Exploring the Memorization-Generalization Continuum in Deep Learning

Ziheng Jiang [*1 2] Chiyuan Zhang [*1] Kunal Talwar [1] Michael C. Mozer [1 3]

Abstract

Human learners appreciate that some facts demand memorization whereas other facts support generalization. For example, English verbs have irregular cases that must be memorized (e.g., GO→WENT) and regular cases that generalize well (e.g., KISS→KISSED, MISS→MISSED). Likewise, deep neural networks have the capacity to memorize rare or irregular forms but nonetheless generalize across instances that share common patterns or structures. We analyze how individual instances are treated by a model on the memorization-generalization continuum via a *consistency score.* The score is the expected accuracy of a particular architecture for a held-out instance on a training set of a fixed size sampled from the data distribution. We obtain empirical estimates of this score for individual instances in multiple data sets, and we show that the score identifies out-of-distribution and mislabeled examples at one end of the continuum and regular examples at the other end. We explore three proxies to the consistency score: kernel density estimation on input and hidden representations; and the time course of training, i.e., learning speed. In addition to helping to understand the memorization versus generalization dynamics during training, the C-score proxies have potential application for out-of-distribution detection, curriculum learning, and active data collection.

1. Introduction

Human learning requires both *memorization* of specific facts and *generalization* to novel situations and contexts. Both memorization and generalization can be needed in a domain, and the boundary between when to memorize and when to generalize can be fuzzy. For example, in learning the past tense form of English verbs, there are some verbs whose past tense must simply be memorized (GO→WENT, EAT→ATE, HIT→HIT) and there are many *regular* verbs that obey the rule of appending "ed" (KISS→KISSED, KICK→KICKED, BREW→BREWED, etc.). Generalization to a novel word typically follows the "ed" rule, for example, BINK→BINKED. Intermediate between the exception verbs and regular verbs are subregularities—a set of exception verbs that have consistent structure (e.g., the mapping of SING→SANG, RING→RANG, SPRING→SPRANG); these subregularities might suggest a generalization rule to novel verbs that would obtain, for example, DRING→DRANG. Note that rule-governed and exception cases can have very similar forms, which increases the difficulty of learning each. Consider one-syllable verbs containing 'ee', which include the regular cases NEED→NEEDED and BEEP→BEEPED as well as the exception cases MEET→MET, FEEL→FELT, and SEEK→SOUGHT. Generalization from the rule-governed cases can hamper the learning of the exception cases and vice-versa. Indeed, children learning English initially master high frequency exception verbs such as GO→WENT, but after accumulating experience with regular verbs, they then begin to over-regularize by mapping GO→GOED, eventually learning the distinction between the regular and exception verbs; neural nets show the same interesting pattern over the course of training (Rumelhart & McClelland, 1986).

Memorization is tantamount to a look-up table with the individual facts accessible for retrieval. Generalization requires the inference of statistical regularities in the training environment, and the application of procedures or rules for exploiting the regularities. In deep learning, memorization is often considered a failure of a network because memorization implies no generalization. However, mastering a domain involves knowing when to generalize and when not to generalize. Consider the two-class problem with training examples positioned in an input space as in Figure 1a, or positioned in a latent space as in Figure 1b. Instance 3 (the iron throne) is an exception case and there may not exist similar cases in the data enironment. Instance 1 (a generic chair) lies in a region with a consistent labeling and thus seems to follow a strong regularity. Instance 2 (a rocking chair) has a few supporting neighbors, but it lies in a distinct neighborhood from the majority of same-label instances; its neighborhood might be considered a weak regularity.

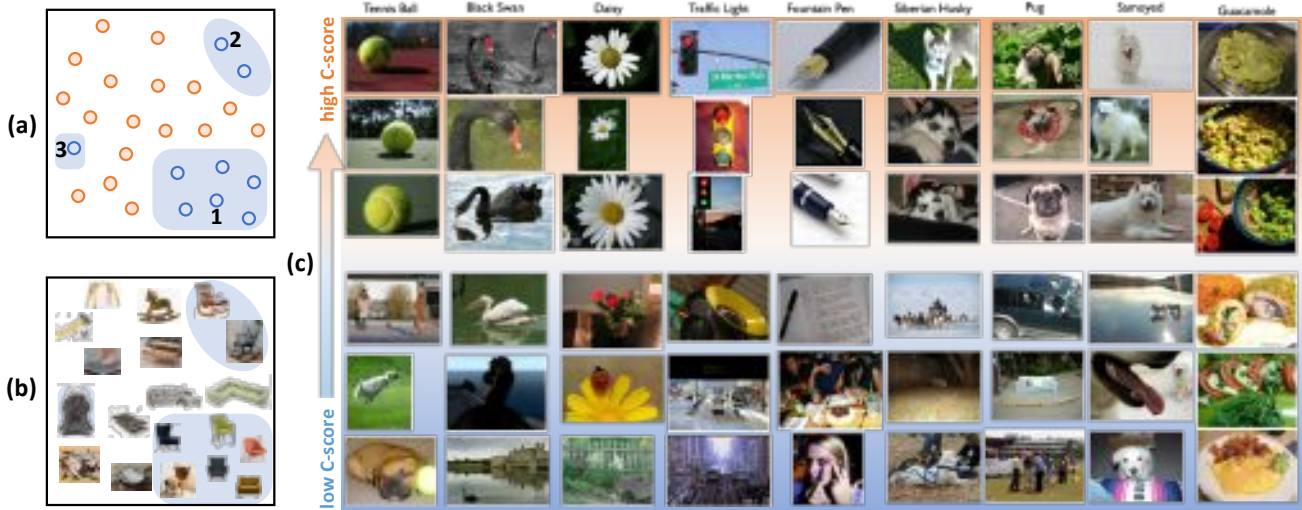In this article, we formalize the notion of strong regularities,

---

Figure 1. Regularities and exceptions in (a) a two-class input space and (b) in a latent space with chairs and non-chairs. (c) Regularities (high C-scores) and exceptions (low C-scores) in ImageNet.

$$\hat{C}_{A,J,n}(x,y) = E_{D \sim \Gamma}\left[\mathbb{P}(\hat{y}_{best} = y | D, x)\right]$$

weak regularities, and exceptions in the context of a deep net. We propose a *consistency score* or *C-score* for an instance x with label y, defined as the expected accuracy of predicted label $\hat{y}$ from a classifier of architecture $\mathcal{A}$ trained on n i.i.d. examples drawn from a data distribution $\mathcal{P}$:

independent identically

One sample.

$$C_{\mathcal{A},\mathcal{P},n}(x,y) = \mathbb{E}_{D^n \sim \mathcal{P}}\left[\mathbb{P}(\hat{y}_{\mathcal{A}} = y | D, x)\right]. \tag{1}$$

Practically, we require that the instance x is excluded from the training set, but under a continuous data distribution, the probability of selecting the same instance for both training and testing is zero. The C-score reflects the consistency the instance is with respect to the training set: in Figure 1a, instance 1 should have a higher C-score than instance 2 which in turn should have a higher C-score than instance 3. The C-score reflects the relationship of each instance to the training population. A low C-score indicates that the instance is not aligned with the training population and therefore learning requires memorization. A high C-score indicates that the instance is supported by the training population and generalization thus follows naturally. The formulation of C-score is closely related to the memorization score from Feldman (2019), which is defined relative to a dataset that includes (x, y) and measures the change in the prediction accuracy on x when (x, y) is removed from the dataset. They use the score to quantify the importance of memorization to achieve optimal generalization on a data distribution containing long tail of rare examples.

For a nearest-neighbor classifier that operates on the input space (Figure 1a), the C-score is related to the literature on outlier detection (Breunig et al., 2000; Ramaswamy et al., 2000; Campos et al., 2016). However, for a deep network, which operates over a latent space (Figure 1b), the C-score depends not just on the training data distribution but on the model architecture, loss function, optimizer, and hyperpa-

rameters. Our work is thus related to adversarial methods to identify outliers in latent space (Lee et al., 2018; Pidhorskyi et al., 2018; Beggel et al., 2019).

The C-score has many potential uses. First, it can assist in understanding a dataset's structure by teasing apart distinct regularities and subregularities. Second, it can be used for detecting out-of-distribution inputs as well as mislabeled instances: these instances will have low C-scores because they have little support from the training distribution, like instance 3 in Figure 1a. Third, it can be used to guide active data collection to improve performance of rare cases that the model treats as exceptions. Fourth, it can be used to prioritize training instances, along the lines of curriculum learning (Bengio et al., 2009; Saxena et al., 2019).

