

# Deep Learning Through the Lens of Example Difficulty

Robert J. N. Baldock\*  
 Google Research, Brain Team  
 rjnbaldock@gmail.com

Hartmut Maennel  
 Google Research, Brain Team  
 hartmutm@google.com

Behnam Neyshabur  
 Google Research, Blueshift Team  
 neyshabur@google.com

## Abstract

Existing work on understanding deep learning often employs measures that compress all data-dependent information into a few numbers. In this work, we adopt a perspective based on the role of individual examples. We introduce a measure of the computational difficulty of making a prediction for a given input: the (*effective*) *prediction depth*. Our extensive investigation reveals surprising yet simple relationships between the prediction depth of a given input and the model’s uncertainty, confidence, accuracy and speed of learning for that data point. We further categorize difficult examples into three interpretable groups, demonstrate how these groups are processed differently inside deep models and showcase how this understanding allows us to improve prediction accuracy. Insights from our study lead to a coherent view of a number of separately reported phenomena in the literature: early layers generalize while later layers memorize; early layers converge faster and networks learn easy data and simple functions first.

## 1 Introduction

Much of the existing work on understanding deep learning “integrates out” the data, viewing the inductive bias of the model, or the properties of the optimizer as central to the success of the approach. Examples of such work include studies of eigenvalues of the Hessian and the geometry of the loss landscape (Ghorbani et al., 2019; Yao et al., 2020; Sagun et al., 2016; Li et al., 2018; Pennington and Bahri, 2017; Sagun et al., 2018), studies of margin and effective generalization measures (Long and Sedghi, 2019; Unterthiner et al., 2020; Jiang et al., 2020, 2018; Kawaguchi et al., 2017) and mean-field studies of stochastic optimization (Smith et al., 2021; Stephan et al., 2017; Smith and Le, 2018). However, in practice, we are rarely concerned with only the average behavior of a model.

One pathway to understanding the principles that govern how deep models process data is to study the properties of deep models for data points with different “amounts” or “types” of *example difficulty*. There are a number of definitions of example difficulty in the literature (E.g. see Carlini et al. (2019); Hooker et al. (2019); Lalor et al. (2018); Agarwal and Hooker (2020)). Two are particularly relevant to this work. Firstly, the probability of predicting the ground truth label for an example, when that example is omitted from the training set (Jiang et al., 2021), which represents a *statistical view* of example difficulty. Secondly, the difficulty of learning an example, parameterized by the earliest training iteration after which the model predicts the ground truth class for that example in all subsequent iterations (Toneva et al., 2019). This measure represents a *learning view* of example difficulty <sup>1</sup>.

These notions suffer from two fundamental limitations. While early-exit strategies in computer vision (Teerapittayanon et al., 2016; Huang et al., 2018) and NLP (Dehghani et al., 2018; Liu et al., 2020b; Schwartz et al., 2020; Xin et al., 2020) suggest predictions for easier examples require less computation, the above example difficulty notions do not encapsulate the processing of data inside a given converged model. Moreover,

---

\*Work completed as part of the Google AI Residency Program

<sup>1</sup>We expand on other notions of example difficulty in Appendix B.

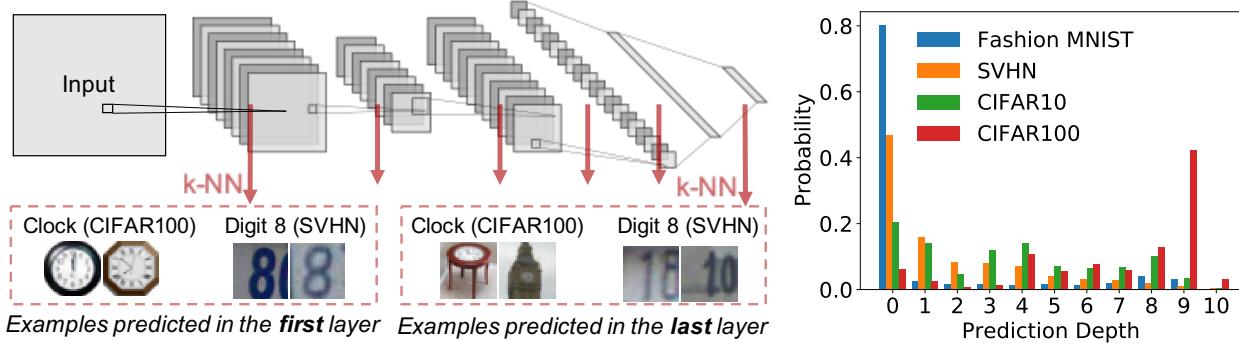


Figure 1: *Deep models use fewer layers to (effectively) determine the prediction for easy examples and more layers for hard examples.* **Left:** A cartoon illustrating the definition of prediction depth (given in Section 2.1). Also shown are training examples from CIFAR100 (“Clock”) and SVHN (“Digit 8”). The examples shown are predicted at the input (first layer) or softmax (last layer) of ResNet18. The examples predicted in the input are visually typical (“easy”), while those predicted in the softmax are mislabeled and/or visually confusing (“hard” examples). To find the prediction depth, we build k-NN classifiers from the embeddings of the training set in different layers of the model. The prediction depth corresponds to the earliest layer at which the predictions of all subsequent k-NN classifiers converge to a fixed label. **Right:** Probability of prediction depth in ResNet18 models for four datasets (training split). We see that the four distributions have different characteristic prediction depths. Ranking the mean prediction depths of these datasets in ascending order, we observe: Fashion MNIST (smallest), SVHN (second), CIFAR10 (third), and CIFAR100 (largest). This order aligns with how one might intuitively rank the difficulties of these classification tasks.

existing notions of example difficulty (E.g. Carlini et al. (2019)) provide a one-dimensional view of difficulty which can not distinguish between examples that are difficult for different reasons.

In this paper, we take a significant step towards resolving the above shortcomings. To take the processing of the data into account we propose a new measure of example difficulty, the prediction depth, which is determined from the hidden embeddings. To escape the one-dimensional view of difficulty, we introduce three distinct difficulty types by relating the hidden embeddings for an input to high-level concepts about example difficulty: “Does this example look mislabeled?”; “Is classifying this example only easy if the label is given?”; “Is this example ambiguous both with and without its label?”. Furthermore, we show how this enhanced notion of example difficulty can unify our understanding of several seemingly unrelated phenomena in deep learning. We hope that the results presented in this work will aid the development of models that capture heteroscedastic uncertainty, our understanding of how deep networks respond to distributional shift, and the advancement of curriculum learning approaches and machine learning fairness. These connections are discussed in Section 5.

**Contributions** Our main contributions are as follows:

- We introduce a measure of *computational example difficulty*: the *prediction depth* (PD). The prediction depth, illustrated in Figure 1, represents the number of hidden layers after which the network’s final prediction is already (effectively) determined (Section 2).
- We show that the prediction depth is larger for examples that visually appear to be more difficult, and that prediction depth is consistent between architectures and random seeds (Section 2.2).
- Our empirical investigation reveals that prediction depth appears to establish a *linear* lower bound on the consistency of a prediction. We further show that predictions are on average more accurate for validation points with small prediction depths (Section 3.1).
- We demonstrate that final predictions for data points that converge earlier during training are typically determined in earlier layers which establishes a correspondence between the training history of the network and the processing of data in the hidden layers (Section 3.2).

- We show that both the adversarial input margin and the output margin are larger for examples with smaller prediction depths. We further design an intervention to reduce the output margin of a network and show that this leads to predictions being made only in the latest hidden layers (Section 3.3).
- We identify three extreme forms of example difficulty by considering the prediction depth in the training and validation splits independently and demonstrate how a simple algorithm that uses the hidden embeddings in one middle layer to make predictions can lead to dramatic improvements in accuracy for inputs that strongly exhibit a specific form of example difficulty (Section 4).
- We use our results to present a coherent picture of deep learning that unify four seemingly unrelated deep learning phenomena: early layers generalize while later layers memorize; networks converge from input layer towards output layer; easy examples are learned first and networks present simpler functions earlier in training (Section 5).

**Experimental Setup:** To ensure that our results are robust to the choice of architectures and datasets, we report empirical findings for ResNet18 (He et al., 2016), VGG16 (Simonyan and Zisserman, 2015) and MLP architectures trained on CIFAR10, CIFAR100 (Krizhevsky et al., 2009), Fashion MNIST (FMNIST) (Xiao et al., 2017) and SVHN (Netzer et al., 2011) datasets. All models were trained using SGD with momentum. Our MLP comprises 7 hidden layers of width 2048 with ReLU activations. Details of the datasets, architectures, and hyperparameters used can be found in Appendix A.

**Related Work:** Our work uses hidden layer probes to determine example difficulty. We have discussed how our study relates to prior work on example difficulty. Hidden layer probes have also been used to study deep learning. Deep k-NN methods (Papernot and McDaniel, 2018) determine their predictions and estimate their own uncertainties by comparing the hidden embeddings of an input to those of the training set. Cohen et al. (2018) showed that SVM, k-Nearest Neighbors (k-NN) and logistic regression probes achieve similar accuracies. However, they did not study the processing of individual data points nor did they relate the k-NN accuracy to notions of example difficulty. Alain and Bengio (2017) used linear classifier probes in the hidden layers to interrogate deep models and demonstrated that linear separability of the embeddings increases monotonically with depth. We provide a more detailed discussion of related work in Appendix B.

## 2 Prediction Depth: a Computational View of Example Difficulty

We discussed the *statistical* and *learning* views of example difficulty in Section 1. In this section, we introduce a *computational* view of example difficulty parametrized by the prediction depth as defined in Section 2.1. This computational view asserts that, for “easy” examples, a deep model’s final prediction is effectively made after only a few layers, while more layers are used for “difficult” examples.

### 2.1 Definition

Asserting that the final prediction is effectively determined in earlier layers of a model, before the output, we estimate the depth at which a prediction is made for a given input as follows <sup>2</sup>:

1. We construct k-NN classifier probes from the embeddings of the training set after particular layers of the network, including the input and the final softmax. The placement of k-NN probes is described in Appendix A.5. We use  $k = 30$  in the k-NN probes. Appendix A.4 establishes that the k-NN accuracies we report are insensitive to  $k$  over a wide range.
2. A prediction is defined to be made at a depth  $L = l$  if the k-NN classification after layer  $L = l - 1$  is different from the network’s final classification, but the classifications of k-NN probes after every layer

---

<sup>2</sup>In the process of arriving at this definition of the prediction depth we considered several alternatives, including using the ground truth class in place of the predicted class and using logistic regression probes in place of k-NN probes. See Appendix E for a discussion on the choices we made in our definition.

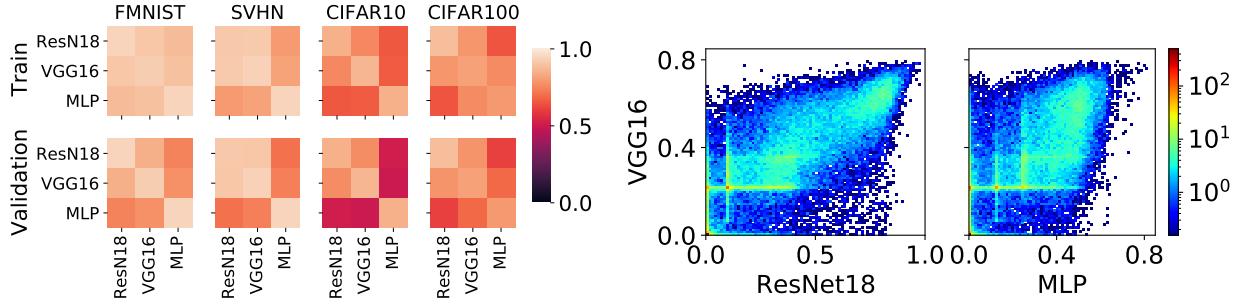


Figure 2: *Consistency of prediction depth between architectures and random seeds.* **Left:** The panel shows the correlation coefficient between prediction depths in different architectures, for both train and validation splits in four datasets. Diagonal comparisons between an architecture and itself show the correlation for the same architecture trained with different random seeds. **Right:** Histograms comparing the mean value of prediction depth obtained for each data point in the training set of CIFAR10 from an ensemble of 250 trained models. In this plot, for visual simplicity, we rescale prediction depth to the interval  $[0, 1]$  for each network. Similar results for all other datasets are presented in Appendix C.1.

$L \geq l$  are all equal to the final classification of the network. Data points consistently classified by all  $k$ -NN probes are determined to be (effectively) predicted in layer 0 (the input)<sup>3</sup>.

It is worth noting that the prediction depth can be calculated for all data points: both in the training and validation splits. This leads to two notions of computational difficulty:

- The difficulty of predicting the (given) class for an input (in the training split)
- The difficulty of making a prediction for an input, unseen in advance (from the validation split)

We examine both notions of computational difficulty in this paper and use the distinction between them to describe different forms of example difficulty in Section 4.

## 2.2 Prediction depth is a meaningful and robust notion of example difficulty

In this section we show that prediction depth agrees with intuitive notions of example difficulty and that it is consistent between different training runs and similar architectures.

**Prediction depth is higher for examples and datasets that seem more difficult** If prediction depth is a sensible measure of example difficulty then we would expect the following sanity checks to be observed:

1. Individual data points that are visually confusing or mislabeled should have larger prediction depths as compared to images that are clear examples of their class.
2. Data points from tasks that are intuitively simpler should have lower prediction depths on average.

Figure 1 shows that the prediction depth passes both of these sanity checks.

**Prediction depth is consistent across random seeds and similar architectures** Figure 2 shows that the prediction depth is highly consistent between different architectures and random seeds for all datasets. Perfect agreement is not expected as different deep learning algorithms have different inductive biases which affects the perceived difficulty of examples. We observe stronger correlation between prediction depth for

<sup>3</sup>Implementation details can be found in Appendix A.6.1.

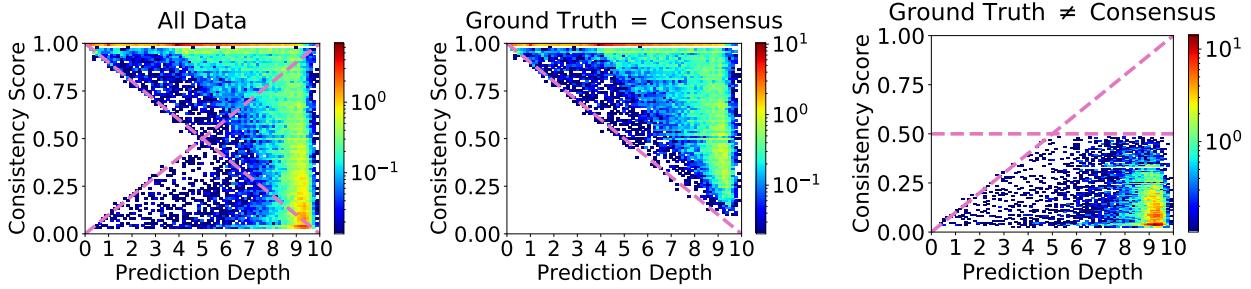


Figure 3: *Consistency score vs. prediction depth in the validation split (left) can be understood as the superposition of two simple functions (middle and right).* We trained 250 ResNet18 models on CIFAR10, with 90:10% random train:validation splits as described in Appendix A. These histograms compare the frequency of correct predictions to the average prediction depth for a data point when it occurs in the validation split. The average prediction depth forms two, surprisingly simple, linear bounds on the consistency score (see Section 3.1 for a full description.) This Figure is reproduced for all datasets and architectures in Appendix C.2, illustrating the consistency of this result.

ResNet18 and VGG16, than between VGG16 and MLP. This may be explained by the fact that ResNet18 and VGG16 are both convolutional networks and we expect their inductive biases to be more similar to one another than to MLP.

### 3 Deep Learning Phenomena Through the Lens of Prediction Depth

In this section, we explore how the prediction depth can be used to better understand three important aspects of Deep Learning: accuracy and consistency of a prediction; the order in which data is learned and the simplicity of the learned function (as measured by the margin) in the vicinity of a data point.

#### 3.1 Depth of a prediction gives a linear lower bound on its consistency

Adopting a *statistical view* of example difficulty, Jiang et al. (2021) identified example difficulty with the expected accuracy of the learning algorithm for a given input, averaged over models trained on different random subsets of the training set with different random seeds. In this section, we clarify the relationship between the prediction depth and the expected accuracy by disentangling the accuracy from the sensitivity of predictions to the particular training split and random seed. Following Jiang et al. (2021), we measure the expected accuracy using the consistency score.

**Consistency score  $\hat{C}$ :** Consistency score is the frequency of classifying an example correctly when it is omitted from the training set. An empirical estimator of the consistency score for a validation point  $(x, y)$  is given by (Jiang et al., 2021):

$$\hat{C}_{A,\mathcal{S}}(x, y) = \hat{\mathbb{E}}_{\tilde{\mathcal{S}} \sim \mathcal{S} \setminus \{(x, y)\}}^r [\delta_{y_A, y}] \quad (1)$$

where  $A$  is a deep learning algorithm (architecture, loss and optimizer),  $y$  is the ground truth class for  $x$ ,  $\tilde{\mathcal{S}}$  is a random subset of  $n$  points sampled from a training dataset  $\mathcal{S}$  excluding  $(x, y)$ ,  $y_A$  is the predicted class of  $x$  for  $A$  trained with data  $\tilde{\mathcal{S}}$ ,  $\delta$  is the Kronecker delta and  $\hat{\mathbb{E}}^r$  denotes empirical averaging with  $r$  i.i.d. samples of such subsets  $\tilde{\mathcal{S}}$ .

Figure 3 (left panel) shows the relationship between consistency score and prediction depth. This plot indicates a surprising piecewise linear boundary which is symmetric around consistency score  $\frac{1}{2}$ . This suggests the existence of a missing concept that could simplify the picture. We next show that the missing concept is the notion of a consensus class which is defined below.

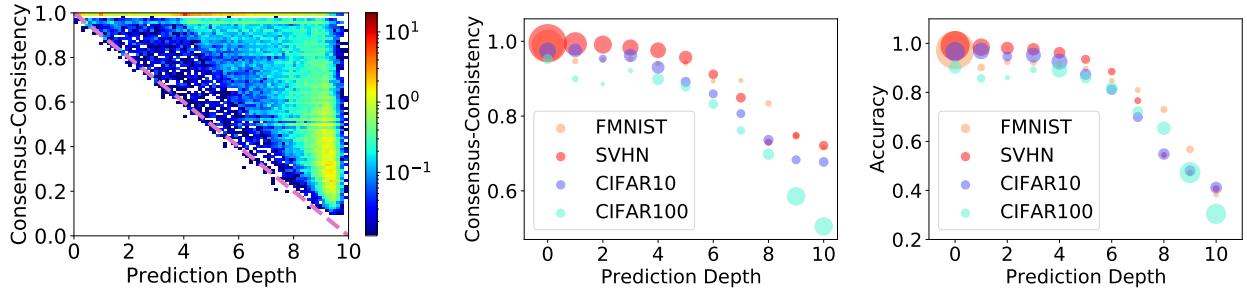


Figure 4: **Left:** Prediction depth provides us with a linear lower bound on consensus-consistency. Results for CIFAR100 with ResNet18. We train 250 models (90:10% random train:validation splits) and compare the average prediction depth when a point occurs in the validation set, to the consensus-consistency of the corresponding predictions. Predictions made for points with low mean prediction depths are highly consistent. Conversely, predictions for points with high mean prediction depths are typically more sensitive to the particular training split and random seed used during training. This left plot shows the result for CIFAR100 with ResNet18. **Middle:** Prediction depth in one model predicts the consensus-consistency of an ensemble that does not include that model. For each dataset we train 25 ResNet18 models with the full training set (see Appendix A). The consensus-consistency of each test point is obtained from 24 of the models, while the prediction depth is obtained from the remaining 1 model. We see that prediction depth in one model predicts the consensus-consistency of a separate ensemble: a measure of the uncertainty of the prediction. The size of each marker in the middle and right plots shows the fraction of the dataset with each prediction depth. **Right:** Prediction depth predicts accuracy. For each dataset we train 250 ResNet18 models (90:10% random train:validation splits). Each time a point appears in the validation split we record the prediction depth and whether the prediction was correct. Predictions made in earlier layers are more likely to be correct. Consistency of these plots is demonstrated for all datasets and architectures in Appendix C.2 where we also describe the relationship between the prediction depth and the entropy of the predictions for an ensemble.

**Consensus class  $\hat{y}_A$ :** The *consensus class* of  $x$  is defined as the predicted class for input  $x$  by a majority voting ensemble of  $r$  models each of which is trained on a randomly chosen subset  $\tilde{\mathcal{S}} \stackrel{n}{\sim} \mathcal{S} \setminus \{(x, y)\}$ <sup>4</sup>.

