
Distributional Generalization: A New Kind of Generalization

Preetum Nakkiran*
Harvard University
preetum@cs.harvard.edu

Yamini Bansal*
Harvard University
ybansal@g.harvard.edu

Abstract

We introduce a new notion of generalization—Distributional Generalization—which roughly states that outputs of a classifier at train and test time are close *as distributions*, as opposed to close in just their average error. For example, if we mislabel 30% of dogs as cats in the train set of CIFAR-10, then a ResNet trained to interpolation will in fact mislabel roughly 30% of dogs as cats on the *test set* as well, while leaving other classes unaffected. This behavior is not captured by classical generalization, which would only consider the average error and not the distribution of errors over the input domain. Our formal conjectures, which are much more general than this example, characterize the form of distributional generalization that can be expected in terms of problem parameters: model architecture, training procedure, number of samples, and data distribution. We give empirical evidence for these conjectures across a variety of domains in machine learning, including neural networks, kernel machines, and decision trees. Our results thus advance our empirical understanding of interpolating classifiers.

*Co-first authors. Author contributions in Appendix A.

1 Introduction

We begin with an experiment motivating the need for a notion of generalization beyond test error.

Experiment 1. Consider a binary classification version of CIFAR-10, where CIFAR-10 images x have binary labels *Animal/Object*. Take 50K samples from this distribution as a train set, but apply the following label noise: flip the label of cats to *Object* with probability 30%. Now train a WideResNet f to 0 train error on this train set. How does the trained classifier behave on test samples? Some potential options are:

1. The test error is uniformly small across all CIFAR-10 classes, since there is only 3% overall label noise in the train set.
2. The test error is moderate, and “spread” across all animal classes. After all, the classifier is not explicitly told what a cat or a dog is, just that they are all animals.
3. The test error is localized: the classifier misclassifies roughly 30% of test cats as “objects”, but all other types of animals are largely unaffected.

In fact, reality is closest to option (3), for essentially any good architecture trained to interpolation. Figure 1 shows the results of this experiment with a WideResNet [Zagoruyko and Komodakis, 2016]. The left panel shows the joint density of (x, y) of inputs x and labels $y \in \{\text{Object}/\text{Animal}\}$ on the train set. Since the classifier f is interpolating, this joint distribution is identical to the classifier’s outputs $(x, f(x))$ on the train set. The right panel shows the joint density of $(x, f(x))$ of the classifier’s predictions on *test inputs* x .

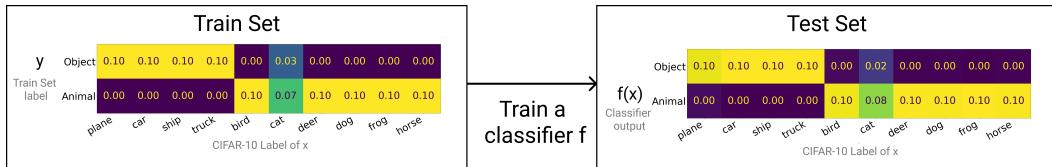


Figure 1: The setup and result of Experiment 1. The CIFAR-10 train set is labeled as either Animals or Objects, with label noise affecting only cats. A WideResNet-28-10 is then trained to 0 train error on this train set, and evaluated on the test set. The joint distribution of $(x, f(x))$ on the train set is close to $(x, f(x))$ on the test set. Full experimental details in Appendix D.2.

This experiment is interesting for several reasons. First, the error is *localized* to cats in test set as it was in the train set, even though no explicit cat labels were provided. Second, the *amount* of error on the cat class is close to the noise applied on the train set. Thus, the behavior of the classifier on the train set *generalizes* to the test set in a certain sense. Third, the trained classifier does *not* behave close to the Bayes optimal classifier for the distribution². However, the classifier does behave close to an optimal *sampler* from the conditional distribution, i.e. $f(x) \sim p(y|x)$.

This behavior would not be captured by solely considering the average test error — it requires reasoning about the entire distribution of classifier outputs. In our work, we show that this experiment is just one instance of a different type of generalization, which we call “Distributional Generalization”. We first describe the mathematical form of this generalization. Then, through extensive experiments, we will show that this type of generalization occurs widely in existing machine learning methods on real datasets, including neural networks, kernel machines and decision trees.

1.1 Distributional Generalization

Supervised learning aims to learn a model that correctly classifies inputs $x \in \mathcal{X}$ from a given distribution \mathcal{D} into classes $y \in \mathcal{Y}$. We want a model with small *test error* on this distribution. In practice, we find such a classifier by minimizing the *train error* of a model on the train set. This procedure is justified when we expect a small generalization gap: the gap between the error on the train and test set. That is, the trained model f should have: $\text{Error}_{\text{TrainSet}}(f) \approx \text{Error}_{\text{TestSet}}(f)$. We

²Nor do we expect it to approach the Bayes optimal one when both data and model size tend to infinity together, as long as the models remain interpolating.

now re-write this classical notion of generalization in a form better suited for our extension.

Classical Generalization: *Let f be a trained classifier. Then f generalizes if:*

$$\mathbb{E}_{\substack{x \sim \text{TrainSet} \\ \hat{y} \leftarrow f(x)}} [\mathbb{1}\{\hat{y} \neq y(x)\}] \approx \mathbb{E}_{\substack{x \sim \text{TestSet} \\ \hat{y} \leftarrow f(x)}} [\mathbb{1}\{\hat{y} \neq y(x)\}] \quad (1)$$

Above, $y(x)$ is the true class of x and \hat{y} is the predicted class. The LHS of Equation 1 is the train error of f , and the RHS is the test error. Crucially, both sides of Equation 1 are expectations of the same function ($T_{\text{err}}(x, \hat{y}) := \mathbb{1}\{\hat{y} \neq y(x)\}$) under different distributions. The LHS of Equation 1 is the expectation of T_{err} under the “Train Distribution” \mathcal{D}_{tr} , which is the distribution over (x, \hat{y}) given by sampling a train point x along with its classifier-label $f(x)$. Similarly, the RHS is under the “Test Distribution” \mathcal{D}_{te} , which is this same construction over the test set. These two distributions are the central objects in our study, and are defined formally in Section 3.2. We can now introduce Distributional Generalization, which is a property of trained classifiers. It is parameterized by a set of bounded functions (“tests”): $\mathcal{T} \subseteq \{T : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]\}$.

Distributional Generalization: *Let f be a trained classifier. Then f satisfies Distributional Generalization with respect to tests \mathcal{T} if:*

$$\forall T \in \mathcal{T} : \mathbb{E}_{\substack{x \sim \text{TrainSet} \\ \hat{y} \leftarrow f(x)}} [T(x, \hat{y})] \approx \mathbb{E}_{\substack{x \sim \text{TestSet} \\ \hat{y} \leftarrow f(x)}} [T(x, \hat{y})] \quad (2)$$

We write this property as $\mathcal{D}_{\text{tr}} \approx^{\mathcal{T}} \mathcal{D}_{\text{te}}$. This states that the train and test distribution have similar expectations for all functions in the family \mathcal{T} . For the singleton set $\mathcal{T} = \{T_{\text{err}}\}$, this is equivalent to classical generalization, but it may hold for much larger sets \mathcal{T} . For example in Experiment 1, the train and test distributions match with respect to the test function “*Fraction of true cats labeled as object*.” In fact, we find that the family \mathcal{T} is so large in practice that it is best to think of Distributional Generalization as stating that the distributions \mathcal{D}_{tr} and \mathcal{D}_{te} are close *as distributions*.

This property becomes especially interesting for interpolating classifiers, which fit their train sets exactly. Here, the Train Distribution $(x_i, f(x_i))$ is exactly equal³ to the original distribution $(x, y) \sim \mathcal{D}$, since $f(x_i) = y_i$ on the train set. In this case, distributional generalization claims that the output distribution $(x, f(x))$ of the model on test samples is close to the *true* distribution (x, y) . The following conjecture specializes Distributional Generalization to interpolating classifiers, and will be the main focus of our work.

Interpolating Indistinguishability Meta-Conjecture (informal): *For interpolating classifiers f , and a large family \mathcal{T} of test functions, the distributions:*

$$(x, f(x))_{x \in \text{TestSet}} \approx^{\mathcal{T}} (x, f(x))_{x \in \text{TrainSet}} \equiv (x, y)_{x, y \sim \mathcal{D}} \quad (3)$$

This is a “meta-conjecture”, which becomes a concrete conjecture once the family of tests \mathcal{T} is specified. One of the main contributions of our work is formally stating two concrete instances of this conjecture— specifying exactly the family of tests \mathcal{T} and their dependence on problem parameters (the distribution, model family, training procedure, etc). It captures behaviors far more general than Experiment 1, and empirically applies across a variety of natural settings in machine learning.

1.2 Summary of Contributions

We extend the classical framework of generalization by introducing Distributional Generalization, in which the train and test behavior of models are close *as distributions*. Informally, for trained classifiers f , its outputs on the train set $(x, f(x))_{x \in \text{TrainSet}}$ are close in distribution to its outputs on the test set $(x, f(x))_{x \in \text{TestSet}}$, where the form of this closeness depends on specifics of the model, training procedure, and distribution. This notion is more fine-grained than classical generalization, since it considers the entire distribution of model outputs instead of just the test error.

We initiate the study of Distributional Generalization across various domains in machine learning. For interpolating classifiers, we state two formal conjectures which predict the form of distributional closeness that can be expected for a given model and task:

³The formal definition of Train Distribution, in Section 3.2, includes the randomness of sampling the train set as well. We consider a fixed train set in the Introduction for sake of exposition.

1. **Feature Calibration Conjecture** (Section 4): Interpolating classifiers, when trained on samples from a distribution, will match this distribution up to all “distinguishable features” (Definition 1).
2. **Agreement Conjecture** (Section 5): For two interpolating classifiers of the same type, trained independently on the same distribution, their *agreement probability* with each other on test samples roughly matches their *test accuracy*.

We perform a number of experiments surrounding these conjectures, which reveal new behaviors of standard interpolating classifiers (e.g. ResNets, MLPs, kernels, decision trees). We prove our conjectures for 1-Nearest-Neighbors (Theorem 1), which suggests some form of “locality” as the underlying mechanism. Finally, we discuss extending these results to non-interpolating methods in Section 7. Our experiments and conjectures shed new light on the structure of interpolating classifiers, which are extensively studied in recent years yet still poorly understood.

2 Related Work

Our work is inspired by the broader study of interpolating and overparameterized methods in machine learning; a partial list of works in this theme includes [Advani and Saxe \[2017\]](#), [Allen-Zhu et al. \[2019\]](#), [Arora et al. \[2019\]](#), [Bartlett et al. \[2020\]](#), [Belkin et al. \[2018a,b, 2019\]](#), [Breiman \[1995\]](#), [Chizat and Bach \[2020\]](#), [Dziugaite and Roy \[2017\]](#), [Geiger et al. \[2019\]](#), [Gerace et al. \[2020\]](#), [Ghorbani et al. \[2019\]](#), [Goldt et al. \[2019\]](#), [Hastie et al. \[2019\]](#), [Ji and Telgarsky \[2019\]](#), [Liang and Rakhlin \[2018\]](#), [Mei and Montanari \[2019\]](#), [Muthukumar et al. \[2020\]](#), [Nakkiran et al. \[2020\]](#), [Neal et al. \[2018\]](#), [Neyshabur et al. \[2018\]](#), [Schapire et al. \[1998\]](#), [Soudry et al. \[2018\]](#), [Zhang et al. \[2016\]](#).

Interpolating Methods. Many of the best-performing techniques on high-dimensional tasks are interpolating methods, which fit their train samples to 0 train error. This includes neural networks and kernels on images [[He et al., 2016](#), [Shankar et al., 2020](#)], and random forests on tabular data [[Fernández-Delgado et al., 2014](#)]. Interpolating methods have been extensively studied both recently and in the past, since we do not theoretically understand their practical success [[Belkin et al., 2018a,b, 2019](#), [Breiman, 1995](#), [Hastie et al., 2019](#), [Liang and Rakhlin, 2018](#), [Mei and Montanari, 2019](#), [Nakkiran et al., 2020](#), [Schapire, 1999](#), [Schapire et al., 1998](#), [Zhang et al., 2016](#)]. In particular, much of the classical work in statistical learning theory (uniform convergence, VC-dimension, Rademacher complexity, regularization, stability) fails to explain the success of interpolating methods [[Belkin et al., 2018a,b](#), [Nagarajan and Kolter, 2019](#), [Zhang et al., 2016](#)]. The few techniques which do apply to interpolating methods (e.g. margin theory [[Schapire et al., 1998](#)]) remain vacuous on modern neural networks and kernels.

Learning Substructures. The observation that powerful networks can pick up on finer aspects of the distribution than their training labels reveal also exists in various forms in the literature. For example, [Gilboa and Gur-Ari \[2019\]](#) found that standard CIFAR-10 networks tend to cluster left-facing and right-facing horses separately in activation space, when visualized via activation atlases [[Carter et al., 2019](#)]. Such fine-structural aspects of distributions can also be seen at the level of individual neurons (e.g. [Bau et al. \[2020\]](#), [Cammarata et al. \[2020\]](#), [Olah et al. \[2018\]](#), [Radford et al. \[2017\]](#), [Zhou et al. \[2014\]](#)).

Decision Trees. In a similar vein to our work, [Olson and Wyner \[2018\]](#), [Wyner et al. \[2017\]](#) investigate decision trees, and show that random forests are equivalent to a Nadaraya–Watson smoother [Nadaraya \[1964\]](#), [Watson \[1964\]](#) with a certain smoothing kernel. Decision trees [[Breiman et al., 1984](#)] are often intuitively thought of as “adaptive nearest-neighbors,” since they are explicitly a spatial-partitioning method [[Hastie et al., 2009](#)]. Thus, it may not be surprising that decision trees behave similarly to 1-Nearest-Neighbors. [Olson and Wyner \[2018\]](#), [Wyner et al. \[2017\]](#) took steps towards characterizing and understanding this behavior – in particular, [Olson and Wyner \[2018\]](#) defines an equivalent smoothing kernel corresponding to a random forest, and empirically investigates the quality of the conditional density estimate. Our work presents a formal characterization of the quality of this conditional density estimate (Conjecture 1), which is a novel characterization even for decision trees, as far as we know.

Kernel Smoothing. The term kernel regression is sometimes used in the literature to refer to kernel *smoothers*, such as the Nadaraya–Watson kernel smoother [[Nadaraya, 1964](#), [Watson, 1964](#)]. But in

this work we use the term “kernel regression” to refer only to regression in a Reproducing Kernel Hilbert Space, as described in the experimental details.

Label Noise. Our conjectures also describe the behavior of neural networks under label noise, which has been empirically and theoretically studied in the past, though not formally characterized before [Belkin et al., 2018b, Chatterji and Long, 2020, Natarajan et al., 2013, Rolnick et al., 2017, Thulasidasan et al., 2019, Zhang et al., 2016, Ziyin et al., 2020]. Prior works have noticed that vanilla interpolating networks are sensitive to label noise (e.g. Figure 1 in Zhang et al. [2016], and Belkin et al. [2018b]), and there are many works on making networks more robust to label noise via modifications to the training procedure or objective [Natarajan et al., 2013, Rolnick et al., 2017, Thulasidasan et al., 2019, Ziyin et al., 2020]. In contrast, we claim this sensitivity to label noise is not necessarily a problem to be fixed, but rather a consequence of a stronger property: distributional generalization.

Conditional Density Estimation. Our density calibration property is similar to the guarantees of a conditional density estimator. More specifically, Conjecture 1 states that an interpolating classifier *samples* from a distribution approximating the conditional density of $p(y|x)$ in a certain sense. Conditional density estimation has been well-studied in classical nonparametric statistics (e.g. the Nadaraya–Watson kernel smoother [Nadaraya, 1964, Watson, 1964]). However, these classical methods behave poorly in high-dimensions, both in theory and in practice. There are some attempts to extend these classical methods to modern high-dimensional problems via augmenting estimators with neural networks (e.g. Rothfuss et al. [2019]). Random forests have also been known to exhibit properties similar to conditional density estimators. This has been formalized in various ways, often only with asymptotic guarantees [Athey et al., 2019, Meinshausen, 2006, Pospisil and Lee, 2018].

