ALL NEURAL NETWORKS ARE CREATED EQUAL

Anonymous authors
Paper under double-blind review

ABSTRACT

One of the unresolved questions in deep learning is the nature of the solutions that are being discovered. We investigate the collection of solutions reached by the same network architecture, with different random initialization of weights and random mini-batches. These solutions are shown to be rather similar - more often than not, each train and test example is either classified correctly by all the networks, or by none at all. Surprisingly, all the network instances seem to share the same learning dynamics, whereby initially the same train and test examples are correctly recognized by the learned model, followed by other examples which are learned in roughly the same order. When extending the investigation to heterogeneous collections of neural network architectures, once again examples are seen to be learned in the same order irrespective of architecture, although the more powerful architecture may continue to learn and thus achieve higher accuracy. This pattern of results remains true even when the composition of classes in the test set is unrelated to the train set, for example, when using out of sample natural images or even artificial images. To show the robustness of these phenomena we provide an extensive summary of our empirical study, which includes hundreds of graphs describing tens of thousands of networks with varying NN architectures, hyper-parameters and domains. We also discuss cases where this pattern of similarity breaks down, which show that the reported similarity is not an artifact of optimization by gradient descent. Rather, the observed pattern of similarity is characteristic of learning complex problems with big networks. Finally, we show that this pattern of similarity seems to be strongly correlated with effective generalization.

1 INTRODUCTION

The recent success of deep networks in solving a variety of classification problems effectively, in some cases reaching human-level precision, is not well understood. One baffling result is the incredible robustness of the learned models: using variants of Stochastic Gradient Descent (SGD), with random weight initialization and random sampling of mini-batches, different solutions are obtained. While these solutions typically correspond to different parameter values and possibly different local minima of the loss function, nevertheless they demonstrate similar performance reliably.

To advance our understating of this issue, we are required to compare different network instances. Most comparison approaches (briefly reviewed in Appendix A) are based on deciphering the internal representations of the learned models (see Lenc & Vedaldi, 2015; Alain & Bengio, 2016; Li et al., 2016; Raghu et al., 2017; Wang et al., 2018). We propose a simpler and more direct approach – comparing networks by their classifications of the data. To this end, we represent each network instance by 2 binary vectors which capture the train and test classification accuracy. Each vector's dimension corresponds to the size of the train/test dataset; each element is assigned 1 if the network classifies the corresponding data point correctly, and 0 otherwise.

Recall the aforementioned empirical observation - different neural network instances, obtained by repeatedly training the same architecture with SGD while randomly sampling its initial weights, achieve similar accuracy. At the very least, this observation predicts that the test-based vector representation of different networks should have similar L_1/L_2 norms. But there is more: it has been recently shown that features of deep networks capture perceptual similarity reliably and consistently, similarly across different instances and different architectures (Zhang et al., 2018). These results seem to suggest that our proposed representation vectors may not only have a similar norm, but should also be quite similar as individual vectors. But similar in what way?

In this paper, we analyze collections of deep neural networks classifiers, where the only constraint is that the instances are trained on the same classification problem, and investigate the similarity between them. Using the representation discussed above, we measure this similarity by two scores, *consistency score* and *consensus score*, as defined in §2. Like other comparison approaches (see Appendix A), our analysis reveals a high level of similarity between trained networks. Interestingly, it reveals a stronger sense of similarity than previously appreciated: not only is the accuracy of all the networks in the collection similar, but so is the pattern of classification. Specifically, at each time point during the learning process (or in each epoch), most of the data points in both the train and test sets are either classified correctly by all the networks, or by none at all.

As shown in §3, these results are independent of choices such as optimization method, hyper-parameter values, the detailed architecture, or the particular dataset. They can be replicated for a fixed test set even when each instance in the collection sees a different train set, as long as the training data is sampled from the same distribution. Moreover, the same pattern of similarity is observed for a wide range of test data, including out-of-sample images of new classes, randomly generated images, or even artificial images generated by StyleGAN (Karras et al., 2019). These results are also reproduce-able across domains, and were reproduced using BiLSTM (Hochreiter & Schmidhuber, 1997) with attention (Bahdanau et al., 2014) for text classification. We may therefore conclude that different network instances compute similar classification functions, even when being trained with different training samples.

It is in the dynamic of learning, where the results of our analysis seem to go significantly beyond what has been shown before, revealing an even more intriguing pattern of similarity between trained NN instances. Since deep NNs are almost always trained using gradient descent, each network can be represented by a time series of train-based and test-based representation vectors, one per epoch. We find that **network instances** in the collection do not only show the same pattern of classification at the end of the training, but they also **evolve in the same way across time and epochs, gradually learning to correctly or incorrectly classify the same examples in the same order**.

When considering bigger classification problems such as the classification of ImageNet with big modern CNN architectures, a more intricate pattern of dynamics is evident: to begin with, all networks wrongly classify most of the examples, and correctly classify a minority of the examples. The learning process is revealed by examples moving from one end (100% false classification) to the other end (100% correct classification), which implies two things: (i) the networks learn to correctly classify examples in the same order; (ii) the networks agree on the examples they misclassify throughout.

As shown in §4, these results hold regardless of the network' architecture. To drive this point home we compare a variety of public domain architectures such as VGG19 (Simonyan & Zisserman, 2014), AlexNet (Krizhevsky et al., 2012), DenseNet (Huang et al., 2017) and ResNet-50 (He et al., 2016). **In all cases, different architectures may learn at a different pace and achieve different generalization accuracy, but they still learn in the same order**. Thus all networks start by learning roughly the same examples, but the more powerful networks may continue to learn additional examples as learning proceeds. A related phenomenon is observed when extending the analysis to simpler learning paradigms, such as deep linear networks, SVM, and KNN classifiers.

Our empirical study extends to cases where these robust patterns of similarity break down, see §5. For example, when randomly shuffling the labels in a known benchmark (Zhang et al., 2016), the agreement between different classifiers disappear. This stands in agreement with (Morcos et al., 2018), where it is shown that networks that generalize are more similar than those that memorize.

Nevertheless, the similarity in learning dynamic is not an artifact of learnability, or the fact that the networks have converged to solutions with similar accuracy. To see this we constructed a test case where shallow CNNs are trained to discriminate an artificial dataset of images of Gabor patches (see Appendix C). Here it is no longer true that different network instances learn in the same order; rather, each network instance follows its own path while converging to the final model. The similarity in learning dynamic is likewise not an artifact of using gradient descent. To see this we use SGD to train linear classifiers to discriminate vectors sampled from two largely overlapping Gaussian distributions. Once again, each classifier follows its own path while converging to the same optimal solution.

2 METHODOLOGY AND NOTATIONS

Given some neural network architecture f, and a labeled dataset $\mathbb{X} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ denotes a single data point and $y_i \in [K]$ its corresponding label, we define and analyze the *consistency* of f when repeatedly trained on $\mathbb{X}$ from scratch. Let $\mathcal{S}_E$ denote the set of different extents (total epochs of $\mathbb{X}$) used to train f, where $|\mathcal{S}_E| = E$. $\forall e \in \mathcal{S}_E$ we create a collection of N instances of f, denoted $\mathcal{F}^e = \{f_1^e, ..., f_N^e\}$. Each instance f_i^e is initialized independently using Xavier initialization (Glorot & Bengio, 2010), then trained with SGD on randomly sampled mini-batches for e epochs.

We measure the consistency of architecture f by comparing the predictions of the different instances in each collection $\mathcal{F}^e$, and analyze this consistency throughout the entire learning process as it changes with e. Thus for epoch e, we define the *consistency score* of an example $(\boldsymbol{x}, \boldsymbol{y})$ as follows:

$$c^e(\boldsymbol{x}, y) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{[f_i^e(\boldsymbol{x})=y]}$$

The consistency score $c^e(\boldsymbol{x}, y)$ measures the classifiers' agreement when $\boldsymbol{x}$ is correctly classified. However, it does not take into account the classifiers' agreement when it is not. We therefore define in addition the *consensus score*, a complementary score that measures the consensus of the classifiers - the largest number of classifiers that classify each example by the same label:

$$s^e(\boldsymbol{x}, y) = \max_{k \in [K]} \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{[f_i^e(\boldsymbol{x})=k]}$$

We say that example $(\boldsymbol{x}_i, y_i)$ is *easier* than example $(\boldsymbol{x}_j, y_j)$ during epoch e, if $c^e(\boldsymbol{x}_i, y_i) \geq c^e(\boldsymbol{x}_j, y_j)$. We say that example $(\boldsymbol{x}_i, y_i)$ is *learned at least as fast as* example $(\boldsymbol{x}_j, y_j)$ if from some epoch e' and onward, example $(\boldsymbol{x}_i, y_i)$ is easier than example $(\boldsymbol{x}_j, y_j)$. This is formally given by: $\exists e' \in \mathcal{S}_E : \forall e > e'\ c^e(\boldsymbol{x}_i, y_i) \geq c^e(\boldsymbol{x}_j, y_j)$.

When all the classifiers in $\mathcal{F}^e$ are identical, the consistency score of each example is either 0 or 1, and its consensus is 1. This results in a perfect bi-modal distribution of consistency scores; we quantify bi-modality using the following measure suggested by Pearson (1894) for an RV X: $kurtusis(X) - skewness^2(X) - 1$; the lower this measure is, the more bi-modal X is. On the other hand, when all classifiers independently choose class labels, the distribution of both scores is expected to resemble a Gaussian[1], centered around the average accuracy of the classifiers for the consistency score, and a slightly higher value for the mean consensus score. It follows that the higher the mean consensus is, and the more bi-modal-like the distribution of consistency scores is around 0 and 1, the more similar the set of classifiers is.

