DOI:10.1145/3446776

Understanding Deep Learning (Still) Requires Rethinking Generalization

By **Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht,** and **Oriol Vinyals**

Abstract

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small gap between training and test performance. Conventional wisdom attributes small generalization error either to properties of the model family or to the regularization techniques used during training.

Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neural networks generalize well in practice. Specifically, our experiments establish that state-of-the-art convolutional networks for image classification trained with stochastic gradient methods easily fit a random labeling of the training data. This phenomenon is qualitatively unaffected by explicit regularization and occurs even if we replace the true images by completely unstructured random noise. We corroborate these experimental findings with a theoretical construction showing that simple depth two neural networks already have perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points as it usually does in practice.

We interpret our experimental findings by comparison with traditional models.

We supplement this republication with a new section at the end summarizing recent progresses in the field since the original version of this paper.

1. INTRODUCTION

For centuries, scientists, policy makers, actuaries, and salesmen alike have exploited the empirical fact that unknown outcomes, be they future or unobserved, often trace regularities found in past observations. We call this idea generalization: finding rules consistent with available data that apply to instances we have yet to encounter.

Supervised machine learning builds on statistical tradition in how it formalizes the idea of generalization. We assume observations come from a fixed data generating process, such as samples drawn from a fixed distribution. In a first optimization step, called *training*, we fit a model to a set of data. In a second step, called *testing*, we judge the model by how well it performs on newly generated data from the very same process.

This notion of generalization as *test-time performance* can seem mundane. After all, it simply requires the model to achieve consistent success on the *same* data generating process as was encountered during training. Yet the seemingly simple question of what theory underwrites the generalization ability of a model has occupied the machine learning research community for decades.

There are a variety of theories proposed to explain generalization.

Uniform convergence, margin theory, and algorithmic stability are but a few of the important conceptual tools to reason about generalization. Central to much theory are different notions of *model complexity*. Corresponding generalization bounds quantify how much data is needed as a function of a particular complexity measure. Despite much significant theoretical work, the prescriptive and descriptive value of these theories remains debated.

This work takes a step back. We do not offer any new theory of generalization. Rather, we offer a few simple experiments to interrogate the empirical import of different purported theories of generalization. With these experiments at hand, we broadly investigate what practices do and do not promote generalization, what does and does not measure generalization?

1.1. The randomization test

In our primary experiment, we create a copy of the training data where we replace each label independently by a random label chosen from the set of valid labels. A dog picture labeled "dog" might thus become a dog picture labeled "airplane". The randomization breaks any relationship between the instance, for example, the image, and the label. We then run the learning algorithm both on the natural data and on the randomized data with identical settings and model choice. By design, no generalization is possible on the randomized data. After all, we fit the model against random labels!

For any purported measure of generalization, we can now compare how it fares on the natural data versus the randomized data. If it turns out to be the same in both cases, it could not possibly be a good measure of generalization for it cannot even distinguish learning from natural data (where generalization is possible) from learning on randomized data (where no generalization is possible). Our primary observation is:

```
Deep neural networks easily fit random
labels.
```

The original version of this paper was published in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

More precisely, when trained on a completely random labeling of the true data, neural networks achieve 0 training error. The test error, of course, is no better than random chance as there is no correlation between the training labels and the test labels. In other words, by randomizing labels alone we can force the generalization error of a model to jump up considerably without changing the model, its size, hyperparameters, or the optimizer. We establish this fact for several different standard architectures trained on the CIFAR10 and ImageNet classification benchmarks. While simple to state, this observation has profound implications from a statistical learning perspective:

1. The effective capacity of neural networks is sufficient for memorizing the entire data set.
2. Even optimization on random labels remains easy. In fact, training time increases only by a small constant factor compared with training on the true labels.
3. Randomizing labels is solely a data transformation, leaving all other properties of the learning problem unchanged.

In particular, we find that many of the more popular explanations of generalization fail to capture what's happening in state-of-the-art deep learning models.

