A Closer Look at Memorization in Deep Networks

Devansh Arpit [*1 2] **Stanisław Jastrzębski** [*3] **Nicolas Ballas** [*1 2] **David Krueger** [*1 2] **Emmanuel Bengio** [4]
Maxinder S. Kanwal [5] **Tegan Maharaj** [1 6] **Asja Fischer** [7] **Aaron Courville** [1 2 8] **Yoshua Bengio** [1 2 9]
Simon Lacoste-Julien [1 2]

Abstract

We examine the role of memorization in deep learning, drawing connections to capacity, generalization, and adversarial robustness. While deep networks are capable of memorizing noise data, our results suggest that they tend to prioritize learning simple patterns first. In our experiments, we expose qualitative differences in gradient-based optimization of deep neural networks (DNNs) on noise vs. real data. We also demonstrate that for appropriately tuned explicit regularization (e.g., dropout) we can degrade DNN training performance on noise datasets without compromising generalization on real data. Our analysis suggests that the notions of effective capacity which are dataset independent are unlikely to explain the generalization performance of deep networks when trained with gradient based methods because training data itself plays an important role in determining the degree of memorization.

1. Introduction

The traditional view of generalization holds that a model with sufficient capacity (e.g. more parameters than training examples) will be able to "memorize" each example, overfitting the training set and yielding poor generalization to validation and test sets (Goodfellow et al., 2016). Yet deep neural networks (DNNs) often achieve excellent generalization performance with massively over-parameterized models. This phenomenon is not well-understood.

From a representation learning perspective, the generalization capabilities of DNNs are believed to stem from their incorporation of good generic priors (see, e.g., Bengio et al. (2009)). Lin & Tegmark (2016) further suggest that the priors of deep learning are well suited to the physical world. But while the priors of deep learning may help explain why DNNs learn to efficiently represent complex real-world functions, they are not restrictive enough to rule out memorization.

On the contrary, deep nets are known to be universal approximators, capable of representing arbitrarily complex functions given sufficient capacity (Cybenko, 1989; Hornik et al., 1989). Furthermore, recent work has shown that the expressiveness of DNNs grows exponentially with depth (Montufar et al., 2014; Poole et al., 2016). These works, however, only examine the *representational capacity*, that is, the set of hypotheses a model is capable of expressing via some value of its parameters.

Because DNN optimization is not well-understood, it is unclear which of these hypotheses can actually be reached by gradient-based training (Bottou, 1998). In this sense, optimization and generalization are entwined in DNNs. To account for this, we formalize a notion of the *effective capacity (EC)* of a learning algorithm $\mathcal{A}$ (defined by specifying both the model *and the training procedure*, e.g.,"train the LeNet architecture (LeCun et al., 1998) for 100 epochs using stochastic gradient descent (SGD) with a learning rate of 0.01") as the set of hypotheses which can be reached by applying that learning algorithm on *some* dataset. Formally, using set-builder notation:

$$EC(\mathcal{A}) = \{h \mid \exists \mathcal{D} \text{ such that } h \in \mathcal{A}(\mathcal{D})\} \ ,$$

where $\mathcal{A}(\mathcal{D})$ represents the set of hypotheses that is reachable by $\mathcal{A}$ on a dataset $\mathcal{D}$[1].

One might suspect that DNNs effective capacity is sufficiently limited by gradient-based training and early stopping to resolve the apparent paradox between DNNs' excellent generalization and their high representational capacity. However, the experiments of Zhang et al. (2017) suggest that this is not the case. They demonstrate that DNNs are

[*]Equal contribution [1]Montréal Institute for Learning Algorithms, Canada [2]Université de Montréal, Canada [3]Jagiellonian University, Krakow, Poland [4]McGill University, Canada [5]University of California, Berkeley, USA [6]Polytechnique Montréal, Canada [7]University of Bonn, Bonn, Germany [8]CIFAR Fellow [9]CIFAR Senior Fellow. Correspondence to: <david.krueger@umontreal.ca>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

[1] Since $\mathcal{A}$ can be stochastic, $\mathcal{A}(\mathcal{D})$ is a set.

able to fit pure noise without even needing substantially longer training time. Thus even the *effective* capacity of DNNs may be too large, from the point of view of traditional learning theory.

By demonstrating the ability of DNNs to "memorize" random noise, Zhang et al. (2017) also raise the question whether deep networks use similar memorization tactics on real datasets. Intuitively, a brute-force memorization approach to fitting data does not capitalize on patterns shared between training examples or features; the *content* of what is memorized is irrelevant. A paradigmatic example of a memorization algorithm is k-nearest neighbors (Fix & Hodges Jr, 1951). Like Zhang et al. (2017), we do not formally define memorization; rather, we investigate this intuitive notion of memorization by training DNNs to fit random data.

Main Contributions

We operationalize the definition of "memorization" as *the behavior exhibited by DNNs trained on noise*, and conduct a series of experiments that contrast the learning dynamics of DNNs on real vs. noise data. Thus, our analysis builds on the work of Zhang et al. (2017) and further investigates the role of memorization in DNNs.

Our findings are summarized as follows:

1. There are qualitative differences in DNN optimization behavior on real data vs. noise. In other words, DNNs do not just memorize real data (Section 3).

