swatch color-randomized test set unless the model is trained with $A_{nd} = 1$ in swatches (full A). In that case, test accuracy matches the same architecture's performance on the standard MNIST dataset (baseline).*

able to recover that baseline test accuracy, which we demonstrate in Figures 3.8 and 3.9.

When annotations are unavailable, our find-another-explanation method discovers diverse classifiers. As we saw with the Toy Color dataset, even if almost every row of A is 0, we can still benefit from explanation regularization (meaning practitioners can gradually incorporate these penalties into their existing models without much upfront investment). However, annotation is never free, and in some cases we either do not know the right explanation or cannot easily encode it. Additionally, we may be interested in exploring the structure of our model and dataset in a less supervised fashion. On real-world datasets, which are usually overdetermined, we can use find-another-explanation to discover θs in shallower local minima that we would normally never explore. Given enough models right for different reasons, hopefully at least one is right for the right reasons.

Figure 3.11 shows find-another-explanation results for our three real-world datasets, with example explanations at each iteration above and model train and test accuracy below. For Iris-Cancer, we find that the initial iteration of the model heavily relies on the Iris features and has high train but low test accuracy, while subsequent iterations have lower train but higher test accuracy (with smaller gradients in Iris components). In other words, we spontaneously obtain a more generalizable model without a predefined A alerting us that the first four features are misleading.

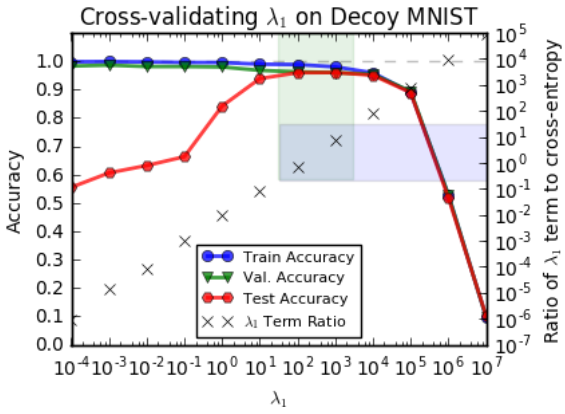


Figure 3.10: *Test accuracy by λ_1 on Decoy MNIST. Because training and validation sets share the same misleading confounds, validation accuracy is a poor proxy for test accuracy. Instead, we find the regime of highest accuracy (highlighted) is where the initial cross-entropy and λ_1 loss terms have similar magnitudes; we recommend choosing λ_1 in this manner. Note that exact equality does not seem to be required; being an order of magnitude off does not significantly affect accuracy.*

Find-another-explanation also overcomes confounds on Decoy MNIST, needing only one iteration to recover baseline accuracy. Bumping λ_1 too high (to the point where its term is a few orders of magnitude larger than the cross-entropy) results in more erratic behavior. Interestingly, in a process remniscent of distillation [174], the gradients themselves become more evenly and intuitively distributed at later iterations. In many cases they indicate that the probabilities of certain digits increase when we brighten pixels along or extend their distinctive strokes, and that they decrease if we fill in unrelated dark areas, which seems desirable. However, by the last iteration, we start to revert to using decoy swatches in some cases.

On 20 Newsgroups, the words most associated with `alt.atheism` and `soc.religion.christian` change between iterations but remain mostly intuitive in their associations. Train accuracy mostly remains high while test accuracy is unstable.

For all of these examples, accuracy remains high even as decision boundaries shift significantly. This may be because real-world data tends to contain significant redundancies.

3.4.3 Limitations

Input gradients provide faithful information about a model's rationale for a prediction but trade interpretability for efficiency. In particular, when input features are not individually meaningful to users (e.g. for individual pixels or word2vec components), input gradients may be difficult to interpret and A may be difficult to specify. Additionally, because they can be 0 far from the decision boundary, they do not capture the idea of salience as well as other methods [241, 222, 162, 17, 209]. However, they are necessarily faithful to the model and easy to incorporate into its loss function. Input gradients are first-order linear approximations of the model; we might call them first-order explanations.

3.5 Conclusion

In this chapter, we showed that:

- On training sets that contain confounds which would fool *any* model trained just to make correct predictions, we can use gradient-based explanation regularization to learn models that still generalize to test. These results imply that gradient regularization actually changes *why* our model makes predictions.
- When we lack expert annotations, we can still use our method in an unsupervised manner to discover models that make predictions for different reasons. This "find-another-explanation" technique allowed us to overcome confounds on Decoy MNIST and Iris-Cancer, and even quantify the ambiguity present in the Toy Color dataset.
- Input gradients are consistent with sample-based methods such as LIME but faster to compute and sometimes more faithful to the model, especially for continuous inputs.

Our consistent results on several diverse datasets show that input gradients merit further investigation as building blocks for optimizable explanations; there exist many options for further advancements such as weighted annotations A, different penalty norms, and more

general specifications of whether features should be positively or negatively predictive of specific classes for specific inputs.

Finally, our "right for the right reasons" approach may be of use in solving related problems, e.g. in integrating causal inference with deep neural networks or maintaining robustness to adversarial examples (which we discuss in Chapter 5). Building on our find-another-explanation results, another promising direction is to let humans in the loop interactively guide models towards correct explanations. Overall, we feel that developing methods of ensuring that models are right for better reasons is essential to overcoming the inherent obstacles to generalization posed by ambiguities in real-world datasets.

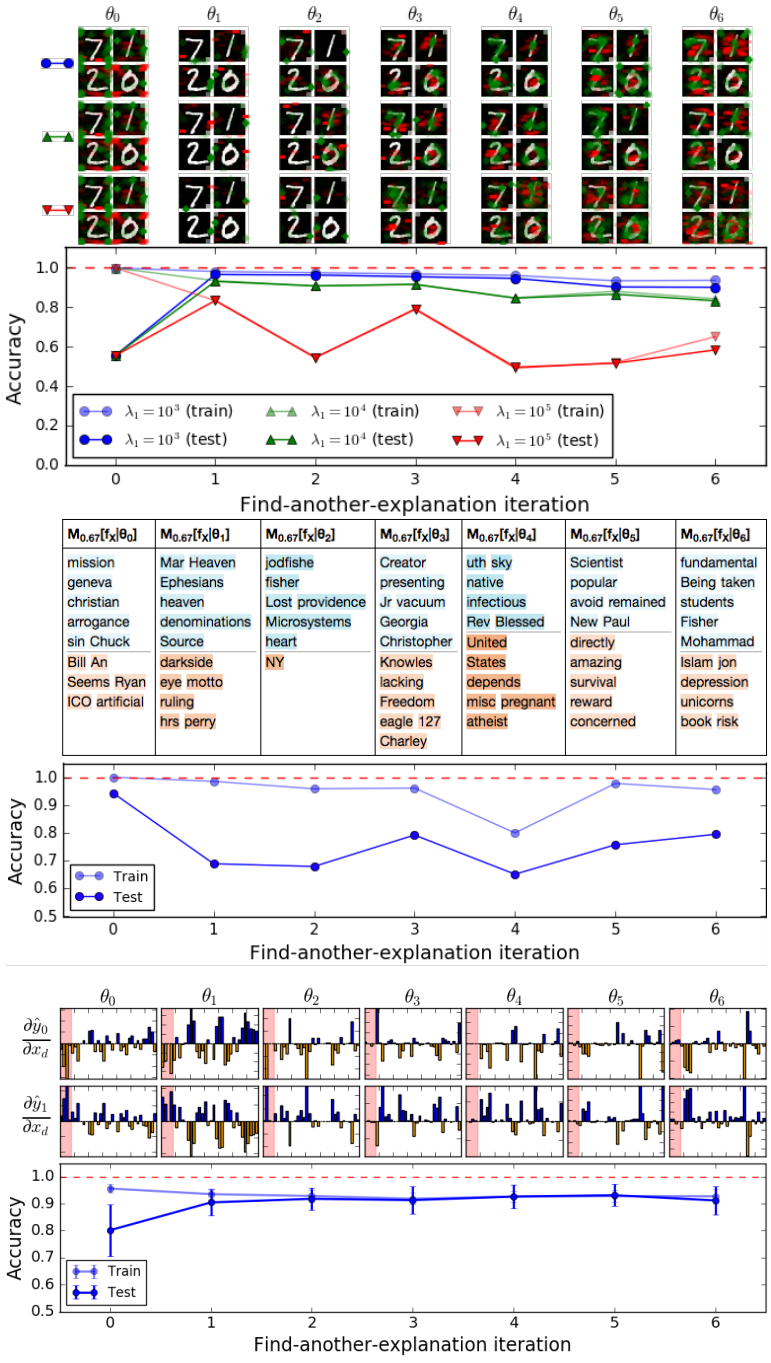


Figure 3.11: *Find-another-explanation results on Iris-Cancer (top; errorbars show standard deviations across 50 trials), 20 Newsgroups (middle; blue supports Christianity and orange supports atheism, word opacity set to magnitude ratio), and Decoy MNIST (bottom, for three values of λ_1 with scatter opacity set to magnitude ratio cubed). Real-world datasets are often highly redundant and allow for diverse models with similar accuracies. On Iris-Cancer and Decoy MNIST, both explanations and accuracy results indicate we overcome confounds after 1-2 iterations without any prior knowledge about them encoded in A.*

Chapter 4

Ensembles of Locally Independent Prediction Models[1]

In Section 3.3.2 of the previous chapter, we introduced "find-another-explanation," an iterative method of training an ensemble of classifiers to make predictions for different reasons by sequentially augmenting A to penalize more features. However, this method has a number of limitations, which this chapter will study and work to overcome.

To illustrate these limitations, let us return to the sports car parable from Chapter 1. In this parable, there are at least two implicit decision rules that could allow for the prediction of whether a passing automobile is a sports car. The correct rule (according to the father) is based on shape, but it is also possible to make a prediction based solely on color (whether the car is red), which is what seems simplest to the son.

In the framework of find-another-explanation, we might hope that an initial model (mimicking the son) would learn to place all of its importance on color, so that a subsequent model, forbidden from considering color, would instead favor shape (assuming for a moment these rules require different features, at least locally). However, it is also possible

[1]This chapter is based on Andrew Ross, Weiwei Pan, and Finale Doshi-Velez. Learning qualitatively diverse and interpretable rules for classification. In *2018 ICML Workshop on Human Interpretability in Machine Learning*, 2018. URL https://arxiv.org/abs/1806.08716 and Andrew Slavin Ross, Weiwei Pan, Leo Anthony Celi, and Finale Doshi-Velez. Ensembles of locally independent prediction models. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1911.01291.

that the initial model might learn to place importance on *both* shape and color. The simplest combination might be a static linear combination of 50% shape and 50% color. If that occurred, find-another-explanation would be helpless, because both implicit decision rules would be represented in the corresponding annotation matrix A_1; subsequent models would either learn the same thing or nothing.

To make matters worse, it could also be the case that the resulting implicit decision rule becomes much less interpretable than either the shape rule or the color rule alone: color could be responsible for 70-90% of the prediction, except on cloudy days, when its contribution might fall to only 10-30%, but not for Toyota sports cars, which have a much more recognizable shape that makes predictions locally invariant to color. Models with such implicit decision rules would be equally accurate on the training set, and it is unclear whether any knowledge-agnostic form of regularization would prevent us from learning them. The very fact that a prediction problem may be solved by multiple interpretable models may preclude us from learning any of them.

The method we present in this Chapter is to *simultaneously* learn a diverse ensemble of models that are encouraged to make the same predictions but extrapolate in maximally different ways. We find that this can sometimes help us learn ensembles whose members are individually more interpretable than any model we could learn on its own, which was our main motivation for developing the method. However, we also spend significant time in this Chapter focusing on more traditional metrics for evaluating ensemble methods, such as predictive accuracy, robustness to distribution shifts, and the correlation of errors on new data, as we find our method to be helpful in this sense as well.

4.1 Introduction

An ensemble is generally more accurate than its constituent models. However, for this to hold true, those models must make different errors on unseen data [86, 58]. This is often described as the ensemble's "diversity."

Despite diversity's well-recognized importance, there is no firm consensus on how best

to foster it. Some procedures encourage it implicitly, e.g. by training models with different inputs [30], while others explicitly optimize for proxies [144] that tend to be functions of differences in training set predictions [125, 32].

However, there has been increasing criticism of supervised machine learning for focusing too exclusively on cases where training and testing data are drawn from the same distribution [139]. In many real-world settings, this assumption does not hold, e.g. due to natural covariate shift over time [179] or selection bias in data collection [239]. Intuitively, we might hope that a "diverse" ensemble would more easily adapt to such problems, since ideally different members would be robust to different shifts. In this paper, however, we find that diverse ensemble methods that only encourage differences in training predictions often perform poorly under mild drift between training and test, in large part because models are not incentivized to make different predictions where there is no data. We also find that ensemble methods that directly optimize for diverse training predictions face inherent tradeoffs between diversity and accuracy and can be very sensitive to hyperparameters.

To resolve these issues, we make two main contributions, specifically (1) a novel and differentiable diversity measure, defined as a formal proxy for the ability of classifiers to *extrapolate* differently away from data, and (2) a method for training an ensemble of classifiers to be diverse by this measure, which we hypothesize will lead to more robust predictions under distributional shifts with no inherent tradeoffs between diversity and accuracy except those imposed by the dataset. We find this hypothesis holds on a range of synthetic and real-world prediction tasks.

4.2 Related Work

Ensembling is a well-established subfield of supervised learning [30, 31, 96, 196], and one of its important lessons is that model diversity is a necessary condition for creating predictive and robust ensembles [120]. There are a number of methods for fostering diversity, which can be roughly divided into two categories: those that implicitly promote diversity by random modifications to training conditions, and those that explicitly promote it by

deliberate modifications to the objective function.

Some implicit diversity methods operate by introducing stochasticity into which models see which parts of the data, e.g. by randomly resampling training examples [30] or subsets of input features [31]. Others exploit model parameter stochasticity, e.g. by retraining from different initializations [117] or sampling from parameter snapshots saved during individual training cycles [99].

Methods that explicitly encourage diversity include boosting [196, 74], which sequentially modifies the objective function of each model to specialize on previous models' mistakes, or methods like negative correlation learning [144] amended cross-entropy [208], and DPPs over non-maximal predictions [171], which simulateously train models with penalities on both individual errors and pairwise similarities. Finally, methods such as Diverse Ensemble Evolution [244] and Competition of Experts [176] use explicit techniques to encourage models to specialize in different regions of input space.

Although at first glance these diverse training techniques seem quite diverse themselves, they are all similar in a crucial respect: they encourage diversity in terms of training set predictions. In the machine learning fairness, adversarial robustness, and explainability communities, however, there has been increasing movement away from the assumption that train is similar to test. For example, many methods for locally explaining ML predictions literally present simplified approximations of how models extrapolate away from given points [19, 183, 190], while adversarial attacks (and defenses) exploit (and mitigate) pathological extrapolation behavior [224, 152], sometimes in an ensemble setting [226]. Although our focus is not explicitly on explanability or adversarial robustness, our method can be seen as a reapplication of techniques in those subfields to the problem of ensemble diversity.

Also related is the subfield of streaming data, which sometimes uses ensemble diversity metrics as a criteria for deciding when covariates have shifted sufficiently to warrant retraining [33, 119]. Although our focus remains on non-streaming classification, the method we introduce may be applicable to that domain.

4.3 Method

In this section, building on Ross et al. [2018], we define our diversity measure and training procedure, beginning with notation. We use x to denote D-dimensional inputs, which are supported over an input space $\Omega_x \subseteq \mathbb{R}^D$. We use y to denote prediction targets in an output space Ω_y. In this paper, Ω_y will be $\mathbb{R}$, and we focus on the case where it represents a log-odds used for binary classification, but our method can be generalized to classification or regression in $\mathbb{R}^K$ given any notion of distance between outputs. We seek to learn prediction models $f(\cdot;\theta) : \Omega_x \to \Omega_y$ (parameterized by θ) that estimate y from x. We assume these models f are differentiable with respect to x and θ (which is true for linear models and neural networks).

In addition, we suppose a joint distribution over inputs and targets $p(x, y)$ and a distribution $p(y|f(x;\theta))$ quantifying the likelihood of the observed target given the model prediction. Typically, during training, we seek model parameters that maximize the likelihood of the observed data, $\mathbb{E}_{p(x,y)}\left[\log p(y|f(x;\theta))\right]$.

4.3.1 Diversity Measure: Local Independence

We now introduce a model diversity measure that quantifies how differently two models generalize over small patches of the data manifold Ω_x. Formally, we define an ϵ-neighborhood of x, denoted $N_\epsilon(x)$, on the data manifold to be the intersection of an ϵ-ball centered at x in the input space, $\mathcal{B}_\epsilon(x) \subset \mathbb{R}^D$, and the data manifold: $N_\epsilon(x) = \mathcal{B}_\epsilon(x) \cap \Omega$. We capture the notion of generalization difference on a small neighborhood of x through an intuitive geometric condition: we say that two functions f and g generalize maximally differently at x if f is invariant in the direction of of the greatest change in g (or vice versa) within an ϵ-neighborhood around x. That is:

$$f(x) = f\left(x_{g_{\max}}\right), \quad \text{for all } \epsilon' < \epsilon, \tag{4.1}$$

where we define $x_{g_{\max}} = \underset{x' \in N_{\epsilon'}(x)}{\arg\max}\, g(x')$. In other words, perturbing x by small amounts to increase g inside N_ϵ does not change the value of f. In the case that a choice of ϵ exists to

satisfy Equation 4.1, we say that f is *locally independent at* x. We call f and g *locally independent* without qualification if for every $x \in \Omega_x$ the functions f and g are locally independent at x for some choice of ϵ. We note that in order for the right-hand side expression of 4.1 to be well-defined, we assume that the gradient of g is not zero at x and that ϵ is chosen to be small enough that g is convex or concave over $N_\epsilon(x)$.

In the case that f and g are classifiers, local independence intuitively implies a kind of dissimilarity between their decision boundaries. For example, if f and g are linear and the data manifold is Euclidean, then f and g are locally independent if and only if their decision boundaries are orthogonal.

This definition motivates the formulation of a diversity measure, $\texttt{IndepErr}(f, g)$, quantifying how far f and g are from being locally independent:

$$\texttt{IndepErr}(f, g) \equiv \mathbb{E}\left[\left(f\left(x_{g_{\max}}\right) - f\left(x\right)\right)^2\right]. \tag{4.2}$$

4.3.2 Local Independence Training (LIT)

Using Equation 4.2, we can formulate an ensemble-wide loss function $\mathcal{L}$ for a set of models $\{\theta_m\}$ as follows, which we call local independence training:

$$\mathcal{L}(\{\theta_m\}) = \sum_m \mathbb{E}_{p(x,y)}\left[-\log p(y|f(x;\theta_m))\right] + \lambda \sum_{\ell \neq m} \texttt{IndepErr}(f(\cdot;\theta_m), f(\cdot;\theta_\ell)). \tag{4.3}$$

The first term encourages each model f_m to be predictive and the second encourages diversity in terms of IndepErr (with a strength hyperparameter λ). Computing IndepErr exactly, however, is challenging, because it requires an inner optimization of g. Although it can be closely approximated for fixed small ϵ with projected gradient descent as in adversarial training [152], that procedure is computationally intensive. If we let $\epsilon \to 0$, however, we can approximate $x_{g_{\max}}$ by a fairly simple equation that only needs to compute ∇g once per x. In particular, we observe that under certain smoothness assumptions on g, with unconstrained

Ω_x,[2] and as $\epsilon \to 0$, we can make the approximation

$$x_{g_{\max}} \approx x + \epsilon \nabla g(x). \tag{4.4}$$

Assuming similar smoothness assumptions on f (so we can replace it by its first-order Taylor expansion), we see that

$$\begin{aligned} f(x_{g_{\max}}) - f(x) &\approx f(x + \epsilon \nabla g(x)) - f(x) \\ &= \left[f(x) + \epsilon \nabla f(x) \cdot \nabla g(x) + \mathcal{O}(\epsilon^2)\right] - f(x) \\ &\approx \epsilon \nabla f(x) \cdot \nabla g(x). \end{aligned} \tag{4.5}$$

In other words, the independence error between f and g is approximately equal to the dot product of their gradients $\nabla f(x) \cdot \nabla g(x)$. Empirically, we find it helpful to normalize the dot product and work in terms of cosine similarity $\cos(\nabla f(x), \nabla g(x)) \equiv \frac{\nabla f(x) \cdot \nabla g(x)}{||\nabla f(x)||_2 ||\nabla g(x)||_2} \in [-1, 1]$. We also add a small constant value to the denominator to prevent underflow.

Alternate statistical formulation: Our local independence penalty can also be interpreted as a way of enforcing statistical independence between *changes* in each function when we add infinitesimal Gaussian perturbations to our data points. More formally, let $x \in \mathbb{R}^D$ be a data point, let f_1 and f_2 be functions from $\mathbb{R}^D \to \mathbb{R}$, and let $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbb{1})$ be a small perturbation. Taylor expanding f_1 and f_2 around x and assuming all gradients are nonzero,

$$f_i(x + \boldsymbol{\epsilon}) \approx f_i(x) + \nabla f_i(x) \cdot \boldsymbol{\epsilon} + \mathcal{O}(\boldsymbol{\epsilon} \cdot \boldsymbol{\epsilon})$$

Defining the changes in each f_i under the perturbation of x to be $\Delta f_i(x)$, and assuming that all gradients are nonzero at x and that we select σ^2 to be small enough that $||\boldsymbol{\epsilon}||^2 \ll ||\boldsymbol{\epsilon}||$ with high probability, we have

$$\Delta f_i(x) \equiv f_i(x + \boldsymbol{\epsilon}) - f_i(x) \approx \nabla f_i(x) \cdot \boldsymbol{\epsilon}$$

[2]The simplifying assumption that $N_\epsilon(x) \approx \mathcal{B}_\epsilon(x)$ in a local neighborhood around x is significant, though not always inappropriate. We discuss both limitations and generalizations in Section B.1.

If we now consider $\text{Cov}(\Delta f_1(x), \Delta f_2(x))$, the covariance of the changes in each function (which we term the "local covariance" between f_1 and f_2 at x), we see that

$$\begin{aligned}
&\text{Cov}(\Delta f_1(x), \Delta f_2(x)) \\
&\quad \approx \mathbb{E}[(\nabla f_1(x) \cdot \boldsymbol{\epsilon} - \mathbb{E}[\nabla f_1(x) \cdot \boldsymbol{\epsilon}]) \\
&\qquad (\nabla f_2(x) \cdot \boldsymbol{\epsilon} - \mathbb{E}[\nabla f_2(x) \cdot \boldsymbol{\epsilon}])] \\
&\quad = \mathbb{E}[(\nabla f_1(x) \cdot \boldsymbol{\epsilon})(\nabla f_2(x) \cdot \boldsymbol{\epsilon})] \\
&\quad = \mathbb{E}[\boldsymbol{\epsilon} \cdot \boldsymbol{\epsilon}] \nabla f_1(x) \cdot \nabla f_2(x) \\
&\quad = \sigma^2 \nabla f_1(x) \cdot \nabla f_2(x)
\end{aligned}$$

Normalizing this quantity, we see the local correlation is

$$\begin{aligned}
\text{Corr}(\Delta f_1(x), \Delta f_2(x)) &= \frac{\nabla f_1(x) \cdot \nabla f_2(x)}{||\nabla f_1(x)||_2 ||\nabla f_2(x)||_2} \\
&= \cos(\nabla f_1(x), \nabla f_2(x)).
\end{aligned}$$

Finally, let's consider the distribution of $\Delta f_i(x)$. Since $\Delta f_i(x)$ is approximately equal to $\nabla f_i(x) \cdot \boldsymbol{\epsilon}$, which is a dot product between a deterministic vector ($\nabla f_i(x)$) and a Gaussian sample ($\boldsymbol{\epsilon}$), then $\Delta f_1(x)$ and $\Delta f_2(x)$ are approximately equal to sums of Gaussian random variables and are therefore themselves Gaussian. Noting that the entropy of Gaussians with covariance matrices Σ is $H(X) = \frac{1}{2} \ln(2\pi e|\Sigma|)$ and for a bivariate Gaussian random variable (X, Y), its covariance determinant $|\Sigma| = \text{Var}(X)\text{Var}(Y) - \text{Cov}(X, Y)^2$,

$$\begin{aligned}
I(X, Y) &= H(X) + H(Y) - H(X, Y) \\
&= \frac{1}{2} \ln \left(\frac{\text{Var}(X)\text{Var}(Y)}{\text{Var}(X)\text{Var}(Y) - \text{Cov}(X, Y)^2} \right) \\
&= -\frac{1}{2} \ln \left(1 - \frac{\text{Cov}(X, Y)^2}{\text{Var}(X)\text{Var}(Y)} \right) \\
&= -\frac{1}{2} \ln \left(1 - \text{Corr}(X, Y)^2 \right).
\end{aligned}$$

So, between two 1D Gaussians, zero correlation implies statistical independence. Therefore, making the input gradients of f_1 and f_2 orthogonal is equivalent to enforcing statistical independence between their outputs when we perturb x with samples from $\mathcal{N}(0, \sigma^2 \mathbb{1})$

as $\sigma \to 0$ (assuming the gradients are nonzero, which in general will be true except at individual points). This could be used as an alternate definition of "local independence."

Final LIT objective term: Motivated by the approximations and discussion above, we substitute

$$\texttt{CosIndepErr}(f, g) \equiv \mathbb{E}\left[\cos^2(\nabla f(x), \nabla g(x))\right] \tag{4.6}$$

into our ensemble loss from Equation (4.3), which gives us a final loss function

$$\mathcal{L}(\{\theta_m\}) = \sum_m \mathbb{E}_{p(x,y)}\left[-\log p(y|f(x;\theta_m))\right] + \lambda \sum_{\ell \neq m} \mathbb{E}_{p(x)}\left[\cos^2(\nabla f(x;\theta_m), \nabla f(x;\theta_\ell))\right] \tag{4.7}$$

Note that we will sometimes abbreviate `CosIndepErr` as $\nabla_{\cos^2}$. In Section 4.4 as well as Figure B.5, we show that `CosIndepErr` is meaningfully correlated with other diversity measures and therefore may be useful in its own right, independently of its use within a loss function.

In relation to "right for the right reasons" loss of Chapter 3, we can also understand this loss as encouraging each network to the right predictions but for maximally different reasons. It differs from the "find-another-explanation" method of Section 3.3.2 however, in that models are trained jointly, and that models can use the same sets of features as long as gradients are orthogonal. This allows LIT to achieve diversity on a much wider set of cases than find-another-explanation.

4.4 Experiments

On synthetic data, we show that ensembles trained with LIT exhibit more diversity in extrapolation behavior. On a range of benchmark datasets, we show that the extrapolation diversity in LIT ensembles corresponds to improved predictive performance on test data that are distributed differently than train data. Finally, in a medical data case study, we show that models in LIT ensembles correspond to qualitatively different and clinically meaningful explanations of the data.

Training: For the experiments that follow, we use 256-unit single hidden layer fully connected neural networks with rectifier activations, trained in Tensorflow with Adam. For the real-data experiments, we use dropout and L2 weight decay with a penalty of 0.0001. Code to replicate all experiments is available at `https://github.com/dtak/lit`.

Baselines: We test local independence training ("LIT") against random restarts ("RRs"), bagging [30] ("Bag"), AdaBoost [89] ("Ada"), 0-1 squared loss negative correlation learning Liu and Yao [1999] ("NCL"), and amended cross-entropy [208] ("ACE"). We omit Zhou et al. [2018] and Parascandolo et al. [2018] which require more complex inner submodular or adversarial optimization steps, but note that because they also operationalize diversity as making different errors on training points, we expect the results to be qualitatively similar to ACE and NCL.

Hyperparameters: For our non-synthetic results, we test all methods with ensemble sizes in $\{2, 3, 5, 8, 13\}$, and all methods with regularization parameters λ (LIT, ACE, and NCL) with 16 logarithmically spaced values between 10^{-4} and 10^{1}, using validation AUC to select the best performing model (except when examining how results vary with λ or size). For each hyperparameter setting and method, we run 10 full random restarts (though within each restart, different methods are tested against the same split), and present mean results with standard deviation errorbars.

4.4.1 2D Conceptual Demonstrations

To provide an initial demonstration of our method and the limitations of training set prediction diversity, we present several sets of 2D synthetic examples in Figure 4.1. These 2D examples are constructed to have data distributions that satisfy our assumption that $N_\epsilon(x) \approx \mathcal{B}_\epsilon(x)$ locally around almost all of the points, but nevertheless contain significant gaps. These gaps result in the possibility of learning multiple classifiers that have perfect accuracy on the training set but behave differently when extrapolating. Indeed, in all of these examples, if we have just two classifiers, they can completely agree on training and

completely disagree in the extrapolation regions.

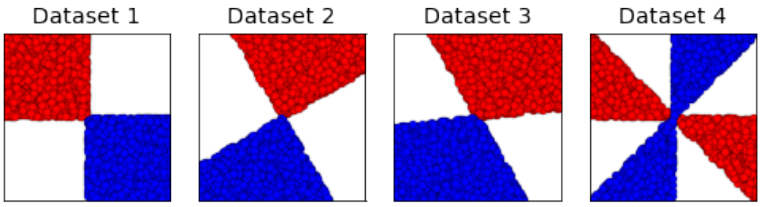


Figure 4.1: *2D synthetic datasets with gaps. We argue that "diverse" ensemble methods applied to these datasets should produce accurate models with different decision boundaries.*

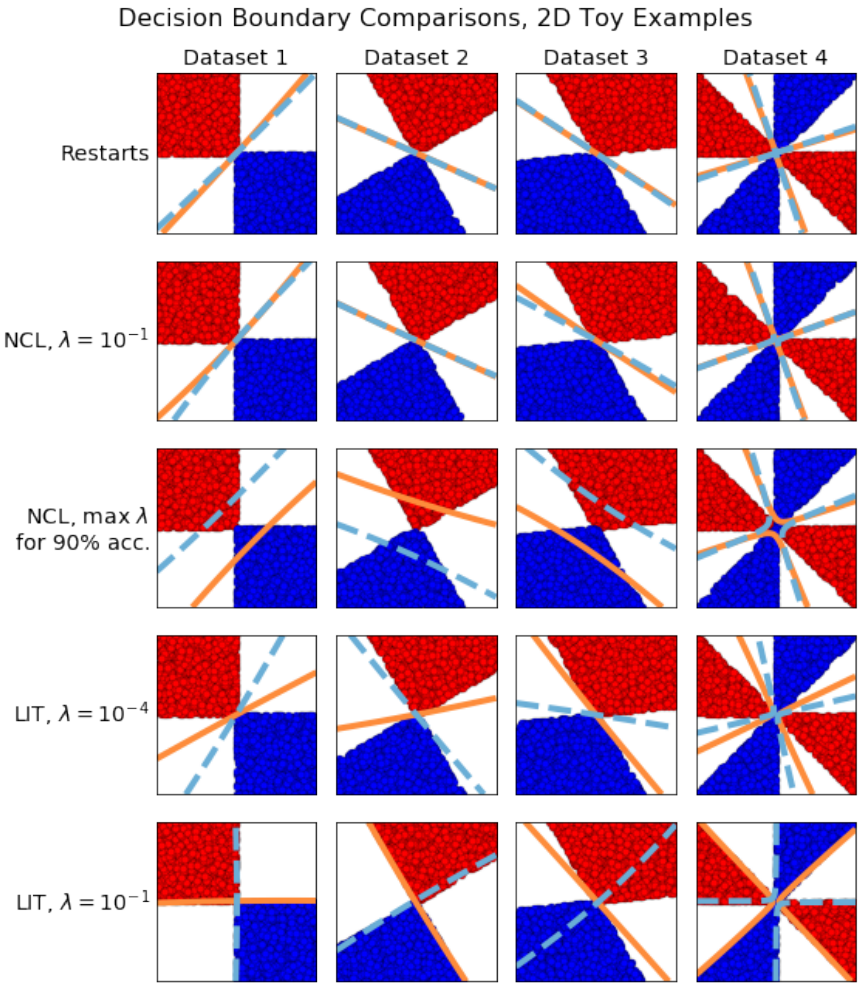


Figure 4.2: *Comparison of local independence training, random restarts and NCL on toy 2D datasets. For each ensemble, the first model's decision boundary is plotted in orange and the other in dashed blue. Both NCL and LIT are capable of producing variation, but in qualitatively different ways.*

In Figure 4.2, we compare the neural network decision boundaries learned by random restarts, local independence training, and negative correlation learning (NCL) on these examples (we use NCL as a state-of-the-art example of an approach that defines diversity with respect to training predictions). Starting with the top and bottom two rows (random restarts and LIT), we find that random restarts give us essentially identical models, whereas LIT outputs models with meaningfully different decision boundaries even at values of λ that are very low compared to its prediction loss term. This is in large part because on most of these tasks (except Dataset 3), there is very little tradeoff to learning a near-orthogonal boundary. At larger λ, LIT outputs decision boundaries that are completely orthogonal (at the cost of a slight accuracy reduction on Dataset 3).

NCL had more complicated behavior, in large part because of its built-in tradeoff between accuracy and diversity. At low values of λ (second from top), we found that NCL produced models with identical decision boundaries, suggesting that training ignored the diversity term. At $\lambda \geq 2$, the predictive performance of one model fell to random guessing, suggesting that training ignored the accuracy term. So in order to obtain meaningfully diverse but accurate NCL models, we iteratively searched for the highest value of λ at which NCL would still return two models at least 90% accurate on the training set (by exponentially shrinking a window between $\lambda = 1$ and $\lambda = 2$ for 10 iterations). What we found (middle row) is that NCL learned to translate its decision boundaries within the support of the training data (incurring an initially modest accuracy cost due to the geometry of the problem) but not modify them outside the training support. Although this kind of diversity is not necessarily bad (since the *ensemble* accuracy remains perfect), it is qualitatively different from the kind of diversity encouraged by LIT—and only emerges at carefully chosen hyperparameter values. The main takeaway from this set of synthetic examples is that methods that encourage diverse extrapolation (like LIT) can produce significantly different ensembles than methods that encourage diverse prediction (like NCL).

Note that the find-another-explanation of Section 3.3.2 would not work for any of these conceptual examples; for Dataset 1, the initial model (i.e. either random restart model)

learns to assign high importance to both features, so the second model is trained with an A matrix of all ones (as in our sports car analogy at the beginning of this chapter; both possible rules are equally important). On the remaining datasets, find-another-explanation would be limited by its restriction that models must use different features, which does not allow for the recovery of diverse decision boundaries that are not axis-aligned.

4.4.2 8D Feature Selection Example

Hearkening back to the sports car parable we discussed at the beginning of this chapter, in Figure 4.3 we introduce an ambiguous 8-dimensional classification task where four non-overlapping combinations of two features could separately be used to predict the class label on the training set (where $p(x)$ is a uniform distribution over the subset of the 8D hypercube $[-20, 20]^8$ where all four of those 2D functions can predict the right label simultaneously). Our goal is both to see if LIT can recover these four 2D functions, and to investigate what normally trained models (of varying model classes) learn.

To investigate these questions, we evaluate each model's predictions over a test set with $p(x)$ set to a uniform distribution across the entirety of the 8D hypercube (which spans a much larger volume). We compute both the average prediction projected down to a 2D grid over each 2D subspace (Figure 4.4), as well as the accuracy of each model's predictions if each of the four different 2D functions were used to define the test set class labels (Test 1-4, Table 4.1). If a model has learned only one of the four 2D functions, then exactly one of its 2D test projections should look similar to the corresponding 2D train projections with the rest random, and exactly one of its test set accuracies should be near 1.0 while the rest should be near 0.5. If a model has learned a combination, then results should be mixed.

In general, we find that individually trained models learn a fairly complex combination of all functions, and that this phenomenon holds true even for "interpretable" model classes like logistic regression and decision trees. These results represent something like the worst-case scenario we outlined in our discussion of the sports car story: the presence of multiple simple implicit decision rules makes the actual function we learn much more

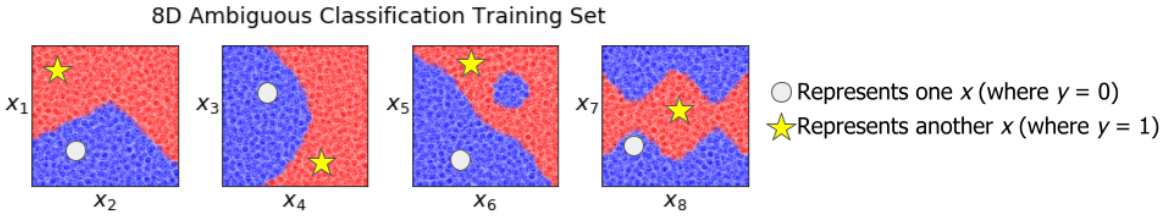


Figure 4.3: *Ambiguous classification task in 8 dimensions (scatterplots show data projected down to different 2D slices). On the training set, data is limited to a small subset of an 8-dimensional hypercube where any of four possible 2D projections ($x_{1,2}$, $x_{3,4}$, $x_{5,6}$, or $x_{7,8}$) redundantly predicts the class label. However, models will be evaluated over the entirety of the hypercube.*

Table 4.1: *Classification results on the 8D ambiguous dataset.*

Model	Train	Test 1	Test 2	Test 3	Test 4
Logistic Regression	0.99	0.70	0.75	0.61	0.49
Decision Tree	1.00	0.57	0.77	0.54	0.57
Random Forest	1.00	0.67	0.72	0.56	0.50
Support Vector Machine	1.00	0.47	0.55	0.59	0.65
Neural Network	1.00	0.76	0.70	0.58	0.54
LIT Model 1	1.00	1.00	0.50	0.49	0.48
LIT Model 2	1.00	0.50	0.99	0.51	0.50
LIT Model 3	1.00	0.50	0.51	0.98	0.51
LIT Model 4	1.00	0.50	0.52	0.52	0.97

8D ambiguous classification accuracy results on task's four test sets for five model classes, plus four-model LIT. Suboptimal accuracies on all test sets (first five rows) imply these models learn a dense combination of almost all functions, even for "interpretable" methods such as logistic regression and decision trees. LIT models, on the other hand, are able to recover each function individually.

complicated. However, we see that local independence training recovers these underlying simple functions, in this case by allocating each model separate features.

4.4.3 Classification Benchmarks

Next, we test our method on several standard binary classification datasets from the UCI and MOA repositories [140, 28]. These are `mushroom`, `ionosphere`, `sonar`, `spectf`, and `electricity` (with categorical features one-hot encoded, and all features z-scored). For all datasets, we randomly select 80% of the dataset for training and 20% for test, then take an additional 20% split of the training set to use for validation. In addition to random splits,

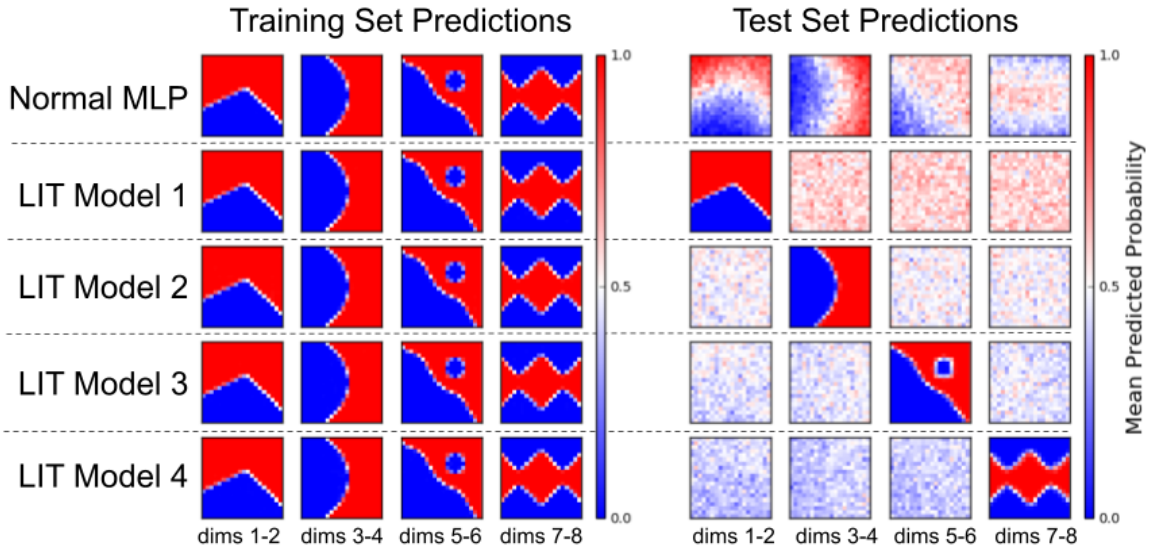


Figure 4.4: *8D ambiguous classification results for normal MLPs (top row) vs. LIT models (bottom four rows). In this case, local independence training outputs models sensitive to disjoint sets of features, which are each arguably simpler to understand than the 8D function learned by a normally trained neural network.*

Table 4.2: *Ensemble results on UCI benchmarks.*

Random Split

Ens.	Mushroom			Ionosphere			Sonar			SPECTF			Electricity		
	AUC	ρ_{av}	$\nabla_{\cos^2}$	AUC	ρ_{av}	$\nabla_{\cos^2}$	AUC	ρ_{av}	$\nabla_{\cos^2}$	AUC	ρ_{av}	$\nabla_{\cos^2}$	AUC	ρ_{av}	$\nabla_{\cos^2}$
RRs	**1.0**	1±.1	.9±0	.95±.03	.9±.1	1±0	.91±.06	.9±.1	1±0	**.80**±.06	.9±.1	1±0	.87±.00	1±0	1±0
Bag	**1.0**	1±0	.9±0	.96±.02	.7±.1	.5±.1	.90±.06	.5±.2	.5±.1	**.80**±.05	.6±.1	.4±.1	.87±.00	.9±0	1±0
Ada	**1.0**	—	—	.95±.03	—	—	.91±.06	—	—	**.80**±.06	—	—	**.88**±.00	.2±0	.2±.1
NCL	**1.0**	1±0	.8±0	.96±.04	.6±.5	.7±.3	**.91**±.06	.6±.5	.7±.4	**.80**±.07	.6±.5	.7±.3	.87±.00	.4±.1	.6±0
ACE	**1.0**	1±0	.9±0	.94±.04	.8±.3	.9±.2	.90±.06	.9±.2	1±.1	**.79**±.06	.8±.4	.9±.2	.87±.00	.9±0	1±0
LIT	**1.0**	.9±.1	0±0	**.98**±.01	.3±.1	0±0	**.92**±.05	.5±.2	0±0	**.81**±.06	.4±.1	0±0	.87±.00	.9±0	.3±.1

Extrapolation Split

Ens.	Mushroom			Ionosphere			Sonar			SPECTF			Electricity		
	AUC	ρ_{av}	$\nabla_{\cos^2}$	AUC	ρ_{av}	$\nabla_{\cos^2}$	AUC	ρ_{av}	$\nabla_{\cos^2}$	AUC	ρ_{av}	$\nabla_{\cos^2}$	AUC	ρ_{av}	$\nabla_{\cos^2}$
RRs	.92±.00	.9±0	.8±0	.87±.02	1±0	1±0	.81±.02	1±0	1±0	**.83**±.05	1±0	1±0	.86±.00	1±0	1±0
Bag	.91±.00	.9±0	.9±0	.89±.04	.6±.1	.5±.1	**.82**±.03	.7±.1	.6±0	**.83**±.05	.6±.1	.4±0	.86±.00	.9±0	.9±0
Ada	.92±.01	—	—	.87±.02	—	—	.81±.03	—	—	**.83**±.05	—	—	.86±.00	.3±.1	.3±.2
NCL	.94±.01	.6±.2	.6±.1	.90±.02	.8±.3	.9±.2	.78±.06	.5±.5	.6±.3	.81±.12	.5±.6	.7±.3	.86±.00	.9±.2	1±.1
ACE	.92±.00	.9±0	.8±0	.90±.03	.3±.4	.5±.3	.77±.06	.6±.5	.7±.3	.72±.16	.5±.6	.7±.4	.86±.00	1±0	1±0
LIT	**.96**±.01	.3±.1	0±0	**.96**±.02	.2±.1	0±0	**.81**±.03	.5±.1	0±0	**.84**±.05	.4±.1	0±0	**.87**±.00	.4±.2	0±0

Benchmark classification results in both the normal prediction task (top) and the extrapolation task (bottom) over 10 reruns, with errorbars based on standard deviations and bolding based on standard error overlap. On random splits, LIT offers modest AUC improvements over random restarts, on par with other ensemble methods. On extrapolation splits, however, LIT tends to achieve higher AUC. In both cases, LIT almost always exhibits low pairwise Pearson correlation between heldout model errors (ρ_{av}), and for other methods, ρ_{av} roughly matches pairwise gradient cosine similarity ($\nabla_{\cos^2}$).