Figure 3 (middle and right) shows how conditioning on whether consensus class matches the ground truth can change the relationship between consistency score and the prediction depth. For points where the consensus class matches the ground truth (middle) we see that the prediction depth forms a, surprisingly simple, linear lower bound on the consistency score. For points where the consensus class differs from the ground truth (right) at low prediction depth the consistency score is bounded from above by a line that reflects the bound from the middle plot in  $\hat{C} = \frac{1}{2}$ , suggesting that such points are repeatedly mislabeled with a wrong class label. At high prediction depth, the consistency score is low, which suggests highly inconsistent predictions and low accuracy. This result suggests a simple hypothesis: that predictions with low prediction depth are consistent with the *consensus class*, whether that matches the ground truth class or not, while predictions made in later layers depend strongly on the specific training split and random seed used for training and initialization. We measure consistency with the consensus class using the consensus-consistency score.

**Consensus-consistency score  $C^*$ :** The fraction of models in an ensemble that predict the ensemble’s consensus class  $\hat{y}_A(x)$  for an unseen input  $x$ .

$$C_{A, \mathcal{S}}^*(x) = \hat{\mathbb{E}}_{\tilde{\mathcal{S}} \stackrel{n}{\sim} \mathcal{S} \setminus \{(x, y)\}}^r [\delta_{y_A, \hat{y}_A(x)}] \quad (2)$$

where the notation is the same as in (1)<sup>5</sup>.

<sup>4</sup>Implementation details can be found in Appendix A.6.2

<sup>5</sup>Consensus-consistency score is a measure of uncertainty and can be used for calibration (Lakshminarayanan et al., 2017; Wenzel et al., 2020; Wen et al., 2019). See Appendix A.6.3 for details of our implementation.

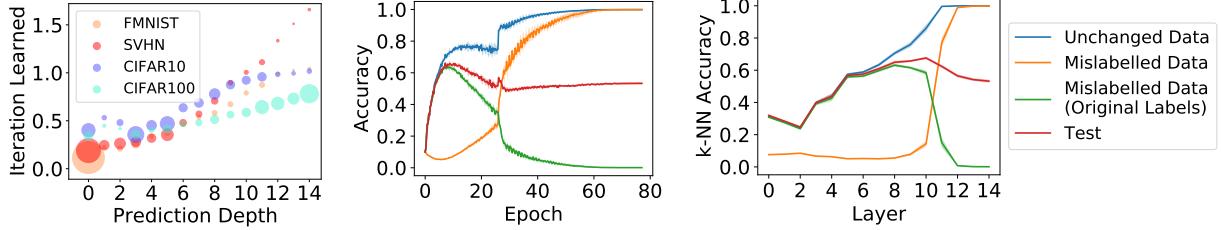


Figure 5: **Left:** Data points with small prediction depths are on average learned before data points with higher prediction depths. We train 250 VGG16 models for each dataset, using a 90:10% random train:validation split as described in Appendix A. Each time an input appears in the validation split we record the prediction depth and the iteration learned in that model. This plot shows the average iteration learned for data points at each prediction depth. Marker size shows the fraction of the dataset with each prediction depth. **Middle and right:** The training learning curve (middle) shares several important features with the inference learning curve (right). Blue, yellow and green curves represent different components of the CIFAR10 training split, in which we have randomized (and fixed) 40% of the labels, and red curves show the test split. The middle and right plots show results from 5 random seeds. The inference learning curve (right) is the sequence of k-NN probe accuracy values for each split. All three plots show results for VGG16. The hyperparameters used are given in Appendix A.

Figure 4 (left) establishes that our simple hypothesis is indeed correct: the prediction depth forms a linear lower bound on the consensus-consistency score for all data points, irrespective of whether the consensus class matches or differs from the ground truth. Interestingly, Figure 4 (middle and right) shows how the prediction depth in a single model, can be used to estimate both of these quantities. That is, predictions of data points with lower prediction depth are both more likely to be consistent and more likely to be correct.

### 3.2 The prediction depth of an input is correlated with its learning difficulty

In Section 3.1, we describe the relationship between the prediction depth, which represents a *computational view* of example difficulty and the consistency and consensus-consistency scores, which represent a *statistical view*. In this section we compare prediction depth to a *learning view* of example difficulty. We measure the difficulty of learning an example by the speed at which the model’s prediction converges for that input during training. The following definition is adapted from Toneva et al. (2019):

**Iteration learned** A data point is said to be learned by a classifier at training iteration  $t = \tau$  if the predicted class at iteration  $t = \tau - 1$  is different from the final prediction of the converged network and the predictions at all iterations  $t \geq \tau$  are equal to the final prediction of the converged network. Data points consistently classified after all training steps and at the moment of initialization, are said to be learned in step  $t = 0$ <sup>6</sup>.

Figure 5 (left plot) shows the positive correlation between the prediction depth and the iteration learned, for all four datasets in VGG16. Consistent results are presented for all architectures and datasets, in both the validation and training splits in Appendix C.3. As a result of the reported correlation, we anticipate that many of the data points correctly classified by the k-NN probe in a particular layer should also be correctly classified by the network at a corresponding interval of training steps. If this is correct then we would expect there to be a visual correspondence between the *training learning curve* (which shows how the accuracy of the network changes during training) and the accuracy of the k-NN probes as data passes from input, through the network, towards the output layer. We call the series of k-NN probe accuracies the *inference learning curve*.

To test this hypothesis we train a model on a training split where a subset of labels are corrupted and compare the training and inference learning curves on four splits of the data: unchanged training data;

<sup>6</sup>Note that this definition can be applied to points in both training and validation splits. In order to compare different models and datasets we rescale the iteration learned in each model so that the 95th percentile occurs at 1.0 and network initialization at 0.

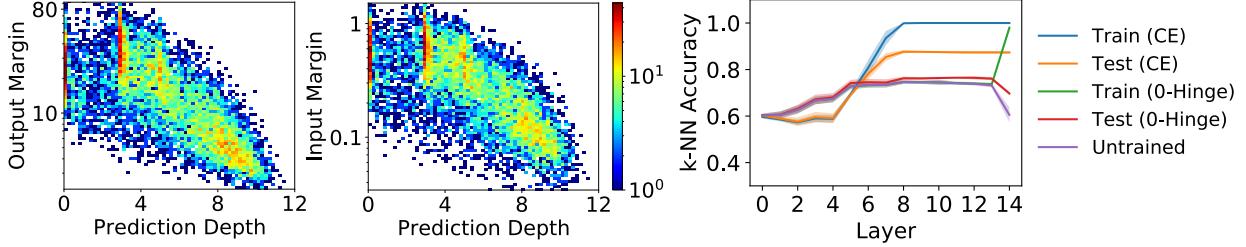


Figure 6: **Left and Middle:** Test examples with smaller prediction depths, on average, have larger output and input margins. We train 25 VGG16 models with different random seeds on CIFAR10 (see Appendix A for details) and compare the mean prediction depth of each test point in these 25 runs to its mean output and input margins (log scales). Correlation coefficients are  $-0.70$  (output margin) and  $-0.69$  (input margin). Although the prediction depth could be at most 14, no data point has an average prediction depth greater than 12. **Right:** An intervention that does not encourage large output margin (“0-Hinge”) results, as predicted, in models where the predictions are effectively determined in higher layers in the network compared to the standard training (“CE”).

mislabeled training data; the original labels of the mislabeled training data and the test split. In Figure 5 (middle and right plots) we see that many of the important features of the training learning curve are indeed present in the inference learning curve. During training (middle), mislabeled data are initially processed as though they are a member of their original class (before they were mislabeled) (Liu et al., 2020a). After an initial period of learning, the network begins to learn the new (random) labels that have been assigned to those data points, so the orange curve moves upwards, and the green curve downwards. At this point, a maximum is observed in the training accuracy (Arpit et al., 2017). In the right plot we see that these same phenomena occur in the inference learning curve.

### 3.3 Deep models exhibit larger margins for inputs with lower prediction depth

It is reported in the literature that deep networks learn functions of increasing complexity during training (Hu et al., 2020; Kalimeris et al., 2019). We frame this observation differently: the learned function is “locally simpler” in the vicinity of data points with smaller prediction depths, and these points are typically learned earlier in training (Section 3.2).

Two known measures of the simplicity of a learned function are the output margin (the difference between the largest and second-largest logits) and the adversarial input margin (the smallest norm required for an adversarial perturbation in the input to change the model’s class prediction). We estimate the adversarial input margin,  $\gamma$ , with a linear approximation (Jiang et al., 2018): for an input  $x$  with predicted class  $i$ ,  $\gamma \simeq \min_{j \neq i} \frac{|z_i - z_j|}{|\nabla_x(z_i - z_j)|}$  where  $z_j$  is the logit returned by the network for class  $j$ . Figure 6 (left and middle plots) show that data points with smaller prediction depths have both larger input and output margins on average and that variances of the input and output margins decrease as the prediction depth increases.

To illustrate the strength of the relationship between the prediction depth and output margin, we demonstrate that reducing the output margin of the learned function results in a model that clusters the data only in the latest layers: such a solution has a very high average prediction depth. We do not minimize the output margin directly but rather use a loss and an optimizer that do not encourage high output margin. Naturally there are many unknowns that may contribute to this effect. We simply report the intervention and the outcome.

The intervention is performed as follows: we construct a loss function that does not promote confidence: a zero-margin hinge loss (“0-Hinge”), and optimize the network using *full-batch* gradient descent with momentum and *very small learning rate*. For an input  $x$  with label  $i$  the 0-Hinge loss is given by  $l(x) = \sum_{j \neq i} \max(0, z_i - z_j)$  where  $z_j$  represents the logit for class  $j$ . The form of this intervention is justified in Appendix A.7. As a

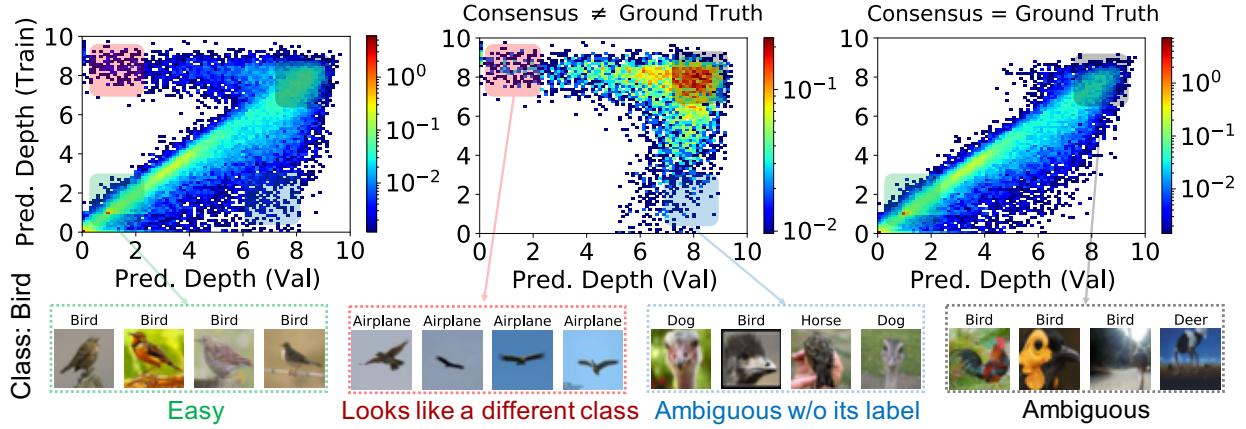


Figure 7: The prediction depth can be the same, or very different for the same input when it occurs in the train and validation splits. Corners of this plot correspond to different forms of example difficulty. (See Section 4 for discussion.) We train 250 ResNet18 models on CIFAR10 with random 90:10% train:validation splits as described in Appendix A. These histograms compare average prediction depth for each data point when it occurs in the validation split vs the training split. This behavior is consistently reproduced for all datasets and architectures in Appendix C.5. Below we show extreme (not hand-chosen) images of “Birds” that appear closest to the corners of this plot. The consensus class is given above each image (tiebreaks favor the class “Bird”).

control, we additionally train a model in the standard fashion using the cross-entropy loss and SGD with momentum and large initial learning rate. Since full-batch gradients are computationally expensive, we train on a subset of CIFAR10 (see Appendix A.7, where we also give the hyperparameters and learning curves.). The output margin obtained with the intervention is 5 orders of magnitude smaller than in the control experiment:  $2.0 \times 10^{-4} \pm 2.0 \times 10^{-4}$  for the 0-Hinge loss and  $1.6 \times 10^1 \pm 0.50 \times 10^1$  for cross-entropy loss. Figure 6 (right) compares the accuracies of the k-NN probes resulting from these training approaches. The 0-Hinge loss training achieves only a marginal improvement in accuracy (red) over an untrained network (purple), and the training split is accurately clustered only in the latest layers. This confirms the predicted behavior: the intervention leads to a model that exhibits both very small average output margins and very late clustering of the data. Very late clustering of the data implies high prediction depths since the k-NN probe classifications change in the latest layers for many data points.

## 4 Beyond a One-Dimensional Picture of Example Difficulty

In this section we transcend the one-dimensional picture of example difficulty by identifying different underlying reasons behind the difficulty of an example, in a way that is general to different architectures and datasets.

Figure 7 shows that the prediction depth can be different when an input occurs in the training split vs. the validation split. Thus, there are two axes of example difficulty:

1. Difficulty of making a prediction when an input is in the validation set
2. Difficulty of finding commonalities during training with other examples of the same ground truth class

**Both axes have a range from “clear” to “ambiguous”.** In Section 3.1 we show that predictions made for validation points with later prediction depths are often inconsistent, with low consensus-consistency. Conversely, a low prediction depth typically indicates an input with high consensus-consistency. For Axis 1 we will identify validation points with low prediction depths as “clear” and those with high prediction depths as “ambiguous”. We will additionally identify a low or high prediction depth in the training split with examples that are respectively “clear” and “ambiguous” on Axis 2. By making combinations of low/high values of ( $\text{PD}_{\text{Val.}}$ ,  $\text{PD}_{\text{Train.}}$ ) we obtain four extremes of example difficulty:

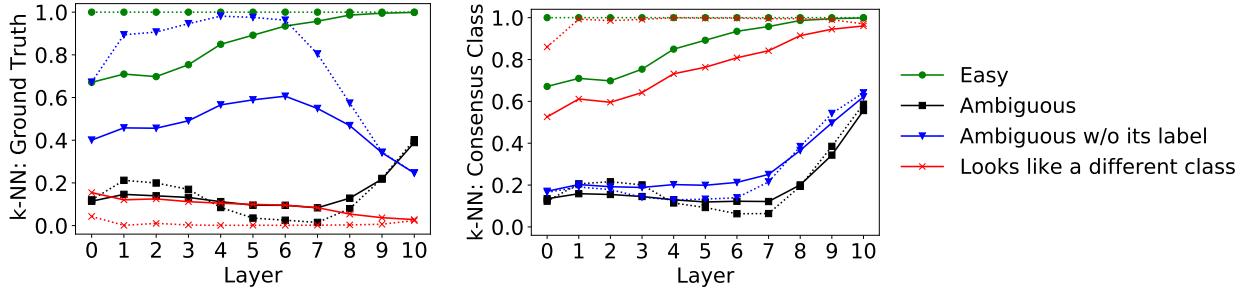


Figure 8: Average  $k$ -NN probe confidence (solid lines) and accuracy (dotted lines) for the ground truth class (left) and consensus class (right), in the validation split for examples exhibiting extreme forms of difficulty. Mean values for 100 examples with each form of difficulty, identified as the 100 examples closest to the corners in Figure 7 (left). This result is for CIFAR10 with ResNet18: similar plots for all datasets and architectures are shown in Appendix C.6. See Section 4 for the discussion of the result and how it can be used to improve prediction accuracy.

**Easy examples:** (Low  $\text{PD}_{\text{Val.}}$ , Low  $\text{PD}_{\text{Train}}$ ). Such examples are often visually typical members of their class and the predicted label nearly always matches the ground truth.

**Looks like a different class:** (Low  $\text{PD}_{\text{Val.}}$ , High  $\text{PD}_{\text{Train}}$ ). In the validation set, there is a clear (and nearly always incorrect) classification for such an input, but it is difficult to connect such inputs to other examples of their ground truth class during training. Mislabeled examples are of this kind, as are visually confusing images which at first appear to show something else.

**Ambiguous unless the label is given:** (High  $\text{PD}_{\text{Val.}}$ , Low  $\text{PD}_{\text{Train}}$ ). These examples are difficult to connect to their predicted class in the validation split but easy to connect to their ground truth class during training. These points may, for example, visually resemble both their own class and another class. They are likely to be misclassified.

**Ambiguous:** (High  $\text{PD}_{\text{Val.}}$ , High  $\text{PD}_{\text{Train}}$ ). These examples may be corrupted or show an example of a rare sub-class. Predictions for these inputs can depend strongly on the random seed used for training and initialization.

In Figure 7 we visualize CIFAR10 ‘‘Bird’’ images with the extreme forms of example difficulty for ResNet18, as identified using the prediction depth in the training and validation splits. In the full dataset (left panel) we see that the prediction depth can be very different in the training and validation splits: the two prediction depths are typically similar for points where the consensus class is equal to the ground truth (right panel), but can be very different when the consensus class is different from the ground truth (middle panel). This behavior is consistently reproduced for all datasets and architectures in Appendix C.5.

Looking at these examples of the class ‘‘Bird’’ with different difficulty types, we observe that ResNet18 finds small garden birds easiest, while birds in flight against a blue background ‘‘look like airplanes’’, ostriches are ‘‘ambiguous without their label’’ and the ‘‘ambiguous’’ examples are either unclear photographs or examples of rare sub-groups that don’t appear frequently in the data. We found the consensus-consistency of inputs that are ‘‘Ambiguous’’ or ‘‘Ambiguous without its label’’ to be significantly lower than those of examples that are ‘‘Easy’’ or ‘‘Look like a different class’’.

In order to better understand how networks process examples with different, extreme forms of example difficulty, Fig. 8 examines how the  $k$ -NN confidence (fraction of votes) and accuracy of the ground truth class and of the consensus class progress, as validation points pass through the network. ‘‘Easy’’ examples are classified as their consensus class (which is equal to their ground truth class) in all  $k$ -NN probes and the confidence in the consensus class steadily increases as data points proceed through the hidden layers. Examples that ‘‘look like a different class’’ are also processed as members of their consensus class, similarly to ‘‘easy’’ examples. However, unlike ‘‘easy’’ examples, their consensus classes do not match their ground

truth classes. Examples that are “ambiguous without their labels” are initially processed as members of their ground truth classes with intermediate confidence, but in later layers become mistaken for their consensus class. “Ambiguous” examples are processed with low confidence and accuracy in the early layers, for both ground truth and consensus classes. In later layers “ambiguous” examples are recognized, with intermediate confidence and accuracy, as members of the consensus class, which matches the ground truth class for a sizeable fraction of “ambiguous” examples.

**Improving the prediction accuracy** Can the prediction accuracy be improved using our understanding of how each class of difficult examples are processed by deep models? Figure 8 suggest that k-NN probes in intermediate layers may be more accurate than the full deep model for examples that are “ambiguous without their label” (data points closest to the lower right corner of Figure 7). In order to test this hypothesis, we compare the accuracy of the k-NN probe in layer 4 to the full model’s prediction for the 100 examples closest to the lower right corner of Figure 8. We obtain a striking improvement in accuracy from 25% to 98% for these examples. This showcases how insights from this study can be directly used to improve prediction accuracy.

## 5 Discussion

**Summary** We have introduced a notion of example difficulty called the prediction depth, which uses the processing of data inside the network to score the difficulty of an example. We have shown how the prediction depth is related to the accuracy and uncertainty of a prediction, the adversarial input margin and the output margin of the learned solution, and that data points that are easier according to the prediction depth are also typically learned earlier in training. We have also shown that the difficulty of an example can be both similar, or very different depending on whether an input appears in the validation split or the training split, and described four extremes of example difficulty. For data points that are “ambiguous without their label”, we have demonstrated how returning the k-NN prediction in a middle layer can lead to impressive increases in model accuracy: for CIFAR10 in ResNet18 we obtained an increase in accuracy from 25% to 98% for the inputs that are most “ambiguous without their label”.

**Connecting known phenomena** In the literature, the following phenomena are separately reported from different experimental paradigms:

1. Early layers generalize while later layers memorize (Stephenson et al., 2021).
2. Model layers converge from input layer towards output layer (Raghu et al., 2017; Morcos et al., 2018).
3. Deep models learn easy data (Jiang et al., 2021; Toneva et al., 2019) and simple functions first (Hu et al., 2020; Kalimeris et al., 2019).