Throughout this paper we empirically investigate the distribution of the two scores defined above, using different architectures and datasets. We also examine the effects of other factors, such as resampling the train data, or classifying out-of-sample test data. For the most part, we focus on CNNs trained on visual classification tasks (although the results are reproduced in other domains), and analyze the distribution of consistency scores throughout the entire learning process.

3 DIVERSITY IN A SINGLE ARCHITECTURE

In this section, we investigate collections of classifiers obtained from a single NN architecture f.

Same training set. We start with the simplest condition, where all instances in collection $\mathcal{F}$ are obtained by training with the same training set $\mathbb{X}$, with different initial conditions and with independently sampled mini-batches. When using datasets of natural images, during learning the consistency among all networks in $\mathcal{F}$ is high, see Figs. 1a,b and Fig. 2.

Upon initialization, all networks are effectively i.i.d random variables, and therefore the distribution of consistency scores is approximately normal around random chance[1] – $\frac{1}{K}$ for K labels. After a few epochs (in many cases a single epoch is enough, see Appendix D), the consistency distribution

[1]For large enough N this follows immediately from the central limit theorem.

changes dramatically, transforming to a bi-modal distribution peaking around 0 and 1. This abrupt distribution change is robust, and rather striking: from a state where most of the examples are being classified correctly by $\frac{1}{K}$ of the networks, now most of the examples are being misclassified by all the networks, while a small fraction is being correctly classified by all the networks. When learning proceeds, the improvement in accuracy affects a shift of points from the peak at 0 to the peak at 1, and the distribution remains bi-modal, see Figs. 1a,b and Fig. 2.

The data is learned in a specific order which is insensitive to the network initialization and the sampling of the mini-batches. This is true for both the train and test sets. It indicates that the networks capture similar functions in corresponding epochs: they classify correctly the same examples, and also consistently misclassify the same examples. Had the learning of the different network instances progressed independently, we would have seen a different dynamic: as illustrated in Fig. 1c, the distribution of consistency scores of independent learners remains Gaussian in all epochs, where the Gaussian's mean slowly shifts while tracking the improved accuracy.

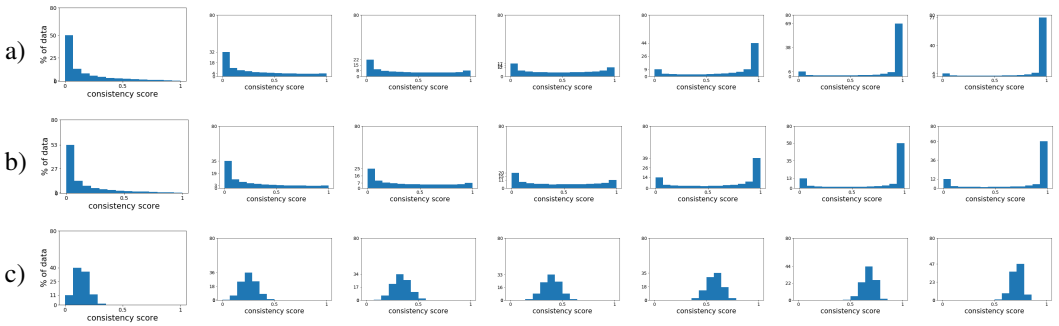


Figure 1: The distribution of consistency scores throughout the entire learning process, for 27 instances of ResNet-50 trained over ImageNet: a) Train set; b) Test set. Epochs shown, from left to right: $1, 2, 5, 30, 40, 70, 100$. All the networks converged before the final epoch. c) Distribution of consistency scores over ImageNet test set, using independent models with accuracy matched to b).

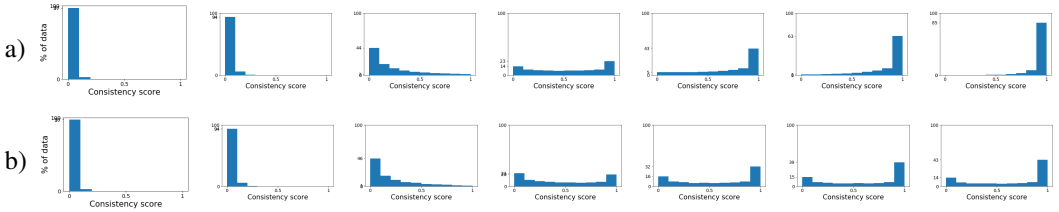


Figure 2: The distribution of consistency scores during the learning process of 20 instances of VGG19 trained on CIFAR-100. Epochs shown: $0, 1, 10, 30, 60, 80, 100$. a) Train set; b) Test set.

The consistency score is not affected by how similar the misclassifications are. For this purpose we have the consensus score, which measures consistency regardless of whether the label is true or false: a consensus score of 1 indicates that all networks have classified the datapoint in the same way, regardless of whether it is correct or incorrect. Fig. 3a shows the distribution of consensus scores for the cases depicted in Fig. 1, showing that indeed all the network instances classify examples in almost the same way, even when they misclassify. Had the learning of the different network instances progressed independently, the dynamic would have been different as shown in Fig. 3b.

Similar results are seen when analyzing a classification problem which involves text, see Fig. 4. We applied a BiLSTM with attention using Glove (Pennington et al., 2014) over 39K training and 1K test questions from stack overflow (Public domain dataset a). Labels consist of 20 mutually exclusive programming language tags assigned by users.

Robustness. The results reported above are extremely robust, seen in all the datasets and architectures that we have investigated. In addition to the results shown above on ImageNet (Deng et al.,

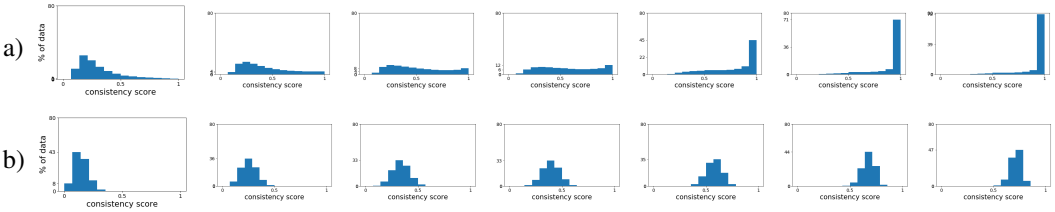


Figure 3: a) The distribution of consensus scores of 27 instances of ResNet-50 models trained on ImageNet, in corresponding epochs as in Fig. 1b. b) Illustration of the distribution of consensus scores using independent models with similar accuracies as a), cf. with Fig. 1c.

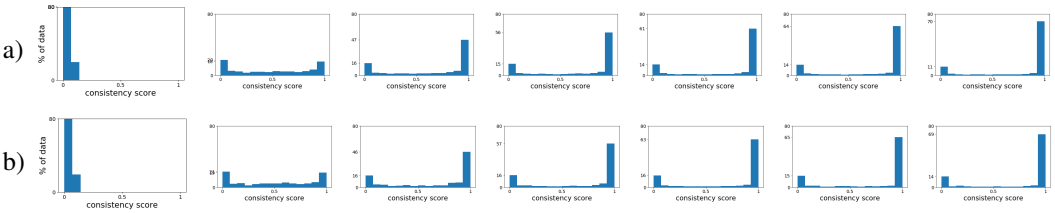


Figure 4: Like Fig. 1, for 100 instances of attention based BiLSTM trained on text classification. Epochs shown: $0, 1, 2, 3, 4, 5, 10$. a) Train set; b) Test set.

2009), CIFAR-100 (Krizhevsky & Hinton, 2009) and the text classification task, similar results are obtained for a wide range of additional image datasets as shown in Appendix D, including: MNIST (LeCun et al., 1998) - Fig. 11, Fashion-MNIST - Fig. 12, CIFAR-10 and CIFAR-100 - Figs. 2,15,17, tiny ImageNet (Public domain dataset b) - Fig. 18, ImageNet - Figs. 1,26, VGGfaces2 (Cao et al., 2018) - Fig. 16, and some subsets of these datasets - Figs. 13,14,22.

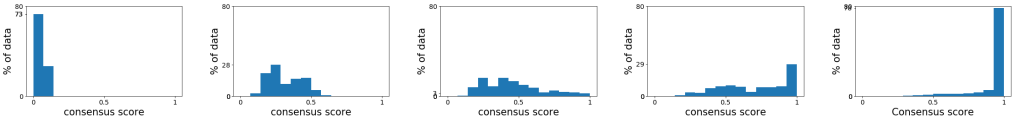
We investigated a variety of public domain architectures, including AlexNet, DenseNet and ResNet-50 for ImageNet, VGG19 and a stripped version of VGG (denoted st-VGG) for CIFAR-10 and CIFAR-100, and several different handcrafted networks for other data sets (see details in Appendix B). The results can be replicated when changing the following hyper-parameters: learning rate, optimizer, batch size, dropout, L2-regularization, width, length and depth of layers, number of layers, number and size of kernels, and activation functions. These hyper-parameters differ across the experiments detailed both in the main paper and in Appendix D.[2]

Different training sets. The observed pattern of similarity does not depend on each network instance being trained by the same train set, as long as each train set is sampled from the same distribution. To see this, we randomly split the train set into several partitions. Each of the N networks in the collection is trained using a random partition, after which we compute the distribution of consistency scores using the unmodified test set. Once again, all the networks trained with the same partition show a bi-modal distribution of consistency scores over their training partition. More interestingly, all networks, regardless of the partition used for training, show a bi-modal distribution of consistency scores over the test set as would have been the case had they been trained on the same train set, see Fig. 22 in Appendix E.