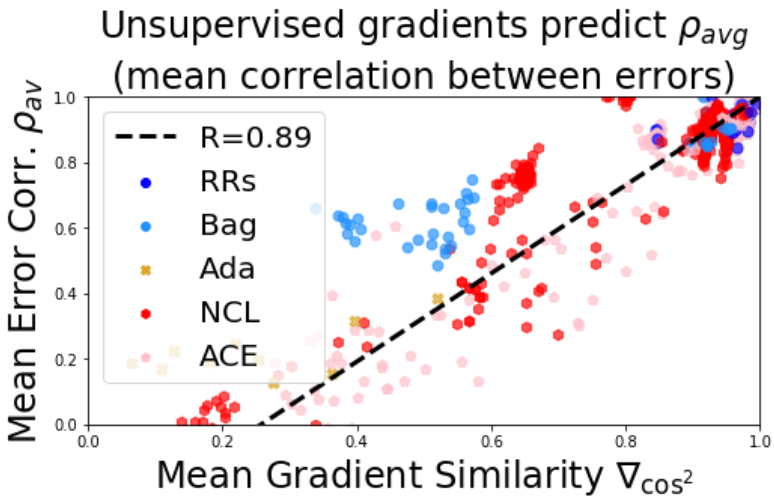


Figure 4.5: *For models not trained to minimize it, the average gradient cosine similarity $\nabla_{\cos^2}$ actually predicts ρ_{av}, the correlation of model errors on the test set, even though $\nabla_{\cos^2}$ is calculated without looking at the test set labels (results across all UCI datasets and many regularization strengths).*

we also introduce an *extrapolation* task, where instead of splitting datasets randomly, we train on the 50% of points closest to the origin (i.e. where $||x||_2$ is less than its median value) and validate/test on the remaining points (which are furthest from the origin). This test is meant to evaluate robustness to covariate shift.

For each ensemble, we measure heldout AUC and accuracy, our diversity metric `CosIndepErr` (abbreviated as $\nabla_{\cos^2}$), and several classic diversity metrics (ρ_{av}, Q_{av}, and κ) defined by Kuncheva and Whitaker [2003]. Table 4.2 compares heldout AUC, ρ_{av}, and $\nabla_{\cos^2}$ after cross-validating λ and the ensemble size. More complete enumerations of AUC, accuracy, and diversity metrics are shown in Figures B.4 and B.5. In general, we find that LIT is competitive on random splits, strongest on extrapolation, and significantly improves heldout prediction diversity across the board. We also find that $\nabla_{\cos^2}$ is meaningfully related to other diversity metrics for all models that do not optimize for it (Figure 4.5).

4.4.4 ICU Mortality Case Study

As a final set of experiments, we run a more in-depth case study on a real world clinical application. In particular, we predict in-hospital mortality for a cohort of $n = 1,053,490$

patient visits extracted from the MIMIC-III database [103] based on on labs, vital signs, and basic demographics. We follow the same cohort selection and feature selection process as Ghassemi et al. [2017]. In addition to this full cohort, we also test on a limited data task where we restrict the size of the training set to $n = 1000$ to measure robustness.

We visualize the results of these experiments in several ways to help tease out the effects of λ, ensemble size, and dataset size on individual and ensemble predictive performance, diversity, and model explanations. Table 4.3 shows overall performance and diversity metrics for these two tasks after cross-validation, along with the most common values of λ and ensemble size selected for each method. Drilling into the $n = 1000$ results, Figure 4.6 visualizes how multiple metrics for performance (AUC and accuracy) and diversity (ρ_{av} and $\nabla_{\cos^2}$) change with λ, while Figure 4.7 visualizes the relationship between optimal λ and ensemble size.

Figure 4.8 (as well as Figures B.2 and B.3) visualize changes in the marginal distributions of input gradients for each model in their explanatory sense [19]. As a qualitative evaluation, we discussed these explanation differences with two intensive care unit clinicians and found that LIT revealed meaningful redundancies in which combinations of features encoded different underlying conditions.

4.5 Discussion

LIT matches or outperforms other methods, especially under data limits or covariate shift. On the UCI datasets under train $\overset{d}{\approx}$ test conditions (random splits), LIT always offers at least modest improvements over random restarts, and often outperforms other baselines. Under extrapolation splits, LIT tends to do significantly better. This pattern repeats itself on the normal vs. data-limited versions of ICU mortality prediction task. We hypothesize that on small or selectively restricted datasets, there is typically more predictive ambiguity, which hurts the generalization of normally trained ensembles (who consistently make similar guesses on unseen data). LIT is more robust to these issues.

Table 4.3: *Ensemble results on ICU mortality prediction tasks.*

ICU Mortality Task, Full Dataset ($n > 10^6$)

Method	AUC	ρ_{av}	$\nabla_{\cos^2}$	#	λ
RRs	.750±.000	.9±0	.9±0	13	—
Bag	.751±.000	.9±0	.9±0	8	—
Ada	**.752**±.003	0±0	0±0	8	—
ACE	.750±.000	.9±0	.9±0	13	$10^{0.33}$
NCL	**.753**±.001	.3±.2	.2±.2	13	$10^{0.00}$
LIT	.750±.001	.8±0	.3±0	3	$10^{-4.00}$

ICU Mortality Task, Limited Slice ($n = 10^3$)

Method	AUC	ρ_{av}	$\nabla_{\cos^2}$	#	λ
RRs	.684±.001	.8±0	.8±0	8	—
Bag	.690±.002	.5±0	.3±0	8	—
Ada	.678±.003	.6±0	.5±0	2	—
ACE	.684±.001	.8±0	.8±0	2	$10^{-2.67}$
NCL	.697±.006	.2±.4	.6±.2	13	$10^{0.33}$
LIT	**.711**±.001	.1±0	0±0	13	$10^{-2.33}$

Quantitative results on the ICU mortality prediction task, where # and λ signify the most commonly selected values of ensemble size and regularization parameter chosen for each method. On the full data task, although all methods perform similarly, NCL and AdaBoost edge out slightly, and LIT consistently selects its weakest regularization parameter. On the limited data task, LIT significantly outperforms baselines, with NCL and Bagging in second, ACE indistinguishable from restarts, and significantly worse performance for AdaBoost (which overfits).

Gradient cosine similarity can be a meaningful diversity metric. In Table 4.2, Figure 4.5, as well as our more complete results in Figure B.5, we saw that for non-LIT methods, gradient similarity $\nabla_{\cos^2}$ (which does not require labels to compute) was often similar in value to error correlation ρ_{av} (as well as the interrater agreement κ, or Yule's Q-statistic Q_{av} after a monotonic transformation—all measures which *do* require labels to compute). One potential explanation for this correspondence is that, by our analysis at the end of Section 4.3.2, $\nabla_{\cos^2}$ can literally be interpreted as an average squared correlation (between changes in model predictions over infinitesimal Gaussian perturbations away from each input). We hypothesize that $\nabla_{\cos^2}$ may be a useful quantity independently of LIT.

LIT is less sensitive to hyperparameters than baselines, but ensemble size matters more. In both our synthetic examples (Figure 4.2) and our ICU mortality results (Figures 4.6

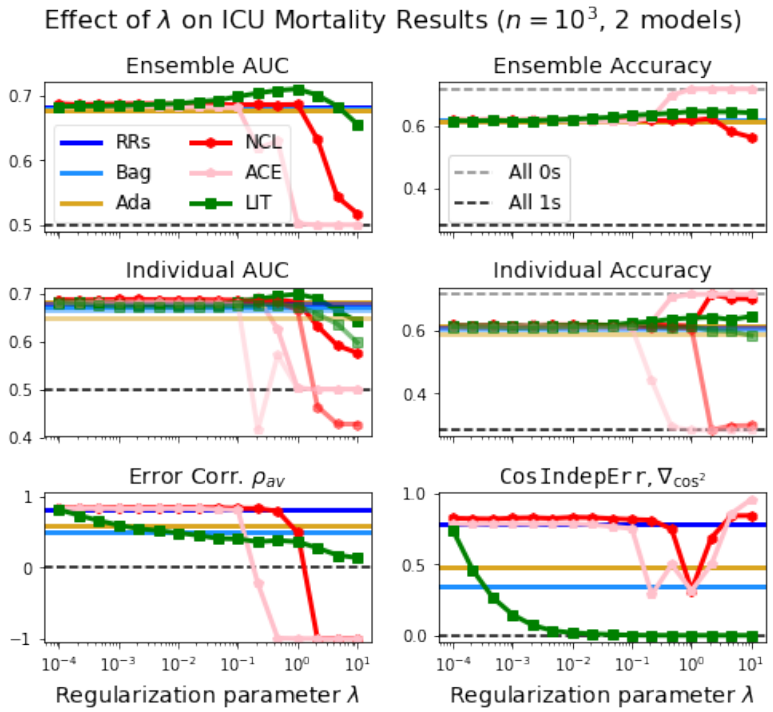


Figure 4.6: *Changes in individual AUC/accuracy and ensemble diversity with λ for two-model ensembles on the ICU mortality dataset (averaged across 10 reruns, error-bars omitted for clarity). For NCL and ACE, there is a wide low-λ regime where they are indistinguishable from random restarts. This is followed by a very brief window of meaningful diversity (around $\lambda = 1$ for NCL, slightly lower for ACE), after which both methods output pairs of models which always predict 0 and 1 (respectively), as shown by the error correlation dropping to -1. LIT, on the other hand, exhibits smooth drops in individual model predictive performance, with error correlation falling towards 0. Results for other ensemble sizes were qualitatively similar.*

and 4.7), we found that LIT produced qualitatively similar (diverse) results over several orders of magnitude of λ. NCL, on the other hand, required careful tuning of λ to achieve meaningful diversity (before its performance plummeted). In line with the results from our synthetic examples, we believe this difference stems from the fact that NCL's diversity term is formulated as a direct tradeoff with individual model accuracy, so the balance must be precise, whereas LIT's diversity term can theoretically be completely independent of individual model accuracy (which is true by construction in the synthetic examples). However, datasets only have the capacity to support a limited number of (mostly or completely) locally independent models. On the synthetic datasets, this capacity was exactly 2, but on real data, it is generally unknown, and it may be possible to achieve similar

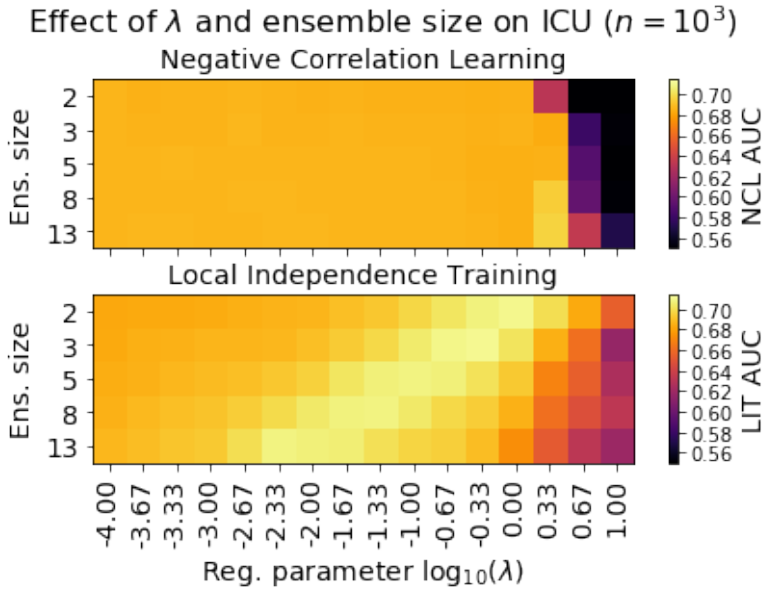


Figure 4.7: *Another exploration of the effect of ensemble size and λ on ICU mortality predictions. In particular, we find that for LIT on this dataset, the optimal value of λ depends on the ensemble size in a roughly log-linear relationship. Because D-dimensional datasets can support a maximum of D locally independent models (and only one model if the data completely determines the decision boundary), it is intuitive that there should be an optimal value. For NCL, we also observe an optimal value near $10^{0.33}$, but with a less clear relationship to ensemble size and very steep dropoff to random guessing at slightly higher λ.*

results either with a small fully independent ensemble or a large partially independent ensemble. For example, in Figure 4.7, we show that we can achieve similar improvements to ICU mortality prediction with 2 highly independent ($\lambda = 10^0$) models or 13 more weakly independent ($\lambda = 10^{-2.33}$) models. We hypothesize that the trend-line of optimal LIT ensemble size and λ may be a useful tool for characterizing the amount of ambiguity present in a dataset.

Interpretation of individual LIT models can yield useful dataset insights. In Figure 4.8, we found that in discussions with ICU clinicians, mortality feature assocations for normally trained neural networks were somewhat confusing due to hidden collinearities. As in our synthetic 8D example, LIT models made more clinical sense individually, and the differences between them helped reveal those collinearities (in particular between elevated levels of blood urea nitrogen and creatinine). Because LIT ensembles are often optimal when small,

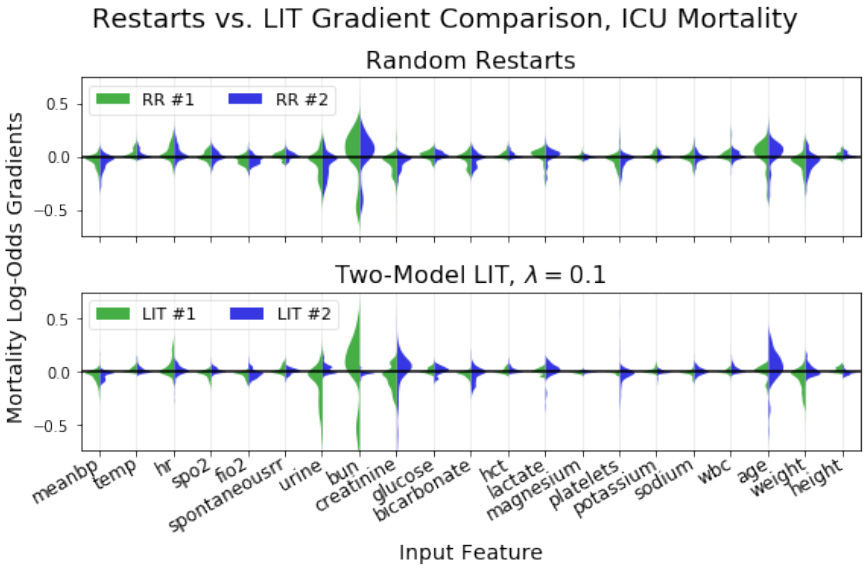


Figure 4.8: *Differences in cross-patient gradient distributions of ICU mortality prediction models for random restart and locally independent ensembles (similar plots for other methods are shown in Figure B.3). Features with mass consistently above the x-axis have positive associations with predicted mortality (increasing them increases predicted mortality) while those with mass consistently below the x-axis have negative associations (decreasing them increases predicted mortality). Distance from the x-axis corresponds to the association strength. Models trained normally (top) consistently learn positive associations with* `age` *and* `bun` *(blood urea nitrogen; larger values indicate kidney failure) and negative associations with* `weight` *and* `urine` *(low weight is correlated with mortality; low urine output also indicates kidney failure or internal bleeding). However, they also learn somewhat negative associations with* `creatinine`*, which confused clinicians because high values are another indicator of kidney failure. When we trained LIT models, however, we found that* `creatinine` *regained its positive association with mortality (in model 2), while the other main features were more or less divided up. This collinearity between* `creatinine` *and* `bun`*/*`urine` *in indicating organ problems (and revealed by LIT) was one of the main insights derived in our qualitative evaluation with ICU clinicians.*

and because individual LIT models are not required to sacrifice accuracy for diversity, they may enable different and more useful kinds of data interpretation than other ensemble methods.

Limitations. LIT does come with restrictions and limitations. In particular, we found that it works well for rectifier activations (e.g. ReLU and softplus[3]) but leads to inconsistent behavior with others (e.g. sigmoid and tanh). This may be related to the linear rather than

[3]Although we used ReLU in our quantitative experiments, we found more consistent behavior in synthetic examples with softplus, perhaps due to its many-times differentiability.

saturating extrapolation behavior of rectifiers. Because it relies on cosine similarity, LIT is also sensitive to relative changes in feature scaling; however, in practice this issue can be resolved by standardizing variables first.

Additionally, our cosine similarity approximation in LIT makes the assumption that the data manifold is locally similar to $\mathbb{R}^D$ near most inputs. However, we introduce generalizations in Section B.1 to handle situations where this is not approximately true (such as with image data).

Finally, LIT requires computing a second derivative (the derivative of the penalty) during the optimization process, which increases memory usage and training time; in practice, LIT took approximately 1.5x as long as random restarts, while NCL took approximately half the time. However, significant progress is being made on making higher-order autodifferentiation more efficient [27], so we can expect improvements. Also, in cases where LIT achieves high accuracy with a comparatively small ensemble size (e.g. ICU mortality prediction), overall training time can remain short if cross-validation terminates early.

4.6 Conclusion

In this paper, we presented a novel diversity metric that formalizes the notion of difference in local extrapolations. Based on this metric we defined an ensemble method, local independence training, for building ensembles of highly predictive base models that generalize differently outside the training set. On datasets we knew supported multiple diverse decision boundaries, we demonstrated our method's ability to recover them. On real-world datasets with unknown levels of redundancy, we demonstrated that LIT ensembles perform competitively on traditional prediction tasks and were more robust to data scarcity and covariate shift (as measured by training on inliers and testing on outliers). Finally, in our case study on a clinical prediction task in the intensive care unit, we provided evidence that the extrapolation diversity exhibited by LIT ensembles improved data robustness and helped us reach meaningful clinical insights in conversations with clinicians. Together, these results suggest that extrapolation diversity may be an important quantity for ensemble

algorithms to measure and optimize.

There are ample directions for future improvements. For example, it would be useful to consider methods for aggregating predictions of LIT ensembles using a more complex mechanism, such as a mixture-of-experts model. Along similar lines, combining pairwise `IndepErrs` in more informed way, such as a determinantal point process penalty [121] over the matrix of model similarities, may help us better quantify the diversity of the ensemble. Another interesting extension of our work would be to prediction tasks in semi-supervised settings, since labels are generally not required for computing local independence error. Finally, as we observe in the Section 4.5, some datasets seem to support a particular number of locally independent models. It is worth exploring how to connect this property to attempts to formally quantify and characterize the complexity or ambiguity present in a prediction task [148, 203, 156].

Chapter 5

Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients[1]

5.1 Introduction

In the previous two chapters, we used input gradient penalties to encourage neural networks to make predictions for specific or different reasons. We demonstrated this on ambiguous, extrapolated, or "decoy" datasets deliberately designed to deceive models making decisions for different reasons. This philosophy of testing—that we should measure generalization by testing on a different distribution than the one we trained on—can be taken to its extreme by testing models in an adversarial setting, where neural networks have known vulnerabilities [224]. In this chapter, we consider whether a knowledge-agnostic application of explanation

[1]This chapter is based on Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

regularization (a uniform L2 penalty on input gradients, analogous to Ridge regression on the model's local linear approximations) could help defend against adversarial examples.

Adversarial examples pose serious obstacles for the adoption of neural networks in settings which are security-sensitive or have legal ramifications [105]. Although many techniques for generating these examples (which we call "attacks") require access to model parameters, Papernot et al. [2017] have shown that it is possible and even practical to attack black-box models in the real world, in large part because of *transferability*; examples generated to fool one model tend to fool *all* models trained on the same dataset. Particularly for images, these adversarial examples can be constructed to fool models across a variety of scales and perspectives [15], which poses a problem for the adoption of deep learning models in systems like self-driving cars.

Although there has recently been a great deal of research in adversarial defenses, many of these methods have struggled to achieve robustness to transferred adversarial examples [225]. Some of the most effective defenses simply detect and reject them rather than making predictions [238]. When the paper this chapter is based on was originally written, the most common, "brute force" solution was adversarial training, where we include a mixture of normal and adversarial examples in the training set [126]. However, Tramèr et al. [2018] showed that the robustness provided by initial forms of adversarial training can be circumvented by randomizing or transferring perturbations from other models (though ensembling helps). As an important note, subsequent work (now as of 2021, the date of this dissertation) has shown that adversarial training actually can be made much more effective with a "strong" adversary—that is, if we spend much more computational effort generating worst-case adversarial examples at each training step. We will discuss this approach and follow-up work at the end of this chapter.

As we noted in Chapter 3, domain experts are also often concerned that DNN predictions are uninterpretable. The lack of interpretability is particularly problematic in domains where algorithmic bias is often a factor [11] or in medical contexts where safety risks can arise when there is mismatch between how a model is trained and used [39]. For computer vision

models (the primary target of adversarial attacks), the most common class of explanation is the saliency map, either at the level of raw pixels, grid chunks, or superpixels [183].

The local linear approximation provided by raw input gradients [19] is sometimes used for pixel-level saliency maps [211]. However, computer vision practitioners tend not to examine raw input gradients because they are noisy and difficult to interpret. This issue has spurred the development of techniques like integrated gradients [222] and SmoothGrad [216] that generate smoother, more interpretable saliency maps from noisy gradients. The rationale behind these techniques is that, while the local behavior of the model may be noisy, examining the gradients over larger length scales in input space provides a better intution about the model's behavior.

However, raw input gradients are *exactly* what many attacks use to generate adversarial examples. Explanation techniques which smooth out gradients in background pixels may be inappropriately hiding the fact that the model is quite sensitive to them. We consider that perhaps the need for these smoothing techniques in the first place is indicative of a problem with our models, related to their adversarial vulnerability and capacity to overfit. Perhaps it is fundamentally hard for adversarially vulnerable models to be interpretable.

On the other hand, perhaps it is hard for interpretable models to be adversarially vulnerable. Our hypothesis is that by training a model to have smooth input gradients with fewer extreme values, it will not only be more interpretable but also more resistant to adversarial examples. In the experiments that follow we confirm this hypothesis using uniform gradient regularization, which optimizes the model to have smooth input gradients with respect to its predictions during training. Using this technique, we demonstrate robustness to adversarial examples across multiple model architectures and datasets, and in particular demonstrate robustness to *transferred* adversarial examples: gradient-regularized models maintain significantly higher accuracy on examples generated to fool other models than baselines. Furthermore, both qualitatively and in human subject experiments, we find that adversarial examples generated to fool gradient-regularized models are, in a particular sense, more "interpretable": they fool humans as well.

5.2 Related Work

In this section, we will (re)introduce notation, and give a brief overview of the baseline attacks and defenses against which we will test and compare our methods. The methods we will analyze again apply to all differentiable classification models $f_\theta(X)$, which are functions parameterized by θ that return predictions $\hat{y} \in \mathbb{R}^{N \times K}$ given inputs $X \in \mathbb{R}^{N \times D}$. These predictions indicate the probabilities that each of N inputs in D dimensions belong to each of K class labels. To train these models, we try to find sets of parameters θ^* that minimize the total information distance between the predictions $\hat{y}$ and the true labels y (also $\in \mathbb{R}^{N \times K}$, one-hot encoded) on a training set:

$$\theta^* = \arg\min_\theta \sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \log f_\theta(X_n)_k, \tag{5.1}$$

which we will sometimes write as

$$\arg\min_\theta H(y, \hat{y}),$$

with H giving the sum of the cross entropies between the predictions and the labels.

5.2.1 Attacks

Fast Gradient Sign Method (FGSM)

Goodfellow et al. [2014] introduced this first method of generating adversarial examples by perturbing inputs in a manner that increases the local linear approximation of the loss function:

$$X_{\text{FGSM}} = X + \epsilon \operatorname{sign}\left(\nabla_x H(y, \hat{y})\right) \tag{5.2}$$

If ϵ is small, these adversarial examples are indistinguishable from normal examples to a human, but the network performs significantly worse on them.

Kurakin et al. [2016] noted that one can iteratively perform this attack with a small ϵ to induce misclassifications with a smaller total perturbation (by following the nonlinear loss function in a series of small linear steps rather than one large linear step).

Targeted Gradient Sign Method (TGSM)

A simple modification of the Fast Gradient Sign Method is the Targeted Gradient Sign Method, introduced by Kurakin et al. [2016]. In the TGSM, we try to decrease a modified version of the loss function that encourages the model to misclassify in a specific way:

$$X_{\text{TGSM}} = X - \epsilon \operatorname{sign}\left(\nabla_x H(y_{\text{target}}, \hat{y})\right), \tag{5.3}$$

where y_{target} encodes an alternate set of labels we would like the model to predict instead. In the digit classification experiments below, we often picked targets by incrementing the labels y by 1 (modulo 10), which we will refer to as y_{+1}. The TGSM can also be performed iteratively.

Jacobian-based Saliency Map Approach (JSMA)

The final attack we consider, the Jacobian-based Saliency Map Approach (JSMA), also takes an adversarial target vector y_{target}. It iteratively searches for pixels or pairs of pixels in X to change such that the probability of the target label is increased and the probability of all other labels are decreased. This method is notable for producing examples that have only been changed in several dimensions, which can be hard for humans to detect. For a full description of the attack, we refer the reader to Papernot et al. [2016].

Newer Attacks

Since Ross et al. [2018] was published, stronger attacks have been developed. Many are based on the iterated FGSM, but with a projection step, so that adversarial examples remain within an ϵ-ball of the original input. For example, Madry et al. [2017] apply projected 20-step gradient descent (modern works often use 50 or 100 steps), and also start the optimization with a random perturbation away from the original input to handle non-convexity. Chen et al. [2018] solve an inner elastic net optimization problem which weights increasing the loss against the perturbation size. These approaches require significantly more computation but are significantly more effective.

5.2.2 Defenses

As baseline defenses, we consider defensive distillation and one-step adversarial training. To simplify comparison, we omit defenses [238, 165] that are not fully architecture-agnostic or which work by detecting and rejecting adversarial examples.

Distillation

Distillation, originally introduced by Ba and Caruana [2014], was first examined as a potential defense by Papernot et al. [2016]. The main idea is that we train the model twice, initially using the one-hot ground truth labels but ultimately using the initial model's softmax probability outputs, which contain additional information about the problem. Since the normal softmax function tends to converge very quickly to one-hot-ness, we divide all of the logit network outputs (which we will call $\hat{z}_k$ instead of the probabilities $\hat{y}_k$) by a temperature T (during training but not evaluation):

$$f_{T,\theta}(X_n)_k = \frac{e^{\hat{z}_k(X_n)/T}}{\sum_{i=1}^{K} e^{\hat{z}_i(X_n)/T}}, \tag{5.4}$$

where we use $f_{T,\theta}$ to denote a network ending in a softmax with temperature T. Note that as T approaches ∞, the predictions converge to $\frac{1}{K}$. The full process can be expressed as

$$\begin{aligned} \theta^0 &= \arg\min_{\theta} \sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \log f_{T,\theta}(X_n)_k, \\ \theta^* &= \arg\min_{\theta} \sum_{n=1}^{N} \sum_{k=1}^{K} -f_{T,\theta^0}(X_n)_k \log f_{T,\theta}(X_n)_k. \end{aligned} \tag{5.5}$$

Distillation is usually used to help small networks achieve the same accuracy as larger DNNs, but in a defensive context, we use the same model twice. It has been shown to be an effective defense against white-box FGSM attacks, but Carlini and Wagner [2016] have shown that it is not robust to all kinds of attacks. We will see that the precise way it defends against certain attacks is qualitatively different than gradient regularization, and that it can actually make the models more vulnerable to attacks than an undefended model.

Adversarial Training

In adversarial training [126], we increase robustness by injecting adversarial examples into the training procedure. We follow the method implemented in Papernot et al. [2016], where we augment the network to run the single-step FGSM on the training batches and compute the model's loss function as the average of its loss on normal and adversarial examples without allowing gradients to propogate so as to weaken the FGSM attack (which would also make the method second-order). We compute FGSM perturbations with respect to predicted rather than true labels to prevent "label leaking," where our model learns to classify adversarial examples more accurately than regular examples.

Stronger Adversarial Training

A more modern approach to adversarial training is to augment the network to run multi-step FGSM or projected gradient descent on each example during each training iteration, per the developments in Section 5.2.1. Although we do not test this approach in this chapter, this approach has been shown to be more effective than one-step adversarial training and produces models whose qualitative behavior is actually more similar to those obtained by gradient regularization, which we will now define. We discuss strong adversarial training more at the end of this chapter.

5.3 Method

We defined our "right for the right reasons" objective in Chapter 3 using an L2 penalty on the gradient of the model's predictions across classes with respect to input features marked irrelevant by domain experts. We encoded their domain knowledge using an annotation matrix A. If we set $A = \mathbb{1}$, however, and consider only the log-probabilities of the predicted classes, we recover what Drucker and Le Cun [1992] introduced as "double backpropagation", which trains neural networks by minimizing not just the "energy" of the network but the rate of change of that energy with respect to the input features. In their

formulation the energy is a quadratic loss, but we can reformulate it almost equivalently using the cross-entropy:

$$\theta^* = \arg\min_\theta \sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \log f_\theta(X_n)_k + \lambda \sum_{d=1}^{D} \sum_{n=1}^{N} \left(\frac{\partial}{\partial x_d} \sum_{k=1}^{K} -y_{nk} \log f_\theta(X_n)_k \right)^2, \tag{5.6}$$

whose objective we can write a bit more concisely as

$$\arg\min_\theta H(y, \hat{y}) + \lambda ||\nabla_x H(y, \hat{y})||_2^2,$$

where λ is again a hyperparameter specifying the penalty strength. The intuitive objective of this function is to ensure that if any input changes slightly, the divergence between the predictions and the labels will not change significantly (though including this term does not guarantee Lipschitz continuity everywhere). Double backpropagation was mentioned as a potential adversarial defense in the same paper which introduced defensive distillation [174], but at publish time, its effectiveness in this respect had not yet been analyzed in the literature – though [83] previously and [91, 53] concurrently consider related objectives, and [180] derive and minimize an upper bound on adversarial vulnerability based on the maximum gradient norm in a ball around each training input. These works also provide stronger theoretical explanations for why input gradient regularization is effective, though they do not analyze its relationship to model interpretability. In this work, we interpret gradient regularization as a quadratic penalty on our model's saliency map.

5.4 Experiments

Datasets and Models

We evaluated the robustness of distillation, adversarial training, and gradient regularization to the FGSM, TGSM, and JSMA on MNIST [132], Street-View House Numbers (SVHN) [166], and notMNIST [36]. On all datasets, we test a simple convolutional neural network with 5x5x32 and 5x5x64 convolutional layers followed by 2x2 max pooling and a 1024-unit fully connected layer, with batch-normalization after all convolutions and both batch-

normalization and dropout on the fully-connected layer. All models were implemented in Tensorflow and trained using Adam [112] with $\alpha = 0.0002$ and $\epsilon = 10^{-4}$ for 15000 minibatches of size of 256. For SVHN, we prepare training and validation set as described in Sermanet et al. [2012], converting the images to grayscale following Grundland and Dodgson [2007] and applying both global and local contrast normalization.

Attacks and Defenses

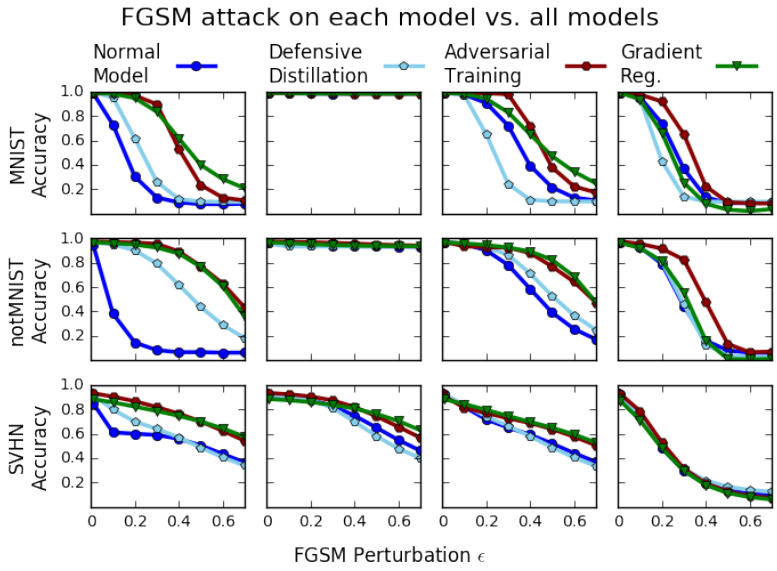


Figure 5.1: *Accuracy of all CNNs on FGSM examples generated to fool undefended models, defensively distilled, adversarially trained, and gradient regularized models (from left to right) on MNIST, SVHN, and notMNIST (from top to bottom). Gradient-regularized models are the most resistant to other models' adversarial examples at high ϵ, while all models are fooled by gradient-regularized model examples. On MNIST and notMNIST, distilled model examples are usually identical to non-adversarial examples (due to gradient underflow), so they fail to fool any of the other models.*

For adversarial training and JSMA example generation, we used the Cleverhans adversarial example library [172]. For distillation, we used a softmax temperature of $T = 50$, and for adversarial training, we trained with FGSM perturbations at $\epsilon = 0.3$, averaging normal and adversarial losses. For gradient regularized models, we use double backpropagation, which provided the best robustness, and train over a spread of λ values. We choose the λ with the highest accuracy against validation black-box FGSM examples but which is still at

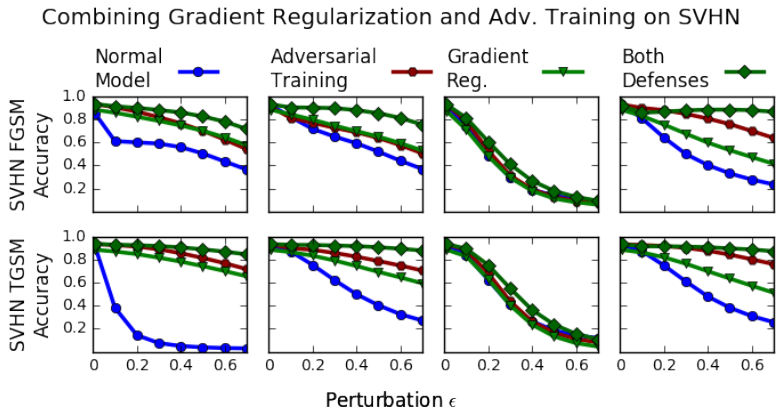


Figure 5.2: *Applying both gradient regularization and adversarial training ("both defenses") allows us to obtain maximal robustness to white-box and normal black-box attacks on SVHN (with a very slight label-leaking effect on the FGSM, perhaps due to the inclusion of the $\nabla_x H(y, \hat{y})$ term). However, no models are able to maintain robustness to black-box attacks using gradient regularization.*

least 97% as accurate on normal validation examples (though accuracy on normal examples tended not to be significantly different). Code for all models and experiments has been open-sourced[2].

Evaluation Metrics

For the FGSM and TGSM, we test *all* models against adversarial examples generated for *each* model and report accuracy. Testing this way allows us to simultaneously measure white- and black-box robustness.

On the JSMA and iterated TGSM, we found that measuring accuracy was no longer a good evaluation metric, since for our gradient-regularized models, the generated adversarial examples often resembled their targets more than their original labels. To investigate this, we performed a human subject experiment to evaluate the legitimacy of adversarial example misclassifications.

[2] https://github.com/dtak/adversarial-robustness-public

5.4.1 Accuracy Evaluations (FGSM and TGSM)

FGSM Robustness

Figure 5.1 shows the results of our defenses' robustness to the FGSM on MNIST, SVHN, and notMNIST for our CNN at a variety of perturbation strengths ϵ. Consistently across datasets, we find that gradient-regularized models exhibit strong robustness to black-box transferred FGSM attacks (examples produced by attacking other models). Although adversarial training sometimes performs slightly better at $\epsilon \leq 0.3$, the value we used in training, gradient regularization generally surpasses it at higher ϵ (see the green curves in the leftmost plots).

The story with white-box attacks is more interesting. Gradient-regularized models are generally more robust to than undefended models (visually, the green curves in the rightmost plots fall more slowly than the blue curves in the leftmost plots). However, accuracy still eventually falls for them, and it does so faster than for adversarial training. Even though their robustness to white-box attacks seems lower, though, the examples produced by those white-box attacks actually fool *all* other models equally well. This effect is particularly pronounced on SVHN. In this respect, gradient regularization may hold promise not just as a defense but as an *attack*, if examples generated to fool them are inherently more transferable.

Models trained with defensive distillation in general perform no better and often worse than undefended models. Remarkably, except on SVHN, attacks against distilled models actually fail to fool all models. Closer inspection of distilled model gradients and examples themselves reveals that this occurs because distilled FGSM gradients vanish – so the examples are not perturbed at all. As soon as we obtain a nonzero perturbation from a different model, distillation's appearance of robustness vanishes as well.

Although one-step adversarial training and gradient regularization seem comparable in terms of accuracy, they work for different reasons and can be applied in concert to increase robustness, which we show in Figure 5.2. In Figure 5.3 we also show that, on normal and one-step adversarially trained black-box FGSM attacks, models trained with these two

defenses are fooled by different sets of adversarial examples. We provide intuition for why this might be the case in Figure 5.4.

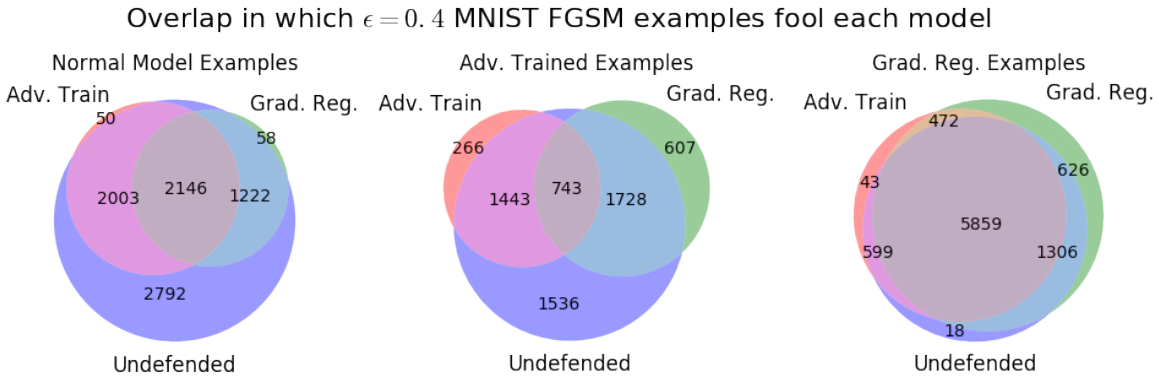


Figure 5.3: *Venn diagrams showing overlap in which MNIST $\epsilon = 0.4$ FGSM examples, generated for normal, one-step adversarially trained, and gradient regularized models, fool all three. Undefended models tend to be fooled by examples from all models, while the sets of one-step adversarially trained model FGSM examples that fool the two defended models are closer to disjoint. Gradient-regularized model FGSM examples fool all models. These results suggest that ensembling different forms of defense may be effective in defending against black box attacks (unless those black box attacks use a gradient-regularized proxy).*

TGSM Robustness

Against the TGSM attack (Figure 5.5), defensively distilled model gradients no longer vanish, and accordingly these models start to show the same vulnerability to adversarial attacks as others. Gradient-regularized models still exhibit the same robustness even at large perturbations ϵ, and again, examples generated to fool them fool other models equally well.