No prior work that we are aware of attempts to characterize the quality of the resulting density estimate via testable assumptions, as we do with our formulation of Conjecture 1. Finally, our motivation is not to design good conditional density estimators, but rather to study properties of interpolating classifiers — which we find happen to share properties of density estimators.

Uncertainty and Calibration. The Agreement Property (Conjecture 2) bears some resemblance to uncertainty estimation (e.g. Lakshminarayanan et al. [2017]), since it estimates the the test error of a classifier using an ensemble of 2 models trained on disjoint train sets. However, there are important caveats: (1) Our Agreement Property only holds on-distribution, and degrades on off-distribution inputs. Thus, it is not as helpful to estimate out-of-distribution errors. (2) It only gives an estimate of the average test error, and does not imply pointwise calibration estimates for each sample.

Feature Calibration (Conjecture 1) is also related to the concepts of calibration and multicalibration [Guo et al., 2017, Hébert-Johnson et al., 2018, Niculescu-Mizil and Caruana, 2005]. In our framework, calibration is implied by Feature Calibration for a specific set of partitions L (determined by level sets of the classifier’s confidence). However, we are not concerned with a specific set of partitions (or “subgroups” in the algorithmic fairness literature) but we generally aim to characterize for which partitions Feature Calibration holds. Moreover, we consider only hard-classification decisions and not confidences, and we study only standard learning algorithms which are not given any distinguished set of subgroups/partitions in advance. Our notion of distributional generalization is also related to the notion of “distributional subgroup overfitting” introduced recently by Yaghini et al. [2019] to study algorithmic fairness. This can be seen as studying distributional generalization for a specific family of tests (determined by distinguished subgroups in the population).

Locality and Manifold Learning. Our intuition for the behaviors in this work is that they arise due to some form of “locality” of the trained classifiers, in an appropriate space. This intuition is present in various forms in the literature, for example: the so-called called “manifold hypothesis,” that natural data lie on a low-dimensional manifold (e.g. Narayanan and Mitter [2010], Sharma and Kaplan [2020]), as well as works on local stiffness of the loss landscape [Fort et al., 2019b], and works showing that overparameterized neural networks can learn hidden low-dimensional structure in high-dimensional settings [Bach, 2017, Chizat and Bach, 2020, Gerace et al., 2020]. It is open to more formally understand connections between our work and the above.

3 Preliminaries

Notation. We consider joint distributions \mathcal{D} on $x \in \mathcal{X}$ and discrete $y \in \mathcal{Y} = [k]$. Let \mathcal{D}^n denote n iid samples from \mathcal{D} and $S = \{(x_i, y_i)\}$ denote a train set. Let \mathcal{F} denote the training procedure of a classifier family (including architecture and training algorithm), and let $f \leftarrow \text{Train}_{\mathcal{F}}(S)$ denote training a classifier f on train set S . We consider classifiers which output hard decisions $f : \mathcal{X} \rightarrow \mathcal{Y}$. Let $\text{NN}_S(x) = x_i$ denote the nearest neighbor to x in train-set S , with respect to a distance metric d . Our theorems will apply to any distance metric, and so we leave this unspecified. Let $\text{NN}_S^{(y)}(x)$ denote the nearest neighbor estimator itself, that is, $\text{NN}_S^{(y)}(x) := y_i$ where $x_i = \text{NN}_S(x)$.

Experimental Setup. Full experimental details are provided in Appendix C. Briefly, we train all classifiers to interpolation unless otherwise specified—that is, to 0 train error. We use standard-practice training techniques for all methods with minor hyperparameter modifications for training to interpolation. In all experiments, we consider only the hard-classification decisions, and not e.g. the softmax probabilities. Neural networks (MLPs and ResNets [He et al., 2016]) are trained with Stochastic Gradient Descent. Interpolating decision trees are trained using the growth rule from Random Forests [Breiman, 2001], growing until all leafs have a single sample. For kernel classification, we consider both kernel regression on one-hot labels and kernel SVM, with small or 0 values of regularization (which is often optimal, as in Shankar et al. [2020]). Section 7 considers non-interpolating versions of the above methods (via early-stopping or regularization).

3.1 Distributional Closeness

For two distributions P, Q over $\mathcal{X} \times \mathcal{Y}$, let $P \approx_{\varepsilon} Q$ denote ε -closeness in total variation distance; that is, $\text{TV}(P, Q) = \frac{1}{2}\|P - Q\|_1 \leq \varepsilon$. Recall that TV-distance has an equivalent variational characterization: For distributions P, Q over $\mathcal{X} \times \mathcal{Y}$, we have

$$\text{TV}(P, Q) = \sup_{T: \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]} \left| \mathbb{E}_{(x, y) \sim P} [T(x, y)] - \mathbb{E}_{(x, y) \sim Q} [T(x, y)] \right|$$

A “test” (or “distinguisher”) here is a function $T : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]$ which accepts a sample from either distribution, and is intended to classify the sample as either from distribution P or Q . TV distance is then the advantage of the best distinguisher among all bounded tests. More generally, for any family $\mathcal{T} \subseteq \{T : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]\}$ of tests, we say distributions P and Q are “ ε -indistinguishable up to \mathcal{T} -tests” if they are close with respect to all tests in class \mathcal{T} . That is,

$$P \approx_{\varepsilon}^{\mathcal{T}} Q \iff \sup_{T \in \mathcal{T}} \left| \mathbb{E}_{(x, y) \sim P} [T(x, y)] - \mathbb{E}_{(x, y) \sim Q} [T(x, y)] \right| \leq \varepsilon \quad (4)$$

This notion of closeness is also known as an Integral Probability Metric [Müller, 1997]. Throughout this work, we will define specific families of distinguishers \mathcal{T} to characterize the sense in which the output distribution $(x, f(x))$ of classifiers is close to their input distribution $(x, y) \sim \mathcal{D}$. When we write $P \approx Q$, we are making an informal claim in which we mean $P \approx_{\varepsilon} Q$ for some small but unspecified ε .

3.2 Framework for Indistinguishability

Here we setup the formal objects studied in the remainder of the paper. This formal description of Train and Test distributions differs slightly from the informal description in the Introduction, because we want to study the generalization properties of an entire end-to-end training procedure ($\text{Train}_{\mathcal{F}}$), and not just properties of a fixed classifier (f). We thus consider the following three distributions over $\mathcal{X} \times \mathcal{Y}$.

| | | |
|---|--|---|
| Source \mathcal{D}: (x, y) where $x, y \sim \mathcal{D}$ | Train \mathcal{D}_{tr}: $(x_{\text{tr}}, f(x_{\text{tr}}))$ $S \sim \mathcal{D}^n, f \leftarrow \text{Train}_{\mathcal{F}}(S),$ $x_{\text{tr}}, y_{\text{tr}} \sim S$ | Test \mathcal{D}_{te}: $(x, f(x))$ $S \sim \mathcal{D}^n, f \leftarrow \text{Train}_{\mathcal{F}}(S),$ $x, y \sim \mathcal{D}$ |
|---|--|---|

The **Source Distribution** \mathcal{D} is simply the original distribution. To sample from the **Train Distribution** \mathcal{D}_{tr} , we first sample a train set $S \sim \mathcal{D}^n$, train a classifier f on it, then output $(x_{\text{tr}}, f(x_{\text{tr}}))$ for a random *train point* x_{tr} . That is, \mathcal{D}_{tr} is the distribution of input and outputs of a trained classifier f .

on its train set. To sample from the **Test Distribution** \mathcal{D}_{te} , do we this same procedure, but output $(x, f(x))$ for a random *test point* x . That is, the \mathcal{D}_{te} is the distribution of input and outputs of a trained classifier f at test time. The only difference between the Train Distribution and Test Distribution is that the point x is sampled from the train set or the test set, respectively.⁴

For interpolating classifiers, $f(x_{\text{tr}}) = y_{\text{tr}}$ on the train set, and so the Source and Train distributions are equivalent:

$$\text{For interpolating classifiers } f: \mathcal{D} \equiv \mathcal{D}_{\text{tr}} \quad (5)$$

Our general thesis is that the Train and Test Distributions are indistinguishable under a variety of test families \mathcal{T} . Formally, we argue that for certain families of tests \mathcal{T} and interpolating classifiers \mathcal{F} ,

$$\text{Indistinguishability Conjecture: } \boxed{\mathcal{D} \equiv \mathcal{D}_{\text{tr}} \approx_{\varepsilon}^{\mathcal{T}} \mathcal{D}_{\text{te}}} \quad (6)$$

Sections 4 and 5 give specific families of tests \mathcal{T} for which these distributions are indistinguishable. The quality of this distributional closeness will depend on details of the classifier family and distribution, in ways which we will specify.

4 Feature Calibration

The distributional closeness of Experiment 1 is subtle, and depends on the classifier architecture, distribution, and training method. For example, Experiment 1 does not hold if we use a fully-connected network (MLP) instead of a ResNet, or if we early-stop the ResNet instead of training to interpolation (see Appendix D.2). Both these scenarios fail in different ways: An MLP cannot properly distinguish cats from dogs even when trained on real CIFAR-10 labels, and so (informally) it has no hope of behaving differently on cats in the setting of Experiment 1. On the other hand, an early-stopped ResNet for Experiment 1 does not label 30% of cats as objects on the *train set*, since it does not interpolate, and thus has no hope of reproducing this behavior on the test set.

We now characterize these behaviors, and their dependency on problem parameters, via a formal conjecture. This conjecture characterizes a family of tests \mathcal{T} for which the output distribution of a classifier $(x, f(x)) \sim \mathcal{D}_{\text{te}}$ is “close” to the source distribution $(x, y) \sim \mathcal{D}$. At a high level, we argue that the distributions \mathcal{D}_{te} and \mathcal{D} are statistically close if we first “coarsen” the domain of x by some labelling $L : \mathcal{X} \rightarrow [M]$. That is, for certain partitions L , the following distributions are statistically close:

$$(L(x), f(x)) \approx_{\varepsilon} (L(x), y)$$

We first explain this conjecture (“Feature Calibration”) via a toy example, and then we state the conjecture formally in Section 4.2.

4.1 Toy Example

Consider a distribution on points $x \in \mathbb{R}^2$ and binary labels y , as visualized in Figure 2A. This distribution consists of four clusters {Truck, Ship, Cat, Dog} which are labeled either Object or Animal, depicted in red and blue respectively. One of these clusters — the Cat cluster — is mislabeled as class Object with probability 30%. Now suppose we have an interpolating classifier f for this distribution, obtained in some way, and we wish to quantify the closeness between distributions $(x, y) \approx (x, f(x))$ on the test set. Figure 2A shows test points x along with their test labels y — these are samples from the source distribution \mathcal{D} . Figure 2B shows these same test points, but labeled according to $f(x)$ — these are samples from the test distribution \mathcal{D}_{te} . The shaded red/blue regions in Figure 2B shows the decision boundary of the classifier f .

These two distributions do not match exactly — there are some test points where the true label y and classifier output $f(x)$ disagree. However, if we “coarsen” the domain into the four clusters {Truck, Ship, Cat, Dog}, then the marginal distribution of labels within each cluster matches

⁴Technically, these definitions require training a fresh classifier for each sample, using independent train sets. We use this definition because we believe it is natural, although for practical reasons most of our experiments train a single classifier f and evaluate it on the entire train/test set.

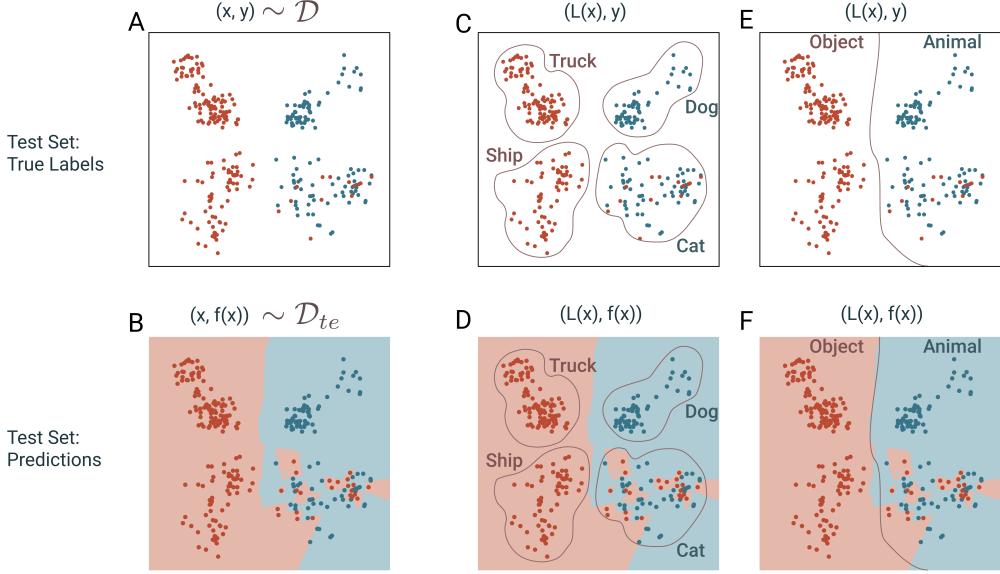


Figure 2: **Toy Example: Feature Calibration.** Schematic of the distributions discussed in Section 4.1, showing a toy example of the Feature Calibration conjecture for several distinguishable features L .

between the classifier outputs $f(x)$ and true labels y . In particular, the fraction of Cat points that are labeled Object is similar between Figures 2C and 2D. This is equivalent to saying that the joint distributions $(L(x), y)$ and $(L(x), f(x))$ are statistically close, for the partition $L : \mathcal{X} \rightarrow \{\text{Truck, Ship, Cat, Dog}\}$. That is, if we can only see points x through their cluster-label $L(x)$, then the distributions (x, y) and $(x, f(x))$ will appear close. These two “coarsened” joint distributions are also what we plotted in Figure 1 from the Introduction, where we considered the partition $L(x) := \text{CIFAR_Class}(x)$, the CIFAR-10 class of x .

There may be many such partitions L for which the above distributional closeness holds. For example, the coarser partition $L_2 : \mathcal{X} \rightarrow \{\text{Animal, Object}\}$ also works in our example, as shown in Figures 2E and 2F. However, clearly not all partitions L will satisfy closeness, since the distributions themselves are not statistically close. For a finer partition which splits the clusters into smaller pieces (e.g. based on the age of the dog), the distributions may not match unless we use very powerful classifiers or many train samples. Intuitively, allowable partitions are those which can be learnt from samples. To formalize the set of allowable partitions L , we define a *distinguishable feature*: a partition of the domain \mathcal{X} that is learnable for a given family of models. For example, in Experiment 1, the partition into CIFAR-10 classes would be a distinguishable feature for ResNets, but not for MLPs. We now state the formal definition of a distinguishable partition and the formal conjecture.

4.2 Formal Definitions

We first define a *distinguishable feature*: a labeling of the domain \mathcal{X} that is learnable for a given family of models. This definition depends on the family of models \mathcal{F} , the distribution \mathcal{D} , and the number of train samples n .

Definition 1 ($(\varepsilon, \mathcal{F}, \mathcal{D}, n)$ -Distinguishable Feature). *For a distribution \mathcal{D} over $\mathcal{X} \times \mathcal{Y}$, number of samples n , family of models \mathcal{F} , and small $\varepsilon \geq 0$, an $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$ -distinguishable feature is a partition $L : \mathcal{X} \rightarrow [M]$ of the domain \mathcal{X} into M parts, such that training a model from \mathcal{F} on n samples labeled by L works to classify L with high test accuracy.*

Precisely, L is a distinguishable feature if the following procedure succeeds with probability at least $1 - \varepsilon$:

1. Sample a train set $S \leftarrow \{(x_i, L(x_i))\}$ of n samples $(x_i, y_i) \sim \mathcal{D}$, labeled by the partition L .

2. Train a classifier $f \leftarrow \text{Train}_{\mathcal{F}}(S)$.
3. Sample a test point $x \sim \mathcal{D}$, and check that f correctly classifies its partition: Output success iff $f(x) = L(x)$.

That is, L is a $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$ -distinguishable feature if:

$$\Pr_{\substack{S = \{(x_i, L(x_i))\}_{x_1, \dots, x_n \sim \mathcal{D}} \\ f \leftarrow \text{Train}_{\mathcal{F}}(S) \\ x \sim \mathcal{D}}} [f(x) = L(x)] \geq 1 - \varepsilon$$

To recap, this definition is meant to capture a labeling of the domain \mathcal{X} that is learnable for a given family of models and training procedure. Note that this definition only depends on the marginal distribution of \mathcal{D} on x , and does not depend on the label distribution $p_{\mathcal{D}}(y|x)$. The definition of distinguishable feature must depend on the classifier family \mathcal{F} and number of samples n , since a more powerful classifier can distinguish more features. Note that there could be many distinguishable features for a given setting $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$ — including features not implied by the class label, such as the presence of grass in a CIFAR-10 image.

