# Let's Agree to Agree: Neural Networks Share Classification Order on Real Datasets

Guy Hacohen [1 2] Leshem Choshen [1] Daphna Weinshall [1]

## Abstract

We report a series of robust empirical observations, demonstrating that deep Neural Networks learn the examples in both the training and test sets in a similar order. This phenomenon is observed in all the commonly used benchmarks we evaluated, including many image classification benchmarks, and one text classification benchmark. While this phenomenon is strongest for models of the same architecture, it also crosses architectural boundaries – models of different architectures start by learning the same examples, after which the more powerful model may continue to learn additional examples. We further show that this pattern of results reflects the interplay between the way neural networks learn benchmark datasets. Specifically, when fixing the architecture, we describe synthetic datasets for which this pattern is no longer observed. When fixing the dataset, we show that other learning paradigms may learn the data in a different order. We hypothesize that our results reflect how neural networks discover structure in natural datasets.

## 1. Introduction

Typically, neural networks (NN) in common use include a large number of parameters and non-linear operations, resulting in a highly non-convex, high-dimensional optimization landscape. These models are usually trained with some variant of Stochastic Gradient Descent (SGD), initialized randomly, thus introducing stochasticity into the training procedure both in the form of initialization and the sampling of gradients. As a result, training the same neural architecture several times generates models with drastically different weights (Li et al., 2015; Yosinski et al., 2015), which presumably correspond to different local minima of the optimization landscape.

[1]School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel [2]Edmond & Lily Safra Center for Brain Sciences, The Hebrew University of Jerusalem, Jerusalem, Israel. Correspondence to: Guy Hacohen <guy.hacohen@mail.huji.ac.il>, Leshem Choshen <leshem.choshen@mail.huji.ac.il>, Daphna Weinshall <daphna@cs.huji.ac.il>.

*Proceedings of the 37th International Conference on Machine Learning*, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

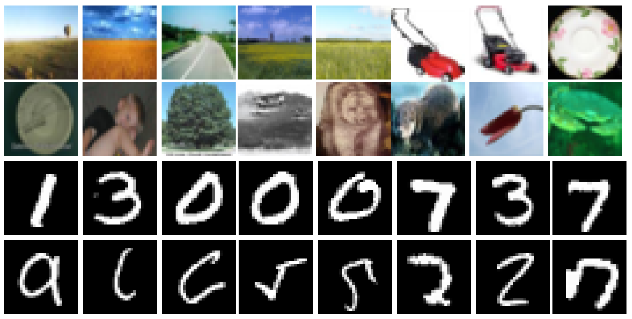


*Figure 1.* Images learned at the beginning (top row) and the end of the training (bottom row). Top: CIFAR-100, bottom: MNIST.

In spite of this observation and for most practical purposes, models trained with the same dataset, architecture and training protocols are typically considered similar by practitioners, as they tend to achieve similar accuracy. Neural comparison methods (Li et al., 2016; Raghu et al., 2017; Morcos et al., 2018) find them to be more similar than models trained on different datasets, using different comparison scores. On the other hand, many applications consider such models to be distinct enough to merit the use of ensemble methods (Le & Yang, 2015; Kantor et al., 2019) and average training epochs (Vaswani et al., 2017; Junczys-Dowmunt et al., 2018). All in all, the question of how to meaningfully evaluate the similarity between trained neural models remains open.

Contemporary approaches tend to measure the similarity between neural models by comparing their underlying compositional properties, as done for example in Lenc & Vedaldi (2015); Alain & Bengio (2016); Li et al. (2016); Raghu et al. (2017); Wang et al. (2018); Cohen et al. (2019). Instead, we propose to measure similarity via the direct comparison of their classification predictions per example, as formally defined in §2. Thus we show empirically that for a wide range of classification benchmarks, including ImageNet (Deng et al., 2009), CIFAR (Krizhevsky & Hinton, 2009), and text classification (see §3.2), models of the same architecture classify the data similarly (see §3) during the entire learning process. As long as the models share the same architecture, this similarity is independent of such choices as optimization and initialization methods, hyper-parameter values, the detailed architecture or the particular dataset. The similarity can be replicated for a given test set even when each model is trained on a different training set, as long as both sets of training data are sampled from the same distribution (see §3.3).

The similarity between different models is not restricted to their accuracy at the end of the training, but rather can be observed throughout the entire learning process (see §3). Specifically, trained models exhibit a similar classification profile in every epoch from the beginning to the end of the training. Combined with the observation that once an example is classified correctly by some model it is rarely misclassified after further training (see Suppl E), we conclude that each architecture learns to classify each benchmark dataset in a specific order, which can be seen in all its models. See examples for CIFAR-100 and MNIST in Fig. 1.

The order in which a dataset is learned seems robust across architectures. In §4 we train several commonly used architectures on the same dataset, resulting in a highly correlated learning order, in spite of there being significant differences in the final accuracy of the architectures. In fact, when tracking the training process of different architectures, we observe that the models of the stronger architecture first learn the examples which have been learned by those the weaker architecture, and only then continue to learn new examples.

Is it possible that this robust similarity is an artifact of the training procedure of NNs, and specifically the use of SGD optimization? In §5 we describe examples of hand-crafted image datasets where these patterns of similarity disappear, suggesting that this is not the case. In these synthetic datasets, which NNs learn successfully, even models from the same architecture seem to learn to classify examples in a different order, depending on their random initialization and mini-batch sampling. Moreover, when training NNs on a dataset with randomly shuffled labels (Zhang et al., 2016), in which no generalization is possible, we find that different models memorize the data in a different order[1].