Following this paper, a coherent and closely related picture emerges:

1. Predictions made in early layers are more likely to be consistent than those made in later layers. Consistent predictions are likely to be correct and the expected accuracy of inconsistent predictions is naturally low (Section 3.1).
2. Data points learned early in training typically have smaller prediction depths than those learned later during training (Section 3.2).
3. On average, deep neural networks exhibit wider input and output margins (common measures of “local simplicity”) in the vicinity of data with smaller prediction depths (Section 3.3).

**Pertinence of example difficulty to topics in machine learning** Curriculum Learning attempts to treat hard examples differently from easy examples during training. Robustness to distribution shifts that change the relative frequencies of common and rare subgroups in the test set (which we have shown can

have different forms of example difficulty) is important for ML Fairness. Methods developed to address heteroscedastic uncertainty typically address example difficulty as a one-dimensional quantity. We expand upon the relevance of our work to these three topics in Appendix D.

**Limitations** We believe that the results we report stem from a deep model’s representation, which is hierarchical by construction. We expect that the same results will therefore apply in larger models, larger datasets, and tasks other than image classification, but testing this remains as further work. Although we demonstrate that returning the results of a hidden k-NN can yield dramatic increases in accuracy for examples that are “ambiguous without their label”, we otherwise do not explore ways to practically apply the insights we present. In particular, we expressly do not claim that all that is required for good accuracy is to reduce the prediction depth: freezing later layers of the network would not be expected to result in good generalization.

## Acknowledgment

We would like to thank Hanie Sedghi, Ilya Tolstikhin, Ibrahim Alabdulmohsin, Daniel Keysers and Julian Eisenschlos for valuable discussions on the topic and Arthur Baldock for proofreading the manuscript.

## References

- Agarwal, C. and Hooker, S. (2020). Estimating example difficulty using variance of gradients. In *ICML, Workshop on Human Interpretability in Machine Learning (WHI)*.
- Alain, G. and Bengio, Y. (2017). Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (Workshop)*.
- Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. In *International Conference on Machine Learning*.
- Bahri, D., Jiang, H., and Gupta, M. (2020). Deep k-NN for noisy labels. In *International Conference on Machine Learning*.
- Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of International Conference on Machine Learning*.
- Carlini, N., Erlingsson, U., and Papernot, N. (2019). Distribution density, tails, and outliers in machine learning: Metrics and applications. *arXiv preprint arXiv:1910.13427*.
- Chatterjee, S. (2019). Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *International Conference on Learning Representations*.
- Cohen, G., Sapiro, G., and Giryes, R. (2018). DNN or k-NN: That is the generalize vs. memorize question. In *NeurIPS, Workshop on Integration of Deep Learning Theories*.
- Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2018). Universal transformers. In *International Conference on Learning Representations*.
- Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.
- Feldman, V. and Zhang, C. (2020). What neural networks memorize and why: Discovering the long tail via influence estimation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

- Ghorbani, B., Krishnan, S., and Xiao, Y. (2019). An investigation into neural net optimization via Hessian eigenvalue density. In *International Conference on Machine Learning*.
- Hacohen, G., Choshen, L., and Weinshall, D. (2020). Let's agree to agree: Neural networks share classification order on real datasets. In *International Conference on Machine Learning*.
- Hacohen, G. and Weinshall, D. (2019). On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Hooker, S., Courville, A., Clark, G., Dauphin, Y., and Frome, A. (2019). What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*.
- Hooker, S., Moorosi, N., Clark, G., Bengio, S., and Denton, E. (2020). Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058*.
- Hu, W., Xiao, L., Adlam, B., and Pennington, J. (2020). The surprising simplicity of the early-time learning dynamics of neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.
- Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., and Weinberger, K. (2018). Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*.
- Jiang, Y., Krishnan, D., Mobahi, H., and Bengio, S. (2018). Predicting the generalization gap in deep networks with margin distributions. In *International Conference on Learning Representations*.
- Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. (2020). Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*.
- Jiang, Z., Zhang, C., Talwar, K., and Mozer, M. C. (2021). Characterizing structural regularities of labeled data in overparameterized models. In *International Conference on Machine Learning*.
- Kalimeris, D., Kaplun, G., Nakkiran, P., Edelman, B., Yang, T., Barak, B., and Zhang, H. (2019). SGD on neural networks learns functions of increasing complexity. In *Advances in Neural Information Processing Systems*, volume 32.
- Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.
- Kendall, A. and Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.
- Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Keskar, N. S., Nocedal, J., Tang, P. T. P., Mudigere, D., and Smelyanskiy, M. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*.
- Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big transfer (BiT): General visual representation learning. In *European Conference on Computer Vision*.
- Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Technical Report*.

- Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.
- Lalor, J. P., Wu, H., Munkhdalai, T., and Yu, H. (2018). Understanding deep learning performance through an examination of test set difficulty: A psychometric case study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*.
- Liu, S., Niles-Weed, J., Razavian, N., and Fernandez-Granda, C. (2020a). Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems*, 33.
- Liu, W., Zhou, P., Wang, Z., Zhao, Z., Deng, H., and JU, Q. (2020b). FastBERT: A self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044.
- Long, P. M. and Sedghi, H. (2019). Generalization bounds for deep convolutional neural networks. In *International Conference on Learning Representations*.
- Mangalam, K. and Prabhu, V. (2019). Do deep neural networks learn shallow learnable examples first? In *ICML, Workshop on Identifying and Understanding Deep Learning Phenomena*.
- Morcos, A. S., Raghu, M., and Bengio, S. (2018). Insights on representational similarity in neural networks with canonical correlation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*.
- Nagarajan, V., Andreassen, A., and Neyshabur, B. (2021). Understanding the failure modes of out-of-distribution generalization. In *International Conference on Learning Representations*.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. *Technical Report*.
- Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.
- Papernot, N. and McDaniel, P. (2018). Deep k-Nearest Neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*.
- Pennington, J. and Bahri, Y. (2017). Geometry of neural network loss surfaces via random matrix theory. In *International Conference on Machine Learning*.
- Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. (2017). SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.
- Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. (2019). Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning*.
- Sagun, L., Bottou, L., and LeCun, Y. (2016). Eigenvalues of the Hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*.
- Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. (2018). Empirical analysis of the Hessian of over-parametrized neural networks. In *International Conference on Learning Representations (Workshop)*.
- Sanger, T. D. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE transactions on Robotics and Automation*, 10(3):323–333.

- Schwartz, R., Stanovsky, G., Swayamdipta, S., Dodge, J., and Smith, N. A. (2020). The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.
- Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.
- Smith, S. L., Dherin, B., Barrett, D. G., and De, S. (2021). On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*.
- Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. (2018). Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*.
- Smith, S. L. and Le, Q. V. (2018). A Bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*.
- Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. (2018). The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878.
- Stephan, M., Hoffman, M. D., Blei, D. M., et al. (2017). Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35.
- Stephenson, C., suchismita padhy, Ganesh, A., Hui, Y., Tang, H., and Chung, S. (2021). On the geometry of generalization and memorization in deep neural networks. In *International Conference on Learning Representations*.
- Teerapittayanon, S., McDanel, B., and Kung, H.-T. (2016). BranchyNet: Fast inference via early exiting from deep neural networks. In *International Conference on Pattern Recognition*.
- Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., and Gordon, G. J. (2019). An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.
- Unterthiner, T., Keysers, D., Gelly, S., Bousquet, O., and Tolstikhin, I. (2020). Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*.
- Wen, Y., Tran, D., and Ba, J. (2019). BatchEnsemble: An alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*.
- Wenzel, F., Snoek, J., Tran, D., and Jenatton, R. (2020). Hyperparameter ensembles for robustness and uncertainty quantification. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.
- Wu, X., Dyer, E., and Neyshabur, B. (2021). When do curricula work? In *International Conference on Learning Representations*.
- Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.
- Xin, J., Tang, R., Lee, J., Yu, Y., and Lin, J. (2020). DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.
- Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. (2020). PyHessian: Neural networks through the lens of the Hessian. In *International Conference on Machine Learning (Workshop)*.
- Zielinski, P., Krishnan, S., and Chatterjee, S. (2020). Weak and strong gradient directions: Explaining memorization, generalization, and hardness of examples at scale. *arXiv preprint arXiv:2003.07422*.

## A Detailed Description of Experiments, Architectures and Hyper-parameter Optimization

For each combination of dataset (CIFAR10, CIFAR100, Fashion MNIST, SVHN) and architecture (ResNet18, VGG16, MLP) we train 250 models with a 10% validation split selected at random each time, and an additional 25 models on the full training set.

### A.1 Datasets

CIFAR10 / CIFAR100:

Reference: ([Krizhevsky et al., 2009](#)). License: MIT.

URL: <https://www.cs.toronto.edu/~kriz/cifar.html>

Fashion MNIST:

Reference: ([Xiao et al., 2017](#)). License: MIT.

URL: <https://github.com/zalandoresearch/fashion-mnist>

Street View House Numbers:

Reference: ([Netzer et al., 2011](#)). License: CC0.

URLs: <http://ufldl.stanford.edu/housenumbers/>

<https://www.kaggle.com/stanforddu/street-view-house-numbers>

### A.2 Architectures

#### A.2.1 ResNet18

We implemented the standard ResNet18 architecture for CIFAR10 ([He et al., 2016](#)), except that we replaced Batch Norm with Group Norm and applied Weight Standardization, following recent state of the art ([Kolesnikov et al., 2020](#)).

#### A.2.2 VGG16

We used VGG16 ([Simonyan and Zisserman, 2015](#)), except that we removed the final three dense layers: a standard modification for datasets smaller than ImageNet. We also did not use batch-norm or dropout: our focus is on understanding trends in example difficulty and we do not expect the results to be dependent on these devices.

#### A.2.3 MLP

Our MLP architecture comprises seven hidden layers with ReLU activations. We chose seven layers after performing the experiments shown in Figure A.9. There we show the accuracies of k-NN probes placed after each operation of two MLP architectures, depths 15 layers and 7 layers, both of width 2048. We used CIFAR10 with 40% fixed random label noise as a reasonably difficult model classification task, to choose the depth.

#### A.2.4 Data augmentation

We did not apply data augmentation: different data augmentation schemes could be expected to have disparate effects on different examples, but we do not expect them to change the overall phenomena that we report here. We leave the use of data augmentation to subsequent studies.

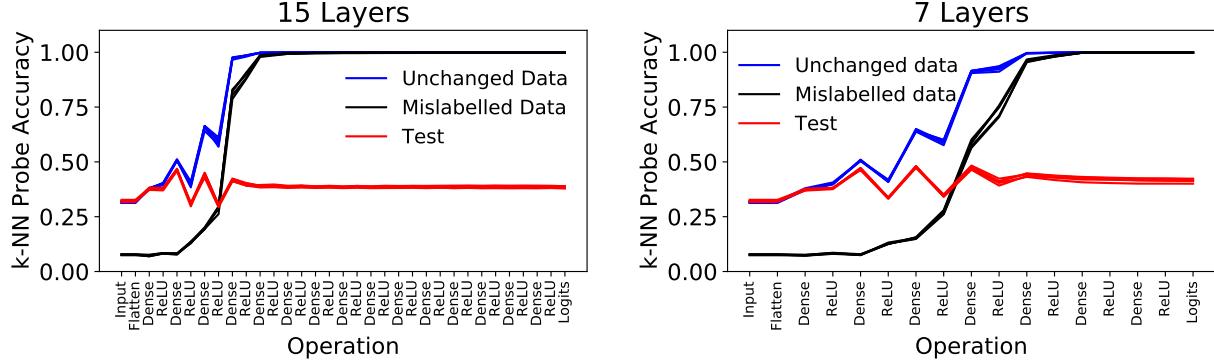


Figure A.9: Seven layers are sufficient in MLP for CIFAR10 with 40% random label noise. CIFAR10 with 40% random label noise. For this plot, k-NN probes were placed after every operation in two MLP architectures of the same width (2048) but different depths. **Left:** 15 dense layers; **Right:** 7 dense layers. Separate accuracies are reported for the test split, those data points in the training split with unchanged labels and the randomly mislabeled data in the training split.

### A.3 Hyperparameter optimization

For each architecture and dataset we initially performed  $10^4$  steps of SGD with momentum, using all combinations of the following hyperparameters: learning rate  $\in [4 \times 10^1, 1 \times 10^{-1}, 4 \times 10^{-2}, 1 \times 10^{-2}, 4 \times 10^{-3}, 1 \times 10^{-3}, 4 \times 10^{-4}, 1 \times 10^{-4}]$ ; momentum  $\in [0.0, 0.5, 0.9, 0.95]$ ; weight decay  $\in [0, 5 \times 10^{-4}]$ . In CIFAR10, we additionally considered a learning rate of  $2 \times 10^{-2}$ . For each dataset and architecture we selected the 7 most accurate and stable training curves, extended the number of training steps and added a learning rate schedule, reducing the learning rate in steps of  $\frac{1}{5}$ . At least two rounds of optimization were performed to adapt the learning rate schedule for each combination of architecture and dataset. In each case a mini-batch size of 256 was used. The final parameters obtained are shown in Table 1, which also gives the hyperparameters used in Sec. 3.3 and Appendix A.2.3 for CIFAR10 with 40% label noise. Final accuracies of the trained models are given in Table 2.

### A.4 Convergence and consistency of k-NN probe accuracies

We tested the convergence of  $k$  in k-NN for VGG16 on CIFAR10. Figure A.10 shows the accuracies of k-NN probes after every operation of the network for  $k \in [3, 10, 30]$ . We see that these k-NN probe accuracies are insensitive to  $k$  for  $k = 30$ .

Figure A.9 shows separate results for five independent training runs. Similarly, Figure 5 (right) and Figure 6 (right) each show the mean and uncertainty on the k-NN probe accuracies from 5 independent runs. The spread of results in these figures is tight, demonstrating consistency of the results.

### A.5 Placement of k-NN probes

For prediction depth, in MLP we constructed k-NN probes after the dense operations and the softmax, in VGG16 after the convolutions and softmax, and in ResNet18 we constructed the probes after the initial Group Norm operation, the sum operations at the end of each block and after the softmax operation.

From figures A.9 and A.10 it is clear that there are upper and lower envelopes that bound the k-NN probe accuracies: the lower envelope corresponds to the ReLU activations and the upper envelope to the operations immediately preceding them. We chose the preceding operations which, in effect, conceptually shifts the

	Learning Rate	Momentum	Weight Decay	Schedule / steps
SVHN				
ResNet18	$4 \times 10^{-2}$	0.95	0.0	[7000]
VGG16	$4 \times 10^{-2}$	0.9	0.0	[3000, 6000, 1000]
MLP	$4 \times 10^{-2}$	0.9	0.0	[2500, 5500, 2000]
Fashion MNIST				
ResNet18	$1 \times 10^{-2}$	0.95	0.0	[4000, 3000]
VGG16	$1 \times 10^{-2}$	0.95	0.0	[3000, 6000, 1000]
MLP	$4 \times 10^{-2}$	0.5	0.0	[10000, 2500]
CIFAR10				
ResNet18	$4 \times 10^{-2}$	0.95	0.0	[7000]
VGG16	$4 \times 10^{-2}$	0.9	0.0	[5000, 1000]
MLP	$2 \times 10^{-2}$	0.9	0.0	[5000, 1250, 1000]
CIFAR10 w/ 40% (Fixed) Randomized Labels				
VGG16	$4 \times 10^{-2}$	0.9	0.0	[5000, 10000]
MLP	$2 \times 10^{-2}$	0.9	0.0	[12000, 1250, 4000]
CIFAR100				
ResNet18	$1 \times 10^{-1}$	0.95	0.0	[6000]
VGG16	$4 \times 10^{-2}$	0.9	0.0	[2500, 7500]
MLP	$1 \times 10^{-1}$	0.95	0.0	[2500, 6000, 1500]

Table 1: Training parameters for each model and dataset.

	SVHN	Fashion MNIST	CIFAR10	CIFAR100
ResNet18	95%	93%	83%	56%
VGG16	95%	93%	83%	45%
MLP	85%	90%	59%	29%

Table 2: Final accuracies of the trained models.

ReLU activations to the “start” of a layer rather than the “end” of the preceding layer.

## A.6 Notes on definitions

### A.6.1 Consistency of the model’s prediction with the k-NN probe after the softmax layer

Deep classifier models are trained to create linear separation of the classes in the softmax layer. There is nearly perfect agreement between the k-NN probe after the softmax layer and predictions of the full model. In the rare case that the k-NN probe after the softmax predicts a different class from the full network we do not assign a prediction depth. Such data points are extremely rare: we found zero such data points in the large majority of models and always fewer than  $1 \times 10^4$ .

### A.6.2 Tiebreaks in the consensus class

When obtaining the consensus class, if predictions are tied between more than one class and the ground truth is in the tiebreak, then we break the tie in favor of the ground truth class. If the ground truth is not in the tiebreak then we report the tied class with the lowest integer index. This choice was motivated by ease of implementation. We are confident that the overall results we report are unaffected by this choice.

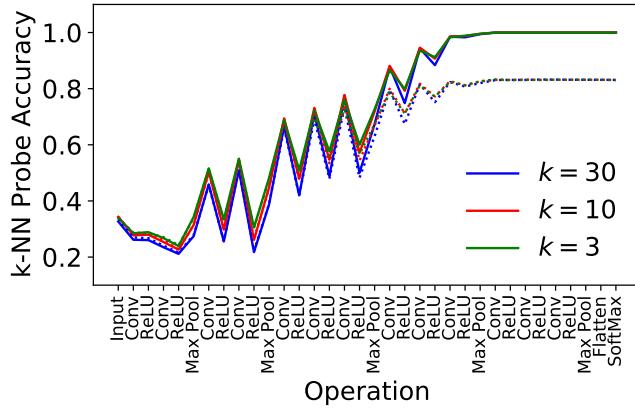


Figure A.10: CIFAR10, VGG16.  $k$ -NN probe accuracies after each operation for  $k \in [3, 10, 30]$ . Solid lines: training set. Dotted lines: test set. Differences in these results are comparable to the scatter observed for networks trained with different random seeds at  $k = 30$ .

### A.6.3 Estimating the consensus-consistency

We used the same ensemble to obtain both the consensus class  $\hat{y}_A(x)$  and the consensus-consistency score. Thus we are reporting relationships between observables for a given ensemble. This is a biased estimator of (2): an unbiased estimator could have been constructed by training an additional set of models to obtain the consensus class, but at greater cost. We are confident that this does not affect the conclusions of this study.

## A.7 Justification and hyperparameters for the output margin intervention

A number of published works informed the design of our intervention. Firstly, Soudry et al. (2018) demonstrate that the cross-entropy (CE) loss leads to large margins. In contrast to the cross-entropy, the 0-Hinge loss has zero gradient if the prediction is correct, so it does not push the model to become arbitrarily confident. Secondly, Keskar et al. (2017) show that smaller batch sizes lead to the discovery of flatter minima, which also corresponds to a wider margin (Neyshabur et al., 2017). Thirdly, Keskar et al. (2017), Smith and Le (2018) and Smith et al. (2018) show that the gradient noise level in stochastic gradient descent is proportional to  $\frac{\text{Learning Rate}}{\text{Batch Size}}$ . Having an appreciable noise level early in training plays an important role in finding the flatter minima with larger output margins reported in Keskar et al. (2017). Our intervention to minimize the margin therefore combines both of the following changes:

1. Changing the loss from cross-entropy to the 0-Hinge loss
2. Minimizing the learning rate and making the batch size as large as possible

To test whether both or only one of these changes is required to obtain small output margins, we performed separate runs, without any intervention, applying the changes individually and applying them together. The starting point (the control) is training with cross-entropy loss and SGD with momentum and large initial learning rate.

We trained VGG16 on CIFAR10. The hyperparameters, presented in Table 3, were set for each loss, to obtain nearly smooth learning curves for full-batch gradient descent and very noisy learning curves for SGD. In Figure A.11 we show the learning curves for these models. Since full-batch gradients are expensive to compute we restricted the experiments to separating two classes (“Horse” and “Deer”) with 4096 training images in total (evenly split).

Name	Batch Size	Initial Learning Rate	Schedule / Steps	Momentum
CE, SGD	256	$4 \times 10^{-3}$	[3200]	0.9
CE, GD	4096	$6.4 \times 10^{-6}$	$[1.2 \times 10^6]$	0.9
0-Hinge, SGD	256	$4 \times 10^{-2}$	[5000, 2500]	0.9
0-Hinge, GD	4096	$6.4 \times 10^{-5}$	$[8 \times 10^5]$	0.95

Table 3: Hyperparameters for all combinations of CE vs. 0-Hinge loss and SGD with momentum and large initial learning rate vs. GD with momentum and small learning rate. In the learning rate schedules we reduced the learning rate by a factor of  $\frac{1}{5}$  for each new set of training steps. Weight decay was not employed in these calculations since we do not expect typical, modest amounts of weight decay to qualitatively affect the results.