Out of sample test sets. Using the collection of ResNet-50 instances whose analysis is shown in Figs. 1,3, we further examined the consensus score[3] of the collection on out of sample test sets as shown in Fig. 5. We see that the consensus is always higher than expected from independent classifiers. Interestingly, the more natural the images are and the more similar the distributions of the train and test images are, the higher the consensus is and the further away it gets from the consensus of independent classifiers.

[2]All the code is ready for distribution, and will be published upon acceptance.

[3]Since the classes in the test sets are not present in the train, only the consensus score remains relevant.



(a) (b) (c) (d) (e)

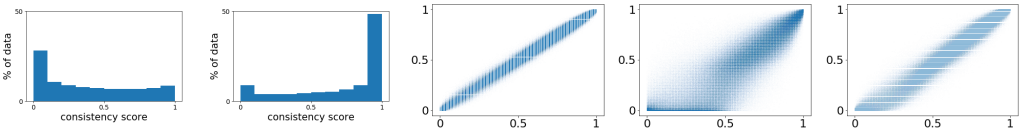
Figure 5: The distribution of consensus score[3] for 27 instances of ResNet-50 trained on ImageNet, which are used to classify the following test dataset: (a) Random classification. (b) Images generated by the random sampling of pixels from a normal distribution. (c) Image generated by StyleGAN trained on ImageNet. (d) Natural Images from a different dataset - Indoor Scene Recognition (Quattoni & Torralba, 2009). (e) ImageNet test set.

4 Cross architectures diversity

We now extend the analysis of the previous section to include networks instances which are generated by different architectures. In §4.3 we discuss comparisons with other classifiers.

4.1 Different public domain convolutional neural networks

We start by directly extending the previous analysis to two collections generated by two different architectures. Each architecture is characterized by its own learning pace, therefore it makes little sense to compare consistency epoch-wise. Instead, we first match epochs between the two collections: in a matched pair of epochs $[e_1, e_2]$, by definition the mean error of the first collection in epoch e_1 is equivalent to the mean error of the second collection in epoch e_2[4]. For each pair of epochs, we merge the corresponding collections of the two architectures and compute the consistency of the merged collection. We call this score *cross-consistency*. In Fig. 6a,b we plot the distribution of the cross-consistency score in two matched pairs of epochs, when comparing ResNet-50 and AlexNet; comparative results for additional pairs of architectures are shown in Appendix D, Figs. 23,24. As before, the dynamics of the cross-consistency is bi-modal (mean Pearson of 0.54 for test and 0.65 for train), suggesting that the network instances of both architectures learn similar classifiers.



(a) 0.39 Accuracy (b) 0.71 Accuracy (c) ResNet-ResNet (d) ResNet-AlexNet (e) ResNet-DenseNet

Figure 6: Comparing collections of networks of different architectures - ResNet-50, AlexNet, and DenseNet, trained on ImageNet. (a-b) The distribution of the *cross-consistency* score in two matching pairs of epochs. (a) Accuracy 0.39: ResNet-50 epoch 4, Alexnet epoch 30. (b) Accuracy 0.71: ResNet-50 epoch 40, Alexnet epoch 80. (c-e) The correlation of the *accessibility score* in 3 pairs of collections of different architectures trained on ImageNet. X-axis and Y-axis: *accessibility score* for the first and second collection in the pair respectively. c) Two collections of ResNet-50, 13 instances in each. (d) 27 instances of ResNet-50 and 22 instances of AlexNet. (d) 27 instances of ResNet-50 and 6 instances of DenseNet.

The distribution of the *cross-consistency* score in Fig. 6a,b implies that the two architectures, ResNet-50 and AlexNet, learn the data in the same order when trained on the same dataset. We wish to measure this implication directly. To this end, we measure for a given point and a given collection, how fast the point has been learned by all the instances in the collection. Note that the sooner a point is being learned by all network instances, the higher its average consistency should be when computed over all epochs. We therefore call a point's average consistency its *accessibility score*, and correlate this score across two collections to compare the order of learning.

We start by correlating the *accessibility score* of two collections generated by the same architecture. When comparing two collections of ResNet-50, the correlation is almost 1 ($r = 0.99,\ p \leq 10^{-50}$, Fig. 6c). When comparing two collections of two different architectures: ResNet-50 and AlexNet

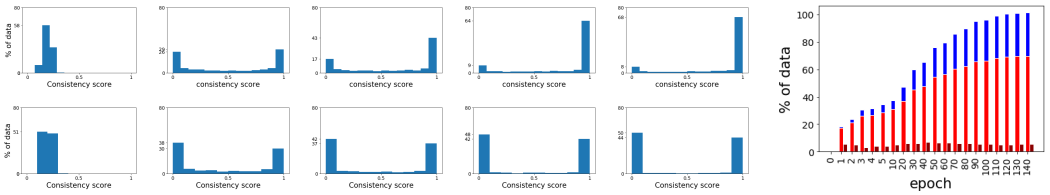
[4]Equivalence is determined up to a tolerance of $\pm 1\%$; results are not sensitive to this value.

($r = 0.87,\ p \leq 10^{-50}$, Fig. 6d) or ResNet-50 and DenseNet ($r = 0.97,\ p \leq 10^{-50}$, Fig. 6e), once again the correlation is high. These results are quite surprising given how the error rates of the three architectures differ: AlexNet Top-1 error: 0.45, ResNet-50 Top-1 error: 0.24, DenseNet Top-1 error: 0.27. The results of comparing additional pairs of competitive ImageNet architectures are shown in Appendix D, Figs. 23,24. The results have been replicated for other datasets, including: VGG19 and st-VGG on CIFAR-10 and CIFAR-100 (Fig. 25).

4.2 Linear networks

Convolutional Neural Networks where the internal operations are limited to linear operators (Oja, 1992) define an important class of CNNs, as their linearity is often exploited in the theoretical investigation of deep learning. It is natural to wonder, therefore, whether the bi-modal behavior observed in general CNNs also occurs in this case. The answer is in the affirmative.

We train 100 st-VGG networks on the small-mammals dataset (see Appendix C). By replacing all the activation layers by the identity operator, and changing the max-pooling into average-pooling, a linear CNN architecture is obtained. The performance of these linear networks is weaker (0.43 average accuracy) than the original non-linear networks (0.56 average accuracy). Still, the distribution of the consistency scores throughout the entire learning process is bi-modal (maximum Pearson: 0.055), and this bi-modality is even more pronounced than the bi-modality in the non-linear case (maximum Pearson: 0.22). The bi-modal dynamics of st-VGG can be seen in the top row of Fig. 7a, compared to the dynamics of linear st-VGG in similar epochs at the bottom row of Fig. 7a.



(a) consistency dynamics (b) performance comparison

Figure 7: Comparison of linear and non linear networks. (a) Distribution of consistency scores during the learning process using: Top - st-VGG, Bottom - linear st-VGG. Epochs shown, from left to right: 0, 10, 30, 90, 140. (b) Non-linear networks easily learn the same points that are easily learned by linear networks (in red), and more (in blue), see text.

Linear networks converge in just a few epochs, hence ruling the *cross-consistency score* uninformative in this case. Nevertheless, we still observe that linear and non-linear networks learn examples in roughly the same order, as shown in Fig. 7b. To understand Fig. 7b, we define for each epoch the set of "easiest" points, including all the points whose corresponding consistency score is larger than 0.9. For each epoch, there are two such sets: $\mathcal{C}^e_{NL}$ defined by the non-linear st-VGG, and $\mathcal{C}^e_L$ defined by the linear st-VGG. In Fig. 7b, for each epoch, the red bar depicts the number of points shared by the two sets $|\mathcal{C}^e_{NL} \cap \mathcal{C}^e_L|$, the additional blue bar depicts $|\mathcal{C}^e_{NL} \setminus \mathcal{C}^e_L|$, and the dark-red bar depicts $|\mathcal{C}^e_L \setminus \mathcal{C}^e_{NL}|$. In the beginning, the linear and non-linear variants learn roughly the same examples, while in more advanced epochs the non-linear networks continue to learn examples that remain hard for the linear networks. Moreover, since $|\mathcal{C}^e_L \setminus \mathcal{C}^e_{NL}|$ is always rather small, it follows that examples which are easy for linear networks are for the most part also easy for non-linear networks, but not vice versa. Thus, the non-linear networks classify correctly most of the examples that are classified by the linear networks and more, in agreement with the results described in §4.1.

4.3 Cross architectures diversity - other learning paradigms

Up to now, we investigated a variety of neural network architectures, revealing a common learning pattern. Can we see the same commonalities with other classification methods and paradigms? First, we consider boosting based on linear classifiers as weak learners, because the training of both neural networks and AdaBoost share a dynamic aspect: in neural networks training accuracy increases with time due to the use of GD, while in AdaBoost accuracy increases over time due to the accumulation

of weak learners. We find that in both dynamics, there are commonalities in the learning order of examples. Next, we consider other machine learning paradigms, including SVM, KNN classifier, perceptron, decision tree, random forest and Gaussian naïve Bayes. Interestingly, we still find a strong correlation with the order of learning as defined above, in that these classifiers tend to fit those examples which the neural networks learn first. These results are discussed in full in Appendix F.