Is it possible that this robust similarity is an artifact of the structure of each dataset, and the way it is being discovered? In other words, is it all about typical points being learned before atypical points, regardless of the nature of the classifier? We show that this is not the case, and that the order by which benchmark datasets are learned is unique to NNs. Specifically, in §6 we analyze the order in which a benchmark dataset is learned by an AdaBoost classifier that employs weak linear classifiers. We show that this order has a low correlation with the order observed when training NNs on the same dataset.

The empirical observations described above seem to echo the interplay between how NNs learn, and the complexity of datasets which are used to evaluate these NNs. Possibly they also reflect the way NNs discover structure in a given dataset, where learning order corresponds with data complexity. In other words, some examples are consistently easier than others for NNs to learn. (We discuss the relationship with curriculum learning and hard data mining in §7.) Notably, we failed to find a real dataset for which NNs differ. This may indicate the existence of a common structure in real datasets, which our synthetic datasets do not exhibit.

### 1.1. Summary of Contribution

- We propose a direct way to compare between different neural models using the *Tp-agreement* score (§2).
- We empirically show that models that share the same architecture learn real datasets in the same order (§3), and propose a measure to this effect termed *Accessibility* score (§2).
- We argue that the learning order emerges from the coupling of neural architectures and benchmark datasets. To support shis, we show that neural architectures can learn synthetic datasets without any specific order (§5), and likewise non-neural architectures can learn benchmark datasets without any specific order (§6).
- We show that models with different architectures can learn benchmark datasets at a different pace and performance, while still inducing a similar order (§4). Specifically, we see that stronger architectures start off by learning the same examples that weaker networks learn, then move on to learning new examples.

## 2. Methodology and Notations

Consider some 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 example and $y_i \in [K]$ its corresponding label. We create a collection of $N$ models of $f$, denoted $\mathcal{F}^e = \{f_1^e, ..., f_N^e\}$. Each model $f_i^e \in \mathcal{F}^e$ is initialized independently using the same distribution[2], then trained with SGD on randomly sampled mini-batches for $e \in \mathbb{N}$ epochs. Let $\mathcal{S}_E$ denote the set of different extents (total epochs of $\mathbb{X}$) used to train $f$.

We represent each model $f_i^e \in \mathcal{F}^e$ by two binary classification vectors that capture the accuracy per sample, computed separately for the train and test sets. Each vector's

[1]Compare with Morcos et al. (2018), where it is shown that NNs that generalize are more similar than those that memorize.

[2]We observe similar results with various commonly used initialization methods, see Suppl D.

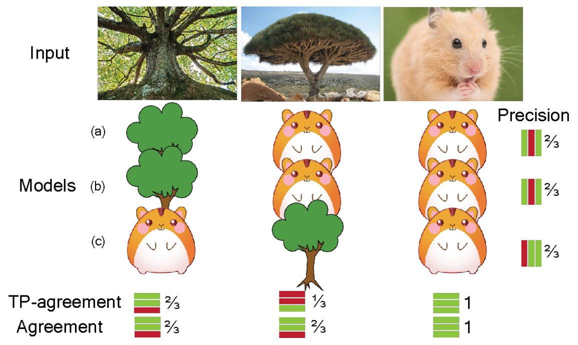


*Figure 2.* Illustration of the *Tp-agreement* and *Agreement* scores. Each row (a,b,c) corresponds to a model performing a binary classification task of classifying between images of trees and hamsters. All models have $\frac{2}{3}$ precision while performing the task, calculated as the row-average of the correct classifications. However, examples show different *Tp-agreement* scores – which can be calculated as the column-average of the correct classifications.

dimension corresponds to the size of the dataset (train or test), where each element is assigned 1 if the NN classifies the corresponding example correctly, and 0 otherwise. We use these vector representations to define and analyze the True-Positive agreement (*Tp-agreement*) of $\mathcal{F}^e$, when each model $f_i^e \in \mathcal{F}^e$ is trained on $\mathbb{X}$ from scratch for $e$ epochs. We analyze the *Tp-agreement* throughout the entire learning process, for different values of $e \in \mathcal{S}_E$.

More specifically, for each epoch $e \in \mathcal{S}_E$ we define the ***Tp-agreement*** of example $(\boldsymbol{x}, \boldsymbol{y})$ as follows:

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

The *Tp-agreement* $TPa^e(\boldsymbol{x}, y)$ of example $(\boldsymbol{x}, y)$ measures the average accuracy on $(\boldsymbol{x}, y)$ of $N$ networks which were trained for exactly $e$ epochs each. Unlike precision, *Tp-agreement* measures the average accuracy of a single example over multiple models, as opposed to precision which measures the average accuracy of a single model over multiple examples (see illustration in Fig. 2).

Note that the *Tp-agreement* score does not take into account the classifiers' consistency when they misclassify. We, therefore, define a complementary ***Agreement*** score that measures the agreement among the classifiers – the largest fraction of classifiers that predict the same label for $\boldsymbol{x}$:

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

Fig. 2 illustrates these definitions. When all the classifiers in $\mathcal{F}^e$ are identical, the *Tp-agreement* of each example is either 0 or 1, and its *Agreement* is 1. This results in a perfect bi-modal distribution of the *Tp-agreement* scores over the examples. On the other hand, if the identity of the correctly classified examples per classifier is independent, the distribution of both scores is expected to resemble a Gaussian[3]. Its center is the average accuracy of the classifiers in $\mathcal{F}^e$, in the case of the *Tp-agreement*, and a slightly higher value for the *Agreement* score. It follows that the higher the mean *Agreement* is, and the more bi-modal-like the distribution of *Tp-agreement* is around 0 and 1, the more similar the set of classifiers is. We measure the bimodality of random variable $Z$ using the following score: $kurtosis(Z) - skewness^2(Z) - 1$ (Pearson, 1894); the lower the score is, the more bi-modal the distribution of $Z$ is.