There are many reasons why the C-score as defined in Equation 1 cannot be computed. The underlying data distribution is not known. The expectation must be approximated by sampling. Each sample requires model training. Thus, we seek computationally efficient proxies for the C-score. Ideally the score could be obtained from an untrained network or a single network early in the time course of training.

In our work, we estimate a ground-truth C-score for a dataset via holdout performance on trained networks. Figure 1c shows examples of various ImageNet classes with low and high estimated C-scores. Given these estimates, we investigate various proxies to the C-score which include measures based on: density estimation (in input, latent, and gradient spaces), and the time course of learning within a single training run. Our key contributions are as follows.

- We obtain empirical estimates of the C-score for individual instances in MNIST, CIFAR-10, CIFAR-100, and ImageNet. Estimation requires training up to 20,000 net-

work replications per data set, permitting us to sort instances into those satisfying strong regularities, those satisfying weaker regularities, and exception (outlier) cases.

- Because empirical estimation of the C-score is computationally costly, we define and evaluate a set of candidate C-score proxies. We identified a lightweight proxy score, the *cumulative binary training loss*, that correlates strongly ($\rho = .864$) with the C-score and can be computed for free for all instances in the training set. We note that this result is nontrivial because the C-score is defined for held-out instances, whereas the cumulative binary training loss is defined over a training set.
- We explored the relationship between the C-score and learning dynamics, finding that the lower the C-score, the more slowly an instance is learned and the lower the learning rate required for the instance to be learned.

2. Related Work

Mangalam & Prabhu (2019) compared the training of Random Forests and SVMs with deep networks and found that deep learning prioritize examples that are learnable by shallow models. Arpit et al. (2017) looked at memorization in deep learning by studying gradient based learning algorithms on noise vs. real data. They found that with *carefully tuned* explicit regularization, a network's capability of memorizing the noisy data can be effectively controlled without compromising the generalization performance on real data.

Carlini et al. (2018) proposed multiple measures for finding prototypical examples that are intrinsic to the dataset, and could lead to good performance when training only on those examples. In contrast, our C-score captures the statistical regularity combining biases in both the data and the learning algorithm. Moreover, training only on a small subset of examples with high C-scores do not necessarily lead to good performance as statistical regularity realizes only when enough supporting examples are present. Examples with low C-scores are also not necessarily unimportant for learning. Some metrics used in their studies are similar to ours. The closest pair is their *model confidence* and the *learning speed* studied by us. Note the former ignores the labels, which we use to quantify the learning speed. The *holdout retraining* and the *ensemble agreement* metrics used in Carlini et al. (2018) is conceptually similar to the our holdout procedures. But their retraining is a two-stage training procedure which involves pre-training and fine-tuning; their ensemble agreement mixes architectures with heterogeneous capacities and ignores the label information.

Feldman (2019) constructed a theoretical model to show that when the data distribution has a long tail of rare examples, memorization is necessary for optimal learning. In their proof, a score was proposed to quantify the memorization of an example. Our C-score closely resemble their defini-

tion. The main difference is that memorization in Feldman (2019) is defined relative to a given dataset, whereas C-score evaluates the expected accuracy when trained on i.i.d. sampled subset of varying size n. We also aim to understand how C-score depends on n whereas Feldman (2019); Feldman & Zhang (2019) focus on the effect that memorized examples have on the test set accuracy. Another line of recent theoretical work studies *interpolation* (e.g. Belkin et al., 2018a;b; Liang & Rakhlin, 2018; Belkin et al., 2019), which means the model perfectly fits the training data. It is shown that in some cases interpolation is harmless for optimal generalization. Note interpolation does not necessarily imply memorization (consider fitting a linear classifier on two classes with well separated clusters).

3. Empirical Estimation of the C-score

Computing the C-score by our definition (Equation 1) is not feasible in practice because the underlying data distribution is typically unknown, and even if it were, the expectation cannot be computed analytically. In practice, we usually have a fixed data set $\hat{\mathcal{D}}$ consisting of N i.i.d. samples from the underlying distribution; for example, with the CIFAR-10 image classification task, we have 50,000 training examples. An estimate of the C-score can be computed by replacing the expectation in (1) with empirical averaging and by sampling i.i.d. subsets of a given size n from the fixed data set. We thus define the *empirical C-score* for an instance (x, y), based on the estimator of memorization score from Feldman (2019) proposed in Feldman & Zhang (2019):

$$\hat{C}_{\mathcal{A},\hat{\mathcal{D}},n}(x, y) = \hat{\mathbb{E}}^r_{D \overset{n}{\sim} \hat{\mathcal{D}} \setminus \{(x,y)\}} \left[\mathbb{P}(\hat{y}_{\mathcal{A}} = y | D, x) \right], \qquad (2)$$

where D is a subset of size n uniformly sampled from $\hat{\mathcal{D}}$ excluding (x, y), and $\hat{\mathbb{E}}^r$ denotes empirical averaging with r i.i.d. samples of such subsets. Because of the cost of computing $\hat{C}$ for individual (x, y) is prohibitive, we instead use a k-fold validation procedure. Specifically, we evaluate each fold on the $N - n$ instances not considered for training, and determine the empirical C-score for a given instance using only the folds in which the instance is in the held-out set. We refer to this procedure as *holdout validation*, summarized in Algorithm 1.

Because each data set is a different size and we require $n < N$, we find it convenient to refer not to the absolute number of examples, n, but to the percentage $s \triangleq 100n/N$ of $\hat{\mathcal{D}}$ used for training, which we refer to as the *subset ratio*, with $s \in [0, 100]$. We use a 3-layer fully connected network for MNIST, Inception for CIFAR-10 / CIFAR-100 and ResNet-50 for ImageNet. Please refer to Appendix A for the full details on architectures and hyper-parameters.

Figure 2 shows the distribution of empirical C-scores for CIFAR-10 for $s \in \{10, 20, 30, ..., 90\}$. For each level of s,

Algorithm 1 Holdout validation

Input: Data set $\hat{\mathcal{D}} = (X, Y)$ with N examples
Input: n: number of instances used for training
Input: k: number of folds
Output: $\tilde{C} \in \mathbb{R}^N$: C-score estimate of each example
    Initialize binary mask matrix $M \leftarrow 0^{k \times N}$
    Initialize 0-1 loss matrix $L \leftarrow 0^{k \times N}$
    for $i \in (1, 2, \dots, k)$ **do**
        Sample n random indices I from $\{1, \dots, N\}$
        $M[i, I] \leftarrow 1$
        Train $\hat{f}$ from scratch with the subset $X[I], Y[I]$
        $L[i, :] \leftarrow \mathbf{1}[\hat{f}(X) \neq Y]$
    end for
    Initialize score estimation vector $\hat{C} \leftarrow 0^N$
    for $j \in (1, 2, \dots, N)$ **do**
        $Q \leftarrow \neg M[:, j]$
        $\tilde{C}[j] \leftarrow \text{sum}(\neg L[:, Q]) / \text{sum}(Q)$
    end for

Figure 3. (left) histogram of estimated integral C-scores for CIFAR-10; (right) C-scores by histogram bin as a function of subset ratio s. Monotonic ordering does not depend on s.

$k = 2000$ train/evaluation folds are run. Beyond giving a sense of what fraction of the data set must be used for training to obtain good generalization, the Figure suggests that floor and ceiling effects may concentrate instances, making it difficult to distinguish them based on their C-scores if s is too small or too large (will justify shortly). Rather than trying to determine the 'just right' value of s, we compute a C-score marginalized over s under a uniform distribution. The left panel of Figure 3 shows a histogram of these *estimated integral C-scores*. Although the bulk of the scores are on the high end, they are more widely distributed than in the histogram for any particular s (Figure 2).

We stratify the instances by their integral C-score into 30 bins, as indicated by the coloring of the bars of the histogram in Figure 3. In the right panel of the Figure, we separately plot the mean C-score for the instances in a bin as a function of the subset ratio s. Note that the monotonic ordering of C-scores does not vary with s, but instances bunch up at low C-scores for small s and at high C-scores for larger s, indicated by the opacity of the open circles in the Figure. (The semi-transparent circles become opaque when superimposed on one another.) Bunching makes the instances less discriminable. At the low end of the integrated C-scores (cyan lines), note that the curves drop *below* chance (0.1 for CIFAR-10) with increasing s. We conjecture that these instances are am-

biguous (e.g., visually similar to instances from a different class), and as the data set grows, regularities in other classes systematically pull these ambiguous instances in the wrong direction. This behavior is analogous to the phenomenon we mentioned earlier that children increase their production of verb overregularization errors (GO→GOED) as they acquire more exposure to a language.