Extending on this first set of experiments, we also replace the true images by completely random pixels (e.g., Gaussian noise) and observe that convolutional neural networks continue to fit the data with zero training error. This shows that despite their structure, convolutional neural nets can fit random noise. We furthermore vary the amount of randomization, interpolating smoothly between the case of no noise and complete noise. This leads to a range of intermediate learning problems where there remains some level of signal in the labels. We observe a steady deterioration of the generalization error as we increase the noise level. This shows that neural networks are able to capture the remaining signal in the data while at the same time fit the noisy part using brute-force.

We discuss in further detail below how these observations rule out several standard generalization bounds as possible explanations for the generalization performance of state-of-the-art neural networks.

1.2. The role of regularization

Regularization can be thought of as the operational counterpart of a notion of model complexity. When the complexity of a model is very high, regularization introduces algorithmic tweaks intended to reward models of lower complexity. Regularization is a popular technique to make optimization problems "well posed": when an infinite number of solutions agree with the data, regularization breaks ties in favor of the solution with lowest complexity.

Our second set of experiments interrogates the role that regularization plays in training overparameterized neural networks. Our experiments reveal that most of the regularization techniques in deep learning are not necessary for generalization: if we turn off the regularization parameters, test-time performance remains strong. Hence, explicit regularization alone does not suffice to explain how deep learning models generalize. To summarize our finding:

Explicit regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error.

While explicit regularizers like "dropout" and "weight-decay" may not be essential for generalization, it is certainly the case that not all models that fit the training data well generalize well. Indeed, in neural networks, we almost always choose our model as the output of running stochastic gradient descent. Appealing to linear models, we analyze how SGD acts as an implicit regularizer. For linear models, SGD always converges to a solution with small norm. Hence, the algorithm itself is implicitly regularizing the solution. Indeed, we show on small data sets that even Gaussian kernel methods can generalize well with no regularization. Though this does not explain why certain architectures generalize better than other architectures, it does suggest that more investigation is needed to understand exactly what the properties are inherited by models that were trained using SGD.

1.3. Finite sample expressivity

We complement our empirical observations with a theoretical construction showing that generically large neural networks can express any labeling of the training data. More formally, we exhibit a very simple two-layer ReLU network with $p = 2n + d$ parameters that can express any labeling of any sample of size n in d dimensions. A previous construction due to Livni et al.[22] achieved a similar result with far more parameters, namely, $O(dn)$. While our depth-2 network inevitably has large width, we can also come up with a depth k network in which each layer has only $O(n/k)$ parameters.

While prior expressivity results focused on what functions neural nets can represent over the entire domain, we focus instead on the expressivity of neural nets with regard to a finite sample. In contrast to existing depth separations[13, 15, 40, 11] in function space, our result shows that even depth-2 networks of linear size can already represent any labeling of the training data.

1.4. Related prior work

Below we discuss some related prior work. In Section 6.1, we discuss recent work that followed the initial publication of our work.

Barlett[4] proved bounds on the fat shattering dimension of multilayer perceptrons with sigmoid activations in terms of the ℓ_1-norm of the weights at each node. This important result gives a generalization bound for neural nets that is independent of the network size. However, for RELU networks, the ℓ_1-norm is no longer informative. This leads to the question of whether there is a different form of capacity control that bounds generalization error for large neural nets. This question was raised in a thought-provoking work by Neyshabur et al.,[30] who argued through experiments that network size is not the main form of capacity control for

neural networks. An analogy to matrix factorization illustrated the importance of implicit regularization.

Hardt et al.[18] give an upper bound on the generalization error of a model trained with stochastic gradient descent in terms of the number of steps gradient descent took. Their analysis goes through the notion of *uniform stability*.[8] As we point out in this work, uniform stability of a learning algorithm is independent of the labeling of the training data. Hence, the concept is not strong enough to distinguish between the models trained on the true labels (small generalization error) and models trained on the random labels (large generalization error). This also highlights why the analysis of Hardt et al.[18] for nonconvex optimization was rather pessimistic, allowing only a very few passes over the data. Our results show that even empirically training neural networks is not uniformly stable for many passes over the data.