2. DNNs learn simple patterns first, before memorizing (Section 4). In other words, DNN optimization is *content-aware*, taking advantage of patterns shared by multiple training examples.

3. Regularization techniques can differentially hinder memorization in DNNs while preserving their ability to learn about real data (Section 5).

2. Experiment Details

We perform experiments on MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky et al.) datasets. We investigate two classes of models: 2-layer multi-layer perceptrons (MLPs) with rectifier linear units (ReLUs) on MNIST and convolutional neural networks (CNNs) on CIFAR10. If not stated otherwise, the MLPs have 4096 hidden units per layer and are trained for 1000 epochs with SGD and learning rate 0.01. The CNNs are a small Alexnet-style CNN[2] (as in Zhang et al. (2017)), and are trained using SGD with momentum=0.9 and learning rate of 0.01, scheduled to drop by half every 15 epochs.

Following Zhang et al. (2017), in many of our experiments we replace either (some portion of) the labels (with random labels), or the inputs (with i.i.d. Gaussian noise matching the real dataset's mean and variance) for some fraction of the training set. We use *randX* and *randY* to denote datasets with (100%, unless specified) noisy inputs and labels (respectively).

3. Qualitative Differences of DNNs Trained on Random vs. Real Data

Zhang et al. (2017) empirically demonstrated that DNNs are capable of fitting random data, which implicitly necessitates some high degree of memorization. In this section, we investigate whether DNNs employ similar memorization strategy when trained on real data. In particular, our experiments highlight some qualitative differences between DNNs trained on real data vs. random data, supporting the fact that DNNs do not use brute-force memorization to fit real datasets.

3.1. Easy Examples as Evidence of Patterns in Real Data

A brute-force memorization approach to fitting data should apply equally well to different training examples. However, if a network is learning based on patterns in the data, some examples may fit these patterns better than others. We show that such "easy examples" (as well as correspondingly "hard examples") are common in real, but not in random, datasets. Specifically, for each setting (real data, randX, randY), we train an MLP for a single epoch starting from 100 different random initializations and shufflings of the data. We find that, for real data, many examples are consistently classified (in)correctly after a single epoch, suggesting that different examples are significantly easier or harder in this sense. For noise data, the difference between examples is much less, indicating that these examples are fit (more) independently. Results are presented in Figure 1.

For randX, apparent differences in difficulty are well modeled as random Binomial noise. For randY, this is not the case, indicating some use of shared patterns. Visualizing first-level features learned by a CNN supports this hypothesis (Figure 2).

[2]Input $\rightarrow$ Crop(2,2) $\rightarrow$ Conv(200,5,5) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ MaxPooling(3,3) $\rightarrow$ Conv(200,5,5) $\rightarrow$ BN$\rightarrow$ ReLU$\rightarrow$ MaxPooling(3,3) $\rightarrow$ Dense(384) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ Dense(192) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ Dense(#classes) $\rightarrow$ Softmax. Here Crop(. , .) crops height and width from both sides with respective values.

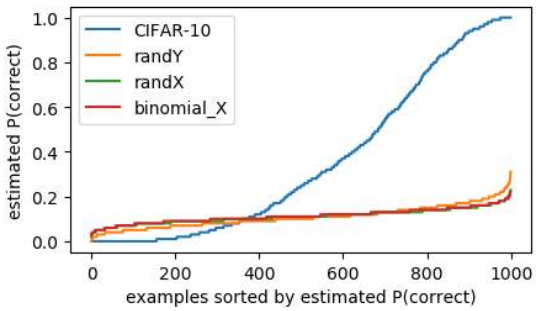


Figure 1. Average (over 100 experiments) misclassification rate for each of 1000 examples after one epoch of training. This measure of an example's difficulty is much more variable in real data. We conjecture this is because the easier examples are explained by some simple patterns, which are reliably learned within the first epoch of training. We include 1000 points samples from a binomial distribution with $n = 100$ and p equal to the average estimated P(correct) for randX, and note that this curve closely resembles the randX curve, suggesting that random inputs are all equally difficult.

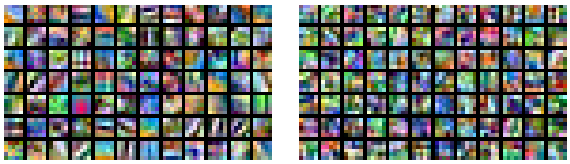


Figure 2. Filters from first layer of network trained on CIFAR10 (left) and randY (right).

3.2. Loss-Sensitivity in Real vs. Random Data

To further investigate the difference between real and fully random inputs, we propose a proxy measure of memorization via gradients. Since we cannot measure quantitatively how much each training sample $\mathbf{x}$ is memorized, we instead measure the effect of each sample on the average loss. That is, we measure the norm of the loss gradient with respect to a previous example $\mathbf{x}$ after t SGD updates. Let $\mathcal{L}_t$ be the loss after t updates; then the sensitivity measure is given by

$$g_{\mathbf{x}}^t = \|\partial \mathcal{L}_t / \partial \mathbf{x}\|_1 \ .$$

The parameter update from training on $\mathbf{x}$ influences all future $\mathcal{L}_t$ indirectly by changing the subsequent updates on different training examples. We denote the average over $g_{\mathbf{x}}^t$ after T steps as $\bar{g}_{\mathbf{x}}$, and refer to it as *loss-sensitivity*. Note that we only report ℓ^1-norm results, but that results stay very similar using ℓ^2-norm and infinity norm.