One way to better understand the differences between gradient-regularized, normal, and distilled models is to examine the log probabilities they output and the norms of their loss function input gradients, whose distributions we show in Figure 5.6 for MNIST. We can see that the different defenses have very different statistics. Probabilities of non-predicted classes tend to be small but remain nonzero for gradient-regularized models, while they vanish on defensively distilled models evaluated at $T = 0$ (despite distillation's stated purpose of discouraging certainty). Perhaps because $\nabla \log p(x) = \frac{1}{p(x)} \nabla p(x)$, defensively distilled

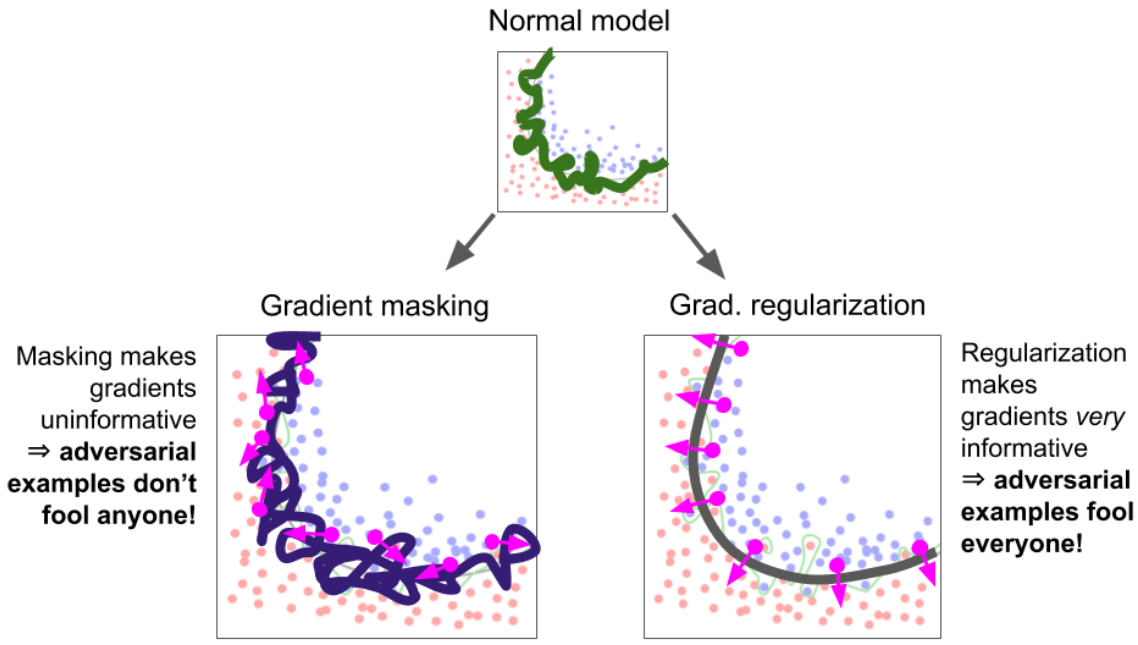


Figure 5.4: *Conceptual illustration of the difference between gradient regularization and gradient masking. In (idealized) gradient masking, input gradients are completely uninformative, so following them doesn't affect either the masked model's predictions or those of any other model. In gradient regularization, gradients actually become more informative, so following them will ultimately fool all models. However, because gradients are also smaller, perturbations need to be larger to flip predictions. Unregularized, unmasked models are somewhere in between. We see quantitative support for this interpretation in Figure 5.3, as well as qualitative evidence in Figure 5.9.*

models' non-predicted log probability input gradients are the largest by many orders of magnitude, while gradient-regularized models' remain controlled, with much smaller means and variances. The other models lie between these two extremes. While we do not have a strong theoretical argument about what input gradient magnitudes *should* be, we believe it makes intuitive sense that having less variable, well-behaved, and non-vanishing input gradients should be associated with robustness to attacks that consist of small perturbations in input space.

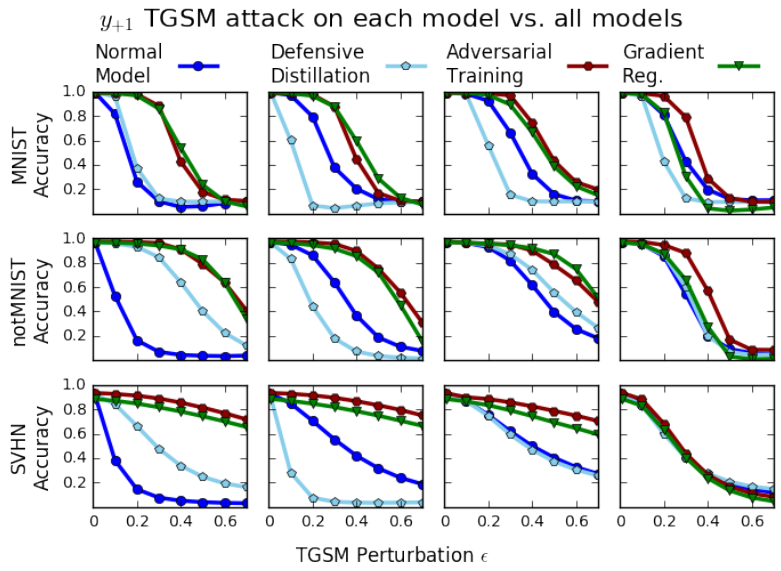


Figure 5.5: *CNN accuracy on y_{+1} TGSM examples generated to fool the four models on three datasets (see Figure 5.1 for more explanation). Gradient-regularized models again exhibit robustness to other models' adversarial examples. Distilled model adversarial perturbations fool other models again since their input gradients no longer underflow.*

5.4.2 Human Subject Study (JSMA and Iterated TGSM)

Need for a Study

Accuracy scores against the JSMA can be misleading, since without a maximum distortion constraint it necessarily runs until the model predicts the target. Even with such a constraint, the perturbations it creates sometimes alter the examples so much that they no longer resemble their original labels, and in some cases bear a greater resemblance to their targets. Figure 5.7 shows JSMA examples on MNIST for gradient-regularized and distilled models which attempt to convert `0`s and `1`s into every other digit. Although all of the perturbations "succeed" in changing the model's prediction, in the gradient-regularized case, many of the JSMA examples strongly resemble their targets.

The same issues occur for other attack methods, particularly the iterated TGSM, for which we show confusion matrices for different models and datasets in Figure 5.8. For the gradient-regularized models, these psuedo-adversarial examples quickly become almost prototypical examples of their targets, which is not reflected in accuracies with respect to

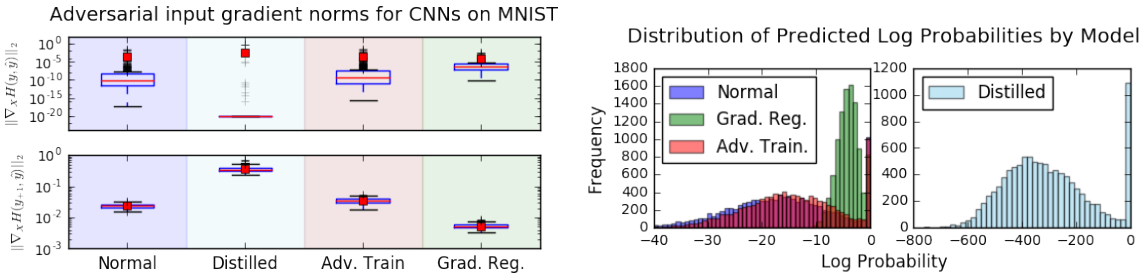


Figure 5.6: *Distributions of (L2 norm) magnitudes of FGSM input gradients (top), TGSM input gradients (middle), and predicted log probabilities across all classes (bottom) for each defense. Note the logarithmic scales. Gradient-regularized models tend to assign non-predicted classes higher probabilities, and the L2 norms of the input gradients of their FGSM and TGSM loss function terms have similar orders of magnitude. Distilled models (evaluated at $T = 0$) assign extremely small probabilities to all but the predicted class, and their TGSM gradients explode while their FGSM gradients vanish (we set a minimum value of 10^{-20} to prevent underflow). Normal and adversarially trained models lie somewhere in the middle.*

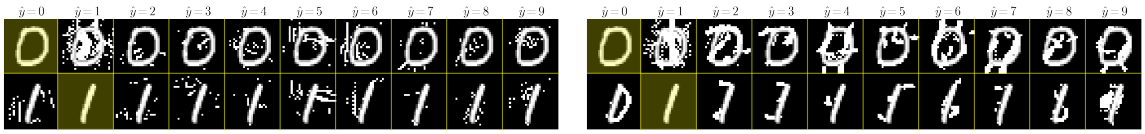


Figure 5.7: *Results of applying the JSMA to MNIST `0` and `1` images with maximum distortion parameter $\gamma = 0.25$ for a distilled model (left) and a gradient-regularized model (right). Examples in each row start out as the highlighted digit but are modified until the model predicts the digit corresponding to their column or the maximum distortion is reached.*

the original labels.

To test these intuitions more rigorously, we ran a small pilot study with 11 subjects to measure whether they found examples generated by these methods to be more or less plausible instances of their targets.

Study Protocol

The pilot study consisted of a quantitative and qualitative portion. In the quantitative portion, subjects were shown 30 images of MNIST JSMA or SVHN iterated TGSM examples. Each of the 30 images corresponded to one original digit (from 0 to 9) and one model (distilled, gradient-regularized, or undefended). Note that for this experiment, we used $\nabla_x H(\frac{1}{K}, \hat{y})$ gradient regularization, ran the TGSM for just 10 steps, and trained models for 4 epochs at a learning rate of 0.001. This procedure was sufficient to produce examples with

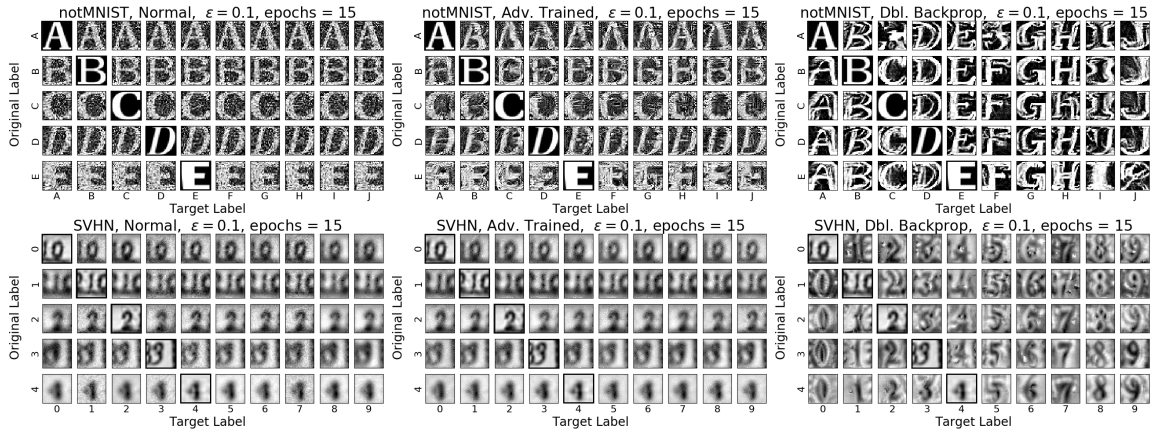


Figure 5.8: *Partial confusion matrices showing results of applying the iterated TGSM for 15 iterations at* $\epsilon = 0.1$. *Each row is generated from the same example but modified to make the model to predict every other class. TGSM examples generated for gradient-regularized models (right) resemble their targets more than their original labels and may provide insight into what the model has learned. Animated versions of these examples can be seen at* `http://goo.gl/q8ZM1T`.

explanations similar to the longer training procedure used in our earlier experiments, and actually increased the robustness of the undefended models (adversarial accuracy tends to fall with training iteration). Images were chosen uniformly at random from a larger set of 45 examples that corresponded to the first 5 images of the original digit in the test set transformed using the JSMA or iterated TGSM to each of the other 9 digits (we ensured that all models misclassified all examples as their target). Subjects were not given the original label, but were asked to input what they considered the most and second-most plausible predictions for the image that they thought a reasonable classifier would make (entering N/A if they thought no label was a plausible choice). In the qualitative portion that came afterwards, users were shown three 10x10 confusion matrices for the different defenses on MNIST (Figure 5.7 shows the first two rows) and were asked to write comments about the differences between the examples. Afterwards, there was a short group discussion. This study was performed in compliance with the institution's IRB.

Table 5.1: *Adversarial example user study results.*

	MNIST (JSMA)		SVHN (TGSM)	
Model	human fooled	mistake reasonable	human fooled	mistake reasonable
normal	2.0%	26.0%	40.0%	63.3%
distilled	0.0%	23.5%	1.7%	25.4%
grad. reg.	**16.4%**	**41.8%**	**46.3%**	**81.5%**

Quantitative feedback from the human subject experiment. "human fooled" columns record what percentage of examples were classified by humans as most plausibly their adversarial targets, and "mistake reasonable" records how often humans either rated the target plausible or marked the image unrecognizable as any label (N/A).

Study Results

Table 5.1 shows quantitative results from the human subject experiment. Overall, subjects found gradient-regularized model adversarial examples most convincing. On SVHN and especially MNIST, humans were most likely to think that gradient-regularized (rather than distilled or normal) adversarial examples were best classified as their target rather than their original digit. Additionally, when they did not consider the target the most plausible label, they were most likely to consider gradient-regularized model mispredictions "reasonable" (which we define in Table 5.1), and more likely to consider distilled model mispredictions unreasonable. p-values for the differences between normal and gradient regularized unreasonable error rates were 0.07 for MNIST and 0.08 for SVHN.

In the qualitative portion of the study (comparing MNIST JSMA examples), *all* of the written responses described significant differences between the insensitive model's JSMA examples and those of the other two methods. Many of the examples for the gradient-regularized model were described as "actually fairly convincing," and that the normal and distilled models "seem to be most easily fooled by adding spurious noise." Few commentators indicated any differences between the normal and distilled examples, with several saying that "there doesn't seem to be [a] stark difference" or that they "couldn't

describe the difference" between them. In the group discussion one subject remarked on how the perturbations to the gradient-regularized model felt "more intentional", and others commented on how certain transitions between digits led to very plausible fakes while others seemed inherently harder. Although the study was small, both its quantitative and qualitative results support the claim that gradient regularization, at least for the two CNNs on MNIST and SVHN, is a credible defense against the JSMA and the iterated TGSM, and that distillation is not.

5.4.3 Connections to Interpretability

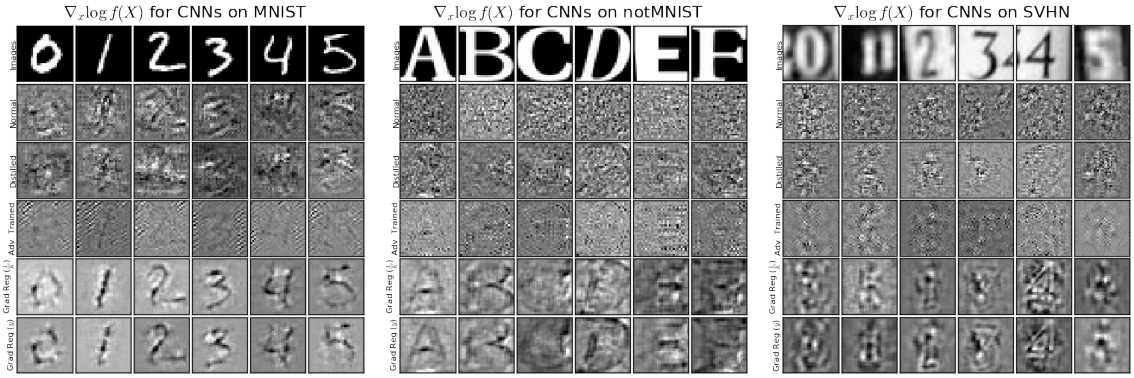


Figure 5.9: *Input gradients $\nabla_x H(\frac{1}{K}, \hat{y})$ that provide a local linear approximation of normal models (top), distilled models at $T = 50$ (second from top), adversarially trained models (middle), and models trained with $\nabla_x H(\frac{1}{K}, \hat{y})$ and $\nabla_x H(y, \hat{y})$ gradient regularization (bottom two). Whitening black pixels or darkening white pixels makes the model more certain of its prediction. In general, regularized model gradients appear smoother and make more intuitive sense as local linear approximations.*

Finally, we present a qualitative evaluation suggesting a connection between adversarial robustness and interpretability. In the literature on explanations, input gradients are frequently used as explanations [19], but sometimes they are noisy and not interpretable on their own. In those cases, smoothing techniques have been developed [216, 209, 222] to generate more interpretable explanations, but we have already argued that these techniques may obscure information about the model's sensitivity to background features.

We hypothesized that if the models had more interpretable input gradients without the need for smoothing, then perhaps their adversarial examples, which are generated directly

from their input gradients, would be more interpretable as well. That is, the adversarial example would be more obviously transformative away from the original class label and towards another. The results of the user study show that our gradient-regularized models have this property; here we ask if the gradients are more interpretable as explanations.

In Figure 5.9 we visualize input gradients across models and datasets, and while we cannot make any quantitative claims, there does appear to be a qualitative difference in the interpretability of the input gradients between the gradient-regularized models (which were relatively robust to adversarial examples) and the normal and distilled models (which were vulnerable to them). One-step adversarially trained models seem to exhibit slightly more interpretable gradients, but not nearly to the same degree as gradient-regularized models. When we repeatedly apply input gradient-based perturbations using the iterated TGSM (Figure 5.8), this difference in interpretability between models is greatly magnified, and the results for gradient-regularized models seem to provide insight into what the model has learned. When gradients become interpretable, adversarial images start resembling feature visualizations [168]; in other words, they become explanations.

5.5 Discussion

In this chapter, we showed that:

- Gradient regularization slightly outperforms one-step adversarial training as a defense against black-box transferred FGSM examples from undefended models.
- Gradient regularization significantly increases robustness to white-box attacks, though not quite as much as adversarial training.
- Adversarial examples generated to fool gradient-regularized models are more "universal;" they are more effective at fooling all models than examples from unregularized models.
- Adversarial examples generated to fool gradient-regularized models are more in-

terpretable to humans, and examples generated from iterative attacks quickly come to legitimately resemble their targets. This is not true for distillation or adversarial training.

The conclusion that we would *like* to reach is that gradient-regularized models are right for better reasons. Although they are not completely robust to attacks, their correct predictions and their mistakes are both easier to understand. To fully test this assertion, we would need to run a larger and more rigorous human subject evaluation that also tests adversarial training and other attacks beyond the JSMA, FGSM, and TGSM.

Connecting what we have done back to the general idea of explanation regularization, we saw in Equation 5.6 that we could interpret our defense as a quadratic penalty on our CNN's saliency map. Imposing this penalty had both quantitative and qualitative effects; our gradients became smaller but also smoother with fewer high-frequency artifacts. Since gradient saliency maps are just normals to the model's decision surface, these changes suggest a qualitative difference in the "reasons" behind our model's predictions. Many techniques for generating smooth, simple saliency maps for CNNs *not* based on raw gradients have been shown to vary under meaningless transformations of the model [111] or, more damningly, to remain invariant under extremely meaningful ones [4] – which suggests that many of these methods either oversimplify or aren't faithful to the models they are explaining. Our approach in this chapter was, rather than simplifying our explanations of fixed models, to optimize our *models* to have simpler explanations. Their increased robustness can be thought of as a useful side effect.

Although the problem of adversarial robustness in deep neural networks is still very much an open one, these results may suggest a deeper connection between it and interpretability. No matter what method proves most effective in the general case, we suspect that any progress towards ensuring either interpretability or adversarial robustness in deep neural networks will likely represent progress towards both.

5.5.1 Newer developments

In this chapter, we described both adversarial training and distillation as achieving "robustness" through gradient masking, which makes gradients less interpretable and (potentially) makes models right for *worse* reasons. However, Tsipras et al. [2018] and Engstrom et al. [2019] show that adversarial training against a much stronger adversary (many-step projected gradient descent) actually makes gradients more human-interpretable in the same way as gradient regularization.

Ilyas et al. [2019] explore this point further by showing that most image classification problems commonly considered in the literature are ambiguous: they can be solved either using low-frequency shape-based features (which are relatively easy for humans to interpret), or using high-frequency features that are legitimately correlated with class labels, but which most humans would not be able to comprehend. Strong adversarial training and gradient regularization both encourage models to utilize the lower-frequency, more interpretable features (rather than the high-frequency features, or a confusing combination of both).

In this sense, we can understand image classification problems in the same way as the ambiguous synthetic datasets that we introduced in the first two chapters. There are two potential classes of implicit decision rules that an accurate model could learn, one of which is generally more interpretable (and robust) than the other. Normal gradient descent produces models that use a complicated combination of both classes of rules. Gradient descent with gradient or perturbation-based regularization reshapes the optimization landscape to favor the interpretable class. This brings us back full circle to our parables in Chapter 1.

Part III

Interpretable Representations

Chapter 6

Background on Representations

6.1 Definitions and Related Work

The last three chapters have (hopefully) told a relatively coherent story: learning to predict a set of class labels y from associated inputs x is usually ambiguous, which makes models unnecessarily complex and vulnerable to distribution shifts, but we can mitigate these problems by encoding our domain knowledge in terms of input features, and then regularizing their gradients (or other related quantities).

However, sometimes domain knowledge is difficult to express in terms of input features. In real life, we often reason and explain in terms of concepts defined at varying levels of abstraction, as Keil [2006] notes:

> Explanations [...] suffer if presented at the wrong level of detail. Thus, if asked why John got on the train from New Haven to New York, a good explanation might be that he had tickets for a Broadway show. An accurate but poor explanation at too low a level might say that he got on the train because he moved his right foot from the platform to the train and then followed with his left foot. An accurate but poor explanation at too high a level might say that he got on the train because he believed that the train would take him to New York from New Haven.

In other words, there is often a conceptual hierarchy of reasons behind any event, and user knowledge might be specific to a certain level. Although it is possible that we could come

closer to encoding certain types of conceptual knowledge if we also regularized feature interactions [124, 228], input importance [116, 206], or input similarity [43], none of those approaches seems to solve the problem in general. It would be extremely helpful (both for explaining the reasons behind predictions and constraining them) if machine learning models and their users could communicate in a common conceptual language, ideally one that supports some notion of hierarchy. But that is a daunting task.

Concepts and Classifier Representations: However, it has long been noted that neural networks, which consist of many layers of recursive "feature extractors," actually operate by extracting progressively more abstract features from low-level inputs [241]. In other words, these vectors of "activations" at each hidden layer, often called embeddings or **representations**, are structurally similar to the kind of conceptual hierarchies we seek, though they may not always correspond in practice.

Several works have attempted to investigate whether and when they do. On classification datasets whose labels can be hierarchically decomposed into sub-concepts, Bau et al. [2017] find that different neural architectures and training methods lead to differing levels of correspondence between representation components and concept hierarchies. Given small datasets of examples of human concepts, Kim et al. [2017] test whether (and in what way) those concepts are meaningfully extracted by classifier representations and important for predictions, specifically by finding directions (called concept activation vectors, or CAVs) that best represent each concept.

From an alternate angle, Olah et al. [2017] visualize what inputs maximally activate representation dimensions (or combinations of them), so that human users can attempt to associate those dimensions (or combinations of them) with concepts; this approach is called feature visualization. Because the representations they consider are very large, Olah et al. [2018] use non-negative matrix factorization (NMF) to find linear combinations that best summarize the entire set, which are fewer in number and whose maximally-activating inputs tend to be more human-interpretable. In a similar spirit, Ghorbani et al. [2019] use image segmentation techniques to automatically discover CAVs that are influential to the

network, which they demonstrate are also often meaningful to human users.

Both of these approaches (trying to map pre-labeled human concepts to network representations, or inspecting unlabeled network representations and trying to map them back to human concepts) can be very helpful for explaining classifiers at higher levels of abstraction. If the network parameters responsible for generating the representations are fixed, they can also be adapted to allow for concept regularization using the techniques of Chapter 3, but with gradients taken with respect to concept-aligned representation directions (see Stammer et al. [2020] for an excellent example).

However, there are many potential issues with relying on the representations learned by standard classifiers. For one, it is possible that correspondences between human concepts and model representations will be weak or nonexistent. If an image classifier comes to rely exclusively on non-robust, high-frequency features in the sense of Ilyas et al. [2019], then its representations will not truly correspond to concepts that mean anything to human users, even if there remains some correlation. Additionally, even if we can identify a direction in representation space that has a strong aggregate association with a concept (e.g. by using a set of examples to learn a CAV, or by interpreting an example that maximizes a linear combination of dimensions identified by NMF), it is unclear whether this direction will continue to describe the concept at every point in space, because neural networks are nonlinear. To make progress, we may need to explicitly learn representations which are more interpretable, robust, or simply right.

Disentanglement and Generative Model Representations: The increasingly popular subfield of "disentangled representations" has much to say about what it means to learn the right representation [184, 92, 46, 45, 145], which directly inspires the contributions of the remaining chapters. Although we will leave most of our discussion (and criticism) of specific disentanglement methods and metrics until later, the subfield is generally focused on learning generative model representations that fully summarize the data (i.e. can be used

to generate new data[1]), but in a way that disentangles different "independent" ground-truth factors (that actually generated the data) into different representation dimensions (with varying definitions of independence). Informally, representations obtained by such disentangled representation learning methods are believed to be more interpretable than those obtained by standard techniques.

However, this belief has not faced much rigorous testing, in large part because interpretability—especially for representations—is difficult to quantify. Additionally, many disentangled representation learning methods make very strong assumptions about the (lack of) underlying structure in the dataset, e.g. that the ground-truth factors behind the dataset are statistically independent. However, many real-world generative processes contain highly structured dependencies.

6.2 Outline of Part III

In the next two chapters of Part III, we work towards resolving some of these problems, making the following contributions towards learning (or measuring if we have learned) the right representation:

- In Chapter 7, we introduce (and rigorously test) an interactive technique for both explaining representations and evaluating their interpretability. We use this technique to evaluate different interpretable representation learning methods and relate disentanglement metrics to human factors.

- In Chapter 8, we help generalize the idea of disentanglement to representations which are hierarchical in a novel sense: their dimensions are organized into a tree, but only the dimensions along a particular path are active at any given time. This structure allows us to learn explicit global structural models of our data while reducing

[1]Representation learning in generative modeling is distinct from representation learning in classification; if a generative model is expressive enough to generate every instance x in the support of $p(x)$, then it possible to convert between x and its representation without losing any information. In contrast, in neural networks trained for classification, each layer's representation of x discards more and more information irrelevant to the prediction task, so may be impossible to map backwards—though this is less true for robust classifiers [68]

the number of local dimensions that users must simultaneously consider (without sacrificing expressivity). We introduce benchmarks, algorithms, and metrics for this kind of hierarchical disentanglement.

Chapter 7

Evaluating the Interpretability of Generative Models by Interactive Reconstruction[1]

Before we can research methods to learn interpretable representations, we first need to operationalize what we mean by representation interpretability. While it would be nice to provide a simple, uncontroversial definition and measurement method in a few paragraphs, it turns out this is actually a very difficult problem in its own right. As such, we devote an entire chapter to the question, specifically for the case of generative models (with a focus on autoencoders).

7.1 Introduction

Many have speculated that machine learning (ML) could unlock significant new insights in science and medicine thanks to the increasing volume of digital data [87]. However, much of this data is unlabeled, making it difficult to apply many traditional ML techniques.

[1]This chapter is based on Andrew Slavin Ross, Nina Chen, Elisa Zhao Hang, Elena L. Glassman, and Finale Doshi-Velez. Evaluating the interpretability of generative models by interactive reconstruction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021. doi: 10.1145/3411764.3445296.

Generative modeling, a subfield of ML that does not require labels, promises to help by distilling high dimensional input data into lower dimensional meaningful axes of variation, which we call representations. To be most useful in "unlocking insights," though, these representations must be understood by human researchers.

Motivated by the need for human understanding, a burgeoning research area of interpretable ML has emerged [60, 79]. While some of this work has used user studies to quantify interpretability [100, 118, 178, 129], there have been concerns within the HCI community that these studies do not generalize to more realistic use cases [34]. These studies are also largely in the context of discriminative rather than generative modeling—even in the few that consider representations [14]. Within generative modeling, ML researchers have tried to quantify interpretability via measures of *disentanglement*, which measure how well individual representation dimensions match ground-truth factors that generated the data [184]. However, this work is not tested on actual human users, nor are disentanglement measures computable without knowing ground-truth factors.

In this work, we develop a method for quantifying the interpretability of generative models by measuring how well users can interactively manipulate representations to reconstruct target instances. To validate it, we use both MTurk and lab studies to determine whether models known to be understandable a priori can be distinguished from those known to be complex, and also whether our quantitative metrics match qualitative feedback from users. We also investigate the relationship between our human-grounded interpretability measures and synthetic disentanglement measures.

Our main contributions are as follows:

- A task for evaluating the interpretability of generative models, where users interactively manipulate representation dimensions to reconstruct target instances.
- Large-scale experiments on Amazon Mechanical Turk and smaller-scale think-aloud studies showing our task distinguishes entangled from disentangled models and that performance is meaningfully related to human model understanding, as demonstrated and reported by study participants.

- Novel results suggesting that ML methods which improve disentanglement on synthetic datasets also improve interpretability on real-world datasets.

7.2 Related Work

7.2.1 Human-Centered Interpretability Measures

While there have been criticisms that "interpretability" is ill-defined [142], several works have focused on quantifying it, particularly for discriminative models [141, 60, 72, 1, 129, 8, 178, 214, 198, 100, 118]. Per Doshi-Velez and Kim [60] and Miller [161], these works typically ground interpretability as the capacity of the model to be sufficiently understood in an appropriate context, and operationalize it as a user's ability to perform various tasks given visualizations of a model according to some performance measures—where the tasks and measures are presumed to be relevant to the desired context. They then study the effect of varying visualizations or models on task performance measures.

As a concrete example, Kulesza et al. [122] present different visualizations of a random forest [31] model, and as their task, ask users a series of questions in a thinkaloud-style protocol [136]. Their performance measure is the difference between the number of accurate and inaccurate statements about the model made by the user, which they compute by transcribing interviews and individually categorizing each statement. As their theoretical grounding, they draw on notions of understanding from Norman [167], who defines understanding in terms of user mental model accuracy. They find that visualizations produce more accurate mental models when complete (the whole truth) and sound (nothing but the truth), even if satisfying those conditions dramatically increases complexity.

More commonly, interpretability is operationalized as simulability: whether humans can use visualizations to predict the behavior of the model in new circumstances [141, 60, 100, 178, 214, 129, 128]. Though the theoretical grounding of this method is perhaps less clear than the mental model accuracy paradigm of Kulesza et al. [122] or the cognitive load paradigm of Abdul et al. [1], simulation tasks have the advantage of being model-agnostic

and easy to analyze programmatically, which allows them to be used in semi-automated human-in-the-loop optimization procedures such as Lage et al. [128]. However, such simulability tasks generally do not present "the whole truth" of the model at once. Our method extends the simulability paradigm from discriminative to generative modeling, but in a way that presents the whole truth of the model.

7.2.2 Interpretable Representation Learning

Background. Representation learning is generally concerned with finding ways of associating high-dimensional *instances*, which we denote x, with low-dimensional *representations*, which we denote z. Representation learning methods roughly fall in two categories: *generative modeling*, which maps low-dimensional representations z to high-dimensional instances x, and embedding, which maps high-dimensional instances x to low-dimensional representations z. Examples of generative models include Hidden Markov Models [75], Latent Dirichlet Allocation [29], and GANs [80]. Examples of embeddings include t-SNE [151] and the latent spaces of deep classification models. Examples of both simultaneously (i.e. approximately invertible mappings) include PCA [104] and autoencoders [95].

Disentangled representations and disentanglement measures. Disentangled representations [26, 57, 93, 46] seek mappings between high-dimensional inputs and low-dimensional representations such that representation dimensions correspond to the ground-truth factors that generated the data (which are presumed to be interpretable). To evaluate a model's disentanglement, many papers compute *disentanglement measures* with respect to known ground-truth factors. However, not only do there exist many competing measures [93, 45, 66, 145, 94, 204], but, to our knowledge, there exists no work that compares them to human notions of understandability. Our work both performs this comparison and provides a way to evaluate representation learning methods on real-world datasets, where we cannot rely on disentanglement measures because the ground truth is generally unavailable to us.

Explaining and visualizing representations. To be understood, a representation must be visualized or explained. For generative models, this usually means explaining each

dimension. Many works [113, 93, 46, 45, 110] show a "latent traversal" of instances with linearly varying values of a specific dimension and constant values of other dimensions. Others find or construct "exemplar" instances that maximize or minimize particular (combinations) of dimensions [168, 169, 9]. Another visualization technique, less common because it requires setting up an interactive interface, is to let users dynamically modify representation dimensions and see how corresponding instances change in real time, e.g. using sliders [84]. Related approaches have been explored for discriminative models with predefined meaningful features, e.g. Krause et al. [118], Wexler et al. [236], and Cai et al. [37], who use sliders to display exemplars matching user-defined concept activation vectors [109]. However, to our awareness, interactive manipulation of autonomously learned generative model dimensions has not been considered in an HCI context, especially to quantify interpretability. We use interactive slider-based visualizations as a subcomponent of our task, and test against baseline approaches using exemplars and traversals.

We do note there are other representation learning visualization methods that could be used in interpretability quantification but do not provide insight into the individual meanings of dimensions and are often specific to embeddings rather than generative models. Examples include Praxis [40], the embedding projector [215], and a parallel-coordinates-inspired extension by Arendt et al. [14], which further reduce embedding dimensionality down to 2D or 3D with PCA or t-SNE [151]. All of these methods are interactive and can help users understand the geometry of the data and what information the representation preserves, but they do not explain the dimensions of variation themselves. Bau et al. [22, 23] both visualize and quantify the interpretability of representations by relating dimensions to dense sets of auxiliary labels, but their approach (while highly effective) is not applicable to most datasets. Our focus is on quantifying how well users can understand representations on their own, in terms of their dimensions, without further projection or side-information.

7.3 Approach

We now define our proposed task for evaluating the interpretability of generative models, which we call "interactive reconstruction," and ground it as a measure of understanding.

7.3.1 The Interactive Reconstruction Task

Assume we are given a generative model $g(z)$: $\mathbb{R}^{D_z} \to \mathbb{R}^{D_x}$, which maps from representations z to instances x. Also assume we are given a distribution $p(Z)$ from which we can sample representations z (which may be approximated by sampling from a dataset $\{z^{(1)}, z^{(2)}, ..., z^{(N)}\}$), as well as a list of permitted domains $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2, ..., \mathcal{Z}_{D_z})$ for each dimension of z (e.g. a closed interval if z_i is continuous, or a set of supported values if z_i is discrete). Finally, assume we are given some distance metric $d(x, x')$ and threshold ϵ.

The task consists of a sequence of N_q questions. For each, we sample two representation values z, z' iid from $p(Z)$, which have corresponding instance values $x = g(z)$ and $x' = g(z')$. In an interface, we visualize x and x' along with their distance $d(x, x')$ and its proximity to the threshold ϵ. We also present manipulable controls that allow the user to modify each component of z within $\mathcal{Z}$ to interactively change $x = g(z)$ and thus the distance $d(x, x')$. The user's goal is to manipulate the controls as efficiently as possible to bring $d(x, x') \leq \epsilon$, thereby reconstructing the target. Once this condition is achieved, we repeat the process for newly sampled z and z'. If a user actively attempts to solve a question for a threshold amount of time T to no avail, we permit them to move on.

During each question, we continuously record the values of z, the errors $d(x, x')$, the dimensions i being changed, and the directions of modification (increasing or decreasing). These records let us precisely replay user actions and from them, derive a rich set of performance metrics, which we enumerate in Section 7.4.4. We hypothesize that, while the task will be possible to complete without understanding the model, users will perform it more reliably and efficiently when they intuitively understand what representation dimensions mean. We also hypothesize that the process of performing it will *teach* users these intuitive meanings when they exist—that is, when the model is interpretable.

The above is a general definition of the interactive reconstruction task. To apply it to a specific problem, however, a number of implementation choices must be made:

- **Control of representations** z: There are many different ways of inputting values for representation dimensions (e.g., numeric fields vs. sliders for continuous dimensions, or radio buttons vs. dropdowns for discrete dimensions), and also many different ways of arranging and annotating them (e.g., allowing reordering, grouping, and labeling, which can be helpful for higher-dimensional z).

- **Visualization of instances** x**:** Although some instance modalities have canonical visualizations (e.g., images), others can be visualized in many ways (e.g., patient medical records). Visualizations should make it easy to recognize the effects of changing z and compare whether x and x' are becoming closer with respect to $d(\cdot, \cdot)$.

- **Choice of distance metric** $d(x, x')$**, distance threshold** ϵ**, and time threshold** T**:** These critical parameters are best chosen together. For our experiments, we relied on small studies for each dataset, seeking $d(\cdot, \cdot)$ and ϵ that captured when users subjectively felt they had manipulated x to match x' sufficiently closely, and setting T to a round number on the order of twice the median duration.

We describe our dataset-specific choices in Section 7.4.3, but recommend these be re-tuned for new applications.

7.3.2 Theoretical Grounding for the Interactive Reconstruction Task

A number of sources in the HCI literature motivate and ground our interactive reconstruction task as a meaningful measure of understanding. First, as recommended by Kulesza et al. [122], our method attempts to present the "whole truth" and "nothing but the truth:" we visualize the entire model without any simplifying approximations. For generative models, visualizing the full model also means, to the greatest extent possible, visualizing dimensions jointly rather than separately (as is the case when using, e.g., latent traversals). We adopt this strategy in line with cognitive load theory as articulated by Sweller [223], who argues

that "understanding" emerges from explaining interactions between elements of a schema ("the basic unit of knowledge"), and that explaining interacting elements separately will make material harder to understand, rather than easier to understand, because of the split attention effect [41].

We also attempt to avoid split attention effects between the visualization and the proxy task. In many interpretability measurement methods, these two components are visually separate, e.g., in different portions of the screen [100, 128, 129, 178]. Such physical separation often makes it possible or even preferable to complete the task without engaging with the visualization, e.g., by guessing multiple-choice answers to complete the task more quickly. In recent HCI studies of interpretability tools, both Buçinca et al. [34] and Kaur et al. [106] note an inconsistency between outcomes in thinkaloud studies, where cognitive engagement with a visualization is forced, and practice, where the visualization can be readily ignored. By closely integrating the visualization and the task, we attempt to avoid this pitfall.

Finally, per Norman's articulation of the importance of feedback for building understanding in human-centered design [167], our task is structured to provide immediate, interactive feedback. Just from looking at the screen, it is always readily apparent to users whether they have gotten the right answer, and whether they are getting closer or further away. In contrast, for most simulation-proxy interpretability measurement tasks [178, 129, 100, 214], if users receive feedback at all, it is sporadic, e.g., about whether a multiple choice answer was correct.

7.3.3 Baseline: the Single-Dimension Task

As a baseline to our interactive reconstruction task, we also consider a "single-dimension" task, inspired by existing interpretability measurement methods for discriminative models (Section 7.2.1). Here, users are given an instance $x = g(z)$ and asked to guess the value of some hidden dimension z_i. To help them, users may view visualizations of other instances as dimension i is varied, shown in a different section of the screen. For each model, users are asked N_q questions about each dimension, and receive feedback after each question

about whether their answer was correct. Details for our dataset-specific implementations are in Section 7.4.5.

The single-dimension task is deliberately designed to violate our theoretical motivations; specifically, (1) it does not present "the whole truth" of the model, (2) it visualizes dimensions individually rather than jointly, (3) it spatially separates the visualization from the task, and (4) it provides feedback sporadically (after each question) rather than interactively.

7.3.4 Quantifying Quality of Interpretability Measurement Methods

Measuring the accuracy of any measurement method generally requires testing with a precisely known quantity. In this paper, we *assume ground-truth knowledge* that a particular model is more interpretable than another by working with synthetic datasets constructed for the express purpose of making this assumption reasonable, and supported by qualitative studies. We then test how well different interpretability measurement methods detect this assumed ground-truth difference. Where possible, we support our assumptions with quantitative measures of models' correspondence to ground-truth, e.g. disentanglement measures.

Our approach contrasts with the strategy of Buçinca et al. [34], who evaluate interpretability evaluation "proxy tasks" by how well they predict performance on downstream tasks. Although the ultimate motivation for interpretability research is to improve performance on downstream tasks ranging from better human+AI collaboration [20] to auditing for safety [39], performance on these tasks has no explicit correspondence to understanding. Additionally, in research settings, we often lack access to the true downstream tasks that motivate our work; thus our downstream validation task is *also* a proxy. So in effect, we are not evaluating whether a proxy task measures interpretability, but whether one proxy predicts another.

7.4 Implementation

We now describe how we implement our approach for a variety of representation learning models and datasets.