There has been much work on the representational power of neural networks, starting from universal approximation theorems for multi-layer perceptrons.[12, 25, 13, 24, 15, 40, 11] All of these results are at the *population* level characterizing which mathematical functions certain families of neural networks can express over the entire domain. We instead study the representational power of neural networks for a finite sample of size n. This leads to a very simple proof that even $O(n)$-sized two-layer perceptrons have universal finite-sample expressivity.

2. EFFECTIVE CAPACITY OF NEURAL NETWORKS

The size of a model family is often huge as it counts all possible functions in a certain set, including those that are unlikely to be found by the learning algorithm. By *effective capacity*, we informally refer to the size of the subset of models that is effectively achievable by the learning procedure. The capacity of this subset could be much smaller as it contains only "well-behaved" functions produced by some specific optimization algorithms, with bounded computation budget, and sometimes with explicit or implicit regularizations. Our goal is to understand the effective model capacity of feed-forward neural networks. Toward this goal, we choose a methodology inspired by nonparametric randomization tests. Specifically, we take a candidate architecture and train it both on the true data and on a copy of the data in which the true labels were replaced by random labels. In the second case, there is no longer any relationship between the instances and the class labels. As a result, learning is impossible. Intuition suggests that this impossibility should manifest itself clearly during training, for example, by training not converging or slowing down substantially. To our surprise, several properties of the training process for multiple standard architectures are largely unaffected by this transformation of the labels. This poses a conceptual challenge. Whatever justification we had for expecting a small generalization error to begin with must no longer apply to the case of random labels.

To gain further insight into this phenomenon, we experiment with different levels of randomization exploring the continuum between no label noise and completely corrupted labels. We also try out different randomizations of the inputs (rather than labels), arriving at the same general conclusion.

The experiments are run on two image classification datasets, the CIFAR10 dataset and the ImageNet ILSVRC 2012 dataset. We test the *Inception V3* architecture on ImageNet and a smaller version of Inception, Alexnet, and MLPs on CIFAR10. Please see Appendix A of Zhang et al.[44] for more details of the experimental setup.

2.1. Fitting random labels and pixels

We run our experiments with the following modifications of the labels and input images:

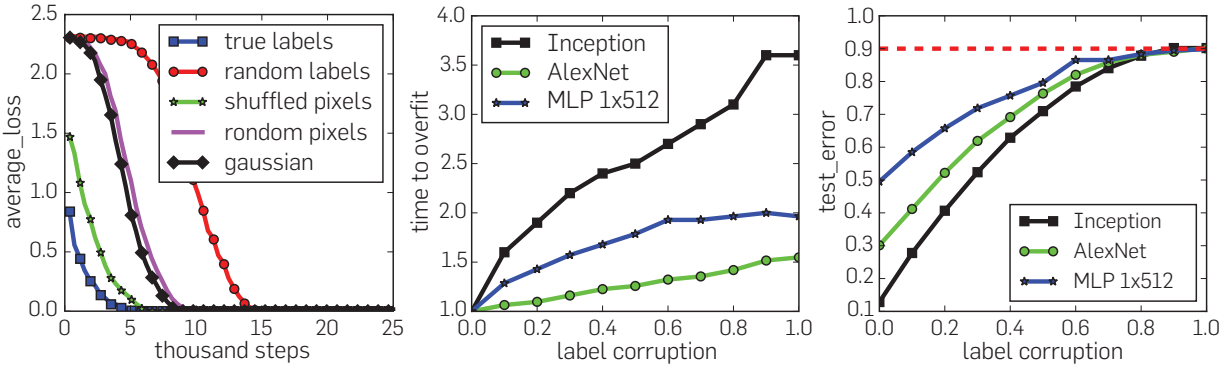
- **True labels**: the original dataset without modification.
- **Partially corrupted labels**: independently with probability p, the label of each image is corrupted as a uniform random class.
- **Random labels**: all the labels are replaced with random ones.
- **Shuffled pixels**: a random permutation of the pixels is chosen and then *the same* permutation is applied to all the images in both training and test set.
- **Random pixels**: a different random permutation is applied to each image independently.
- **Gaussian**: A Gaussian distribution (with matching mean and variance to the original image dataset) is used to generate random pixels for each image.