We compute $g_{\mathbf{x}}^t$ by unrolling t SGD steps and applying backpropagation over the unrolled computation graph, as done by Maclaurin et al. (2015). Unlike Maclaurin et al. (2015), we only use this procedure to compute $g_{\mathbf{x}}^t$, and do not modify the training procedure in any way.

We find that for real data, only a subset of the training set has high $\bar{g}_{\mathbf{x}}$, while for random data, $\bar{g}_{\mathbf{x}}$ is high for virtually all examples. We also find a different behavior when *each example* is given a unique class; in this scenario, the network has to learn to identify each example uniquely, yet still behaves differently when given real data than when given random data as input.

We visualize (Figure 3) the spread of $\bar{g}_{\mathbf{x}}$ as training progresses by computing the Gini coefficient over $\mathbf{x}$'s. The Gini coefficient (Gini, 1913) is a measure of the inequality among values of a frequency distribution; a coefficient of 0 means exact equality (i.e., all values are the same), while a coefficient of 1 means maximal inequality among values. We observe that, when trained on real data, the network has a high $\bar{g}_{\mathbf{x}}$ for a few examples, while on random data the network is sensitive to most examples. The difference between the random data scenario, where we know the neural network needs to do memorization, and the real data scenario, where we're trying to understand what happens, leads us to believe that this measure is indeed sensitive to memorization. Additionally, these results suggest that when being trained on real data, the neural network probably does not memorize, or at least not in the same manner it needs to for random data.

In addition to the different behaviors for real and random data described above, we also consider a class specific loss-sensitivity: $\bar{g}_{i,j} = \mathbb{E}_{(x,y)} 1/T \sum_t^T |\partial \mathcal{L}_t(y = i)/\partial x_{y=j}|$, where $\mathcal{L}_t(y = i)$ is the term in the crossentropy sum corresponding to class i. We observe that the loss-sensitivity w.r.t. class i for training examples of class j is higher when $i = j$, but more spread out for real data (see Figure 4). An interpretation of this is that for real data there are more interesting cross-category patterns that can be learned than for random data.

Figure 3 and 4 were obtained by training a fully-connected network with 2 layers of 16 units on 1000 downscaled 14×14 MNIST digits using SGD.

3.3. Capacity and Effective Capacity

In this section, we investigate the impact of capacity and effective capacity on learning of datasets having different amounts of random input data or random labels.

3.3.1. Effects of Capacity and Dataset Size on Validation Performances

In a first experiment, we study how overall model capacity impacts the validation performances for datasets with different amounts of noise. On MNIST, we found that the optimal validation performance requires a higher capacity model in the presence of noise examples (see Figure 5). This trend was consistent for noise inputs on CIFAR10, but

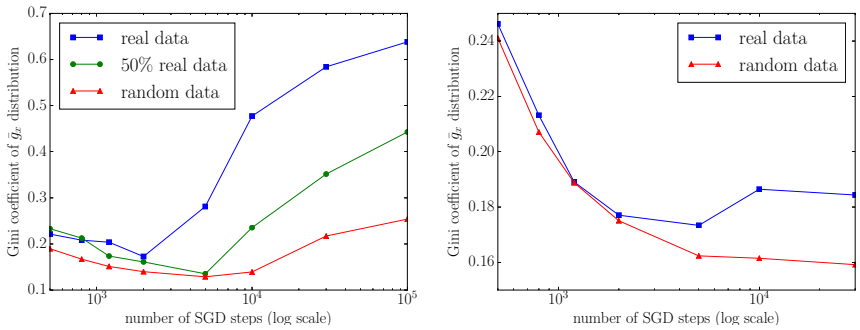


Figure 3. Plots of the Gini coefficient of $\bar{g}_{\mathbf{x}}$ over examples $\mathbf{x}$ (see section 3.2) as training progresses, for a 1000-example real dataset (14x14 MNIST) versus random data. On the left, Y is the normal class label; on the right, there are as many classes as examples, the network has to learn to map each example to a unique class.

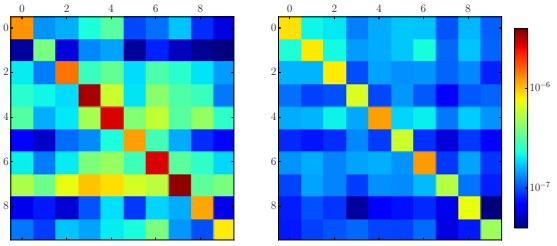


Figure 4. Plots of per-class g_x (see previous figure; log scale), a cell i, j represents the average $|\partial\mathcal{L}(y=i)/\partial x_{y=j}|$, i.e. the loss-sensitivity of examples of class i w.r.t. training examples of class j. Left is real data, right is random data.