7.4.1 Datasets

We considered three datasets, visualized in Figure 7.1:

- **dSprites** [158], a 64×64 binary image dataset with five ground-truth factors of generation z: shape, rotation, size, x-position, and y-position. We chose dSprites due to its popularity in the disentanglement literature (Section 7.2.2).
- **Sinelines**, a 64-dimensional timeseries dataset we developed for this study. Each instance $x = \langle x_1, x_2, ..., x_{64} \rangle$ is a mixture of a line and a sine wave, generated by mapping: $x_t = z_1 t + z_2 + z_3 \sin(z_4 t + z_5)$, where $z_1 \sim \text{Uniform}(-1, 1)$ is slope, $z_2 \sim \mathcal{N}(0, 1)$ is intercept, $z_3 \sim \text{Exponential}(1)$ is amplitude, $z_4 \sim \text{Exponential}(1)$ is frequency, and $z_5 \sim \text{Uniform}(0, 2\pi)$ is phase. We make z 5-dimensional for consistency with dSprites, but make x a timeseries rather than an image to probe sensitivity to instance modality.
- **MNIST** [132], a popular benchmark in the ML and interpretable representation learning literature consisting of images of handwritten digits from 0 to 9. Although the MNIST dataset lacks ground-truth representations, it does contain labels indicating depicted digits.

7.4.2 Models

On dSprites and Sinelines, we tested the following models:

- **"Ground-truth" (GT)** generative models, constructed to have the same relationship between x and z as the (intentionally interpretable) process which generated the examples.

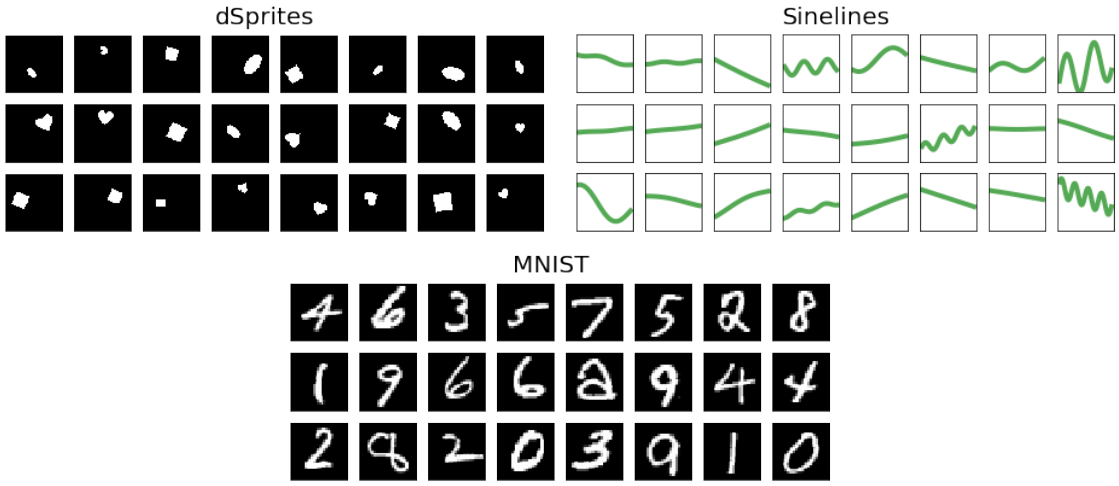


Figure 7.1: *Examples from the dSprites, Sinelines, and MNIST datasets.*

- **Autoencoders (AE)** [95], a baseline nonlinear representation learning method often considered to learn uninterpretable representations.
- **Variational autoencoders (VAE)** [113], which are similar to autoencoders but learn distributions over z, with a prior on z that can be interpreted as a regularizer. Because of this regularization effect, VAEs are widely reputed to learn more interpretable relationships between instances and representations than standard autoencoders.

Our primary goal was to validate that our task could distinguish entangled autoencoders from disentangled ground-truth models (both absolutely and relative to baselines).

Architecturally, for dSprites, we used the same 7-layer convolutional neural networks (CNNs) as Burgess et al. [35], one of the original papers introducing dSprites (with the GT model just using the decoder). For Sinelines, we used 256×256 fully connected networks with ReLU activations (except for the GT model, which was simple enough to implement in closed form). We tested all models at $D_z = 5$ to match ground-truth.

On MNIST, which has no ground-truth model, our goal instead was to test a broader set of popular interpretable representation learning methods that have never been tested with user studies. We again tested on AEs and VAEs, but also included the following models from the disentanglement literature:

- **β-TCVAEs (TC)** [45], a modification of the VAE that is trained to learn representation dimensions z_i that are statistically independent, generally considered near state of the art in the disentanglement literature.

- **Semi-supervised β-TCVAEs (SS)**, a modified β-TCVAE we explicitly train to disentangle digit identity from style (in a discrete dimension). Though we lack full ground truth, we expect SS to be less entangled than TC.

- **InfoGAN (IG)** [46], a generative adversarial [80] disentanglement method that also learns to disentangle digit identity from digit style, but imperfectly and without supervision.

We tested all of these MNIST models (AE, VAE, TC, SS, and IG) at $D_z = 5$ but additionally tested AE, TC, and SS at $D_z = 10$ to probe sensitivity to representation dimensionality. Architecturally, for all models, we use the same CNN architecture given in the papers introducing InfoGAN [46] and β-TCVAE [45] (Table 4), changing only the size of the latent dimension D_z. Training details were chosen to match original specifications where possible; see Section C.1 of the Appendix for additional details as well as model loss functions.

On dSprites and Sinelines, we quantified the extent to which our models matched ground truth with disentanglement measures. Specifically, we computed the DCI disentanglement score [66] and the mutual information gap (MIG) [45], which are commonly included in disentanglement papers. In addition to these overall scores, we computed pairwise mutual information to visualize dependence on a per-dimension basis. Figure 7.2 shows each of these metrics for each dimension and dataset. On both datasets, GT models are perfectly disentangled and AE models are heavily entangled. VAEs are somewhere in the middle, partially disentangling horizontal and vertical position from shape, scale, and rotation on dSprites, and partially disentangling linear from sinusoidal factors on Sinelines. As mentioned previously in Section 7.3.4, we use these metrics to support our assumption that GT models are more interpretable than AEs. On MNIST, we have no ground truth model, but we hypothesize that the semi-supervised (SS) model will be most interpretable due to

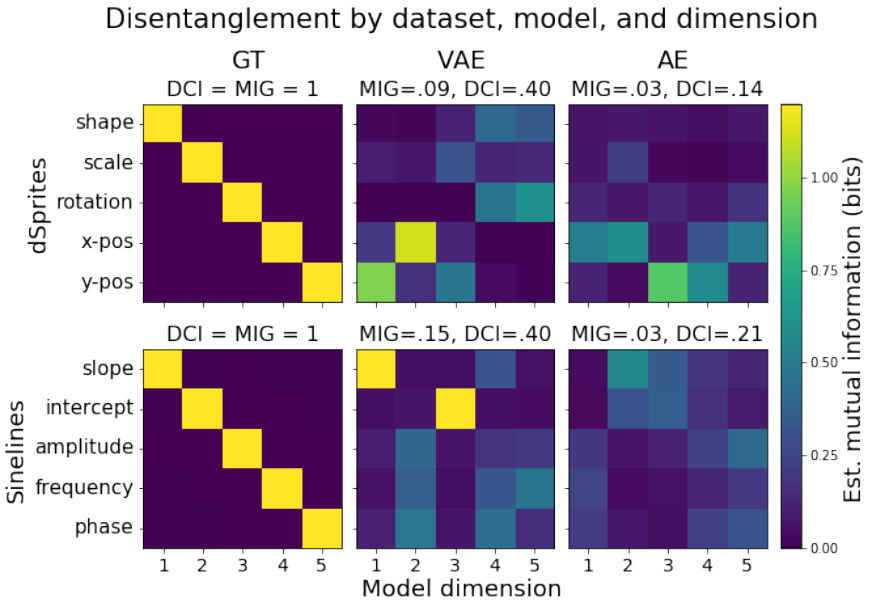


Figure 7.2: *Disentanglement scores (in plot titles) and pairwise mutual information (in heatmaps, approximated using 2D histograms) between true generative factors and representation dimensions. By construction, ground-truth (GT) models are perfectly disentangled, while VAEs learn to partially concentrate information about certain ground-truth factors into individual representation dimensions. AEs exhibit less clear relationships and have the lowest disentanglement scores.*

disentanglement between its continuous dimensions and ground-truth digit identity.

7.4.3 Interface

Figure 7.3 shows examples of the user-facing interface for this method, implemented for two different datasets. The following interface elements correspond to the interactive reconstruction task parameters described in Section 7.3.1:

Control of representations z: To specify closed-interval continuous dimensions of z, we use slider components, while to specify finite-support discrete dimensions, we use unlabeled radio buttons. We sampled initial z and target z' from $p(Z)$ set to the empirical distribution, or specifically a heldout testing split of the dataset. Slider ranges were determined by taking the empirical minimum and maximum values of each dimension z_i over this heldout split. The one exception to this procedure was for the InfoGAN model, where sampling and limits were determined from the prior.

Visualization of instances x: For images, visualization was straightforward, while for Sinelines, we used line charts with dashed lines at $y = 0$ and appropriate limits. In addition to visualizing x and x' side-by-side, we provided an option to overlay them with partial transparency, which we found was helpful in pilot experiments for fine-tuning. This defaulted to on for our synthetic experiments, but was defaulted to off for MNIST after pilot users expressed the side-by-side view was more helpful. For MNIST in particular, to facilitate remembering (and recording) user impressions of dimension meanings, we allowed users to input custom labels next to the corresponding controls in the interface.

Distance and time thresholds: For each dataset, we defined $d(x, x')$ as a Jaccard distance [135], i.e., the fraction of disagreeing dimensions of x and x' to total active dimensions of x and x', with dataset-specific definitions of agreement and activity. Exact expressions are given in Section C.2 of the Appendix. Although these choices worked well for black and white images and consistently-scaled timeseries, different metrics might be necessary for other data modalities; we discuss this further in Section 7.7. Because $d(x, x')$ was between 0 and 1, we visualized it as an agreement percentage rather than a distance, and chose $\epsilon = 0.1$ (or a 90% agreement threshold) for synthetic datasets and $\epsilon = 0.25$ (or a 75% agreement threshold) for MNIST.

The time threshold T for skipping questions was set to 30 seconds for dSprites and Sinelines and 45 seconds for MNIST. We paused this hidden countdown whenever users were inactive for more than 3 seconds.

We implemented tasks as single-page, client-side web applications, with machine learning models running directly in users' web browsers after being converted to TensorFlow.js [217]. Despite the fairly large size of certain models (e.g. 7-layer convolutional neural networks), only two users out of hundreds reported problems running them interactively in-browser. Links to the task for all models can be found at `http://hreps.s3.amazonaws.com/quiz/manifest.html`.

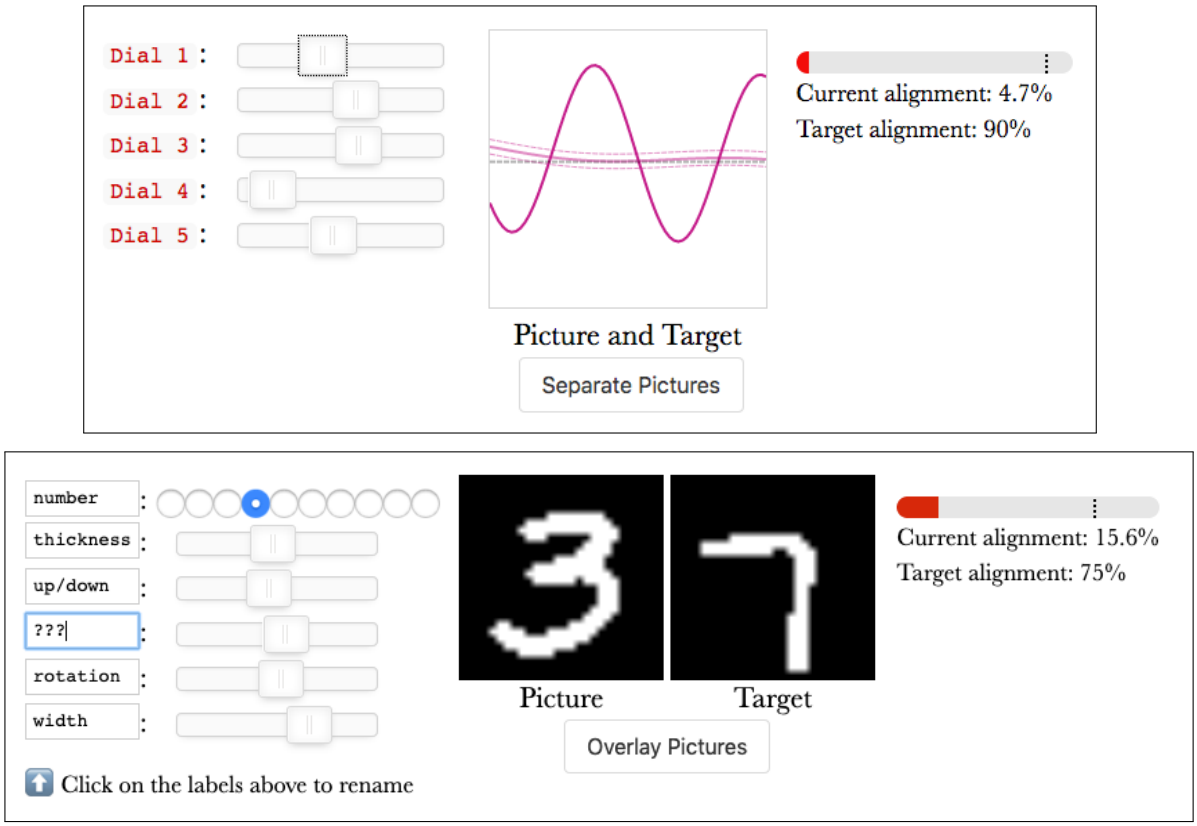


Figure 7.3: *Screenshots of the interactive reconstruction task on Sinelines (top, with x and x' overlaid and dotted lines indicating the region of allowable alignment) and MNIST (bottom, with separated x and annotations for z).*

7.4.4 Interactive Reconstruction Metrics

We computed the following performance metrics from our records of users performing the interactive reconstruction task:

- **Completion rate:** For each model, we measure the fraction of questions the user solves, i.e. moves the sliders and/or chooses radio buttons to make $d(x, x') \leq \epsilon$, rather than skipping after T seconds.
- **Response time:** For each model, we measure the average time it takes participants to complete each question.
- **Slide distance:** For each model, we measure the average total distance moved by sliders, in units of full slider widths, for each question. Discrete changes count as 1.

- **Error AUC:** We measure the "area under the curve" of mean squared difference between x and x' over the total time the user is attempting to solve the quiz.
- **Self-reported difficulty:** After answering a "stage" of questions about a model, users rate that difficulty from 1 to 7 using a Single Ease Question (SEQ) [195].

We store snapshots of x and z every 100ms while the value is changing in a constant direction, or whenever the active direction or dimension changes. While this list is not an exhaustive enumeration of all possible metrics applicable to the task, others can be defined and computed post-hoc as we store a nearly complete record of user actions over time.

7.4.5 Single-Dimension Task Parameters

In this section, we describe how we instantiated the single-dimension task for the synthetic datasets. This task has two primary implementation parameters: the prediction task and the visualization.

For the prediction task, we opted for a multiple-choice classification task where, for a particular example x visualized in the same way as in the interactive reconstruction task, users decided whether z_i is "Low," "Medium," or "High." These regimes were respectively defined in terms of the 1st-5th, 48th-52nd, and 95th-99th percentiles of the marginal distribution of encoded x in a held-out split of the original dataset. We sampled z by first sampling from the empirical joint distribution (i.e. a heldout split of the dataset), then overriding z_i to a value selected uniformly from one of these regimes. Users answered two questions for each dimension i, and received the correct answer as feedback.

For the visualization, which is the sole task component dependent on the model, we tested two versions of the task, one using *latent traversals* and the other using *synthetic exemplars*, as described in Section 7.2.2. For traversal visualizations, for 5 randomly sampled values of z, we plotted x values corresponding to overriding z_i to 7 linearly spaced values between "Low" and "High" as defined above. For exemplar visualizations, we showed "Low," "Medium," and "High" bins with 8 examples each. Both of these visualizations force

users to generalize from finite samples, which can lead to ambiguities if randomly sampled z are not diverse. To mitigate this potential problem, we provided users with a "Show More Examples" button. Screenshots for the single-dimension task on the dSprites dataset are shown in Figure 7.4.

7.4.6 Single-Dimension Metrics

We recorded several performance metrics specific to the single-dimension task:

- **Correctness rate:** For each model, we measured the percentage of questions the participant answered correctly.
- **Self-reported confidence:** For each model, we measured whether users agreed with the statement "I'm confident I'm right" on a 5-point Likert scale.
- **Self-reported understanding:** Similar Likert measurement, but for "The dial makes sense." (We referred to dimensions as "dials" in the interface.)

In addition to these task-specific metrics, we also recorded response time and self-reported difficulty with the SEQ, which were shared in common with interactive reconstruction.

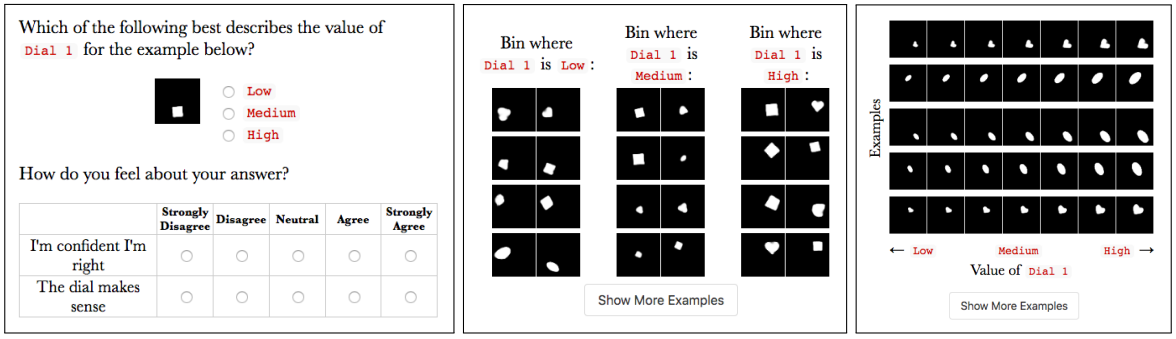


Figure 7.4: *Screenshots of the dSprites single-dimension task (left), showing ground truth visualizations with exemplars (center) and traversals (right).*

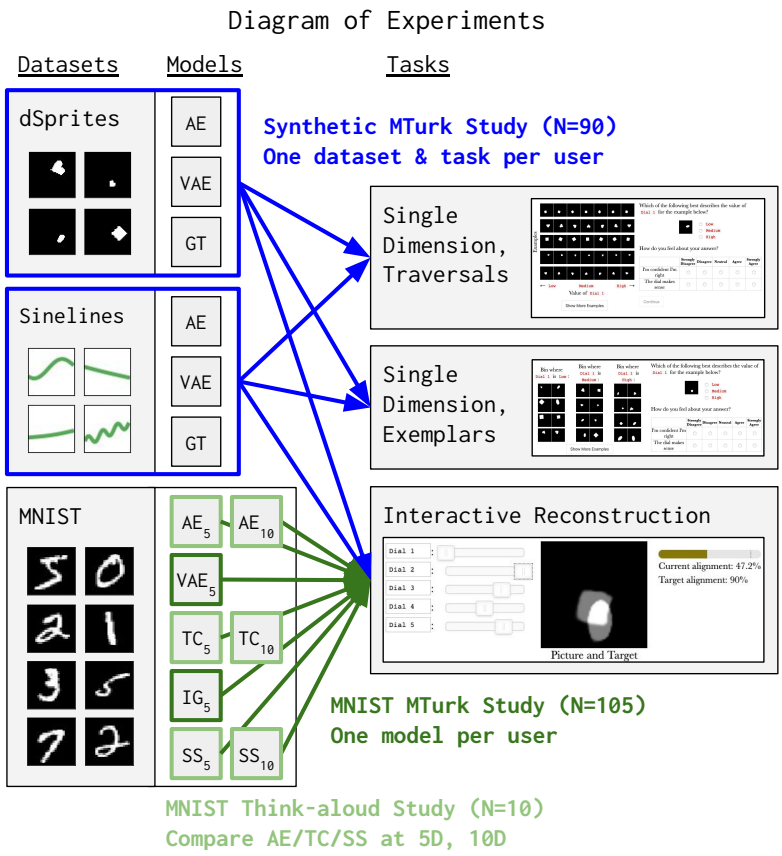


Figure 7.5: *Diagram of experiments performed (except the small synthetic pilot). On synthetic datasets, experiments were designed to evaluate the best task (assuming the best model). On MNIST, experiments were designed to evaluate the best model (assuming the best task).*

7.5 Study Methodology

With implementation decisions now specified, we describe the four studies we ran, two larger-scale studies on MTurk and two smaller-scale lab study sessions (see also Figure 7.5 for a graphical depiction). All experiments began with a consent form, a tutorial, and practice questions on an easy example. Experiments were deployed on the web and are available at `http://hreps.s3.amazonaws.com/quiz/manifest.html`. Code for our study is available at `https://github.com/dtak/interactive-reconstruction`.

7.5.1 Experimental Design

Synthetic Think-aloud Pilot

We began by running a small pilot study with $N = 3$ users (referred to as U1, U2, and U3) to test out each possible task on the dSprites and Sinelines synthetic datasets (interactive reconstruction, single-dimension with exemplars, and single-dimension with traversals), with each user completing two of the six task/dataset conditions with two of the six models (drawn randomly without replacement to ensure complete coverage). Participants were asked to think aloud and describe their strategies for solving each problem. Interviews were recorded and transcribed, and the feedback was used to make minor clarifying changes to the interface.

Synthetic MTurk Study

We then ran a larger version of the synthetic dataset study on MTurk, where each participant ($N = 15$ per dataset and task, and $N = 90$ total) completed one of the six possible dataset/task conditions, but for all three models (AE, VAE, and GT, with the order randomized). To keep overall quiz length manageable, N_q was set to 5 for interactive reconstruction and 10 for single-dimension tasks (that is, 2 questions per dimension). Differences were analyzed with repeated-measures ANOVA and paired t-tests.

MNIST Think-aloud Study

We next ran a think-aloud study with $N = 10$ participants on MNIST, testing out a wider variety of models but only for the interactive reconstruction task. Each participant (whom we refer to as P1-10) completed interactive reconstruction for three models (AE, TC, and SS, order drawn randomly without replacement), with odd and even-numbered participants working with 5D and 10D representations respectively. Model-specific stages ended after 7 questions had been completed or 10 minutes had elapsed. After each stage, in addition to the single ease question [195], users entered raw NASA-TLX scores [88] and answered

Likert scale questions about whether they felt they understood representation dimensions. Differences were analyzed with repeated-measures ANOVA and paired t-tests. During each stage, users were also encouraged to label dimensions if possible and continuously describe their impressions of the task, which we recorded.

MNIST MTurk Study

Finally, we ran a large-scale MTurk study on MNIST. As with the synthetic dataset study, we had $N = 15$ participants per condition (for $N = 105$ participants total). However, instead of showing all models and varying the task, we limited the task to interactive reconstruction and varied the model, specifically showing each participant one of AE_5, VAE_5, IG_5, TC_5, SS_5, AE_{10}, or SS_{10}. N_q was increased from 5 to 7 for additional signal and to make the task sufficiently long. Differences were analyzed with one-way ANOVA and independent t-tests.

7.5.2 Recruitment

Think-aloud Studies

For our think-aloud studies, we recruited undergraduate and graduate students from computer science mailing lists at an academic institution, and compensated participants with Amazon gift cards ($15/hour for synthetic pilot study sessions, which we increased to $20/hour for MNIST to incentivize recruitment for a larger study). We recruited $N = 3$ participants for the synthetic pilot and $N = 10$ for MNIST.

On the synthetic datasets, two participants were male, one was female, and all were graduate students. On MNIST, two were male, eight were female, three were undergraduates, and seven were graduate students. All participants in both studies were aged 18-34.

MTurk Studies

For each MTurk experimental condition (6 for synthetic datasets and 7 for MNIST), we recruited $N = 15$ participants on Amazon Mechanical Turk with unique worker IDs. This

translates to a total of 90 participants on the synthetic dataset study and 105 participants for the MNIST-based study.

For single-dimension tasks, participants were excluded if they answered practice questions incorrectly. For traversal visualizations, the retention rate was 71%, while for exemplars, it was 91%. For interactive reconstruction, we included all participants who successfully completed the study on synthetic datasets (as users could not proceed past the practice questions without answering them correctly), but added an additional inclusion criteria on MNIST requiring participants to have reconstructed at least one of seven instances successfully. We adopted this criteria in part because we noticed several participants did not realize they could use radio buttons (which were not included in the practice questions) on the IG and SS tasks (never changing them during any question), and in part to filter out users who gave up completely on extremely hard models (e.g. the 10D AE). Retention rates were highest (88%) for the SS_{10} model and lowest (42%) for the AE_{10}, with an overall average of 67%. In total, 156 participants were needed to get 105 retained participants on MNIST and 115 participants were needed to get 90 retained participants on synthetic tasks.

Compensation was set with the intention of ensuring that actively engaged MTurk users would earn at least \$15/hr, with time estimates based on pilot experiments and interactive reconstruction time limits. On the synthetic datasets, users were paid \$6 for both single-dimension and interactive reconstruction tasks, which took an average of 13 minutes (\$27/hr). The interactive reconstruction experiments took a slightly longer average of 18 minutes (\$20/hr), with 95% of participants finishing under 34 minutes (\$11/hr). For the MNIST dataset, users were paid \$3.75 for completing tasks, which took an average of 15 minutes (\$15/hr), with 80% of participants finishing under 20 minutes (\$12/hr) and 95% finishing under 38 minutes (\$6/hr).

Demographic information was recorded in a post-quiz questionnaire. In the synthetic study, participants were generally young (58% between 18-34), North American (68%, with most others from Asia or South America), college-educated (79% had Bachelor's degrees or above) and male (64%), with gender recorded per guideline G-4 of [197]. Demographics

were similar on MNIST (49% aged 18-34, 59% North American, 71% male, and 81% with Bachelor's or above).

7.6 Results

Since the think-aloud and MTurk study participants worked with the same datasets and completed similar or identical tasks, we will first present results from all the synthetic dataset studies and then move on to results from the MNIST dataset studies. Adding up ANOVAs and t-tests across all our experiments and pairs of models, we ran 435 statistical tests, giving us a Bonferroni-corrected threshold of 0.00011 for an initial α of 0.05.

7.6.1 Synthetic Study Results

MTurk, Interactive Reconstruction

As shown in Figure 7.6, **the interactive reconstruction task clearly differentiated ground truth (GT) models from AEs.** Completion rates were significantly different on both dSprites ($F_{2,28} = 32.3, p<0.0001$) and Sinelines ($F_{2,28}=23.4, p<0.0001$) in the directions we hypothesized. GT model completion rates on both dSprites (.75±.26) and Sinelines (.83±.19) were significantly higher than AE completion rates (.15±.19, $t_{14}=8.5$, $p<0.0001$ on dSprites and .34±.26, $t_{14}=6.4$, $p<0.0001$ on Sinelines). VAE models, though only marginally significantly different, were in the middle (.51±.33 on dSprites and .63±.28 on Sinelines). These results closely mirror the disentanglement metrics in Figure 7.2.

We also saw significant differences in self-reported difficulty ($F_{2,28} = 32.3, p<0.0001$ on dSprites, $F_{2,28}=11.7, p=0.0002$ on Sinelines) and error AUC ($F_{2,28}=31.6$, $p<0.0001$ on dSprites, $F_{2,28}=14.1$, $p<0.0001$ on Sinelines). Between AE and GT models, these differences were large and least marginally significant. VAEs were no longer exactly in the middle, however. Instead, they more closely matched the AE model on dSprites and the GT model on Sinelines—e.g. for difficulty, average scores for the AE, VAE, and GT were 6.9, 6.4, and 4.2 on dSprites vs. 5.9, 4.1, and 4.0 on Sinelines.

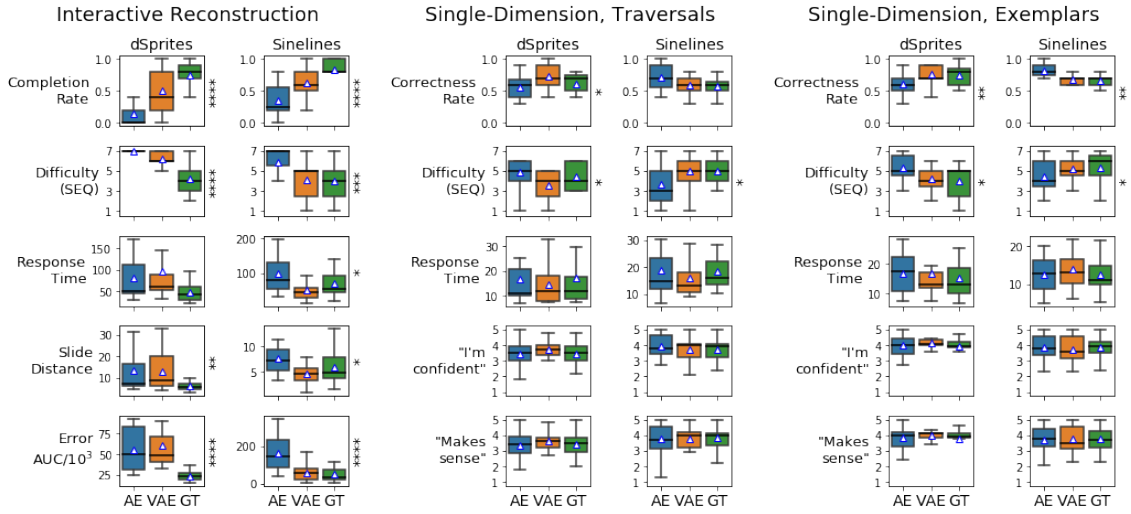


Figure 7.6: *Boxplots of MTurk dependent variables across the 15 participants for each synthetic data task condition. Blue triangles indicate means, black lines indicate medians, and stars indicate p-values for differences between means (****=p<0.0001, ***=p<0.001, **=p<0.01, *=p<0.05). Expected differences between AE and GT models emerge clearly from interactive reconstruction results, but not single-dimension results.*

Think-aloud, Interactive Reconstruction

Interactive reconstruction task users felt they understood GT models, did not understand AEs, and partially understood VAEs in a way that matched the disentanglement scores in Figure 7.2. On dSprites, U1 assigned meanings to GT model dimensions almost immediately, noting on the first question that *"[dial 1] is changing its shape, [...] dial 2 is changing the size, [...] and then 3 seems like it's changing the rotation, and 4, probably x-position. And then 5 is y-direction I guess?"* For the AE, however, U1 *"d[id]n't understand what the dials [we]re doing."* On Sinelines, U2 found the GT model *"much easier to figure out"* than the AE, where it was *"hard to tease out exactly what each [dimension] is doing."* The difficulty of understanding the AE model caused them to switch strategies from *"match[ing] [dimensions] sequentially"* (which they could do with the GT model) and instead *"look[ing] at the current alignment and then mak[ing] sure that number is increasing"* by *"tweak[ing] the dials and see[ing] what happens."*

Users partially understood VAEs. On the Sinelines VAE, U2 reported they could *"quickly figure out"* there *"was one dial that [controlled] slope"* and another that *"did [vertical] translation"*,

but that *"squiggliness"* was *"quite hard."* These reports match the dependence plots in Figure 7.2, which show that the Sinelines VAE disentangled linear slope and intercept from sine wave parameters. On the dSprites VAE, U1 felt *"dials 1, 2 and 3 are more predictable"* after determining *"dial 3 is changing the size"* while *"dials 1 and 2 are moving the object in the diagonal way."* These comments again matched the dependence plots in Figure 7.2, which suggest position and size are controlled by VAE dimensions 1-3, while shape and rotation are controlled by dimensions 4-5. However, U1 did find the *"diagonal"* relationship (which was also nonlinear) confusing, which forced them to randomly experiment more within each group of dimensions. In contrast, for U2, linear slope and intercept were disentangled both from sine wave dimensions and each other. This difference may explain why, relative to the GT, VAEs had higher MTurk difficulty ratings and error AUCs on dSprites than Sinelines (despite having similar DCI scores).

MTurk, Single-Dimension

The single-dimension task did not clearly differentiate models. In the MTurk results, no differences in any metrics were significant at $\alpha = 0.00011$, though correctness rates for the exemplar-based version of the task were closest ($F_{2,28}$=5.8, p=0.008 for dSprites and $F_{2,28}$=6.6, p=0.005 for Sinelines, though the most significant relationships were in the wrong direction). Response time and self-reported confidence/understanding were almost identical across models, datasets, and visualizations. On dSprites, we did observe that GT models had higher correctness rates (.75$\pm$.14) and lower difficulty ratings (4.0$\pm$1.6) than AEs, which had .61$\pm$.18, p=.01 for correctness and 5.3$\pm$1.3, p=.03 for difficulty (on exemplars). However, on Sinelines, AEs actually emerged with the *highest* average correctness rates in the exemplar-based MTurk study (AE=.81$\pm$.15 vs. GT=.65$\pm$.11, p=0.003), as well as the lowest difficulty (AE=4.5$\pm$1.6 vs. GT=5.3$\pm$1.5, p=0.003). Our qualitative results below (as well as the dimension-by-dimension correctness rate breakdown in Figure 7.7) suggest this is not because users understood the AE, but because of helpful AE pathologies and unhelpful GT symmetries.

Think-aloud, Single-Dimension

Users performed the task without trying to understand the model. Although U1 initially tried to formulate *"hypotheses"* about the meanings of dimensions, they found these *"irrelevant for [them] to make decisions."* U2 described their strategy as *"match[ing] directly"* without trying to *"tease apart the different dimensions,"* and P3 called it *"blind trust"* in *"visual matching."*

Visualization pathologies also biased the results in directions unrelated to understanding. On Sinelines, we noticed a lack of diversity in "High" and "Low" samples from AEs, which sometimes made the matching problem trivially easy for all participants. For GT models, we also noticed questions for particular were pathologically hard due to symmetries in the generative process. One example is that for the rotation feature on dSprites and the phase feature on Sinelines, "Low" values near 0 and "High" values near 2π were nearly indistinguishable. For GT models on Sinelines, participants also found it hard to detect changes in amplitude at low frequencies, as these could be alternately explained by slope, and harder still to detect changes in frequency at low amplitudes, as near-linear examples were effectively unaltered.

Exemplars vs. Traversals.

Though not central to our narrative, we found some evidence that exemplars may have been more effective than traversals for single-dimension performance. U3, who completed versions with both visualizations, reported that the task was easier with exemplars because seeing instances clustered into bins *"visually help[ed them] understand the three categories,"* whereas with the traversal visualization, it was necessary to *"mentally create"* those categories by imagining spatial *"divisions."* U3 also hypothesized it was easier for them to detect patterns on the right-hand side of traversal visualizations due to their experience *"reading from left to right,"* which they were concerned might introduce *"a bias"* in their answers. We see such evidence of spatial bias in Figure 7.7, whose bottom-left plot shows that MTurk users struggled to answer traversal questions about dSprites x-position, where the

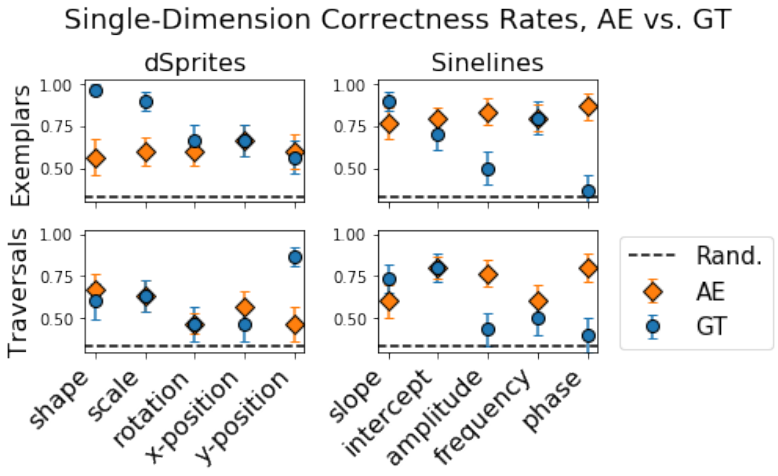


Figure 7.7: *AE and GT correctness rates by representation dimension for MTurk single-dimension tasks. Dots show means with standard errors, x-axis shows GT dimensions (AE order is arbitrary). Compared to AEs, some GT dimensions had much lower or higher correctness rates than others, rising to near 100% for attributes like shape, scale, and slope and falling to near 33% (the rate of random guessing) for others, especially periodic attributes like rotation and phase. These differences suggest that questions about certain conceptually simple dimensions were pathologically difficult to answer from static visualizations.*

movement of the shape was in the same direction as the movement of the images, but answered questions about y-position almost perfectly, where the movement of the shape was orthogonal to the movement of images. Meanwhile, accuracy on these questions for exemplars (top-left plot) was intermediate in both cases, showing less negative bias for x-position though also less positive bias for y-position. Overall, these results suggest that visualization details matter for performance, but also that it may be better to utilize time rather than space when visualizing changes in spatial features, which is effectively what occurs with interactivity.

7.6.2 MNIST Study Results

MTurk, Interactive Reconstruction

As shown in Figure 7.8, **Interactive reconstruction metrics clearly distinguished models, especially AE vs. SS.** In general, we found significant differences between models (F_{28}=4.7, p=0.0003 for completion rate, F_{28}=4.9, p=0.0002 for difficulty, and F_{28}=11.4, p<0.0001

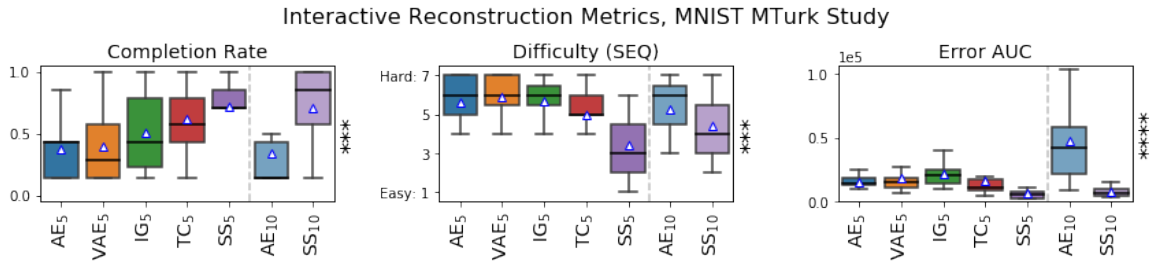


Figure 7.8: *Boxplots across the 15 MTurk participants for each MNIST model, for dependent variables that were significant in the synthetic tasks. Blue triangles indicate means, black lines indicate medians, and stars indicate significance as in Figure 7.6. As with the qualitative studies, semi-supervised (SS) models performed best by each measure, while standard autoencoders (AE) performed near the worst, especially at higher representation dimensions (right).*

for error AUC), with the sharpest pairwise contrasts being significant or nearly significant differences between AE and SS models. For example, at 5D for the AE vs. SS, completion rates were 0.38 ± 0.23 vs. 0.72 ± 0.24 (t_{28}=-3.9, p=0.0006), subjective difficulty was 5.6 ± 1.6 vs. 3.4 ± 1.5 (t_{28}=3.7, p=0.001), and error AUC (given in units of 1000s for conciseness) was 15.7 ± 4.3 vs. 6.3 ± 4.7 (t_{28}=5.5, p<0.0001). The next-best performing model was the β-TCVAE (TC), with an average completion rate of 0.61 ± 0.27, subjective difficulty of 5.0 ± 1.5, and error AUC of 16.0 ± 14.0. Although the average error AUC was higher for the TC model than the AE, this was largely due to a small number of extreme outliers (note the high variance). Comparing medians, we have 10.5 for the TC model and 14.5 for the AE model, which matches the overall trend of AE < TC < SS. Results for the IG and especially the VAE were generally slightly worse and closer to AE performance.

Performance degraded when increasing dimensionality, but not as badly for methods thought to be interpretable. When we increased D_z from 5 to 10, average completion rate fell (0.38→0.35 for the AE, 0.73→0.71 for the SS), error AUC rose (15.7→47.8 for the AE, 6.3→7.6 for the SS), and subjective difficulty generally rose (3.4→4.4 for the SS, though it fell slightly from 5.6→5.3 for the AE), though none of these results were near significance except the AE's increase in error AUC (t=-3.7, p=0.002). Examining medians, however, we find that median completion rate fell dramatically for the AE (3/7→1/7, the minimum possible value for retention), while for the SS it actually *rose* (5/7→6/7), suggesting that

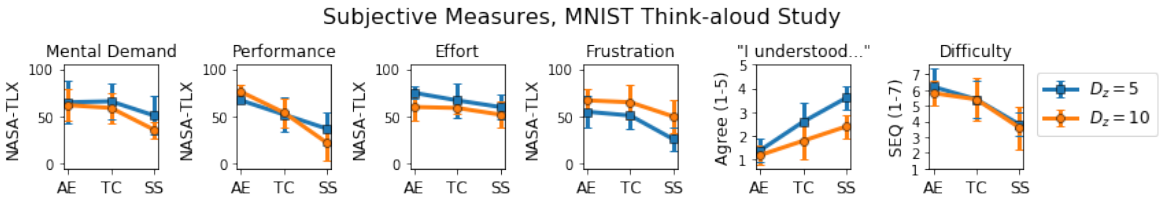


Figure 7.9: *Subjective measures for the MNIST think-aloud study (means ± standard deviations). The first four measures are NASA-TLX scores [88], the fifth is users' subjective agreement (from 1-5) with the statement "I understood what many of the dimensions meant," and the last is a single ease question (SEQ) [195] assessment of difficulty from 1-7. Users rated SS models easiest across all measures, though for higher D_z, subjective understandability was lower, frustration was higher, and perceived performance gaps were greater.*

degradation from increasing dimensionality was more drastic for the AE than the SS.