As shown in the next section, the distribution of *Tp-agreement* over examples is mostly bi-modal during the entire learning process. Specifically, for most examples the *Tp-agreement* is 0 at the beginning of learning, and then rapidly changes to 1 at some point during learning (see Suppl E). This property suggests that data is learned in a specific order. To measure how fast an example is learned by some architecture $f$, we note that the faster the example is learned, the higher its *Tp-agreement* for all epochs $e \in \mathcal{S}_E$ must be. Therefore, we define the ***Accessibility score*** of an example to be its averaged *Tp-agreement* over all epochs, formally given by $\mathbb{E}_{e\in\mathcal{S}_E}[TPa^e(\boldsymbol{x}, y)]$.

## 3. Diversity in a Single Architecture

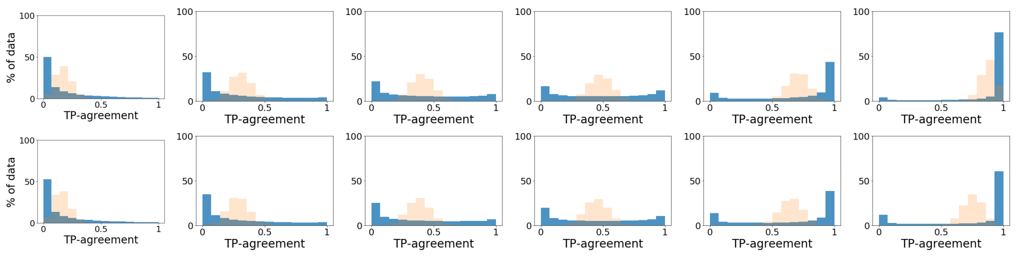
In this section, we investigate collections of classifiers obtained from a single architecture $f$.

### 3.1. Same Training Set

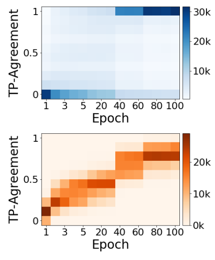
We start with the simplest condition, where all models in collection $\mathcal{F}$ are obtained by training with the same training set $\mathbb{X}$, with different initial conditions and with independently sampled mini-batches. When using common benchmark datasets, the *Tp-agreement* distribution over both train and test sets is bi-modal, see Figs. 3, 4 and Suppl D.

Upon initialization, all models are effectively i.i.d random variables, and the distribution of *Tp-agreement* scores is approximately Gaussian around random chance. After a few epochs (in many cases a single epoch, see Suppl E), the *Tp-agreement* distribution changes dramatically, becoming bi-modal with peaks 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 models, now most of the examples are being misclassified by all the models, while a small fraction is being correctly classified by all the models. When learning proceeds, the improvement in accuracy affects a shift of points from the peak at 0 to the peak at 1 while the distribution remains bi-modal, see Figs. 3, 4 and Suppl D.

[3]For large $N$ this follows from the central limit theorem.

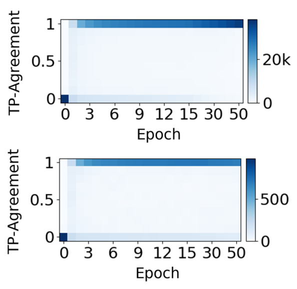


(a) *Tp-agreement* distribution per epoch.

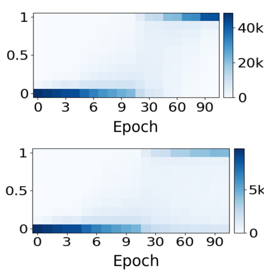


(b) All epochs

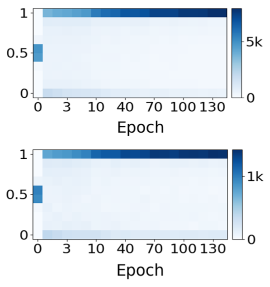
*Figure 3.* The distribution of *Tp-agreement* scores throughout the entire learning process, for 27 models of ResNet-50 trained over ImageNet. a) The distribution at specific epochs, where epochs $1, 2, 5, 30, 40, 100$ are shown respectively from left to right. In each plot, the null hypothesis is shown in fading orange, describing the distribution of *Tp-agreement* scores using random classification vectors with matching accuracy. Top: train data, bottom: validation data. b) Combined *Tp-agreement* distribution during the entire learning process. The $X$-axis corresponds to epochs, while the $Y$-axis corresponds to the *Tp-agreement* score. Intensity depicts the corresponding number of images achieving each score in each epoch. Top: combined distribution using the validation set of ImageNet, which illustrates that the models learn in a similar order (see text). Bottom: combined distribution of the null hypothesis of independent models with similar accuracy. Note the clear qualitative difference between the two cases: real classifiers (top) show a clear bi-modal behavior, while independent classifiers (bottom) show a uni-modal behavior.



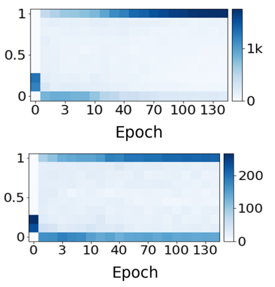
(a) NLP task



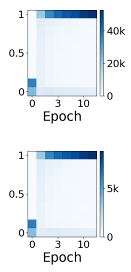
(b) CIFAR-100



(c) Cats and Dogs



(d) Small mammals



(e) MNIST

*Figure 4.* The combined distribution of *Tp-agreement* scores during the entire learning process of both train (top) and test (bottom) datasets with several architectures and datasets. As in Fig. 3, the $X$-axis corresponds to epochs and the $Y$-axis corresponds to the *Tp-agreement* score, while intensity depicts the corresponding number of images achieving each score in each epoch. Cases shown: a) 100 attention based BiLSTM models trained on text classification; b) 20 VGG-19 models trained on CIFAR-100; c) 100 models of st-VGG trained on the Cats and Dogs dataset; d) 100 models of st-VGG trained on the small-mammals dataset; e) 100 models of a small architecture trained on MNIST (see Suppl B,C for details of architectures and datasets). In all cases, clearly both the train and test datasets are learned in the same order.