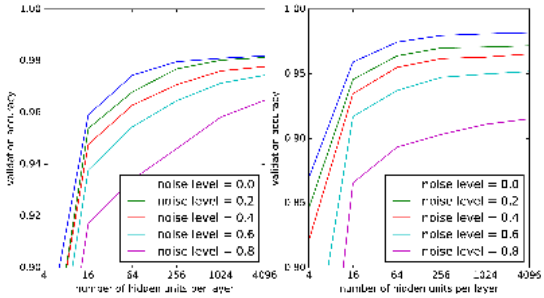


Figure 5. Performance as a function of capacity in 2-layer MLPs trained on (noisy versions of) MNIST. For real data, performance is already very close to maximal with 4096 hidden units, but when there is noise in the dataset, higher capacity is needed.

we did not notice any relationship between capacity and validation performance on random *labels* on CIFAR10.

This result contradicts the intuitions of traditional learning theory, which suggest that capacity should be restricted, in order to enforce the learning of (only) the most regular patterns. Given that DNNs can perfectly fit the training set in any case, we hypothesize that that higher capacity allows the network to fit the noise examples in a way that does not interfere with learning the real data. In contrast, if we were simply to *remove* noise examples, yielding a smaller (clean) dataset, a *lower* capacity model would be able to achieve optimal performance.

3.3.2. Effects of Capacity and Dataset Size on Training Time

Our next experiment measures time-to-convergence, i.e. how many epochs it takes to reach 100% training accuracy. Reducing the capacity or increasing the size of the dataset slows down training as well for real as for noise data[3]. However, the effect is more severe for datasets containing noise, as our experiments in this section show (see Figure 6).

Effective capacity of a DNN can be increased by increasing the representational capacity (e.g. adding more hidden units) or training for longer. Thus, increasing the number of hidden units decreases the number of training iterations needed to fit the data, up to some limit. We observe *stronger* diminishing returns from increasing representational capacity for real data, indicating that this limit is lower, and a smaller representational capacity is sufficient, for real datasets.

Increasing the number of examples (keeping representational capacity fixed) also increases the time needed to memorize the training set. In the limit, the representational capacity is simply insufficient, and memorization is not feasible. On the other hand, when the relationship between inputs and outputs is meaningful, new examples sim-

[3] Regularization can also increase time-to-convergence; see section 5.

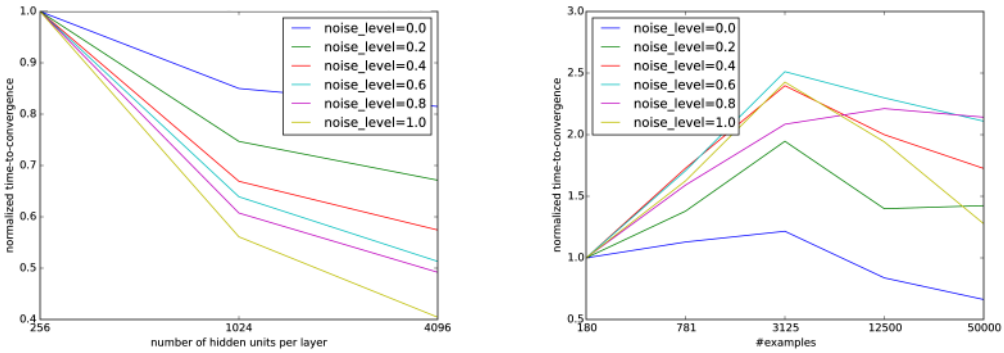


Figure 6. Time to convergence as a function of capacity with dataset size fixed to 50000 (left), or dataset size with capacity fixed to 4096 units (right). "Noise level" denotes to the proportion of training points whose inputs are replaced by Gaussian noise. Because of the patterns underlying real data, having more capacity/data does not decrease/increase training time as much as it does for noise data.

ply give more (possibly redundant) clues as to what the input $\rightarrow$ output mapping is. Thus, in the limit, an idealized learner should be able to predict unseen examples perfectly, absent noise. Our experiments demonstrate that time-to-convergence is not only longer on noise data (as noted by Zhang et al. (2017)), but also, *increases* substantially as a function of dataset size, relative to real data. Following the reasoning above, this suggests that our networks are learning to extract patterns in the data, rather than memorizing.

4. DNNs Learn Patterns First

This section aims at studying how the complexity of the hypotheses learned by DNNs evolve during training for real data vs. noise data. To achieve this goal, we build on the intuition that the number of different decision regions into which an input space is partitioned reflects the complexity of the learned hypothesis (Sokolic et al., 2016). This notion is similar in spirit to the degree to which a function can scatter random labels: a higher density of decision boundaries in the data space allows more samples to be scattered.

Therefore, we estimate the complexity by measuring how densely points on the data manifold are present around the model's decision boundaries. Intuitively, if we were to randomly sample points from the data distribution, a smaller fraction of points in the proximity of a decision boundary suggests that the learned hypothesis is simpler.