Think-aloud, Interactive Reconstruction

Multiple sources of evidence suggest users understood the SS well and the AE poorly. The first source is subjective measures, which are plotted in Figure 7.9. Though not always significant at $N = 5$, they suggest not just that the difficulty and cognitive load of the task varied across models (with AE generally rated hardest and SS easiest), but also that dimension understandability varied in the same way, e.g., with SS rated most understandable at 5D with 3.6±0.5 and AE rated least understandable with 1.4±0.5 (t_4=11.0, p=0.0004). When increasing D_z to 10, frustration grew and subjective understanding of SS models fell (to 2.4±0.5, t_4=6, p=0.004), but relative differences remained fairly consistent.

The second source of evidence is agreement between user-entered dimension labels, shown for $D_z = 5$ in Figure 7.1. On average, only 20% of users were able to assign labels to any AE dimensions, but 100% were able to do so for at least one SS or TC dimension, with all dimensions at $D_z = 5$ and many at $D_z = 10$ having at least 80% coverage. Additionally, for many SS dimensions, these independently-assigned labels agreed closely. TC model labels were less consistent, suggesting more entanglement with the digit.

The third source of evidence is users' verbal descriptions. None of the 10 participants made any comments indicating that they found the AE comprehensible, with P3 commenting that "*the dials didn't have any discernible meaning*", and P8 concluding that "*the meanings of*

Table 7.1: *Labels assigned to MNIST $D_z = 5$ representation dimensions.*

Model	Dimension	User-Assigned Labels
AE_5	Continuous 1	"791"
	Continuous 2	"thickness"
	Continuous 3	"rotate"
	Continuous 4	"stretch"
	Continuous 5	—
TC_5	Continuous 1	"thickness", "0-1", "closedness", "2-0"
	Continuous 2	"shrinking horizontally", "paint white", "0-1"
	Continuous 3	**"diagonal", "rotate"**, "4-5-3"
	Continuous 4	"add bottom left", **"3-5-6", "3-5-blob"**
	Continuous 5	"ccw", **"6-7", "7-6?"**
SS_5	Discrete 1	**"number", "Digit", "Class", "digit"**
	Continuous 1	"up", "Widthish", "paint white", "rotate cw, change focus"
	Continuous 2	**"ccw", "LR Skew", "Rotation", "rotate", "skew rotate cw/ccw"**
	Continuous 3	"curve", **"TB Skew", "up down bias", "focus up/dwn"**
	Continuous 4	"thickness", **"Wide", "width", "horiz thickness somewhat"**
	Continuous 5	"left", **"Thick", "paint black", "line thickness/focus but white"**

All labels assigned to $D_z = 5$ models by participants in the MNIST thinkaloud study. Bold text shows labels experimenters identified as consistent between participants. The semi-supervised (SS) model had both the most labels and the most consistent labels, while the autoencoder (AE) had the fewest. Labels for $D_z = 10$ are given in Table C.1 in the Appendix.

the dials aren't helpful in this one for solving the problem most efficiently." In contrast, 9/10 participants felt that the SS model was "*the easiest to understand and label*" (P3), with 100% agreeing it was more comprehensible than the AE. This was in large part "*because it [...] let you choose the number*" (P4, with all participants commenting on this in some manner), though 8/10 participants also expressed that the continuous dimensions were at least partially understandable. For example, P1 noted that they "*understood what most of the dimensions were doing, especially rotating*." However, unlike with the synthetic GT models, no participants claimed complete understanding; for example, P4 felt "*there were some dials that were easy to tell, but there were others that were more obscure*."

The TC model consistently fell between the others, with 8/10 commenting that it was more comprehensible than the AE, and 9/10 commenting that it was less comprehensible than the SS. Subjective descriptions varied; compared to the AE, P9 felt able to "*understand and write down and remember*" what TC dials meant, despite being "*worried*" that "*the framework [they] built might not be accurate*." P7 felt they could "*understand physically*" what

certain dimensions were "*trying to do*" but had trouble expressing it verbally; they described it as being like a "*matching game*" where "*you know what you mean but there's not an agreed upon word for it*."

Differences in understanding led to differences in strategy. When users understood dimensions, they used that understanding to perform the task more efficiently. On the SS model, P5 "*used the dials that [they] understood first*" and "*had an idea of which way to move*" them, while P8 felt it was "*pretty easy to know which [SS] dial to pick*." P7 was able "*to be intuitive*" when they "*could figure out what the dials meant*." P1 even felt a degree of mastery, saying that for the SS model, they "*had it down to a more exact science*."

In contrast, when users did not understand dimensions, they gravitated towards an inefficient but less cognitively taxing strategy similar to gradient ascent. P3 described this strategy as trying to "*watch one slider and look at the alignment number of see if it reaches a local max*," with some participants "*proceeding through dials one at a time*" (P10) and others selecting them in "*a very random order*" (P8). Users generally did not generally want to resort to this strategy; for the AE, P9 started off by putting in significant effort "*trying to see if [they] could find any understanding of what the dials meant*" but "*it didn't work*," and with reluctance, their "*strategy changed*" to "*finicking with the dials*." However, P8 felt that for complicated models, the gradient ascent strategy was actually "*less mentally demanding*" because "*not even trying to figure out*" dimension meanings meant the task was sufficiently mindless that they could "*do something else at the same time*," such as "*hold a conversation*." Users felt this was inefficient, with P2 commenting that "*this is definitely not the fastest strategy*," but when "*there wasn't any easy way to define anything [...] to do it systematically feels really annoying*" (P8). These comments may help to explain why the differences in subjective measures of effort were less significant than differences in frustration, understanding, and performance.

7.7 Discussion

As shown in Figure 7.6 and Table 7.2, **interactive reconstruction metrics differentiated entangled and disentangled models, both absolutely and relative to baselines.** On both

Table 7.2: *Representation learning model metrics and links.*

Model Information			Disentanglement		Interactive Reconstruction			
Dataset	Model	MSE	DCI [66]	MIG [45]	Completion %	Difficulty	Error AUC/10^3	Link
dSprites	AE_5	**6.5**	0.14	0.03	0.15±0.19	6.93±0.25	55.8±25.6	
	VAE_5	8.1	0.40	0.09	0.51±0.33	6.20±0.91	60.1±26.6	
	GT_5	8.9	**1.00**	**1.00**	**0.75±0.26**	**4.20±1.60**	**23.6±5.9**	
Sinelines	AE_5	0.6	0.21	0.03	0.34±0.26	5.87±1.67	163.5±90.0	
	VAE_5	3.3	0.40	0.15	0.63±0.28	4.07±1.84	59.6±47.8	
	GT_5	**0.0**	**1.00**	**1.00**	**0.83±0.19**	**4.00±1.71**	**52.0±48.9**	
MNIST	AE_5	**15.5**	—	—	0.38±0.23	5.60±1.62	15.7±4.3	
	VAE_5	16.2	—	—	0.40±0.31	5.87±1.15	18.6±13.7	
	IG_5	—	—	—	0.51±0.31	5.67±1.30	21.3±8.6	
	TC_5	25.4	—	—	0.62±0.26	5.00±1.41	16.0±13.1	
	SS_5	20.8	—	—	**0.73±0.24**	**3.40±1.54**	**6.3±4.7**	
MNIST	AE_{10}	**7.2**	—	—	0.35±0.29	5.27±1.73	47.8±34.0	
	TC_{10}	24.9	—	—	—	—	—	
	SS_{10}	20.7	—	—	**0.71±0.29**	**4.40±1.40**	**7.6±3.3**	

MTurk interactive reconstruction metric means and standard deviations along with general metrics for each of the models used in all experiments. Mean squared error (MSE) measures autoencoders' errors reconstructing inputs not seen during training. DCI and MIG are measures of disentanglement applicable to synthetic datasets. Bold entries indicate least error / greatest interpretability or disentanglement in each group. Links go directly to tasks for each model (skipping instructions).

synthetic datasets, these metrics (specifically completion rate, difficulty rating, and error AUC) clearly distinguished the entangled AE from the disentangled GT model. Single-dimension task metrics, on the other hand, showed a much less clear relationship with disentanglement, sometimes predicting that the highly entangled autoencoder was more interpretable than ground truth. On MNIST, interactive reconstruction metrics continued to meaningfully differentiate models in a manner consistent with hypotheses from the disentangled representations literature (e.g. β-TCVAEs especially semi-superivsed β-TCVAEs were more interpretable than AEs and standard VAEs).

Interactive reconstruction metrics measured understanding. This is a major claim, but our results suggest that performing the task helped users understand models, and that understanding models helped users perform the task. First, we know that after performing interactive reconstruction, users felt they understood the ground-truth interpretable models much more than unregularized autoencoder models. On MNIST, where there was no GT

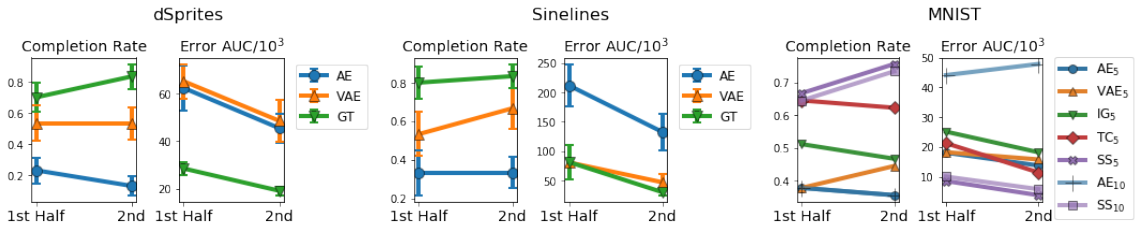


Figure 7.10: *Changes in average MTurk interactive reconstruction metrics between the first and last* $(N_q - 1)/2$ *questions (error-bars show standard error and are omitted on MNIST for readability). GT/SS models show the most consistent improvements over questions (suggesting conscious effort / learning) while AEs sometimes show degradation (suggesting users give up).*

model, users held similar feelings towards the SS vs. the AE. Although one could attribute these feelings to an illusion of explanatory depth [193], that would not explain why user feelings were consistent for the same pairs of models or why users assigned similar labels to dimensions of models they felt they understood (Table 7.1). It also would not explain evidence of learning across trials, e.g. why performance increased most consistently for interpretable models (Figure 7.10). Overall, the evidence suggests that users achieved meaningfully different levels of model understanding by performing the interactive reconstruction (and not the single-dimension) task.

Second, we found that when users felt they understood the meanings of dimensions, they could modify them in a single pass through the list, knowing in advance in which direction and how much to change them. This allowed them to solve problems efficiently, leading directly to high completion rates and low error AUC. When users did not understand dimensions, we found that they tended to adopt less efficient gradient ascent or random experimentation procedures with numerous loops through the full set of dimensions (or a partial set, in the intermediate-understanding case).

A competing hypothesis could be that, rather than gaining understanding, users *always* used gradient ascent or experimented randomly, but just happened to more easily stumble upon solutions with the models we assumed were "interpretable" than the models we assumed were "uninterpretable." In certain cases, this effect may be partially operative. For example, on MNIST, AEs consistently have the lowest reconstruction error (shown in

Table 7.2), implying they can reconstruct a wider variety of images and therefore have a larger "search space." Meanwhile, TCs consistently have the highest reconstruction error (25.4 vs. 15.5 at $D_z = 5$) and therefore the smallest search space, so randomly experimenting users ought to happen upon solutions more quickly.

However, this hypothesis is at odds with our qualitative observations, and also conflicts with our quantitative results in many cases. For example, on MNIST, the SS model has lower reconstruction error (20.8 at $D_z = 5$) and thus a larger search space than the TC model, but it performs much better. On Sinelines, the GT model has precisely zero reconstruction error but still performs best. Although having a smaller search space may be helpful (and arguably less expressive models may often be easier to understand), we posit that task performance depends more strongly on the understandability of a search space, rather than its size.

Generalizability. It is worth emphasizing that interactive reconstruction differentiated models in qualitatively similar ways across multiple datasets (dSprites, Sinelines, and MNIST) and user groups (workers on Amazon Mechanical Turk and students in Computer Science). This consistency suggests that interactive reconstruction can be useful as a research tool for comparing interpretable representation learning methods in varied contexts. Some innovation (beyond choosing appropriate parameters per Section 7.3.1) may still be required to adapt the method to non-visual or discontinuous data modalities (e.g. audio, text, or medical records) or to specific user groups in application-grounded contexts (per Doshi-Velez and Kim [60]). However, while it is desirable for interpretable representation learning algorithms to generalize to all contexts, interpretability *measurement* tools can afford to be a little more domain-specific, as long as they can still identify the algorithms that output the most interpretable models across contexts.

Limitations. Interactive reconstruction has limitations not shared by the single-dimension task which our experiments do not fully explore. First, these tasks require interactive visualization, which may be technically challenging for practitioners to implement, though the increasing usability of efficient web-based machine learning frameworks [217] helps.

Second, interacting with all dimensions simultaneously may be overwhelming for models with many tens or hundreds of representation dimensions. Potential workarounds include allowing users to annotate, as with MNIST, and/or group dimensions, or defining tasks over subgroups rather than the full set. However, we note that 100D representations may be inherently uninterpretable due to limits on working memory [223]—unless the models are structured so that only sparse subsets of dimensions need to be considered simultaneously (a strategy we consider in Chapter 8).

Our task also requires setting several parameter choices sensibly. For example, despite our initial pilots on MNIST, 8/10 think-aloud study participants made at least one comment that our distance metric, the intersection-over-union alignment percentage, did not match their intuitive notion of perceptual similarity, with P3 commenting that sometimes the images "*look similar but the alignment [percentage] doesn't reflect that*." Although it was uncommon for users to reach the threshold alignment ϵ without feeling that the images were similar, they felt frustrated that changes in $d(x, x')$ seemed unrelated to progress early on. Although we feel confident our method will generalize to many datasets and data types, our experience suggests practitioners will need to take care when selecting metrics $d(x, x')$ and thresholds ϵ. Future work could explore choosing example-specific thresholds or, on visual data, using metrics explicitly designed to model perceptual similarity [234, 242, 54] (if they can be evaluated efficiently in-browser).

Finally, interactive reconstruction is specific to generative models, e.g. autoencoders and GANs. Single-dimension tasks, however, can be made to support the other main category of representations, embeddings, via feature visualization [168]. Evaluating embedding interpretability is an area for future work. Cavallo et al. [40], Smilkov et al. [215], and Arendt et al. [14] may provide a foundation.

7.8 Conclusion

Developing reliable methods of evaluating interpretability is important for progress in interpretable ML. In this study, we introduced an interactive reconstruction task for evaluating

the interpretability of generative models, which have largely gone unstudied in the growing literature on human factors in ML. We validated our method by verifying it was effective at identifying ground-truth differences in model interpretability—both absolutely and relative to baselines—and that differences in objective performance metrics corresponded to meaningful differences in subjective understanding, which was measured in multiple independent ways. We then applied it to a wide range of representation learning methods from the disentanglement literature, and found evidence that methods which have been shown to improve disentanglement on synthetic data, e.g., [45], also improve interpretability on real data. To our awareness, ours is the first study providing such evidence.

Chapter 8

Benchmarks, Algorithms, and Metrics for Hierarchical Disentanglement[1]

In the previous chapter, we introduced a method to help (measure how well) humans understand generative model representations (e.g. autoencoders). We also provided evidence that particular models (e.g. the β total correlation autoencoder) learn more interpretable representations than baselines. This brings us closer to the vision outlined in Chapter 6; if we can learn representations which are sufficiently interpretable to human domain experts and which correspond sufficiently well to human domain experts' existing conceptual frameworks, then we can learn new discriminative models on top of those representations, potentially using concept gradients or more general forms of explanation regularization.

However, we still have a major unsolved problem: how do we learn a "sufficiently interpretable" representation that "corresponds sufficiently well" to the right concepts? Are existing disentangled representation learning methods actually sufficient? Perhaps in specific cases, but there are two general problems we will try to tackle in this chapter.

The first problem is that many disentanglement methods adopt the heuristic that the "right representation" ought to be *factorized*; i.e. that each representation dimension should

[1]This chapter is based on Andrew Slavin Ross and Finale Doshi-Velez. Benchmarks, algorithms, and metrics for hierarchical disentanglement. *arXiv preprint arXiv:2102.05185*, 2021.

be statistically independent. Many have already argued this is a problematic assumption, and even a problematic objective [145]; we will go into more details below.

The second, and potentially more pressing, problem (which we discussed in the conclusion of the previous chapter) is that for complicated enough instances x, we may need z to be high-dimensional. If z has too many dimensions, then the cognitive load required to understand it may be too great, even if the statistical independence objective makes it possible to understand the effect of each dimension somewhat independently.

In human conceptual schemas (i.e. in the formulation of Sweller [1994]), we tend to manage complexity by using hierarchical structure: schemas tend to contain a relatively small number of elements that need to be understood simultaneously, even if those elements themselves contain subschemas. What if representations could operate in a similar way? In this chapter, we will introduce mixed discrete-continuous representations whose dimensions are organized into trees such that dimensions in different branches *cannot be active simultaneously*. In principle, this allows for representations that, despite having a potentially high dimensionality D_z, only require users to consider relationships between $O(\log D_z)$ dimensions at any given time (within coherent scopes defined by the discrete variables). This kind of representation could allow for interpretability at realistic scale.

8.1 Introduction

Autoencoders aim to learn structure in data by compressing it to a lower-dimensional representation with minimal loss of information. Although this has proven useful in many applications [133], the individual dimensions of autoencoder representations are often inscrutable, even when the underlying data is generated by simple processes. Motivated by needs for interpretability [9, 155], fairness [51], and generalizability [26], as well as a basic intuition that representations should model the data correctly, a subfield has emerged which applies representation learning algorithms to synthetic datasets and checks how well representation dimensions "disentangle" the known ground-truth factors behind the dataset.

Perhaps the most common disentanglement approach has been to learn flat, continuous vector representations whose dimensions are statistically independent (and evaluate them using metrics that assume ground-truth factors are independent), reasoning that factorization is a useful proxy [184, 93, 45, 110]. However, this problem is not identifiable [145], and it seems unlikely that continuous, factorized, flat representations are the optimal choice for modeling many real-world generative processes, which are often highly structured.To address aspects of this problem, some approaches generalize to partially discrete representations [102], or encourage independence only conditionally, based on hierarchies or causal graphs [71, 227]. Almost all approaches require side-information, either about specific instances or about the global structure of the dataset.

Our approach in this paper is ambitious: we introduce (1) a flexible framework for modeling deep hierarchical structure in datasets, (2) novel algorithms for learning both structure and structured autoencoders entirely from data, which we apply to (3) novel benchmark datasets, and evaluate with (4) novel hierarchical disentanglement metrics. Our framework is based on the idea that data may lie on multiple manifolds with different intrinsic dimensionalities, and that certain (hierarchical groups of) dimensions may only become active for a subset of the data.[2] Though at first glance this approach seems it should worsen, not improve, identifiability, our assumption of geometric structure also serves as an inductive bias that empirically helps us learn representations that more faithfully (and explicitly) model ground-truth generative processes.

8.2 Related Work

Though interest in disentanglement is longstanding [199, 50, 25], a relatively recent resurgence has focused on **flat factorized representations**. Ridgeway [2016] provide an influential

[2]As a concrete example, consider the problem of learning representations of medical phenotypes of patients with and without diabetes mellitus, a complex disease with multiple types and subtypes [10]. Some underlying factors of phenotype variation—as well as the intrinsic complexity of these variations—are likely specific to the disease, its types, or its subtypes [7]. A representation that faithfully modeled the true factors of variation would need to be deeply hierarchical, with some dimensions only active for certain subtypes.

survey of such representations, arguing for their usefulness. Higgins et al. [2017] develop β-VAE, which tries to encourage factorization in variational autoencoders (VAEs, Kingma and Welling [2013]) by increasing the KL divergence penalty. Chen et al. [2018] and Kim and Mnih [2018] factorize by directly penalizing the total correlation (TC) between dimensions. Mixed discrete-continuous extensions are developed for KL by Dupont [2018] and TC by Jeong and Song [2019].

Although these innovations are specific to flat representations, there has been work on **certain forms of hierarchy**. Esmaeili et al. [2019] encourage different degrees of factorization within and across subgroups, which can be nested. However, they only apply their method in shallow contexts, and subgroups must be provided rather than learned from data. Choi et al. [2020] learn mixed discrete-continuous representations where some continuous dimensions are "public" in scope (and globally independent), while others are "private" to a categorical (and conditionally independent of siblings). However, they do not support deep hierarchies, and the structure of categorical, public, and private variables is provided rather than learned. GINs [218] can infer the dimensionality of such categorical-specific continuous dimensions groups, but still must be given the (shallow) categorical structure. FineGAN [212] learns a kind of structure, but enforces a specific shallow hierarchy of background, shape, and pose. Adams et al. [2010] model data with arbitrarily wide and deep trees using Bayesian non-parametric methods. However, there is no explicit encoder (representations are inferred via MCMC), and all features are binary. Our method attempts to provide the best of all these worlds; from data alone, we learn autoencoders whose representations have mixed discrete-continuous structure of arbitrary width and depth.

Our approach is also complementary to **recent shifts in the disentanglement literature**, especially from causality researchers. Parascandolo et al. [2018] and Träuble et al. [2020] argue for learning representations that disentangle causal mechanisms rather than statistically independent factors. Locatello et al. [2019] show that disentangling globally independent factors is non-identifiable, and suggest a shift in focus to inductive biases, weak supervision, and datasets with ground-truth interactions. The literature on learning representations with

side-information is rich [157, 210], and recent advances in disentanglement with extremely weak supervision are notable: Locatello et al. [2020] learn disentangled representations given instance-pairs that differ only by sparse sets of ground-truth factors, and Klindt et al. [2021] use similar principles to disentangle factors that vary sparsely over time. In this work, we return to the problem of learning disentangled representations from data alone. As our inductive bias to reduce (though not eliminate) non-identifiability, we assume the data contains discrete hierarchical structure that can be inferred geometrically. Though this introduces challenging new problems, it also creates opportunities to learn interpretable global summaries of the data.

Other related approaches not directly in this line of research include relational autoencoders [233], which model structure between non-iid flat data, and graph neural networks [56], which learn flat representations of structured data. In contrast, we model structure *within* flat inputs. Also relevant are advances in object representations, such as slot attention [147]. While this area has generally not focused on hierarchically nested objects, it does learn structure and seamlessly handles sets; we view our method as complementary. Finally, our hierarchy detection method is closely related to work in manifold learning. We build on work in multiple- and robust manifold learning [153, 154], contributing new innovations on top of them.

8.3 Hierarchical Disentanglement Framework

In this section, we outline our framework for modeling hierarchical structure in representations. In our framework, we associate individual data points with paths down a *dimension hierarchy* (examples in Fig. 8.1). Dimension hierarchies consist of dimension group nodes (shown as boxes), each of which can have any number of continuous dimensions (shown as ovals) and an optional categorical variable (diamonds) that leads to other groups based on its value. For any data point, we "activate" only the dimensions along its corresponding path. Notation-wise, $\texttt{root}(Z)$ denotes the group at the root of a hierarchy, and $\texttt{children}(Z_j)$ denotes the child groups of a categorical dimension Z_j. In the context of a dataset, for a

dimension Z_j or a dimension group g, $\texttt{on}(Z_j)$ or $\texttt{on}(g)$ denotes the subset of the dataset where that Z_j or g is active.

This framework can be readily extended to support multiple categorical variables per node (e.g. recursing on both segment halves in the timeseries dataset defined below) or even DAGs, such that instances can be associated with directed flows down multiple paths. For simplicity, however, we narrow our scope to tree structures in this work.

8.4 Hierarchical Disentanglement Benchmarks

For new frameworks, it is especially important to have synthetic benchmarks for which the true structure is known and ground truth disentanglement scores can be computed. Below we further describe the two benchmarks from Fig. 8.1.

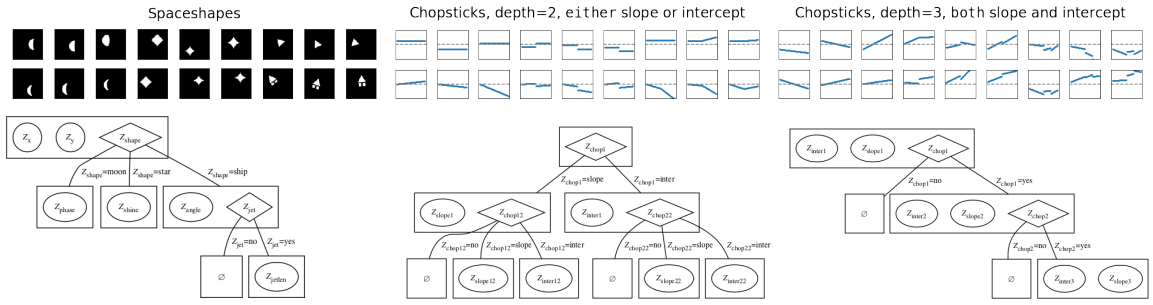


Figure 8.1: *Examples and ground-truth variable hierarchies for Spaceshapes and two different variants of Chopsticks. Continuous variables are shown as circles and discrete variables are shown as diamonds. Discrete variables have subhierarchies of additional variables that are only active for particular discrete values.*

8.4.1 Spaceshapes

Our first benchmark dataset is Spaceshapes, a binary 64x64 image dataset meant to hierarchically extend dSprites, a shape dataset common in the disentanglement literature [158]. Like dSprites, Spaceshapes images have location variables `x` and `y`, as well as a categorical `shape` with three options (in our case, `moon`, `star`, and `ship`). However, depending on shape, we add other continuous variables with very different effects: `moons` have a `phase`; `stars` have a sharpness to their `shine`; and `ships` have an `angle`. Finally, `ships` can optionally

have a `jet`, which has a length (`jetlen`), but this is only defined at the deepest level of the hierarchy. The presence of `jetlen` alters the intrinsic dimensionality of the representation; it can be either 3D or 4D depending on the path. As in dSprites, variables are sampled from continuous or discrete uniforms.

8.4.2 Chopsticks

Our second benchmark, Chopsticks, is actually a family of arbitrarily deep timeseries datasets. Chopsticks samples are 64D linear segments. Each segment can have a uniform-sampled `slope` and/or `intercept`; different Chopsticks variants can have one, the other, `both`, or `either` but not both. For all variants, segments initially span the whole interval. However, a coin is then flipped to determine whether to `chop` the segment, in which case we add a uniform offset to the slope and/or intercept of the right half. We repeat this process recursively up to a configurable maximum `depth`, biasing probabilities so that we have equal probability of stopping at each level. Each chop requires increasing local dimensionality to track additional slopes and intercepts. Although the underlying process is quite simple, the structure can be made arbitrarily deep, making it a useful benchmark for testing structure learning.

Although these datasets are designed to have clear hierarchical structure, in certain cases, there are multiple dimension hierarchies that could arguably describe the same dataset. See Fig. D.7 for more and §8.6.1 for how we handle them.

8.5 Hierarchical Disentanglement Algorithms

We next present a method for learning hierarchical disentangled representations from data alone. We split the problem into two brunch-themed algorithms, MIMOSA (which infers hierarchies) and COFHAE (which trains autoencoders).

8.5.1 Learning Hierarchies with MIMOSA

Algorithm 1 MIMOSA(X)

1: Encode the data X using a smooth autoencoder to reduce the initial dimensionality. Store as Z.
2: Construct a neighborhood graph on Z using a Ball Tree [170].
3: Run LocalSVD (Algorithm 3) on each point in Z and its neighbors to identify local manifold directions.
4: Run BuildComponent (Algorithm 5) to successively merge neighboring points with similar local manifold directions.
5: Run MergeComponents (Algorithm 6) to combine similar components over longer distances and discard outliers.
6: Run ConstructHierarchy (Algorithm 7) to create a dimension hierarchy based on which components enclose others.
7: **return** the hierarchy and component assignments.

The goal of our first algorithm, MIMOSA (**M**ulti-manifold **I**so**M**ap **O**n **S**mooth **A**utoencoder), is to learn a hierarchy $\hat{H}$ from data, as well as an assignment vector $\hat{A}_n$ of data points to hierarchy leaves. MIMOSA consists of the following steps (see Appendix for Algorithms 3-7 and complexity, and Fig 8.2 for an example):

Dimensionality Reduction (Algorithm 1, line 1): We start by performing an initial reduction of X to Z using a flat autoencoder. While we could start with $Z = X$, performing this reduction saves computation as later steps (e.g. finding neighbors) scale linearly with $|Z|$. Although this requires choosing $|Z|$, we find the exact value is not critical as long as it exceeds the (max) intrinsic dimensionality of the data. We also find it important to use differentiable activation functions (e.g. Softplus rather than ReLU) to keep latent manifolds smooth; see Fig. D.1 for more.

Manifold Decomposition (Algorithms 3-6): We decompose Z into a set of manifold "components" by computing SVDs locally around each point and merging neighboring points with sufficiently similar subspaces. We then perform a second merging step over longer lengthscales, combining equal-dimensional components with similar local SVDs along their nearest boundary points and discarding small outliers, which we found was necessary to handle interstitial gaps when two manifolds intersect. The core of this step is based

on a multi-manifold learning method [153], but we make efficiency as well as robustness improvements by combining ideas from RANSAC [73] and contagion dynamics [154]. The merging step is a new contribution.

It bears emphasis that manifold decomposition, which groups points based on the similarity of local principal components, is distinct from clustering, which groups points based on proximity. On our benchmarks, even hierarchical iterative clustering methods like OPTICS [12] will not suffice, as nearby points may lie on different manifolds.

Hierarchy Identification (Algorithm 7): We construct a tree by drawing edges from low-dimensional components to the higher-dimensional components that best "enclose" them, which we define using a ratio of inter-component to intra-component nearest neighbor distances; we believe this is novel. We use this tree and the component dimensionalities to construct a dimension hierarchy and a set of assignments from points to paths, which we output.

Hyperparameters: Each of these steps has several hyperparameters, and we provide a full listing and sensitivity study in §D.3. The one we found most critical was the minimum SVD similarity to merge neighboring points.

8.5.2 Training Autoencoders with COFHAE

Our first stage, MIMOSA, gives us the hierarchy and assignments of data down it. In the second stage, COFHAE (**CO**nditionally **F**actorized **H**ierarchical **A**uto**E**ncoder, Algorithms 2 and 8), we learn an autoencoder that respects this hierarchy via (differentiable) masking operations that impose structure on flat representations.

Hierarchical Encoding: Instances x pass through a neural network encoder to an initial vector z_{pre}, whose dimensions correspond to all continuous variables in the hierarchy as well as the one-hot encoded categorical variables. Categorical dimensions (denoted a') pass through a softmax with temperature τ to softly mask z_{pre} based on the hierarchy.

Algorithm 2 COFHAE(X)

hierarchy, assignments = MIMOSA(X) # Algorithm 1
HAE_θ = init_hierarchical_autoencoder(hierarchy)
D_ψ = init_discriminator()
for $x, a \sim$ minibatch(X, assignments) **do**
  $a', z = \text{HAE}_\theta$.encode($x; \tau$) # Algorithm 8
  $x' = \text{HAE}_\theta$.decode(concat(a', z)) # normal NN
  $z' = \text{copy}(z)$
  for $i = 1..|z_0|$ **do**
    shuffle $z'_{:,i}$ over minibatch indices where $\texttt{on}(z_{:,i})$
  end for
  $\mathcal{L}_\theta = \sum_n \mathcal{L}_x(x'_n, x_n) + \lambda_1 \mathcal{L}_a(a'_n, a_n) - \lambda_2 \log \frac{D_\psi(z_n)}{1 - D_\psi(z_n)}$
  $\mathcal{L}_\psi = \sum_n -\log D_\psi(z'_n) - \log(1 - D_\psi(z_n))$
  $\theta = \text{descent\_step}(\theta, \mathcal{L}_\theta)$
  $\psi = \text{descent\_step}(\psi, \mathcal{L}_\psi)$
end for
return HAE_θ

Supervising Assignments: Hierarchical encoding outputs estimated assignments a'. We add a penalty $\mathcal{L}_a(a', a)$, weighted by λ_1, to make these close to MIMOSA values a.

Conditional Factorization: Kim and Mnih [2018] penalize the total correlation (TC) between dimensions of flat continuous representations z with two tricks. First, noting that TC is the KL divergence between $q(z)$ (the joint distribution of the encoded z) and $\bar{q}(z) \equiv \prod_{j=1}^{|z|} q(z_j)$ (the product of its marginals), they approximate samples from $\bar{q}(z)$ by randomly permuting the values of each z_i across batches [13]. Second, they approximate the KL divergence between the two distributions using the density ratio trick [221] on an auxiliary discriminator $D_\psi(z)$, where $KL(q(z)||\bar{q}(z)) \approx \log \frac{D_\psi(z)}{1 - D_\psi(z)}$ if $D_\psi(z)$ outputs accurate probabilities of z having been sampled from $\bar{q}$. We adopt a similar approach, except instead of permuting each z_i across the full batch $\mathcal{B}$, we only permute it where it is *active*, i.e. $\mathcal{B} \cap \texttt{on}(z_i)$ (defined using the hardened version of the mask). This approximates a hierarchical version of $\bar{q}(z)$ where each dimension distribution is a mixture of 0 and the product of its *active* marginals. $D_\psi(z)$ then lets us estimate the KL between this distribution and $q(z)$, which we penalize and weight with λ_2.

This approach presumes ground-truth continuous variables should be conditionally

independent given categorical values, which is a major assumption. However, it is less strict than the assumption taken by many disentanglement methods, i.e. that continuous variables are independent marginally, and it may remain useful as an inductive bias.

8.6 Hierarchical Disentanglement Metrics

In this section, we develop metrics for quantifying how well learned representations and hierarchies match ground-truth.

8.6.1 Desiderata and Invariances

Our goal in designing metrics is to measure whether we have learned the "right representation," both in terms of global structure and specific variable correspondences. In an ideal world, we would measure whether a learned representation Z is identically equal to a ground-truth V. However, most existing disentanglement metrics are invariant to permutations, so that dimensions V_i can be reordered to match different Z_j, as well as univariate transformations, so that the values of Z_j do not need to be identical to V_i. In the case of methods like the SAP score [123], these univariate transformations must be linear, but as the uniformity of scaling can be arbitrary, we permit general nonlinear transformations, as long as they are 1:1, or invertible. Also, in the hierarchical case, there are certain ambiguities about the right vertical placement of continuous variables. For example, on Spaceshapes, the `phase`, `shine`, and `angle` variables could all be "merged up" to a single top-level variable whose effect changes based on `shape`. Alternatively, `x` and `y` position could be "pushed down" and duplicated for each shape despite their analogous effects (see Fig. D.7 for an illustration). Such "merge up" and "push down" transformations change the vector representation, but leave local dimensionality and the group structure of the hierarchy unchanged. We defer the problem of deciding the most natural vertical placement of continuous variables to future work, and make our main metrics invariant to them.

8.6.2 MIMOSA Metrics: H-error, Purity, Coverage

The first metric we use to evaluate MIMOSA is the **H-error**, which measures whether learned hierarchy $\hat{H}$ has the same essential structure as the ground-truth hierarchy H. To compute the H-error, we iterate over all possible paths p and p' down both H and $\hat{H}$, and attempt to pair them based on whether the minimum downstream dimensionality of p and p' matches at each respective node. The number of unpaired paths in either hierarchy is taken to be the H-error. This metric can only be 0 if both hierarchies have the same dimensionality structure, but is invariant to the "merge up" and "push down" operations described in §8.6.1.

The second MIMOSA metric is **purity**, which measures whether the assignments output by MIMOSA match ground-truth. To compute purity, we iterate through points assigned to each path $\hat{p}$ in $\hat{H}$, find the path p in H to which most of them belong, and then compute the fraction of points in $\hat{p}$ that belong to the majority p. Then we average these purity scores across $\hat{H}$, weighting by the number of points in $\hat{p}$. This metric only falls below 1 when we group together points with different ground-truth assignments.

The final metric we use to evaluate MIMOSA is **coverage**. Since MIMOSA discards small outlier components, it is possible that the final set of assignments will not cover the full training set. If almost all points are discarded this way, the other metrics may not be meaningful. As such, we measure coverage as the fraction of the training set which is not discarded. We note that hyperparameters can be tuned to ensure high coverage without knowing ground-truth assignments.

8.6.3 COFHAE Metrics: R^4 and R^4_c Scores

Per our desiderata, we seek to check whether every ground-truth variable V_i can be mapped invertibly to some learned dimension Z_j. As a preliminary definition, we say that a learned Z_j *corresponds* to a ground-truth V_i over some set $\mathcal{S} \subseteq \mathbb{R}$ if a bijection between them exists;

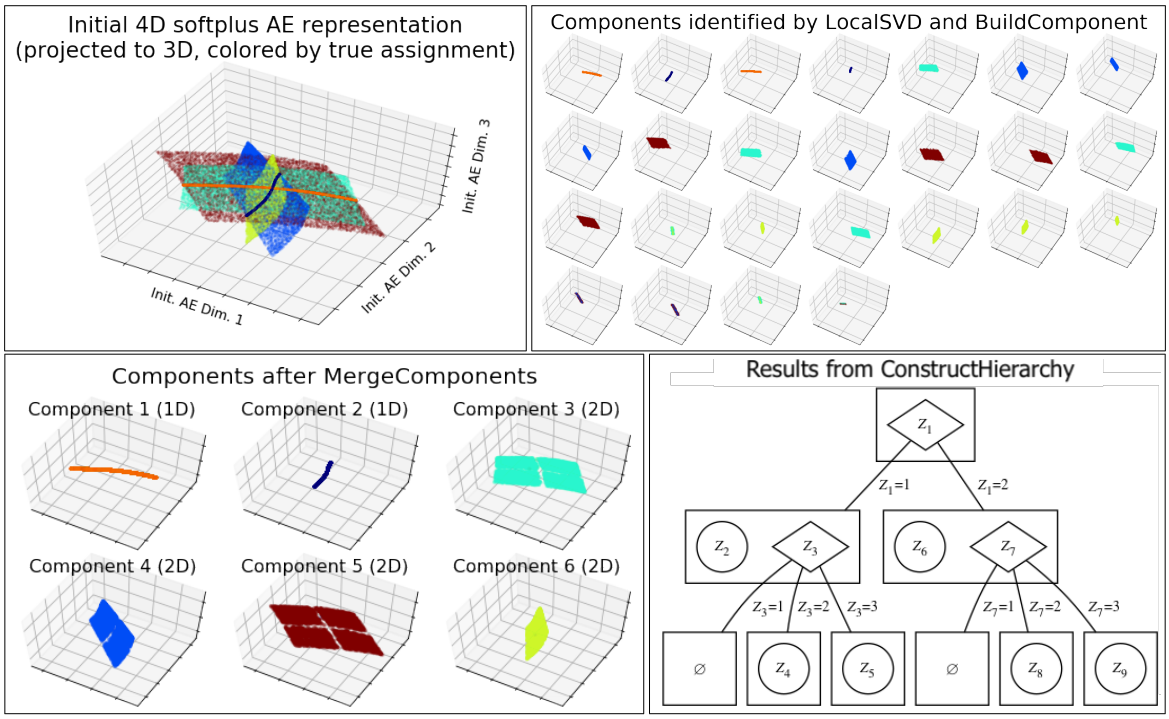


Figure 8.2: *Illustration of the stages of MIMOSA for the depth-2* `either` *version of Chopsticks (all colors based on ground-truth assignments). MIMOSA learns an initial 4D softplus AE representation (top left), decomposes it into lower-dimensional components of contiguous points with similar local SVD directions (top right), merges components with similar edges across longer distances and discards outliers (bottom left), and finally infers a dimension hierarchy (bottom right). In this case, correspondence to ground-truth is very close (99.8% component purity, covering 93.7% of the training set, with the correct hierarchical relationships). Examples for other datasets (without intermediate components) are shown in Figs. D.8–D.13 of the Appendix.*

that is,

$$\exists\, f(\cdot) : \mathcal{S} \to \mathbb{R} \text{ s.t. } f(V_i) = Z_j \text{ and } f^{-1}(Z_j) = V_i \tag{8.1}$$

We say that Z *disentangles* V if all V_i have a corresponding Z_j. To measure the extent to which bijections exist, we can simply try to learn them (over random splits of many paired samples of V_i and Z_j). Concretely, for each pair of learned and true dimensions, we train univariate models to map in both directions, compute their coefficients of determination

(R^2), and take their geometric mean:

$$\begin{aligned} f &\equiv \min_{f \in \mathcal{F}} \mathbb{E}_{\text{train}} \left[(f(X) - Y)^2 \right] \\ R^2(X \to Y) &\equiv \mathbb{E}_{\text{test}} \left[1 - \frac{\sum (f(X) - Y)^2}{\sum (\mathbb{E}[Y] - Y)^2} \right] \\ R^2(X \leftrightarrow Y) &\equiv \sqrt{\lfloor R^2(X \to Y) \rfloor_+ \lfloor R^2(Y \to X) \rfloor_+}, \end{aligned} \tag{8.2}$$

where we average over train/test splits (we use 5), assume $\mathcal{F}$ is sufficiently flexible to contain the optimal bijection (we use gradient-boosted decision trees), and assume our dataset is large enough to reliably identify $f \in \mathcal{F}$. In the limit, $R^2(X \leftrightarrow Y)$ can only be 1 if a bijection exists, as any region of non-zero mass in the joint distribution of X and Y where this is false would imply $\mathbb{E}[(f(X) - Y)^2] > 0$ or $\mathbb{E}[(f(Y) - X)^2] > 0$. In the special case that Y is discrete rather than continuous, we use classifiers for f and accuracy instead of R^2, but the same argument holds.