For MNIST, CIFAR-10, and CIFAR-100, Figure 4 presents instances that have varying estimated integral C-scores. Each block of examples is one category; the left, middle, and right columns have high, intermediate, and low C-scores, respectively. The homogeneity of examples in the left column suggests a large cluster of very similar images that form a functional prototype. In contrast, many of the examples in the right column are ambiguous or even mislabeled.

3.1. Point Estimation of Integral C-score

The integral estimation computed in the previous section requires invoking the holdout validation procedure for a range of s, with each invocation involving training on the order of 2000 networks. For large-scale data sets like ImageNet, the computational cost of this approximate integration procedure is too high. Consequently, we investigate the feasibility of approximating the integral C-score with a *point estimate*, i.e., selection of the s that best represents the integral score. By 'best represents,' we mean that the ranking of instances by the integral score matches the ranking by the score for a particular s. Figure 5 shows rank correlation between integral score and score for a given s, as a function of s. The left and right graphs plot two different rank correlation measures, Spearman's ρ and Kendall's τ, respectively. Each curve in a graph corresponds to a a particular data set. Examining the green CIFAR-10 curve, there is a peak at $s = 30$ for both measures, indicating that $s = 30$ yields the best point-estimate approximation for the integral C-score. That the peak is at an intermediate s is consistent with the observation from Figure 3 that the C-score bunches together instances for low and high s.

For MNIST, a less challenging data set than CIFAR-10, the peak is lower, at $s = 10$; for CIFAR-100, a more challeng-

Figure 2. Histogram of C-scores for CIFAR-10 for subset ratios $s \in \{10, 20, 30, \dots 90\}$. The vertical axis is the percentage of the 50,000 instances that fall into a bin.
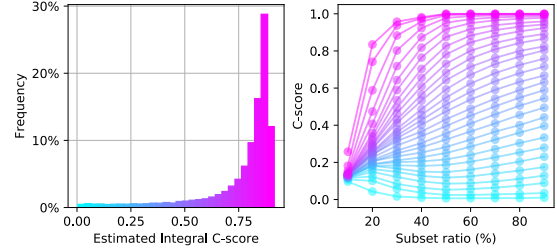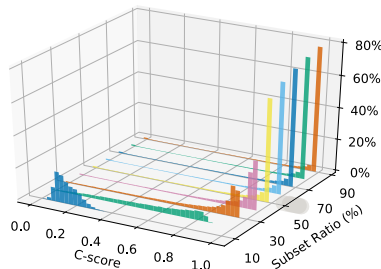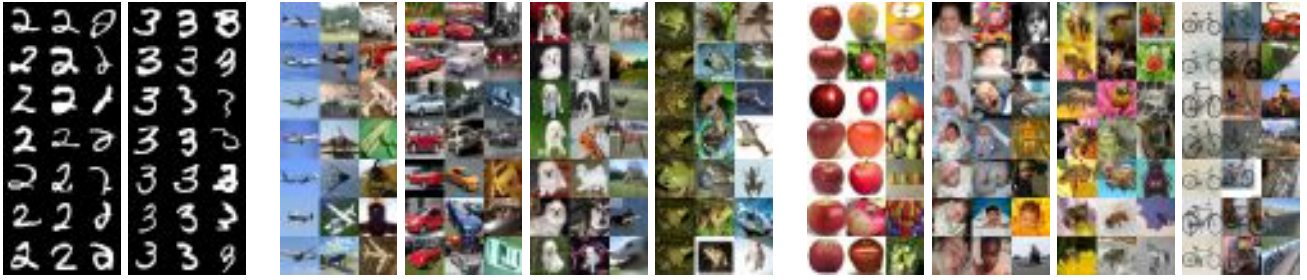
Figure 4. Examples from MNIST (blocks 1, 2), CIFAR-10 (blocks 3-6), and CIFAR-100 (blocks 7-10). Each block shows a single class; the left, middle, and right columns of a block depict instances with high, intermediate, and low C-scores, respectively.
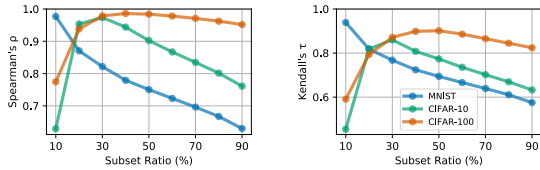

Figure 5. (left) Spearman's ρ and (right) Kendall's τ measures of rank correlation between the integral C-score and the C-score for a particular subset ratio, s. The peak of each curve indicates the training set size that best reveals generalization of the model.

ing data set than CIFAR-10, the peak is higher, at $s = 40$ or $s = 50$. Thus, the peak appears to shift to larger s for more challenging data sets. This finding is not surprising: more challenging data sets require a greater diversity of training instances in order to observe generalization.

3.2. ImageNet

In addition to MNIST, CIFAR-10, and CIFAR-100, we conducted experiments with ImageNet. Due to the large data set size (1.2M examples), we picked a single s for our C-score estimate. Based on the fact that the optimal s increases with data set complexity, we picked $s = 70$ for ImageNet. In particular, we train 2,000 ResNet-50 models each with a random 70% subset of the ImageNet training set, and compute the C-scores for all the training examples.

Figure 6. The joint distribution of C-score per-class means and standard deviations on ImageNet. Image samples from representative classes (indicated by ⋆'s) are shown in Figure 7.
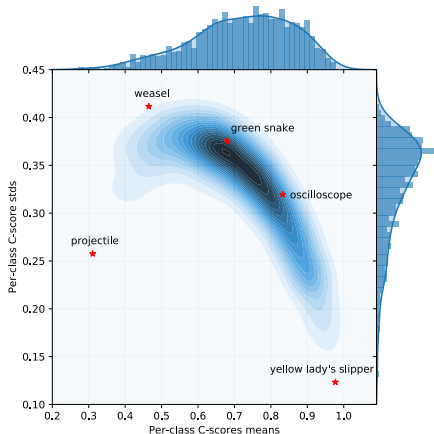
The examples shown in Figure 1c are ranked according to this C-score estimate. Because ImageNet has 1,000 classes, we cannot offer a simple overview over the entire dataset as in MNIST and CIFAR. Thus, we focus on analyzing the behaviors of individual classes. Specifically, we compute the mean and standard deviation (SD) of the C-scores of all the examples in a particular class. The mean C-scores indicates the relative difficulty of classes, and the SD indicates the diversity of examples within each class. The two-dimensional histogram in Figure 6 depicts the joint distribution of mean and SD across all classes. A strong correlation is observed: classes with high mean C-scores tend to have low variances. We selected several classes with various combinations of mean and SD, indicated by the ⋆'s in Figure 6. We then selected sample images from the top 1%, 35% and 99% percentile ranked by the C-score within each class, and show them in Figure 7.

The class *projectile* has C-scores spread out the value range. In contrast, the class *weasel* has large masses on both low and high C-scores, leading to larger variance than *projectile*. The class *green snake* from the high density region of the 2D histogram in Figure 6 represent common cases in the 1,000 ImageNet classes: while highly regular examples dominate, there are also usually a non-trivial amount of outliers or ambiguous examples that need to be memorized in training. The class *oscilloscope* is similar to *green snake* except with higher mean and lower SD. On the other extreme of the spectrum is the class *yellow lady's slipper*, which mostly contain highly regular examples. From the image samples, we can see even the 99% percentile ranked examples enjoy a consistent color scheme with the rest of the images.