4.1. Critical Sample Ratio (CSR)

Here we introduce the notion of a *critical sample*, which we use to estimate the density of decision boundaries as discussed above. Critical samples are a subset of a dataset such that for each such sample $\mathbf{x}$, there exists at least one adversarial example $\hat{\mathbf{x}}$ in the proximity of $\mathbf{x}$. Specifically, consider a classification network's output vector $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \in \mathbb{R}^k$ for a given input sample $\mathbf{x} \in \mathbb{R}^n$ from the data manifold. Formally we call a dataset sample $\mathbf{x}$ a *critical sample* if there exists a point $\hat{\mathbf{x}}$ such that,

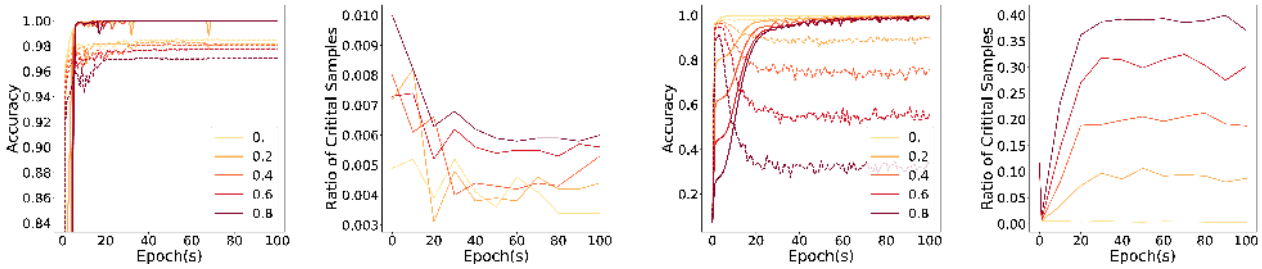
$$\arg\max_i f_i(\mathbf{x}) \neq \arg\max_j f_j(\hat{\mathbf{x}}) \quad (1)$$
$$\text{s.t. } \|\mathbf{x} - \hat{\mathbf{x}}\|_\infty \leq r$$

where r is a fixed box size. As in recent work on adversarial examples (Kurakin et al., 2016) the above definition depends only on the predicted label $\arg\max_i f_i(\mathbf{x})$ of $\mathbf{x}$, and not the true label (as in earlier work on adversarial examples, such as Szegedy et al. (2013); Goodfellow et al. (2014)).

Following the above argument relating complexity to decision boundaries, a higher number of critical samples indicates a more complex hypothesis. Thus, we measure complexity as the *critical sample ratio (CSR)*, that is, the fraction of data-points in a set $|\mathcal{D}|$ for which we can find a critical sample: $\frac{\#\text{critical samples}}{|\mathcal{D}|}$.

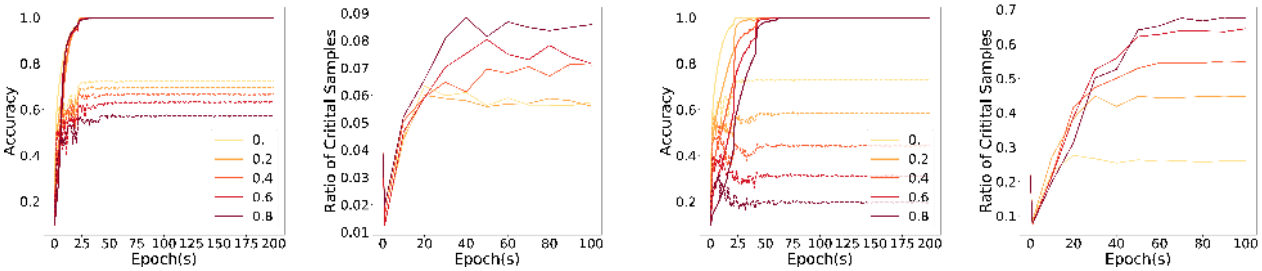
To identify whether a given data point $\mathbf{x}$ is a critical samples, we search for an adversarial sample $\hat{\mathbf{x}}$ within a box of radius r. To perform this search, we propose using Langevin dynamics applied to the fast gradient sign method (FGSM, Goodfellow et al. (2014)) as shown in algorithm 1[4]. We refer to this method as Langevin adversarial sample search (LASS). While the FGSM search algorithm can get stuck at a points with zero gradient, LASS explores the box more thoroughly. Specifically, a problem with first order gradient search methods (like FGSM) is that there might exist training points where the gradient is 0, but with a large 2^{nd} derivative corresponding to a large change in prediction in the neighborhood. The noise added by the LASS algorithm during the search enables escaping from such points.

[4]In our experiments, we set $\alpha = 0.25$, $\beta = 0.2$ and η is samples from standard normal distribution.



(a) Noise added on classification inputs. (b) Noise added on classification labels.

Figure 7. Accuracy (left in each pair, solid is train, dotted is validation) and Critical sample ratios (right in each pair) for MNIST.



(a) Noise added on classification inputs. (b) Noise added on classification labels.

Figure 8. Accuracy (left in each pair, solid is train, dotted is validation) and Critical sample ratios (right in each pair) for CIFAR10.