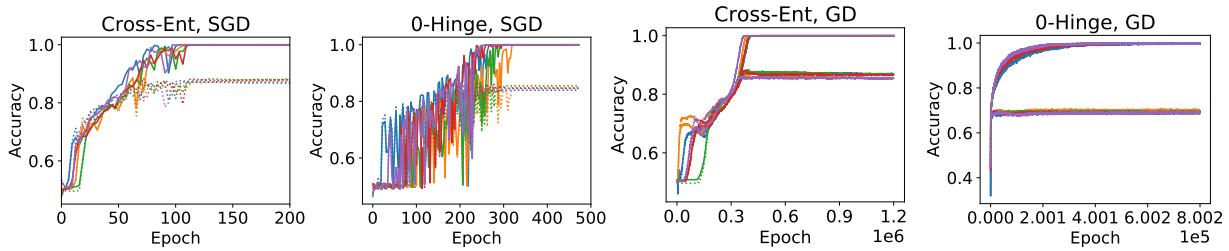


Figure A.11: Training curves for Cross-Entropy and 0-Hinge Losses, with either SGD with momentum and large initial learning rate, or GD with momentum and a small learning rate. The initial learning rates and schedules are set to obtain nearly smooth learning curves for GD and noisy learning curves for SGD. Each plot shows five separate learning curves. Solid lines show training accuracies and dotted lines show test accuracies.

Table 4 lists the mean accuracy and output margin for all four combinations of loss function and optimizer. We can see that the combination of both changes yields the smallest mean output margin,  $10^2$  times smaller than the next smallest margin. Figure A.12 presents the k-NN probe accuracies in the hidden layers for all four combinations of loss and optimizer. The combined intervention, which has the smallest margin, leads to the data being accurately clustered in the very latest layers.

Name	Mean Accuracy	Mean Output Margin
CE, SGD	87.6%	$1.6 \times 10^{-1}$
CE, GD	86.7%	$1.1 \times 10^{-1}$
0-Hinge, SGD	83.9%	$6 \times 10^{-2}$
0-Hinge, GD	69.5%	$2.0 \times 10^{-4}$

Table 4: Mean accuracy and output margin for CE vs. 0-Hinge losses and SGD with momentum and large initial learning rate vs. GD with momentum and small learning rate.

## B Further Related Work

Previous studies of deep learning on the level of individual data points have: sought to explain its accuracy by focusing on the interference of per-example gradients during training (Chatterjee, 2019; Zielinski et al., 2020); improved our understanding of deep learning by studying its performance on datasets with partially randomized labels, which corresponds to a specific binary partitioning of example difficulty Arpit et al. (2017); quantified example difficulty using 5 different observables: 1) the change in a network’s output for elements of the training set after subsequent fine-tuning on a disjoint dataset, 2) the adversarial input margin of an

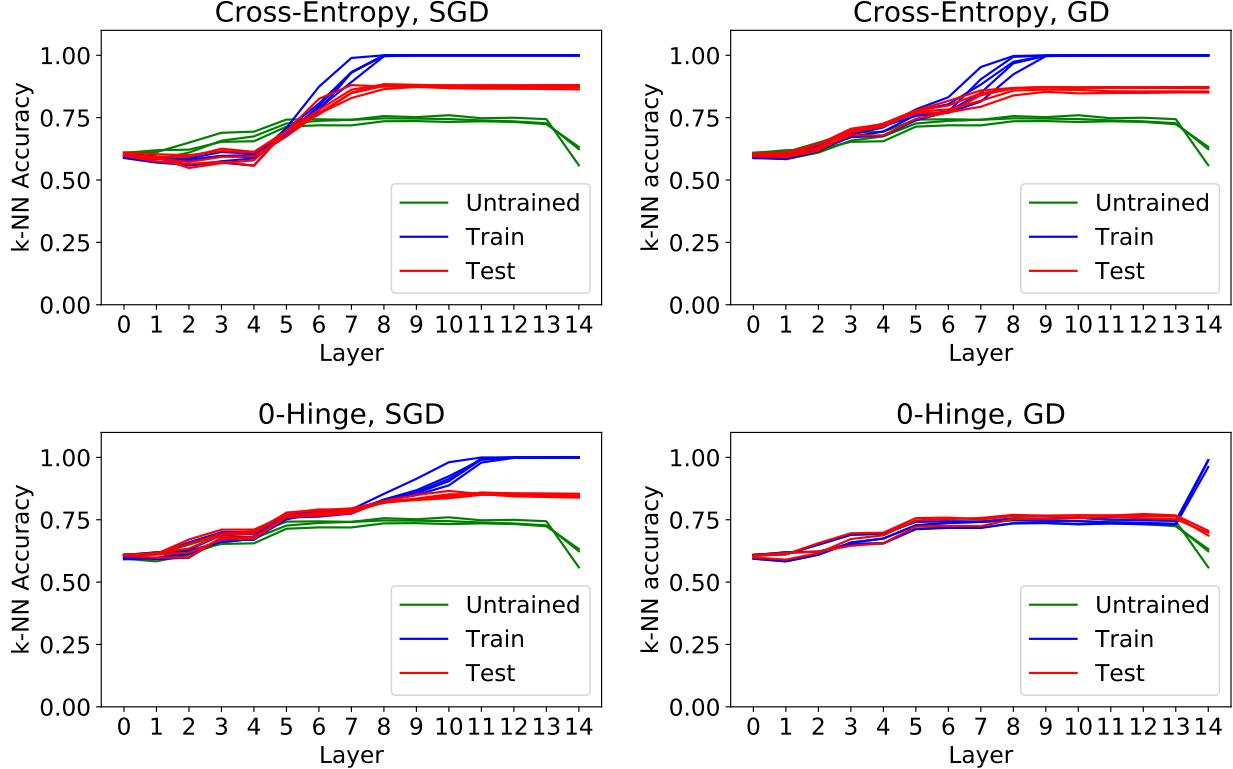


Figure A.12: *Accuracies of  $k$ -NN probes in the hidden layers of VGG16, resulting from each combination of Cross-Entropy vs. 0-Hinge loss and SGD with momentum and large initial learning rate vs. GD with momentum and small learning rate.* In each case we compare to the probes for untrained (freshly initialized) networks. Only the 0-Hinge with gradient descent using momentum and small learning rate (“0-Hinge, GD”) leads to clustering in the latest layers.

example, 3) the agreement of models in an ensemble, 4) the average confidence of models in an ensemble, and 5) the disparate impact of differential privacy [Carlini et al. \(2019\)](#); identified difficult examples with those disproportionately impacted by pruning and compression [Hooker et al. \(2019\)](#), with those whose classifications are more often forgotten during training [Toneva et al. \(2019\)](#), and with those that are least likely to be correctly classified in the validation set [Jiang et al. \(2021\)](#); demonstrated a correspondence between those examples that a human finds difficult and examples a machine finds difficult [Lalor et al. \(2018\)](#). In contrast to these works, we study the computational difficulty of inferring the class of an input: the amount of computation used to connect that input with its class label inside the network. Our definition of example difficulty is precisely described in Section 2.

In [Hacohen et al. \(2020\)](#) the authors report that the order during training in which data points are learned is common between different architectures and random seeds in deep learning. In light of the correlation between prediction depth and the order of learning data points (as reported in Section 3.2), their result reflects the sanity checks performed in Section 2.2: that prediction depth is consistent between architectures and random seeds.

Distinct from the forms of example difficulty we describe in Section 4, [Hooker et al. \(2019\)](#) propose four different forms of example difficulty: “ground truth label incorrect or inadequate”, “multiple-object image”, “corrupted image”, “fine-grained classification”. The forms of difficulty we describe in this paper follow directly from the computational difficulty of the examples, derived from the model’s behavior. In contrast, [Hooker](#)

et al. (2019) employ intuitive notions of difficulty to define their four forms and ask humans to assign difficult examples to these categories.

The Deep k-Nearest Neighbors method Papernot and McDaniel (2018) builds a series of k-NN probes in the hidden spaces of the network. When a test example is processed by the network, Deep k-NN identifies the nearest neighbors of the example in every layer, and then classifies the example according to the class labels of the aggregated nearest neighbors. By comparing the number of neighbors the example has of the predicted class to the number of similarly labeled nearest neighbors that were recorded (across all layers) for examples in a hold-out test set, Deep k-NN is able to quantify the probability that the prediction is correct and to identify OOD examples. However, the authors do not report the phenomena reported here. Our results may yet enable the development of new Deep k-NN methods. Another algorithm Bahri et al. (2020) constructs a k-NN probe in the logit space of a network, and demonstrates that this enables improved detection of mislabeled data.

## C Consistency of the Main Results Reported in the Paper

### C.1 Consistency of prediction depth between architectures

To visually reinforce the correlations reported in Figure 2 (right), Figures C.13 to C.16 reproduce the result from Figure 2 (right) for all datasets in both the training and validation splits. For each combination of dataset and architecture we trained 250 models with random 90:10% training:validation splits as described in Appendix A. These histograms compare the mean prediction depths of the data points between different architectures. Separate plots are shown for the training and validation splits. In each case we've rescaled prediction depth to the interval [0, 1] for visual ease of comparison between datasets. Each histogram is accompanied by the corresponding Spearman's Correlation Coefficient.

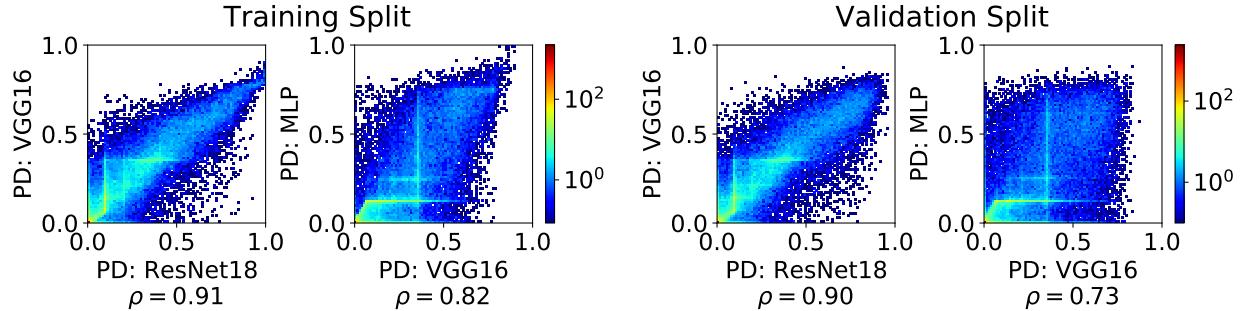


Figure C.13: *Consistency of prediction depth between architectures for SVHN*. Histograms comparing the mean value of prediction depth obtained for each data point, across the ensemble of trained models. Left pair: training split. Right pair: validation split. Spearman's Correlation Coefficient is given beneath each plot. See Appendix C.1 for details.

### C.2 Relationship between prediction depth and prediction consistency

Figures C.17 and C.18 reproduce the results of Figure 3 and Figure 4 (left) for every dataset and architecture. The gradients of the linear bounds reported in the paper depend on the difficulty of the classification task: easier tasks are solved after fewer layers.

Figure C.19 reproduces Figure 4 (middle) for every dataset and architecture. Similarly, Figure C.20 reproduces Figure 4 (right) for all datasets and architectures. Related to Figure C.19, in Figure C.21 we show that the prediction depth in one model can be used to estimate the prediction entropy of an ensemble

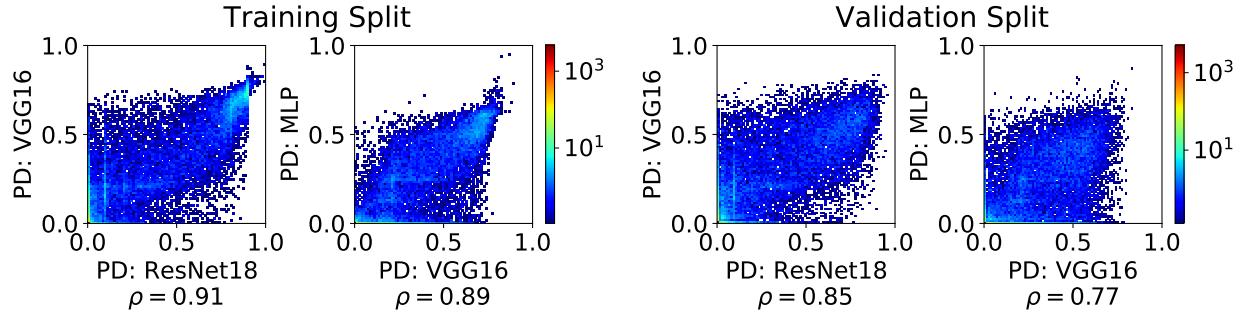


Figure C.14: *Consistency of prediction depth between architectures for Fashion MNIST.* Histograms comparing the mean value of prediction depth obtained for each data point, across the ensemble of trained models. Left pair: training split. Right pair: validation split. Spearman’s Correlation Coefficient is given beneath each plot. See Appendix C.1 for details.

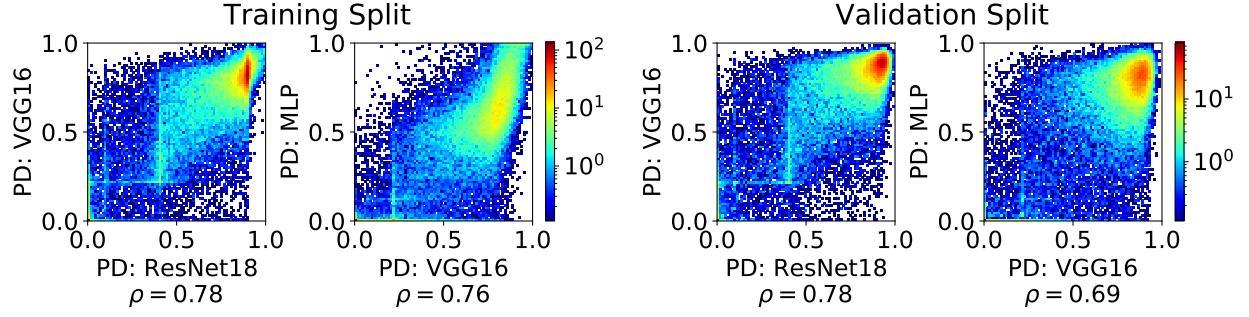


Figure C.15: *Consistency of prediction depth between architectures for CIFAR100.* Histograms comparing the mean value of prediction depth obtained for each data point, across the ensemble of trained models. Left pair: training split. Right pair: validation split. Spearman’s Correlation Coefficient is given beneath each plot. See Appendix C.1 for details.

of models, where members of the ensemble have the same architecture and are trained using the same hyperparameters but with different random seeds.

**Prediction entropy:** The entropy of predictions in an ensemble for an unseen input  $x$ . Consider an ensemble of models trained on  $r$  random subsets of the complete dataset  $\tilde{\mathcal{S}} \sim \mathcal{S} \setminus \{(x, y)\}$  (which explicitly do not include  $(x, y)$ ). We obtain the normalized histogram of the one-hot predictions of this ensemble for the input  $x$ . The prediction entropy is the entropy of that histogram. For  $N$  classes the entropy of the prediction histogram is given by

$$S(x) = - \sum_{i=1}^N p_i(x) \log p_i(x) \quad (3)$$

where  $p_i(x)$  represents the fraction of models that predicted the class  $i$  for input  $x$ .

Figure C.22 shows the histogram of average prediction depth (validation set) vs. prediction entropy for each dataset and architecture. We remark that the mean prediction depth defines a linear upper bound on the prediction entropy similar to the corresponding linear lower bound on the consensus-consistency score (Figures C.17 and C.18).

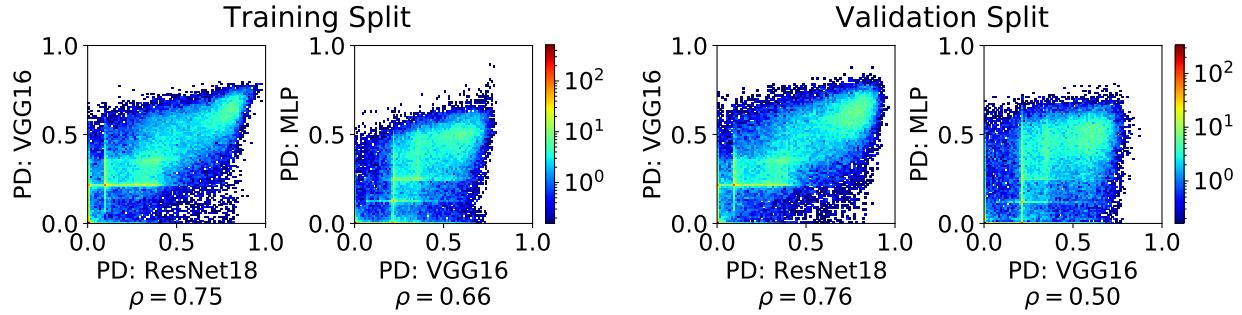


Figure C.16: *Consistency of prediction depth between architectures for CIFAR10.* Histograms comparing the mean value of prediction depth obtained for each data point, across the ensemble of trained models. Left pair: training split. Right pair: validation split. Spearman’s Correlation Coefficient is given beneath each plot. See Appendix C.1 for details.

### C.3 Comparison of prediction depth and iteration learned

Figure C.23 reproduces the result shown in Figure 5 (left) for every architecture and dataset. To give a more complete picture of the relationship between the prediction depth and the iteration learned, Figures C.24 to C.27 show histograms of the mean prediction depth and iteration learned for each data point when it occurs in both the training and validation splits. As described in Appendix A, for each dataset and architecture we trained 250 models with random 90:10% validation:train splits. Each time a data point appears in either split we record the prediction depth and the iteration learned. These histograms compare the mean prediction depth to the mean iteration learned for all data points in both the train and validation splits. The Spearman’s Correlation Coefficient is given beneath each plot.

### C.4 Consistency of margin results

Figures C.28 to C.31 reproduce Figure 6 (left and middle) for all datasets and architectures in both the training and test splits.

### C.5 Consistent two-dimensional relationship between prediction depths in the training and validation splits

Figures C.32 to C.35 demonstrate consistency of the histograms shown in Figure 7 for all datasets and architectures. As described in Appendix A, for each dataset and architecture we trained 250 models with random 90:10% validation:train splits. Each time a data point appears in either split we record the prediction depth. These histograms compare the mean prediction depths in the two splits for all data points which can be very different from each other, depending on whether the consensus class matches or differs from the ground truth class.

### C.6 Evolution of clustering in the hidden layers for the different forms of example difficulty

Figures C.36 to C.47 reproduce similar behavior to that shown in Figure 8 for all datasets and architectures. Please see Figure 8 for a detailed description.