To measure whether a *set* of variables Z disentangles another *set* of variables V, we check if, for each V_i, there is at least one Z_j for which $R^2(V_i \leftrightarrow Z_j) = 1$:

$$R^4(V, Z) \equiv \frac{1}{|V|} \sum_i \max_j R^2(V_i \leftrightarrow Z_j), \tag{8.3}$$

We call this the "right-representation" R^2, or R^4 score. Note that this metric is related to the existing SAP score [123], except we allow for nonlinearity, require high R^2 in both directions, and take the maximum over each score column rather than the difference between the top two entries (which avoids assuming ground-truth is factorized).

Although R^4 is useful for measuring correspondence between sets of variables that are both always active, it does not immediately apply to hierarchical representations unless inactive variables are represented somehow, e.g. as 0 (an arbitrary implementation decision that affects R^2 by changing $\mathbb{E}[Y]$). It also lacks invariance to merge-up and push-down operations. Instead, we seek *conditional correspondence* between V_i and a set of dimensions in

Z, defined as

$$\begin{aligned}
&\text{for all } V_i \in \texttt{on}(V_i)\ \exists\ \mathcal{Z}_i = \{Z_j, Z_k, \ldots\} \text{ s.t.} \\
&\quad (a)\ V_i \text{ corresponds to } Z_j \text{ over } \texttt{on}(V_i) \cap \texttt{on}(Z_j), \\
&\quad (b)\ \texttt{on}(Z_j) \cap \texttt{on}(Z_k) = \emptyset \text{ for all } j \neq k, \text{and} \\
&\quad (c)\ \textstyle\bigcup_{z \in \mathcal{Z}_i} \texttt{on}(z) = \texttt{on}(V_i),
\end{aligned} \tag{8.4}$$

or rather that we can find some tiling of $\texttt{on}(V_i)$ into regions where it corresponds 1:1 with different Z_j which are never active simultaneously. This allows for one Z_j to correspond to non-overlapping elements of V (e.g. merging up), as well as for one V_i to be modeled by non-overlapping elements of Z (e.g. pushing down).

We can then formulate a conditional R^4_c score which quantifies how closely conditional correspondence holds:

$$R^2_c(V_i, g) \equiv \max\left(\max_{j \in g}\left(R^2\big(V_i \leftrightarrow Z_j \big| \texttt{on}(V_i) \cap \texttt{on}(g)\big), \sum_{g' \in \texttt{children}(Z_j)} R^2_c(V_i, g') \frac{|\texttt{on}(V_i) \cap \texttt{on}(g')|}{|\texttt{on}(V_i)|}\right)\right),$$

for a dimension group g; the overall disentanglement is:

$$R^4_c(V \leftrightarrow Z) \equiv \frac{1}{|V|} \sum_{i=1}^{|V|} R^2_c(V_i, \texttt{root}(Z)). \tag{8.5}$$

In the special case that V and Z are flat, R^4_c reduces to R^4. We note that even for flat representations, the R^4 score may be a useful measure of disentanglement when ground-truth variables are not factorized.

8.7 Experimental Setup

Benchmarks: We ran experiments on nine benchmark datasets: Spaceshapes, and eight variants of Chopsticks (varying `slope`, `intercept`, `both`, and `either` at recursion depths of 2 and 3). See §8.4 for more details, and Fig. D.14 for preliminary experiments on noisy data.

Algorithms: In addition to COFHAE with MIMOSA, we trained the following baselines: autoencoders (AE), variational autoencoders [113] (VAE), the β-total correlation autoencoder [45] (TCVAE), and FactorVAE [110]. We also ran COFHAE ablations using the ground-truth hierarchy and assignments, testing all possible combinations of loss terms and comparing conditional vs. marginal TC penalties; results are in Fig. 8.4. See §D.1 for training and model details.

Metrics: To evaluate hierarchies, we computed purity, coverage, and H-error (§8.6.2). Results are in Table 8.1. To measure disentanglement, we primarily use R^4_c (§8.6.3); results for all datasets and models are in Fig. 8.3. We also compute the following baseline metrics: the SAP score [123] (SAP), the mutual information gap [45] (MIG, estimated using 2D histograms), the FactorVAE score [110] (FVAE), and the DCI disentanglement score [66] (DCI). Most implementations were adapted from `disentanglement_lib` [145]. We also compute our marginal R^4 score. Results across metrics are shown for a subset of datasets and models in Fig. 8.5.

Hyperparameters: COFHAE is only given instances X, which complicates cross-validation. However, we can still tune parameters to ensure assignments a' match MIMOSA outputs a and reconstruction loss for x is low (which can fail to happen if the adversarial term dominates). Over a grid of τ in $\{\frac{1}{2}, \frac{2}{3}, 1\}$, λ_1 in $\{10, 100, 1000\}$, and λ_2 in $\{1, 10, 100\}$, we select the model with the lowest *training* reconstruction loss $\mathcal{L}_x$ from the $\frac{1}{3}$ with the lowest assignment loss $\mathcal{L}_a$. For MIMOSA, hyperparameters can be tuned to ensure high coverage (purity and H-error require side-information); see §D.3 for more. For β-TCVAE and FactorVAE, we show results for β=5 and γ=10, but test both across $\{5, 10, 25, 50\}$ in Fig. D.5.

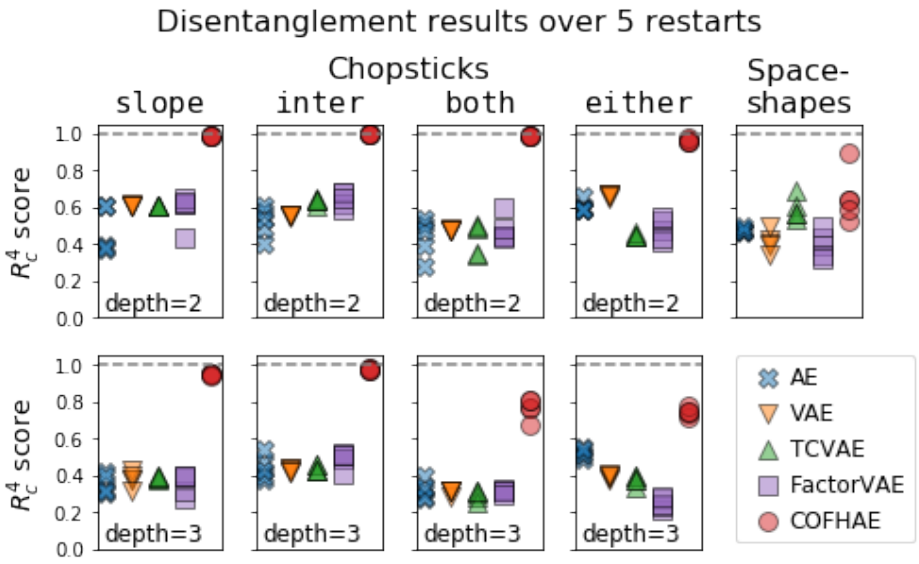


Figure 8.3: *Hierarchical disentanglement results for representation learning methods (baselines and COFHAE + MIMOSA) over all nine datasets. COFHAE almost perfectly disentangles ground-truth on the six simplest versions of Chopsticks, with some degradations on the two most complex versions (with very deep hierarches) and on Spaceshapes (with a shallower hierarchy, but higher-dimensional inputs). Baseline methods were generally much more entangled, though β-TCVAE is competitive on Spaceshapes.*

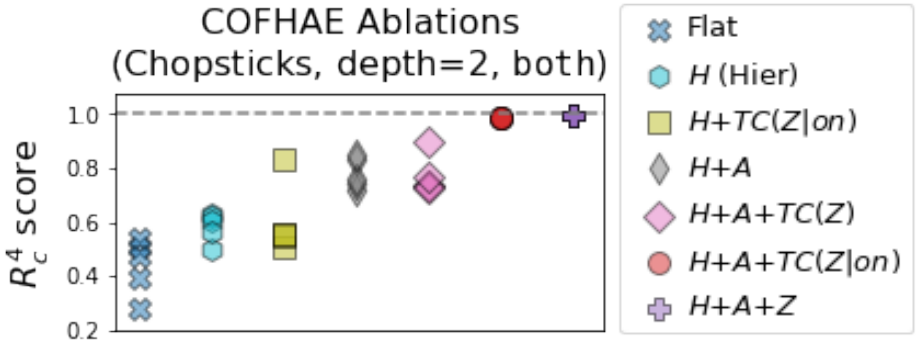


Figure 8.4: *Ablation study for COFHAE on the depth-2* `both` *version of Chopsticks (over 5 restarts). Hierarchical disentanglement is low for flat AEs (Flat); adding the ground-truth hierarchy H improves it (Hier H), as does also adding supervision for ground-truth assignments A (H+A). Adding a FactorVAE-style marginal TC penalty (H+A+$TC(Z)$) does not appear to help disentanglement, but making that TC penalty conditional (H+A+$TC(Z|on)$, i.e. COFHAE) brings it close to the near-optimal disentanglement of a hierarchical model whose latent representation is fully supervised (H+A+Z). However, the hierarchical conditional TC penalty fails to produce this same disentanglement without any supervision over assignments (H+$TC(Z)$).*

Table 8.1: *MIMOSA results.*

MIMOSA Metric	Chopsticks, depth=2				Chopsticks, depth=3				Space-shapes
	`inter`	`slope`	`both`	`either`	`inter`	`slope`	`both`	`either`	
Purity	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	.98±0.0	.95±0.0	.94±0.0	.93±0.0	1.0±0.0
Coverage	.99±0.0	.99±0.0	.96±0.0	.93±0.0	.98±0.0	.98±0.0	.82±0.01	.75±0.01	1.0±0.0
H-error	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	2.6±1.34	0.0±0.0

MIMOSA results across all datasets, with means and standard deviations across 5 restarts. In general, MIMOSA components contained points only from single ground-truth sets of paths (purity), were inclusive of most points in the training set (coverage), and resulting in perfectly accurate hierarchies (H errors), with the greatest or only exception being the Chopsticks depth-3 `either` dataset (where we tended to recover only 12 of its 14 possible paths).

8.8 Results and Discussion

MIMOSA consistently recovered the right hierarchies. Per Table 8.1, we consistently found the right hierarchy for all datasets except depth-3 `either`-Chopsticks, but even there results were close, generally recovering 12 of 14 possible hierarchy paths (see Fig. D.13 for more details). Purity and coverage were also high, often near perfect as in Spaceshapes or depth-2 Chopsticks.

COFHAE significantly outperformed baselines. Per Fig. 8.3, COFHAE R_c^4 scores were near-perfect for 6 out of 9 datasets, and better than baselines on all. On Spaceshapes and the depth-3 `either` and `both` versions of Chopsticks, scores were slightly worse. Part of this suboptimality could be due to non-identifiability. For Spaceshapes and the `both` versions of Chopsticks, dimension group nodes contain multiple continuous variables, which even conditionally can be modeled by multiple factorized distributions [145]. However, optimization issues could also be at fault, as we do not see suboptimal R_c^4 on Chopsticks until a depth of 3, and even supervised $H+A+Z$ models occasionally fail to converge on Spaceshapes. Kim and Mnih [2018] note that the relatively low-dimensional discriminator used by FactorVAE is easier to optimize than the generally high-dimensional discriminators used in GANs, which are notoriously tricky to train [160]. In our case, flattened hierarchy vectors can be high-dimensional (e.g. Fig. D.15), and in any given batch, instances corresponding to differ-

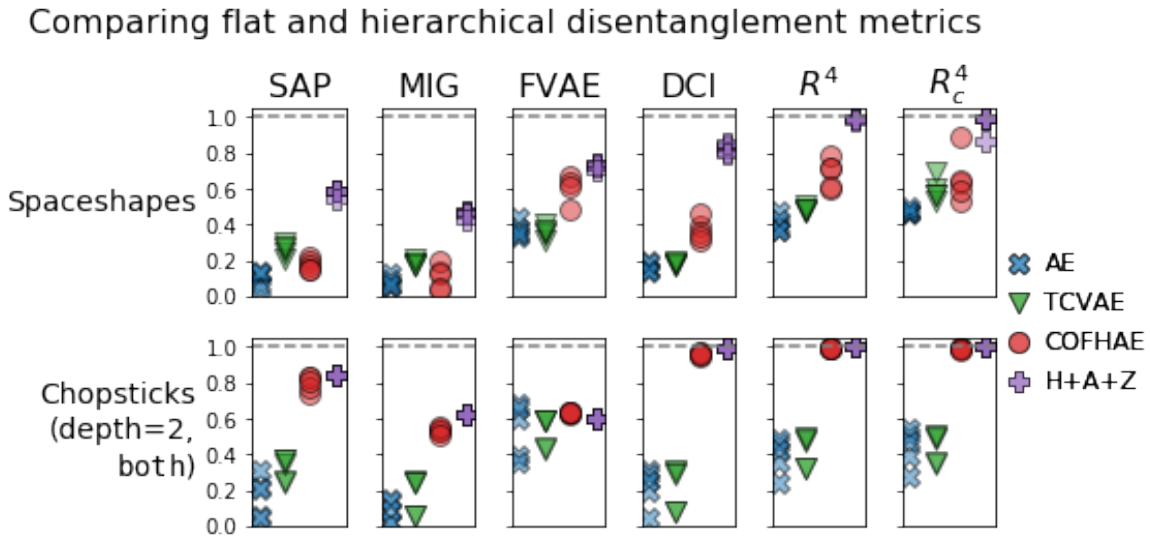


Figure 8.5: *Comparison of disentanglement metrics across two datasets and four models. Only R^4 and R_c^4 correctly and consistently award near-optimal scores to the supervised H+A+Z model.*

ent paths down the hierarchy may have different numbers of samples (potentially requiring larger batch sizes or stratified sampling to ensure sufficient coverage). Finally, alongside non-identifiability and optimization issues, MIMOSA errors (e.g. merge-up/push-down differences for Spaceshapes and suboptimal purity and coverage for Chopsticks) also may play a role, as evidenced by performance improvements in our full COFHAE ablations in Fig. D.4. Despite all of these issues, COFHAE is still closer to optimal than any of our baseline algorithms.

R_c^4 provides more insight into disentanglement than baselines. One way to evaluate an evaluation metric is to test it against a precisely known quantity. In this case, we know the $H+A+Z$ model, whose encoder is supervised to match ground-truth, should receive a near-perfect score. The only metrics to do this consistently are R^4 and R_c^4. Note that the DCI disentanglement score, based on the entropy of normalized feature importances from an estimator predicting single ground-truth factors from all learned dimensions, comes close. Intuitively, this metric could behave similarly to R^4 if its estimator was trained to be sparse (placing importance on as few dimensions as possible). However, using R^2s of univariate

estimators is more direct, and also incorporates information from the DCI informativeness score.

Another way to evaluate an evaluation metric is to test whether quantitative differences capture salient qualitative differences. To this point, specifically to compare R^4 and R^4_c, we consider several examples in Fig. D.6 and Fig. 8.6. First, we see that for the Spaceshapes COFHAE model in Fig. D.6c, its R^4_c score (0.89) is higher than its R^4 (0.79). This increase is due to the fact that R^4 penalizes the "push-down" differences (§8.6.1) between the learned and true factors representing `x` and `y` position, while R^4_c is invariant to them. However, the overall increase is less dramatic than one might expect due to modest decreases in correspondence scores for other dimensions (e.g. 0.98→0.89 for `jetlen`), which occur because R^4_c is not biased by spurious equality between dimensions which are both inactive. Another example of a difference between R^4 and R^4_c (illustrating invariance to "merging up" rather than "pushing down") is for the Spaceshapes β-TCVAE in Fig. D.6b. In this case, histograms show that one β-TCVAE variable (Z_3) corresponds closely to both moon `phase` and star `shine` (and to a lesser extent, `jetlen`), only one of which is active at a time. The R^4 score (0.47) assigns low scores to these correspondences, but R^4_c (0.69) properly factors them in.

COFHAE and MIMOSA subcomponents improve performance. Though COFHAE contains many moving parts, results in Fig. 8.4 and Fig. D.4 suggest they all count. Autoencoders only achieve optimal disentanglement if provided with the hierarchy, assignments, and a conditional (not marginal) penalty on the TC of continuous variables; no partial subset suffices. In the Appendix, Fig. D.3 shows ablations and sensitivity analyses for MIMOSA that validate its subcomponents are important as well.

8.8.1 Visualizing Hierarchical Representations

Although not the focus of this work, we can visualize (or evaluate the interpretability of) hierarchical representations by straightforward extensions of the methods from Chapter 7. In

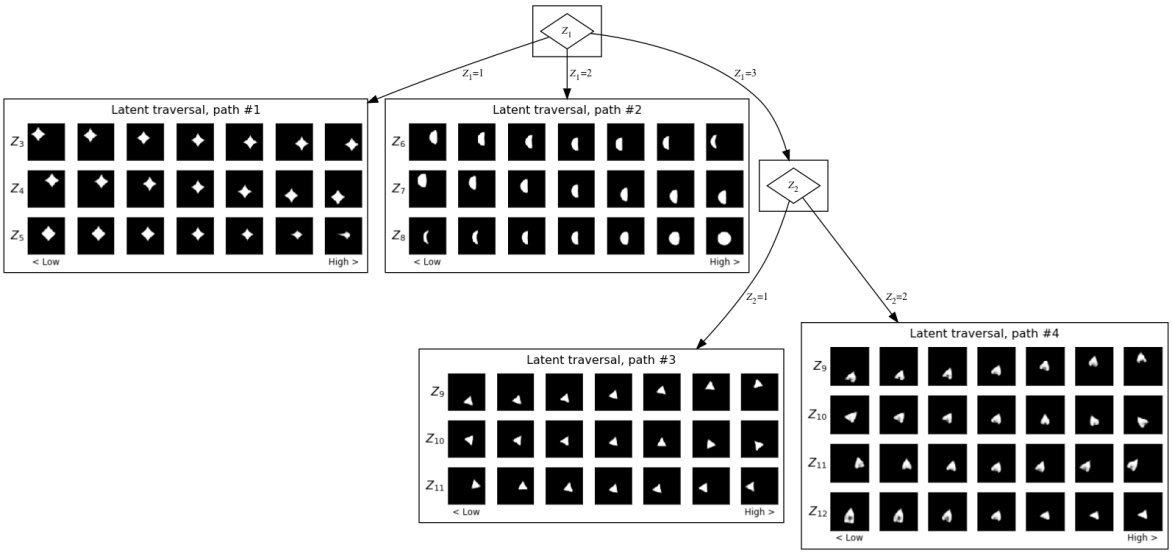


Figure 8.6: *Hierarchical latent traversal plot of the best-performing Spaceshapes COFHAE model ($R_c^4 = 0.89$). Individual latent traversals show the effects of linearly sweeping each* active *dimension from its 1st to 99th percentile value (center column shows the same input with intermediate values for all active dimensions). Consistent with Fig. D.6c, the model is not perfectly disentangled, though primary correspondences are clear: star* `shine` *is modeled by Z_5, moon* `phase` *is modeled by Z_8, ship* `angle` *is modeled by Z_{10}, ship* `jetlen` *is modeled by Z_{12}, and $(\mathtt{x}, \mathtt{y})$ are modeled by (Z_3, Z_4), (Z_6, Z_7), and (Z_{11}, Z_9) respectively for each shape.*

Figure 8.6, we illustrate how to visualize hierarchical representations with latent traversals, while in Figure 8.7, we show screenshots of interactive reconstruction. A live demo of hierarchical interactive reconstruction can be accessed here.

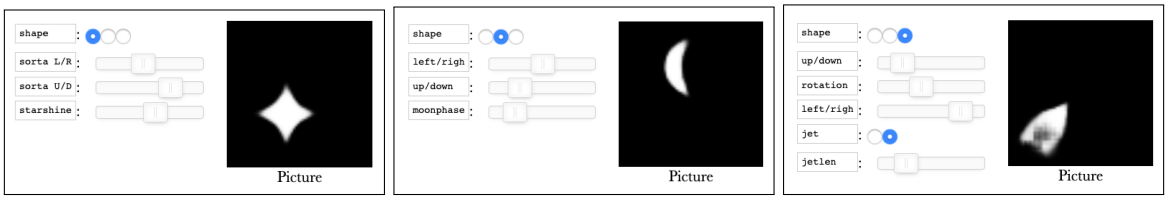


Figure 8.7: *Hierarchical interactive reconstruction example on Spaceshapes (close-up on sliders and output image). Based on selected radio buttons, different groups of inputs appear, which can include both sliders and more radio buttons.*

8.9 Conclusion

In this work, we introduced the problem of hierarchical disentanglement, where ground-truth representation dimensions are organized into a tree and activated or deactivated based on the values of categorical dimensions. We presented benchmarks, algorithms, and metrics for learning and evaluating such hierarchical representations.

There are a number of promising avenues for future work. One is extending the method to handle a wider variety of underlying structures, e.g. non-hierarchical dimension DAGs, or integrating our method with object representation techniques to better model generative processes involving ordinal variables or unordered sets [147]. Another is to better solve or understand hierarchical disentanglement as we have already formulated it, e.g. by improving robustness to noise, or providing a better theoretical understanding of identifiability and when we can guarantee methods will succeed. Finally, there are ample opportunities to apply these techniques to real-world cases that we expect to have hierarchical structure, such as causal inference, patient phenotype, or population genetics datasets.

More generally, we feel it is important for representation learning to move beyond flat vectors, and work towards explicitly modeling the rich structure contained in the real world. For a long time, many symbolic AI and cognitive science researchers have argued that AI progress should be evaluated not by improvements in accuracy or reconstruction error, but by how well we can build models that build their own interpretable models of the world [130]. Our work takes steps in this direction.

Part IV

Conclusion

Chapter 9

Conclusion

9.1 Summary of Contributions

In this dissertation, we argued that many prediction problems are ambiguous, and that machine learning models trained to solve them tend to be right (on the training distribution) but for the wrong reasons (implying fragility to test-time shifts in $p(x)$, even if $p(y|x)$ is unchanged). We then argued that we might significantly improve robustness (and learn models which are right for better reasons) by incorporating human domain knowledge into model objective functions. In Part II, we used input gradient penalties to accomplish this task in a variety of contexts, demonstrating we could bias models towards learning particular implicit decision rules (Chapter 3), improve the diversity (and interpretability) of ensembles (Chapter 4), and improve the adversarial robustness (and interpretability) of convolutional neural networks (Chapter 5).

Although more recent approaches based on aggregating gradients over paths [70] or optimizing inner gradient-based objectives over regions [69] perform better, the success of these approaches still supports our primary point: we can learn models which are more interpretable, robust, and aligned with expert knowledge by optimizing not just model predictions, but how predictions change with meaningful changes to our inputs.

However, we also argued that certain kinds of expert knowledge are difficult to translate

into "meaningful changes to our inputs," and instead are best expressed in terms of abstract (and hierarchically organized) concepts [107]. To handle these cases, we argued that it might be necessary to learn more human-compatible representations of our data. Therefore, in Part III, we made two contributions towards learning such representations: we developed a method for measuring representation interpretability (Chapter 7), as well as methods and metrics for learning and evaluating hierarchical representations (Chapter 8), which we believe could be significantly more interpretable and true to the data than baselines.

Methodologically, we took pains in both parts to ensure that whenever we evaluated explanations or interpretability, we also had some way of comparing our evaluations to ground-truth. For example, in Chapter 3, we knew which features in our synthetic dataset actually mattered for prediction, and we knew that models fooled by decoy datasets were sensitive to the corresponding confounds. In much of Chapter 7, we knew which models were most and least interpretable beforehand, because they were constructed with the express purpose of justifying such assumptions—and even then we still cross-checked our conclusions with many different metrics from user studies. Interpretability research has many methodological and epistemological pitfalls [194, 4, 6, 85, 34], but we hope our evaluation strategies helped us avoid at least some of them.

9.2 Future Directions

To close, we will sketch out several directions for future work.

Combining the chapters: In the immediate term, there is more we could do to combine the contributions of our chapters. By adding dropdowns and checkboxes to the interactive reconstruction interface we presented in Chapter 7, we could easily apply the same interpretability measurement technique to the kinds of representations we learned in Chapter 8 (see an example here), allowing us to test our hypothesis that they might be more interpretable. We could also then attempt to use the representations we learn and explain with the methods from Part III to help domain experts encode knowledge in situations analogous

to those of Part II, in a grand study that utilizes every chapter of this dissertation.

However, creating an interface to integrate all these disparate ML methods seems challenging from an HCI perspective, and there are still limits on the kinds of knowledge we can encode with them, even with perfect hierarchical disentangled representations (which might themselves need knowledge to identify [145]). Many long-term challenges remain.

Interfaces for human-AI intersubjectivity: In this general spirit, though, Olah et al. [2018] envision a design space of "interpretability interfaces" with a formal grammar for expressing, exploring, and exploiting explanations. They visualize this design space in a grid of relationships between different "substrates" of the design, which include groups of neurons, dataset examples, and model parameters—the latter of which presents an opportunity "to consider interfaces for *taking action* in neural networks." If human-defined concepts, disentangled representations, or other forms of explanation are included as additional substrates, one can start to imagine a very general framework for expressing priors or constraints on relationships between them.

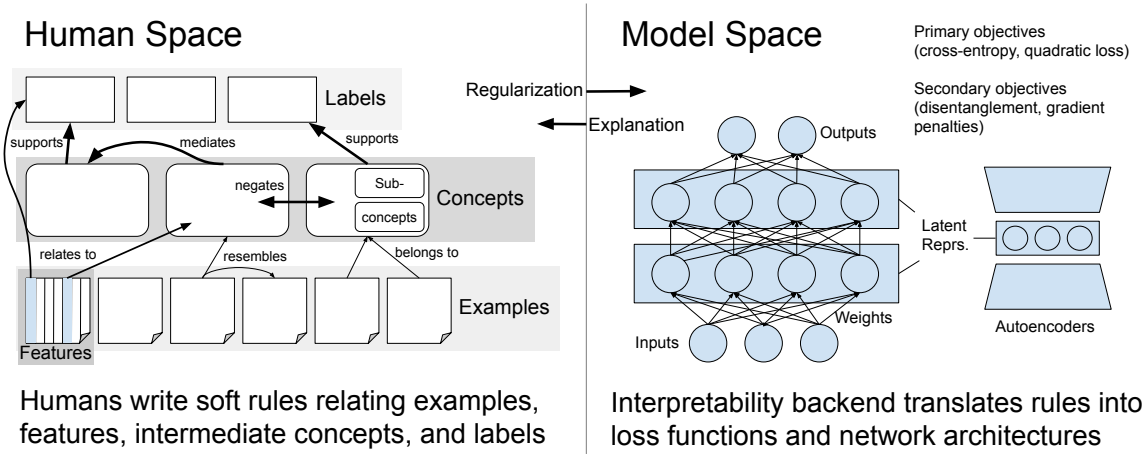


Figure 9.1: *Schematic diagram of an interpretability interface.*

How would humans actually express these kinds of objectives? One interface worth emulating could be that introduced by popular libraries for weak supervision [181] or probabilistic soft logic [18]. In these frameworks, users can specify "soft" logical rules for

labeling datasets or constraining relationships between atoms and substrates of a system, whose weights are inferred by the framework. While these inference problems are nontrivial, and in general there may be complex interactions between rules that are difficult to capture, the interface these libraries expose to users is expressive and potentially worth emulating in an interpretability interface. For example, we could imagine writing soft rules relating:

- dataset examples to each other (e.g. these examples should be conceptually similar with respect to a task)
- dataset examples to concepts (e.g. these are examples of a concept)
- features to concepts (e.g. this set of features is related to this concept, this other set is not; in this specific case, these features contribute positively)
- concepts to predictions (e.g. the presence of this concept makes this prediction more or less likely, except when this other concept is present)
- concepts to concepts (e.g. this concept is hierarchically nested within another, or only conditionally active)

These rules could be "compiled" into architectural constraints or additional energy terms in component model loss functions. We present a schematic diagram of how a system like this might work in Figure 9.1.

Many ML and HCI challenges would need to be solved in order to build such a system, especially if it should be useful. The bulk of the work might lie in attempting to align human concepts with representations. If 1:1 mappings between human concepts and latent representations do not emerge naturally from disentanglement methods, even allowing for hierarchical relationships, steps could be taken to optimize model representations to better match human understanding (e.g. with partial supervision) or to help humans better understand model representations (e.g. with interactive or feature visualization). This process of reaching user-model intersubjectivity might require multiple stages of identification and refinement, but seems possible in principle. And perhaps arriving at a

shared conceptual framework for understanding a problem is where the work of teaching and learning *ought* to lie, regardless of whether the teachers and learners are human.

Static concepts vs. dynamic narratives: We began by stating that "the motivation for this dissertation is easiest to express with a story." That is no accident. We, as human beings, rely heavily on stories or narratives in our explanations for external phenomena—as well as for our own lives and experiences [107, 161]. Stories (with an arrow of time) are also essential to causality [177]. For machine learning models to make predictions for the right reasons, their *reasoning* may need to incorporate dynamic temporal structure alongside static hierarchical structure.

9.3 Final Thoughts

Building on Keil [2006], Doshi-Velez and Kim [2017] argue that interpretability is necessary when our systems have a certain *incompleteness*. In the context of this dissertation, we have considered the kind of incompleteness that results from ambiguity in a dataset. When datasets do not fully specify their decision boundaries, we have freedom to learn many models that could accurately classify them. The vast majority will be right in training but for the wrong reasons, implying vulnerability to distribution shifts. Unless we can be certain such shifts will not occur in practice, we must be extremely careful, and take steps to identify and mitigate as many problems as possible (if we still must use ML at all).

The promise of machine learning to improve human lives is great, but so is its peril. Machine learning models are being used in more and more critical situations, such as bail determination [11] and child maltreatment risk screening [48], without much oversight or awareness of their fragility. Even if the models were "correct," many of these applications would remain horribly wrong [201, 67, 163]. In this thesis, we presented methods and ideas to help practitioners encode domain knowledge into machine learning objective functions. We believe good could result from learning models that are right for the right reasons, but only if they are *used* for the right reasons.

References

[1] Ashraf Abdul, Christian von der Weth, Mohan Kankanhalli, and Brian Y Lim. Cogam: Measuring and moderating cognitive load in machine learning model explanations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.

[2] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

[3] Ryan Adams, Hanna Wallach, and Zoubin Ghahramani. Learning the structure of deep sparse graphical models. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[4] Julius Adebayo, Justin Gilmer, Ian Goodfellow, and Been Kim. Local explanation methods for deep neural networks lack sensitivity to parameter values. 2018.

[5] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in Neural Information Processing Systems*, 31:9505–9515, 2018.

[6] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. Debugging tests for model explanations. *arXiv preprint arXiv:2011.05429*, 2020.

[7] Emma Ahlqvist, Petter Storm, Annemari Käräjämäki, Mats Martinell, Mozhgan Dorkhan, Annelie Carlsson, Petter Vikman, Rashmi B Prasad, Dina Mansour Aly, Peter Almgren, et al. Novel subgroups of adult-onset diabetes and their association with outcomes: a data-driven cluster analysis of six variables. *The Lancet Diabetes & Endocrinology*, 6(5):361–369, 2018.

[8] Hiva Allahyari and Niklas Lavesson. User-oriented assessment of classification model understandability. In *11th scandinavian conference on Artificial intelligence*. IOS Press, 2011.

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

[10] American Diabetes Association. Diagnosis and classification of diabetes mellitus. *Diabetes Care*, 2005.

[11] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. How we analyzed the compas recidivism algorithm. *ProPublica*, 2016.

[12] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod Record*, 28(2):49–60, 1999.

[13] Miguel A Arcones and Evarist Gine. On the bootstrap of u and v statistics. *The Annals of Statistics*, pages 655–674, 1992.

[14] Dustin L Arendt, Nasheen Nur, Zhuanyi Huang, Gabriel Fair, and Wenwen Dou. Parallel embeddings: a visualization technique for contrasting learned representations. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 259–274, 2020.

[15] Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

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

[17] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[18] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *Journal of Machine Learning Research*, 18(109): 1–67, 2017.

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

[20] Gagan Bansal, Besmira Nushi, Ece Kamar, Walter S Lasecki, Daniel S Weld, and Eric Horvitz. Beyond accuracy: The role of mental models in human-ai team performance. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, pages 2–11, 2019.

[21] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[22] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 3319–3327. IEEE, 2017.

[23] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Visualizing and understanding gans. 2019.

[24] Edward W Beals. Bray-curtis ordination: an effective strategy for analysis of multivariate ecological data. In *Advances in ecological research*, volume 14, pages 1–55. Elsevier, 1984.

[25] Yoshua Bengio. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37. Springer, 2013.

[26] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[27] Michael Betancourt. A geometric theory of higher-order automatic differentiation. *arXiv preprint arXiv:1812.11592*, 2018.

[28] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *Journal of Machine Learning Research*, 11, 2010. URL http://portal.acm.org/citation.cfm?id=1859903.

[29] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[30] Leo Breiman. Bagging predictors. *Machine learning*, 24(2), 1996.

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

[32] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 2005.

[33] Dariusz Brzezinski and Jerzy Stefanowski. Ensemble diversity in evolving data streams. In *International Conference on Discovery Science*, 2016.

[34] Zana Buçinca, Phoebe Lin, Krzysztof Z Gajos, and Elena L Glassman. Proxy tasks and subjective measures can be misleading in evaluating explainable ai systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 454–464, 2020.

[35] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[36] Yaroslav Butalov. The notMNIST dataset. http://yaroslavvb.com/upload/notMNIST/, 2011.

[37] Carrie J Cai, Emily Reif, Narayan Hegde, Jason Hipp, Been Kim, Daniel Smilkov, Martin Wattenberg, Fernanda Viegas, Greg S Corrado, Martin C Stumpe, et al. Human-centered tools for coping with imperfect algorithms during medical decision-making. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2019.

[38] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

[39] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

[40] Marco Cavallo and Çağatay Demiralp. A visual interaction framework for dimensionality reduction based data exploration. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[41] Paul Chandler and John Sweller. The split-attention effect as a factor in the design of instruction. *British Journal of Educational Psychology*, 62(2):233–246, 1992.

[42] Arjun Chandrasekaran, Deshraj Yadav, Prithvijit Chattopadhyay, Viraj Prabhu, and Devi Parikh. It takes two to tango: Towards theory of ai's mind. *arXiv preprint arXiv:1704.00717*, 2017.

[43] Chaofan Chen, Oscar Li, Alina Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574*, 2018.

[44] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[45] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018.

[46] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[47] Jaewoong Choi, Geonho Hwang, and Myungjoo Kang. Discond-vae: disentangling continuous factors from the discrete. *arXiv:2009.08039*, 2020.

[48] Alexandra Chouldechova, Diana Benavides-Prado, Oleksandr Fialko, and Rhema Vaithianathan. A case study of algorithm-assisted decision making in child maltreatment hotline screening decisions. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 134–148, New York, NY, USA, 23–24 Feb 2018. PMLR. URL http://proceedings.mlr.press/v81/chouldechova18a.html.

[49] Eric Chu, Deb Roy, and Jacob Andreas. Are visual explanations useful? a case study in model-in-the-loop prediction. *arXiv preprint arXiv:2007.12248*, 2020.

[50] Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 1994.

[51] Elliot Creager, David Madras, Jörn-Henrik Jacobsen, Marissa Weis, Kevin Swersky, Toniann Pitassi, and Richard Zemel. Flexibly fair representation learning by disentanglement. In *International Conference on Machine Learning*, 2019.

[52] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[53] Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, pages 4281–4290, 2017.

[54] Steffen Czolbe, Oswin Krause, Ingemar Cox, and Christian Igel. A loss function for generative neural networks based on watson's perceptual model. *Advances in Neural Information Processing Systems*, 33, 2020.

[55] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

[56] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 2016.

[57] Guillaume Desjardins, Aaron Courville, and Yoshua Bengio. Disentangling factors of variation via generative entangling. *arXiv preprint arXiv:1210.5474*, 2012.

[58] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, 2000.

[59] Jeff Donahue and Kristen Grauman. Annotator rationales for visual recognition. In *2011 International Conference on Computer Vision*, pages 1395–1402. IEEE, 2011.

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

[61] Hubert L Dreyfus and Stuart E Dreyfus. What artificial experts can and cannot do. *AI & society*, 6(1):18–26, 1992.

[62] Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

[63] Jianzhun Du, Andrew Slavin Ross, Yonadav Shavit, and Finale Doshi-Velez. Controlled direct effect priors for bayesian neural networks. In *NeurIPS 2019 Workshop on Bayesian Deep Learning*, 2019.

[64] Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. Learning credible dnns via incorporating prior knowledge and model local explanation. *Knowledge and Information Systems*, 63(2):305–332, 2021.

[65] Emilien Dupont. Learning disentangled joint continuous and discrete representations. In *Advances in Neural Information Processing Systems*, 2018.

[66] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

[67] Laurel Eckhouse, Kristian Lum, Cynthia Conti-Cook, and Julie Ciccolini. Layers of bias: A unified approach for understanding problems with risk assessment. *Criminal Justice and Behavior*, 46(2):185–209, 2019.

[68] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019.

[69] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Learning perceptually-aligned representations via adversarial robustness. *arXiv preprint arXiv:1906.00945*, 2019.

[70] Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*, 2019.

[71] Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, N Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem van de Meent. Structured disentangled representations. *International Conference on Artificial Intelligence and Statistics*, 2019.

[72] Shi Feng and Jordan Boyd-Graber. What can ai do for me? evaluating machine learning interpretations in cooperative play. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 229–239, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362726. doi: 10.1145/3301275.3302265. URL https://doi.org/10.1145/3301275.3302265.

[73] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[74] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 1997.

[75] Zoubin Ghahramani and Michael I Jordan. Factorial hidden markov models. In *Advances in Neural Information Processing Systems*, pages 472–478, 1996.

[76] Marzyeh Ghassemi, Mike Wu, Michael C Hughes, Peter Szolovits, and Finale Doshi-Velez. Predicting intervention onset in the icu with switching state space models. *AMIA Summits on Translational Science Proceedings*, 2017, 2017.

[77] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

[78] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/77d2afcb31f6493e350fca61764efb9a-Paper.pdf.

[79] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[80] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[81] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[82] Mark Grundland and Neil A Dodgson. Decolorize: Fast, contrast enhancing, color to grayscale conversion. *Pattern Recognition*, 40(11):2891–2896, 2007.

[83] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[84] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution. https://worldmodels.github.io.

[85] Leif Hancox-Li and I Elizabeth Kumar. Epistemic values in feature importance methods: Lessons from feminist epistemology. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 817–826, 2021.

[86] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10), 1990.

[87] MM Hansen, T Miron-Shatz, AYS Lau, and C Paton. Big data in science and healthcare: a review of recent literature and perspectives. *Yearbook of medical informatics*, 23(01): 21–26, 2014.

[88] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[89] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2009.

[90] Yotam Hechtlinger. Interpretation of prediction models using the input gradient. *arXiv preprint arXiv:1611.07634*, 2016.

[91] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2263–2273, 2017.

[92] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[93] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[94] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

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

[96] Tin Kam Ho. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1. IEEE, 1995.

[97] Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine, and Steven M. Drucker. Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300809. URL https://doi.org/10.1145/3290605.3300809.

[98] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *NeurIPS*, 2019.

[99] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[100] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

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

[102] Yeonwoo Jeong and Hyun Oh Song. Learning discrete and continuous factors of data via alternating disentanglement. *arXiv:1905.09432*, 2019.

[103] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and

Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

[104] Ian T Jolliffe. Principal components in regression analysis. In *Principal component analysis*, pages 129–155. Springer, 1986.

[105] Hyeon-Woo Kang and Hang-Bong Kang. Prediction of crime occurrence from multi-modal data using deep learning. *PloS one*, 12(4):e0176244, 2017.

[106] Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376219. URL https://doi.org/10.1145/3313831.3376219.

[107] Frank C Keil. Explanation and understanding. *Annu. Rev. Psychol.*, 57:227–254, 2006.

[108] Been Kim, Justin Gilmer, Fernanda Viegas, Ulfar Erlingsson, and Martin Wattenberg. Tcav: Relative concept importance testing with linear concept activation vectors. *arXiv preprint arXiv:1711.11279*, 2017.

[109] 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.

[110] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, 2018.

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

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

[113] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[114] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[115] David A. Klindt, Lukas Schott, Yash Sharma, Ivan Ustyuzhaninov, Wieland Brendel, Matthias Bethge, and Dylan Paiton. Towards nonlinear disentanglement in natural data with temporal sparse coding. In *International Conference on Learning Representations*, 2021.

[116] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[117] John F Kolen and Jordan B Pollack. Back propagation is sensitive to initial conditions. In *Advances in neural information processing systems*, 1991.

[118] Josua Krause, Adam Perer, and Kenney Ng. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5686–5697, 2016.

[119] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 2017.

[120] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, 1995.

[121] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 2012.

[122] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? ways explanations impact end users' mental models. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 3–10. IEEE, 2013.

[123] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv:1711.00848*, 2017.

[124] I Elizabeth Kumar, Carlos Scheidegger, Suresh Venkatasubramanian, and S Friedler. Shapley residuals: Quantifying the limits of the shapley value for explanations. In *ICML Workshop on Workshop on Human Interpretability in Machine Learning (WHI)*, 2020.

[125] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2), 2003.