4. C-Score Proxies

Given meaningful estimates of the C-score, we now investigate various proxies to the C-score. To unwind the logic of our investigation, the C-score relates to the consistency of a given (x, y) instance with the rest of the data set. We've shown that it is useful for understanding the data set structure and for identifying outliers and mislabeled instances. However, it is expensive to estimate. Our goal in this section is to identify proxy measures strongly correlated with the C-
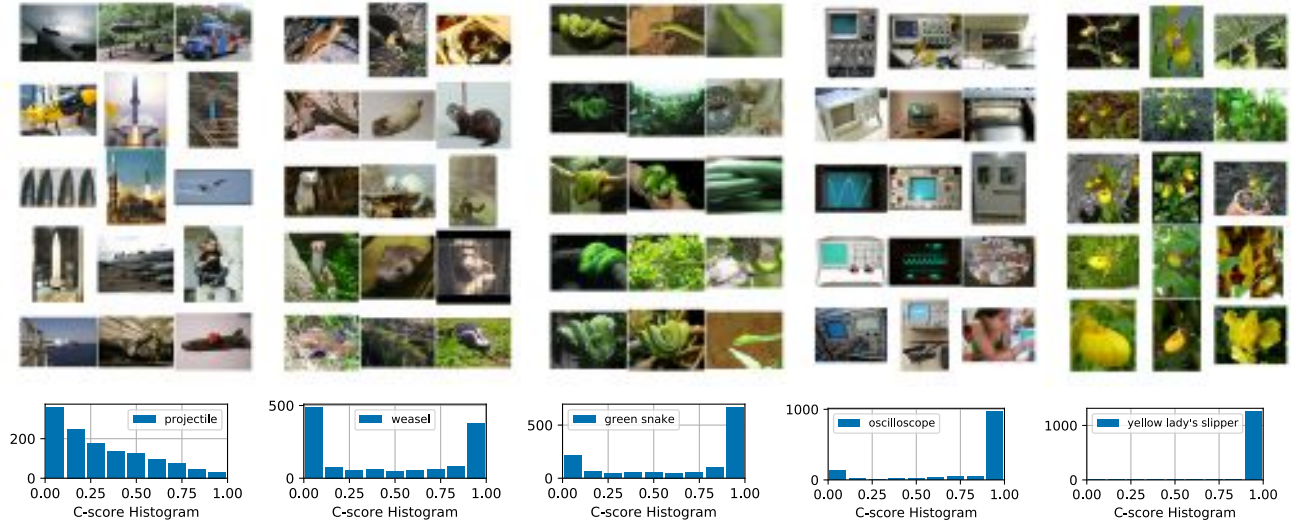
Figure 7. Example images from ImageNet. The 5 classes are chosen to have representative per-class C-score mean–standard-deviation profiles, as shown in Figure 6. For each class, the three columns show sampled images from the (C-score ranked) top 1%, 35% and 99% percentiles, respectively. The bottom pane shows the histograms of the C-scores in each of the 5 classes.

score that can be estimated before or while a model is training *on the training instances alone*. We emphasize this latter point because if we are successful in estimating C-scores for training examples, we should also be able to estimate performance of as-yet-unseen data. We explore two C-score proxy measures based on density estimation—in input and in latent space—as well as a measure based on accuracy over the time course of training. In addition, we discuss a gradient-based measure related to the neural tangent kernel (Jacot et al., 2018) in the supplementary materials (Appendix C). All of these measures have the property that they require training only a single instance of the model and they can be used to estimate performance on a training example without explicit holdout.

4.1. Kernel Density Estimation in the Input Space

In this section, we study C-score proxies based on kernel density estimation. Intuitively, an example is consistent with the data distribution if it lies near other examples having the same label. However, if the example lies far from instances in the same class or lies near instances of different classes, one might not expect it to generalize. Based on this intuition, we define a relative local-density score:

$$\hat{C}^{\pm L}(x, y) = \frac{1}{N} \sum_{i=1}^{N} 2(\mathbf{1}[y = y_i] - \frac{1}{2})K(x_i, x), \quad (3)$$

where $K(x, x') = \exp(-\|x - x'\|^2/h^2)$ is an RBF kernel with the bandwidth h, and $\mathbf{1}[\cdot]$ is the indicator function. We introduce two additional scores as a means of determining what information in the density is critical to predicting the C-score. First, we define a class-conditional density:

$$\hat{C}^{L}(x, y) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[y = y_i]K(x_i, x), \quad (4)$$

Table 1. Rank correlation between C-score and kernel density estimation based score on inputs. Measured with Spearman's ρ and Kendall's τ rank correlations, respectively.

ρ	$\hat{C}$	$\hat{C}^L$	$\hat{C}^{\pm L}$	τ	$\hat{C}$	$\hat{C}^L$	$\hat{C}^{\pm L}$
MNIST	0.094	0.396	0.302		0.070	0.298	0.224
CF-10	−0.064	−0.009	0.083		−0.042	−0.006	0.055
CF-100	−0.098	0.117	0.105		−0.066	0.078	0.070

(Because we are mainly interested in the relative ranking of examples, we do not normalize the score to form a proper probability density function.) If $\hat{C}^{\pm L}(x, y)$ is a better proxy than $\hat{C}^L(x, y)$, then the contrast between classes is critical. Second, we define a class-independent density:

$$\hat{C}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x_i, x). \quad (5)$$

If $\hat{C}^L(x, y)$ is a better proxy than $\hat{C}(x)$, then the class labels are critical.

Table 1 shows the agreement between our three proposed proxy scores and the estimated C-score. Agreement is quantified by two rank correlation measures on three data sets. As anticipated, the input-density score that ignores labels, $\hat{C}(x)$, and the class-conditional density, $\hat{C}^L(x, y)$, have poor agreement. However, so does the class-relative score, $\hat{C}^{\pm L}(x, y)$. We therefore move on to examining the relationship among instances in hidden space.

4.2. Kernel Density Estimation in Hidden Space

Using the penultimate layer of the network as a representation of an image, we evaluate three proxy scores: $\hat{C}_h^{\pm L}(x, y)$, $\hat{C}_h^L(x, y)$, and $\hat{C}_h(x)$, with the subscript h indicating that
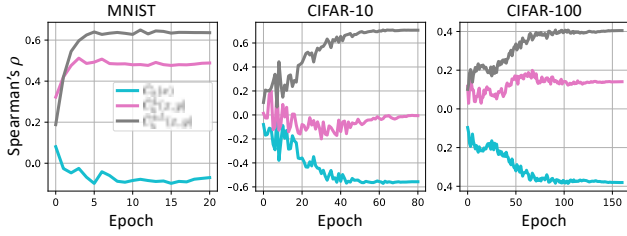
Figure 8. The rank correlation between C-score and kernel density estimation based score on hidden representations measured with Spearman's ρ.
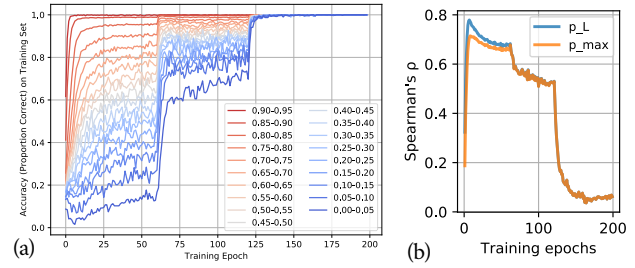

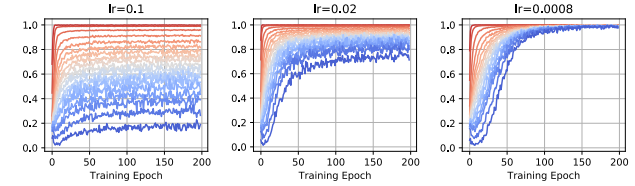*Figure 9.* (a) learning speed of groups ranked by C-scores. (b) rank correlation between confidence based scores and C-score.

Figure 10. Learning speed of groups ranked by C-scores under constant learning rate (*lr*) of 0.1, 0.02 and 0.0008, respectively.

the score operates in hidden space. For each score and data set, we compute Spearman's rank correlation between the proxy score and the C-score. We drop Kendall's τ as it closely tracked Spearman's ρ in our previous experiments. Because the embedding changes as the network is trained, we plot the correlation as a function of training epoch in Figure 8. For all three data sets, the proxy score that correlates best with the C-score is $\hat{C}_h^{\pm L}(x, y)$ (grey line), followed by $\hat{C}_h^{L}(x, y)$ (pink line), then $\hat{C}_h(x)$ (blue line). Clearly, appropriate use of labels helps with the ranking. However, our proxy $\hat{C}_h^{\pm L}$ uses the labels in an ad hoc manner. In Appendix C, we discuss a more principled measure based on gradient vectors and relate it to the neural tangent kernel (Jacot et al., 2018).

The results reveal interesting properties of the hidden representation. One might be concerned that as training progresses, the representations will optimize toward the classification loss and may discard inter-class relationships that could be potentially useful for other downstream tasks (Scott et al., 2018). However, our results suggest that $\hat{C}_h^{\pm L}$ does not diminish as a predictor of the C-score, even long after training converges. Thus, at least some information concerning the relation between different examples is retained in the representation, even though intra- and inter-class similarity is not very relevant for a classification model. To the extent that the hidden representation—crafted through a discriminative loss—preserves class structure, one might expect that the C-score could be predicted without label reweighting; however, the poor performance of $\hat{C}_h$ suggests otherwise.