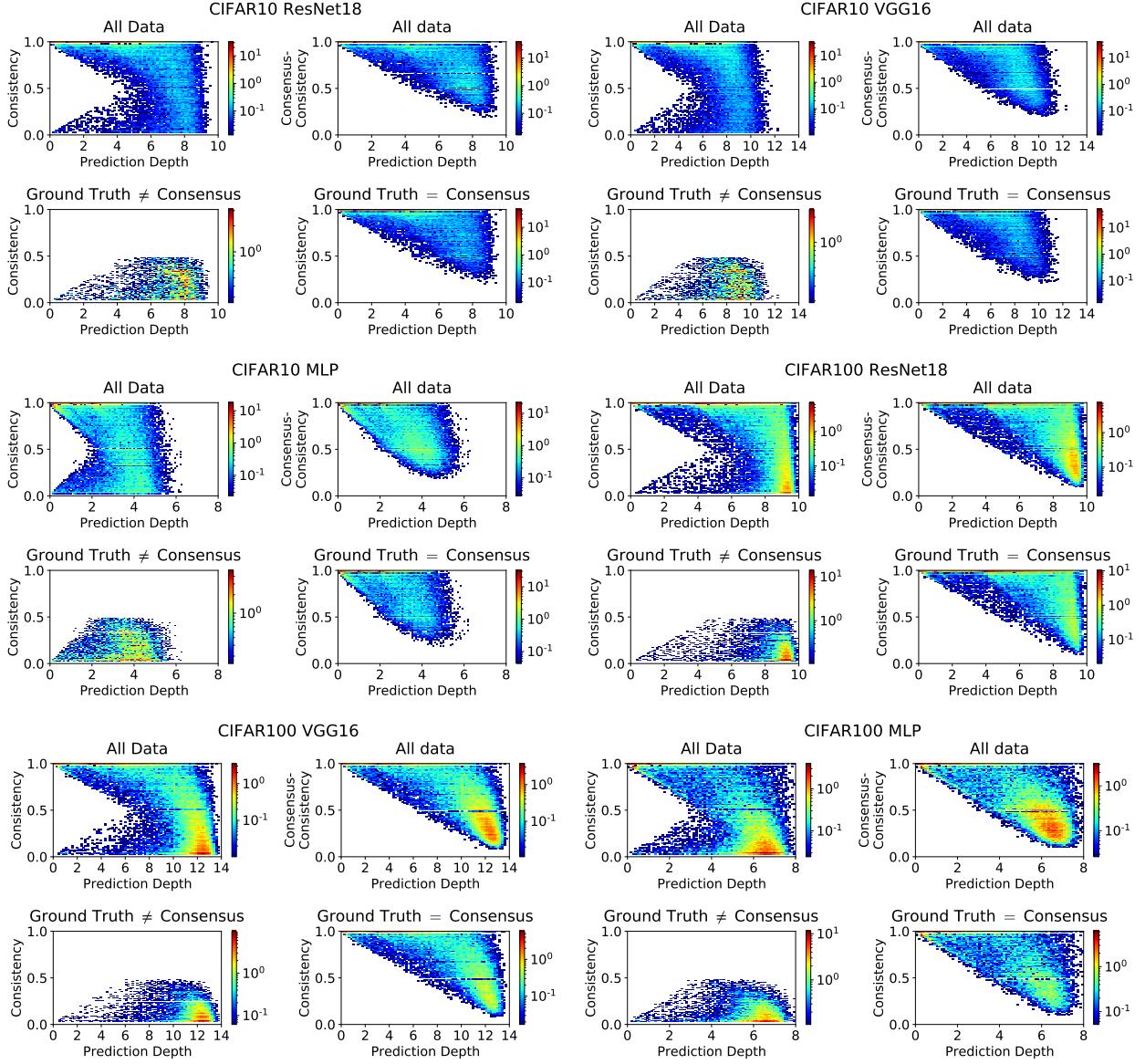


Figure C.17: This figure demonstrates the consistency of the behavior shown in Figure 3 and Figure 4 (left) for all architectures with CIFAR10 and CIFAR100.

## D Pertinence of example difficulty to topics in machine learning

We will describe the relevance of our work to distribution shift and robustness; algorithmic fairness, curriculum learning and models that explicitly address heteroscedastic uncertainty.

**Distribution Shift and Robustness:** Recent work has hypothesized that the linear relationship between the performance of a model before and after distribution shift could potentially be explained in a theory based on the difficulty of examples (Recht et al., 2019). Recent work has additionally discussed how examples that belong to a minority group might appear difficult to classify correctly under distribution shift (Nagarajan et al., 2021). Therefore it seems natural to suppose that the richer picture of example

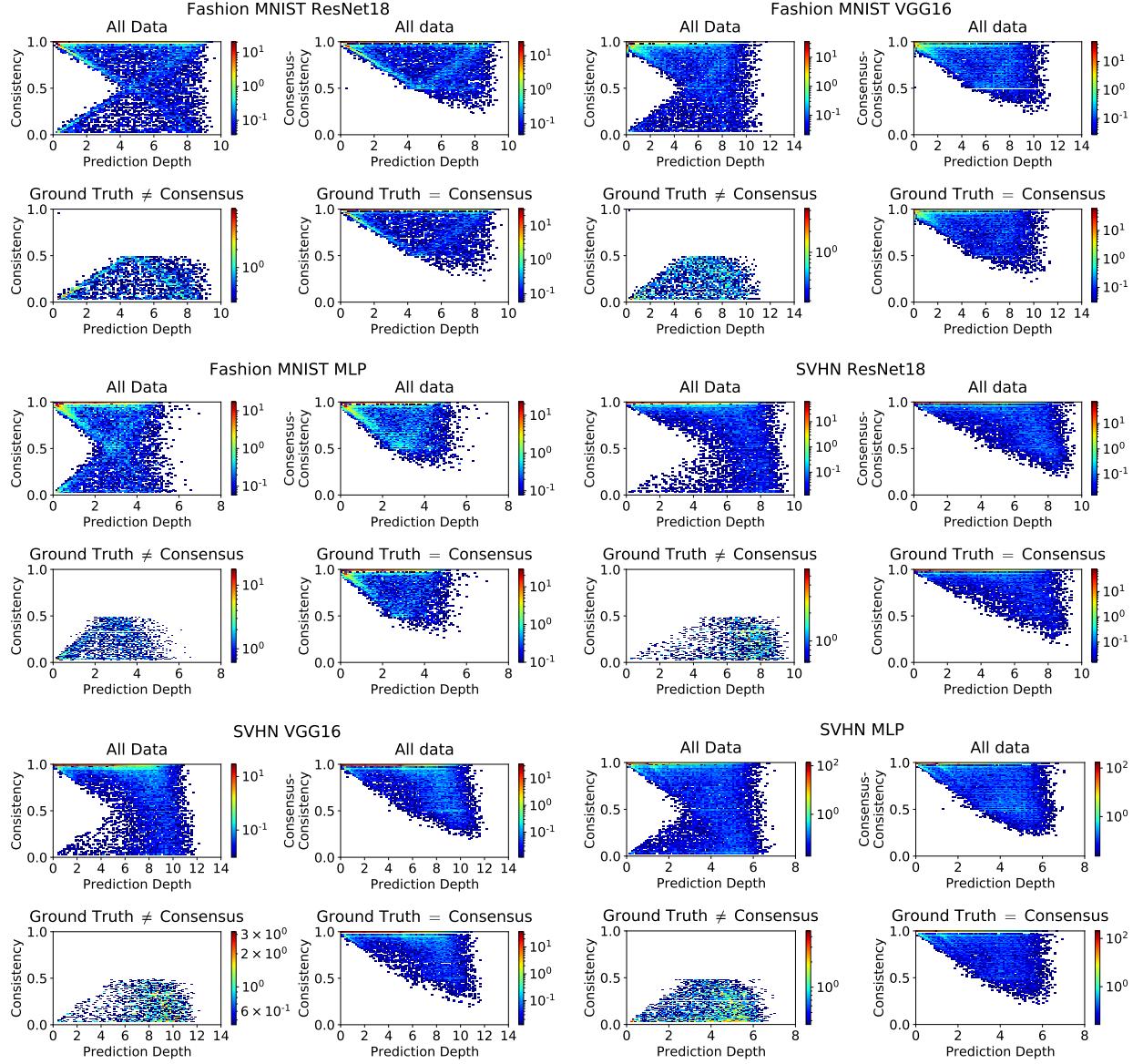


Figure C.18: This figure demonstrates the consistency of the behavior shown in Figure 3 and Figure 4 (left) for all architectures with Fashion MNIST and SVHN.

difficulty we introduce could lead to a deeper understanding of distribution shift and aid with the development of more robust algorithms.

**Curriculum Learning:** This class of training algorithms exploits additional information about a dataset (obtained in advance) to present easier examples earlier in the training process (Elman, 1993; Sanger, 1994; Bengio et al., 2009). Different notions of difficulty have been the subject of several related studies (Bengio et al., 2009; Toneva et al., 2019; Hacohen and Weinshall, 2019) and it has been shown that (neglecting the cost of obtaining the curriculum) following a curriculum can improve training time significantly, particularly for large training data (Wu et al., 2021). We envisage that richer, more effective curricula could be designed by distinguishing different forms of example difficulty. This could,

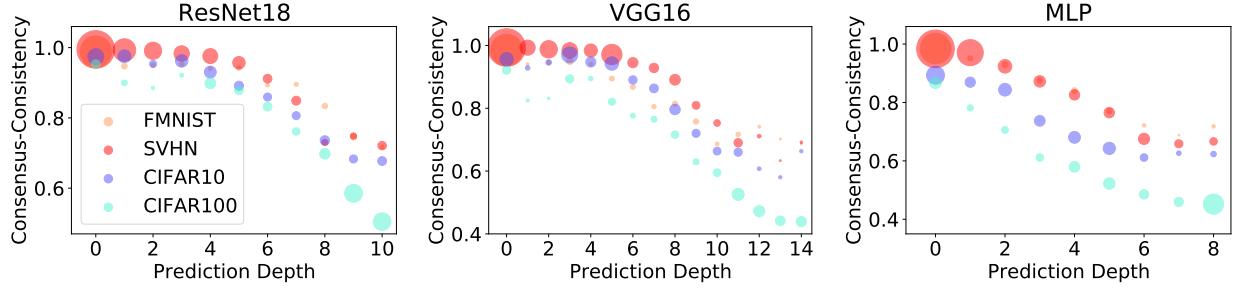


Figure C.19: This figure demonstrates the consistency of the result shown in Figure 4 (middle) for all datasets and architectures.

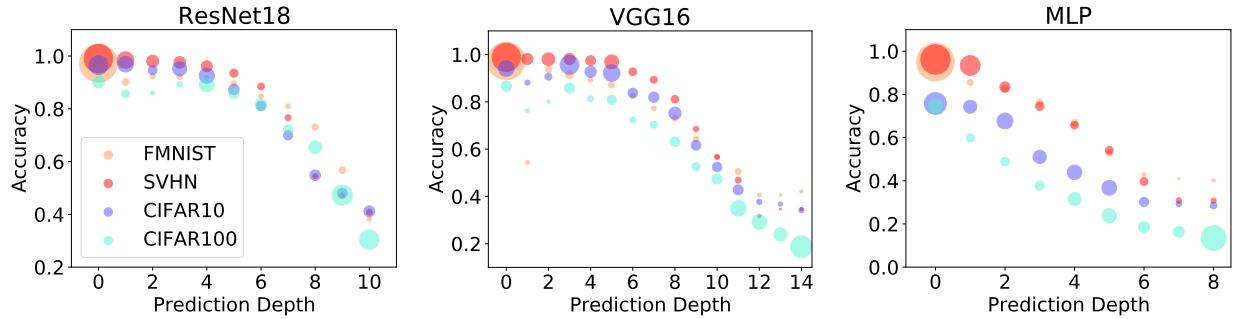


Figure C.20: This figure demonstrates the consistency of the result shown in Figure 4 (right) for all datasets and architectures.

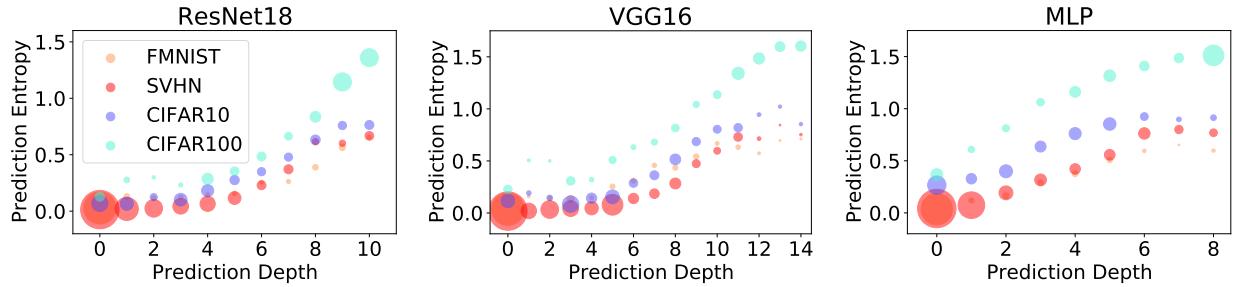


Figure C.21: *The prediction depth in one model can be used to estimate the prediction entropy of an ensemble.* The size of the marker indicates the fraction of data points with each prediction depth. We trained 25 models on each dataset and architecture with different random seeds. We take the prediction depth from one trained model and report the average prediction entropy of the corresponding data points, where the prediction entropy is determined from the remaining 24 models. As in Figure C.19, predictions for data points with smaller prediction depths have lower mean entropy (are more consistent) than those of data points with larger prediction depths.

for example, be achieved setting the curriculum according to a each data point's location in Figure 7.

**Algorithmic Fairness:** We have seen that mislabeled data is processed similarly to data that simply looks

mislabeled to the algorithm (both “look like a different class”). This presents a fairness challenge when filtering “noisy labels”. Similarly, we have seen that examples of rare subgroups (which are essential to include in the training set for robustness (Feldman and Zhang, 2020) and fairness (Hooker et al., 2020) are processed similarly to truly “ambiguous” inputs. Finding ways to deal with “label noise” without biasing against these subgroups remains an open challenge. In further work, we anticipate that examining datasets in an enlarged space of different example difficulty measures (Jiang et al., 2021; Toneva et al., 2019; Carlini et al., 2019; Hooker et al., 2019; Lalor et al., 2018; Agarwal and Hooker, 2020) may allow algorithms that distinguish between these different sources of label noise to reach higher accuracy and to be fairer.

**Heteroscedastic Uncertainty:** There are a class of models with two heads, one to model the mean and the other the uncertainty of the prediction (E.g. Kendall and Gal (2017); Kendall et al. (2018)). These models learn to become uncertain on difficult inputs and treat example difficulty as a one-dimensional quantity. It seems highly likely that this uncertainty will lead to the model down-weighting examples of rare subgroups in the data. We suggest that methods for modeling uncertainty could additionally be tasked with estimating the location of a training point in Figure 7. It seems plausible to suppose that new models able to distinguish the form of an example’s difficulty could later be refined to be fairer, more accurate and better calibrated.

## E Alternative Definitions for Prediction Depth

Instead of using the network’s final prediction on a data point to assign the prediction depth, one could instead use the ground truth label. This would require a different rule for assigning a prediction depth to validation data points that are incorrectly classified as compared to data points that are correctly classified. We consider our definition to be simpler than combining two separate rules.

One could alternatively have defined the prediction depth for each example by first leaving it out of the training set, and then training networks of different depths to identify the number of layers required to classify it correctly. In fact, architectures of different depths have different inductive biases, so the relative difficulty of inputs can become inverted with changing depth (Mangalam and Prabhu, 2019). Such an approach would be expensive but could lead to a rich picture of how example difficulty changes with architecture.

Another potential approach would have been to use a linear classifier such as Logistic Regression in the embedding spaces. Indeed linear probes, logistic regression and SVM probes have been previously applied to the hidden spaces of DNNs (E.g. Cohen et al. (2018); Alain and Bengio (2017)).

Figure E.48 compares the behavior of k-NN probes and Logistic Regression (LR) probes after the convolution operations of VGG16 with CIFAR10. LR is able to completely separate the training set after the first convolution operation. We also show the behavior when training LR on a random 50% of the dataset and predicting on the other half. k-NN shows lower accuracy until the classes become entirely clustered. We chose k-NN probes for this investigation.

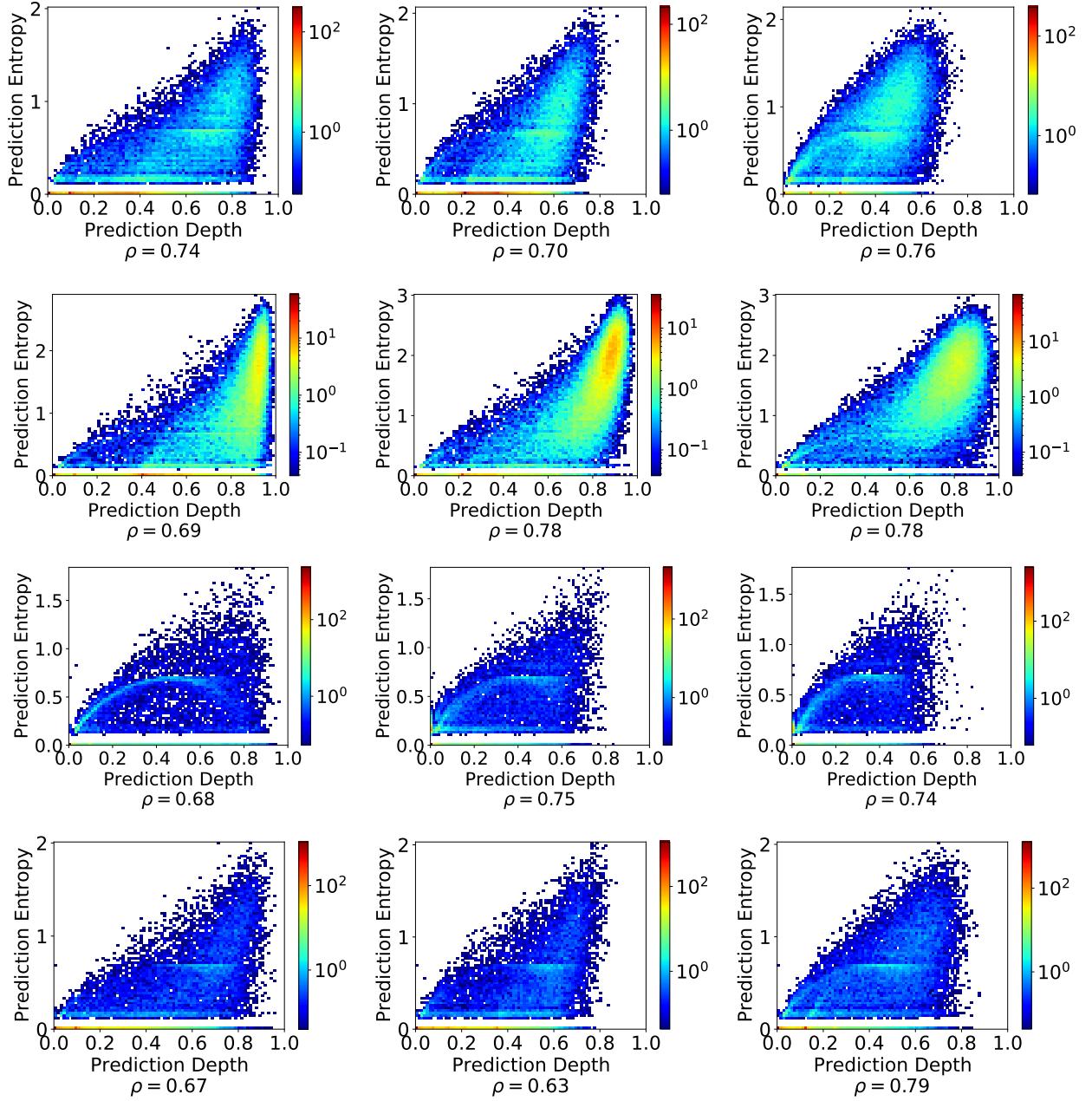


Figure C.22: First (Top) Row: CIFAR10. Second Row: CIFAR100. Third Row: Fashion MNIST. Fourth (Bottom) Row: SVHN. Left Column: ResNet18. Middle Column: VGG16. Right Column: MLP. Histograms showing consistency of the relationship between prediction depth in the validation set and prediction entropy of an ensemble. As described in Appendix A, for each dataset and architecture we trained 250 models with random 90:10% validation:train splits. Each time a data point appears in the validation split we record the prediction depth and the prediction. These histograms compare the average prediction depth for each data point to its prediction entropy. We observe that the prediction depth gives linear upper bounds for the prediction entropy as it does linear lower bounds for the consensus-consistency (Figures C.17 and C.18).

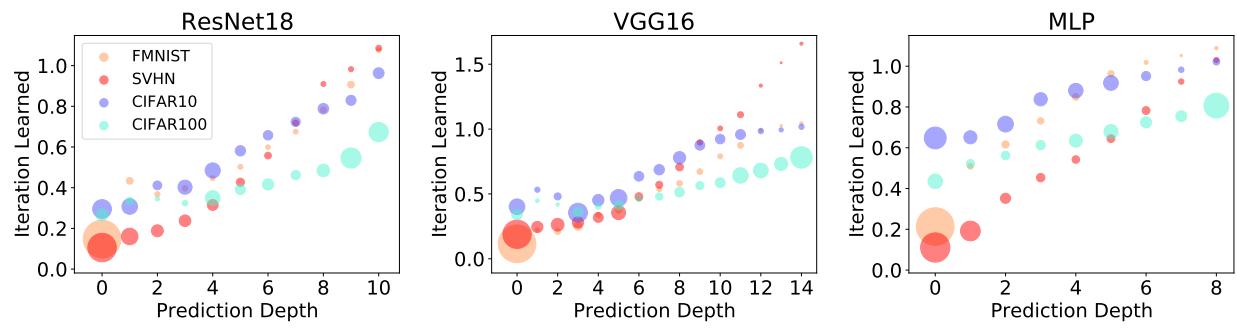


Figure C.23: This figure demonstrates the consistency of the result shown in Figure 5 (left) for all datasets and architectures.