Our main conjecture in this section is that the test distribution $(x, f(x)) \sim \mathcal{D}_{\text{te}}$ is statistically close to the source distribution $(x, y) \sim \mathcal{D}$ when the domain is “coarsened” by a distinguishable feature. That is, the distributions $(L(x), f(x))$ and $(L(x), y)$ are *statistically close* for all distinguishable features L . Formally:

Conjecture 1 (Feature Calibration). *For all natural distributions \mathcal{D} , number of samples n , family of interpolating models \mathcal{F} , and $\varepsilon \geq 0$, the following distributions are statistically close for all $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$ -distinguishable features L :*

$$(L(x), f(x)) \underset{\substack{f \leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}^n) \\ x, y \sim \mathcal{D}}}{\approx_{\varepsilon}} (L(x), y) \quad (7)$$

Notably, this holds for all distinguishable features L , and it holds “automatically” — we simply train a classifier, without specifying any particular partition. The statistical closeness predicted is within ε , which is determined by the ε -distinguishability of L (we usually think of ε as small). As a trivial instance of the conjecture, suppose we have a distribution with deterministic labels, and consider the ε -distinguishable feature $L(x) := y(x)$, i.e. the label itself. The ε here is then simply the test error of f , and Conjecture 1 is true by definition. The formal statements of Definition 1 and Conjecture 1 may seem somewhat arbitrary, involving many quantifiers over $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$. However, we believe these statements are natural. To support this, in Section 4.5 we prove that Conjecture 1 is formally true as stated for 1-Nearest-Neighbor classifiers.

Connection to Indistinguishability. Conjecture 1 can be equivalently phrased as an instantiation of our general Indistinguishability Conjecture: the source distribution \mathcal{D} and test distribution \mathcal{D}_{te} are “indistinguishable up to L -tests”. That is, Conjecture 1 is equivalent to the statement

$$\mathcal{D}_{\text{te}} \approx_{\varepsilon}^{\mathcal{L}} \mathcal{D} \quad (8)$$

where \mathcal{L} is the family of all tests which depend on x only via a distinguishable feature L . That is, $\mathcal{L} := \{(x, y) \mapsto T(L(x), y) : (\varepsilon, \mathcal{F}, \mathcal{D}, n)$ -distinguishable feature L and $T : [M] \times \mathcal{Y} \rightarrow [0, 1]\}$. In other words, \mathcal{D}_{te} is indistinguishable from \mathcal{D} to any distinguisher that only sees the input x via a distinguishable feature $L(x)$.

4.3 Experiments

We now empirically validate our conjecture in a variety of settings in machine learning, including neural networks, kernel machines, and decision trees. To do so, we begin by considering the simplest possible distinguishable feature, and progressively consider more complex ones. Each of the experimental settings below highlights a different aspect of interpolating classifiers, which may be of independent theoretical or practical interest. We summarize the experiments here; detailed descriptions are provided in Appendix D.

Constant Partition: Consider the trivially-distinguishable *constant* feature $L(x) = 0$. Then, Conjecture 1 states that the marginal distribution of class labels for any interpolating classifier $f(x)$ is

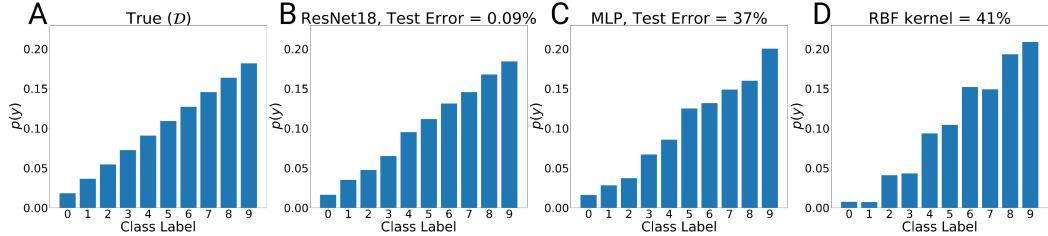


Figure 3: **Feature Calibration for Constant Partition L :** The CIFAR-10 train and test sets are class rebalanced according to (A). Interpolating classifiers are trained on the train set, and we plot the class balance of their outputs on the test set. This roughly matches the class balance of the train set, even for poorly-generalizing classifiers.

close to the true marginals $p(y)$. That is, irrespective of the classifier’s test accuracy, it outputs the “right” proportion of class labels on the test set, even when there is strong class imbalance.

To show this, we construct a dataset based on CIFAR-10 that has class imbalance. For class $k \in \{0\dots9\}$, sample $(k+1) \times 500$ images from that class. This will give us a dataset where classes will have marginal distribution $p(y=\ell) \propto \ell + 1$ for classes $\ell \in [10]$, as shown in Figure 3. We do this both for the training set and the test set, to keep the distribution \mathcal{D} fixed. We then train a variety of classifiers (MLPs, Kernels, ResNets) to interpolation on this dataset, which have varying levels of test errors (9-41%). The class balance of classifier outputs on the (rebalanced) test set is then close to the class balance on the train set, even for poorly generalizing classifiers. Full experimental details and results are described in Appendix D. Note that a 1-nearest neighbors classifier would have this property.

Class Partition: We now consider settings (datasets and models) where the original class labels are a distinguishable feature. For instance, the CIFAR-10 classes are distinguishable by ResNets, and MNIST classes are distinguishable by the RBF kernel. Since the conjecture holds for any arbitrary label distribution $p(y|x)$, we consider many such label distributions and show that, for instance, the joint distributions $(\text{Class}(x), y)$ and $(\text{Class}(x), f(x))$ are close. This includes the setting of Experiments 1 and 2 from the Introduction.

In Figure 4, we mislabel class 0 \rightarrow 1 with probability p in the CIFAR-10 train set. This gives us the joint distribution shown in Figure 4A. We then train a WideResNet-28-10 on this noisy distribution. Figure 4B shows the joint distribution on the test set. Figure 4C shows the $(1, 0)$ entry of this matrix as we vary $p \in [0, 1]$. The Bayes optimal classifier for this distribution would behave as a step function (shown in red), and a classifier that obeys Conjecture 1 exactly would follow the diagonal (in green). The actual experiment (in blue) is close to the behavior predicted by Conjecture 1.

In fact, our conjecture holds even for a joint density determined by a random confusion matrix on CIFAR-10. In Figure 5, we first generate a random sparse confusion matrix on 10 classes, such that each class is preserved with probability 50% and flipped to one of two other classes with probability 20% and 30% respectively. We then apply label noise with this confusion matrix to the train set, and measure the confusion matrix of the trained classifier on the test set. As expected, the train and test confusion matrices are close, and share the same sparsity pattern.

Figure 6 shows a version of this experiment for decision trees on the molecular biology UCI task. The molecular biology task is a 3-way classification problem: to classify the type of a DNA splice junction (donor, acceptor, or neither), given the sequence of DNA (60 bases) surrounding the junction. We add varying amounts of label noise that flips class 2 to class 1 with a certain probability, and we observe that interpolating decision trees reproduce this same structured label noise on the test set. We also demonstrate similar experiments with the Gaussian kernel on MNIST (Figure 10), and several other UCI tasks (Appendix D).

Multiple features: We now consider a setting where we may have many distinguishable features for a single classification task. The conjecture states that the network should be automatically calibrated for all distinguishable features, even when it is not explicitly provided any information about these features. For this, we use the CelebA dataset [Liu et al., 2015], which contains images of celebrities with various labelled binary attributes per-image (“male”, “blond hair”, etc). Some of these attributes

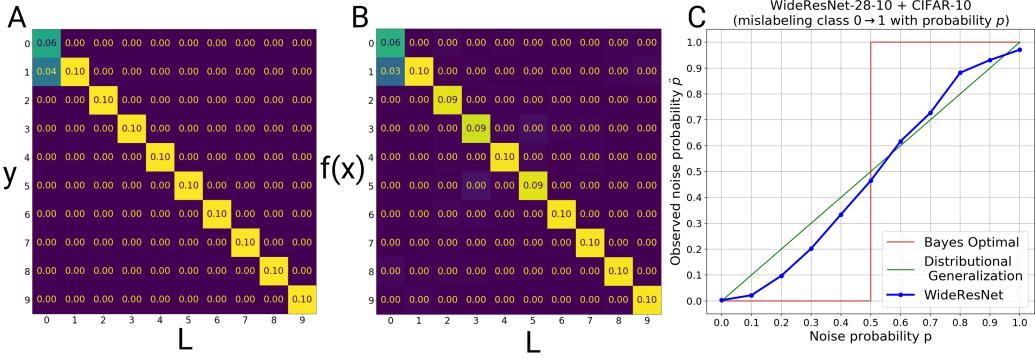


Figure 4: **Feature Calibration with original classes on CIFAR-10:** We train a WRN-28-10 on the CIFAR-10 dataset where we mislabel class $0 \rightarrow 1$ with probability p . (A): Joint density of the distinguishable features L (the original CIFAR-10 class) and the classification task labels y on the train set for noise probability $p = 0.4$. (B): Joint density of the original CIFAR-10 classes L and the network outputs $f(x)$ on the test set. (C): Observed noise probability in the network outputs on the test set (the $(1, 0)$ entry of the matrix in B) for varying noise probabilities p

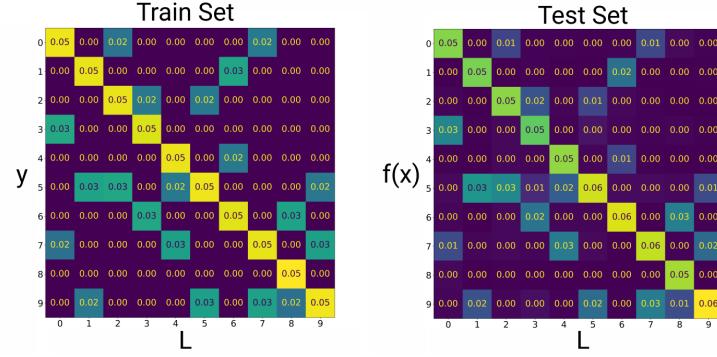


Figure 5: **Feature Calibration with random confusion matrix on CIFAR-10:** Left: Joint density of labels y and original class L on the train set. Right: Joint density of classifier predictions $f(x)$ and original class L on the test set, for a WideResNet28-10 trained to interpolation. These two joint densities are close, as predicted by Conjecture 1.

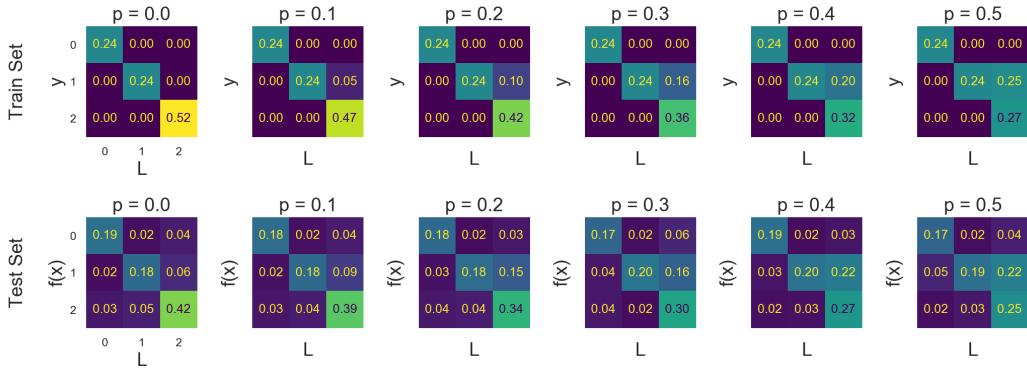


Figure 6: **Feature Calibration for Decision trees on UCI (molecular biology):** We add label noise that takes class 2 to class 1 with probability $p \in [0, 0.5]$. The top row shows the confusion matrix of the true class $L(x)$ vs. the label y on the train set, for varying levels of noise p . The bottom row shows the corresponding confusion matrices of the classifier predictions $f(x)$ on the test set, which closely matches the train set, as predicted by Conjecture 1.

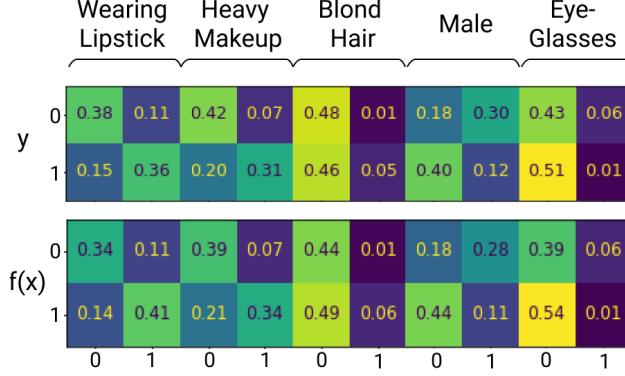


Figure 7: **Feature Calibration for multiple features on CelebA:** We train a ResNet-50 to perform binary classification task on the CelebA dataset. The top row shows the joint distribution of this task label with various other attributes in the dataset. The bottom row shows the same joint distribution for the ResNet-50 outputs on the test set. Note that the network was not given any explicit inputs about these attributes during training.

| Model | AlexNet | ResNet50 |
|--|--------------|----------|
| ImageNet accuracy | 0.565 | 0.761 |
| Accuracy on terriers | 0.572 | 0.775 |
| Accuracy for binary {dog/not-dog} | 0.984 | 0.996 |
| Accuracy on {terrier/not-terrier} among dogs | 0.913 | 0.969 |
| Fraction of real-terriers among dogs | 0.224 | 0.224 |
| Fraction of predicted-terriers among dogs | 0.209 | 0.229 |

Table 1: **Feature Calibration on ImageNet:** ImageNet classifiers are calibrated with respect to dogs. For example, all classifiers predict terrier for roughly $\sim 22\%$ of all dogs (last row), though they may mistake which specific dogs are terriers. See Table 3 in the Appendix for more models.

form a distinguishable feature for ResNet50 as they are learnable to high accuracy [Jahandideh et al., 2018]. We pick one of the hard attributes as the target classification task, where a ResNet-50 achieves 80% accuracy. Then we confirm that the output distribution is calibrated with respect to the attributes that form distinguishable features. In this setting, the label distribution is deterministic, and not directly dependent on the distinguishable features, unlike the experiments considered before. Yet, as we see in Figure 7, the classifier outputs are correctly calibrated for each attribute. Full details of the experiment are described in Appendix D.5.

Coarse Partition: Consider AlexNet trained on ImageNet ILSVRC-2012 [Russakovsky et al., 2015], a 1000-class image classification problem including 116 varieties of dogs. The network only achieves 56.5% accuracy on the test set, but it has higher accuracy on coarser label partitions: for example, it will at least classify most dogs as dogs (with 98.4% accuracy), though it may mistake the specific dog variety. In this example, $L(x) \in \{\text{dog, not-dog}\}$ is the distinguishable feature. Moreover, the network is *calibrated* with respect to dogs: 22.4% of all dogs in ImageNet are Terriers, and indeed, the network classifies 20.9% of all dogs as Terriers (though it has 9% error in which specific dogs it classifies as Terriers). We include similar experiments with ResNets and kernels in Appendix D.

4.4 Discussion

Conjecture 1 claims that \mathcal{D}_{te} is close to \mathcal{D} up to all tests which are *themselves learnable*. That is, if an interpolating method is capable of learning a certain partition of the domain, then it will also produce outputs that are calibrated with respect to this partition, when trained on any problem. This conjecture thus gives a way of quantifying the resolution with which classifiers approximate the source distribution \mathcal{D} , via properties of the classification algorithm itself. This is in contrast to many

classical ways of quantifying the approximation of density estimators, which rely on *analytic* (rather than *operational*) distributional assumptions [Tsybakov, 2008, Wasserman, 2006].