Surprisingly, stochastic gradient descent with unchanged hyperparameter settings can optimize the weights to fit to random labels perfectly, even though the random labels completely destroy the relationship between images and labels. We further break the structure of the images by shuffling the image pixels, and even completely resampling the random pixels from a Gaussian distribution. But the networks we tested are still able to fit.

Figure 1a shows the learning curves of the Inception model on the CIFAR10 dataset under various settings. We expect the objective function to take longer to start decreasing on random labels because initially the label assignments for every training sample are uncorrelated. Therefore, large prediction errors are backpropagated to make large gradients for parameter updates. However, since the random labels are fixed and consistent across epochs, the network starts fitting after going through the training set multiple times. We find the following observations for fitting random labels very interesting: (a) we do not need to change the learning rate schedule; (b) once the fitting starts, it converges quickly; and (c) it converges to (over)fit the training set perfectly. Also note that "random pixels" and "Gaussian" start converging faster than "random labels." This might be because with random pixels, the inputs are more separated from each other than natural images that originally belong to the same category, therefore, easier to build a network for arbitrary label assignments.

On the CIFAR10 dataset, Alexnet and MLPs all converge to zero loss on the training set. The shaded rows in Table 1 show the exact numbers and experimental setup. We also tested random labels on the ImageNet dataset. As shown

Figure 1. Fitting random labels and random pixels on CIFAR10. (a) The training loss of various experiment settings decaying with the training steps. (b) The relative convergence time with different label corruption ratio. (c) The test error (also the generalization error since training error is 0) under different label corruptions.



(a) Learning curves (b) Convergence slowdown (c) Generalization error growth

Table 1. The training and test accuracy (in %) of various models on the CIFAR10 dataset.

Model	# params	Random crop	Weight decay	Train accuracy	Test accuracy
Inception	1,649,402	Yes	Yes	100.0	89.05
		Yes	No	100.0	89.31
		No	Yes	100.0	86.03
		No	No	100.0	85.75
(fitting random labels)		No	No	100.0	9.78
Inception w/o BatchNorm	1,649,402	No	Yes	100.0	83.00
		No	No	100.0	82.00
(fitting random labels)		No	No	100.0	10.12
Alexnet	1,387,786	Yes	Yes	99.90	81.22
		Yes	No	99.82	79.66
		No	Yes	100.0	77.36
		No	No	100.0	76.07
(fitting random labels)		No	No	99.82	9.86
MLP 3 × 512	1,735,178	No	Yes	100.0	53.35
		No	No	100.0	52.39
(fitting random labels)		No	No	100.0	10.48
MLP 1 × 512	1,209,866	No	Yes	99.80	50.39
		No	No	100.0	50.51
(fitting random labels)		No	No	99.34	10.61

Performance with and without data augmentation and weight decay are compared. The results of fitting random labels are also included.

in the last three rows of Table 2 in Appendix of Zhang et al.,[44] although it does not reach the perfect 100% top-1 accuracy, 95.20% accuracy is still very surprising for 1.2 million random labels from 1000 categories. Note that we did not do any hyperparameter tuning when switching from the true labels to the random labels. It is likely that with some modification of the hyperparameters, perfect accuracy could be achieved on random labels. The network also manages to reach ∼90% top-1 accuracy even with explicit regularizers turned on.

Partially corrupted labels. We further inspect the behavior of neural network training with a varying level of label corruptions from 0 (no corruption) to 1 (complete random labels) on the CIFAR10 dataset. The networks fit the corrupted training set perfectly for all the cases. Figure 1b shows the slowdown of the convergence time with increasing level of label noises. Figure 1c depicts the test errors after convergence. Since the training errors are always zero, the test errors are the same as generalization errors. As the noise level approaches 1, the generalization errors converge to 90%—the performance of random guessing on CIFAR10.