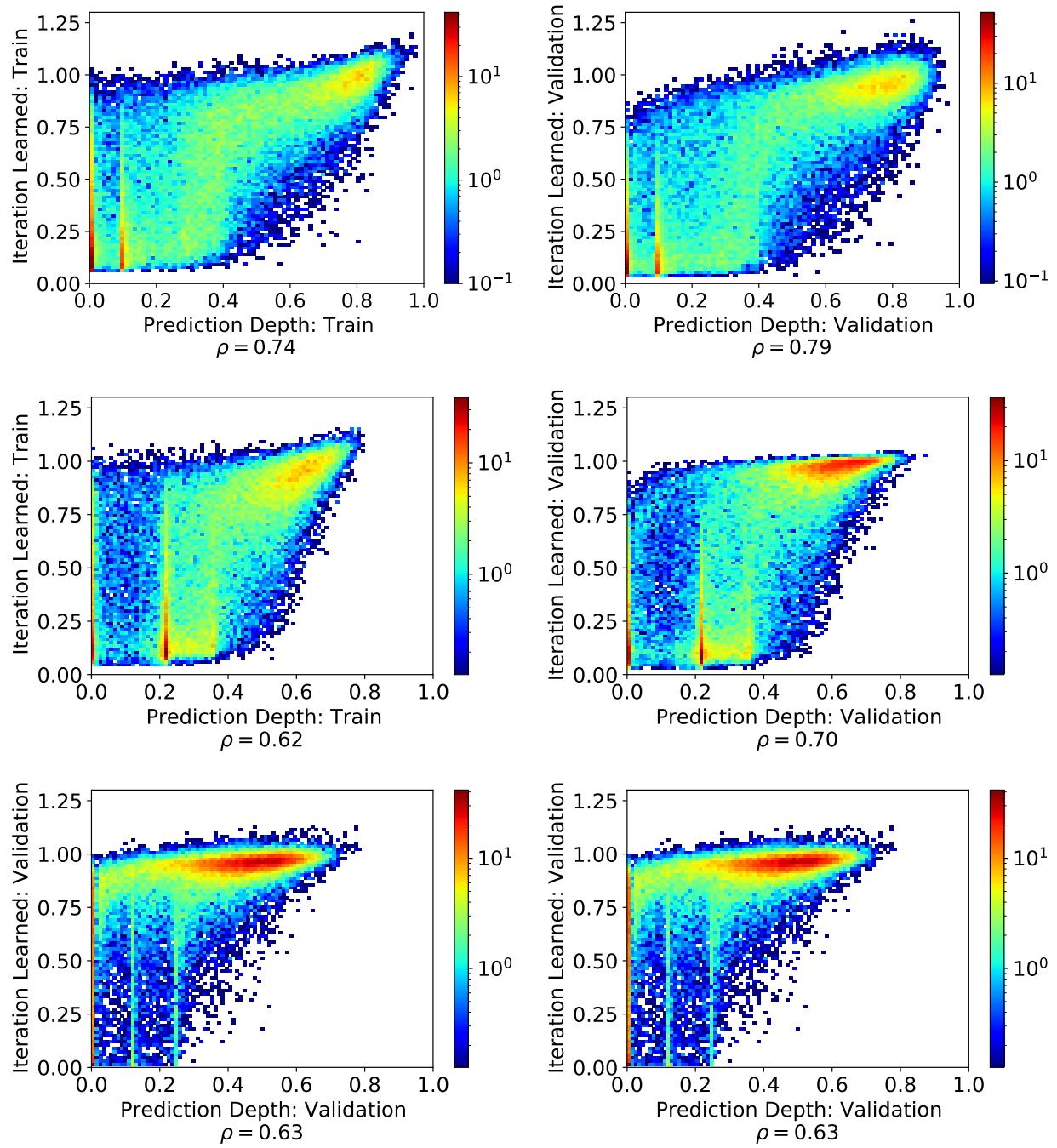


Figure C.24: CIFAR10. Top row: ResNet18. Middle row: VGG16. Bottom row: MLP. Histogram comparing the mean prediction depth to the mean iteration learned when each data point occurs in either the training split (left column) or the validation split (right column). See Appendix C.3 for a description of the experiments performed.

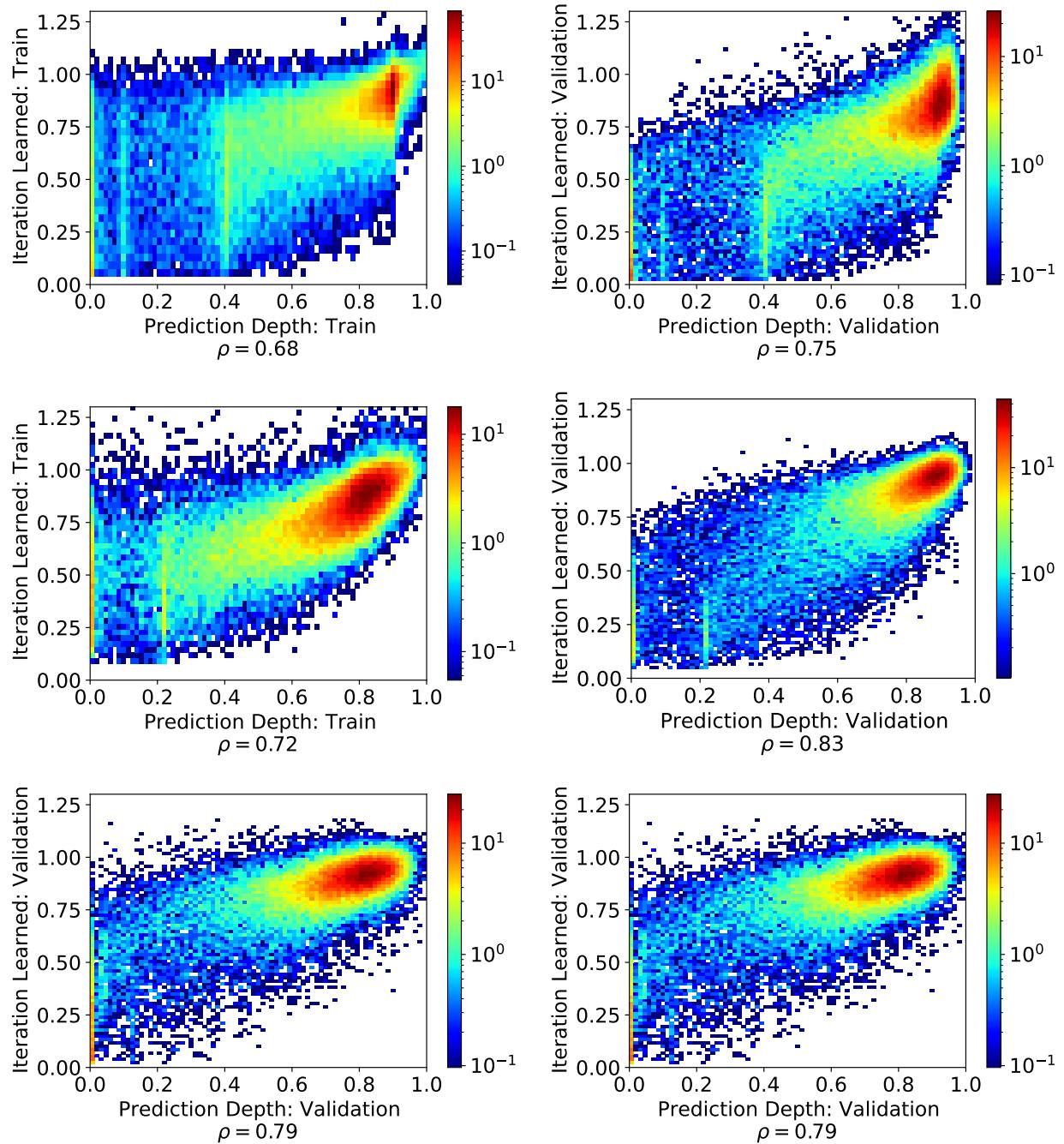


Figure C.25: CIFAR100. Top row: ResNet18. Middle row: VGG16. Bottom row: MLP. Histogram comparing the mean prediction depth to the mean iteration learned when each data point occurs in either the training split (left column) or the validation split (right column). See Appendix C.3 for a description of the experiments performed.

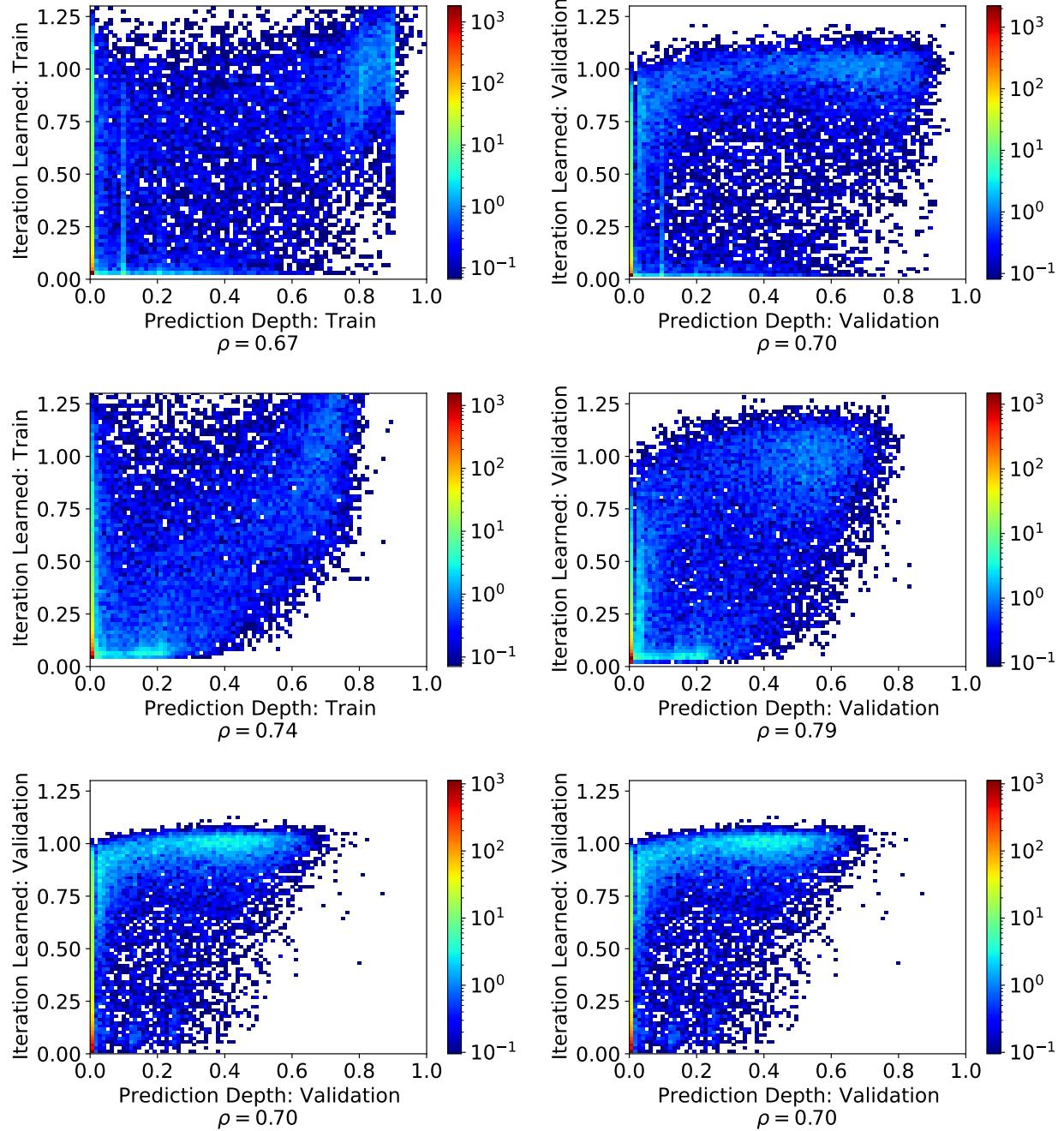


Figure C.26: Fashion MNIST. Top row: ResNet18. Middle row: VGG16. Bottom row: MLP. Histogram comparing the mean prediction depth to the mean iteration learned when each data point occurs in either the training split (left column) or the validation split (right column). In this case, the large majority of the data is already learned in the input layer. See Appendix C.3 for a description of the experiments performed.

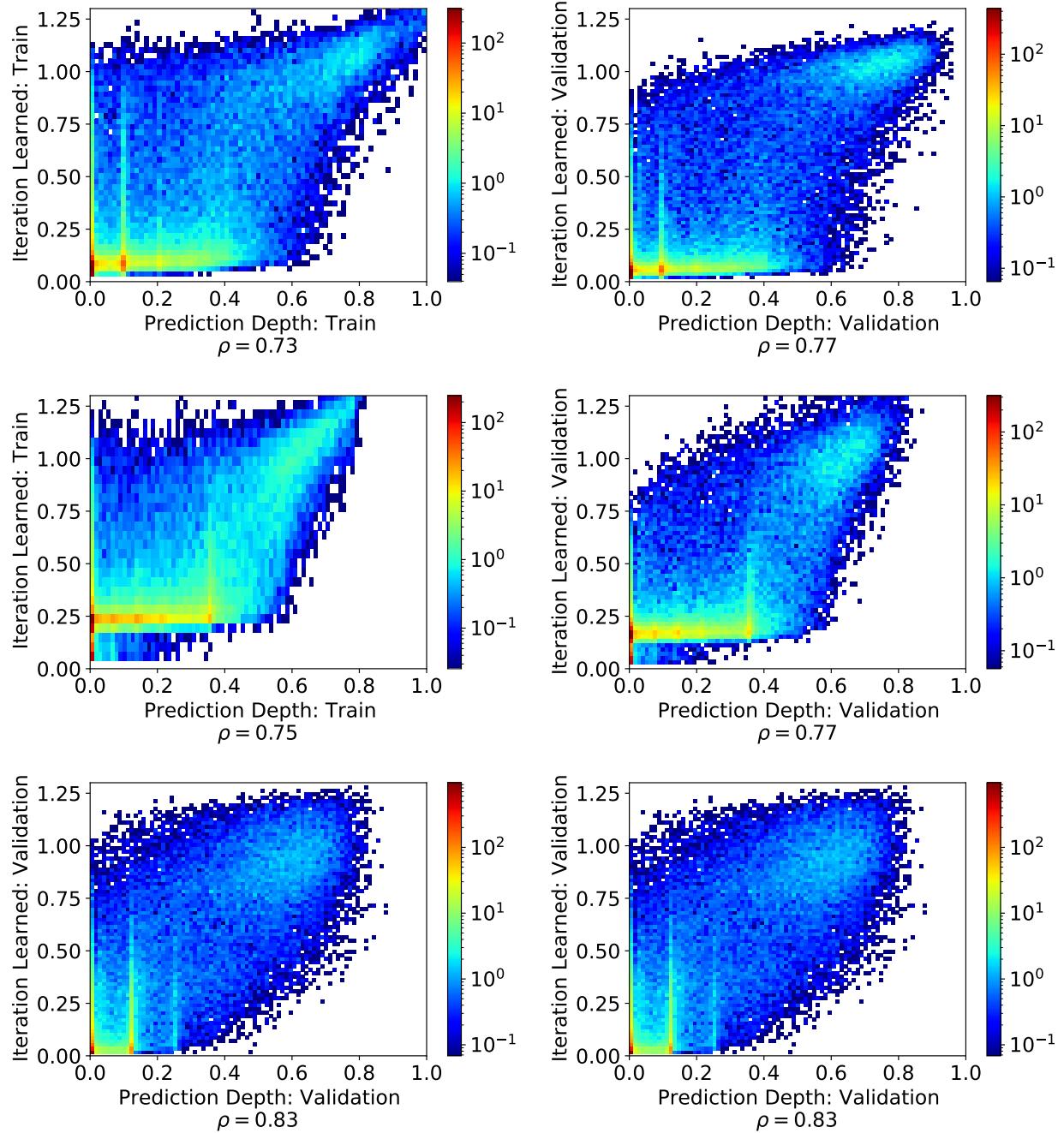


Figure C.27: SVHN. Top row: ResNet18. Middle row: VGG16. Bottom row: MLP. Histogram comparing the mean prediction depth to the mean iteration learned when each data point occurs in either the training split (left column) or the validation split (right column). See Appendix C.3 for a description of the experiments performed.

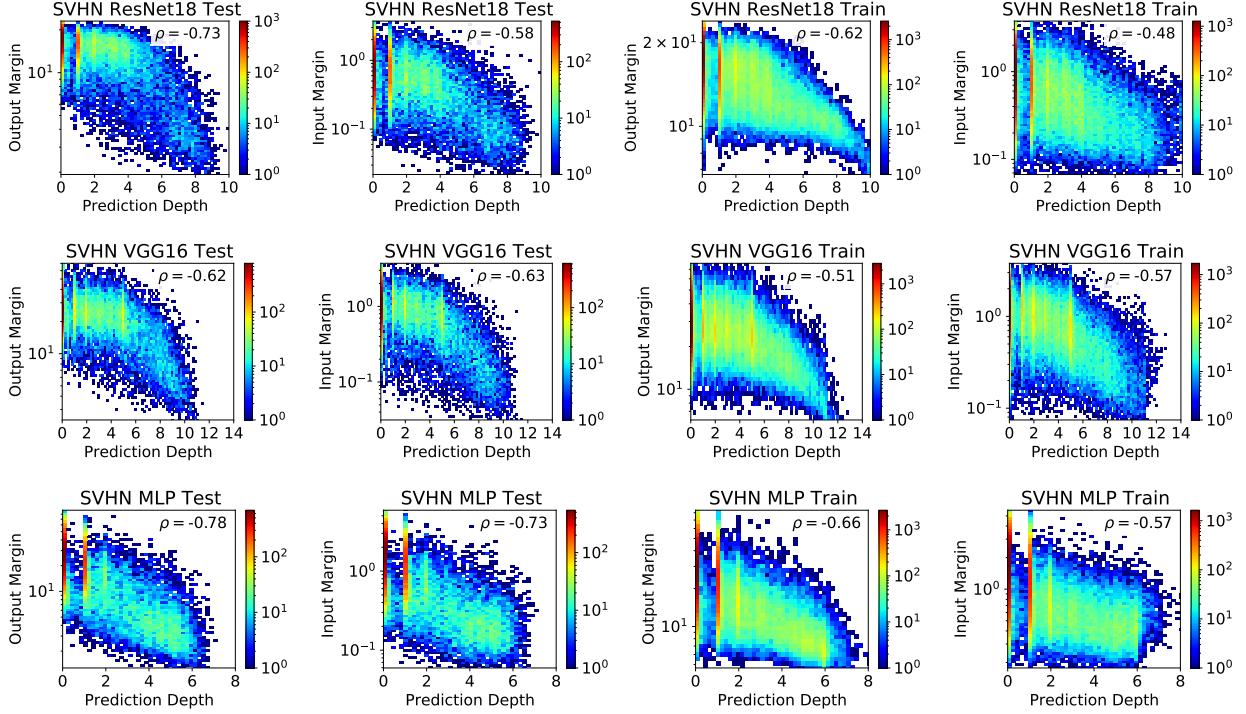


Figure C.28: Consistency of Figure 6, showing the correlation between prediction depth, and the input and output margins (log scale) for both the test and training splits of SVHN. The correlation coefficient between the prediction depth and the logarithm of the margin is given in each plot. For each architecture, we train 25 models with different random seeds on the full training split. We record the input and output margins together with the prediction depth for every data point in both the train and test splits. These histograms compare the mean values of each margin to the mean prediction depth for all data points.