Proper Scoring Rules. If the loss function used in training is a *strictly-proper scoring rule* such as cross-entropy [Gneiting and Raftery, 2007], then we may expect that in the limit of a large-capacity network and infinite data, training on samples $\{(x_i, y_i)\}$ will yield a good density estimate of $p(y|x)$ at the softmax layer. However, this is not what is happening in our experiments: First, our experiments consider the hard-decisions, not the softmax outputs. Second, we observe Conjecture 1 even in settings without proper scoring rules (e.g. kernel SVM and decision trees).

4.5 1-Nearest-Neighbors Connection

Here we show that the 1-nearest neighbor classifier provably satisfies Conjecture 1, under mild assumptions. This is trivially true when the number of train points $n \rightarrow \infty$, such that the train points pack the domain. However, we do not require any such assumptions: the theorem below applies generically to a wide class of distributions, with no assumptions on the ambient dimension of inputs, the underlying metric, or smoothness of the source distribution. All the distributional requirements are captured by the preconditions of Conjecture 1, which require that the feature L is ε -distinguishable to 1-Nearest-Neighbors. The only further assumption is a weak regularity condition: sampling the nearest neighbor train point to a random test point should yield (close to) a uniformly random test point. In the following, $\text{NN}_S(x)$ refers to the nearest neighbor of point x among points in set S .

Theorem 1. *Let \mathcal{D} be a distribution over $\mathcal{X} \times \mathcal{Y}$, and let $n \in \mathbb{N}$ be the number of train samples. Assume the following regularity condition holds: Sampling the nearest neighbor train point to a random test point yields (close to) a uniformly random test point. That is, suppose that for some small $\delta \geq 0$,*

$$\{\text{NN}_S(x)\}_{\substack{S \sim \mathcal{D}^n \\ x \sim \mathcal{D}}} \approx_{\delta} \{x\}_{x \sim \mathcal{D}} \quad (9)$$

Then, Conjecture 1 holds. For all $(\varepsilon, \text{NN}, \mathcal{D}, n)$ -distinguishable partitions L , the following distributions are statistically close:

$$\{(y, L(x))\}_{x, y \sim \mathcal{D}} \approx_{\varepsilon + \delta} \{(\text{NN}_S^{(y)}(x), L(x))\}_{\substack{S \sim \mathcal{D}^n \\ x, y \sim \mathcal{D}}} \quad (10)$$

The proof of Theorem 1 is straightforward, and provided in Appendix G. We view this theorem both as support for our formalism of Conjecture 1, and as evidence that the classifiers we consider in this work have *local* properties similar to 1-Nearest-Neighbors.

Note that Theorem 1 does not hold for the k-nearest neighbor classifier (k-NN), which takes the plurality vote of K neighboring train points. However, it is somewhat more general than 1-NN: for example, it holds for a randomized version of k-NN which, instead of taking the plurality, randomly picks one of the K neighboring train points (potentially weighted) for the test classification.

4.6 Pointwise Density Estimation

In fact, we could hope for an even stronger property than Conjecture 1. Consider the familiar example: we mislabel 20% of dogs as cats in the CIFAR-10 training data, and train an interpolating ResNet on this train set. Conjecture 1 predicts that, *on average* over all test dogs, roughly 20% of them are classified as cats. In fact, we may expect this to hold pointwise for each dog: For a single test dog x , if we train a new classifier f (on fresh iid samples from the noisy distribution), then $f(x)$ will be cat roughly 20% of the time. That is, for each test point x , taking an ensemble over independent train sets yields an estimate of the conditional density $p(y|x)$. Informally:

$$\text{With high probability over test } x \sim \mathcal{D} : \Pr_{f \leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}^n)}[f(x) = \ell] \approx p(y = \ell | x) \quad (11)$$

where the probability on the LHS is over the random sampling of train set, and any randomness in the training procedure. This behavior would be stronger than, and not implied by, Conjecture 1. We give preliminary experiments supporting such a pointwise property in Appendix D.7.

5 Agreement Property

We now present an “agreement property” of various classifiers. This property is independent of the previous section, though both are instantiations of our general indistinguishability conjecture. We

claim that, informally, the test accuracy of a classifier is close to the probability that it agrees with an identically-trained classifier on a disjoint train set.

Conjecture 2 (Agreement Property). *For certain classifier families \mathcal{F} and distributions \mathcal{D} , the test accuracy of a classifier is close to its agreement probability with an independently-trained classifier. That is, let S_1, S_2 be independent train sets sampled from \mathcal{D}^n , and let f_1, f_2 be classifiers trained on S_1, S_2 respectively. Then*

$$\Pr_{\substack{S_1 \sim \mathcal{D}^n \\ f_1 \leftarrow \text{Train}_{\mathcal{F}}(S_1) \\ (x,y) \sim \mathcal{D}}} [f_1(x) = y] \approx \Pr_{\substack{S_1, S_2 \sim \mathcal{D}^n \\ f_i \leftarrow \text{Train}_{\mathcal{F}}(S_i) \\ (x,y) \sim \mathcal{D}}} [f_1(x) = f_2(x)] \quad (12)$$

Moreover, this holds with high probability over training f_1, f_2 : $\Pr_{(x,y) \sim \mathcal{D}} [f_1(x) = y] \approx \Pr_{(x,y) \sim \mathcal{D}} [f_1(x) = f_2(x)]$.

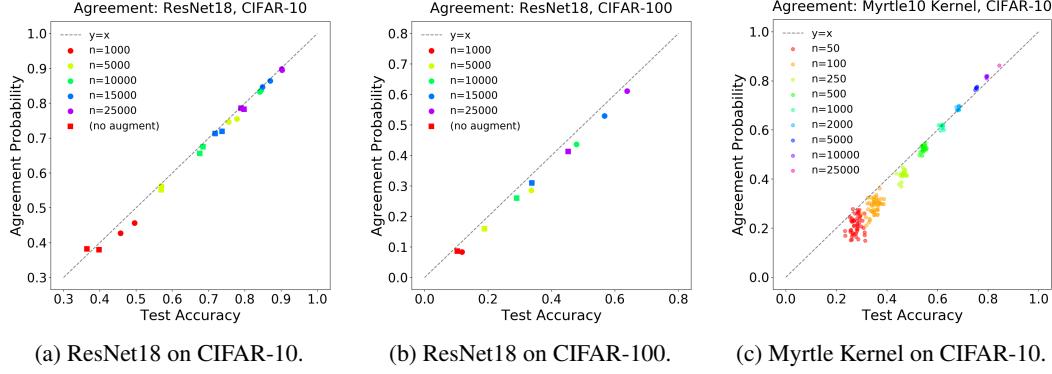


Figure 8: **Agreement Property on CIFAR-10/100.** For two classifiers trained on disjoint train sets, the probability they agree with each other (on the test set) is close to their test accuracy.

The agreement property (Conjecture 2) is surprising for several reasons. First, suppose we have two classifiers f_1, f_2 which were trained on independent train sets, and both achieve test accuracy say 50% on a 10-class problem. That is, they agree with the true label $y(x)$ w.p. 50%. Depending on our intuition, we may expect: (1) They agree with each other much less than they agree with the true label, since each individual classifier is an independently noisy version of the truth, or (2) They agree with each other much more than 50%, since classifiers tend to have “correlated” predictions. However, neither of these are the case in practice.

Second, it may be surprising that the RHS of Equation 12 is an estimate of the test error that requires only unlabeled test examples x . This observation is independently interesting, and may be relevant for applications in uncertainty estimation and calibration. Conjecture 2 also provably holds for 1-Nearest-Neighbors in some settings, under stronger assumptions (Theorem 2 in Appendix G).

Connection to Indistinguishability. Conjecture 2 is in fact an instantiation of our general indistinguishability conjecture. Informally, we can “swap y for $f_2(x)$ ” in the LHS of Equation 12, since they are indistinguishable. Formally, consider the specific test

$$T_{\text{agree}} : (x, \hat{y}) \mapsto \mathbb{1}\{f_1(x) = \hat{y}\} \quad (13)$$

where $f_1 \leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}^n)$. The expectation of this test under the Source Distribution \mathcal{D} is exactly the LHS of Equation 12, while the expectation under the Test Distribution \mathcal{D}_{te} is exactly the RHS. Thus, Conjecture 2 can be equivalently stated as

$$\mathcal{D} \approx^{T_{\text{agree}}} \mathcal{D}_{\text{te}}. \quad (14)$$

5.1 Experiments

In our experiments, we train a pair of classifiers f_1, f_2 on random disjoint subsets of the train set for a given distribution. Both classifiers are otherwise trained identically, using the same architecture, number of train-samples n , and optimizer. We then plot the test error of f_1 against the agreement probability $\Pr_{x \sim \text{TestSet}} [f_1(x) = f_2(x)]$. Figure 8 shows experiments with ResNet18 on CIFAR-10

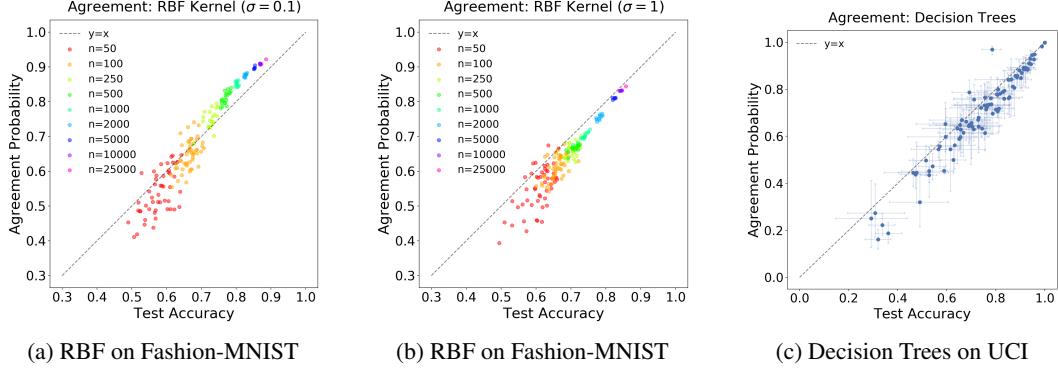


Figure 9: **Agreement Property for RBF and decision trees.** For two classifiers trained on disjoint train sets, the probability they agree with each other (on the test set) is close to their test accuracy. For UCI, each point corresponds to one UCI task, and error bars show 95% Clopper-Pearson confidence intervals in estimating population quantities.

and CIFAR-100, as well as the Myrtle10 kernel from [Shankar et al. \[2020\]](#), with varying number of train samples n . These classifiers are trained with standard-practice training procedures (SGD with standard data-augmentation for ResNets), with no additional hyperparameter tuning. Figure 9 shows experiments with the RBF Kernel on Fashion-MNIST, and decision trees on 92 UCI classification tasks. The Agreement Property approximately holds for all pairs of identical classifiers, and continues to hold even for “weak” classifiers (e.g. when f_1, f_2 have high test error). Full experimental details and further experiments are in Appendix E.

5.2 Potential Mechanisms

We now consider, and refute, several potential mechanisms which could explain the experimental results of Conjecture 2.

5.2.1 Bimodal Samples

A simple model which would exhibit the Agreement Property is the following: Suppose test samples x come in two types: “easy” or “hard.” All classifiers get “easy” samples correct, but they output a uniformly random class on “hard” samples. That is, for a fixed x , consider the probability that a freshly-trained classifier gets x correct. “Easy” samples are such that

$$\text{For } x \in \text{EASY: } \Pr_{f \leftarrow \text{Train}(\mathcal{D}^n)} [f(x) = y(x)] = 1$$

while “hard” samples have a uniform distribution on output classes $[K]$:

$$\text{For } x \in \text{HARD: } \Pr_{f \leftarrow \text{Train}(\mathcal{D}^n)} [f(x) = i] = \frac{1}{K} \quad \forall i \in [K]$$

Notice that for HARD samples x , a classifier f_1 agrees with the true label y with exactly the same probability that it agrees with an independent classifier f_2 (because both f_1, f_2 are uniformly random on x). Thus, the agreement property (Conjecture 2) holds exactly under this model. However, this strict decomposition of samples into “easy” and “hard” does not appear to be the case in the experiments (see Appendix E.3, Figure 22).

5.3 Pointwise Agreement

We could more generally posit that Conjecture 2 is true because the Agreement Property holds *pointwise* for most test samples x . That is, Equation (12) would be implied by:

$$\text{w.h.p. for } (x, y) \sim \mathcal{D} : \Pr_{f_1 \leftarrow \text{Train}(\mathcal{D}^n)} [f_1(x) = y] \approx \Pr_{\substack{f_1 \leftarrow \text{Train}(\mathcal{D}^n) \\ f_2 \leftarrow \text{Train}(\mathcal{D}^n)}} [f_1(x) = f_2(x)] \quad (15)$$

This was the case for the EASY/HARD decomposition above, but could be true in more general settings. Equation (15) is a “pointwise calibration” property that would allow estimating the probability of making an error on a test point x by simply estimating the probability that two independent classifiers agree on x . However, we find (perhaps surprisingly) that this is not the case. That is, Equation (12) holds on average over $(x, y) \sim \mathcal{D}$, but not pointwise for each sample. We give experiments demonstrating this in Appendix E.3. Interestingly, 1-nearest neighbors can satisfy the agreement property of Claim 2 without satisfying the “pointwise agreement” of Equation 15. It remains an open problem to understand the mechanisms behind Agreement Matching.

6 Limitations and Ensembles

The conjectures presented in this work are not fully specified, since they do not exactly specify which classifiers or distributions for which they hold. We experimentally demonstrate instances of these conjectures in various “natural” settings in machine learning, but we do not yet understand which assumptions on the distribution or classifier are required. Some experiments also deviate slightly from the predicted behavior (e.g. the kernel experiments in Figures 3 and 9). Nevertheless, we believe our conjectures capture the essential aspects of the observed behaviors, at least to first order. It is an important open question to refine these conjectures and better understand their applications and limitations—both theoretically and experimentally.

6.1 Ensembles

We could ask if all high-performing interpolating methods used in practice satisfy our conjectures. However, an important family of classifiers which fail our Feature Calibration Conjecture are ensemble methods:

1. Deep ensembles of interpolating neural networks [Lakshminarayanan et al., 2017].
2. Random forests (i.e. ensembles of interpolating decision trees) [Breiman, 2001].
3. k-nearest neighbors (roughly “ensembles” of 1-Nearest-Neighbors) [Fix and Hodges, 1951].

The pointwise density estimation discussion in Section 4.6 sheds some light on these cases. Notice that these are settings where the “base” classifier in the ensemble obeys Feature Calibration, and in particular, acts as an approximate conditional density estimator of $p(y|x)$, as in Section 4.6. That is, if individual base classifiers f_i approximately act as samples from

$$f_i(x) \sim p(y|x)$$

then for sufficiently many classifiers $\{f_1, \dots, f_k\}$ trained on independent train sets, the ensembled classifier will act as

$$\text{plurality}(f_1, f_2, \dots, f_k)(x) \approx \underset{y}{\operatorname{argmax}} p(y|x)$$

Thus, we believe ensembles fail our conjectures because, in taking the plurality vote of base classifiers, they are approximating $\underset{y}{\operatorname{argmax}} p(y|x)$ instead of the conditional density $p(y|x)$ itself. Indeed, in the above examples, we observed that ensemble methods behave much closer to the Bayes-optimal classifier than their underlying base classifiers (especially in settings with label noise).

7 Distributional Generalization: Beyond Interpolating Methods

The previous sections have focused primarily on *interpolating* classifiers, which fit their train sets exactly. Here we discuss the behavior of non-interpolating methods, such as early-stopped neural networks and regularized kernel machines, which do not reach 0 train error.

For non-interpolating classifiers, their outputs on the train set $(x, f(x))_{x \sim \text{TrainSet}}$ will *not* match the original distribution $(x, y) \sim \mathcal{D}$. Thus, there is little hope that their outputs on the test set will match the original distribution, and we do not expect the Indistinguishability Conjecture to hold. However, the Distributional Generalization framework does not require interpolation, and we could still expect that the train and test distributions are close ($\mathcal{D}_{\text{tr}} \approx^{\mathcal{T}} \mathcal{D}_{\text{te}}$) for some family of tests \mathcal{T} . For example, the following is a possible generalization of Feature Calibration (Conjecture 1).