2.2. Implications

In light of our randomization experiments, we discuss how our findings pose a challenge for several traditional approaches for reasoning about generalization.

Rademacher complexity and VC-dimension. Rademacher complexity is commonly used and flexible complexity measure of a hypothesis class. The empirical Rademacher complexity of a function class $\mathcal{F}$ on a dataset $\{z_1, ..., z_n\}$ is defined as

$$\hat{\mathfrak{R}}_n(\mathcal{F}) = \mathbb{E}_\sigma\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(z_i)\right] \quad (1)$$

where $\sigma_1, ..., \sigma_n \in \{\pm 1\}$ are i.i.d. uniform random variables. Usually we aim to bound the Rademacher complexity of the *loss function class* $\mathcal{L} = \{\ell(z = (x, y)) = \ell(h(x), y) : h \in \mathcal{H}\}$, where $z_i = (x_i, y_i)$ are input-output pairs. For L-Lipschitz loss function ℓ and real valued hypothesis class $\mathcal{H}$, $\hat{\mathfrak{R}}_n(\mathcal{L}) \le L\hat{\mathfrak{R}}_n(\mathcal{H})$ by contraction lemma. The Rademacher complexity measures the ability of a function class to fit random ±1 binary

label assignments, which closely resemble our randomization test. Since our empirical results on randomization tests suggest that many neural networks fit the training set with random labels perfectly, we expect that $\hat{\mathfrak{R}}_n(\mathcal{L})$ approximately achieves the maximum for the corresponding loss class $\mathcal{L}$. For example, for the indicator loss, $\hat{\mathfrak{R}}_n(\mathcal{L}) \approx 1$. This is a trivial upper bound on the Rademacher complexity that does not lead to useful generalization bounds in realistic settings. A similar reasoning applies to VC-dimension and its continuous analog fat-shattering dimension, unless we further restrict the network. While Barlett[4] proves a bound on the fat-shattering dimension in terms of ℓ_1 norm bounds on the weights of the network, this bound does not apply to the ReLU networks that we consider here. This result was generalized to other norms by Neyshabur et al.,[31] but even these do not seem to explain the generalization behavior that we observe.

Uniform stability. Stepping away from complexity measures of the hypothesis class, we can instead consider properties of the algorithm used for training. This is commonly done with some notion of stability, such as *uniform stability*. Uniform stability of an algorithm A measures how sensitive the algorithm is to the replacement of a single example. However, it is solely a property of the algorithm, which does not take into account specifics of the data or the distribution of the labels. It is possible to define weaker notions of stability.[27, 32, 36] The weakest stability measure is directly equivalent to bounding generalization error and does take the data into account. However, it has been difficult to utilize this weaker stability notion effectively.

3. THE ROLE OF REGULARIZATION

Most of our randomization tests are performed with explicit regularization turned off. Regularizers are the standard tool in theory and practice to mitigate overfitting in the regime when there are more parameters than data points.[42] The basic idea is that although the original hypothesis is too large to generalize well, regularizers help confine learning to a subset of the hypothesis space with manageable complexity. By adding an explicit regularizer, say by penalizing the norm of the optimal solution, the effective Rademacher complexity of the possible solutions is dramatically reduced.

As we will see, in deep learning, explicit regularization seems to play a rather different role. As the bottom rows of Table 2 in Appendix of Zhang et al.[44] show, even with dropout and weight decay, InceptionV3 is still able to fit the random training set extremely well if not perfectly. Although not shown explicitly, on CIFAR10, both Inception and MLPs still fit perfectly the random training set with weight decay turned on. However, AlexNet with weight decay turned on fails to converge on random labels. To investigate the role of regularization in deep learning, we explicitly compare behavior of deep nets learning with and without regularizers.