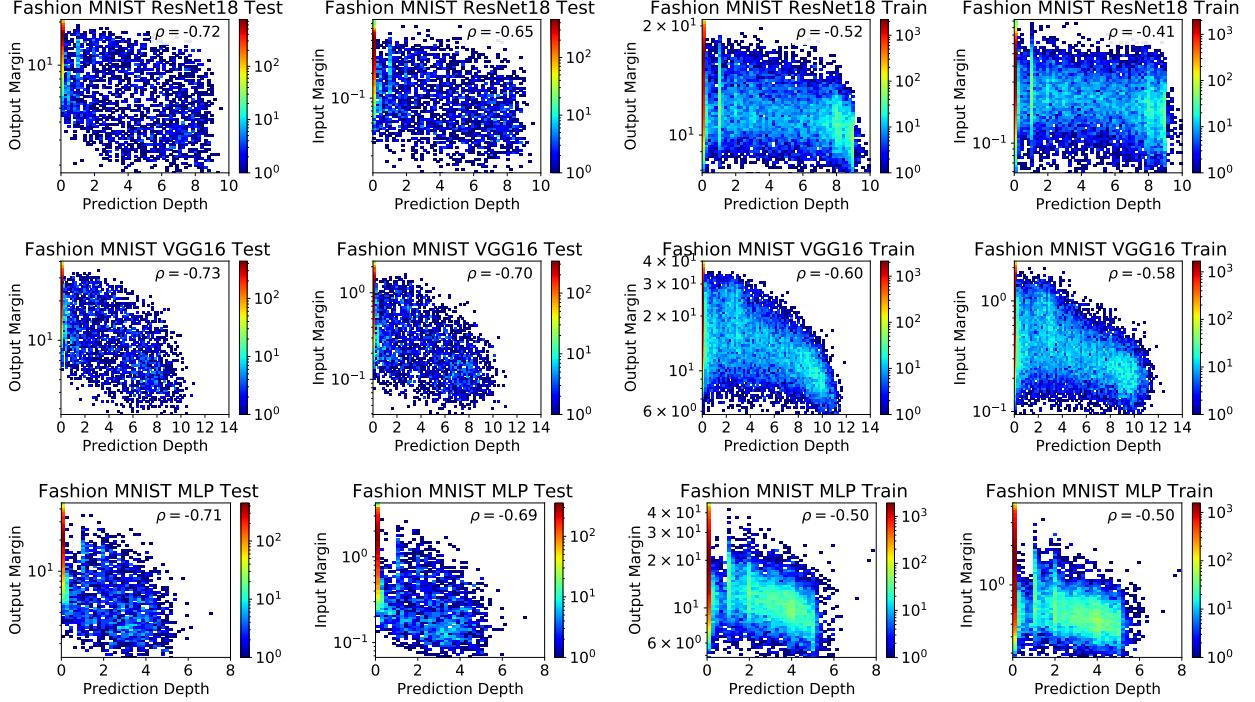


Figure C.29: Consistency of Figure 6, showing the correlation between prediction depth, and the input and output margins (log scale) for both the test and training splits of Fashion MNIST. The correlation coefficient between the prediction depth and the logarithm of the margin is given in each plot. For each architecture, we train 25 models with different random seeds on the full training split. We record the input and output margins together with the prediction depth for every data point in both the train and test splits. These histograms compare the mean values of each margin to the mean prediction depth for all data points.

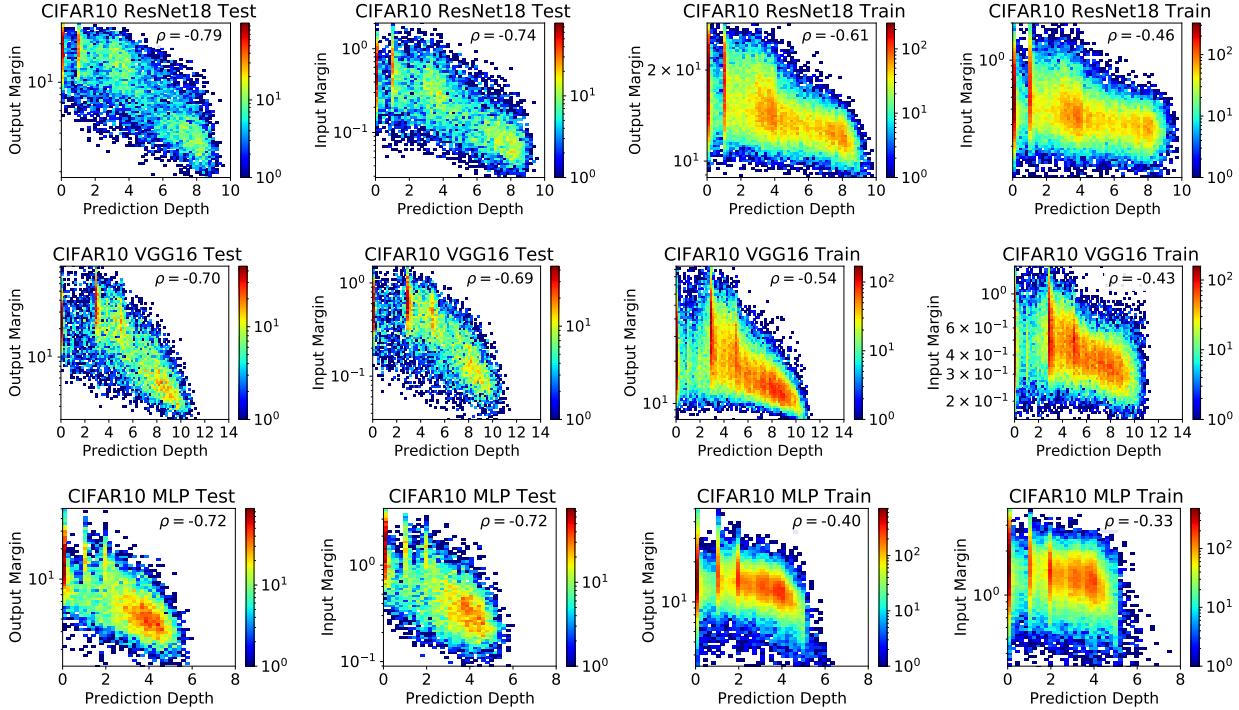


Figure C.30: Consistency of Figure 6, showing the correlation between prediction depth, and the input and output margins (log scale) for both the test and training splits of CIFAR10. The correlation coefficient between the prediction depth and the logarithm of the margin is given in each plot. For each architecture, we train 25 models with different random seeds on the full training split. We record the input and output margins together with the prediction depth for every data point in both the train and test splits. These histograms compare the mean values of each margin to the mean prediction depth for all data points.

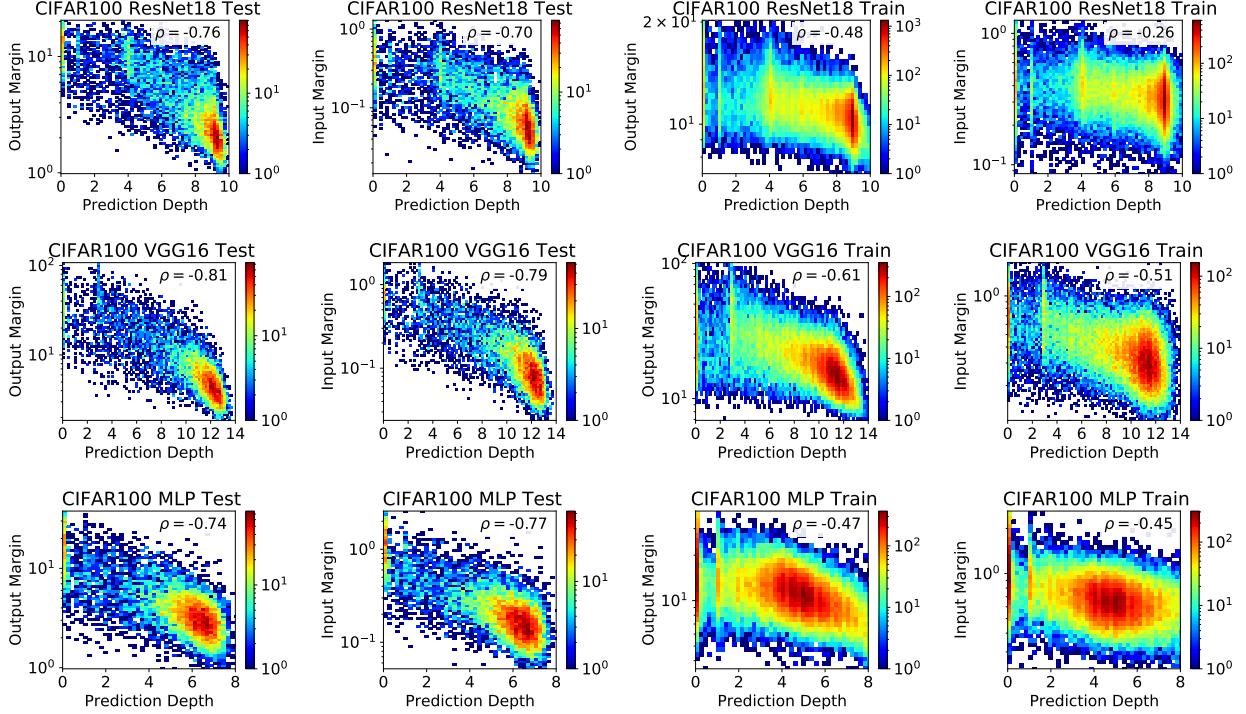


Figure C.31: Consistency of Figure 6, showing the correlation between prediction depth, and the input and output margins (log scale) for both the test and training splits of CIFAR100. The correlation coefficient between the prediction depth and the logarithm of the margin is given in each plot. For each architecture, we train 25 models with different random seeds on the full training split. We record the input and output margins together with the prediction depth for every data point in both the train and test splits. These histograms compare the mean values of each margin to the mean prediction depth for all data points.

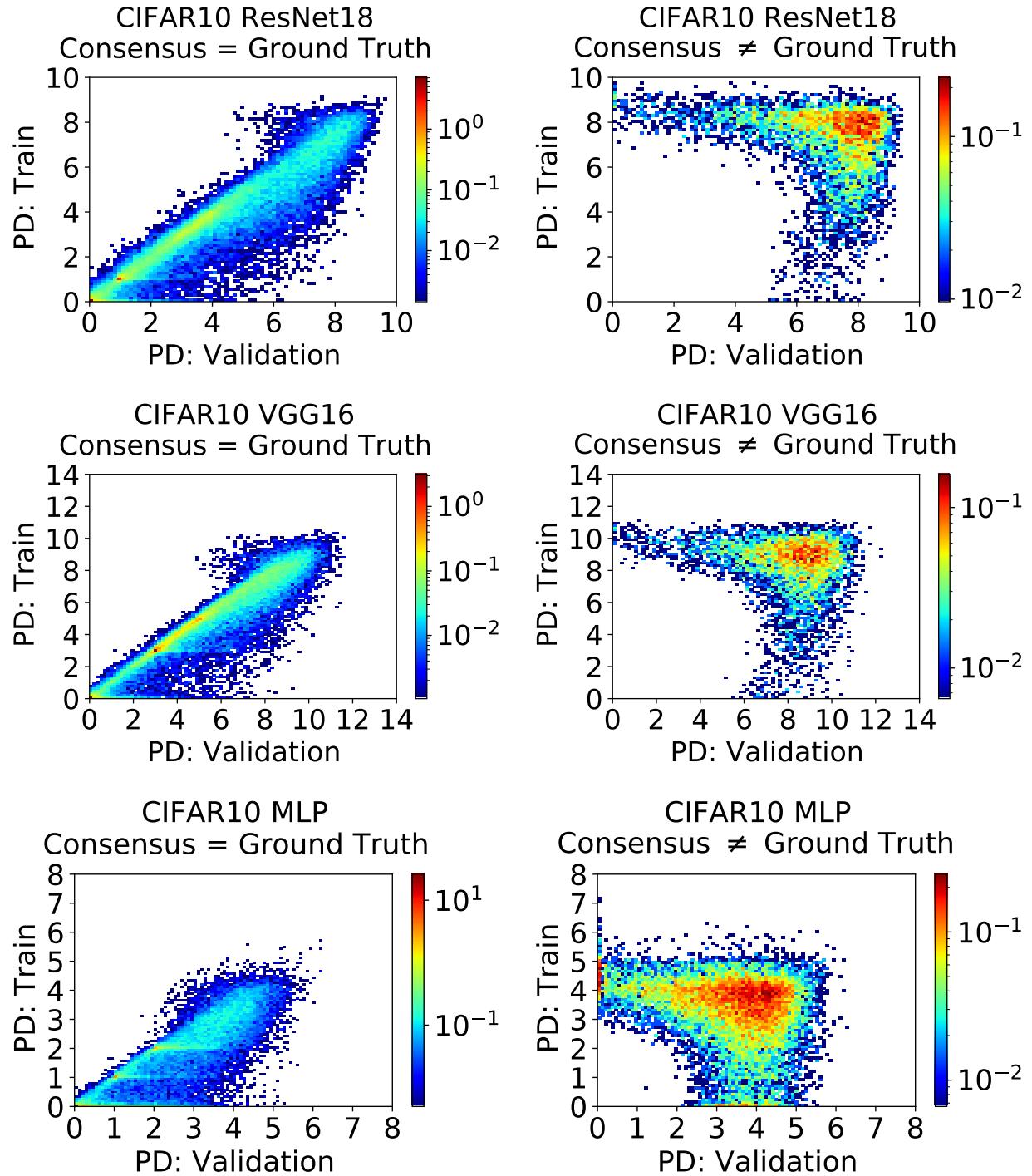


Figure C.32: Demonstrating consistency of the histograms shown in Figure 7 for all architectures on CIFAR10. These histograms compare the mean prediction depth when each data point occurs in either the validation split or the training split. Results are shown separately for data points where the consensus class is the same as or different from the ground truth label. See Appendix C.5 for a description.

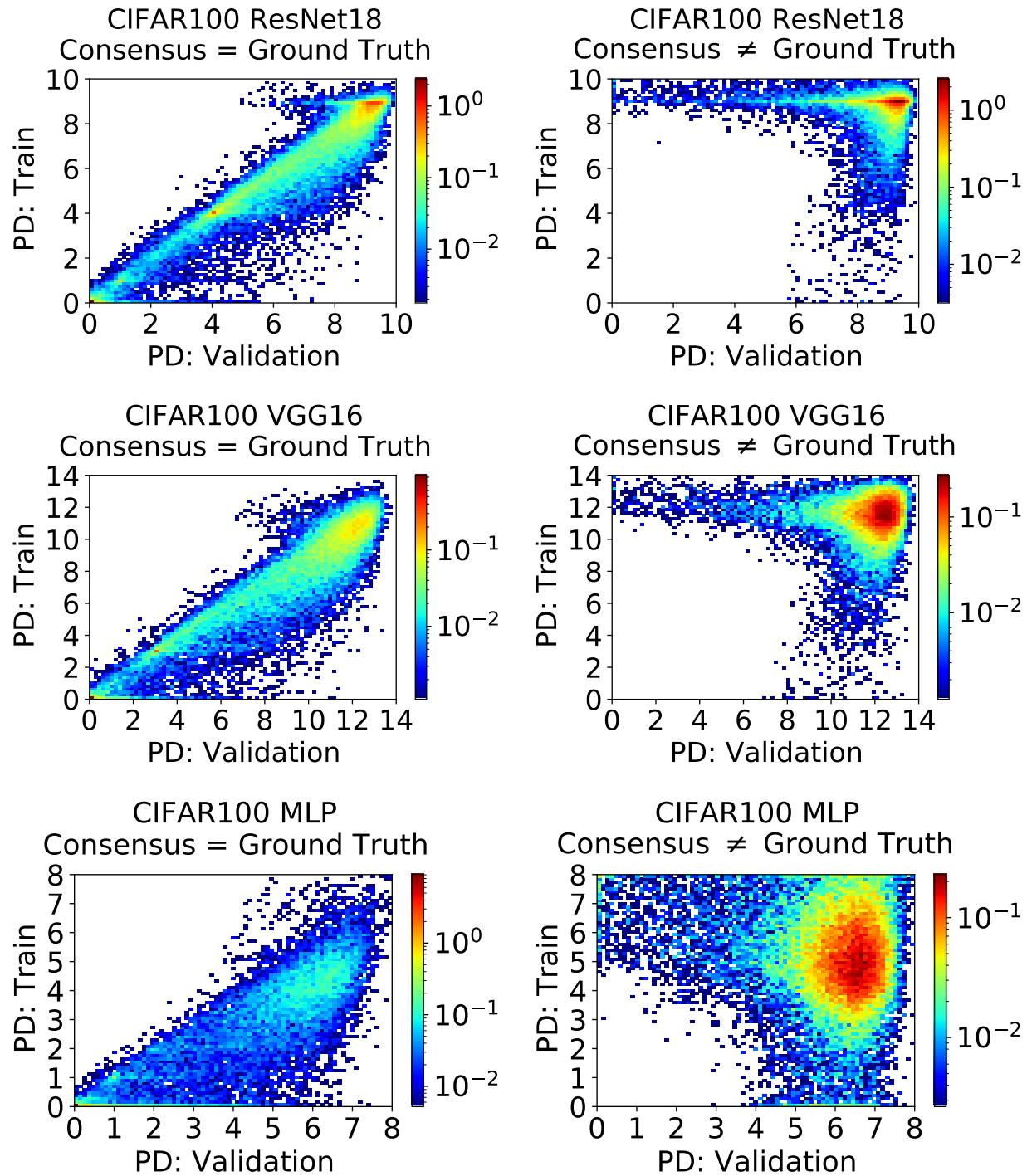


Figure C.33: Demonstrating consistency of the histograms shown in Figure 7 for all architectures on CIFAR100. These histograms compare the mean prediction depth when each data point occurs in either the validation split or the training split. Results are shown separately for data points where the consensus class is the same as or different from the ground truth label. See Appendix C.5 for a description.

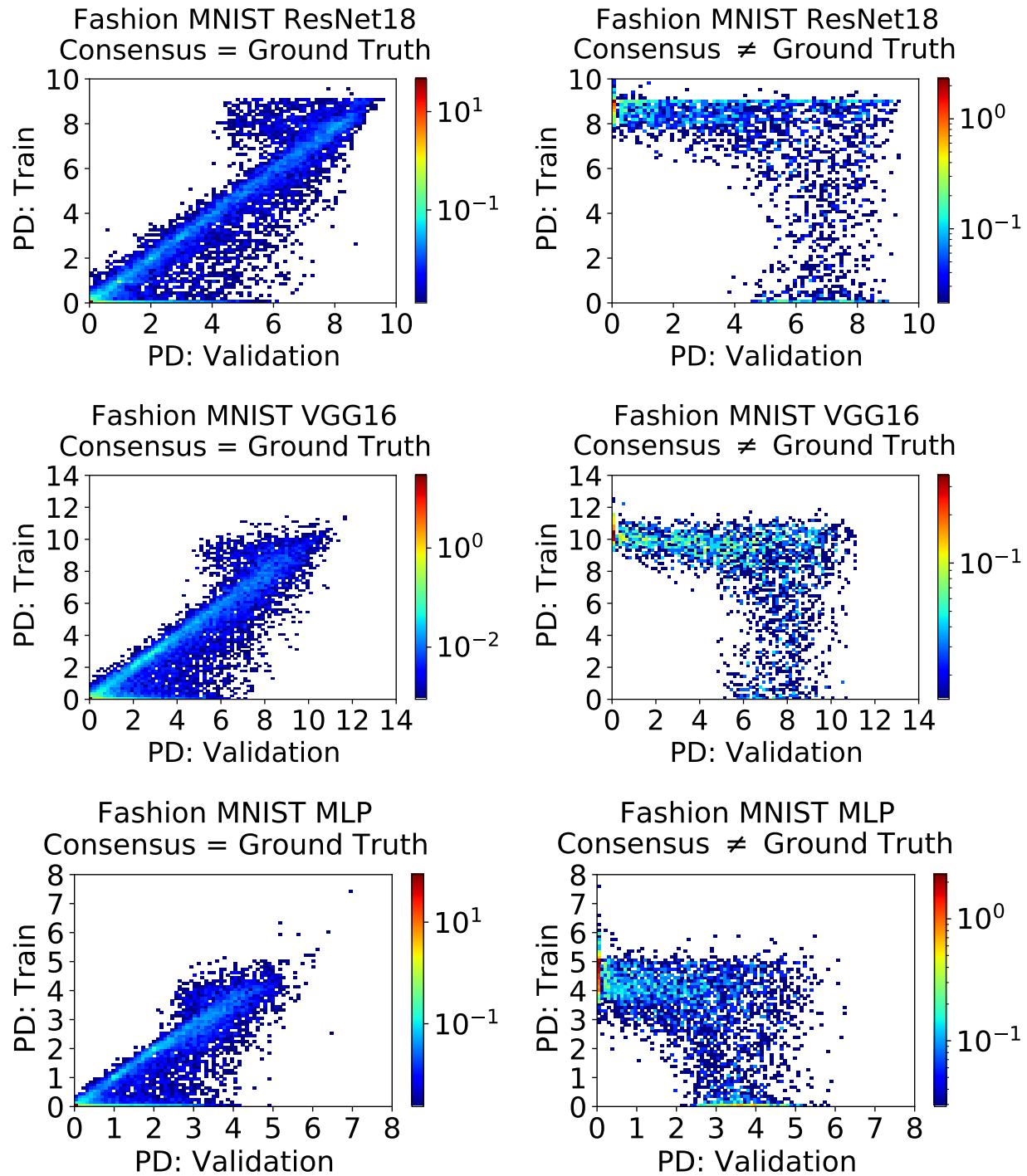


Figure C.34: Demonstrating consistency of the histograms shown in Figure 7 for all architectures on Fashion MNIST. These histograms compare the mean prediction depth when each data point occurs in either the validation split or the training split. Results are shown separately for data points where the consensus class is the same as or different from the ground truth label. See Appendix C.5 for a description.