The data is learned in a specific order which is insensitive to the initialization and the sampling of the mini-batches. This is true for both the train and test sets. It indicates that the models 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 models progressed independently, we would have seen a different dynamic. To rule the independence assumption out, we evaluate the null hypothesis - the independent progress of learning - by calculating the *Tp-agreement* over a set of $N$ independent random classification vectors, with specific accuracy as a baseline. As seen in Figs. 3a,3b (in fading orange), the distribution of *Tp-agreement* in this case remains Gaussian in all epochs, where the Gaussian's mean slowly shifts while tracking the improved accuracy.

*Tp-agreement* is not affected by the consensus of the networks over the misclassified points, unlike the *Agreement* score which measures agreement regardless of whether the label is true or false. Thus an *Agreement* score of 1 indicates that all models have classified the datapoint in the same way, regardless of whether it is correct or incorrect. Fig. 5 (blue) shows the distribution of *Agreement* scores for the cases depicted in Fig. 3, showing that indeed all the models classify examples in almost the same way, even when they misclassify. Had the learning of the different models progressed independently, the dynamic would have resembled a moving Gaussian, as can be seen in Fig. 5 (faded orange).

Similar results are seen when analyzing a classification problem that involves text, see Fig. 4a. We applied a BiLSTM with attention using Glove (Pennington et al., 2014) over 39K training and 1K test questions from stack overflow

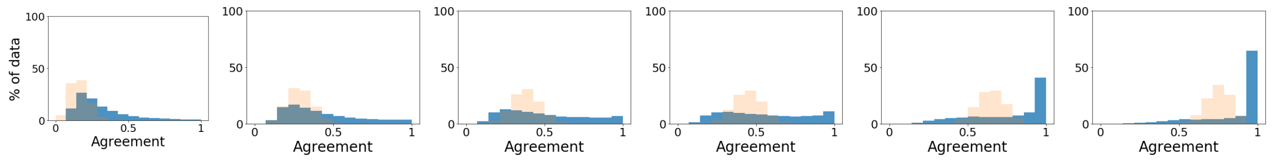


*Figure 5.* The distribution of *Agreement* scores of 27 ResNet-50 models trained on ImageNet, in corresponding epochs as in Fig. 3a. In each plot, the distribution of the null hypothesis is shown in fading orange, modeling the expected results for models that do not learn in a similar order (see text for details).

(Public domain dataset a). Labels consist of 20 mutually exclusive programming language tags assigned by users.

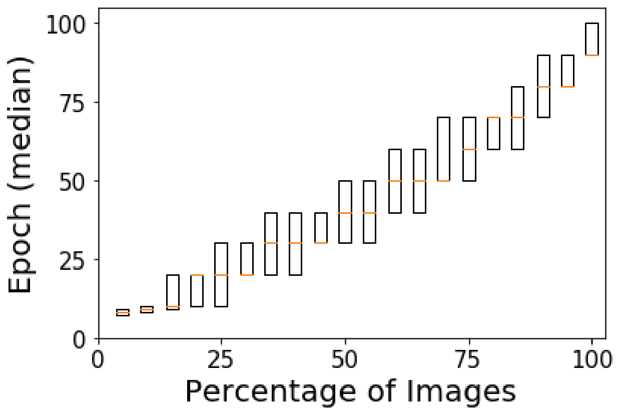


*Figure 6.* Analysis of 20 models of VGG-19 trained on CIFAR-100. Examples are sorted by the *Accessibility* score (see §2), and each box represents 5% of the examples with similar score. The orange bar represents the median epoch in which each set of images is learned, and each box represents the median's confidence intervals.

The robust order of learning when training models from a single architecture allows us to measure for each example the epoch in which it is effectively learned. Specifically, for each example and each model we define the epoch in which an example is learned as the last epoch after which it is being correctly classified. In Fig. 6 we plot the median value (over all models) of this measure for small sets of examples, where the examples are sorted based on their *Accessibility* score. We note the relatively low variance in learning epochs between different models in the collection. This result illustrates the robustness of the order in which examples are learned and the similarity between models from the same architecture. Moreover, we were able to reproduce these results for many datasets and architectures, see the following section and Fig. 4.

## 3.2. Robustness

The results reported above are extremely robust, seen in all the datasets and architectures that we have investigated, except for the synthetic datasets we artificially created specifically for this purpose (see §5). In addition to the results shown above for ImageNet (Fig. 3), CIFAR-100 (Fig. 4) and a text classification task (Fig. 4), similar results were obtained for a wide range of additional image datasets as shown in Suppl D, including: MNIST (LeCun et al., 1998) – Figs. 4e,13, Fashion-MNIST (Xiao et al., 2017) – Fig. 14, CIFAR-10 (Krizhevsky & Hinton, 2009) and CIFAR-100 – Figs. 4b,6,17,19, tiny ImageNet (Public domain dataset b) – Fig. 20, ImageNet – Figs. 3,28, VGGfaces2 (Cao et al., 2018) – Fig. 18, and subsets of these datasets – Figs. 4c,4d,15,16,24.