Instead of doing a full survey of all kinds of regularization techniques introduced for deep learning, we simply take several commonly used network architectures and compare the behavior when turning off the equipped regularizers. The following regularizers are covered:

- **Data augmentation**: augment the training set via domain-specific transformations. For image data, commonly used transformations include random cropping, random perturbation of brightness, saturation, hue, and contrast.
- **Weight decay**: equivalent to a ℓ_2 regularizer on the weights; also equivalent to a hard constrain of the weights to an Euclidean ball, with the radius decided by the amount of weight decay.
- **Dropout**[39]: mask out each element of a layer output randomly with a given dropout probability. Only the Inception V3 for ImageNet uses dropout in our experiments.

Table 1 shows the results of Inception, Alexnet, and MLPs on CIFAR10, toggling the use of data augmentation and weight decay. Both regularization techniques help to improve the generalization performance, but even with all of the regularizers turned off, all of the models still generalize very well.

Table 2 in Appendix of Zhang et al.[44] shows a similar experiment on the ImageNet dataset. A 18% top-1 accuracy drop is observed when we turn off all the regularizers. Specifically, the top-1 accuracy without regularization is 59.80%, while random guessing only achieves 0.1% top-1 accuracy on ImageNet. More strikingly, with data augmentation on but other explicit regularizers off, Inception is able to achieve a top-1 accuracy of 72.95%. Indeed, it seems like the ability to augment the data using known symmetries is significantly more powerful than just tuning weight decay or preventing low training error.

Inception achieves 80.38% top-5 accuracy without regularization, while the reported number of the winner of ILSVRC 2012 achieved 83.6%. So while regularization is important, bigger gains can be achieved by simply changing the model architecture. It is difficult to say that the regularizers count as a fundamental phase change in the generalization capability of deep nets.

3.1. Implicit regularizations

Early stopping was shown to implicitly regularize on some convex learning problems.[21, 43] In Table 2 in Appendix of Zhang et al.,[44] we show in parentheses the best test accuracy along the training process. It confirms that early stopping could *potentially*[a] improve the generalization performance. Figure 2a shows the training and testing accuracy on ImageNet. The shaded area indicates the accumulative best test accuracy, as a reference of potential performance gain for early stopping. However, on the CIFAR10 dataset, we do not observe any potential benefit of early stopping.

Batch normalization is an operator that normalizes the layer responses within each mini-batch. It has been widely adopted in many modern neural network architectures such as Inception and Residual Networks. Although not explicitly designed for regularization, batch normalization is usually found to improve the generalization performance. The

[a] We say "potentially" because to make this statement rigorous, we need to have another isolated test set and test the performance there when we choose early stopping point on the first test set (acting like a validation set).

Inception architecture uses a lot of batch normalization layers. To test the impact of batch normalization, we create a "Inception w/o BatchNorm" architecture that is exactly the same as Inception, except with all the batch normalization layers removed. Figure 2b compares the learning curves of the two variants of Inception on CIFAR10, with all the explicit regularizers turned off. The normalization operator helps stabilize the learning dynamics, but the impact on the generalization performance is only 3∼4%. The exact accuracy is also listed in the section "Inception w/o BatchNorm" of Table 1.

In summary, our observations on both explicit and implicit regularizers are consistently suggesting that regularizers, when properly tuned, could help to improve the generalization performance. However, it is unlikely that the regularizers are the fundamental reason for generalization, as the networks continue to perform well after all the regularizers removed.

4. FINITE-SAMPLE EXPRESSIVITY

Much effort has gone into characterizing the expressivity of neural networks, for example, Cybenko[12], Mhaskar[25], Delalleau and Bengio[13], Mhaskar and Poggio[24], Eldan and Shamir[15], Telgarsky[40], Cohen and Shashua[11] Almost all of these results are at the "population level" showing what functions of the entire domain can and cannot be represented by certain classes of neural networks with certain number of parameters. For example, it is known that at the population level, depth k is generically more powerful than depth $k-1$.

We argue that what is more relevant in practice is the expressive power of neural networks on a finite sample of size n. It is possible to transfer population level results to finite sample results using uniform convergence theorems. However, such uniform convergence bounds would require the sample size to be polynomially large in the dimension of the input and exponential in the depth of the network, posing a clearly unrealistic requirement in practice.