5 When consistency distribution is no longer bi-modal

In Section 3 we discussed the characteristic bi-modal distribution of consistency scores, illustrated in Fig. 1, which has appeared in all the experiments presented until now, in both the train and test sets. We have already seen that this bi-modality weakens as the similarity between the distributions of train and test data is reduced (see Fig. 5). In this section, we investigate the circumstances under which the bi-modal distribution of consistency scores is no longer seen.

Learning to see Gabor patches. The bi-modal distribution of consistency scores through all stages of learning is not an inherent property of neural networks. We demonstrate this point using a dataset of artificial images, consisting of Gabor patches: the dataset contains 12 overlapping classes which differ from each other in orientation, parity and color (see Appendix C). 100 instances of st-VGG have been trained to classify this data. Now the distribution of consistency scores, shown in Fig. 8, is no longer bi-modal. Rather, the distribution is approximately normal most of the time. As learning proceeds, the mean of the distribution slowly shifts towards 1, and the width of the distribution seems to expand. At convergence, the models have reached similar performance, and the bi-modal characteristics partially re-appears on the test data.

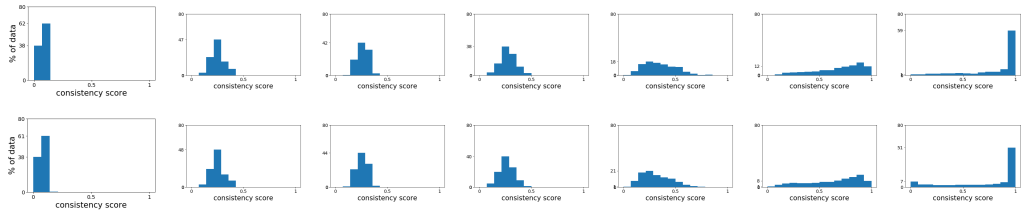


Figure 8: Artificial dataset of Gabor patches, 100 st-VGG networks. The distribution of consistency scores is shown at epochs: $0, 1, 5, 10, 20, 40, 100$. Top: train data; bottom: test data.

Random labels. Bi-modality seems to be associated with successful generalization. To see this, we take the small-mammals dataset, and reshuffle the labels such that every image is assigned a random label (following Zhang et al., 2016). In this case, training accuracy can reach 100% when disabling dropout (that acts as a regularizer), which indicates that the networks can memorize the data. Interestingly, the distribution of consistency scores in no longer bi-modal, with minimum Pearson score of 1.07 on train set and 1.35 on the test set during the entire learning process. Rather, the distribution in each epoch resembles a Gaussian centered around the mean accuracy of the networks, see Fig. 9. These results are in agreement with the results of Morcos et al. (2018), which show different network dynamics when networks perform memorization or generalization.

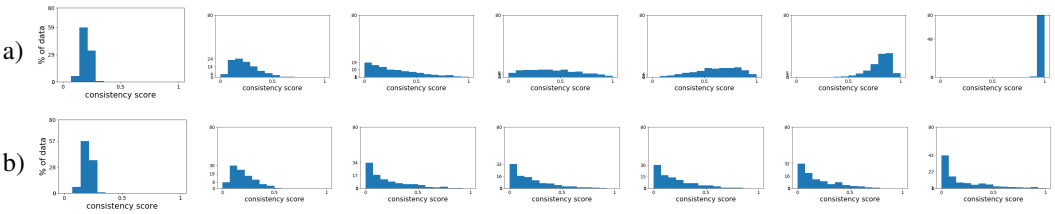


Figure 9: Small-mammals dataset, randomly shuffled labels, 100 st-VGG networks. The distribution of consistency scores is shown at epochs: $0, 1, 5, 20, 30, 40, 50$. Top: train data; bottom: test data.

Fully connected networks. Bi-modality is not an artifact of using gradient descent for optimization. This can be seen in the following analysis. Specifically, we consider a fully connected neural network architecture, with 2 intermediate layers, ELU and dropout. The networks are trained to discriminate points sampled from two largely overlapping Gaussian distributions in high dimension. The dynamic distribution of consistency scores is shown in Fig. 10, and resembles the distribution of independent networks shown in Fig. 1c. While the final solutions of the networks are similar, the order in which examples are being learned when employing SGD optimization is different.

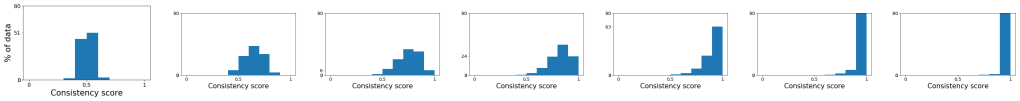


Figure 10: Consistency distribution of 100 fully connected networks on test data. Epochs shown: $0, 1, 2, 3, 5, 10, 20$.

6 Summary and discussion

We empirically show that neural networks learn similar classification functions. More surprisingly with respect to earlier work, the learning dynamics is also similar, as they seem to learn similar functions also in intermediate stages of learning, before convergence. This is true for a variety of architectures, including different CNN architectures and LSTMs, irrespective of size and other hyper-parameters of the learning algorithms. We have verified this pattern of results using many different CNN architectures, including most of those readily available in the public domain, and many of the datasets of natural images which are in common use when evaluating deep learning.

The similarity of network instances is measured in the way they classify examples, including known (train) and new examples. Typically, the similarity over test data is as pronounced as it is over train data, as long as the train and test examples are sampled from the same distribution. We show that this similarity extends also to out of sample test data, but it seems to decrease as the gap between the distribution of the train data and the test data is increased.

This pattern of similarity crosses architectural borders: while different architectures may learn at a different speed, the data is learned in the same order. Thus all architectures which reach a certain error rate seem to classify, for the most part, the same examples in the same manner. We also see that stronger architectures, which reach a lower generalization error, seem to start by first learning the examples that weaker architectures classify correctly, followed by the learning of some more difficult examples. This may suggest that the order in which data is learned is an internal property of the data.

We also discuss cases where this similarity breaks down, indicating that the observed similarity is not an artifact of using stochastic gradient descent. Rather, the observed pattern of similarity seems to characterize the learning of complex problems with big networks. Curiously, the deeper the network is and the more non-linearities it has, and even though the model has more learning parameters, the progress of learning in different network instances becomes more similar to each other. Un-intuitively, this suggests that in a sense the number of degrees of freedom in the learning process is reduced, and that there are fewer ways to learn the data. This effect seems to force different networks, as long they are deep enough, to learn the dataset in the same way. This counter-intuitive result joins other non-intuitive results, like the theoretical result that a deeper linear neural network converges faster to the global optimum than a shallow network (Arora et al., 2018).

We also show that the observed pattern of similarity is strongly correlated with effective generalization. What does it tell us about the generalization of neural networks, a question which is considered by many to be poorly understood? Neural networks can memorize an almost limitless number of examples, it would seem. To achieve generalization, most training protocols employ some regularization mechanism which does not allow for unlimited data memorization. As a result, the network fits only the train and test examples it would normally learn first, which are, based on our analysis, also the "easier" (or more typical) examples. We hypothesize that this may explain why a regularized network discovers robust solutions, with little variability among its likely instances.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 233–242. JMLR. org, 2017.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face and Gesture Recognition*, 2018.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. *arXiv preprint arXiv:1904.03626*, 2019.

Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9: 1735–1780, 1997.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

David Krueger, Nicolas Ballas, Stanislaw Jastrzebski, Devansh Arpit, Maxinder S Kanwal, Tegan Maharaj, Emmanuel Bengio, Asja Fischer, and Aaron Courville. Deep nets don't learn via memorization. 2017.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pp. 1189–1197, 2010.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 991–999, 2015.

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *Iclr*, 2016.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems*, pp. 5727–5736, 2018.

Erkki Oja. Principal components, minor components, and linear neural networks. *Neural networks*, 5 (6):927–935, 1992.

Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110, 1894.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

Public domain dataset a. Stack overflow bigquery program language dataset. Online: `https://storage.googleapis.com/tensorflow-workshop-examples/stack-overflow-data.csv`, 2019. Accessed: 2019-09-24.

Public domain dataset b. Tiny imagenet challenge. Online: `https://tinyimagenet.herokuapp.com`, 2019. Accessed: 2019-05-22.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420. IEEE, 2009.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, pp. 6076–6085, 2017.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

Andrew I Schein and Lyle H Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. In *Advances in Neural Information Processing Systems*, pp. 9584–9593, 2018.

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

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

A Related work

How deep neural networks generalize is an open problem (Kawaguchi et al., 2017). The expressiveness of NNs is broad (Cybenko, 1989), and they can learn any arbitrary complex function (Hornik et al., 1989). This extended capacity can indeed be reached, and neural networks can memorize datasets with randomly assigned labels (Zhang et al., 2016). Nevertheless, the dominant hypothesis today is that in natural datasets they "prefer" to learn an easier hypothesis that fits the data rather than memorize it all (Zhang et al., 2016; Arpit et al., 2017). Our work is consistent with a hypothesis which requires fewer assumptions, see Section 6.

The direct comparison of neural representations is regarded to be a hard problem, due to a large number of parameters and the many underlying symmetries. Many non-direct approaches are available in the literature: (Li et al., 2016; Wang et al., 2018) compare subsets of similar features across multiple networks, which span similar low dimensional spaces, and show that while single neurons can vary drastically, some features are reliably learned across networks. (Raghu et al., 2017) proposed the SVCCA method, which can compare layers and networks efficiently, with an amalgamation of SVD and CCA. They showed that multiple instances of the same converged network are similar to each other and that networks converge in a bottom-up way, from earlier layers to deeper ones. Morcos et al. (2018) builds off the results of (Raghu et al., 2017), further showing that networks which generalize are more similar than ones which memorize, and that similarity grows with the width of the network.