We investigated a variety of architectures, including AlexNet (Krizhevsky et al., 2012), DenseNet (Huang et al., 2017) and ResNet-50 (He et al., 2016) for ImageNet, VGG19 (Simonyan & Zisserman, 2014) and a stripped version of VGG (denoted st-VGG) for CIFAR-10 and CIFAR-100, and several different handcrafted architectures for other data sets (see details in Suppl B). The results can be replicated when changing various hyper-parameters, including the learning rate, optimizer, batch size, dropout rate, weight decay, width, length and depth of layers, number of layers, kernel size, initialization, and activation functions. These *Accessibility*-parameters differ across the experiments detailed both in the main paper and in Suppl D.

## 3.3. Different Training Sets

The observed pattern of similarity, and the order in which data is learned do not depend on the specific train set, but rather on the distribution the train set is sampled from. To see this, we randomly split the train set into several partitions. We trained a different collection of $N$ NNs on each of the random partitions, and computed the distribution of *Tp-agreement* scores according to each partition for both the train set and the unmodified test set. Once again, all the NNs trained with the same partition showed a bi-modal distribution of *Tp-agreement* scores during the entire learning process, over both their training partition and the common test set. Interestingly, the order in which the common test set was learned in each case was similar, depicting an almost perfect correlation between *Accessibility* scores calculated based on different partitions. Additional details are provided in Suppl E and Fig. 24.

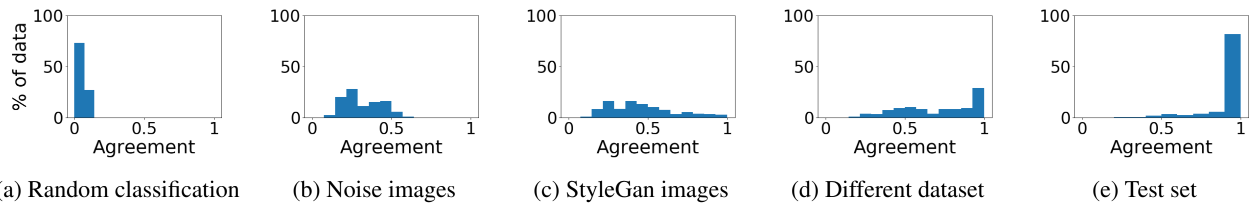


*Figure 7.* The distribution of the *Agreement* score for 27 models of ResNet-50 trained on ImageNet. a) The *Agreement* distribution of independent classifiers. (b-e) The *Agreement* distribution when the 27 models are used to classify the following test datasets: (b) images generated by the random sampling of pixels from a normal distribution; (c) images generated by StyleGAN (Karras et al., 2019) trained on ImageNet; (d) natural images from a different dataset – Indoor Scene Recognition (Quattoni & Torralba, 2009); (e) the imageNet test set. Clearly, test datasets whose image statistics is more similar to the train data show a higher *Agreement* score.

### 3.4. Out of Sample Test Sets

Using the collection of ResNet-50 models whose analysis is shown in Figs. 3,5, we further examined the *Agreement*[4] of the collection on out of sample test sets as shown in Fig. 7. We see that the *Agreement* is always higher than the *Agreement* that would be achieved by a random assignment of labels. Interestingly, the more natural the images are and the more similar the distributions of the train and test images are, the higher the *Agreement* is and the further away it gets from the *Agreement* of random assignment of labels. This property may be used as a tool for novelty detection, and was left for future work.

## 4. Cross Architectures Diversity

We now extend the analysis of the previous section to include NN instances of different architectures. In §6 we discuss comparisons with other learning paradigms.

### 4.1. Comparing Different Public Domain CNNs

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 the *Tp-agreement* scores across identical epochs. Instead, we compare collections in matching epochs with equivalent accuracy[5]. Given specific accuracy, we compare the number of examples that are classified correctly by both architectures, to the number of examples which are classified correctly only by a single architecture.

In Fig. 8a we show such comparative results using a collection of 27 models of ResNet-50 and 22 models of AlexNet trained on ImageNet. We see that as the accuracy improves, the number of examples classified correctly by both models increases, while the number of examples classified correctly by only a single model remains constant and low. Recall that ResNet-50 reaches a much higher final accuracy on ImageNet, as compared to AlexNet. In order to compare the two architectures beyond the final accuracy of AlexNet, subsequent epochs of ResNet-50 are compared to the final epoch of AlexNet after convergence. This transition is marked by a dashed vertical line in Fig. 8a. We see that ResNet-50 first learns all the examples AlexNet was able to learn, then continues to learn new examples.

The order in which different architectures learn the data is highly correlated, as can be seen when correlating the *Accessibility score* of two collections, where each collections is generated by the same architecture. Thus, when comparing two collections of ResNet-50, the correlation is almost $1$ ($r = 0.99,\ p \leq 10^{-50}$, Fig. 8b). The correlation remains high when comparing two collections of two different architectures: ResNet-50 and AlexNet ($r = 0.87,\ p \leq 10^{-50}$, Fig. 8c) or ResNet-50 and DenseNet ($r = 0.97,\ p \leq 10^{-50}$, Fig. 8d). These results are surprising given how different the error rates of the three architectures are: AlexNet with Top-1 error of $0.45$, ResNet-50 with Top-1 error of $0.24$, and DenseNet with Top-1 error of $0.27$. Additional results, comparing additional pairs of competitive ImageNet architectures, are shown in Suppl D Figs. 25,26. The results have been replicated for other settings, including VGG19 and st-VGG on CIFAR-10 and CIFAR-100 (see Fig. 27).

### 4.2. Linear Networks

Convolutional Neural Networks (CNN) 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. We observe that the bi-modal behavior observed in general CNNs also occurs in this case.