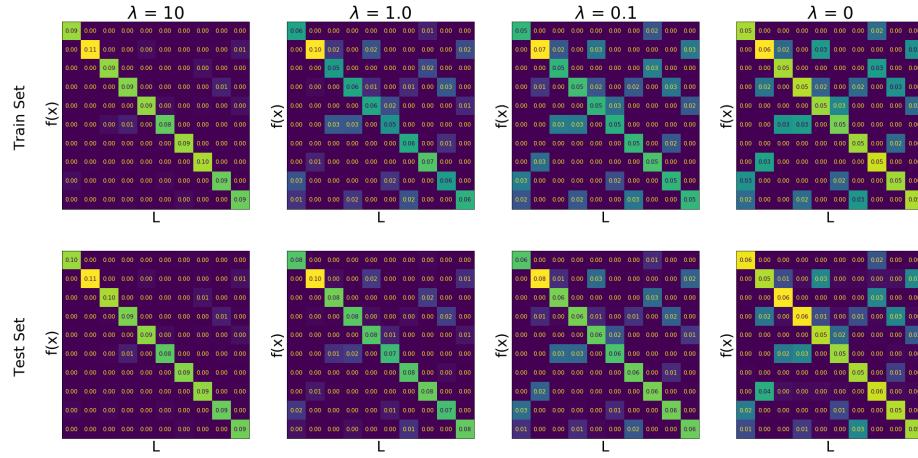


Figure 10: Distributional Generalization for Gaussian Kernel on MNIST. We apply label noise from a random sparse confusion to the MNIST train set. We then train a Gaussian Kernel for classification, with varying L_2 regularization λ . The top row shows the confusion matrix of predictions $f(x)$ vs true labels $L(x)$ on the train set, and the bottom row shows the corresponding confusion matrix on the test set. Larger values of regularization prevents the classifier from fitting label noise on the train set, and this behavior is mirrored almost identically on the test set. Note that all classifiers above are trained on the same train set, with the same label noise.

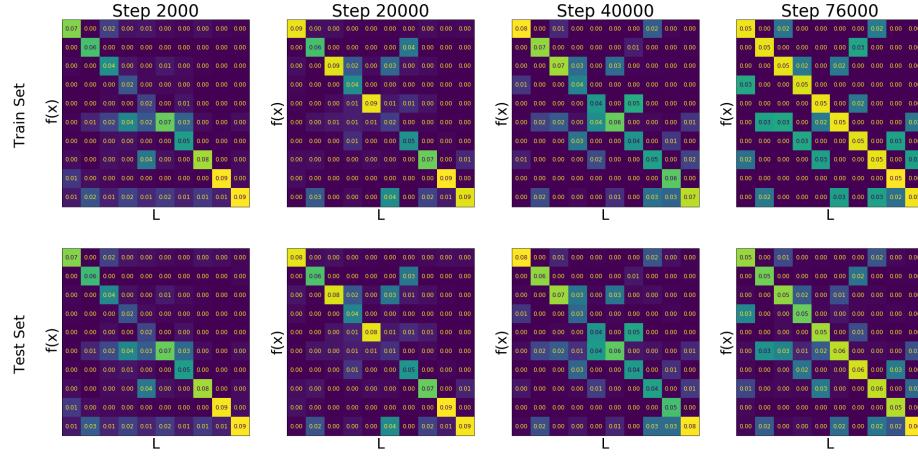


Figure 11: Distributional Generalization for WideResNet on CIFAR-10. We apply label noise from a random sparse confusion to the CIFAR-10 train set. We then train a single WideResNet28-10, and measure its predictions on the train and test sets over increasing train time (SGD steps). The top row shows the confusion matrix of predictions $f(x)$ vs true labels $L(x)$ on the train set, and the bottom row shows the corresponding confusion matrix on the test set. As the network is trained for longer, it fits more of the noise on the train set, and this behavior is mirrored almost identically on the test set.

Conjecture 3 (Generalized Feature Calibration, informal). *For trained classifiers f , the following distributions are statistically close for many partitions L of the domain:*

$$(L(x_i), f(x_i)) \underset{x_i \sim \text{TrainSet}}{\approx} (L(x), f(x)) \underset{x \sim \text{TestSet}}{\approx}$$
 (16)

We leave unspecified the exact set of partitions L for which this holds— unlike Conjecture 1, where we specified L as the set of all distinguishable features. In this generalized case, we do not yet understand the appropriate notion of “distinguishable feature”⁵. However, we give experimental evidence that suggests some refinement of Conjecture 3 is true.

In Figure 10 we train Gaussian kernel regression on MNIST, with label noise determined by a random sparse confusion matrix on the train set (analogous to the setting of Figure 5; experimental details in Appendix C). We vary the amount of ℓ_2 regularization, and plot the confusion matrix of predictions on the train and test sets. With $\lambda = 0$ regularization, the kernel interpolates the noise in the train set exactly, and reproduces this noise on the test set as expected. With higher regularization, the kernel no longer interpolates the train set, but the test and train confusion matrices remain close. That is, regularization prevents the kernel from fitting the noise on both the train and test sets in a similar way. Remarkably, higher regularization yields a classifier closer to Bayes-optimal. Figure 11 shows an analogous experiment for neural networks on CIFAR-10, with early-stopping in place of regularization: early in training, neural networks do not fit their train set, but their test and train confusion matrices remain close throughout training. These experiments suggest that Distributional Generalization is a meaningful notion even for non-interpolating classifiers. Formalizing and investigating this further is an interesting area for future study.

8 Conclusion and Discussion

In this work, we presented a new set of empirical behaviors of standard interpolating classifiers. We unified these under the framework of Distributional Generalization, which states that outputs of trained classifiers on the test set are “close” in distribution to their outputs on the train set. For interpolating classifiers, we stated several formal conjectures (Conjectures 1 and 2) to characterize the form of distributional closeness that can be expected.

Beyond Test Error. Our work proposes studying the *entire distribution* of classifier outputs on test samples, beyond just its test error. We show that this distribution is often highly structured, and we take steps towards characterizing it. Surprisingly, modern interpolating classifiers appear to satisfy certain forms of distributional generalization “automatically,” despite being trained to simply minimize train error. This even holds in cases when satisfying distributional generalization is in conflict with satisfying classical generalization—that is, when a distributionally-generalizing classifier must necessarily have high test error (e.g. Experiment 1). We thus hope that studying distributional generalization will be useful to better understand modern classifiers, and to understand generalization more broadly.

Classical Generalization. Our framework of Distributional Generalization can be insightful even to study classical generalization. That is, even if we ultimately want to understand test error, it may be easier to do so through distributional generalization. This is especially relevant for understanding the success of interpolating methods, which pose challenges to classical theories of generalization. Our work shows new empirical behaviors of interpolating classifiers, as well as conjectures characterizing these behaviors. This sheds new light on these poorly understood methods, and could pave the way to better understanding their generalization.

Interpolating vs. Non-interpolating Methods. Our work also suggests that interpolating classifiers should be viewed as conceptually different objects from non-interpolating ones, even if both have the same test error. In particular, an interpolating classifier will match certain aspects of the original distribution, which a non-interpolating classifier will not. This also suggests, informally, that interpolating methods should not be seen as methods which simply “memorize” their training data in a naive way (as in a look up table)— rather this “memorization” strongly influences the classifier’s decision boundary (as in 1-Nearest-Neighbors).

⁵For example, when considering early-stopped neural networks, it is unclear if the partition L should be distinguishable with respect to the early-stopped network or its fully-trained counterpart.

8.1 Open Questions

Our work raises a number of open questions and connections to other areas. We briefly collect some of them here.

1. As described in the Limitations (Section 6), we do not precisely understand the set of distributions and interpolating classifiers for which our conjectures hold. We empirically tested a number of “realistic” settings, but it is open to state formal assumptions defining these settings.
2. It is open to theoretically prove versions of Distributional Generalization for models beyond 1-Nearest-Neighbors. This is most interesting in cases where Distributional Generalization is at odds with classical generalization (e.g. Figure 4c).
3. It is open to understand the mechanisms behind the Agreement Property (Section 5), theoretically or empirically.
4. In some of our experiments (e.g. Section 4.6), ensembling over independent random-initializations had a similar effect to ensembling over independent train sets. This is related to works on deep ensembles [Fort et al., 2019a, Lakshminarayanan et al., 2017] as well as random forests for conditional density estimation [Athey et al., 2019, Meinshausen, 2006, Pospisil and Lee, 2018]. Investigating this further is an interesting area of future work.
5. There are a number of works suggesting “local” behavior of neural networks, and these are somewhat consistent with our locality intuitions in this work. However, it is open to formally understand whether these intuitions are justified in our setting.
6. We give two families of tests \mathcal{T} for which our Interpolating Indistinguishability conjecture (Equation 3) empirically holds. This may not be exhaustive – there may be other ways in which the source distribution \mathcal{D} and test distribution \mathcal{D}_{te} are close. Indeed, we give preliminary experiments for another family of tests, based on student-teacher training, in Appendix B. It is open to explore more ways in which Distributional Generalization holds, beyond the tests presented here.

Acknowledgements

We especially thank Jacob Steinhardt and Boaz Barak for useful discussions during this work. We thank Vaishaal Shankar for providing the Myrtle10 kernel, the ImageNet classifiers, and advice regarding UCI tasks. We thank Guy Gur-Ari for noting the connection to existing work on networks picking up fine-structural aspects of distributions. We also thank a number of people for reviewing early drafts or providing valuable comments, including: Collin Burns, Mihaela Curmei, Benjamin L. Edelman, Sara Fridovich-Keil, Boriana Gjura, Wenshuo Guo, Thibaut Horel, Meena Jagadeesan, Dimitris Kalimeris, Gal Kaplun, Song Mei, Aditi Raghunathan, Ludwig Schmidt, Ilya Sutskever, Yaodong Yu, Kelly W. Zhang, Ruiqi Zhong.

Work supported in part by the Simons Investigator Awards of Boaz Barak and Madhu Sudan, and NSF Awards under grants CCF 1565264, CCF 1715187 and IIS 1409097. Computational resources supported in part by a gift from Oracle, and Microsoft Azure credits (via Harvard Data Science Initiative). P.N. supported in part by a Google PhD Fellowship. Y.B is partially supported by MIT-IBM Watson AI Lab.

Technologies. This work was built on the following technologies: NumPy [Harris et al., 2020, Oliphant, 2006, Van Der Walt et al., 2011], SciPy [Virtanen et al., 2020], scikit-learn [Pedregosa et al., 2011], PyTorch [Paszke et al., 2019], W&B [Biewald, 2020], Matplotlib [Hunter, 2007], pandas [pandas development team, 2020, Wes McKinney, 2010], SLURM [Yoo et al., 2003], Figma. Neural networks trained on NVIDIA V100 and 2080 Ti GPUs.

References

- Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.
- Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6158–6169, 2019.
- Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332, 2019.
- Susan Athey, Julie Tibshirani, Stefan Wager, et al. Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178, 2019.
- Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.
- Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 2020.
- David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 2020.
- Mikhail Belkin, Daniel J Hsu, and Partha Mitra. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *Advances in neural information processing systems*, pages 2300–2311, 2018a.
- Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. *arXiv preprint arXiv:1802.01396*, 2018b.
- Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.
- Lukas Biewald. Experiment tracking with weights and biases, 2020. URL <https://www.wandb.com/>. Software available from wandb.com.
- Leo Breiman. Reflections after refereeing papers for nips. *The Mathematics of Generalization*, pages 11–15, 1995.
- Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.
- Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, and Ludwig Schubert. Thread: Circuits. *Distill*, 2020. doi: 10.23915/distill.00024. <https://distill.pub/2020/circuits>.
- Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4(3):e15, 2019.
- Niladri S Chatterji and Philip M Long. Finite-sample analysis of interpolating linear classifiers in the overparameterized regime. *arXiv preprint arXiv:2004.12019*, 2020.
- Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. *arXiv preprint arXiv:2002.04486*, 2020.
- Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133–3181, 2014.

Evelyn Fix and JL Hodges. *Discriminatory analysis: nonparametric discrimination, consistency properties*. USAF School of Aviation Medicine, 1951.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019a.

Stanislav Fort, Paweł Krzysztof Nowak, Stanislaw Jastrzebski, and Srinivas Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019b.

Mario Geiger, Stefano Spigler, Stéphane d’Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, and Matthieu Wyart. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115, 2019.

Federica Gerace, Bruno Loureiro, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Generalisation error in learning with random features and the hidden manifold model. *arXiv preprint arXiv:2002.09339*, 2020.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *arXiv preprint arXiv:1904.12191*, 2019.

Dar Gilboa and Guy Gur-Ari. Wider networks learn better features. *arXiv preprint arXiv:1909.11572*, 2019.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Sebastian Goldt, Madhu S Advani, Andrew M Saxe, Florent Krzakala, and Lenka Zdeborova. Generalisation dynamics of online learning in over-parameterised neural networks. *arXiv preprint arXiv:1901.09085*, 2019.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, Sep 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Úrsula Hébert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*, pages 1939–1948, 2018.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

- Rashidedin Jahandideh, Alireza Tavakoli Targhi, and Maryam Tahmasbi. Physical attribute prediction using deep residual neural networks. *arXiv preprint arXiv:1812.07857*, 2018.
- Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *arXiv preprint arXiv:1909.12292*, 2019.
- Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.
- Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Tengyuan Liang and Alexander Rakhlin. Just interpolate: Kernel "ridgeless" regression can generalize. *arXiv preprint arXiv:1808.00387*, 2018.
- Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.
- Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.
- Nicolai Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun): 983–999, 2006.
- Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.
- Vidya Muthukumar, Kailas Vodrahalli, Vignesh Subramanian, and Anant Sahai. Harmless interpolation of noisy data in regression. *IEEE Journal on Selected Areas in Information Theory*, 2020.
- Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.
- Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning, 2019.
- Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020.
- Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. In *Advances in neural information processing systems*, pages 1786–1794, 2010.
- Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.
- Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.
- Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.
- Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.
- Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.
- Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

- Matthew A Olson and Abraham J Wyner. Making sense of random forest probabilities: a kernel perspective. *arXiv preprint arXiv:1812.05792*, 2018.
- The pandas development team. pandas-dev/pandas: Pandas, February 2020. URL <https://doi.org/10.5281/zenodo.3509134>.
- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019. URL <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Taylor Pospisil and Ann B Lee. Rfcde: Random forests for conditional density estimation. *arXiv preprint arXiv:1804.05753*, 2018.
- Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.
- Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.
- David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.
- Jonas Rothfuss, Fabio Ferreira, Simon Walther, and Maxim Ulrich. Conditional density estimation with neural networks: Best practices and benchmarks. *arXiv preprint arXiv:1903.00954*, 2019.
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.
- Robert E Schapire. Theoretical views of boosting. In *European conference on computational learning theory*, pages 1–10. Springer, 1999.
- Robert E Schapire, Yoav Freund, Peter Bartlett, Wee Sun Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.
- Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Ludwig Schmidt, Jonathan Ragan-Kelley, and Benjamin Recht. Neural kernels without tangents. *arXiv preprint arXiv:2003.02237*, 2020.
- Utkarsh Sharma and Jared Kaplan. A neural scaling law from the dimension of the data manifold. *arXiv preprint arXiv:2004.10802*, 2020.
- Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Sunil Thulasidasan, Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. Combating label noise in deep learning using abstention. *arXiv preprint arXiv:1905.10964*, 2019.

Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008.

Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: <https://doi.org/10.1038/s41592-019-0686-2>.

Larry Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

Geoffrey S Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

Abraham J Wyner, Matthew Olson, Justin Bleich, and David Mease. Explaining the success of adaboost and random forests as interpolating classifiers. *The Journal of Machine Learning Research*, 18(1):1558–1590, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Mohammad Yaghini, Bogdan Kulynych, and Carmela Troncoso. Disparate vulnerability: On the unfairness of privacy attacks against machine learning. *arXiv preprint arXiv:1906.00389*, 2019.

Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

Liu Ziyin, Blair Chen, Ru Wang, Paul Pu Liang, Ruslan Salakhutdinov, Louis-Philippe Morency, and Masahito Ueda. Learning not to learn in the presence of noisy labels. *arXiv preprint arXiv:2002.06541*, 2020.

A Author Contributions

PN designed the initial neural network experiments which initiated this study. PN and YB brainstormed the formalization of the experimental observations and PN devised the final version of the definitions, conjectures, and their framing as a version of generalization. YB designed the experiments to stress test the Feature Calibration Conjecture under various settings and conducted the final experiments that appear in the Feature Calibration section. PN discovered and investigated the Agreement Property and did the Student-Teacher section. PN did the kernel and decision tree experiments, literature review, and nearest-neighbor proofs. Both authors wrote the paper.