In various machine learning methods such as curriculum learning (Bengio et al., 2009), self-paced learning (Kumar et al., 2010) and active learning (Schein & Ungar, 2007), examples are presented to the learner in a specific order (Hacohen & Weinshall, 2019; Jiang et al., 2017). Although conceptually similar, here we analyze the order in which examples are learned, while the aforementioned methods seek ways to alter it. Likewise, the design of effective initialization methods is a striving research area (Erhan et al., 2010; Glorot & Bengio, 2010; Rumelhart et al., 1988). Here we do not seek to improve these methods, but rather analyze the properties of a collection of network instances generated by the same initialization methodology.

B Architectures

In addition to the public domain architectures described in §3, we also experimented with some handcrafted networks. Such networks are simpler and faster to train, and are typically used to investigate the learning of less commonly used datasets, such as the small-mammals dataset and tiny ImageNet. Below we list all the architectures used in this paper.

st-VGG. A stripped version of VGG which we used in many experiments. It is a convolutional neural network, containing 8 convolutional layers with 32, 32, 64, 64, 128, 128, 256, 256 filters respectively. The first 6 layers have filters of size 3×3, and the last 2 layers have filters of size 2×2. Every second layer there is followed by 2×2 max-pooling layer and a 0.25 dropout layer. After the convolutional layers, the units are flattened, and there is a fully-connected layer with 512 units followed by 0.5 dropout. When training with random labels, we removed both dropout layers to enable proper training, as suggested in Krueger et al. (2017). The batch size we used was 100. The output layer is a fully connected layer with output units matching the number of classes in the dataset,

followed by a softmax layer. We trained the network using the SGD optimizer, with cross-entropy loss. When training st-VGG, we used a learning rate of 0.05 which decayed by a factor of 1.8 every 20 epochs.

Small st-VGG. To compare st-VGG with another architecture, we created a smaller version of it: we used another convolutional neural network, containing 4 convolutional layers with 32, 32, 64, 64 filters respectively, with filters of size 3×3. Every second layer there is followed by 2×2 max-pooling and a 0.25 dropout layer. After the convolutional layers, the units are flattened, and there is a fully-connected layer with 128 units followed by 0.5 dropout. The output layer is a fully connected layer with output units matching the number of classes in the dataset, followed by a softmax layer. We trained the network using the SGD optimizer, with cross-entropy loss. We trained this network with the same learning rate and batch size as st-VGG.

MNIST architecture. When experimenting with the MNIST dataset, we used some arbitrary small architecture for simplicity, as most architectures are able to reach over 0.99 accuracy. The architecture we used had 2 convolutional layers, with 32 and 64 filters respectively of size 3×3. After the convolutions, we used 2×2 max-pooling, followed by 0.25 dropout. Finally, we used a fully connected layer of size 128 followed by 0.5 dropout and Softmax. We used a learning rate of 1 for 12 epochs, using AdaDelta optimizer and a batch size of 100.

Fully connected architecture. When experimenting with fully connected networks, we used a 4 layers network, which simply flattened the data, followed by 2 fully connected layers with 1024 units, followed byn an output layer with softmax. We used 0.5 dropout after each fully connected layer. Since these networks converge fast, a wide range of learning rates can be used. Specifically, we used 0.04. We experimented with a wide range of numbers of fully connected layers, reaching similar results.

BiLSTM with Attention. When experimenting on textual data we used a GloVe embeddings, a layer of BiLSTM of size 300, 0.25 dropout and recurrent dropout, an attention layer, a fully connected layer of size 256 with 0.25 dropout and a last fully connected layer to extract output. The networks were optimized using Adam optimization with a learning rate of 0.005 and a batch size of 256.

C DATASETS

Small Mammals. The small-mammals dataset used in the paper is the relevant super-class of the CIFAR-100 dataset. It contains 2500 train images divided into 5 classes equally, and 500 test images. Each image is of size $32 \times 32 \times 3$. This dataset was chosen due to its small size, which allowed for efficient experimentation. All the results observed in this dataset were reproduced on large, public domain datasets, such as CIFAR-100, CIFAR-10, and ImageNet.

Insect. Similarly to the small mammals dataset, the relevant super-class of CIFAR-100.

Fish. Similarly to the small mammals dataset, the relevant super-class of CIFAR-100.

Cats and Dogs. The cats and dogs dataset is a subset of CIFAR-10. It uses only the 2 relevant classes, to create a binary problem. Each image is of size $32 \times 32 \times 3$. The dataset is divided to 20000 train images (10000 per class) and 2000 test images (1000 per class).

Gabor. The Gabor dataset used in the paper, is a dataset we created which contains 12 classes of Gabor patches. Each class contains 100 images of Gabor patches which vary in size and orientation. Classes differ from each other in 3 parameters: 1) Parity - there is a different class for odd and even Gabor patches (corresponding to the use of sine or cosine respectively). 2) RGB channel - each class depicts the Gabor patch in a single RGB channel. 3) Orientation - each class can have one of the following base orientations: $45°, 90°, 135°, 180°$. The orientation of each class varies by $\pm 30°$, making some of the classes non-separable, while some classes are. Code for creating this dataset will be published upon acceptance.

Gaussian. The Gaussian dataset used in the fully connected case, is a 2-classes dataset. One class is sampled from a multivariate Gaussian with mean 0 and $\Sigma = I$, while the other class is sampled from a multivariate Gaussian with mean 0.1 and $\Sigma = I$. Other choices for the mean and variance yield similar results. Each sampled vector was of dimension 3072, and then reshaped to $32 \times 32 \times 3$ to resemble the shape of CIFAR images. Each class contained 2500 train images and 500 test images.

VGGFace2 subset. We created a classification task for face recognition, using a subset of 10 classes from VGGFace2. We chose the classes containing the largest number of images. We chose 600 images from each class arbitrarily to be the train set, while the remaining points (between 89 and 243) served as the test set. Each image was resized to $64 \times 64 \times 3$, using center cropping while maintaining aspect ratio.

Stack Overflow. The data from Stack Overflow is publicly shared and used for tutorials. It contains 39K training samples and 1K test samples, each tagged with one of 20 programming languages as the language the question asks about. Each question must be regarded more as a paragraph than a sentence. Many words, terms and symbols are expected to be domain-dependent, and therefore under-represented in the embeddings.

D ROBUSTNESS OF RESULTS

Similar qualitative results were obtained in all the experiments with natural datasets. To maintain a fair comparison across epochs, the results for each shown epoch e (effectively epoch extent) were obtained by independently training a different set of N networks from scratch for e epochs. The specific set of epochs $\mathcal{S}_E$, where $|\mathcal{S}_E| = 7$, that was used in each plot was determined arbitrarily, to evenly span all sections of learning. All the networks in all test cases converged before the final epoch plotted.

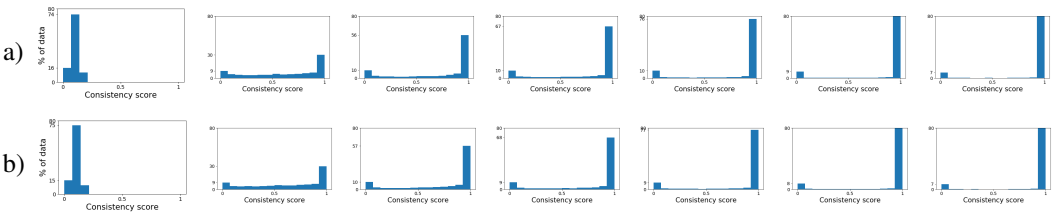


Figure 11: The distribution of consistency scores during the learning process of 100 instances of small architecture (see Appendix B) trained on MNIST. Epochs shown: $0, 1, 2, 3, 5, 10, 20$. We used a low learning rate (0.001) to avoid convergence after one epoch. a) Train set; b) Test set.

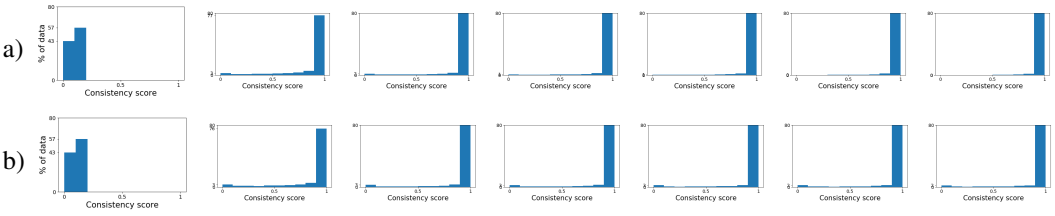


Figure 12: The distribution of consistency scores during the learning process of 100 instances of st-VGG (see Appendix B) trained on Fashion-MNIST. Epochs shown: $0, 1, 5, 10, 15, 20, 25$. a) Train set; b) Test set.