Specifically, we define a linear st-VGG by replacing all the non-linear layers of st-VGG with their linear equivalents (see Suppl B). We train $100$ linear st-VGG on the small-mammals dataset (see Suppl C). The performance

[4]Since the classes in the test sets are not present in the train, only the *Agreement* remains relevant.

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

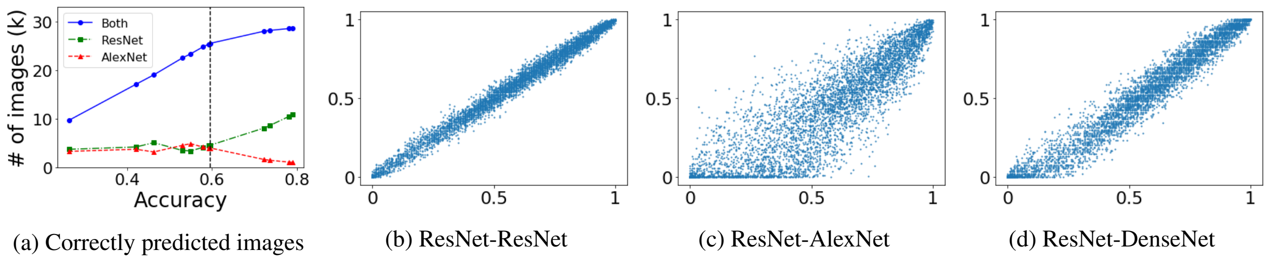


*Figure 8.* Comparison of the learning order induced by models with different architectures – 27 models of ResNet-50, 22 of AlexNet, and 6 of DenseNet, all trained on ImageNet. a) The collections of ResNet and AlexNet are used to consturct a single ensemble classifier for ResNet-50 and AlexNet respectively, using the majority vote. We calculate the number of examples from the validation set which are classified correctly by either both ensembles (blue solid line), or only by a single ensemble (dotted green for ResNet and dashed red for AlexNet). Comparison is done during learning in corresponding epochs, where both ensembles reach the same accuracy. Since ResNet-50 reaches 80% accuracy while AlexNet only reaches 60% accuracy, there are no corresponding epochs beyond 60%, an event marked by a vertical dashed black line. Beyond this line, we compare ResNet-50 to the converged AlexNet. Before the convergence of AlexNet, both architectures learn the same examples during training. After convergence, ResNet first learns those examples classified correctly only by AlexNet, and then carries on to learn additional examples. (b-d) Correlation between the *Accessibility score* when using ImageNet train set, and considering 3 pairs of architectures. The $X$-axis corresonds to the *Accessibility score* of the first architecture in the pair, while the $Y$-axis corresponds to the second architecture in the pair. We considered the following pairs: (b) two disjoint collections of 13 ResNet-50 models; (c) 27 ResNet-50 models and 22 AlexNet models; (d) 27 ResNet-50 models and 6 DenseNet models. Overall, all the pairs show high correlation between the *Accessibility scores* of the corresponding collections in the pair.

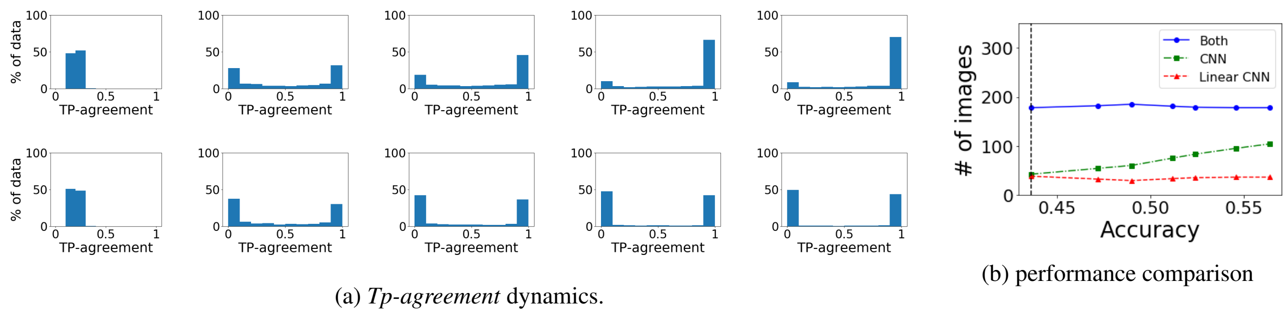
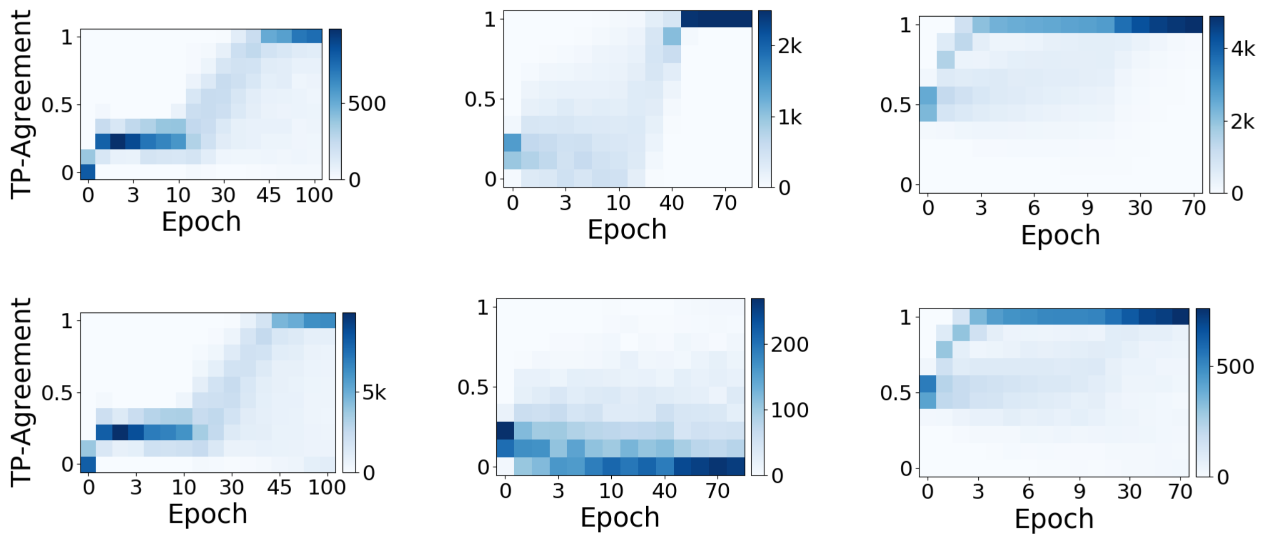