B Student-Teacher Indistinguishability

Here we show another instance of the Indistinguishability Conjecture, by giving another way in which the distributions \mathcal{D}_{te} and \mathcal{D} are close – specifically, we claim they are roughly indistinguishable with respect to *training*. That is, training a student-network on samples from $(x, y) \sim \mathcal{D}$ yields a similar model as training on pseudo-labeled samples $(x, f(x)) \sim \mathcal{D}_{te}$, as long as the student is “weaker” than the teacher f .

We specifically consider a setup where a teacher network is trained on n samples, and a student network is trained on $k \ll n$ samples. In this setting, we claim that teacher-labeled samples are “as good as” real samples to the student – in that the student achieves similar test accuracy whether trained on real or pseudo-labeled samples. (In practice we find that $k \leq n/2$ is sufficient, though this limit does not appear to be fundamental.)

To see why this may be surprising, consider a setting where we use a teacher that is only say 80% accurate. Now, we may expect that training a student on the pseudo-labeled distribution will always be worse than training on the true labels. After all, the pseudo-labels are only 80% correct. However, we find that when the student is trained on less than half the number samples than the teacher was trained on, the student does just as well as if it were trained on true labels.

Experiments. We use a ResNet18 for all student and teacher models. In Figure 12, we use a fixed teacher network trained on $n \in \{5000, 10000\}$ samples from CIFAR-10. For each k , we compare the test error of the following two networks:

1. A student ResNet18 on trained on k pseudo-labeled samples (call this model $G_{n,k}$: a student trained on k pseudo-samples from a teacher trained on n samples).
2. A ResNet18 trained on k true samples.

When $k \leq n/2$, the test error of the student is close to that of a network trained on true samples. When $k \gg n/2$ however, the student can distinguish whether it is being trained on real or pseudo-labeled samples.

Figure 13 shows test errors of $G_{n,k}$ for all $n, k \in \{1000, 2000, 5000, 10000, 15000, 25000\}$. The test error of $G_{n,k}$ appears to depend only on $\min(n, k)$ – intuitively, the test error is bottlenecked by the minimum power of student and teacher.

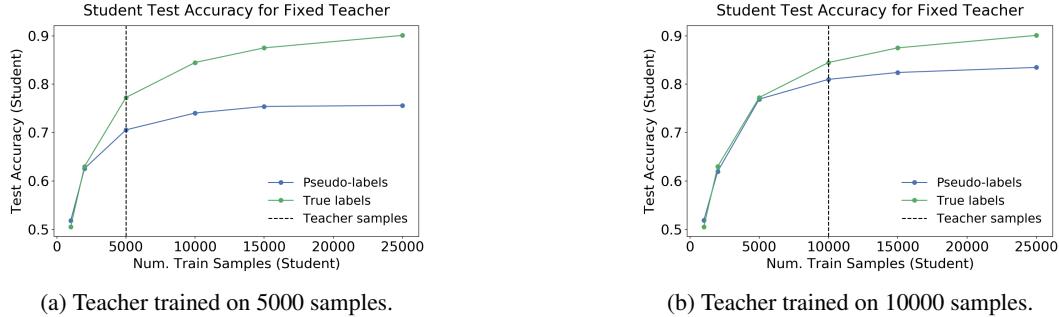


Figure 12: **Pseudo-labeling.** Accuracy of student when trained on true labels vs. pseudo-labels.

Discussion Previous sections considered tests which were asked to distinguish the distributions \mathcal{D} and \mathcal{D}_{te} based on a single sample from either distribution. Here, we consider a more powerful test, which is given access to k iid samples from either \mathcal{D} or \mathcal{D}_{te} .⁶ This student-teacher indistinguishability is also essentially equivalent to the following claim: We cannot learn a ResNet-discriminator between distributions \mathcal{D} and \mathcal{D}_{te} , given $k \leq n/2$ samples from each distribution.

⁶This is not fundamentally different from a single sample test, via a hybrid argument.

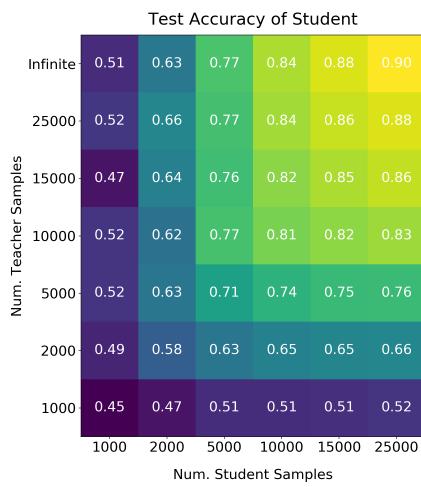


Figure 13: Test error of a student trained on k pseudo-labeled samples (x-axis) from a teacher trained on n samples (y-axis). The top row ($n = \infty$) shows a student trained on true samples from the distribution.

C Experimental Details

Here we describe general background, and experimental details common to all sections. Then we provide section-specific details below.

C.1 Datasets

We consider the image datasets CIFAR-10 and CIFAR-100 [Krizhevsky et al., 2009], MNIST [LeCun et al., 1998], Fashion-MNIST [Xiao et al., 2017], CelebA [Liu et al., 2015], and ImageNet [Russakovsky et al., 2015].

We also consider tabular datasets from the UCI repository Dua and Graff [2017]. For UCI data, we consider the 121 classification tasks as standardized in Fernández-Delgado et al. [2014]. Some of these tasks have very few examples, so we restrict to the 92 classification tasks from Fernández-Delgado et al. [2014] which have at least 200 total examples.

C.2 Models

We consider neural-networks, kernel methods, and decision trees.

C.2.1 Decision Trees

We train interpolating decision trees using a growth rule from Random Forests [Breiman, 2001, Ho, 1995]: selecting a split based on a random \sqrt{d} subset of d features, splitting based on Gini impurity, and growing trees until all leafs have a single sample. This is as implemented by Scikit-learn Pedregosa et al. [2011] defaults with `RandomForestClassifier(n_estimators=1, bootstrap=False)`.

C.2.2 Kernels

Throughout this work we consider classification via kernel regression and kernel SVM. For M -class classification via kernel regression, we follow the methodology in e.g. Belkin et al. [2018b], Rahimi and Recht [2008], Shankar et al. [2020]. We solve the following convex problem for training:

$$\alpha^* := \underset{\alpha \in \mathbb{R}^{N \times M}}{\operatorname{argmin}} \|K\alpha - y\|_2^2 + \lambda \alpha^T K \alpha$$

where $K_{ij} = k(x_i, x_j)$ is the kernel matrix of the training points for a kernel function k , $y \in \mathbb{R}^{N \times M}$ is the one-hot encoding of the train labels, and $\lambda \geq 0$ is the regularization parameter. The solution can be written

$$\alpha^* = (K + \lambda I)^{-1} y$$

which we solve numerically using SciPy `linalg.solve` [Virtanen et al., 2020]. We use the explicit form of all kernels involved. That is, we do not use random-feature approximations [Rahimi and Recht, 2008], though we expect they would behave similarly.

The kernel predictions on test points are then given by

$$g_\alpha(x) := \sum_{i \in [N]} \alpha_i k(x_i, x) \tag{17}$$

$$f_\alpha(x) := \underset{j \in [M]}{\operatorname{argmax}} g_\alpha(x)_j \tag{18}$$

where $g(x) \in \mathbb{R}^M$ are the kernel regressor outputs, and $g(x) \in [M]$ is the thresholded classification decision. This is equivalent to training M separate binary regressors (one for each label), and taking the argmax for classification. We usually consider *unregularized* regression ($\lambda = 0$), except in Section 7.

For kernel SVM, we use the implementation provided by Scikit-learn [Pedregosa et al., 2011] `sklearn.svm.SVC` with a precomputed kernel, for inverse-regularization parameter $C \geq 0$ (larger C corresponds to smaller regularization).

Types of Kernels. We use the following kernel functions $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$.

| | MLP | ResNet18 | WideResNet-28-10 | ResNet50 |
|---------------------------------------|--|--|--|---|
| Batchsize | 128 | 128 | 128 | 32 |
| Epochs | 820 | 200 | 200 | 50 |
| Optimizer | Adam $(\beta_1 = 0.9, \beta_2 = 0.999)$ | SGD + Momentum (0.9) | SGD + Momentum (0.9) | SGD |
| Learning rate (LR) schedule | Constant LR = 0.001 | Initial LR = 0.05 scale by 0.1 at epochs (80, 120) | Initial LR = 0.1 scale by 0.2 at epochs (80, 120, 160) | Initial LR = 0.001, scale by 0.1 if training loss stagnant for 2000 gradient steps |
| Data Augmentation | Random flips + RandomCrop(32, padding=4) | | | |
| CIFAR-10 Error | $\sim 40\%$ | $\sim 8\%$ | $\sim 4\%$ | N/A |

Table 2: Hyperparameters used to train the neural networks and their errors on the unmodified CIFAR-10 dataset

- Gaussian Kernel (RBF): $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|_2^2}{2\tilde{\sigma}^2})$.
- Laplace Kernel: $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|_2}{\tilde{\sigma}})$.
- Myrtle10 Kernel: This is the compositional kernel introduced by [Shankar et al. \[2020\]](#). We use their exact kernel for CIFAR-10.

For the Gaussian and Laplace kernels, we parameterize bandwidth by $\sigma := \tilde{\sigma}/\sqrt{d}$. We use the following bandwidths, found by cross-validation to maximize the unregularized test accuracy:

- MNIST: $\sigma = 0.15$ for RBF kernel.
- Fashion-MNIST: $\sigma = 0.1$ for RBF kernel. $\sigma = 1.0$ for Laplace kernel.
- CIFAR-10: Myrtle10 Kernel from [Shankar et al. \[2020\]](#), and $\sigma = 0.1$ for RBF kernel.

C.2.3 Neural Networks

We use 4 different neural networks in our experiments. We use a multi-layer perceptron, and three different Residual networks.

MLP: We use a Multi-layer perceptron or a fully connected network with 3 hidden layers with 512 neurons in each layer. A hidden layer is followed by a BatchNormalization layer and ReLU activation function.

WideResNet: We use the standard WideResNet-28-10 described in [Zagoruyko and Komodakis \[2016\]](#). Our code is based on [this repository](#).

ResNet50: We use a standard ResNet-50 from the PyTorch library [\[Paszke et al., 2017\]](#).

ResNet18: We use a modification of ResNet18 [He et al. \[2016\]](#) adapted to CIFAR-10 image sizes. Our code is based on [this repository](#).

For Experiment 1 and 2 and Section 4, the hyperparameters used to train the above networks are given in Table 2.

D Feature Calibration: Appendix

D.1 A guide to reading the plots

All the experiments in support of Conjecture 1 (experiments in Section 4 and the Introduction) involve various quantities which we enumerate here

1. Inputs x : Each experiment involves inputs from a standard dataset like CIFAR-10 or MNIST. We use the standard train/test splits for every dataset.
2. Distinguishable feature $L(x)$: This feature depends only on input x . We consider various features like the original classes itself, a superset of classes (as in coarse partition) or some secondary attributes (like the binary attributes provided with CelebA)
3. Output labels y : The output label may be some modification of the original labels. For instance, by adding some type of label noise, or a constructed binary task as in Experiment 1
4. Classifier family F : We consider various types of classifiers like neural networks trained with gradient based methods, kernel and decision trees.

In each experiment, we are interested in two joint densities $(y, L(x))$, which depends on our dataset and task and is common across train and test, and $(f(x), L(x))$ which depends on the interpolating classifiers outputs on the *test* set. Since $y, L(x)$ and $f(x)$ are discrete, we will look at their discrete joint distributions. We sometimes refer to $(y, L(x))$ as the train joint density, as at interpolation $(y, L(x)) = (f(x), L(x))$ for all training inputs x . We also refer to $(f(x), L(x))$ as the test density, as we measure this only on the test set.

D.2 Experiment 1

Experimental details: We now provide further details for Experiment 1. We first construct a dataset from CIFAR-10 that obeys the joint density $(y, L(x))$ shown in Figure 1 left panel. We then train a WideResNet-28-10 (WRN-28-10) on this modified dataset to zero training error. The network is trained with the hyperparameters described in Table 2. We then observe the joint density $(f(x), L(x))$ on the test images and find that the two joint densities are close as shown in Figure 1.

We now consider a modification of this experiment as follows:

Experiment 2. Consider the following distribution over images x and binary labels y . Sample x as a uniformly random CIFAR-10 image, and sample the label as $p(y|x) = \text{Bernoulli}(\text{CIFAR_Class}(x)/10)$. That is, if the CIFAR-10 class of x is $k \in \{0, 1, \dots, 9\}$, then the label is 1 with probability $(k/10)$ and 0 otherwise. Figure 14 shows this joint distribution of (x, y) . As before, train a WideResNet to 0 training error on this distribution.

In this experiment too, we observe that the train and test joint densities are close as shown in Figure 14.

Now, we repeat the same experiment, but with an MLP instead of WRN-28-10. The training procedure is described in Table 2. This MLP has an error on 37% on the original CIFAR-10 dataset.

Since this MLP has poor accuracy on the original CIFAR-10 classification task, it does not form a distinguishable partition for it. As a result, the train and test joint densities (Figure 15) do not match as well as they did for WRN-28-10.

D.3 Constant Partition

We now describe the experiment for a constant partition $L(x) = 0$. For this experiment, we first construct a dataset based on CIFAR-10 that has class-imbalance. For class $k \in \{0, \dots, 9\}$, sample $(k+1) \times 500$ images from that class. This will give us a dataset where classes will have marginal distribution $p(y = \ell) \propto \ell + 1$ for classes $\ell \in [10]$, as shown in Figure 3. We do this both for the training set and the test set, to keep the distribution \mathcal{D} fixed.

Now, we train the MLP, ResNet-18 and RBF Kernel on this dataset. We plot the resulting $p(f(x))$ for each of these models below. That is, we plot the fraction of test images for which the network outputs $\{0, 1, \dots, 9\}$ respectively. As predicted, the networks closely track the train set.

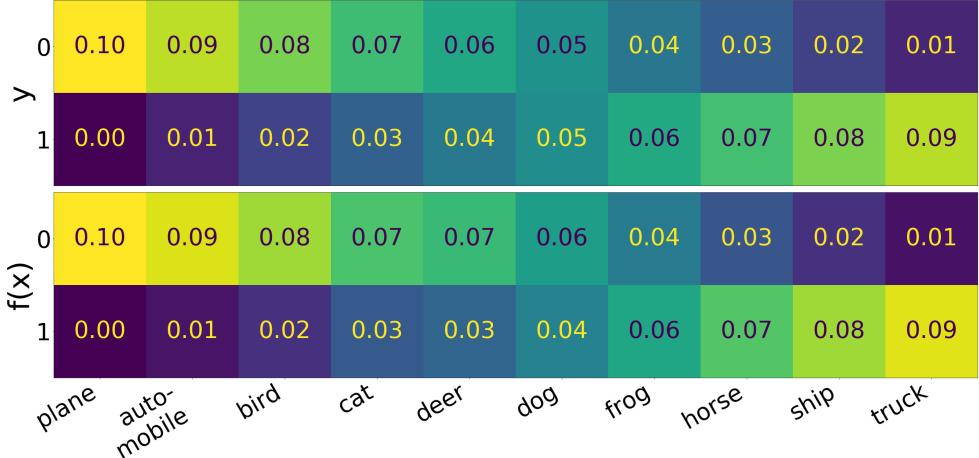


Figure 14: **Distributional Generalization in Experiment 2.** Joint densities of the distributions involved in Experiment 2. The top panel shows the joint density of labels on the train set: $(\text{CIFAR_Class}(x), y)$. The bottom panels shows the joint density of classifier predictions on the test set: $(\text{CIFAR_Class}(x), f(x))$. Distributional Generalization claims that these two joint densities are close.