Algorithm 1 Langevin Adversarial Sample Search (LASS)

Require: $\mathbf{x} \in \mathbb{R}^n$, α, β, r, noise process η
Ensure: $\hat{\mathbf{x}}$

converged = FALSE
$\tilde{\mathbf{x}} \leftarrow \mathbf{x}$; $\hat{\mathbf{x}} \leftarrow \emptyset$
while not converged or max iter reached **do**
  $\Delta = \alpha \cdot \text{sign}(\frac{\partial f_k(\mathbf{x})}{\partial \mathbf{x}}) + \beta \cdot \eta$
  $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} + \Delta$
  for $i \in [n]$ **do**
    $\tilde{\mathbf{x}}_i \leftarrow \begin{cases} \mathbf{x}_i + r \cdot \text{sign}(\tilde{\mathbf{x}}_i - \mathbf{x}^i) & if |\tilde{\mathbf{x}}_i - \mathbf{x}_i| > r \\ \tilde{\mathbf{x}}_i & otherwise \end{cases}$
  end for
  if $\arg\max_i f(\mathbf{x}) \neq \arg\max_i f(\tilde{\mathbf{x}})$ **then**
    converged = TRUE
    $\hat{\mathbf{x}} \leftarrow \tilde{\mathbf{x}}$
  end if
end while

4.2. Critical Samples Throughout Training

We now show that the number of critical samples is much higher for a deep network (specifically, a CNN) trained on noise data compared with real data. To do so, we measure the number of critical samples in the validation set[5], throughout training[6]. Results are shown in Figure 9. A

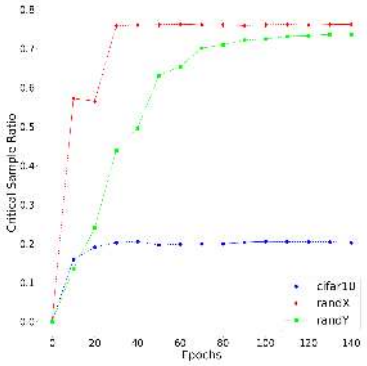


Figure 9. Critical sample ratio throughout training on CIFAR-10, random input (randX), and random label (randY) datasets.

[5] We also measure the number of critical samples in the training sets. Since we train our models using log loss, training points are pushed away from the decision boundary even after the network learns to classify them correctly. This leads to an initial rise and then fall of the number of critical samples in the training sets.

[6] We use a box size of 0.3, which is small enough in a 0-255 pixel scale to be unnoticeable by a human evaluator. Different values for r were tested but did not change results qualitatively

higher number of critical samples for models trained on noise data compared with those trained on real data suggests that the learned decision surface is more complex for noise data (randX and randY). We also observe that the CSR increases gradually with increasing number of epochs and then stabilizes. This suggests that the networks learn gradually more complex hypotheses during training for all three datasets.

In our next experiment, we evaluate the performance and critical sample ratio of datasets with 20% to 80% of the training data replaced with either input or label noise. Results for MNIST and CIFAR-10 are shown in Figures 7 and 8, respectively. For both randX and randY datasets, the CSR is higher for noisier datasets, reflecting the higher level of complexity of the learned prediction function. The final and maximum validation accuracies are also both lower for noisier datasets, indicating that the noise examples interfere somewhat with the networks ability to learn about the real data.

More significantly, for randY datasets (Figures 7(b) and 8(b)), the network achieves maximum accuracy on the validation set before achieving high accuracy on the training set. Thus the model first learns the simple and general patterns of the real data before fitting the noise (which results in decreasing validation accuracy). Furthermore, as the model moves from fitting real data to fitting noise, the CSR greatly increases, indicating the need for more complex hypotheses to explain the noise. Combining this result with our results from Section 3.1, we conclude that real data examples are easier to fit than noise.

5. Effect of Regularization on Learning

Here we demonstrate the ability of regularization to degrade training performance on data with random labels, while maintaining generalization performance on real data. Zhang et al. (2017) argue that explicit regularizations are not the main explanation of good generalization performance, rather SGD based optimization is largely responsible for it. Our findings extend their claim and indicate that explicit regularizations can substantially limit the speed of memorization of noise data without significantly impacting learning on real data.

We compare the performance of CNNs trained on CIFAR-10 and randY with the following regularizers: dropout (with dropout rates in range 0-0.9), input dropout (range 0-0.9), input Gaussian noise (with standard deviation in range 0-5), hidden Gaussian noise (range 0-0.3), weight decay (range 0-1) and additionally dropout with adversarial training (with weighting factor in range 0.2-0.7 and dropout in

and lead to the same conclusions

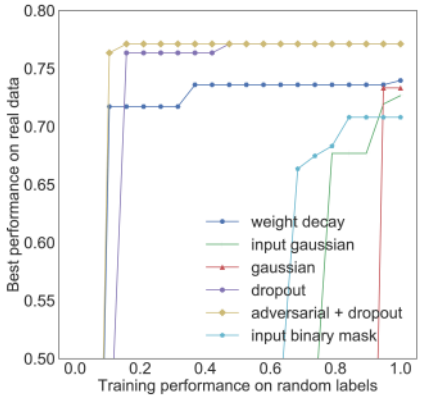


Figure 10. Effect of different regularizers on train accuracy (on noise dataset) vs. validation accuracy (on real dataset). Flatter curves indicate that memorization (on noise) can be capped without sacrificing generalization (on real data).

rate range 0.03-0.5).[7] We train a separate model for every combination of dataset, regularization technique, and regularization parameter.