[126] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[127] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[128] Isaac Lage, Andrew Ross, Samuel J Gershman, Been Kim, and Finale Doshi-Velez. Human-in-the-loop interpretability prior. In *Advances in neural information processing systems*, pages 10159–10168, 2018.

[129] Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Sam Gershman, and Finale Doshi-Velez. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*, 2019.

[130] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

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

[132] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 2010.

[133] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

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

[135] Michael Levandowsky and David Winter. Distance between sets. *Nature*, 234(5323): 34–35, 1971.

[136] Clayton Lewis. *Using the" thinking-aloud" method in cognitive interface design*. IBM TJ Watson Research Center Yorktown Heights, NY, 1982.

[137] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. *arXiv preprint arXiv:1506.01066*, 2015.

[138] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

[139] Percy Liang. How should we evaluate machine learning for ai? Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[140] M. Lichman. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2013.

[141] Brian Y Lim, Anind K Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2119–2128, 2009.

[142] Zachary Chase Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016. URL http://arxiv.org/abs/1606.03490.

[143] Anna V Little, Jason Lee, Yoon-Mo Jung, and Mauro Maggioni. Estimation of intrinsic dimensionality of samples from noisy low-dimensional manifolds in high dimensions with multiscale svd. In *IEEE/SP 15th Workshop on Statistical Signal Processing*, 2009.

[144] Yong Liu and Xin Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6), 1999.

[145] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in

the unsupervised learning of disentangled representations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4114–4124, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/locatello19a.html.

[146] Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. *arXiv:2002.02886*, 2020.

[147] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 2020.

[148] Ana C Lorena, Luís PF Garcia, Jens Lehmann, Marcilio CP Souto, and Tin K Ho. How complex is your classification problem? a survey on measuring classification complexity. *arXiv preprint arXiv:1808.03591*, 2018.

[149] Scott Lundberg and Su-In Lee. An unexpected unity among methods for interpreting model predictions. *arXiv preprint arXiv:1611.07478*, 2016.

[150] 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.

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

[152] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[153] Suchismit Mahapatra and Varun Chandola. S-isomap++: multi manifold learning from streaming data. In *IEEE International Conference on Big Data*, 2017.

[154] Barbara I Mahler. Contagion dynamics for manifold learning. *arXiv:2012.00091*, 2020.

[155] Charles Marx, Richard Phillips, Sorelle Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. Disentangling influence: Using disentangled representations to audit model predictions. *Advances in Neural Information Processing Systems*, 2019.

[156] Charles Marx, Flavio Calmon, and Berk Ustun. Predictive multiplicity in classification. In *International Conference on Machine Learning*, pages 6765–6774. PMLR, 2020.

[157] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. *Advances in Neural Information Processing Systems*, 2016.

[158] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[159] Dougal Mclaurin, David Duvenaud, and Matt Johnson. Autograd. https://github.com/HIPS/autograd, 2017.

[160] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning*, 2018.

[161] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

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

[163] Paul Mozur. One Month, 500,000 Face Scans: How China Is Using AI to Profile a Minority. *The New York Times*, 2019. URL https://www.nytimes.com/2019/04/14/technology/china-surveillance-artificial-intelligence-racial-profiling.html.

[164] W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *International Conference on Learning Representations*, 2018.

[165] Aran Nayebi and Surya Ganguli. Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202*, 2017.

[166] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[167] Don Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[168] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

[169] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010. https://distill.pub/2018/building-blocks.

[170] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[171] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. *arXiv preprint arXiv:1901.08846*, 2019.

[172] Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

[173] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[174] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[175] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[176] Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. Learning independent causal mechanisms. In *International Conference on Machine Learning*, 2018.

[177] Judea Pearl. Causal inference. In *Causality: Objectives and Assessment*, pages 39–58, 2010.

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

[179] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. 2009.

[180] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[181] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *arXiv preprint arXiv:1711.10160*, 2017.

[182] Marco Tulio Ribeiro. LIME. `https://github.com/marcotcr/lime`, 2016.

[183] 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.

[184] Karl Ridgeway. A survey of inductive biases for factorial representation-learning. *arXiv preprint arXiv:1612.05299*, 2016.

[185] 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.

[186] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[187] Andrew Ross, Weiwei Pan, and Finale Doshi-Velez. Learning qualitatively diverse and interpretable rules for classification. In *2018 ICML Workshop on Human Interpretability in Machine Learning*, 2018. URL https://arxiv.org/abs/1806.08716.

[188] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[189] Andrew Slavin Ross and Finale Doshi-Velez. Benchmarks, algorithms, and metrics for hierarchical disentanglement. *arXiv preprint arXiv:2102.05185*, 2021.

[190] 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 Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2662–2670, 2017. doi: 10.24963/ijcai.2017/371. URL https://doi.org/10.24963/ijcai.2017/371.

[191] Andrew Slavin Ross, Weiwei Pan, Leo Anthony Celi, and Finale Doshi-Velez. Ensembles of locally independent prediction models. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1911.01291.

[192] Andrew Slavin Ross, Nina Chen, Elisa Zhao Hang, Elena L. Glassman, and Finale Doshi-Velez. Evaluating the interpretability of generative models by interactive reconstruction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021. doi: 10.1145/3411764.3445296.

[193] Leonid Rozenblit and Frank Keil. The misunderstood limits of folk science: An illusion of explanatory depth. *Cognitive science*, 26(5):521–562, 2002.

[194] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215, 2019.

[195] Jeff Sauro and Joseph S Dumas. Comparison of three one-question, post-task usability questionnaires. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1599–1608, 2009.

[196] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2), 1990.

[197] Morgan Klaus Scheuerman, Katta Spiel, Oliver L Haimson, Foad Hamidi, and Stacy M Branham. Hci guidelines for gender equity and inclusivity. https://www.morgan-klaus.com/gender-guidelines.html, 2019.

[198] Ute Schmid, Christina Zeller, Tarek Besold, Alireza Tamaddoni-Nezhad, and Stephen Muggleton. How does predicate invention affect human comprehensibility? In *International Conference on Inductive Logic Programming*, pages 52–67. Springer, 2016.

[199] Jürgen Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

[200] Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2(8):476–486, 2020.

[201] Andrew D Selbst, Danah Boyd, Sorelle A Friedler, Suresh Venkatasubramanian, and Janet Vertesi. Fairness and abstraction in sociotechnical systems. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 59–68, 2019.

[202] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-CAM: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

[203] Lesia Semenova, Cynthia Rudin, and Ronald Parr. A study in rashomon curves and volumes: A new perspective on generalization and model simplicity in machine learning. *arXiv preprint arXiv:1908.01755*, 2019.

[204] Anna Sepliarskaia, Julia Kiseleva, and Maarten de Rijke. Evaluating disentangled representations. *arXiv preprint arXiv:1910.05587*, 2019.

[205] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3288–3291. IEEE, 2012.

[206] Xiaoting Shao, Arseny Skryagin, P Schramowski, W Stammer, and Kristian Kersting. Right for better reasons: Training differentiable models by constraining their influence function. In *Proceedings of Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[207] Hua Shen and Ting-Hao Huang. How useful are the machine-generated interpretations to general users? a human evaluation on guessing the incorrectly predicted labels. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 8, pages 168–172, 2020.

[208] Ron Shoham and Haim Permuter. Amended cross-entropy cost: An approach for encouraging diversity in classification ensemble (brief announcement). In *International Symposium on Cyber Security Cryptography and Machine Learning*, 2019.

[209] 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.

[210] Narayanaswamy Siddharth, Brooks Paige, Jan-Willem Van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, 2017.

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

[212] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[213] Sameer Singh, Marco Tulio Ribeiro, and Carlos Guestrin. Programs as black-box explanations. *arXiv preprint arXiv:1611.07579*, 2016.

[214] Dylan Slack, Sorelle A Friedler, Carlos Scheidegger, and Chitradeep Dutta Roy. Assessing the local interpretability of machine learning models. *arXiv preprint arXiv:1902.03501*, 2019.

[215] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*, 2016.

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

[217] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, et al. Tensorflow. js: Machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350*, 2019.

[218] Peter Sorrenson, Carsten Rother, and Ullrich Köthe. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). *arXiv:2001.04872*, 2020.

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

[220] Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. *arXiv preprint arXiv:2011.12854*, 2020.

[221] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

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

[223] John Sweller. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, 4(4):295–312, 1994.

[224] 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.

[225] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[226] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkZvSe-RZ.

[227] Frederik Träuble, Elliot Creager, Niki Kilbertus, Anirudh Goyal, Francesco Locatello, Bernhard Schölkopf, and Stefan Bauer. Is independence all you need? on the generalization of representations learned from correlated data. *arXiv:2006.07886*, 2020.

[228] Michael Tsang, Sirisha Rambhatla, and Yan Liu. How does this interaction affect me? interpretable attribution for feature interactions. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6147–6159. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/443dec3062d0286986e21dc0631734c9-Paper.pdf.

[229] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[230] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 2016.

[231] Paul E Utgoff. Incremental induction of decision trees. *Machine learning*, 4(2):161–186, 1989.

[232] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

[233] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

[234] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[235] Ethan Weinberger, Joseph Janizek, and Su-In Lee. Learning deep attribution priors based on prior knowledge. *Advances in Neural Information Processing Systems*, 33, 2020.

[236] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019.

[237] Yuxin Wu et al. Tensorpack. https://github.com/tensorpack/, 2016.

[238] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[239] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Twenty-first International Conference on Machine learning*, 2004.

[240] Omar Zaidan, Jason Eisner, and Christine D Piatko. Using "annotator rationales" to improve machine learning for text categorization. In *HLT-NAACL*, pages 260–267. Citeseer, 2007.

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

[242] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[243] Ye Zhang, Iain Marshall, and Byron C Wallace. Rationale-augmented convolutional neural networks for text classification. *arXiv preprint arXiv:1605.04469*, 2016.

[244] Tianyi Zhou, Shengjie Wang, and Jeff A Bilmes. Diverse ensemble evolution: Curriculum data-model marriage. In *Advances in Neural Information Processing Systems 31*. 2018.

Appendix A

Appendix to Chapter 3

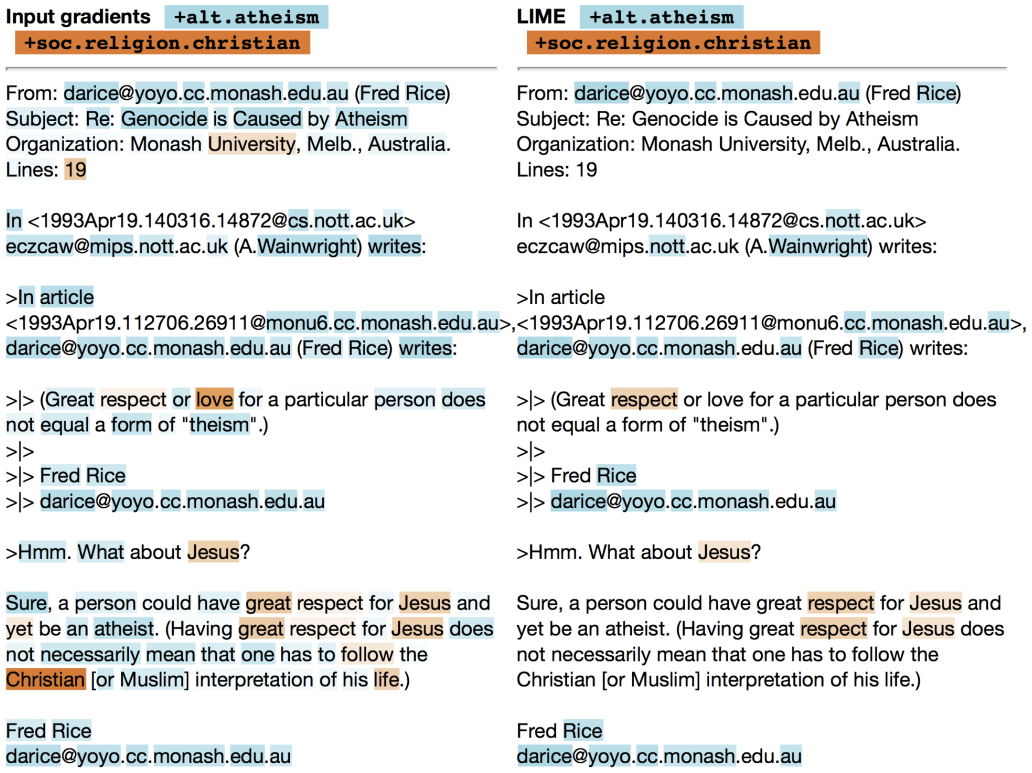


Figure A.1: *Longer 20 Newsgroups example. Blue supports the predicted label, orange opposes it, and* $opacity_i = |w_i| / \max |w|$. *LIME and input gradients never disagree, but gradients may provide a fuller picture of the model's behavior because of LIME's limits on features and samples (especially for long documents).*

Input gradients `+soc.religion.christian` `+alt.atheism`

From: creps@lateran.ucs.indiana.edu (Stephen A. Creps)
Subject: Re: What WAS the immaculate conception
Organization: Indiana University
Lines: 28

In article
<May.14.02.11.19.1993.25177@athos.rutgers.edu>
seanna@bnr.ca (Seanna (S.M.) Watson) wrote:
>I have quite a problem with the idea that Mary never committed a sin.
>Was Mary fully human? If it is possible for God to miraculously make
>a person free of original sin, and free of committing sin their whole
>life, then what is the purpose of the Incarnation of Jesus? Why can't
>God just repeat the miracle done for Mary to make all the rest of us
>sinless, without the need for repentance and salvation and all that?

Yes, Mary is fully human. However, that does not imply that she was just as subject to sin as we are. Catholic doctrine says that man's nature is good (Gen 1:31), but is damaged by Original Sin (Rom 5:12-16). In that case, being undamaged by Original Sin, Mary is more fully human than any of the rest of us.

You ask why God cannot "repeat the miracle" of Mary's preservation from Original Sin. A better way to phrase it would be "why _did_ He not" do it that way, but you misunderstand how Mary's salvation was obtained. Like ours, the Blessed Virgin Mary's salvation was obtained through the merits of the Sacrifice of Christ on the Cross. However, as God is not bound by time, which is His creation, God is free to apply His Sacrifice to anyone at any time, even if that person lived before Christ came to Earth, from our time-bound perspective. Therefore, Christ's Death and Resurrection still served a necessary purpose, and were necessary even for Mary's salvation.

- - - - - - - - - -
Steve Creps, Indiana University
creps@lateran.ucs.indiana.edu

LIME `+soc.religion.christian` `+alt.atheism`

From: creps@lateran.ucs.indiana.edu (Stephen A. Creps)
Subject: Re: What WAS the immaculate conception
Organization: Indiana University
Lines: 28

In article
<May.14.02.11.19.1993.25177@athos.rutgers.edu>
seanna@bnr.ca (Seanna (S.M.) Watson) wrote:
>I have quite a problem with the idea that Mary never committed a sin.
>Was Mary fully human? If it is possible for God to miraculously make
>a person free of original sin, and free of committing sin their whole
>life, then what is the purpose of the Incarnation of Jesus? Why can't
>God just repeat the miracle done for Mary to make all the rest of us
>sinless, without the need for repentance and salvation and all that?

Yes, Mary is fully human. However, that does not imply that she was just as subject to sin as we are. Catholic doctrine says that man's nature is good (Gen 1:31), but is damaged by Original Sin (Rom 5:12-16). In that case, being undamaged by Original Sin, Mary is more fully human than any of the rest of us.

You ask why God cannot "repeat the miracle" of Mary's preservation from Original Sin. A better way to phrase it would be "why _did_ He not" do it that way, but you misunderstand how Mary's salvation was obtained. Like ours, the Blessed Virgin Mary's salvation was obtained through the merits of the Sacrifice of Christ on the Cross. However, as God is not bound by time, which is His creation, God is free to apply His Sacrifice to anyone at any time, even if that person lived before Christ came to Earth, from our time-bound perspective. Therefore, Christ's Death and Resurrection still served a necessary purpose, and were necessary even for Mary's salvation.

- - - - - - - - - -
Steve Creps, Indiana University
creps@lateran.ucs.indiana.edu

Figure A.2: *Even longer 20 Newsgroups example that highlights LIME's limitations on long documents.*

Appendix B

Appendix to Chapter 4

B.1 Imposing Penalties over Manifolds

In the beginning of our derivation of `CosIndepErr` (Equation 4.4), we assumed that locally, $N_\epsilon(x) \approx \mathcal{B}_\epsilon(x)$. However, in many cases, our data manifold Ω_x is much lower dimensional than $\mathbb{R}^D$. In these cases, we have additional degrees of freedom to learn decision boundaries that, while locally orthogonal, are functionally equivalent over the dimensions which matter. To restrict spurious similarity, we can project our gradients down to the data manifold. Given a local basis for its tangent space, we can accomplish this by taking dot products between ∇f and ∇g and each tangent vector, and then use these two vectors of dot products to compute the cosine similarity in Equation 4.6. More formally, if $J(x)$ is the Jacobian matrix of manifold tangents at x, we can replace our regular cosine penalty with

$$\begin{aligned} \texttt{ManifIndepErr}(f, g) &\equiv \mathbb{E}\left[\cos^2(\nabla f\|_J(x), \nabla g\|_J(x)),\right] \\ \text{where} \quad \nabla f\|_J(x) &= \nabla f(x)^\intercal J(x), \\ \nabla g\|_J(x) &= \nabla g(x)^\intercal J(x) \end{aligned} \tag{B.1}$$

An example of this method applied to a toy example is given in Figure B.1. Alternatively, if we are using projected gradient descent adversarial training to minimize the original formulation in Equation 4.2, we can modify its inner optimization procedure to project

input gradient updates back to the manifold.

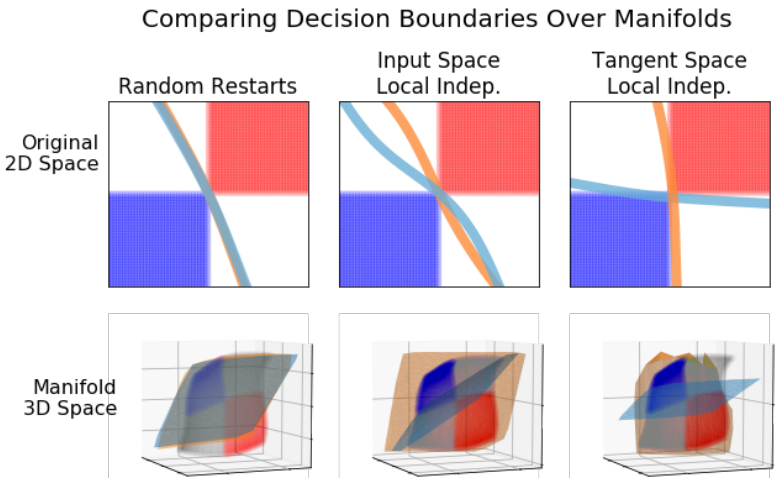


Figure B.1: *Synthetic 2D manifold dataset (randomly sampled from a neural network) embedded in $\mathbb{R}^3$, with decision boundaries shown in 2D chart space (top) and the 3D embedded manifold space (bottom). Naively imposing LIT penalties in $\mathbb{R}^3$ (middle) leads to only slight differences in the chart space decision boundary, but given knowledge of the manifold's tangent vectors (right), we can recover maximally different chart space boundaries.*

For many problems of interest, we do not have a closed form expression for the data manifold or its tangent vectors. In this case, however, we can approximate one, e.g. by performing PCA or training an autoencoder. Local independence training can also simply be used on top of this learned representation directly.

B.2 Additional Figures

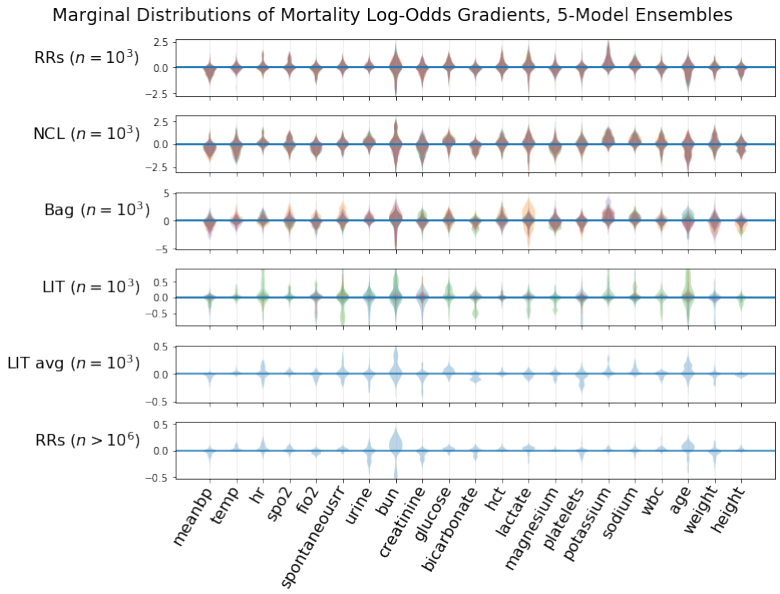


Figure B.2: *Violin plots showing marginal distributions of ICU mortality input gradients across heldout data for 5-model ensembles trained on the $n = 1000$ slice (top 5 plots) and restarts on the full dataset (bottom). Distributions for each model in each ensemble are overlaid with transparency in the top 4 plots. From the top, we see that restarts and NCL learn models with similar gradient distributions. Bagging is slightly more varied, but only LIT (which performs significantly better on the prediction task) exhibits significant differences between models. When LIT gradients on this limited data task are averaged (second from bottom), their distribution comes to resemble (in both shape and scale) that of a model trained on the full dataset (bottom), which may explain LIT's stronger performance.*

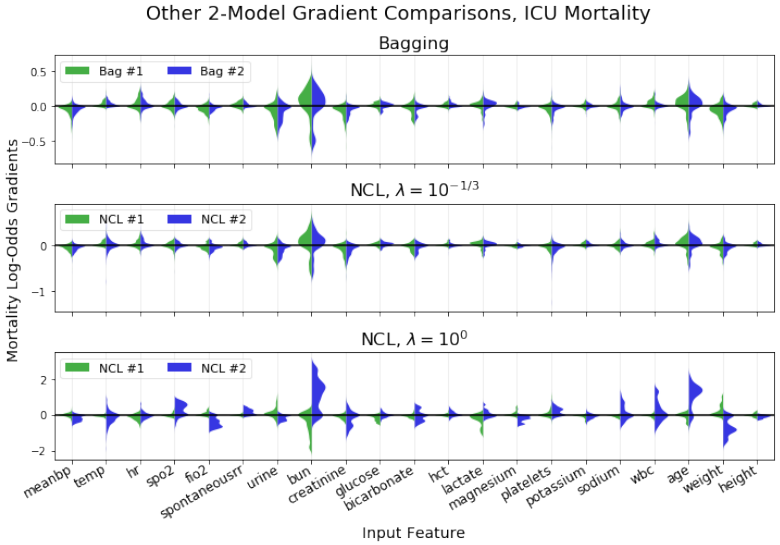


Figure B.3: *Companion to Figure 4.8 showing differences in the distributions of input gradients for other 2-model ensemble methods. Bagging is largely identical to random restarts, while NCL exhibits a sharp transition with λ.*

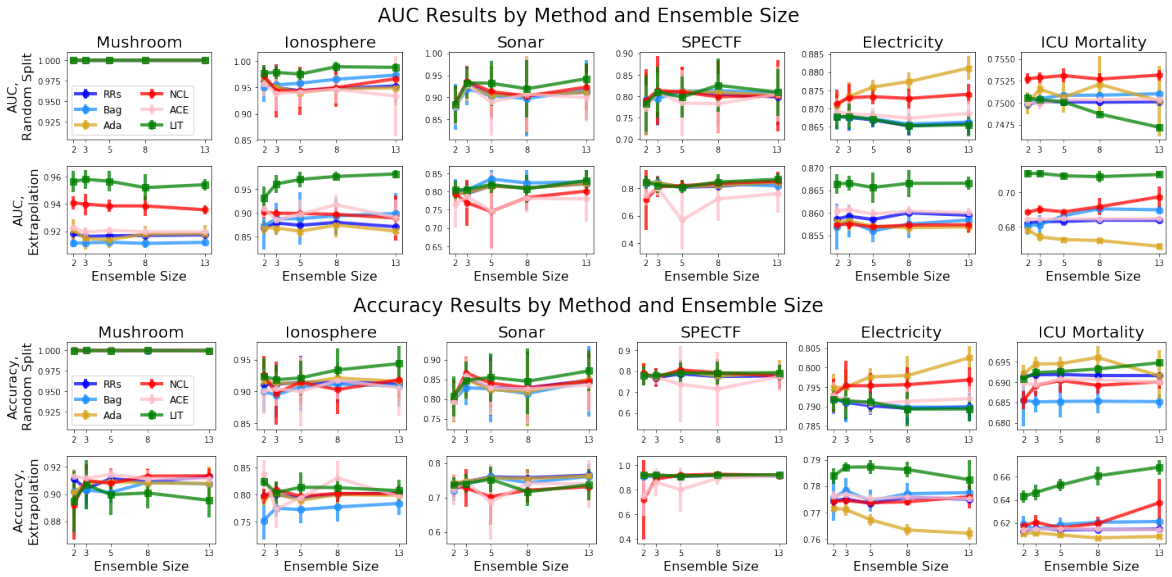


Figure B.4: *Full ensemble AUC and accuracy results by method and ensemble size. LIT usually beats baselines when train $\neq$ test, but the optimal ensemble size (cross-validated in the main paper, but expanded here) can vary.*

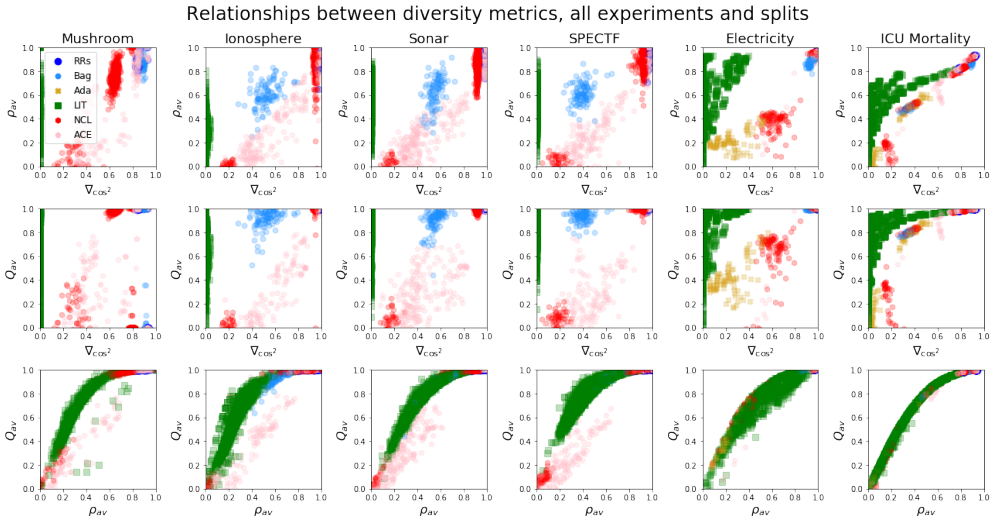


Figure B.5: *Empirical relationship between our similarity metric (or penalty) $\nabla_{\cos^2}$ and more classic measures of prediction similarity such as error correlation (ρ_{av}) and the Q-statistic (Q_{av}), with one marker for every method, λ, dataset, split, ensemble size, and restart. In general, we find meaningful relationships between $\nabla_{\cos^2}$ and classic diversity metrics, despite the fact that $\nabla_{\cos^2}$ does not require ground-truth labels. The bottom row of this figure also shows that LIT models (green) tend to have lower and more widely varying Q_{av} and ρ_{av}, indicating greater ability to control heldout prediction diversity through training λ. We also measured the interrater agreement κ but we found the results almost identical to ρ_{av} and omit them to save space.*

Appendix C

Appendix to Chapter 7

C.1 Additional Model Details

In this section, we present additional background and details about the representation learning models considered.

C.1.1 Loss Functions.

Autoencoders [2] (AEs) are trained simply to compress and reconstruct x:

$$\mathcal{L}_{\text{AE}}(x) = \mathcal{L}(x, \texttt{dec}(\texttt{enc}(x))),$$

where $\mathcal{L}$ is an individual example reconstruction loss. We used cross-entropy (Bernoulli negative log-likelihood) for dSprites and MNIST and mean-squared error (Gaussian negative log-likelihood) for Sinelines.

Ground-truth models (GTs) are trained equivalently to the autoencoder, except we omit the encoder and instead provide ground-truth factors z alongside x:

$$\mathcal{L}_{\text{GT}}(x, z) = \mathcal{L}(x, \texttt{dec}(z))$$

Note that on Sinelines, we did not actually *train* the GT model because in that case, it was simple to implement in closed form. It might have been possible to implement the GT

dSprites model this way as well, but since the original generating code is not available in the dataset repository [158], we opted for a model.

Variational autoencoders [113] (VAEs) are similar to autoencoders, except that $\texttt{enc}(x)$ outputs not a single value, but a distribution over z. The VAE training objective includes an expectation of the reconstruction error over this distribution, as well as a regularization term meant to ensure that $\texttt{enc}(x)$ stays close to a prior $p(z)$ (in our case, an isotropic unit Gaussian):

$$\mathcal{L}_{\text{VAE}}(x) = \mathbb{E}_{z\sim\texttt{enc}(x)}\left[\mathcal{L}(x, \texttt{dec}(z))\right] + \text{KL}(\texttt{enc}(x)||p(z))$$

Note that the expectation is generally approximated in training with a single sample.

β-total correlation autoencoders (β-TCVAEs, abbreviated further to TC) [45] are identical to VAEs, except that an additional penalty is applied to the total correlation between representation dimensions:

$$\mathcal{L}_{\text{TC}}(x) = \mathcal{L}_{\text{VAE}}(x) + (\beta - 1)\text{TC}(\texttt{enc}(x)),$$

where for some joint distribution $q(z)$, the total correlation $\text{TC}(q(z))$ is equivalent to the KL divergence between $q(z)$ and the product of its marginals, $\text{KL}(q(z)||\prod_j q(z_j))$. Penalizing total correlation reduces statistical dependence between dimensions and is thought to improve interpretability. We use an approximation of this objective developed by Chen et al. [45] but there are others, e.g. Kim et al. [110]. For all experiments, we used $\beta = 10$.

Our semi-supervised (SS) variant of the β-total correlation autoencoder augments its encoded representation z with an additional categorical dimension where we explicitly provide the class label y:

$$\begin{aligned}\mathcal{L}_{\text{SS}}(x, y) = \mathbb{E}_{z\sim\texttt{enc}(x)}&\left[\mathcal{L}(x, \texttt{concat}(y, \texttt{dec}(z)))\right] \\ &+ \text{KL}(\texttt{enc}(x)||p(z)) \\ &+ (\beta - 1)\text{TC}(\texttt{enc}(x))\end{aligned}$$

By providing y as side-information during training, we effectively disentangle digit identity from the continuous part of the representation z, making the SS slightly more similar to

ground-truth (even though there is no complete ground-truth for MNIST, the dataset we consider). The SS model's representation of digit style may still be entangled, but hopefully less so than other models (as the SS model still employs all the same tricks as the TC model).

Finally, InfoGAN [46] (IG) is a generative adversarial network [80] trained to reach equilibrium between two losses: a discriminator attempting to distinguish real from fake images, and generator attempting to create images x that fool the discriminator from a latent code z (with maximal information between "interpretable" components of z and the generated image). We refer to the original citation for more details [46].

C.1.2 Training Details.

All autoencoder models are trained in Tensorflow with the Adam optimizer [112], with a batch size of 64 or 128 (for MNIST vs. others) and for the minimum number of epochs necessary to surpass 100,000 iterations. Matching Burgess et al. [35], the learning rate was set to 0.0005 for dSprites and its Tensorflow-default value of 0.001 for MNIST and Sinelines. InfoGANs were trained using an implementation from the Tensorpack library [237]. See https://github.com/dtak/interactive-reconstruction for code.

C.2 Distance Metrics and Thresholds

In this section, we describe our choices of distance metric $d(x, x')$ and threshold distance ϵ for each of the three datasets. In general, for all three datasets, we used metrics that measured the fraction of disagreeing dimensions, with dataset-specific definitions of disagreement.

C.2.1 Sinelines

For Sinelines, we defined distance as the fraction of inputs that disagreed by more than 0.5 (approximately 2% of the range of x):

$$d_{\text{Sinelines}}(x, x^*) \triangleq \frac{1}{64} \sum_{i=1}^{64} \mathbb{1}(|x_i - x_i^*| > 0.5) \tag{C.1}$$

This metric has the advantage of being between 0 and 1, which allows us to visualize users' proximity to solving each problem with a progress bar (after subtracting the distance from 1). We set the threshold distance value ϵ to 0.1 (which we visualized as a 90% agreement target).

C.2.2 dSprites

For dSprites, a BW image dataset where a large fraction of pixels are always within almost any threshold value due to black backgrounds, we instead used L1 distance normalized by the total difference away from black backgrounds (which corresponds to Bray-Curtis similarity [24], and can be seen as a relaxation of intersection-over-union):

$$d_{\text{dSprites}}(x, x^*) \triangleq \frac{\sum_i |x_i - x_i^*|}{\sum_i |x_i| + |x_i^*|} \tag{C.2}$$

We again used the threshold of $\epsilon = 0.1$ and visualized progress in terms of a 90% alignment target.

C.2.3 MNIST

For MNIST, we initially tried the same distance metric as with dSprites, but found in pilot experiments that the soft relaxation of intersection-over-union led to confusing behavior with highly regularized architectures like the β-total correlation autoencoder, which often have relatively gradual transitions between black and white portions of generated images. Instead, we opted to binarize autoencoder outputs in our visualization and use an exact intersection-over-union similarity metric:

$$d_{\text{MNIST}}(x, x') \triangleq 1 - \frac{\sum_i \lfloor x_i \rceil \wedge \lfloor x_i' \rceil}{\sum_i \lfloor x_i \rceil \vee \lfloor x_i' \rceil}, \tag{C.3}$$

with a threshold of $\epsilon = 0.25$ (visually presented as an alignment target of 75%). We chose this threshold by asking pilot users to align x and x' without a target threshold and verbally indicating when they felt they were "far away" vs. "close enough," and then finding a value that consistently separated those two states across examples.

C.3 Additional Screenshots

In Figures C.1, C.2, and C.3, we show additional screenshots for more tasks, models, and datasets. We also showcase the instructional practice questions, where users were shown a simple "circles" dataset with just two factors of variation (radius and foreground/background color). Note that all of the tasks are available at `http://hreps.s3.amazonaws.com/quiz/manifest.html`.

Table C.1: *Labels assigned to MNIST $D_z = 10$ representation dimensions.*

Model	Dimension	User-Assigned Labels
AE_{10}	Continuous 1	"cross"
	Continuous 4	"intensity", "0-1"
	Continuous 7	"thickness"
TC_{10}	Continuous 1	"numbers", "reflect but not really", "number?", "little"
	Continuous 2	"x", "ignore", "nothing", "no"
	Continuous 3	"thickness", "numb"
	Continuous 4	"expands", "numb"
	Continuous 5	**"makes things thick", "thickness", "thicker"**
	Continuous 6	"x", "ignore", "nothing", "nothing", "no"
	Continuous 7	"x", "ignore", "nothing", "nothing", "no"
	Continuous 8	**"tilt on center", "rotation", "orient"**
	Continuous 9	"squish into the center", "numb"
	Continuous 10	"six", "stretch", "0"
SS_{10}	Discrete 1	**"Number", "Number", "number"**
	Continuous 1	"spirals", "lengthening", "width"
	Continuous 2	"curviness", "y bubble"
	Continuous 3	"x", "ignore", "nothing", "no"
	Continuous 4	**"thickness", "thick", "thickness", "thick", "thickness"**
	Continuous 5	"rotation"
	Continuous 6	"x", "ignore", "nothing", "little"
	Continuous 7	"x", "moves a thing", "no"
	Continuous 8	"tilt", "squishes"
	Continuous 9	**"semi tilt", "rotation"**
	Continuous 10	"x", "migrates one thing", "nothing", "little"

All labels assigned to $D_z = 10$ models by participants in the MNIST thinkaloud study. Bold font shows labels experimenters identified as consistent between participants, with dimensions that had little effect on representations due to TC regularization shown in gray. TC and SS models had more labels (and more consistent labels) than AEs.

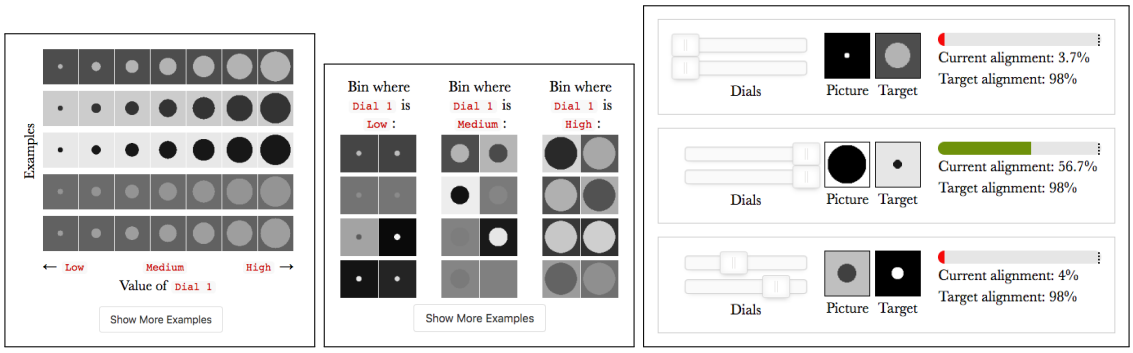


Figure C.1: *Traversals (left), exemplars (middle), and interactive reconstruction (right) for practice question problem.*

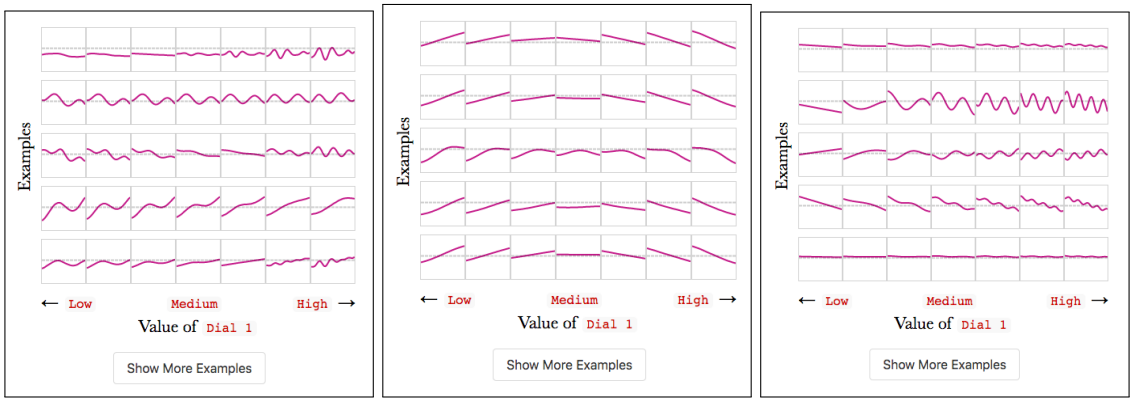


Figure C.2: *Sinelines traversals for a random dimension of the AE (left), VAE (middle), and GT model (right).*

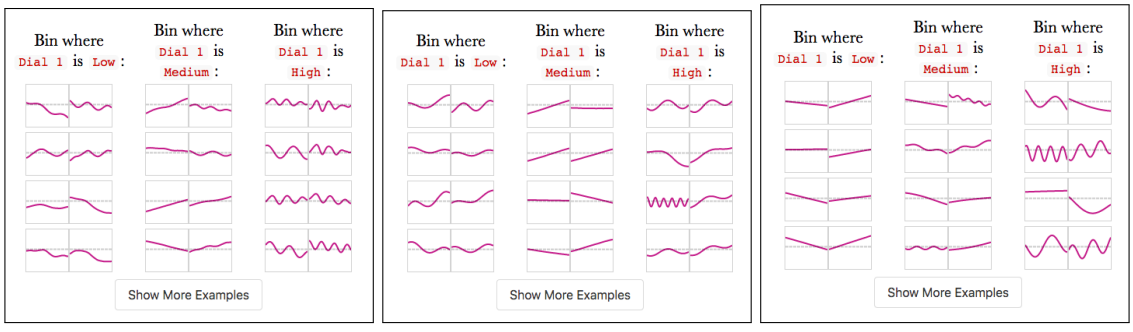


Figure C.3: *Sinelines exemplars for a random dimension of the AE (top), VAE (middle), and GT model (bottom).*

Appendix D

Appendix to Chapter 8

D.1 Training and Architecture Details

For Chopsticks, our encoders and decoders used two hidden layers of width 256, and our loss function $\mathcal{L}_x$ was defined as a zero-centered Gaussian negative log likelihood with $\sigma = 0.1$. For Spaceshapes, encoders and decoders used the 7-layer convolutional architecture from Burgess et al. [2018], and our loss function $\mathcal{L}_x$ was Bernoulli negative log likelihood. All models were implemented in Tensorflow. Code to reproduce experiments will be made available at `https://github.com/dtak/hierarchical-disentanglement`.

For both models, the assignment loss $\mathcal{L}_a$ was set to mean-squared error, but only for assignments that were defined. This was implemented by setting undefined assignment components to -1, and then defining $\mathcal{L}_a(a, a') = \sum_i \mathbb{1}[a'_i \geq 0](a_i - a'_i)^2$.