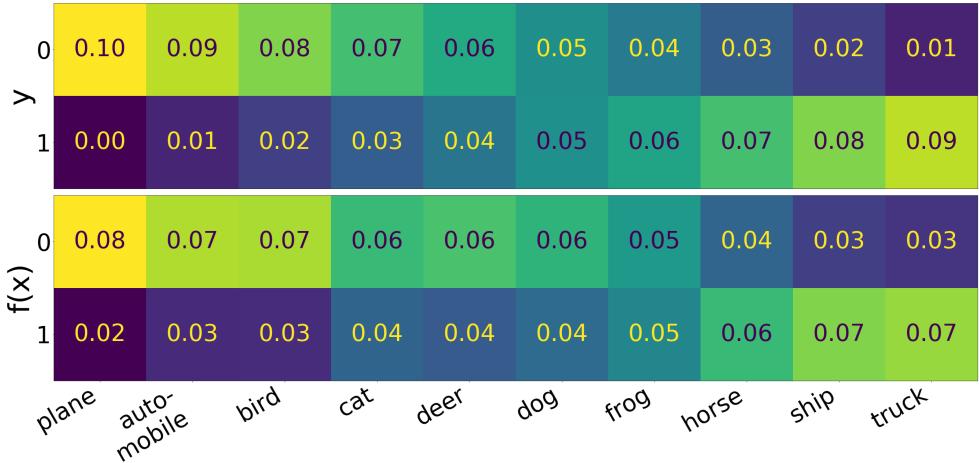


Figure 15: Joint density of $(y, \text{Class}(x))$, top, and $(f(x), \text{Class}(x))$, bottom, for test samples (x, y) from Experiment 2 for an MLP.

D.4 Class Partition

D.4.1 Neural Networks and CIFAR-10

We now provide experiments in support of Conjecture 1 when the class itself is a distinguishable partition. We know that WRN-28-10 achieves an error of 4% on this dataset. Hence, the original labels in CIFAR-10 form a distinguishable partition for this dataset. To demonstrate that Conjecture 1 holds, we consider different structured label noise on the CIFAR-10 dataset. To do so, we apply a variety of confusion matrices to the data. That is, for a confusion matrix $C : 10 \times 10$ matrix, the element c_{ij} gives the joint density that a randomly sampled image had original label j , but is flipped to class i . For no noise, this would be an identity matrix.

We begin by a simple confusion matrix where we flip only one class $0 \rightarrow 1$ with varying probability p . Figure 4 shows one such confusion matrix for $p = 0.4$. We then train a WideResNet-28-10 to zero train error on this dataset. We use the hyperparameters described in C.2. We find that the classifier

outputs on the test set closely track the confusion matrix that was applied to the distribution. Figure 4 shows that this is independent of the value of p and continues to hold for $p \in [0, 1]$.

To show that this is not dependent on the particular class used, we also show that the same holds for a random confusion matrix. We generate a sparse confusion matrix as follows. We set the diagonal to 0.5. Then, for every class j , we pick any two random classes for and set them to 0.2 and 0.3. We train a WRN-28-10 on it and report the test confusion matrix. The resulting train and test densities are shown in Figure 5 and also below for clarity.

D.4.2 Decision Trees

Similar results hold for decision trees; here we show experiments on two UCI tasks: wine and mushroom.

The wine task is a 3-way classification problem: to identify the cultivar of a given wine (out of 3 cultivars), given 13 physical attributes describing the wine. Figure 16 shows an analogous experiment with label noise taking class 1 to class 2.

The mushroom task is a 2-way classification problem: to classify the type of edibility of a mushroom (edible vs poisonous) given 22 physical attributes (e.g. stalk color, odor, etc). Figure 17 shows an analogous experiment with label noise flipping class 0 to class 1.

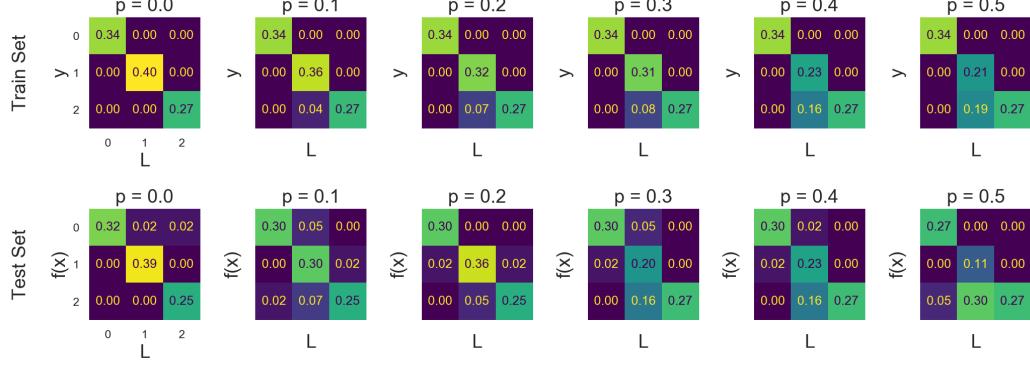


Figure 16: Decision trees on UCI (wine). We add label noise that takes class 1 to class 2 with probability $p \in [0, 0.5]$. Each column shows the test and train confusion matrices for a given p . Note that this decision trees achieve high accuracy on this task with no label noise (leftmost column). We plot the empirical joint density of the train set, and not the population joint density of the train distribution, and thus the top row exhibits some statistical error due to small-sample effects.

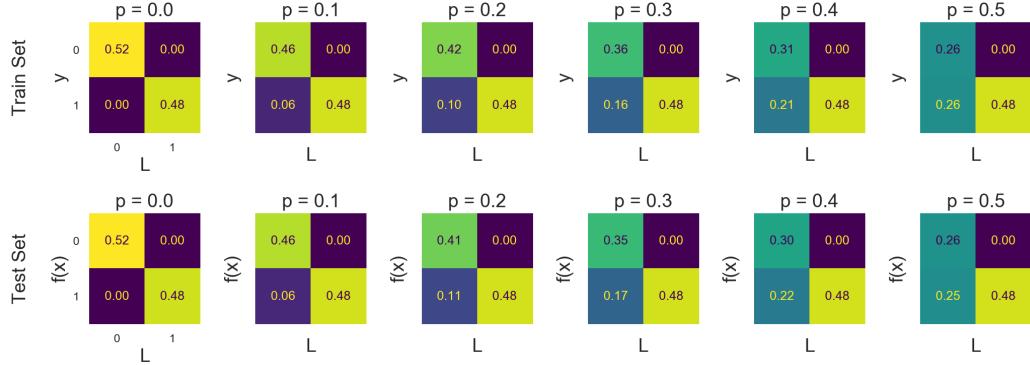


Figure 17: Decision trees on UCI (mushroom). We add label noise that takes class 1 to class 2 with probability $p \in [0, 0.5]$. Each column shows the test and train confusion matrices for a given p . Note that this decision trees achieve high accuracy on this task with no label noise (leftmost column).

| Model | AlexNet | ResNet18 | ResNet50 | BagNet8 | BagNet32 |
|--|---------|----------|----------|---------|----------|
| ImageNet Accuracy | 0.565 | 0.698 | 0.761 | 0.464 | 0.667 |
| Accuracy on dogs | 0.588 | 0.729 | 0.793 | 0.462 | 0.701 |
| Accuracy on terriers | 0.572 | 0.704 | 0.775 | 0.421 | 0.659 |
| Accuracy for binary {dog/not-dog} | 0.984 | 0.993 | 0.996 | 0.972 | 0.992 |
| Accuracy on {terrier/not-terrier} among dogs | 0.913 | 0.955 | 0.969 | 0.876 | 0.944 |
| Fraction of real-terriers among dogs | 0.224 | 0.224 | 0.224 | 0.224 | 0.224 |
| Fraction of predicted-terriers among dogs | 0.209 | 0.222 | 0.229 | 0.192 | 0.215 |

Table 3: ImageNet classifiers are calibrated with respect to dogs: All classifiers predict terrier for roughly $\sim 22\%$ of all dogs (last row), though they may mistake which specific dogs are terriers.

D.5 Multiple Features

For the CelebA experiments, we train a ResNet-50 to predict the attribute {Attractive, Not Attractive}. We choose this attribute because a ResNet-50 performs poorly on this task (test error $\sim 20\%$) and has good class balance. We choose an attribute with poor generalization because the conjecture would hold trivially for if the network generalizes well. We initialize the network with a pretrained ResNet-50 from the PyTorch library [Paszke et al. \[2017\]](#) and use the hyperparameters described in Section C.2 to train on this attribute. We then check the train/test joint density with various other attributes like Male, Wearing Lipstick etc. Note that the network is not given any label information for these additional attributes, but is calibrated with respect to them. That is, the network says $\sim 30\%$ of images that have ‘Heavy Makeup’ will be classified as ‘Attractive’, even if the network makes mistakes on which particular inputs it chooses to do so. Loosely, this can be viewed as the network performing 1-Nearest-Neighbor classification in a metric space that is well separated for each of these distinguishable features.

D.6 Coarse Partition

We now consider cases where the original classes do not form a distinguishable partition for the classifier in consideration. That is, the classifier is not powerful enough to obtain low error on the original dataset, but can perform well on a coarser division of the classes.

To verify this, we consider a division of the CIFAR-10 classes into Objects {airplane, automobile, ship, truck} vs Animals {cat, deer, dog, frog}. An MLP trained on this problem has low error ($\sim 8\%$), but the same network performs poorly on the full dataset ($\sim 37\%$ error). Hence, Object vs Animals forms a distinguishable partition with MLPs. In Figure 18a, we show the results of training an MLP on the original CIFAR-10 classes. We see that the network mostly classifies objects as objects and animals as animals, even when it might mislabel a dog for a cat.

We perform a similar experiment for the RBF kernel on Fashion-MNIST, with partition {clothing, shoe, bag}, in Figure 18b.

ImageNet experiment. In Table 3 we provide results of the terrier experiment in the body, for various ImageNet classifiers. We use publicly available pretrained ImageNet models from [this repository](#), and use their evaluations on the ImageNet test set.

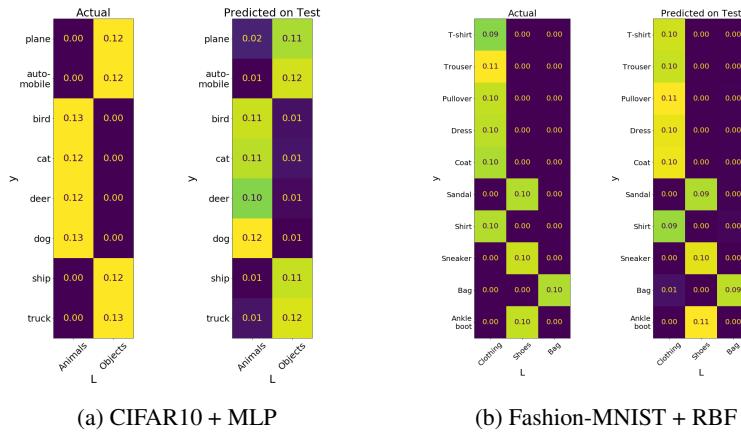


Figure 18: Coarse partitions as distinguishable features: We consider a setting where the original classes are not distinguishable, but the a superset of the classes are.

D.7 Pointwise Density Estimation

Here we describe an informal version of the conditional-density estimation property, and give preliminary experimental evidence to support it. We do not yet understand this property deeply enough to state it formally, but we believe this informal version captures the essential behavior.

Conjecture 4 (Conditional Density Estimation, Informal). *For all distributions $\mathcal{D} \equiv p(x, y)$, number of samples n , family of models \mathcal{F} , and $\varepsilon > 0$, let L_{fine} be a “finest distinguishable partition” — that is, informally an $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$ -distinguishable partition that cannot be refined further. Then:*

$$\text{With high probability over } x \sim \mathcal{D} : \quad \{f(x)\}_{f \leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}^n)} \approx_{\varepsilon} p(y|L_{\text{fine}}(x))$$

Conjecture 4 is a *pointwise* version of Conjecture 1: it says that for most inputs x , we can sample from the conditional density $p(y|L_{\text{fine}}(x))$ by training a fresh classifier f on iid samples from \mathcal{D} , and outputting $f(x)$. We think of $L_{\text{fine}}(x)$ as the “local neighborhood of x ”, or the finest class of x as distinguishable by neural networks. We would ideally like to sample from $p(y|x)$, but the best we can hope for is to sample from $p(y|L_{\text{fine}}(x))$.

Note that this pointwise conjecture must necessarily invoke a notion of “finest distinguishable partition”, which we do not formally define, while Conjecture 1 applied to all distinguishable partitions.

D.7.1 Experiment: Train-set Ensemble

We now give preliminary evidence for this pointwise density estimation. Consider the following simple distribution: MNIST with label noise that flips class 0 to class 1 with probability 40%. To check the pointwise conjecture, we would like to train an ensemble of classifiers $f_i \leftarrow \text{Train}(\mathcal{D}^n)$ on fresh samples from this distribution. We do not have sufficient samples to use truly independent samples, so we approximate this as follows:

1. Sample 5k random examples from the MNIST-train set.
2. Add independent label noise (flipping 0 to 1 w.p. 40%).
3. Train a Gaussian kernel interpolating classifier on these noisy samples (via the hyperparameters in Appendix C).

We train 100 such classifiers, and let $\{f_i\}$ be the ensemble. Then, we claim that if x_0 is a digit 0 in the test set, the empirical distribution $\{f_i(x_0)\}$ over the ensemble is roughly 60% label-0 and 40% label-1. More generally, for all test points x , the distribution $\{f_i(x)\}$ should be close in total variation distance to $p(y|x)$. In Figure 19 we plot the histogram of this TV distance for all x in the test set

$$H(x) := \text{TV}(\{f_i(x)\}, p(y|x))$$

and we see that $H(x)$ is concentrated at 0, with $\mathbb{E}_x[H(x)] \approx 0.036$.

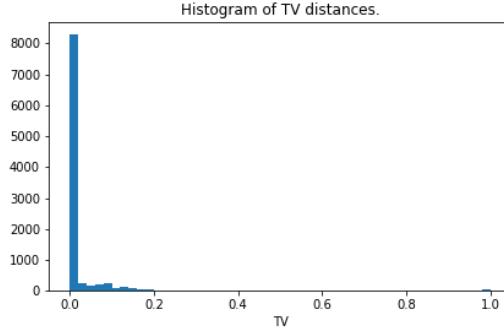


Figure 19: MNIST Ensemble.

Discussion. Here we obtained a conditional density estimate of $p(y|x)$ by training an ensemble of kernel classifiers on (approximately-) independent train sets. In fact, we observed that for some classifiers we can *re-use* the same train set, and get the same behavior – that is, training

an ensemble with fresh random initialization alone. In these cases, randomness in the training procedure is somehow able to substitute for randomness in the sampling procedure. This cannot hold for deterministic training procedures, such as kernel regression, but we have observed it for neural-networks and decision trees. The corresponding statement about decision trees is implicit in works on the conditional-density-estimation properties of random forests (e.g. [Olson and Wyner \[2018\]](#)).

E Agreement Property: Appendix

E.1 Experimental Details

For ResNets on CIFAR-10 and CIFAR-100, we use the following training procedure. For $n \leq 25000$, we sample two disjoint train sets S_1, S_2 of size n from the 50K total train samples. Then we train two ResNet18s f_1, f_2 on S_1, S_2 respectively. We optimize using SGD on the cross-entropy loss, with batch size 128, using learning rate schedule 0.1 for $40\lfloor\frac{50000}{n}\rfloor$ epochs, then 0.01 for $20\lfloor\frac{50000}{n}\rfloor$ epochs. That is, we scale up the number of epoches for smaller train sizes, to keep the number of gradient steps constant. We also early-stop optimization when the train loss reaches < 0.0001 , to save computational time. For experiments with data-augmentation, we use horizontal flips and RandomCrop(32, padding=4). We estimate test accuracy and agreement probability on the CIFAR-10/100 test sets.

For the kernel experiments on Fashion-MNIST, we repeat the same procedure: we sample two disjoint train sets from all the train samples, train kernel regressors, and evaluate their agreement on the test set. Each point on the figures correspond to one trial.

For UCI, some UCI tasks have very few examples, and so here we consider only the 92 classification tasks from [Fernández-Delgado et al. \[2014\]](#) which have at least 200 total examples. For each task, we randomly partition all the examples into a 40%-40%-20% split for 2 disjoint train sets, and 1 test set (20%). We then train two interpolating decision trees, and compare their performance on the test set. Decision trees are trained using a growth rule from Random Forests [\[Breiman, 2001, Ho, 1995\]](#): selecting a split based on a random \sqrt{d} subset of d features, splitting based on Gini impurity, and growing trees until all leafs have a single sample. This is as implemented by Scikit-learn [Pedregosa et al. \[2011\]](#) defaults with `RandomForestClassifier(n_estimators=1, bootstrap=False)`.