*Figure 9.* Comparison of linear and non-linear networks trained on the small-mammals dataset. (a) The distribution of *Tp-agreement* scores at specific epochs, where epochs $0, 10, 30, 90, 140$ are shown respectively from left to right. Top – st-VGG, bottom – linear st-VGG. (b) Similar analysis as in Fig. 8a, where solid blue denotes the number of examples classified correctly by both architectures, dotted green denotes examples classified correctly only by st-VGG, and dashed red denotes examples classified correctly only by linear st-VGG. We can see that linear networks converge very fast, while non-linear networks learn the same points that are learned by linear networks, then carry on to learn more examples.

of these linear networks is weaker (0.43 average accuracy) than the original non-linear networks (0.56 average accuracy), and they converge faster. Still, the distribution of the *Tp-agreement* throughout the entire learning process is bi-modal (maximum Pearson bi-modality: 0.06), and this bi-modality is even more pronounced than the bi-modality in the non-linear case (maximum Pearson bi-modality: 0.22). The bi-modal dynamics of st-VGG can be seen in the top row of Fig. 9a, compared to the dynamics of linear st-VGG in similar epochs at the bottom row of Fig. 9a.

Linear networks converge in just a few epochs, which is too fast for the meaningful evaluation of *Accessibility* scores. Nevertheless, we still observe that non-linear networks learn first the examples linear networks do, then continue to learn new examples. In Fig. 9b, we plot for each accuracy the number of images that are classified correctly by both linear and non-linear models, and the number of images that have been learned by a single model only. 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.

These results show that the order of learning is not a direct result of using non-linear operations in neural architectures.



(a) Gabor patches (b) Small mammals – Random labels (c) Gaussian classification

*Figure 10.* Similarly to Fig. 4, we plot the distribution of *Tp-agreement* scores during the entire learning process of both train (top) and test (bottom) datasets for several architectures and synthetic datasets. All the cases analyzed here show no common learning order, and models seem to learn the data independently of each other. The cases include: a) 100 st-VGG models trained on the artificial Gabor patches dataset (see §5.1); b) 100 st-VGG models trained on the small mammals dataset with shuffled labels (Zhang et al., 2016) (see §5.2); c) 100 fully connected networks trained on a Gaussian classification task (see §5.1).

## 5. When Tp-agreement is Not Bi-Modal

In §3 we discussed the characteristic bi-modal distribution of *Tp-agreement* scores, illustrated in Figs. 3,4, which has appeared in all the experiments presented until now, in both the train and test sets. In this section, we investigate the circumstances under which the bi-modal distribution of *Tp-agreement* is no longer seen.

### 5.1. Synthetic Datasets

The bi-modal distribution of *Tp-agreement* scores through all stages of learning is not an inherent property of NNs. We demonstrate this point using a dataset of artificial images, consisting of Gabor patches: the dataset contains 12 overlapping classes that differ from each other in orientation and color (see Suppl C). We trained a collection of 100 st-VGG models on this data. The distribution of *Tp-agreement* scores, shown in Fig. 10a, is no longer bi-modal. Rather, the distribution is approximately normal. 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. These results suggest that networks in the collection have learned the data in different orders.

For some datasets, the order in which NNs learn may be completely independent. We train 100 models of a fully connected architecture (see Suppl B). The NNs are trained to discriminate points sampled from two largely overlapping Gaussian distributions in high dimension. The dynamic distribution of *Tp-agreement* scores is shown in Fig. 10c, and resembles the distribution of *Tp-agreement* of the null hypothesis of random classification vectors. These results suggest that the order of learning in this case is independent across different models.

### 5.2. Random Labels

The bi-modality of *Tp-agreement* 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 (Zhang et al., 2016). In this case, training accuracy reaches 100% while test accuracy remains at chance level, which indicates that the NNs can memorize the data. Interestingly, the distribution of *Tp-agreement* scores is no longer bi-modal, with a minimum Pearson bi-modality 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 NNs, see Fig. 10b. These results are in agreement with the results of Arpit et al. (2017); Morcos et al. (2018), which show different NN dynamics when NNs perform memorization or generalization.

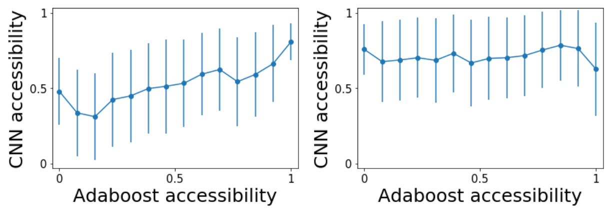


*Figure 11.* Correlations between the learning order of AdaBoost with 100 linear classifiers and 100 models of st-VGG both trained on CIFAR-10. Each value of AdaBoost *Accessibility* (see §6) is matched with the average over the corresponding st-VGG *Accessibility* scores (see §2). Error bars show standard error. Left: AdaBoost trained on pixel values. Right: AdaBoost trained on the embedding obtained from the penultimate layer of Inception-V3 trained on ImageNet.