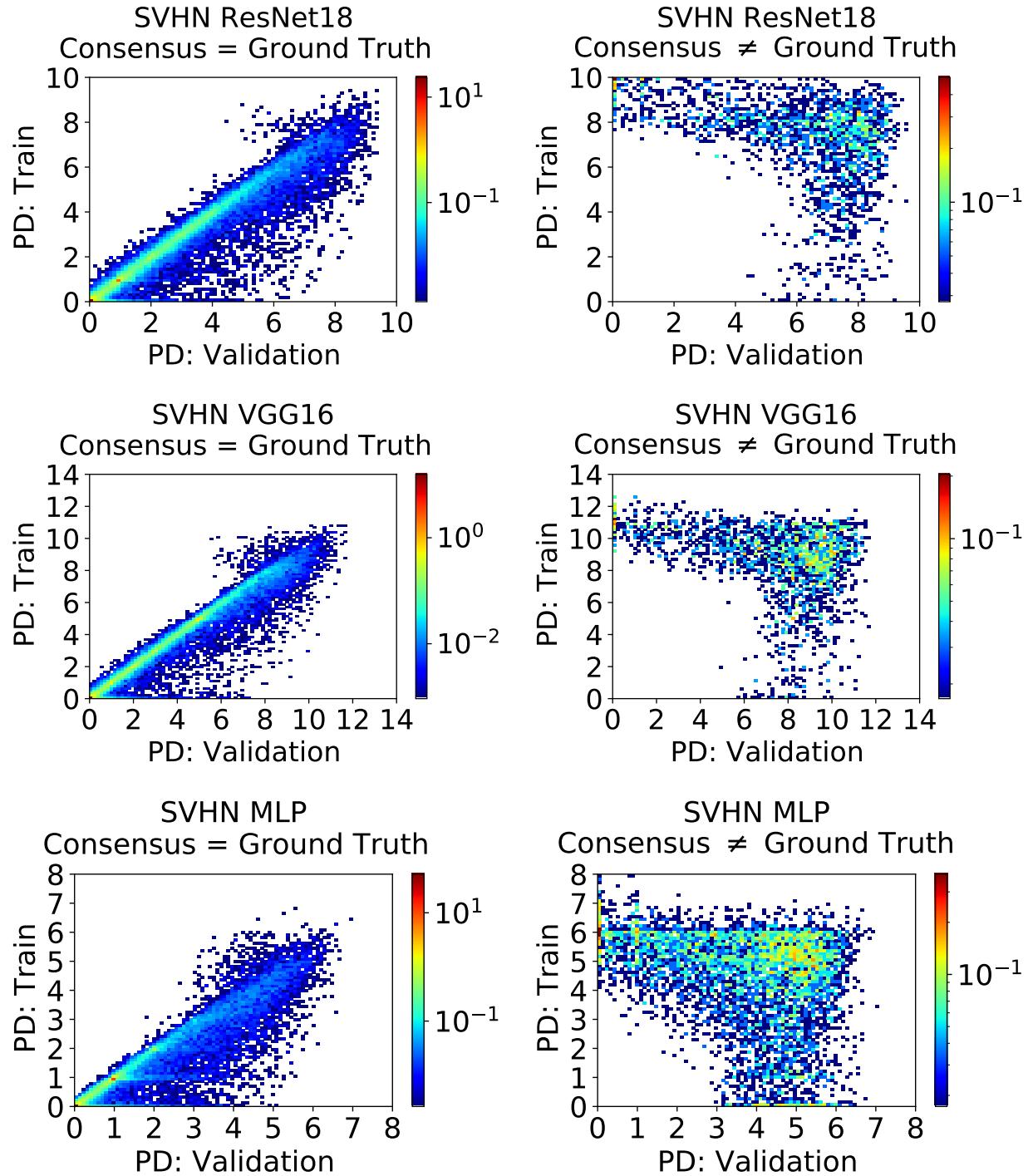


Figure C.35: Demonstrating consistency of the histograms shown in Figure 7 for all architectures on SVHN. These histograms compare the mean prediction depth when each data point occurs in either the validation split or the training split. Results are shown separately for data points where the consensus class is the same as or different from the ground truth label. See Appendix C.5 for a description.

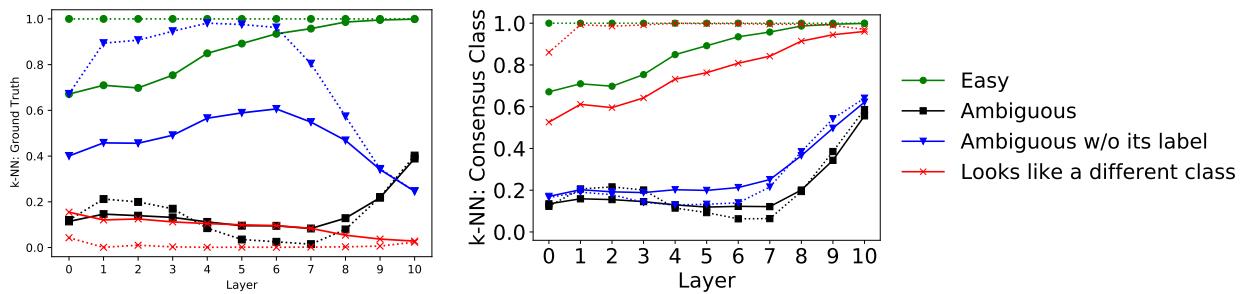


Figure C.36: Reproducing Figure 8 for ResNet18 on CIFAR10.

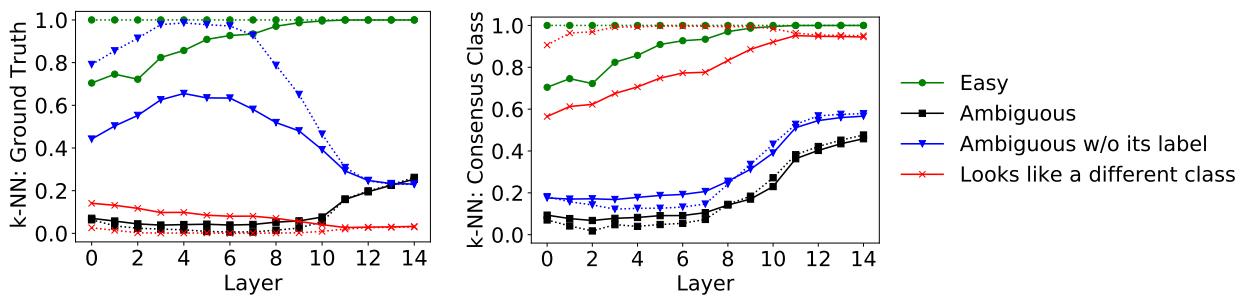


Figure C.37: Reproducing Figure 8 for VGG16 on CIFAR10.

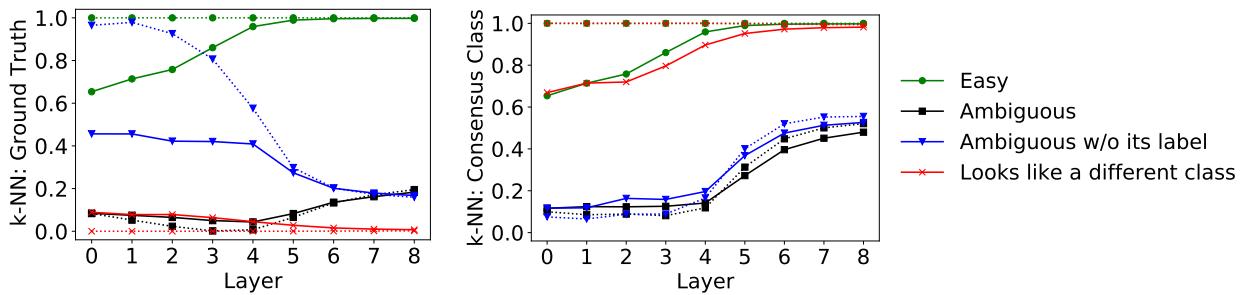


Figure C.38: Reproducing Figure 8 for MLP on CIFAR10.

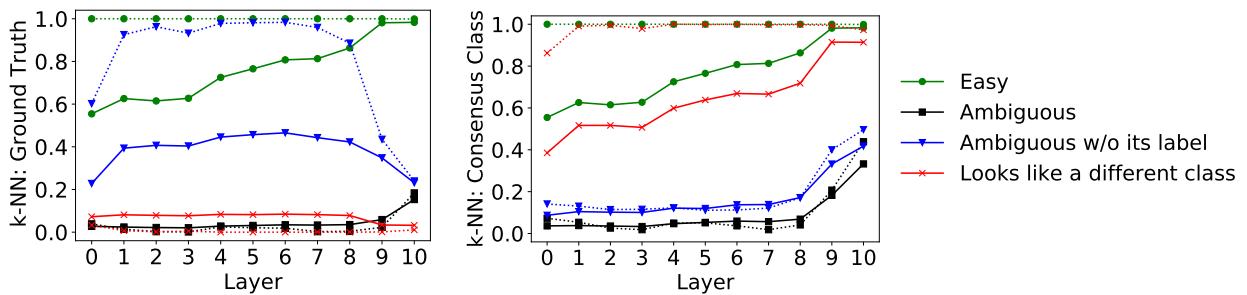


Figure C.39: Reproducing Figure 8 for ResNet18 on CIFAR100.

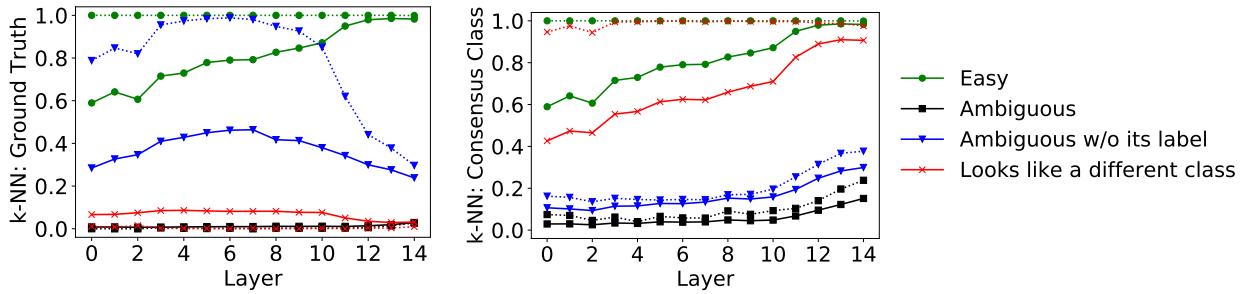


Figure C.40: Reproducing Figure 8 for VGG16 on CIFAR100.

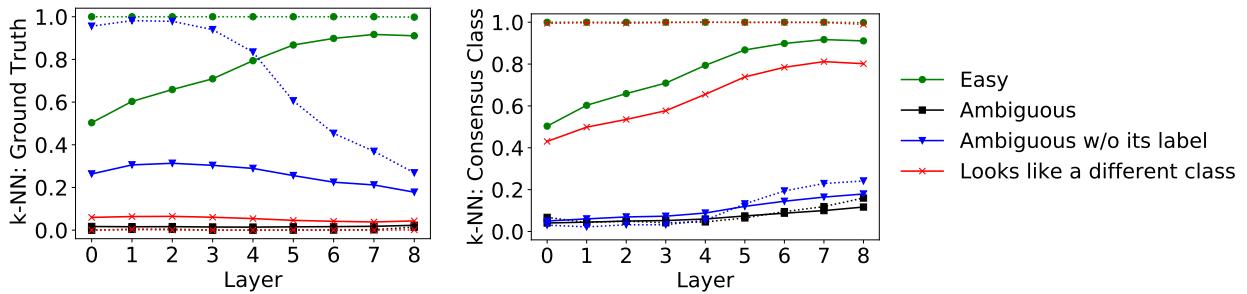


Figure C.41: Reproducing Figure 8 for MLP on CIFAR100.

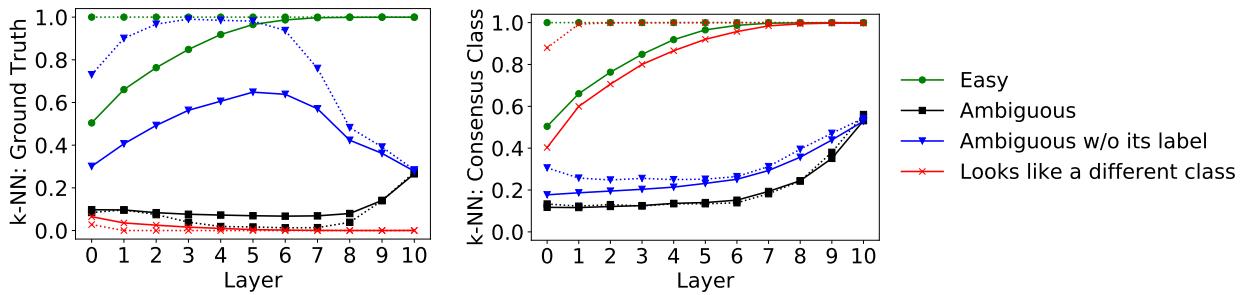


Figure C.42: Reproducing Figure 8 for ResNet18 on SVHN.

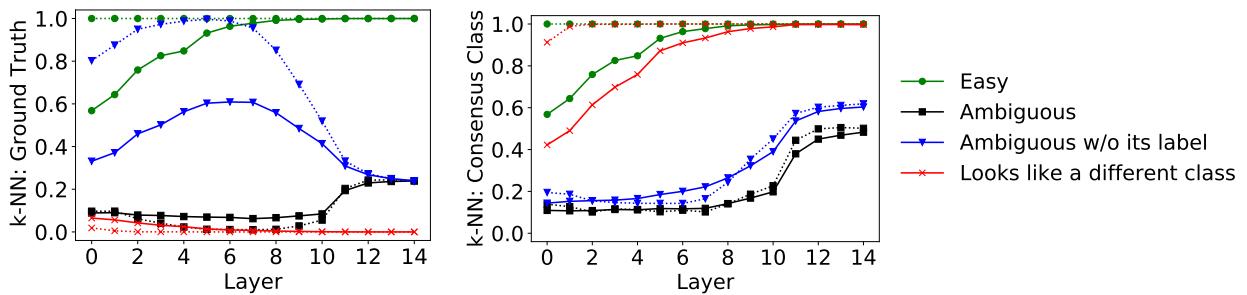


Figure C.43: Reproducing Figure 8 for VGG16 on SVHN.

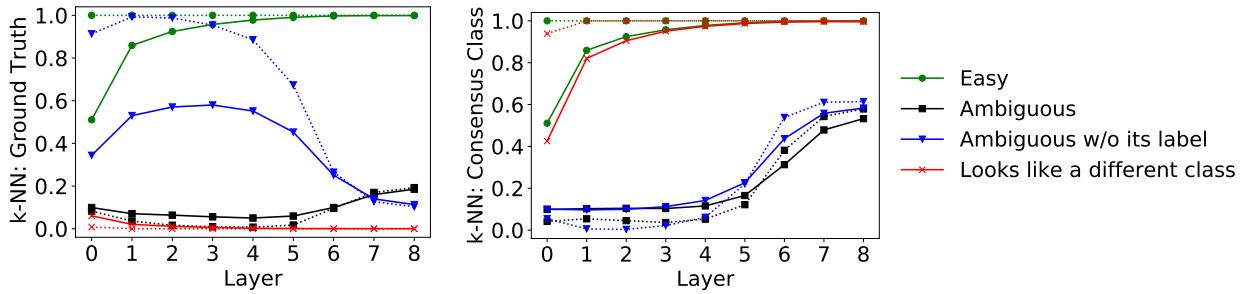


Figure C.44: Reproducing Figure 8 for MLP on SVHN.

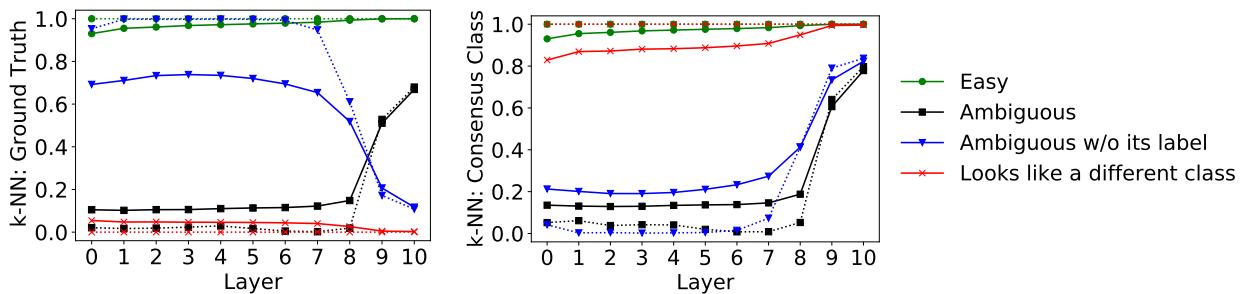


Figure C.45: Reproducing Figure 8 for ResNet18 on Fashion MNIST.

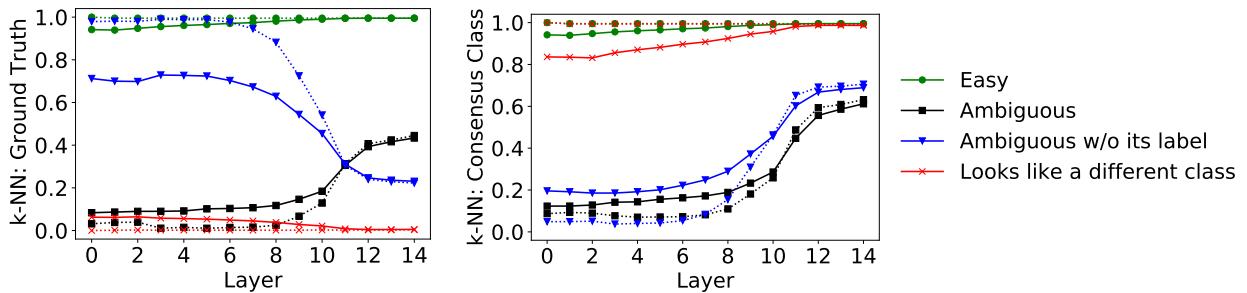


Figure C.46: Reproducing Figure 8 for VGG16 on Fashion MNIST.

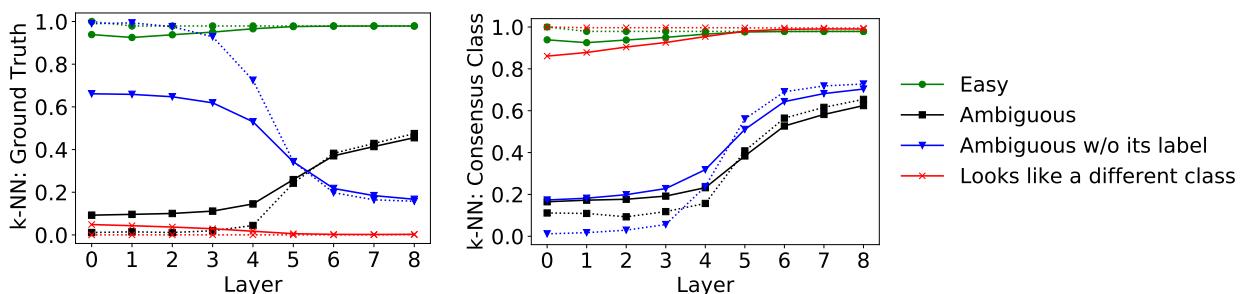


Figure C.47: Reproducing Figure 8 for MLP on Fashion MNIST.

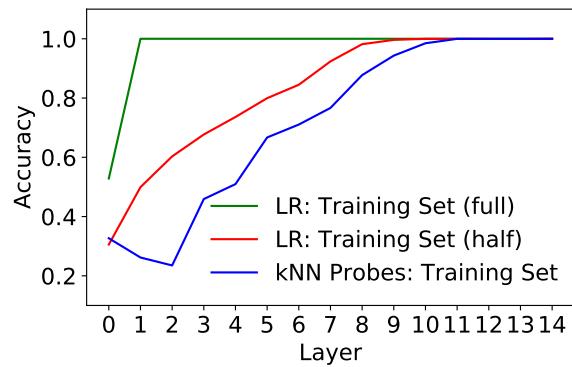


Figure E.48: Comparison of k-NN probe and Logistic Regression (LR) probe accuracies for VGG16 trained on CIFAR10. LR is already able to divide the training set into linearly separated classes after the first convolutional operation. In red we show the accuracy of LR probes trained on a random subset (half) of the data and predicting on the other half. These results are converged (closely repeatable between different trained VGG16 models).