In Figure 9c of the body, each point corresponds to one UCI task, and we plot the means of agreement probability and test accuracy when averaged over 100 random partitions for each task. Figure 20 shows the corresponding plot for a single trial.

E.2 Additional Plots

Figure 21 shows the Laplace Kernel on Fashion-MNIST.

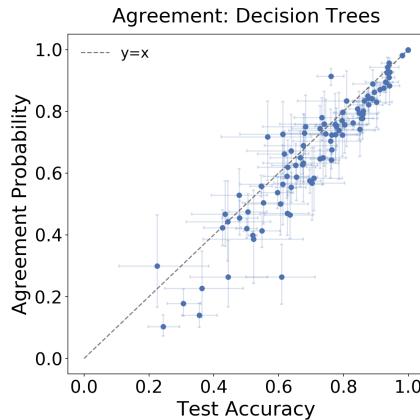


Figure 20: Agreement Probability for a single trial of UCI classification tasks. Analogous to Figure 9c in the body, for a single trial.

E.3 Alternate Mechanisms

We would like to understand the distribution of $f \leftarrow \text{Train}(\mathcal{D}^n)$, in order to evaluate the proposed mechanisms in Sections 5.2.1 and 5.3. Technically, sampling from this distribution requires training a classifier on a *fresh* train set. Since we do not have infinite samples for CIFAR-10, we construct

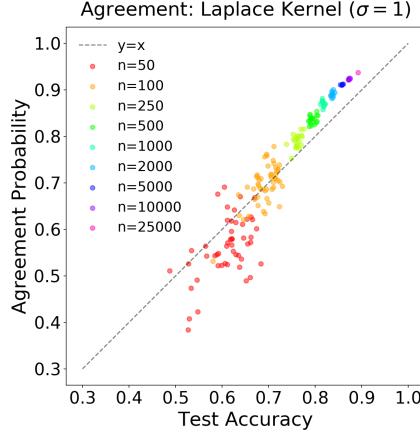


Figure 21: Laplace Kernel on Fashion-MNIST

empirical estimates by training an ensemble of classifiers on random subsets of CIFAR-10. Then, to approximate a sample $f \leftarrow \text{Train}(\mathcal{D}^n)$, we simply sample from our ensemble $f \leftarrow \{f_i\}_i$.

E.3.1 Bimodal Samples

Figure 22 shows a histogram of

$$h(x) := \Pr_{f \leftarrow \text{Train}(\mathcal{D}^n)} [f(x) = y]$$

for test samples x in CIFAR-10, where f is a ResNet18 trained on 5000 samples⁷. This quantity can be interpreted as the “easiness” of a given test sample (x, y) to a certain classifier family.

If the EASY/HARD bimodal model were true, we would expect the distribution of $h(x)$ to be concentrated on $h(x) = 1$ (easy samples) and $h(x) = 0.1$ (hard samples). But this is not the case in Figure 22.

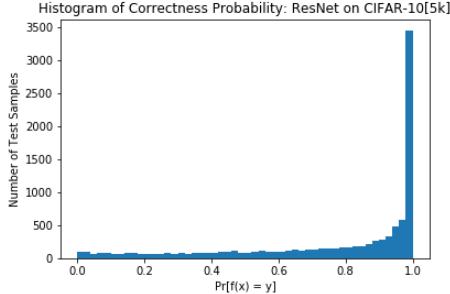


Figure 22: Histogram of sample-hardnesses.

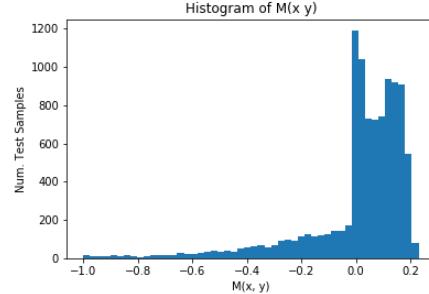


Figure 23: Pointwise agreement histogram.

E.3.2 Pointwise Agreement

We could more generally posit that Conjecture 2 is true because the Agreement Property holds *pointwise* for most test samples x :

$$\text{w.h.p. for } (x, y) \sim \mathcal{D} : \Pr_{f_1 \leftarrow \text{Train}(\mathcal{D}^n)} [f_1(x) = y] \approx \Pr_{\substack{f_1 \leftarrow \text{Train}(\mathcal{D}^n) \\ f_2 \leftarrow \text{Train}(\mathcal{D}^n)}} [f_1(x) = f_2(x)] \quad (19)$$

However, we find (perhaps surprisingly) that this is not the case. To see why this is surprising, observe that Conjecture 2 implies that the agreement probability is close to test accuracy, *in expectation* over

⁷We estimate this probability over the empirical ensemble $f \leftarrow \{f_i\}_i$, where each f_i is a classifier trained on a random $5k$ -subset of CIFAR-10. We train 100 classifiers in this ensemble.

the test sample and the classifiers $f_1, f_2 \leftarrow \text{Train}(\mathcal{D}^n)$:

$$\begin{aligned} \Pr_{\substack{f_1 \\ (x,y) \sim \mathcal{D}}} [f_1(x) = y] &\approx \Pr_{\substack{f_1, f_2 \\ (x,y) \sim \mathcal{D}}} [f_1(x) = f_2(x)] && (\text{Conjecture 2}) \\ \iff \mathbb{E}_{f_1} \mathbb{E}_{x,y \sim \mathcal{D}} [\mathbb{1}\{f_1(x) = y\}] &\approx \mathbb{E}_{f_1, f_2} \mathbb{E}_{x,y \sim \mathcal{D}} [\mathbb{1}\{f_1(x) = f_2(x)\}] && (20) \end{aligned}$$

Swapping the order of expectation, this implies

$$\mathbb{E}_{x,y \sim \mathcal{D}} \underbrace{\left[\mathbb{E}_{f_1, f_2} [\mathbb{1}\{f_1(x) = y\} - \mathbb{1}\{f_1(x) = f_2(x)\}] \right]}_{M(x,y)} \approx 0 \quad (21)$$

Now, we may expect that this means $M(x, y) \approx 0$ pointwise, for most test samples (x, y) . But this is not the case. It turns out that $M(x, y)$ takes on significantly positive and negative values, and these effects “cancel out” in expectation over the distribution, to yield Conjecture 2.

For example, we compute $M(x, y)$ for the Myrtle10 kernel on CIFAR-10 with 1000 train samples.⁸

1. The agreement probability is within 0.8% of the test error (as in Figure 8c), and so

$$\mathbb{E}_{x,y \sim \mathcal{D}} [M(x, y)] \approx 0.008$$

2. However, $M(x, y)$ is not pointwise close to 0. E.g,

$$\mathbb{E}_{x,y \sim \mathcal{D}} [|M(x, y)|] \approx 0.133$$

Figure 23 plots the distribution of $M(x, y)$. We see that some samples (x, y) have high agreement probability, and some low, and these happen to balance in expectation to yield the test accuracy.

⁸We estimate the expectation in $M(x, y)$ by training an ensemble of 5000 pairs of classifiers (f_1, f_2) , each pair on disjoint train samples.

F Non-interpolating Classifiers: Appendix

Here we give an additional example of distributional generalization: in kernel SVM (as opposed to kernel regression, in the main text).

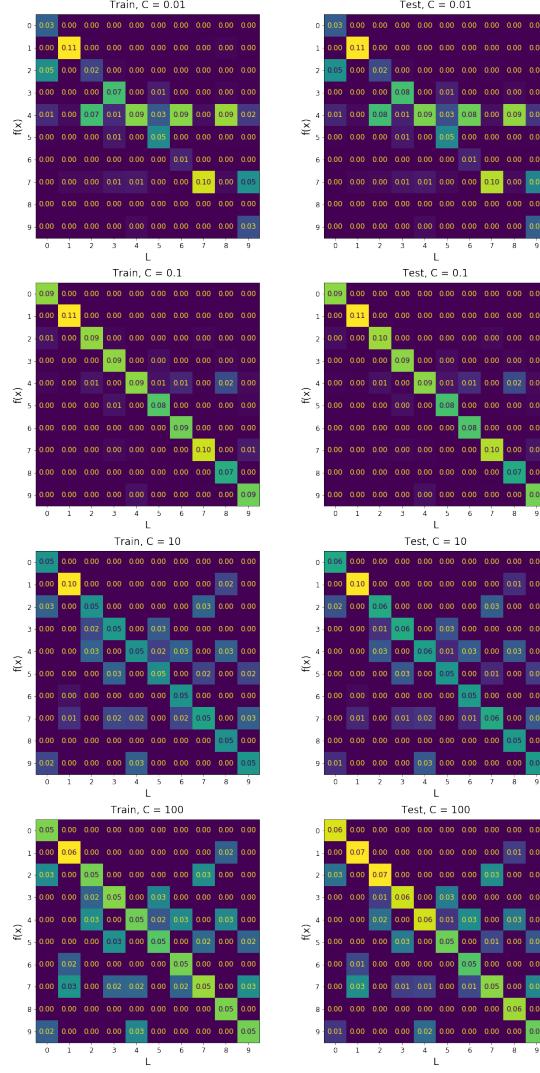


Figure 24: Distributional Generalization. Train (left) and test (right) confusion matrices for kernel SVM on MNIST with random sparse label noise. Each row corresponds to one value of inverse-regularization parameter C . All rows are trained on the same (noisy) train set.

G Nearest-Neighbor Proofs

G.1 Density Calibration Property

Proof of Theorem 1. Recall that L being an $(\varepsilon, \text{NN}, \mathcal{D}, n)$ -distinguishable partition means that nearest-neighbors works to classify $L(x)$ from x :

$$\Pr_{\substack{\{x_i, y_i\} \sim \mathcal{D}^n \\ S = \{(x_i, L(x_i))\} \\ x, y \sim \mathcal{D}}} [\text{NN}_S^{(y)}(x) = L(x)] \geq 1 - \varepsilon \quad (22)$$

Now, we have

$$\{(\text{NN}_S^{(y)}(x), L(x))\}_{S \sim \mathcal{D}^n, x, y \sim \mathcal{D}} \equiv \{(\hat{y}_i, L(x))\}_{\substack{S \sim \mathcal{D}^n \\ \hat{x}_i, \hat{y}_i \leftarrow \text{NN}_S(x) \\ x, y \sim \mathcal{D}}} \quad (23)$$

$$\approx_{\varepsilon} \{(\hat{y}_i, L(\hat{x}_i))\}_{\substack{S \sim \mathcal{D}^n \\ \hat{x}_i, \hat{y}_i \leftarrow \text{NN}_S(x) \\ x, y \sim \mathcal{D}}} \quad (24)$$

$$\approx_{\delta} \{(\hat{y}_i, L(\hat{x}_i))\}_{\hat{x}_i, \hat{y}_i \sim \mathcal{D}} \quad (25)$$

Line (24) is by distinguishability, since $\Pr[L(x) \neq L(\hat{x}_i)] \leq \varepsilon$. And Line (25) is by the regularity condition. \square

G.2 Agreement Property

Theorem 2 (Agreement Property). *For a given distribution \mathcal{D} on (x, y) , and given number of train samples $n \in \mathbb{N}$, suppose NN satisfies the following regularity condition: If we sample two independent train sets S_1, S_2 , then the following two “couplings” are statistically close:*

$$\{(x_i, \text{NN}_{S_2}(x_i))\}_{\substack{S_1 \sim \mathcal{D}^n \\ S_2 \sim \mathcal{D}^n \\ x_i \in R S_1}} \approx_{\delta} \{(\text{NN}_{S_1}(x), \text{NN}_{S_2}(x))\}_{\substack{S_1 \sim \mathcal{D}^n \\ S_2 \sim \mathcal{D}^n \\ x \sim \mathcal{D}}} \quad (26)$$

The LHS is simply a random test point x_i , along with its nearest-neighbor in the train set. The RHS produces an (x_i, x_j) by sampling two independent train sets, sampling a test point $x \sim \mathcal{D}$, and producing the nearest-neighbor of x in S_1 and S_2 respectively.

Then:

$$\Pr_{\substack{S \sim \mathcal{D}^n \\ (x, y) \sim \mathcal{D}}} [\text{NN}_S^{(y)}(x) = y] \approx_{\delta} \Pr_{\substack{S_1 \sim \mathcal{D}^n \\ S_2 \sim \mathcal{D}^n \\ (x, y) \sim \mathcal{D}}} [\text{NN}_{S_1}^{(y)}(x) = \text{NN}_{S_2}^{(y)}(x)] \quad (27)$$

Proof. Let the LHS of Equation (26) be denoted as distribution P over $\mathcal{X} \times \mathcal{X}$. And let Q be the RHS of Equation (26). Let \mathcal{D}_x denote the marginal distribution on x of \mathcal{D} , and let $p(y|x)$ denote the conditional distribution with respect to \mathcal{D} .

The proof follows by considering the sampling of train set S in the following order: first, sample all the x -marginals: sample test point $x \sim \mathcal{D}_x$ and train points $S_x \sim \mathcal{D}_x^n$. Then compute the nearest-neighbors $\hat{x} \leftarrow \text{NN}_{S_x}(x)$. And finally, sample the values y of all the points involved, according to the densities $p(y|x)$.

$$\Pr_{\substack{S \sim \mathcal{D}^n \\ (x,y) \sim \mathcal{D}}} [\text{NN}_S^{(y)}(x) = y] = \mathbb{E}_{\substack{S \sim \mathcal{D}^n \\ (x,y) \sim \mathcal{D}}} [\mathbb{1}\{\text{NN}_S^{(y)}(x) = y\}] \quad (28)$$

$$= \underbrace{\mathbb{E}_{\substack{S_x \sim \mathcal{D}_x^n \\ x \sim \mathcal{D}_x \\ \hat{x} \leftarrow \text{NN}_{S_x}(x)}} \left[\underbrace{\mathbb{E}_{\substack{y \sim p(y|x) \\ \hat{y} \sim p(y|\hat{x})}} [\mathbb{1}\{\hat{y} = y\}]}_{T(x,\hat{x})} \right]}_{(29)}$$

$$= \mathbb{E}_{\substack{S_x \sim \mathcal{D}_x^n \\ x \sim \mathcal{D}_x \\ \hat{x} \leftarrow \text{NN}_{S_x}(x)}} [T(x,\hat{x})] \quad (30)$$

$$= \mathbb{E}_{(x_1,x_2) \sim P} T(x_1,x_2) \quad (P: \text{LHS of Equation (26)})$$

$$\approx_{\delta} \mathbb{E}_{(x_1,x_2) \sim Q} T(x_1,x_2) \quad (Q: \text{RHS of Equation (26)})$$

$$= \mathbb{E}_{\substack{S_1 \sim \mathcal{D}_x^n \\ S_2 \sim \mathcal{D}_x^n \\ x \sim \mathcal{D}_x \\ \hat{x}_1 \leftarrow \text{NN}_{S_1}(x) \\ \hat{x}_2 \leftarrow \text{NN}_{S_2}(x)}} [T(\hat{x}_1,\hat{x}_2)] \quad (31)$$

$$= \mathbb{E}_{\substack{S_1 \sim \mathcal{D}_x^n \\ S_2 \sim \mathcal{D}_x^n \\ x \sim \mathcal{D}_x \\ \hat{x}_1 \leftarrow \text{NN}_{S_1}(x) \\ \hat{x}_2 \leftarrow \text{NN}_{S_2}(x)}} \left[\mathbb{E}_{\substack{\hat{y}_1 \sim p(y|\hat{x}_1) \\ \hat{y}_2 \sim p(y|\hat{x}_2)}} [\mathbb{1}\{\hat{y}_1 = \hat{y}_2\}] \right] \quad (32)$$

$$= \mathbb{E}_{\substack{S_1 \sim \mathcal{D}_x^n \\ S_2 \sim \mathcal{D}_x^n \\ (x,y) \sim \mathcal{D}}} [\mathbb{1}\{\text{NN}_{S_1}^{(y)}(x) = \text{NN}_{S_2}^{(y)}(x)\}] \quad (33)$$

$$= \Pr_{\substack{S_1 \sim \mathcal{D}^n \\ S_2 \sim \mathcal{D}^n \\ (x,y) \sim \mathcal{D}}} [\text{NN}_{S_1}^{(y)}(x) = \text{NN}_{S_2}^{(y)}(x)] \quad (34)$$

as desired. \square