Even at asymptote, $\hat{C}_h^{\pm L}(x, y)$ achieves a peak correlation of only about 0.7 for MNIST and CIFAR-10 and 0.4 for CIFAR-100. Nonetheless, the curves in Figure 8 offer an intriguing hint that information in the time course of training may be valuable for predicting the C-score. We thus investigate the time course of training itself in the next section, specifically, we examine the accuracy of an example in the training set as the network weights evolve.

4.3. Learning Speed

Intuitively, a training example that is consistent with many others should be learned quickly because the gradient steps

for all consistent examples should be well aligned. One might therefore conjecture that strong regularities in a data set are not only better learned at asymptote—leading to better generalization performance—but are also learned *sooner* in the time course of training. This *learning speed* hypothesis is nontrivial, because the C-score is defined for a held-out instance following training, whereas learning speed is defined for a training instance during training.

To test the learning-speed hypothesis, we partitioned examples in the CIFAR-10 data set into bins by integrated C-score, each bin having a width of 0.05. We then train a model on *all* examples in the data set and plot average proportion correct for each bin as a function of training epoch, as shown in Figure 9a. The two jumps in the graph correspond to points at which the learning rate is reduced. Asymptotically, all examples are learned, as one would expect from an overparameterized model. However, interestingly, the (blue) examples having the lowest C-scores are learned most slowly and the (red) examples having the highest C-scores are learned most quickly. Indeed, learning speed is monotonically related to C-score bin.

In Figure 9b, we compute the Spearman's rank correlation between the C-score of an instance and its softmax *confidence* value as a function of training epoch. We consider two definitions of confidence: $\hat{p}_L$, the softmax probability of the target class, and $\hat{p}_{\max}$, the largest probability across all classes. Both correlate well with the C-score early in training, although $\hat{p}_L$ is superior.

We also computed a correlation between an instance's C-score and an explicit measure of learning speed. One might define learning speed as the first epoch at which an example

is classified correctly, but it is known that some instances flip flop between "learned" and "forgotten" states during training (Toneva et al., 2019). Instead, we simply *count* the total number of training epochs in which the instance is classified correctly. To the extent it is learned early and reliably, the count will be large. At the end of training, Spearman's rank correlation between this *cumulative binary training loss (CBTL)* and the C-score is 0.864. Of the various proxies we have presented, the CBTL is best by far: the confidence-based scores (Figure 9b) attained $\rho < 0.800$ and were sensitive to the epoch at which the score was assessed; the best KDE score (Figure 8) attained $\rho \approx 0.700$.

An interesting observation from the learning speed plot in Figure 9a is that the stagewise learning rate decay has a greater impact for examples with lower C-scores. To explore this phenomenon further, we trained three model instances with constant learning rates of 0.1, 0.02, and 0.0008 (the same learning rates used in the stagewise schedule). As Figure 10 shows, larger learning rates appear to limit asymptotic (training) performance of the lower C-score examples. At the end of training, test accuracy for the stagewise learning-rate model is 95.1% (averaged over the last 10 epochs), whereas the constant learning-rate models attain test accuracy of only 84.8%, 91.2%, and 90.8%. Our observations suggest a plausible explanation for why we, like other computer vision researchers, have observed better generalization with stagewise learning rates than with a constant learning rate. Starting with a large learning rate effectively enforces a sort of curriculum in which the model first learns the strongest regularities. At a later stage when the learning rate is lowered and exceptions or outliers are able to be learned, the model has already built a representation based on domain regularities. In contrast, if a constant small learning rate is used (Figure 10, lr=0.0008) the outliers are learned in parallel with the regularities, which may corrupt internal representations.

5. Discussion

We explored the memorization-generalization continuum in deep learning via a *consistency score* that measures the statistical regularity of an instance in the context of a data distribution. We empirically estimated the C-score for individual instances in four data sets and we explored various proxies to the C-score based on density estimation and the time course of training. Our main contributions and take-home messages are as follows.

- We assigned a consistency score (C-score) to every example in MNIST, CIFAR-10, CIFAR-100, and ImageNet. These scores can assist in understanding a data set's structure by teasing apart regularities and subregularities and exception cases. We are currently investigating whether the scores can be used to improve generalization via curriculum learning or instance reweighting, in particular,

with the aim of encouraging networks to discover regularities before exception cases are memorized. The C-score can also be used to identify ambiguous and mislabeled examples for data cleaning and to identify difficult corner cases in safety critical applications (e.g., the perception component of a self-driving car) for active data collection.

- We explored the distribution of C-scores across all four data sets. For every class in MNIST, CIFAR-10, and CIFAR-100, high C-score examples are found that are visually uniform in color, shape, alignment (see Appendix E for more examples). The instances with lowest C-scores are often mislabeled or the salient object in the image belongs to a different class. In ImageNet, some classes do appear to have strong regularities, such as *yellow lady's slipper*. However, other classes, such as *projectile*, are more notable for their extreme diversity. Diversity seems to be reflected in the intra-class C-score variance.

- We identified the cumulative binary training loss (CBTL) as a good proxy to the C-score. The CBTL costs almost nothing to compute and requires just one training run, not thousands like the C-score. Remarkably, the CBTL is based on the *training* performance of an instance, yet it predicts ($\rho = .864$) the C-score, which is the *generalization* performance of that same instance if it were held out of the training set.

- Tracking the learning speed of examples grouped by C-score, we formulated a hypothesis to explain why a stagewise decreasing learning-rate schedule often generalizes better than a constant or adaptive schedule (more evidence in Appendix G). Our analysis suggests that the stage-wise schedule provides scaffolding to build internal representations based on the strongest domain regularities first.

Neural net researchers in the 1980s touted the fact that their models could learn *rule-governed behavior* without explicit rules (Rumelhart & McClelland, 1986). In that era, most AI researchers were focused on constructing expert systems by extracting explicit rules from human domain experts. Expert systems ultimately failed because the diversity and nuance of statistical regularities in a domain was too great for any human to explicate. In the modern deep learning era, researchers have made much progress in automatically extracting regularities from data. Nonetheless, there is still much work to be done to understand these regularities, and how the consistency relationships among instances determine the outcome of learning. By defining and investigating a consistency score, we hope to have made some progress in this direction.

References

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.

Beggel, L., Pfeiffer, M., and Bischl, B. Robust anomaly detection in images using adversarial autoencoders, 2019.

Belkin, M., Hsu, D. J., and Mitra, P. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *Advances in neural information processing systems*, pp. 2300–2311, 2018a.

Belkin, M., Ma, S., and Mandal, S. To understand deep learning we need to understand kernel learning. *arXiv preprint arXiv:1802.01396*, 2018b.

Belkin, M., Hsu, D., and Xu, J. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019.

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.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., Assent, I., and Houle, M. E. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927, 2016.

Carlini, N., Erlingsson, U., and Papernot, N. Prototypical examples in deep learning: Metrics, characteristics, and utility. Technical report, OpenReview, 2018.

Feldman, V. Does learning require memorization? A short tale about a long tail, 2019. arXiv:1906.05271.

Feldman, V. and Zhang, C. Finding the memorized examples via fast influence estimation (working title). Unpublished manuscript. Short summary in https://www.youtube.com/watch?v=YWy2Iwn-1S8, 2019.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Keskar, N. S. and Socher, R. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 7167–7177. Curran Associates, Inc., 2018.

Liang, T. and Rakhlin, A. Just interpolate: Kernel" ridgeless" regression can generalize. *arXiv preprint arXiv:1808.00387*, 2018.

Luo, L., Xiong, Y., Liu, Y., and Sun, X. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2019.

Mangalam, K. and Prabhu, V. U. Do deep neural networks learn shallow learnable examples first? In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019.

Pidhorskyi, S., Almohsen, R., and Doretto, G. Generative probabilistic novelty detection with adversarial autoencoders. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6822–6833. Curran Associates, Inc., 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Ramaswamy, S., Rastogi, R., and Shim, K. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, 29(2):427ãĂŞ438, May 2000. ISSN 0163-5808. doi: 10.1145/335191.335437.

Rumelhart, D. E. and McClelland, J. L. *On Learning the Past Tenses of English Verbs*, pp. 216ãĂŞ271. MIT Press, Cambridge, MA, USA, 1986.

Saxena, S., Tuzel, O., and DeCoste, D. Data parameters: A new family of parameters for learning a differentiable curriculum. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 11093–11103. Curran Associates, Inc., 2019.