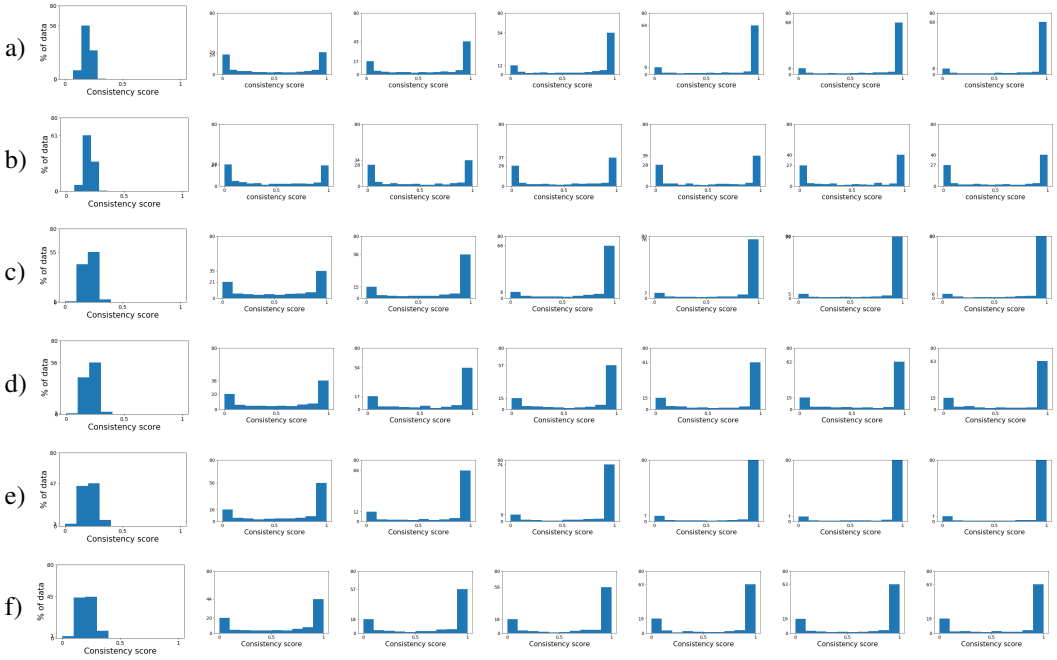


Figure 13: The distribution of consistency scores during the learning process of 100 instances of st-VGG (see Appendix B) on super-classes of CIFAR-100. Epochs shown: $0, 10, 30, 60, 90, 120, 140$. a-b) Train and test sets of the fish dataset respectively; c-d) train and test sets of the insect dataset respectively; e-f) train and test sets of the small mammals dataset respectively.

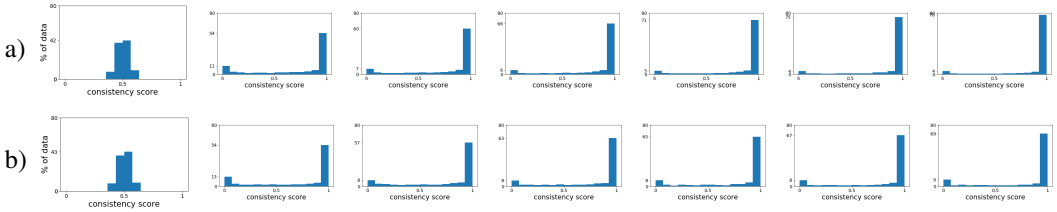


Figure 14: The distribution of consistency scores during the learning process of 100 instances of st-VGG (see Appendix B) trained on the cats and dogs binary dataset. Epochs shown: $0, 10, 30, 60, 90, 120, 140$. a) Train set; b) Test set.

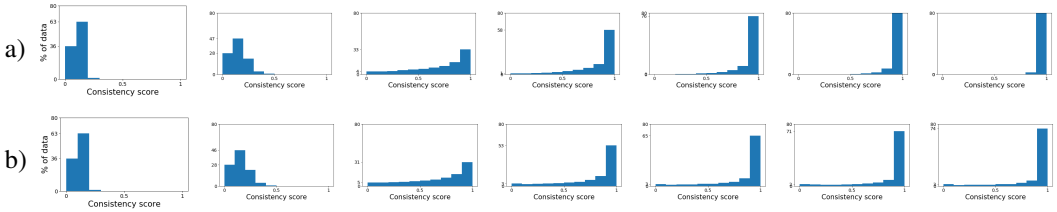


Figure 15: The distribution of consistency scores during the learning process of 19 instances of VGG19, trained on CIFAR-10. Epochs shown: $0, 1, 10, 30, 60, 80, 100$. a) Train set; b) Test set.

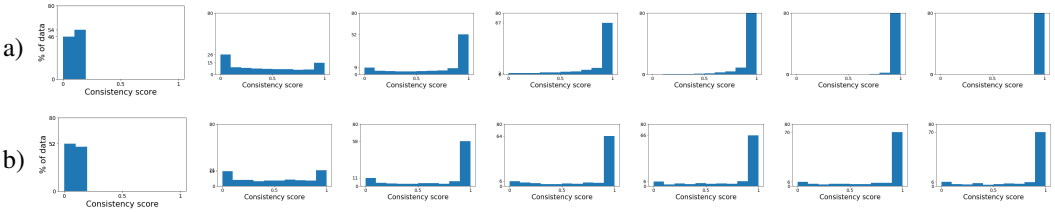


Figure 16: The distribution of consistency scores during the learning process of 20 instances of st-VGG (see Appendix B) trained on the face classification task (see Appendix C). Epochs shown: $0, 1, 10, 20, 30, 40, 60$. a) Train set; b) Test set.

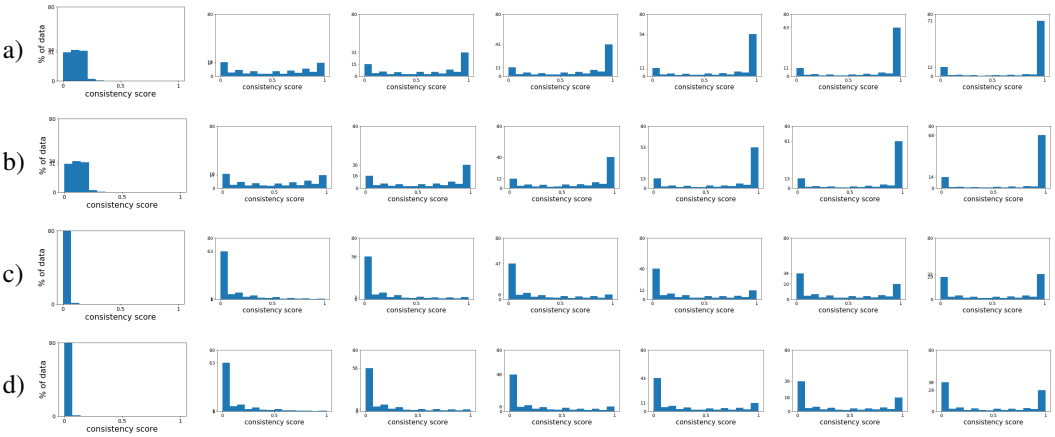


Figure 17: The distribution of consistency scores during the learning process of 100 instances of st-VGG (see Appendix B) trained on CIFAR-10 and CIFAR-100. Epochs shown: $0, 1, 2, 5, 10, 20, 40$. a-b) Train and test sets of CIFAR-10 respectively; c-d) train and test sets of CIFAR-100 respectively.

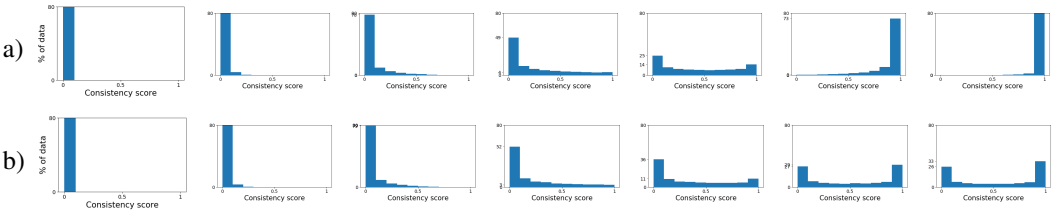


Figure 18: The distribution of consistency scores during the learning process of 100 instances of st-VGG (see Appendix B) trained on Tiny ImageNet. Epochs shown: $0, 1, 5, 10, 20, 50, 70$. a) Train set; b) Test set.

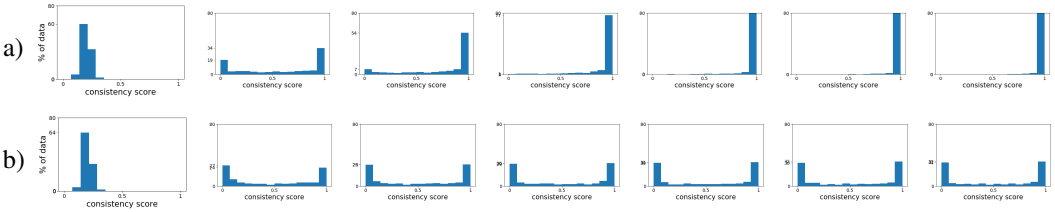


Figure 19: The distribution of consistency scores during the learning process of 100 instances of small st-VGG (see Appendix B) trained on the small-mammals dataset (see Appendix C). Epochs shown: $0, 10, 30, 60, 90, 120, 140$. a) Train set; b) Test set.