We instead directly analyze the finite-sample expressivity of neural networks, noting that this dramatically simplifies the picture. Specifically, as soon as the number of parameters p of a networks is greater than n, even simple two-layer neural networks can represent any function of the input sample. We say that a neural network C can represent any function of a sample of size n in d dimensions if for every sample $S \subseteq \mathbb{R}^d$ with $|S| = n$ and every function $f: S \to \mathbb{R}$, there exists a setting of the weights of C such that $C(x) = f(x)$ for every $x \in S$.

THEOREM 1. *There exists a two-layer neural network with ReLU activations and $2n + d$ weights that can represent any function on a sample of size n in d dimensions.*

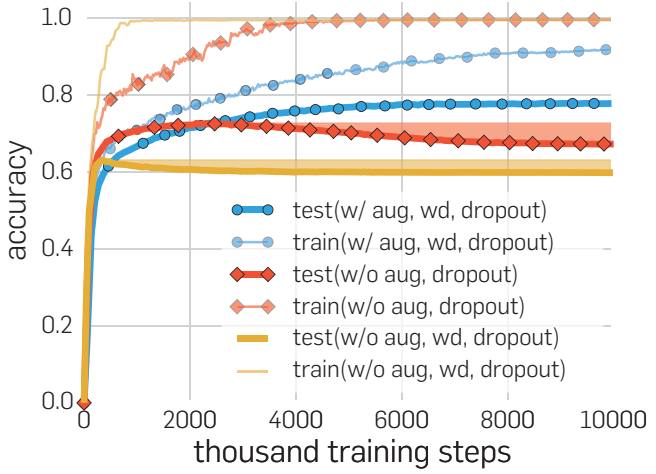
The proof is given in Appendix C of Zhang et al.,[44] where we also discuss how to achieve width $O(n/k)$ with depth k. We remark that it is a simple exercise to give bounds on the weights of the coefficient vectors in our construction. Lemma 1[44] gives a bound on the smallest eigenvalue of the matrix A. This can be used to give reasonable bounds on the weight of the solution w.

5. IMPLICIT REGULARIZATION: AN APPEAL TO LINEAR MODELS

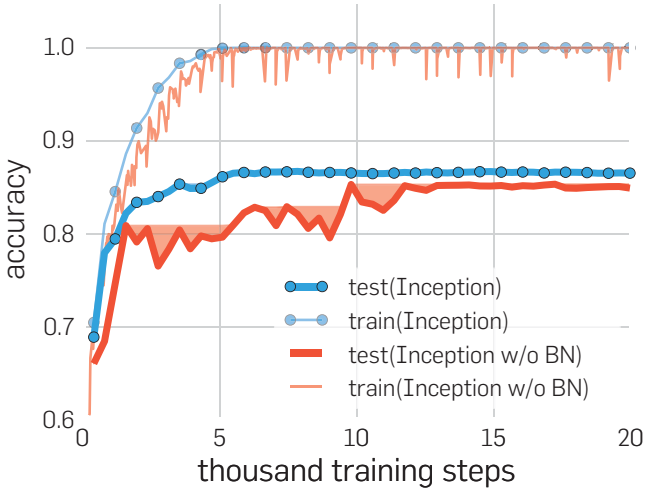
Although deep neural nets remain mysterious for many reasons, we note in this section that it is not necessarily easy to understand the source of generalization for linear models either. Indeed, it is useful to appeal to the simple case of linear models to see if there are parallel insights that can help us better understand neural networks.

Suppose we collect n distinct data points $\{(x_i, y_i)\}$ where x_i

Figure 2. Effects of implicit regularizers on generalization performance. aug is data augmentation; wd is weight decay; BN is batch normalization. The shaded areas are the cumulative best test accuracy, as an indicator of potential performance gain of early stopping. (a) Early stopping could potentially improve generalization when other regularizers are absent. (b) Early stopping is not necessarily helpful on CIFAR10, but batch normalization stabilizes the training process and improves the generalization.