Scott, T., Ridgeway, K., and Mozer, M. C. Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 76–85. Curran Associates, Inc., 2018.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4148–4158, 2017.

A. Experiment Details

The details on model architectures, dataset information and hyper-parameters used in the experiments for empirical estimation of the C-score can be found in Table 2.

To compute the scores based on kernel density estimation on learned representations, we train neural network models with the same specification as in Table 2 on the full training set. We use an RBF kernel $K(x, x') = \exp(-\|x - x'\|^2/h^2)$, where the bandwidth parameter h is adaptively chosen as $1/2$ of the mean pairwise Euclidean distance across the dataset.

The experiments on learning speed are conducted with ResNet-18 on CIFAR-10, trained for 200 epochs while batch size is 32. For optimizer, we use the SGD with the initial learning rate 0.1, momentum 0.9 (with Nesterov momentum) and weight decay is 5e-4. The stage-wise constant learning rate scheduler decrease the learning rate at the 60th, 90th, and 120th epoch with a decay factor of 0.2.

B. Time and Space Complexity

The time complexity of the holdout procedure for empirical estimation of the C-score is $\mathbb{O}(S(kT + E))$. Here S is the number of subset ratios, k is number of holdout for each subset ratio, and T is the average training time for a neural network.

E is the time for computing the score given the k-fold holdout training results, which involves elementwise computation on a matrix of size $k \times N$, and is negligible comparing to the time for training neural networks. The space complexity is the space for training a single neural network times the number of parallel training jobs. The space complexity for computing the scores is $\mathbb{O}(kN)$.

For kernel density estimation based scores, the most expensive part is forming the pairwise distance matrix (and the kernel matrix), which requires $\mathbb{O}(N^2)$ space and $\mathbb{O}(N^2 d)$ time, where d is the dimension of the input or hidden representation spaces.

C. Kernel Density Estimation with Gradient Representations

Most modern neural networks are trained with first order gradient descent based algorithms and variants. In each iteration, the gradient of loss on a mini-batch of training examples evaluated at the current network weights is computed and used to update the current parameter. Let $\nabla_t(\cdot)$ be the function that maps an input-label training pair (the case of mini-batch size one) to the corresponding gradient evaluated at the network weights of the t-th iteration. Then this defines a gradient based representation on which we can compute

Table 2. Details for the experiments used in the empirical estimation of the C-score.

	MNIST	CIFAR-10	CIFAR-100	ImageNet
Architecture	MLP(512,256,10)	Inception[†]	Inception[†]	ResNet-50 (V2)
Optimizer	SGD	SGD	SGD	SGD
Momentum	0.9	0.9	0.9	0.9
Base Learning Rate	0.1	0.4	0.4	0.1×7
Learning Rate Scheduler	∧(15%)[★]	∧(15%)[★]	∧(15%)[★]	LinearRampupPiecewiseConstant[★★]
Batch Size	256	512	512	128×7
Epochs	20	80	160	100
Data Augmentation	······ Random Padded Cropping[⊛] + Random Left-Right Flipping ······			
Image Size	28×28	32×32	32×32	224×224
Training Set Size	60,000	50,000	50,000	1,281,167
Number of Classes	10	10	100	1000

[†] A simplified Inception model suitable for small image sizes, defined as follows:

    Inception :: Conv(3×3, 96) → Stage1 → Stage2 → Stage3 → GlobalMaxPool → Linear.

      Stage1 :: Block(32, 32) → Block(32, 48) → Conv(3×3, 160, Stride=2).

      Stage2 :: Block(112, 48) → Block(96, 64) → Block(80, 80) → Block (48, 96) → Conv(3×3, 240, Stride=2).

      Stage3 :: Block(176, 160) → Block(176, 160).

    Block(C_1, C_2) :: Concat(Conv(1×1, C_1), Conv(3×3,C_2)).

      Conv :: Convolution → BatchNormalization → ReLU.

[★] ∧(15%) learning rate scheduler linearly increase the learning rate from 0 to the *base learning rate* in the first 15% training steps, and then from there linear decrease to 0 in the remaining training steps.

[★★] LinearRampupPiecewiseConstant learning rate scheduler linearly increase the learning rate from 0 to the *base learning rate* in the first 15% training steps. Then the learning rate remains piecewise constant with a 10× decay at 30%, 60% and 90% of the training steps, respectively.

[⊛] Random Padded Cropping pad 4 pixels of zeros to all the four sides of MNIST, CIFAR-10, CIFAR-100 images and (randomly) crop back to the original image size. For ImageNet, a padding of 32 pixels is used for all four sides of the images.

density based ranking scores. The intuition is that in a gradient based learning algorithm, an example is consistent with others if they all compute similar gradients.

Comparing to the hidden representations defined the outputs of a neural network layer, the gradient based representations induce a more natural way of incorporating the label information. In the previous section, we reweight the neighbor examples belonging to a different class by 0 or -1. For gradient based representations, no ad hoc reweighting is needed as the gradient is computed on the loss that has already takes the label into account. Similar inputs with different labels automatically lead to dissimilar gradients. Moreover, this could seamlessly handle labels and losses with rich structures (e.g. image segmentation, machine translation) where an effective reweighting scheme is hard to find. The gradient based representation is closely related to recent developments on Neural Tagent Kernels (NTK) (Jacot et al., 2018). It is shown that when the network width goes to infinity, the neural network training dynamics can be effectively approximately via Taylor expansion at the initial network weights. In other words, the algorithm is effectively learning a *linear* model on the *nonlinear* representations defined by $\nabla_0(\cdot)$. This feature map induces the NTK, and connects deep learning to the literature of kernel machines.

Although NTK enjoys nice theoretical properties, it is challenging to perform density estimation on it. Even for the more practical case of *finite width* neural networks, the gradient representations are of extremely high dimensions as modern neural networks general have parameters ranging from millions to billions (e.g. Tan & Le, 2019; Radford et al., 2019). As a result, both computation and memory requirements are prohibitive if a naive density estimation is to be computed on the gradient representations. We leave as future work to explore efficient algorithms to practically compute this score.

D. Point Estimation of Integral C-score

The histogram of individual point estimated C-scores with fixed subset ratios on CIFAR-10 is shown in Figure 2. The same plot for MNIST and CIFAR-100 are shown in Figure 11.

Similarly, the histogram for the estimated integral C-scores for MNIST and CIFAR-100 are shown in Figure 12, which can be compared with the results for CIFAR-10 in Figure 3 in the main text.

E. Examples of Images Ranked by C-score

Examples with high, middle and low C-scores from a few representative classes of MNIST, CIFAR-10 and CIFAR-100 are show in Figure 4. In this appendix, we depict the results for all the 10 classes of MNIST and CIFAR-10 in Figure 13 and
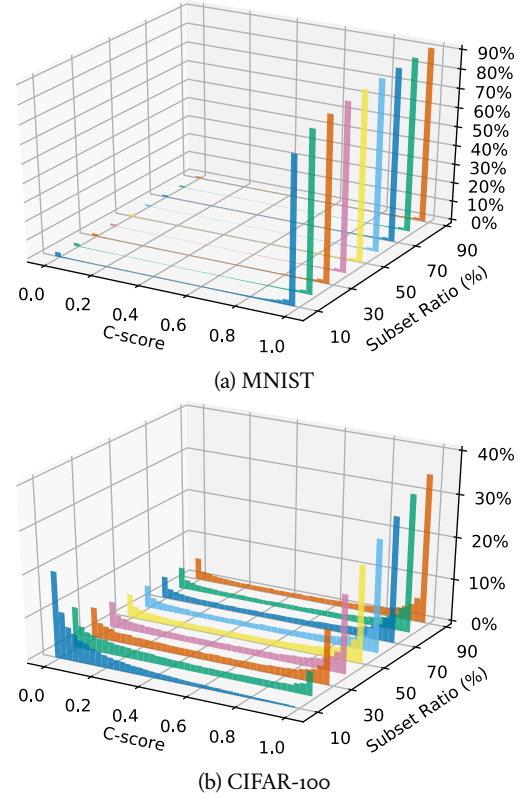