The results are summarized in Figure 10. For each combination of dataset and regularization technique, the final training accuracy on randY (x-axis) is plotted against the best validation accuracy on CIFAR-10 from amongst the models trained with different regularization parameters (y-axis). Flat curves indicate that the corresponding regularization technique can reduce memorization when applied on random labeling, while resulting in the same validation accuracy on the clean validation set. Our results show that different regularizers target memorization behavior to different extent – dropout being the most effective. We find that dropout, especially coupled with adversarial training, is best at hindering memorization without reducing the model's ability to learn. Figure 11 additionally shows this effect for selected experiments (i.e. selected hyperparameter values) in terms of train loss.

6. Related Work

Our work builds on the experiments and challenges the interpretations of Zhang et al. (2017). We make heavy use of their methodology of studying DNN training in the context of noise datasets. Zhang et al. (2017) show that DNNs can perfectly fit noise and thus that their generalization ability cannot be explained through traditional statistical learning theory (e.g., see (Vapnik & Vapnik, 1998; Bartlett et al., 2005)). We agree with this finding, but show in addition that the degree of memorization and generalization in DNNs depends not only on the architecture and training

[7] We perform adversarial training using critical samples found by LASS algorithm with default parameters.

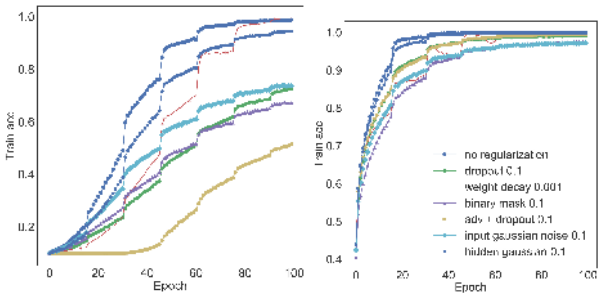


Figure 11. Training curves for different regularization techniques on random label (left) and real (right) data. The vertical ordering of the curves is different for random labels than for real data, indicating differences in the propensity of different regularizers to slow-down memorization.

procedure (including explicit regularizations), *but also on the training data itself*[8].

Another direction we investigate is the relationship between regularization and memorization. Zhang et al. (2017) argue that explicit and implicit regularizers (including SGD) might not explain or limit shattering of random data. In this work we show that regularizers (especially dropout) *do* control the *speed* at which DNNs memorize. This is interesting since dropout is also known to prevent catastrophic forgetting (Goodfellow et al., 2013) and thus in general it seems to help DNNs retain patterns.

A number of arguments support the idea that SGD-based learning imparts a regularization effect, especially with a small batch size (Wilson & Martinez, 2003) or a small number of epochs (Hardt et al., 2015). Previous work also suggests that SGD prioritizes the learning of simple hypothesis first. Sjoberg et al. (1995) showed that, for linear models, SGD first learns models with small ℓ^2 parameter norm. More generally, the efficacy of early stopping shows that SGD first learns simpler models (Yao et al., 2007). We extend these results, showing that DNNs trained with SGD learn patterns before memorizing, *even in the presence of noise examples*.

Various previous works have analyzed explanations for the generalization power of DNNs. Montavon et al. (2011) use kernel methods to analyze the complexity of deep learning architectures, and find that network priors (e.g. implemented by the network structure of a CNN or MLP) control the speed of learning at each layer. Neyshabur et al. (2014) note that the number of parameters does not control the effective capacity of a DNN, and that the reason for DNNs' generalization is unknown. We supplement this result by showing how the impact of representational capacity changes with varying noise levels. While exploring the effect of noise samples on learning dynamics has a long tradition (Bishop, 1995; An, 1996), we are the first to examine *relationships* between the fraction of noise samples and other attributes of the learning algorithm, namely: capacity, training time and dataset size.

Multiple techniques for analyzing the training of DNNs have been proposed before, including looking at generalization error, trajectory length evolution (Raghu et al., 2016), analyzing Jacobians associated to different layers (Wang; Saxe et al., 2013), or the shape of the loss minima found by SGD (Im et al., 2016; Chaudhari et al., 2016; Keskar et al., 2016). Instead of measuring the sharpness of the loss for the learned hypothesis, we investigate the complexity of the learned hypothesis throughout training and across different datasets and regularizers, as measured by the critical sample ratio. Critical samples refer to real data-points that have adversarial examples (Szegedy et al., 2013; Goodfellow et al., 2014) nearby. Adversarial examples originally referred to imperceptibly perturbed data-points that are confidently misclassified. (Miyato et al., 2015) define *virtual* adversarial examples via changes in the predictive distribution instead, thus extending the definition to unlabeled data-points. Kurakin et al. (2016) recommend using this definition when training on adversarial examples, and it is the definition we use.

Two contemporary works perform in-depth explorations of topics related to our work. Bojanowski & Joulin (2017) show that predicting random noise targets can yield state of the art results in unsupervised learning, corroborating our findings in Section 3.1, especially Figure 2. Koh & Liang (2017) use *influence functions* to measure the impact on parameter changes during training, as in our Section 3.2. They explore several promising applications for this technique, including generation of adversarial *training* examples.