## 6. Learning Order With AdaBoost Classifier

Up to now, we investigated a variety of architectures, revealing a common learning order on benchmark datasets. This order may be fully determined by the benchmark dataset, in which case it should be replicated when inspecting other learning paradigms. In this section, we show that this is not the case. To this end, we consider boosting based on linear classifiers as weak learners, since the training of both neural models and AdaBoost share a dynamic aspect: in NN training accuracy increases with time due to the use of SGD, while in AdaBoost accuracy increases over time due to the accumulation of weak learners.

Specifically, we trained AdaBoost with up to 100 linear classifiers on the CIFAR-10 dataset. Each channel in each image is normalized to 0 mean and 1 standard deviation (similarly to the normalization for NNs). The image tensor is then flattened to a vector. As can be seen in Suppl F, the AdaBoost accuracy is increasing as a function of the number of linear classifiers, and as expected, its final accuracy is significantly lower than neural models.

While most of the examples successfully learned by AdaBoost were also learned by the NN, the order in which the examples were learned was different. Similarly to Fig. 8, we compare the correlation between both learning orders (see Fig. 11), showing a low correlation ($r = 0.35, p \leq 10^{-20}$) between the orders. This result demonstrates that non-neural paradigms may learn data in a different order.

Additionally, we trained AdaBoost using as features the penultimate layer of Inception-V3 (Szegedy et al., 2015) trained on ImageNet. In this case, the AdaBoost accuracy increases dramatically (see Suppl F), while the correlation between the learning order gets even lower ($r = 0.05, p \leq 10^{-20}$), see Fig. 11. This result shows that the learning order of NNs is not correlated with the learning order induced by AdaBoost, even when Adaboost uses a representation based on transfer learning, which shows that benchmark datasets can be learned in different ways. These results were replicated for other datasets, including subsets of CIFAR-100 and ImageNet, see Suppl F.

## 7. Summary and Discussion

We empirically show that different neural models learn similar classification functions. We also show that the learning dynamics are similar, as they learn similar functions in all intermediate stages of learning. This is true for a variety of architectures, including different commonly used CNN architectures and LSTMs trained on public domain datasets, and irrespective of size and other hyper-parameters. This pattern of similarity crosses architectural boundaries: while different architectures may learn at a different speed, the data is learned in the same order. Finally, we discuss cases where this similarity breaks down, indicating that the observed similarity is not an artifact of learning using SGD-based algorithms.

While observing that some examples are learned faster than others by almost all the networks from different architectures, this observation does not explain why this happens, or what makes a specific example "easier" to learn than others. We have established that whatever property makes an example "easy" depends on the task, not the image alone, since a certain example in a certain dataset can be both easy or hard, depending on the classification task that is being learned. Further investigation of this fascinating question is left for future work.

A somewhat related issue, the re-sampling of train points during training based on some notion of point difficulty, has been investigated in the context of curriculum learning (Bengio et al., 2009), self-paced learning (Kumar et al., 2010), hard example mining (Shrivastava et al., 2016), and boosting (Hastie et al., 2009). Unlike this work, our measure of difficulty depends on what a network learns most robustly, as opposed to a supervised difficulty score. We note that when using our scores to impose a learning order for the purpose of curriculum learning as suggested in Hacohen & Weinshall (2019), the resulting curriculum did not seem to benefit the learning.

## Acknowledgements

This work was supported in part by a grant from the Israel Science Foundation (ISF) and by the Gatsby Charitable Foundations. We thank Reviewer #2 for proposing a more intuitive graphical representation of the results. This representation features prominently throughout the paper.

## References

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

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., 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.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

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

Cohen, U., Chung, S., Lee, D. D., and Sompolinsky, H. Separability and geometry of object manifolds in deep neural networks. *bioRxiv*, pp. 644658, 2019.

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

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

Glorot, X. and Bengio, Y. 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.

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

Hastie, T., Rosset, S., Zhu, J., and Zou, H. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A. F., Bogoychev, N., Martins, A. F. T., and Birch, A. Marian: Fast neural machine translation in c++. In *ACL*, 2018.

Kantor, Y., Katz, Y., Choshen, L., Cohen-Karlik, E., Liberman, N., Toledo, A., Menczel, A., and Slonim, N. Learning to combine grammatical error corrections. In *BEA@ACL*, 2019.

Karras, T., Laine, S., and Aila, T. 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.

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

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

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

Krueger, D., Ballas, N., Jastrzebski, S., Arpit, D., Kanwal, M. S., Maharaj, T., Bengio, E., Fischer, A., and Courville, A. Deep nets don't learn via memorization. 2017.

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

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

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

Lenc, K. and Vedaldi, A. 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.

Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. E. Convergent learning: Do different neural networks learn the same representations? In *FE@ NIPS*, pp. 196–212, 2015.

Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. E. Convergent learning: Do different neural networks learn the same representations? In *Iclr*, 2016.

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

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

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

Pennington, J., Socher, R., and Manning, C. D. 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.

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

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

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

Saxe, A. M., McClelland, J. L., and Ganguli, S. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 116 23:11537–11546, 2018.

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

Shrivastava, A., Gupta, A., and Girshick, R. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 761–769, 2016.

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

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS*, 2017.

Wang, L., Hu, L., Gu, J., Hu, Z., Wu, Y., He, K., and Hopcroft, J. 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.

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

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.