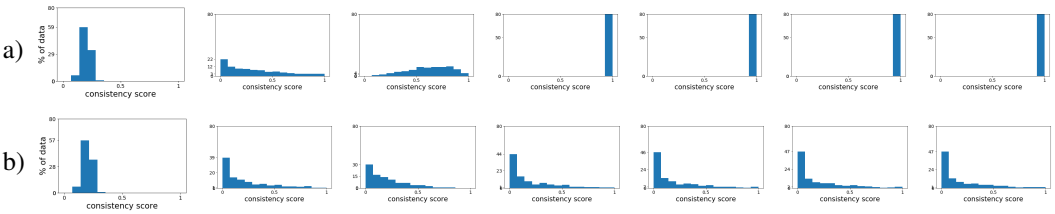


Figure 20: The distribution of consistency scores during the learning process of 100 instances of st-VGG (see Appendix B) trained on the small-mammals dataset (see Appendix C), where labels were assigned randomly to images, as done in Zhang et al. (2016). Epochs shown: $0, 10, 30, 60, 90, 120, 140$. a) Train set; b) Test set.

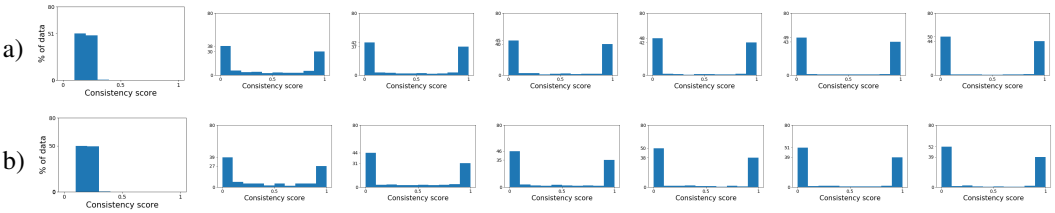


Figure 21: The distribution of consistency scores during the learning process of 100 instances of linear st-VGG (see Appendix B) trained on the small-mammals dataset (see Appendix C). Epochs shown: $0, 10, 30, 60, 90, 120, 140$. a) Train set; b) Test set.

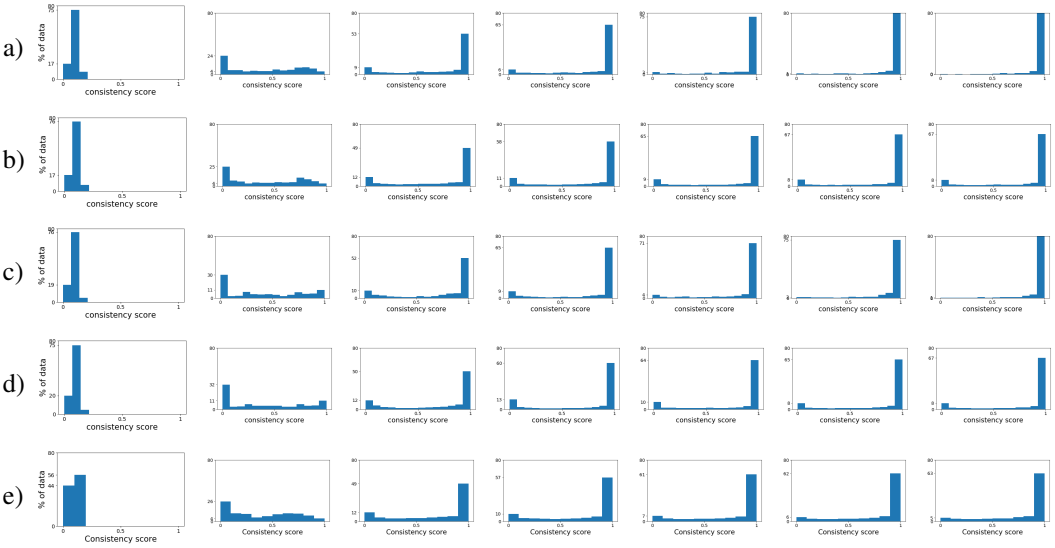


Figure 22: The distribution of consistency scores during the learning process of 100 instances of st-VGG (see Appendix B) trained on the parts of Fashion-Mnist. We divided the train set of Fashion-Mnist into 60 parts of 1000 images. Epochs shown: $0, 1, 5, 10, 20, 30, 40$. a-b) Train and test sets of 100 instances trained on a random part of Fashion-Mnist; c-d) train and test sets of 100 instances trained on another random part of Fashion-Mnist. e) The average distribution of all 60 collections on the Fashion-Mnist test set. The bi-modality presented here indicates that although learned on different training sets, all collections have similar learning order.

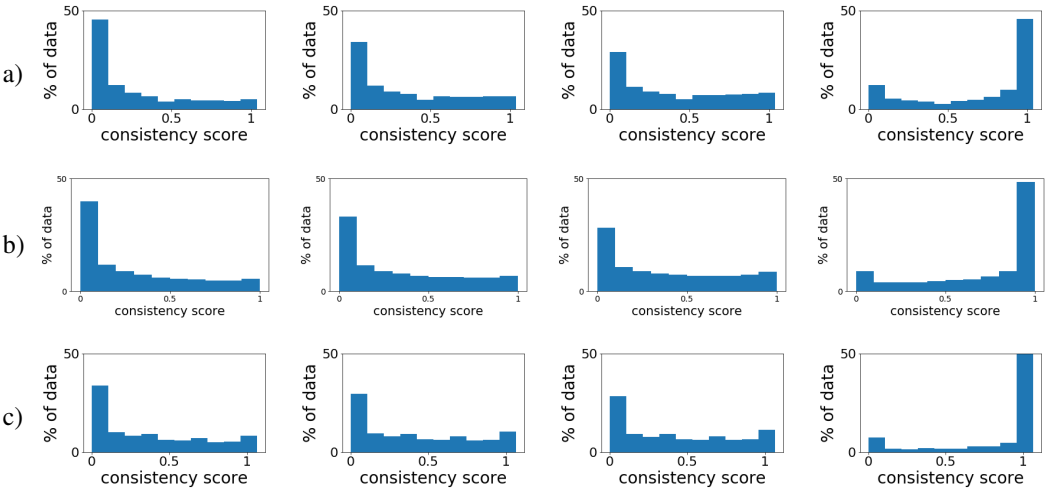


Figure 23: Averaged consistency dynamics of pairs of collections of ImageNet architectures. a) 6 DenseNet instances with 22 AlexNet instances, plotted on accuracies of $26\%, 33\%, 38\%, 67\%$. b) 27 ResNet-50 instances with 22 AlexNet instances, plotted on accuracies of $28\%, 33\%, 38\%, 70\%$. c) 27 ResNet-50 instances with 6 DenseNet, plotted on accuracies of $35\%, 39\%, 40\%, 83\%$. These results follow the same protocol described in Fig. 6.

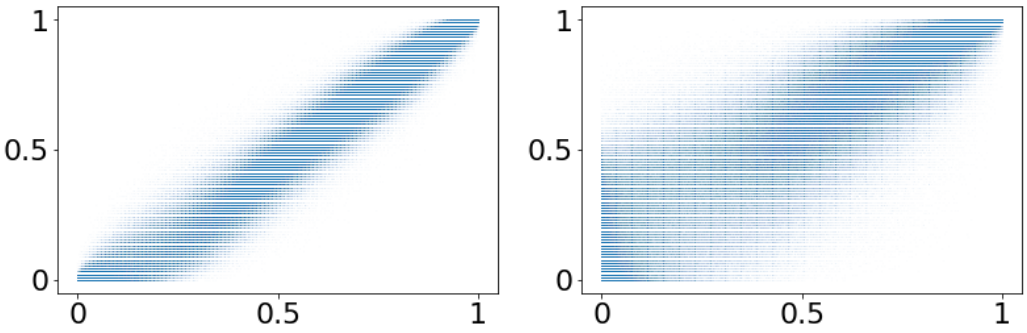
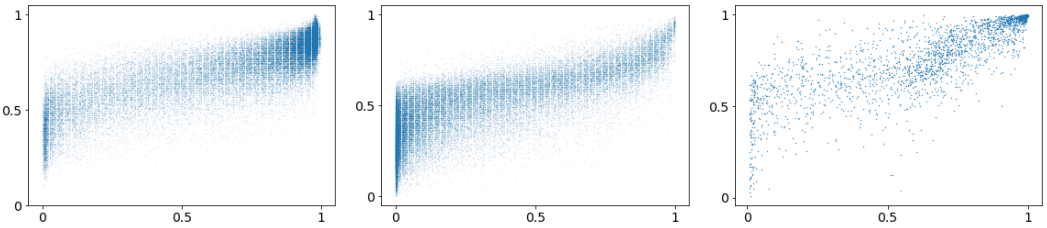


Figure 24: The correlation of the *accessibility score* in 2 pairs of collections of different architectures trained on ImageNet. Left: 27 instances of ResNet-50 and 6 instances of DenseNet, $r = 0.97, p < 10^{-50}$. Right: 22 instances of AlexNet and 6 instances of DenseNet $r = 0.87, p < 10^{-50}$. These results follow the same protocol described in Fig. 6.



(a) CIFAR-10 (b) CIFAR-100 (c) Small mammals

Figure 25: The correlation of the *accessibility score* of different architectures train on the same dataset. a) 100 instances of st-VGG and 19 instances of VGG19 trained on CIFAR-10. $r = 0.83, p < 10^{-50}$. b) 100 instances of st-VGG and 20 instances of VGG19 trained on CIFAR-100. $r = 0.78, p < 10^{-50}$. c) 100 instances of st-VGG and 100 instances of small st-VGG on the samll-mammals dataset $r = 0.82, p < 10^{-50}$. These results follow the same protocol described in Fig. 6.