(a) MNIST

(b) CIFAR-100

Figure 11. Histogram of C-scores for MNIST for subset ratios $s \in \{10, 20, 30, \dots 90\}$. The vertical axis is the percentage of the 50,000 instances that fall into a bin.
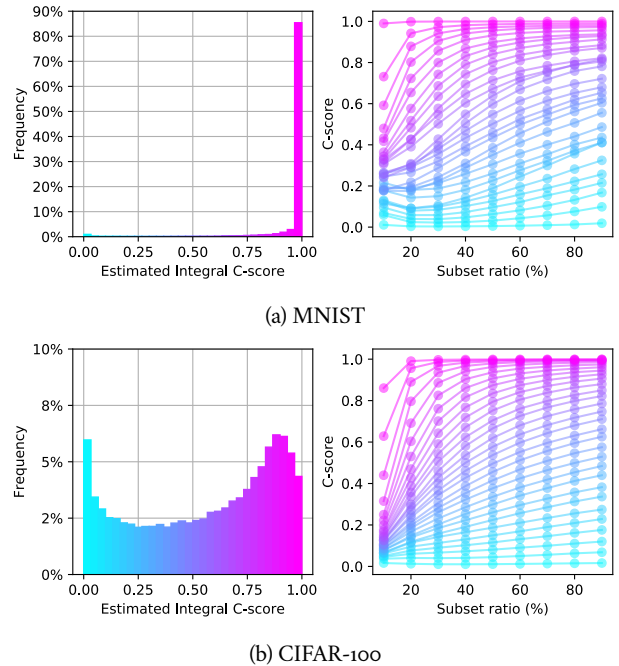

(a) MNIST

(b) CIFAR-100

Figure 12. (left) histogram of estimated integral C-scores; (right) C-scores by histogram bin as a function of subset ratio s.
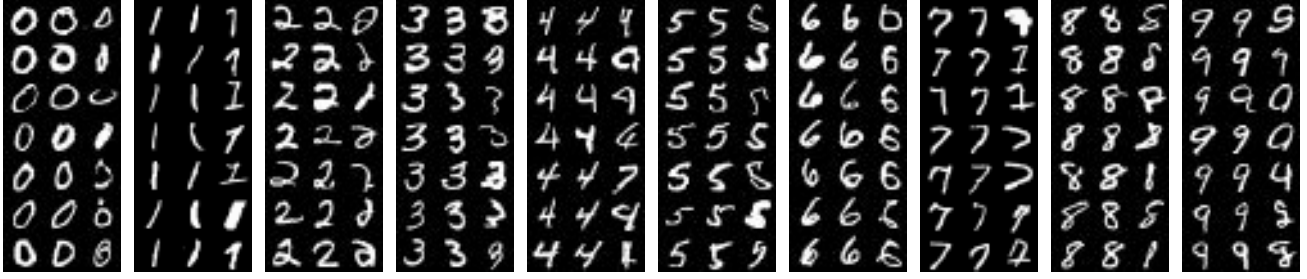
Figure 13. Examples from MNIST. Each block shows a single class; the left, middle, and right columns of a block depict instances with high, intermediate, and low C-scores, respectively.
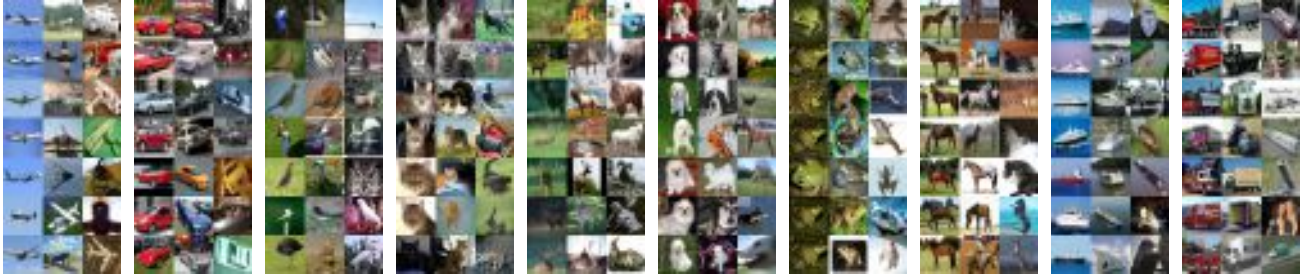

Figure 14. Examples from CIFAR-10. Each block shows a single class; the left, middle, and right columns of a block depict instances with high, intermediate, and low C-scores, respectively.

Figure 14, respectively. The results from the first 60 out of the 100 classes on CIFAR-100 is depicted in Figure 15.

F. What Makes an Item Regular or Irregular?

The notion of regularity is primarily coming from the statistical consistency of the example with the rest of the population, but less from the intrinsic structure of the example's contents. To illustrate this, we refer back to the experiments in Section 4.3 on measuring the learning speed of groups of examples generated via equal partition on the C-score value range [0, 1]. As shown in Figure 3a, the distribution is uneven between high and low C-score values. As a result, the high C-score groups will have more examples than the low C-score groups. This agrees with the intuition that regularity arises from high probability masses.

To test whether an example with top-ranking C-score is still highly regular after the density of its neighborhood is reduced, we redo the experiment, but subsample each group to contain an equal number ($\sim$ 400) of examples. Then we run training on this new dataset and observe the learning speed in each (subsampled) group. The result is shown in Figure 16, which is to be compared with the results without group-size-equalizing in Figure 9a in the main text. The following observations can be made:

1. The learning curves for many of the groups start to overlap with each other.

2. The lower ranked groups now learns faster. For example, the lowest ranked group goes above 30% accuracy near epoch 50. In the original experiment (Figure 9a), this groups is still below 20% accuracy at epoch 50. The model is now learning with a much smaller dataset. Since the lower ranked examples are not highly consistent with the rest of the population, this means there are fewer "other examples" to compete with (i.e. those "other examples" will move the weights towards a direction that is less preferable for the lower ranked examples). As a result, the lower ranked groups can now learn faster.

3. On the other hand, the higher ranked groups now learn slower, which is clear from a direct comparison between Figure 9a and Figure 16. This is because for highly regular examples, reducing the dataset size means removing consistent examples — that is, there are now less "supporters" as oppose to less "competitors" in the case of lower ranked groups. As a result, the learn speed is now slower.