7. Conclusion

Our empirical exploration demonstrates qualitative differences in DNN optimization on noise vs. real data, all of which support the claim that DNNs trained with SGD-variants first use patterns, not brute force memorization, to fit real data. However, since DNNs have the demonstrated ability to fit noise, it is unclear why they find generalizable solutions on real data; we believe that the deep learning priors including distributed and hierarchical representations likely play an important role. Our analysis suggests that memorization and generalization in DNNs depend on network architecture and optimization procedure, but also on the data itself. We hope to encourage future research on how properties of datasets influence the behavior of deep learning algorithms, and suggest a data-dependent understanding of DNN capacity as a research goal.

[8] We conclude the latter part based on experimental findings in sections 3 and 4.2

ACKNOWLEDGMENTS

We thank Akram Erraqabi, Jason Jo and Ian Goodfellow for helpful discussions. SJ was supported by Grant No. DI 2014/016644 from Ministry of Science and Higher Education, Poland. DA was supported by IVADO, CIFAR and NSERC. EB was financially supported by the Samsung Advanced Institute of Technology (SAIT). MSK and SJ were supported by MILA during the course of this work. We acknowledge the computing resources provided by ComputeCanada and CalculQuebec. Experiments were carried out using Theano (Theano Development Team, 2016) and Keras (Chollet et al., 2015).

References

An, Guozhong. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

Bartlett, Peter L, Bousquet, Olivier, Mendelson, Shahar, et al. Local rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537, 2005.

Bengio, Yoshua et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Bishop, Chris M. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Bojanowski, P. and Joulin, A. Unsupervised Learning by Predicting Noise. *ArXiv e-prints*, April 2017.

Bottou, Léon. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Chaudhari, Pratik, Choromanska, Anna, Soatto, Stefano, and LeCun, Yann. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016.

Chollet, François et al. Keras. https://github.com/fchollet/keras, 2015.

Cybenko, George. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

Fix, Evelyn and Hodges Jr, Joseph L. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951.

Gini, Corrado. Variabilita e mutabilita. *Journal of the Royal Statistical Society*, 76(3), 1913.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Goodfellow, Ian J, Mirza, Mehdi, Xiao, Da, Courville, Aaron, and Bengio, Yoshua. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Hardt, Moritz, Recht, Benjamin, and Singer, Yoram. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

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

Im, Daniel Jiwoong, Tao, Michael, and Branson, Kristin. An empirical analysis of deep network loss surfaces. *arXiv preprint arXiv:1612.04010*, 2016.

Keskar, Nitish Shirish, Mudigere, Dheevatsa, Nocedal, Jorge, Smelyanskiy, Mikhail, and Tang, Ping Tak Peter. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Koh, Pang Wei and Liang, Percy. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

Krizhevsky, Alex, Nair, Vinod, and Hinton, Geoffrey. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Kurakin, Alexey, Goodfellow, Ian, and Bengio, Samy. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. The mnist database of handwritten digits, 1998.

Lin, Henry W and Tegmark, Max. Why does deep and cheap learning work so well? *arXiv preprint arXiv:1608.08225*, 2016.

Maclaurin, Dougal, Duvenaud, David K, and Adams, Ryan P. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, pp. 2113–2122, 2015.

Miyato, Takeru, Maeda, Shin-ichi, Koyama, Masanori, Nakae, Ken, and Ishii, Shin. Distributional smoothing with virtual adversarial training. *stat*, 1050:25, 2015.

Montavon, Grégoire, Braun, Mikio L., and Müller, Klaus-Robert. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12, 2011.

Montufar, Guido F, Pascanu, Razvan, Cho, Kyunghyun, and Bengio, Yoshua. On the number of linear regions of deep neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2924–2932. Curran Associates, Inc., 2014.

Neyshabur, Behnam, Tomioka, Ryota, and Srebro, Nathan. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Poole, Ben, Lahiri, Subhaneil, Raghu, Maithreyi, Sohl-Dickstein, Jascha, and Ganguli, Surya. Exponential expressivity in deep neural networks through transient chaos. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3360–3368. Curran Associates, Inc., 2016.

Raghu, Maithra, Poole, Ben, Kleinberg, Jon, Ganguli, Surya, and Sohl-Dickstein, Jascha. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

Saxe, Andrew M, McClelland, James L, and Ganguli, Surya. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Sjoberg, J., Sjoeberg, J., Sjöberg, J., and Ljung, L. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62:1391–1407, 1995.

Sokolic, Jure, Giryes, Raja, Sapiro, Guillermo, and Rodrigues, Miguel RD. Robust large margin deep neural networks. *arXiv preprint arXiv:1605.08254*, 2016.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL http://arxiv.org/abs/1312.6199.

Theano Development Team, and others. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

Vapnik, Vladimir Naumovich and Vapnik, Vlamimir. *Statistical learning theory*, volume 1. Wiley New York, 1998.

Wang, Shengjie. Analysis of deep neural networks with the extended data jacobian matrix.

Wilson, D Randall and Martinez, Tony R. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

Yao, Yuan, Rosasco, Lorenzo, and Caponnetto, Andrea. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations (ICLR)*, 2017.