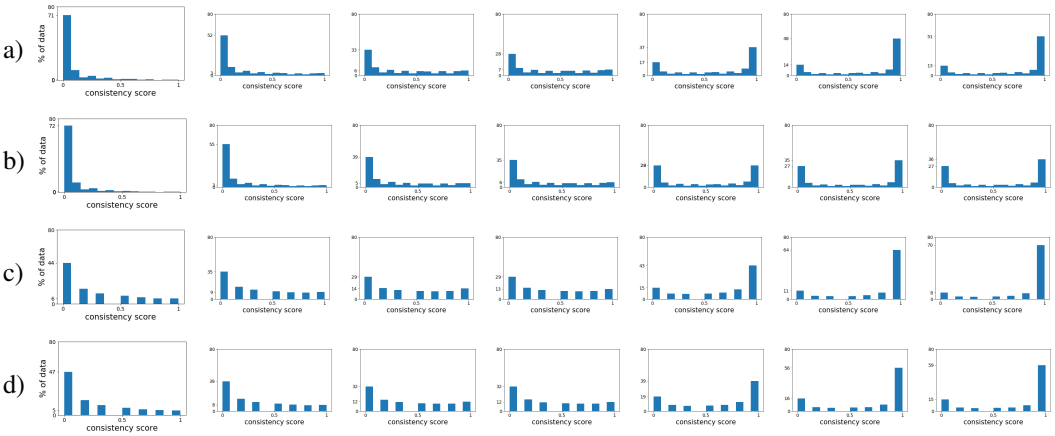
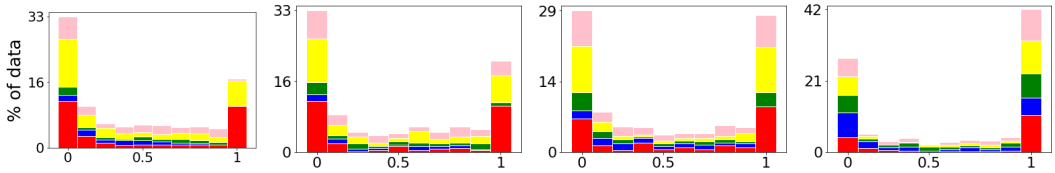


Figure 26: The distribution of consistency scores of different architectures over ImageNet. Epochs shown: $1, 2, 10, 30, 50, 70, 100$. a-b) 22 instances of AlexNet over the train and test sets respectively. c-d) 6 instances of DenseNet over the train and test sets respectively. These results follow the same protocol described in Fig. 1.

E ADDITIONAL RESULTS

Induced class hierarchy. The ranking of training examples induced by the consistency scores typically induces a hierarchical structure over the different classes as well. To see this, we train 100 instances of st-VGG on the small-mammals dataset, and calculate for each image the most frequent class label assigned to it by the collection of networks. In Fig. 27 we plot the histogram of the consistency score (as in Fig. 1), but this time each image is assigned a color, which identifies its most frequent class label (1 of 5 colors). It can be readily seen that at the beginning of learning, only images from 2 classes reach a consistency score of 1. As learning proceeds, more class labels slowly emerge. This result suggests that classes are learned in a specific order, across all networks. Moreover, we can see a pattern in the erroneous label assignments, which suggests that the classifiers initially use fewer class labels, and only become more specific later on in the learning process.



(a) Epoch 1 (b) Epoch 2 (c) Epoch 30 (d) Epoch 140

Figure 27

Dynamics of individual image consistency. We now focus on consistency scores of individual images, as they evolve throughout the entire learning process. For most examples, the score may climb up from random (0.2) to 1 in 1 epoch, it may dip down to 0 and then go up to 1 after a few epochs, or it may go rapidly down to 0. Either, the score remains 1 or 0. These patterns are shown in Fig. 29, and support the bi-modality results we report above. The duration in which a certain example maintains a consistency score 0 correlates with the order of learning: the longer it has 0 consistency, the more difficult it is. A minority of the training examples exhibit different patterns of learning. For example, a few images (the green curve in Fig. 29) begin with a high consistency score (near 1), but after a few epochs their score drops to 0 and remains there.

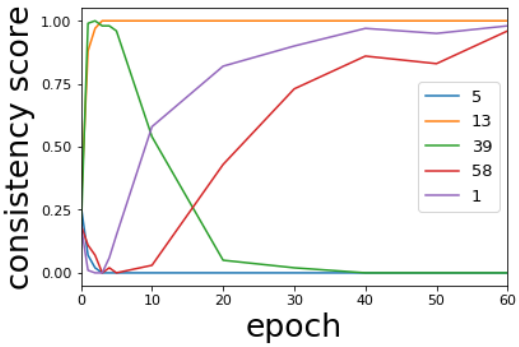


Figure 28: consistency score over time

Figure 29: Consistency score as a function of the epoch, of 5 example images. Results achieved when training st-VGG on the small-mammals dataset.

Diversity in single architecture. The bi-modal phase we report in section 3, was seen in all unmodified datasets we tested, across all architectures. Specifically, we've tested ImageNet on AlexNet ($N = 22$), ResNet-50 ($N = 27$), DenseNet ($N = 7$). Mnist on the Mnist architecture (see Appendix B) with $N = 100$, CIFAR-10 and CIFAR-100 with VGG-16 ($N = 20$) and st-VGG ($N = 100$), tiny ImageNet with st-VGG ($N = 100$), small-mammals dataset with st-VGG ($N = 100$) and small st-VGG ($N = 100$), and finally randomly picked super-classes of CIFAR-100, specifically

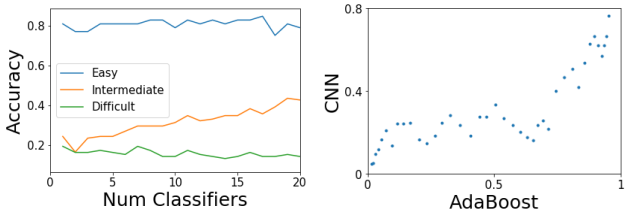
"aquatic-mammals", "insects" and "household furniture" with st-VGG ($N = 100$). The number of instances N is chosen according to our computational capabilities. However, in all cases, picking much smaller N suffice to yield the same qualitative results.

In addition to hyper-parameters which may differ between various architectures, we also experimented with changing the hyper-parameters of st-VGG trained on the small-mammals dataset, always observing the same qualitative result. All experiments used $N = 100$ instances. Specifically, we tried a large range of learning rates, learning rate decay, SGD and Adam optimizers, large range of batch sizes, dropout and L2-regularization.

Cross architectures diversity. In addition to the results in section 4, the same qualitative results were obtained for all 2 architectures we trained on the same unmodified dataset. We conducted the following experiments: ImageNet dataset: ResNet-50 vs DenseNet, AlexNet vs DenseNet. Aquatic-mammals and small-mammals super-classes of CIFAR-100: st-VGG vs small st-VGG, Tiny ImageNet: st-VGG vs small st-VGG, CIFAR-10 and CIFAR-100: VGG19 vs st-VGG. All of which yielding similar results to the ones analyzed in section 4.

F OTHER LEARNING PARADIGMS

Boosting linear classifiers. We use AdaBoost (Hastie et al., 2009) with k weak linear classifiers, trained on the small-mammals dataset. As commonly observed, adding more classifiers (increasing k) improves the final performance. In Fig. 30a we plot accuracy for 3 groups of examples - easy, intermediate and difficult, where grouping is based on the consistency score of the CNN described above. The accuracy of boosting over easy examples is significantly higher than the general accuracy and does not improve as we increase k. This result suggests that most of the easy examples can be classified linearly, and are learned first by both boosting and our CNNs. On the other hand, accuracy when classifying difficult examples is worse than the general accuracy and slowly decreases with k. For intermediate difficulty, the accuracy significantly improves with k. This suggests that the addition of classifiers to the AdaBoost enables the learning of some examples of intermediate difficulty. Overall, the order in which NNs learn the data is positively correlated with the order of AdaBoost, see Fig. 30b.



(a) (b)

Figure 30: (a) Adaboost accuracy as a function of the number of classifiers, for easy, intermediate, and hard examples as grouped by the consistency score of a CNN. (b) Correlation between the measured difficulty based on Adaboost (X-axis) and CNN (Y-axis), with $r = 0.83,\ p \leq 10^{-10}$.

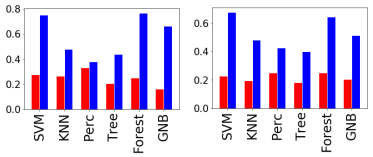


Figure 31: Fraction of correctly classified examples by each classifier, among all examples that are classified correctly (blue) and incorrectly (red) by a CNN. Left: the beginning of learning (epoch 1), right: end of learning (epoch 140). The fraction is much higher among the easier examples.

Other classifiers. We train several classifiers on the small-mammals dataset: SVM (accuracy: 0.48), KNN classifier (accuracy: 0.36), perceptron (accuracy: 0.35), decision tree (accuracy: 0.31), random forest (accuracy: 0.48) and Gaussian naïve Bayes classifier (accuracy: 0.38). All classifiers under-perform the CNN architecture described above (accuracy: 0.56). As in the case of boosting, for each method, most of the examples which are classified correctly are among the first to be learned by the CNN architecture. In other words, for all these methods, which achieve low accuracy, it is the easier examples which are being learned for the most part. This is illustrated in Fig. 31, demonstrating the phenomenon that during the learning process, most of the examples that are being learned by other paradigms are the ones already learned by neural networks.