All activation functions were set to ReLU ($\max(0, x)$), except in the case of the initial smooth autoencoder, where they were replaced with Softplus ($\ln(1 + e^x)$). This initial autoencoder was trained with dimensionality equal to one plus the maximum intrinsic dimensionality of the dataset. We investigate varying this parameter in Fig. D.3 and find it can be much larger, and perhaps would have produced better results (though nearest neighbor calculation and local SVD computations would have been slower).

All models were trained for 50 epochs with a batch size of 256 on a dataset of size

100,000, split 90%/10% into train/test. We used the Adam optimizer with a learning rate starting at 0.001 and decaying by $\frac{1}{10}$ halfway and three-quarters of the way through training.

For COFHAE, we selected softmax temperature τ, the assignment penalty strength λ_1, and the adversarial penalty strength λ_2 based on *training set* reconstruction error and MIMOSA assignment accuracy. Splitting off a separate validation set was not necessary, as the most common problem we faced was poor convergence, not overfitting; the adversarial penalty would dominate and prevent the procedure from learning a model that could reconstruct X or A.

Specifically, for each restart, we ran COFHAE with τ in $\{\frac{1}{2}, \frac{2}{3}, 1\}$, λ_1 in $\{10, 100, 1000\}$, and λ_2 in $\{1, 10, 100\}$. We then selected the model with the lowest training MSE $\sum_n ||x_n - x'_n||_2^2$, but restricting ourselves to the 33.3% of models with the lowest assignment loss $\sum_n \mathcal{L}_a(a_n, a'_n)$.

For evaluating R^4 and R^4_c, we used gradient boosted decision trees, which were faster to train than neural networks.

D.2 Complexity and Runtimes

Per Fig. D.2, the total runtime of our method is dominated by COFHAE, an adversarial autoencoder method which has the same complexity as FactorVAE [110] (linear in dataset size N and number of training epochs, and strongly affected by GPU speed).

MIMOSA could theoretically take more time, however, as the complexity of constructing a ball tree [170] for nearest neighbor queries is $O(|Z|N \log N)$. As such, initial dimensionality reduction is critical; in our Spaceshapes experiments, $|Z|$ is 7, whereas $|X|$ is 4096.

Other MIMOSA steps can also take time. With a `num_nearest_neighbors` of k, the complexity of running local SVD on every point in the dataset is $O(N(|Z|^2 k + |Z|k^2 + k^3))$, providing another reason to reduce initial dimensionality and keep neighborhood size manageable (though ideally k should increase with $|Z|$ to robustly learn local manifold directions). Iterating over the dataset in BuildComponent and computing cosine similarity will also have complexity at least $O(Nkd^3(d + |Z|))$ for components of local dimensionality

d, and detecting component boundaries can actually have complexity $O(Nke^d)$ (if this is implemented, as in our experiments, by checking if projected points are contained in their neighbors' convex hulls—though we also explored a much cheaper $O(Nk^2d)$ strategy of checking for the presence of neighbors in all principal component directions that worked almost as well).

Although these scaling issues are worth noting, MIMOSA was still relatively fast in our experiments, where runtimes were dominated by neural network training (Fig. D.2).

Algorithm 3 LocalSVD(Z)

Run SVD on Z (a design matrix of dimension `num_nearest_neighbors` by `initial_dim`)
if `ransac_frac` < 1 **then**
  for each dimension d from 1 to `initial_dim` $- 1$ **do**
    for each point z_n **do**
      Compute the reconstruction error for z_n using the only first d SVD dimensions
    end for
  end for
  Take the norm of reconstruction errors across dimensions, giving a vector of length `num_nearest_neighbors`
  Re-fit SVD on points whose error-norms are less than the $100 \times$ `ransac_frac` percentile value.
end if
for each dimension d from 1 to `initial_dim` $- 1$ **do**
  Check if the cumulative sum of the first d eigenvalues is at least `eig_cumsum_thresh`
  Check if the ratio of the dth to the $d + 1$st eigenvalue is at least `eig_decay_thresh`
  if both of these conditions are true **then**
    return only the first d SVD components
  end if
end for
return the full set of SVD components otherwise

Algorithm 4 TangentPlaneCos(U, V)

if U and V are equal-dimensional **then**
  return $|\det(U \cdot V^T)|$
else
  return 0
end if

Algorithm 5 BuildComponent(z_i, neighbors, svds)

1: Initialize component to z_i and neighbors z_j not already in other components where TangentPlaneCos($\text{svds}_i, \text{svds}_j$) $\geq$ `cos_simil_thresh` (Algorithm 4).
2: **while** the component is still growing **do**
3: Add all points z_k for which at least `contagion_num` of their neighbors z_ℓ are already in the component with TangentPlaneCos($\text{svds}_k, \text{svds}_\ell$) $\geq$ `cos_simil_thresh`.
4: Skip adding any z_k already in another component.
5: **end while**
6: **return** the set of points in the component

Algorithm 6 MergeComponents(components, svds)

1: Discard components smaller than `min_size_init`.
2: **for** each component c_i **do**
3: Construct a local ball tree for the points in c_i.
4: Set c_i.edges to points not contained in the convex hull of their neighbors in local SVD space.
5: **end for**
6: Initialize edge overlap matrix M of size |components| by |components| to 0.
7: **for** each ordered pair of equal-dimensional components (c_i, c_j) **do**
8: Set M_{ij} to the fraction of points in c_i.edges for which the closest point in c_j.edges has local SVD tangent plane similarity above `cos_simil_thresh`.
9: **end for**
10: Average M with its transpose to symmetrize.
11: Merge all components $c_i \neq c_j$ of equal dimensionality d where $M_{ij} \geq$ `min_common_edge_frac`(d).
12: Discard components smaller than `min_size_merged`.
13: **return** the merged set of components

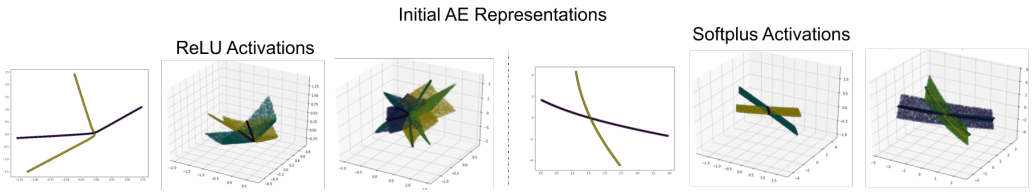


Figure D.1: *Comparison of the latent spaces learned by MIMOSA initial autoencoders with ReLU (top) vs. Softplus (bottom) activations. Each plot shows encoded Chopsticks data samples colored by their ground-truth location in the dimension hierarchy. Because ReLU activations are non-differentiable at* 0*, the resulting latent manifolds contain sharp corners, which makes it difficult for MIMOSA's local SVD procedure to merge points together into the correct components.*

Algorithm 7 ConstructHierarchy(components)

for each component c_i **do**
  Set c_i.neighbor_lengthscale to the average distance of points to their nearest neighbors inside the component (computed using the local ball tree from Algorithm 6)
end for
for each pair of different-dimensional components (c_i, c_j), c_i higher-dimensional **do**
  Compute the average distance from points in c_i to their nearest neighbors in c_j (via ball tree).
  Divide this average distance by c_i.neighbor_lengthscale.
  if the resulting ratio $\leq$ `neighbor_lengthscale_mult` **then**
    Set $c_j \in c_i$ (c_j is enclosed by c_i)
  end if
end for
Create a root node with edges to all components which do not enclose others.
Transform the component enclosure DAG into a tree (where enclosing components are children of enclosed components) by deleting edges which:

1. are redundant because an intermediate edge exists, e.g. if $c_1 \in c_2 \in c_3$, we delete the edge between c_1 and c_3.
2. are ambiguous because a higher-dimensional component encloses multiple lower-dimensional components (i.e. has multiple parents). In that case, preserve only the edge with the lowest distance ratio.

Convert the resulting component enclosure tree into a dimension hierarchy:

1. If the root node has only one child, set it to be the root. Otherwise, begin with a dimension group with a single categorical dimension whose options point to groups containing each child.
2. For the rest of the component tree, add continuous dimensions until the total number of continuous dimensions up to the root equals the component's dimensionality.
3. If a component has children, add a categorical dimension that includes those child groups as options (recursing down the tree), along with an empty group ($\boxed{\emptyset}$) option.

return the dimension hierarchy

Algorithm 8 $\text{HAE}_\theta.\text{encode}(x; \tau)$

1: Encode x using any neural network architecture as a flat vector z_{pre}, with size equal to the number of continuous variables plus the number of categorical options in HAE_θ.hierarchy.
2: Associate each group of dimensions in the flat vector with variables in the hierarchy.
3: For all of the categorical variables, pass their options through a softmax with temperature τ.
4: Use the resulting vector to mask all components of z_{pre} corresponding to variables *below* each option in HAE_θ.hierarchy.
5: **return** the masked representation, separated into discrete a', continuous z, as well as the mask m (for determining active dimensions later).

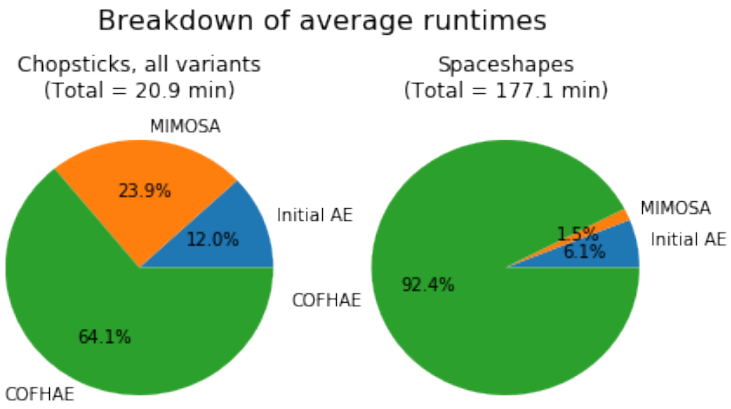


Figure D.2: *Mean runtimes and percentage breakdowns for COFHAE and MIMOSA on Chopsticks and Spaceshapes, based on Tensorflow implementations running on single GPUs (exact model varies between Tesla K80, Tesla V100, GeForce RTX 2080, etc). Runtimes tend to be dominated by COFHAE, which is similar in complexity to existing adversarial methods (e.g. FactorVAE).*

D.3 MIMOSA Hyperparameters

In this section, we list and describe all hyperparameters for MIMOSA, along with values that we used for our main results. We also present sensitivity analyses in Fig. D.3.

MIMOSA initial autoencoder (Algorithm 1, line 1)

- `initial_dim` - the dimensionality of the initial smooth autoencoder. As Fig. D.3 shows, this can be larger than the intrinsic dimensionality of the data, which MIMOSA will estimate. We defaulted to using the max. intrinsic dimensionality plus 1; in a real-world context where this information is not available, it can be estimated by reducing from `initial_dim` $= |X|$ until reconstruction error starts increasing.
- Training and architectural details appropriate for the data modality (e.g. convolutional layers for images). See §D.1 for our choices.

LocalSVD (Algorithm 3)

- `num_nearest_neighbors` - the neighborhood size for local SVD and later traversal. We used 40. Must be larger than `initial_dim`; could also be replaced with a search radius.
- `ransac_frac` - the fraction of neighbors to refit SVD. We used 2/3. Note that we do not run traditional multi-step RANSAC [73], but a two-step approximation, where we define loss by aggregating reconstruction errors across dimensions. Another (slower but potentially more robust) option would be to iteratively refit SVD on the points with lowest reconstruction error at each dimension, and check if the resulting eigenvalues meet our cutoff criteria.
- `eig_cumsum_thresh` - the minimum fraction of variance SVD dimensions must explain to determine local dimensionality. We used 0.95. For noisy or sparse data, it might be useful to reduce this parameter.
- `eig_decay_thresh` - the minimum multiplicative factor by which SVD eigenvalues must decay to determine local dimensionality. We used 4. It might also be useful to

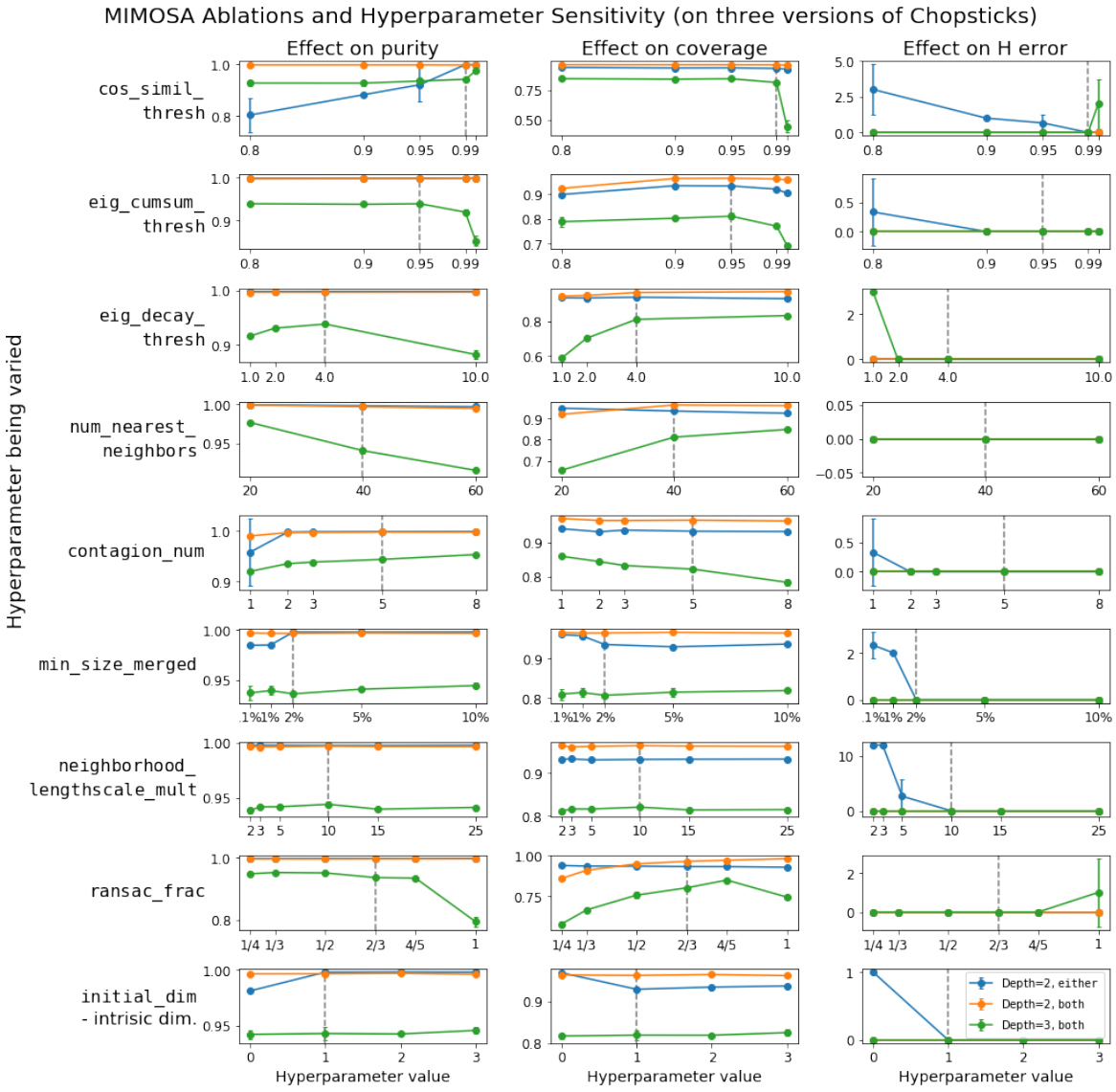


Figure D.3: *Effect of varying different hyperparameters (and ablating different robustness techniques) on MIMOSA. Default values are shown with vertical gray dotted lines, and for each hyperparameter (top to bottom), average coverage (left), purity (middle), and H error (right) when deviating from defaults are shown for three versions of the Chopsticks dataset. Results suggest both a degree of robustness to changes (degradations tend not to be severe for small changes), but also the usefulness of various components; for example, results markedly improve on some datasets with* `contagion_num`>1 *and* `ransac_frac`<1 *(implying contagion dynamics and RANSAC both help). Many parameters exhibit tradeoffs between component purity and dataset coverage.*

reduce this parameter for sparse data.

Note that our LocalSVD algorithm can be seen as a faster version of Multiscale SVD [143], which is used in an analogous way by Mahapatra and Chandola [2017], but would require repeatedly computing singular value decompositions over different search radii for each point.

BuildComponent (Algorithm 5)

- `cos_simil_thresh` - neighbors' local SVDs must be this similar to add to the component. This corresponds to the ϵ parameter from Mahapatra and Chandola [2017]. We used 0.99 for Chopsticks and 0.95 for Spaceshapes; in general, we feel this is one of the most important parameters to tune, and should generally be reduced in the presence of noise or data scarcity.
- `contagion_num` - only add similar points to a manifold component when a threshold fraction of their neighbors have already been added. This is useful for robustness, and corresponds to the T parameter from Mahler [2020] (but expressed as a number rather than a fraction). We used 5 for Chopsticks and 3 for Spaceshapes. Values above 20% of `num_nearest_neighbors` will likely produce poor results, and we found the greatest increases in robustness just going from 1 (or no contagion dynamics) to 2.

MergeComponents (Algorithm 6)

- `min_size_init` - discard initial components smaller than this. We used 0.02% of the dataset size, or about 20 points. This parameter helps speed up the algorithm (by reducing the number of pairwise comparisons) and avoids incorrect merges through single-point components.
- `min_size_merged` - discard merged components smaller than this. We used 2% of the dataset size, or about 2000 points. This parameter helps exclude spurious high-dimensional interstitial points that appear at boundaries where low-dimensional components intersect.

- `min_common_edge_frac(d)` - the minimum fraction of edges that two manifold components must share in common to merge, as a function of dimensionality d. We used $2^{-d-1} + 2^{-d-2}$; this is based on the idea that two neighboring (possibly distorted) hypercubes of dimension d should match on one of their sides; since they have 2^d sides, the fraction of matching edge points would be 2^{-d}. However, for robustness (as not all components will be hypercubes, and even then some edge points may not match), we average this with the smaller fraction that a $d+1$ dimensional hypercube would need, or 2^{-d-1}, for our resulting $2^{-d-1} + 2^{-d-2}$. We found that this choice was not critical in preliminary experiments, as matches were common for components with the same true assignments and rare for others, but it could become more important for sparse or noisy data.

ConstructHierarchy (Algorithm 7)

- `neighbor_lengthscale_mult` - the threshold for deciding whether a higher-dimensional component "encloses" a lower-dimensional component, expressed as a ratio of (1) the average distance from lower-dimensional component points to their nearest neighbors in the higher-dimensional component (inter-component distance), to (2) the average distance of points in the higher-dimensional component to their nearest neighbors in that same component (intra-component distance). We used 10, which we found was robust for our benchmarks, though it may need to be increased if ground-truth components are higher-dimensional than those in our benchmarks.

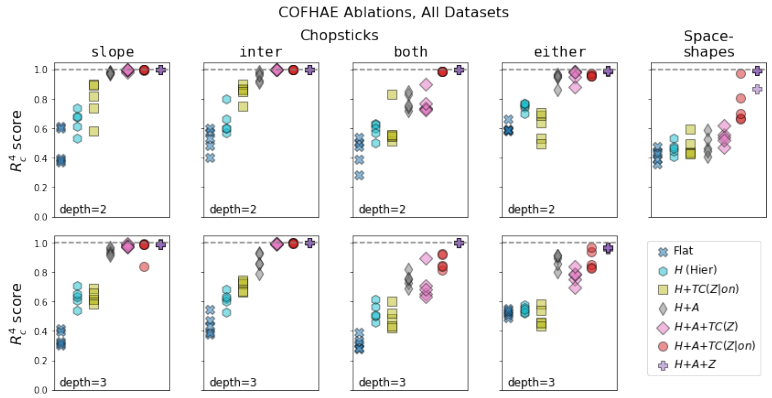


Figure D.4: *A fuller version of main paper Fig. 8.4 showing COFHAE ablations on all datasets. Hierarchical disentanglement tends to be low for flat AEs (Flat), better with ground-truth hierarchy H (Hier H), and even better after adding supervision for ground-truth assignments A (H+A). Adding a FactorVAE-style marginal TC penalty (H+A+TC(Z)) sometimes helps disentanglement, but making that TC penalty conditional (H+A+TC(Z|on), i.e. COFHAE) tends to help more, bringing it close to the near-optimal disentanglement of a hierarchical model whose latent representation is fully supervised (H+A+Z). Partial exceptions include the hardest three datasets (Spaceshapes and depth-3 compound Chopsticks), where disentanglement is not consistently near 1; this may be due to non-identifiability or adversarial optimization difficulties.*

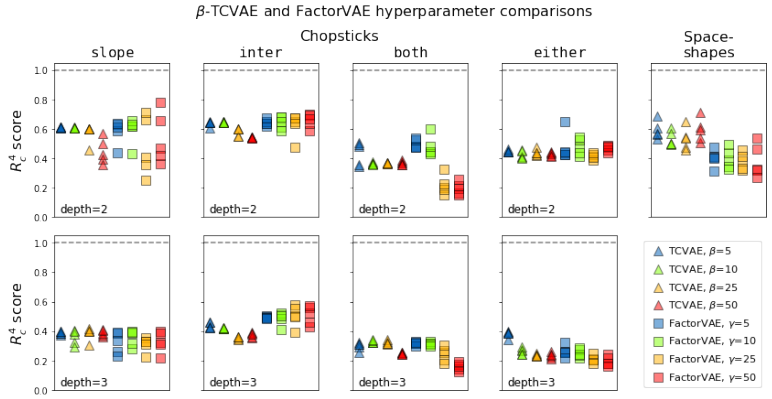
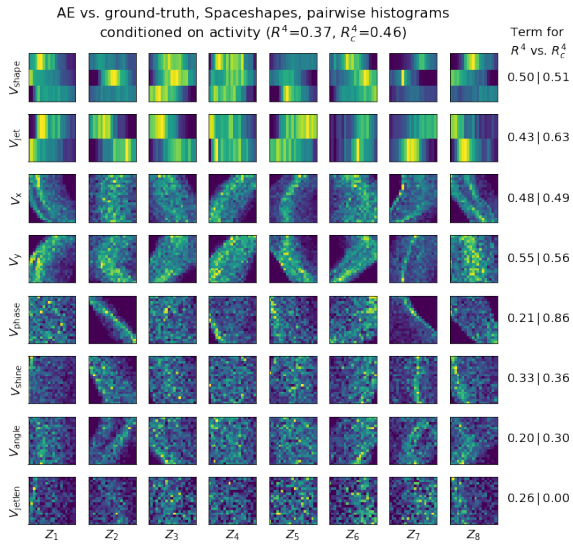
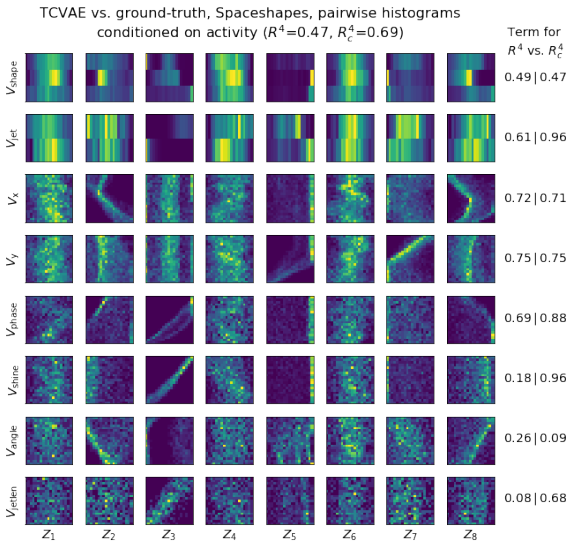


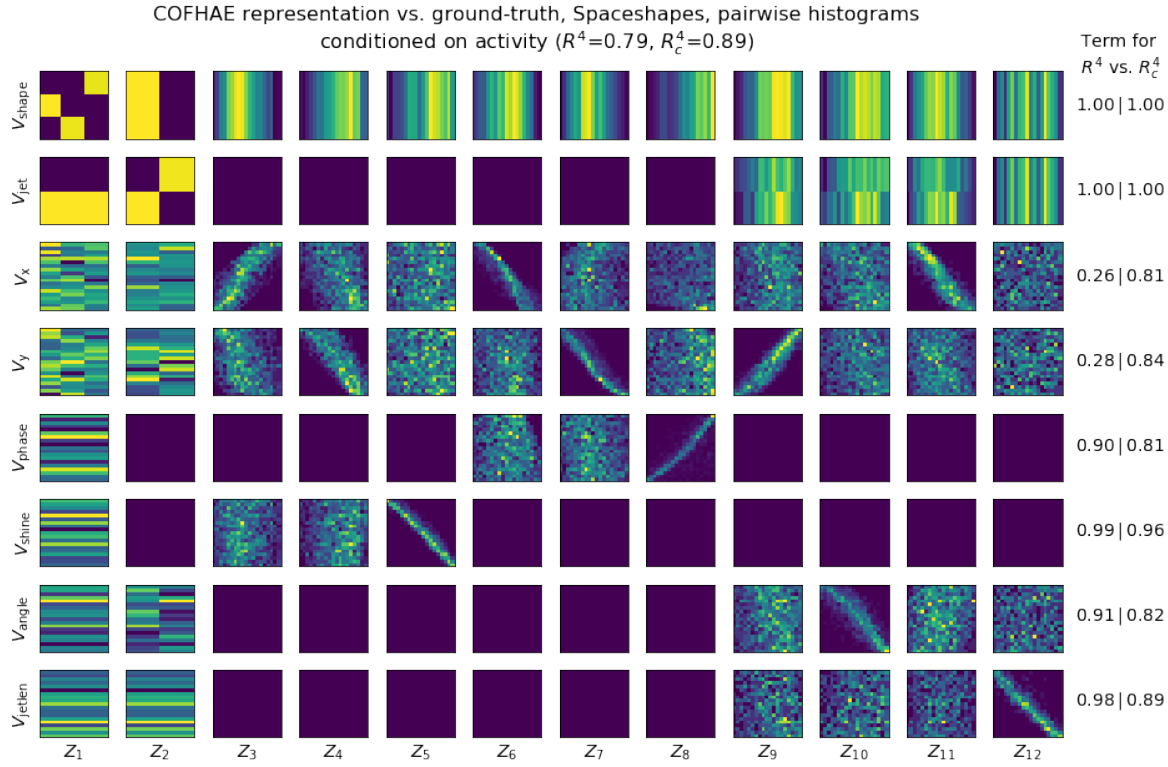
Figure D.5: *Varying disentanglement penalty hyperparameters for baseline algorithms (TCVAE and FactorVAE). In contrast to COFHAE, no setting produces near-optimal disentanglement, even sporadically.*



(a) *AE pairwise histograms and* R^4/R^4_c *scores*



(b) *TCVAE pairwise histograms and* R^4/R^4_c *scores*



(c) *COFHAE pairwise histograms and* R^4/R^4_c *scores*

Figure D.6: *Pairwise histograms of ground-truth vs. learned variables for a flat autoencoder (top left), β-TCVAE (top right), and the best-performing run of COFHAE (bottom) on Spaceshapes. Histograms are conditioned on both variables being active, and dimension-wise components of the* R^4_c *score are shown on the right. β-TCVAE does a markedly better job disentangling certain components than the flat autoencoder, but in this case, COFHAE is able to fully disentangle the ground-truth by modeling the discrete hierarchical structure. See Fig. 8.6 for a latent traversal visualization.*

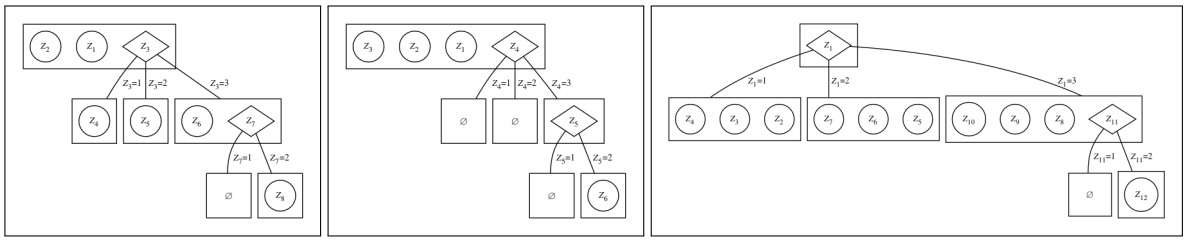


Figure D.7: *Three different potential hierarchies for Spaceshapes which all have the same structure of variable groups and dimensionalities, but with different distributions of continuous variables across groups. The ambiguity in this case is that the continuous variable that modifies each shape (*`phase, shine, angle`*) could either be a child of the corresponding shape category, or be "merged up" and combined into a single top-level continuous variable that controls the shape in different ways based on the category. Alternatively, the location variables* `x` *and* `y` *could instead be "pushed down" from the top level and duplicated across each shape category. In each of these cases, the learned representation still arguably disentangles the ground-truth factors—in the sense that for any fixed categorical assignment, there is still 1:1 correspondence between all learned and ground-truth continuous factors. We deliberately design our R^4_c and H-error metrics in §8.6 to be invariant to these transformations, leaving this specific disambiguation to future work.*

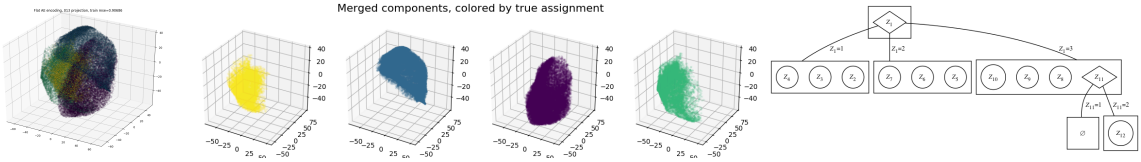


Figure D.8: *MIMOSA-learned initial encoding (left), components (middle), and hierarchy (right) for Spaceshapes. Initial points are in 7 dimensions and projected to 3D for plotting. Three identified components are 3D and one is 4D. Analogue of Fig. 8.2 in the main text.*

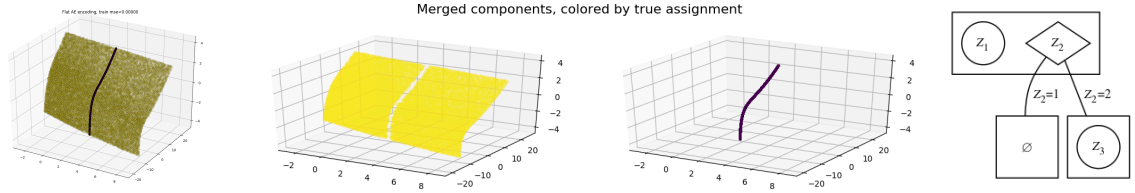


Figure D.9: *MIMOSA-learned initial encoding (left), 2D and 1D components (middle), and hierarchy (right) for depth-2 Chopsticks varying the slope. Analogue of Fig. 8.2 in the main text.*

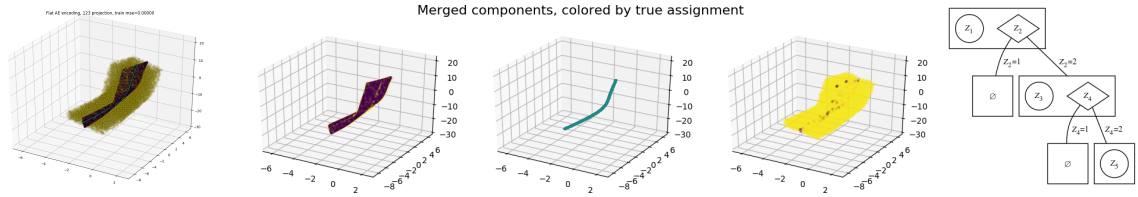


Figure D.10: *MIMOSA-learned initial encoding (left), 2D, 1D, and 3D components (middle), and hierarchy (right) for depth-3 Chopsticks varying the slope. Initial points are in 4 dimensions and projected to 3D for plotting. Analogue of Fig. 8.2 in the main text.*

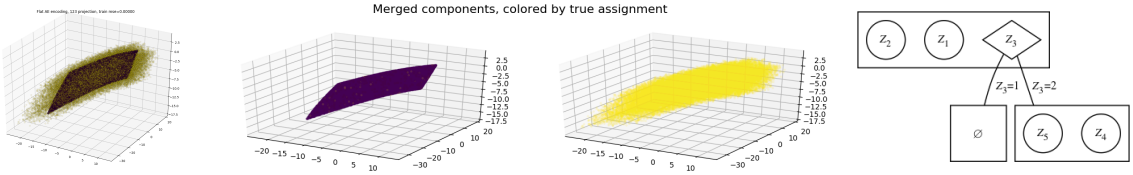


Figure D.11: *MIMOSA-learned initial encoding (left), 2D and 4D components (middle), and hierarchy (right) for depth-2 Chopsticks varying* `both` *slope and intercept. Initial points are in 5 dimensions and projected to 3D for plotting. Analogue of Fig. 8.2 in the main text.*

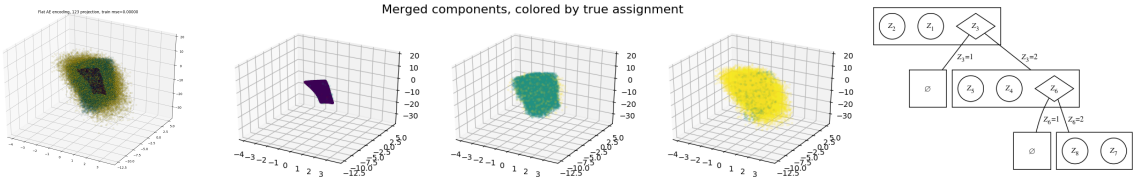


Figure D.12: *MIMOSA-learned initial encoding (left), 2D, 4D, and 6D components (middle), and hierarchy (right) for depth-2 Chopsticks varying* `both` *slope and intercept. Initial points are in 7 dimensions and projected to 3D for plotting. Analogue of Fig. 8.2 in the main text.*

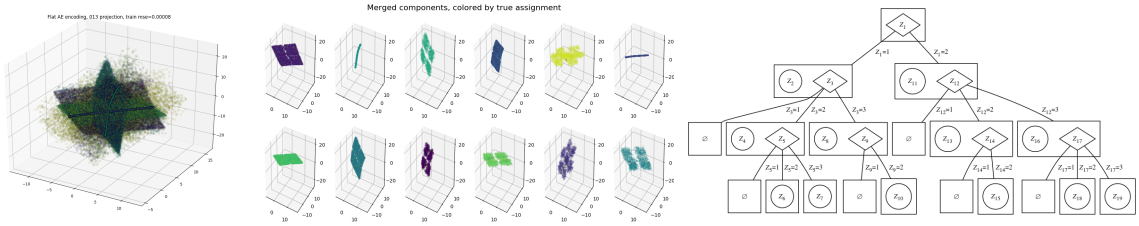
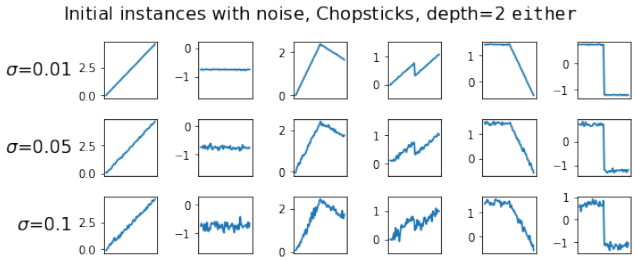
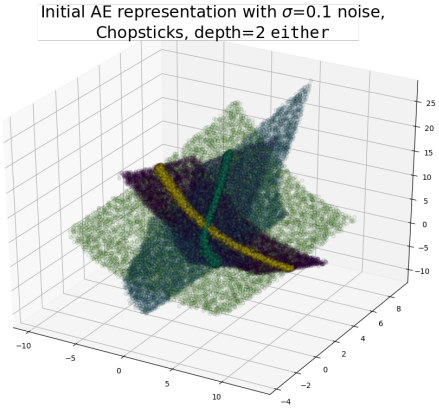


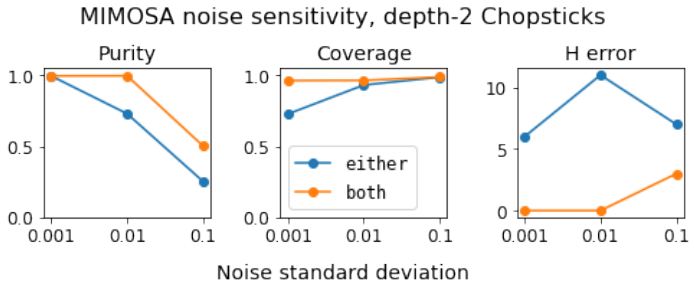
Figure D.13: *MIMOSA-learned initial encoding (left), 1D-3D components (middle), and hierarchy (right) for depth-3 Chopsticks varying either slope or intercept. Note that the learned hierarchy is not quite correct (two nodes at the deepest level are missing). Initial points are in 5 dimensions and projected to 3D. Analogue of Fig. 8.2.*



(a) *Chopsticks Xes corrupted by Gaussian noise.*



(b) *Effect of noise on initial autoencoder Zs.*



(c) *Effect of noise on MIMOSA for two variants.*

Figure D.14: *Illustration of the sensitivity of MIMOSA to data noise. In preliminary experiments, we find that noise poses the greatest problem for identifying the lowest-dimensional components, e.g. the 1D components in (b) that end up being classified as 2D or 3D. Tuning parameters would help, but we lack labels to cross-validate.*

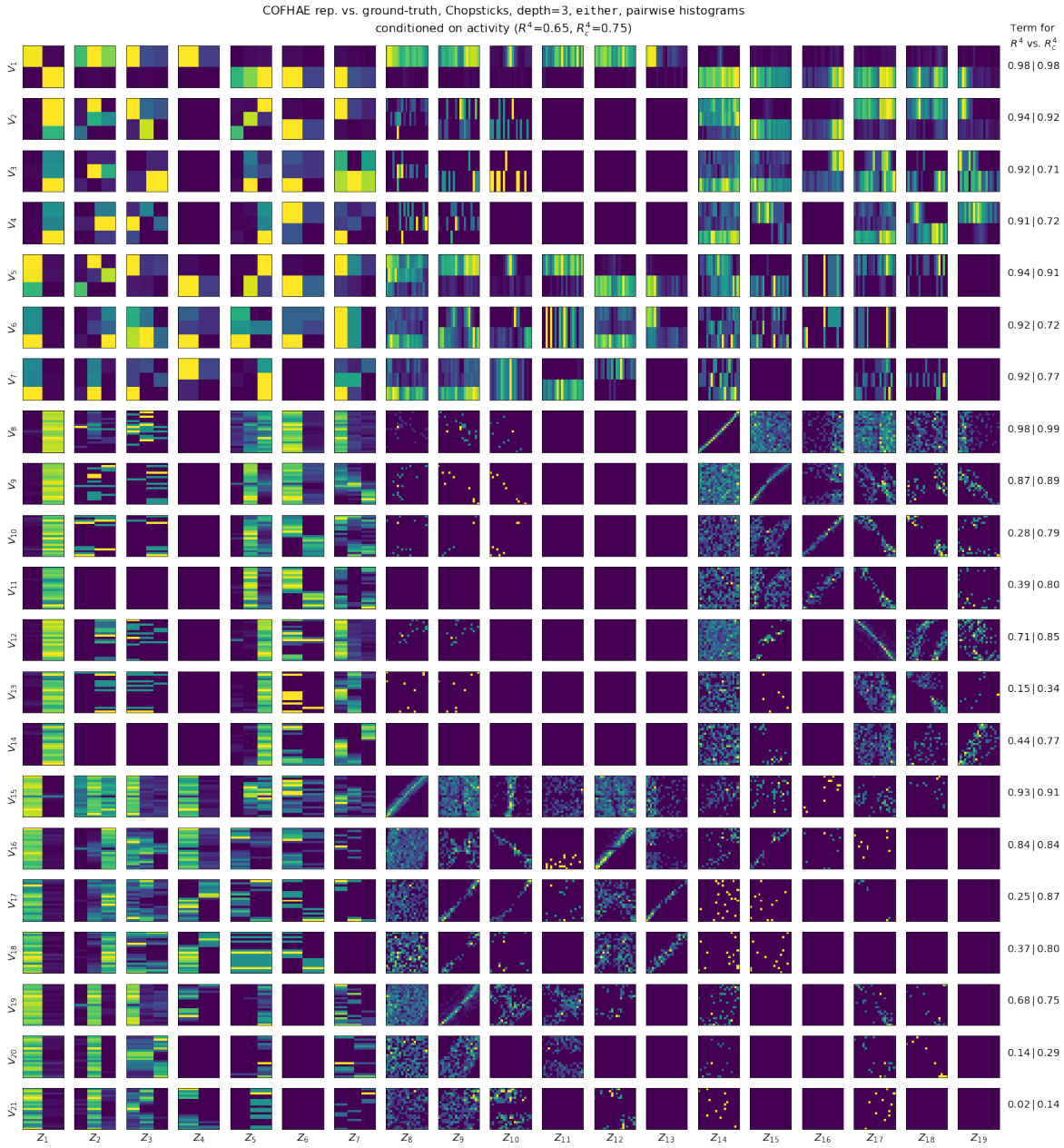


Figure D.15: *Pairwise histograms of ground-truth vs. learned variables for COFHAE on the most complicated hierarchical benchmark (Chopsticks at a recursion depth of 3 varying* `either` *slope or intercept). Histograms are conditioned on both variables being active, and dimension-wise components of the R^4_c score are shown on the right. Despite the depth of the hierarchy, COFHAE representations model it fairly well.*