4. Even though the learning curves are now overlapping, the highest ranked group and the lowest ranked group are still clearly separated. The potential reason is that while the lower ranked examples can be outliers in many different ways, the highest ranked examples are probably regular in a single (or very few) visual clusters (see the top ranked examples in Figure 4). As a result, the
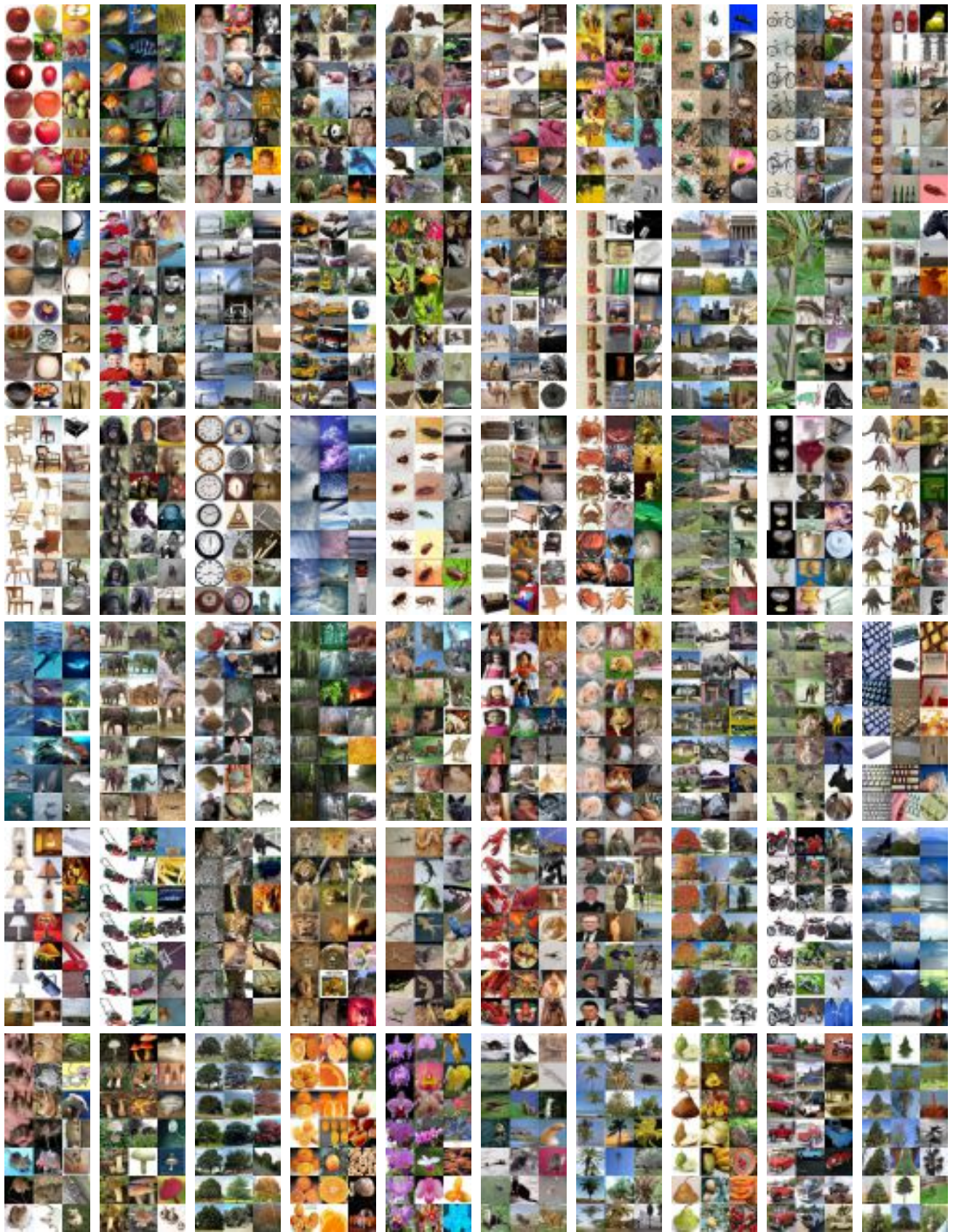
Figure 15. Examples from CIFAR-100. Each block shows a single class; the left, middle, and right columns of a block depict instances with high, intermediate, and low C-scores, respectively. The first 60 (out of the 100) classes are shown.
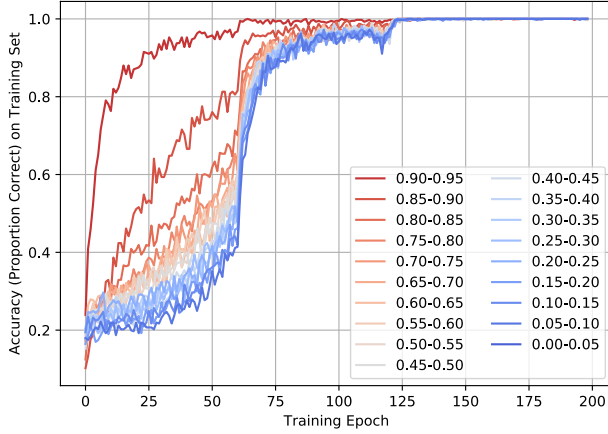
Figure 16. Learning speed of group of examples ranked by C-scores, with equal number (400) of examples in each group via subsampling.

within group diversities of the highest ranked groups are still much smaller than the lowest ranked groups.

In summary, the regularity of an example arises from its consistency relation with the rest of the population. A regular example in isolation is no different to an outlier. Moreover, it is also not merely an intrinsic property of the data distribution, but is closely related to the model, loss function and learning algorithms. For example, while a picture with a red lake and a purple forest is likely be considered an outlier in the usual sense, for a model that only uses grayscale information it could be highly regular.

G. Learning Rate Scheduling and Generalization

In Section 4.3 we used the C-score grouping to compare the learning dynamics of a stage-wise constant learning rate scheduler and a constant learning rate scheduler. The observations lead to an interesting hypothesis for explaining why stage-wsie constant learning rate usually perform better and is preferred in many computer vision tasks. We provide more details here, and also compare to Adam, an optimizer with adaptive learning rate scheduling.

Figure 17 shows the learning speed of groups of examples on CIFAR-10 ranked by C-score, with SGD using stage-wise constant learning rate scheduling. This is the same as Figure 9a, replicated here for easy comparison. In Figure 18 we show the learning speeds of groups trained with SGD using constant learning rate scheduling. The 4 panels show the results for the each of the values used in the 4 stages of the stage-wise scheduler. In Figure 19 we also present the training results with the Adam optimizer, using the default base learning rate of 0.001. Adaptive algorithms like Adam scale
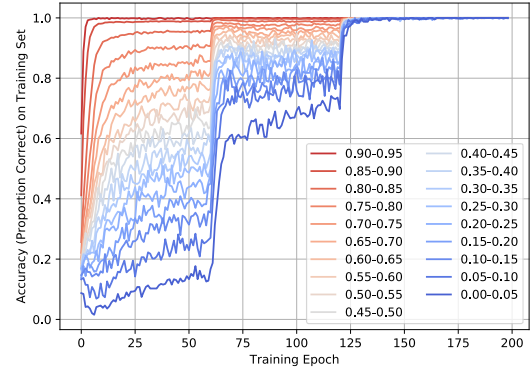

Figure 17. Learning speed of examples grouped by C-score with SGD using stage-wise constant learning rate.
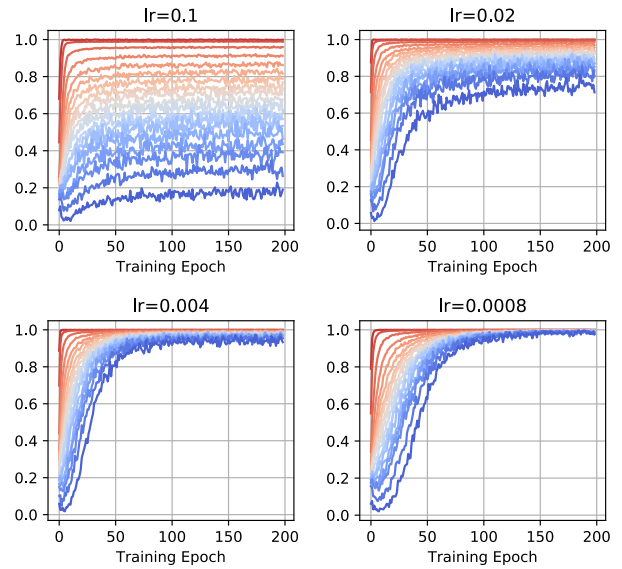

Figure 18. Learning speed of examples grouped by C-score with SGD using constant learning rate. The 4 different learning rates correspond to the constants used in the stage-wise scheduler in Figure 17.
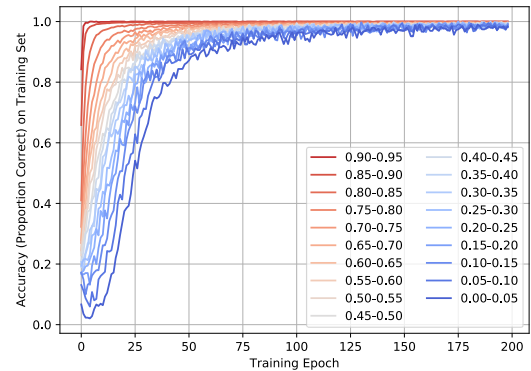

Figure 19. Learning speed of examples grouped by C-score with Adam optimizer.

Table 3. Test performance of models trained with various optimizers and learning rate schedulers on CIFAR-10.

Optimizer	Learning Rate	Test Accuracy (%)
SGD	Stage-wise	**95.14**
SGD	0.1	84.84
SGD	0.02	91.19
SGD	0.004	92.05
SGD	0.0008	90.82
Adam	Adaptive	92.97

the learning rate automatically and usually converge faster than vanilla SGD. However, it is observed that faster convergence from adaptive algorithms usually leads to worse generalization performances (Wilson et al., 2017; Keskar & Socher, 2017; Luo et al., 2019). In fact, similar behaviors are observed, as summarized in Table 3.

To restate the hypothesis: the reason that stage-wise learning rate scheduler generalize better than others is that it delayed the memorization of outliers (low C-score examples) to later stages. In the first stage, when only the regular examples are learned, the patterns and structures discovered in those regular examples can be used to build a generalizable representation. In later stages, the memorization of outliers will not seriously disrupt the learned representation as the learning rate is much smaller than the earlier stages. In contrast, both Adam and SGD with (small) constant learning rate learn the examples across all C-score ranges fairly quickly. As a result, the model do not have a chance to build a generalizable representation from a clean subset of highly regular examples.

Our experiments are by no means extensive enough to fully verify this hypothesis. However, we think this is an very interesting side observation from our experiments that is worth mention. It also provide a concrete example of how our C-score indexing could be useful for research topics on analyzing and understanding. We leave it as future work to systematically investigate the aforementioned hypothesis.