(a) Inception on ImageNet



(b) Inception on CIFAR10

is d-dimensional feature vectors and y_i is labels. Letting loss denote a nonnegative loss function with loss(y, y) = 0, consider the *empirical risk minimization* (ERM) problem

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \text{loss}(w^T x_i, y_i) \tag{2}$$

If $d \geq n$, then we can fit any labeling. But is it then possible to generalize with such a rich model class and no explicit regularization?

Let X denote the $n \times d$ data matrix whose i-th row is x_i^T. If X has rank n, then the system of equations $Xw = y$ has an infinite number of solutions regardless of the right-hand side. We can find a global minimum in the ERM problem (2) by simply solving this linear system.

But do all global minima generalize equally well? Is there a way to determine when one global minimum will generalize whereas another will not? One popular way to understand quality of minima is the curvature of the loss function at the solution. But in the linear case, the curvature of all optimal solutions is the same.[9] To see this, note that in the case when y_i is a scalar,

$$\nabla^2 \frac{1}{n}\sum_{i=1}^{n} \text{loss}(w^T x_i, y_i) = \frac{1}{n} X^T \text{diag}(\beta) X,$$

where $\beta_i = \left.\frac{\partial^2 \text{loss}(z, y_i)}{\partial z^2}\right|_{z=y_i}$. A similar formula can be found when y is vector valued. In particular, the Hessian is not a function of the choice of w. Moreover, the Hessian is degenerate at all global optimal solutions.

If curvature does not distinguish global minima, what does? A promising direction is to consider the workhorse algorithm, stochastic gradient descent (SGD), and inspect which solution SGD converges to. Since the SGD update takes the form $w_{t+1} = w_t - \eta_t e_t x_{i_t}$ where η_i is the step size and e_t is the prediction error loss. If $w_0 = 0$, we must have that the solution has the form $w = \sum_{i=1}^{n} \alpha_i x_i$ for some coefficients α. Hence, if we run SGD we have that $w = X^T\alpha$ lies in the span of the data points. If we also perfectly interpolate the labels, we have $Xw = y$. Enforcing both of these identities, this reduces to the single equation

$$XX^T\alpha = y \tag{3}$$

which has *a unique solution*. Note that this equation only depends on the dot-products between the data points x_i. We have thus derived the "kernel trick"[34]—albeit in a roundabout fashion.

We can therefore perfectly fit any set of labels by forming the Gram matrix (aka the *kernel matrix*) on the data $K = XX^T$ and solving the linear system $K\alpha = y$ for α. This is an $n \times n$ linear system that can be solved on standard workstations whenever n is less than a hundred thousand, as is the case for small benchmarks like CIFAR10 and MNIST.

Quite surprisingly, fitting the training labels exactly yields excellent performance for convex models. On MNIST with no preprocessing, we are able to achieve a test error of 1.2% by simply solving $K\alpha = y$ with a Gaussian kernel on the pixel representation. Note that this is not exactly simple as the kernel matrix requires 30 GB to store in memory. Nonetheless, this system can be solved in under 3 minutes on a commodity workstation with 24 cores and 256 GB of RAM with a conventional LAPACK call. By first applying a Gabor wavelet transform to the data and then solving (3), the error on MNIST drops to 0.6%. Surprisingly, adding regularization does not improve either model's performance!

Similar results follow for CIFAR10. Simply applying a Gaussian kernel on pixels and using no regularization achieves 46% test error. By preprocessing with a random convolutional neural net with 32,000 random filters, this test error drops to 17% error[b]. Adding ℓ_2 regularization further reduces this number to 15% error. Note that this is without any data augmentation.

Note that this kernel solution has an appealing interpretation in terms of implicit regularization. Simple algebra reveals that it is equivalent to the *minimum ℓ_2-norm* solution of $Xw = y$. That is, out of all models that exactly fit the data, SGD will often converge to the solution with minimum norm. It is very easy to construct solutions of $Xw = y$ that do not generalize: for example, one could fit a Gaussian kernel to data and place the centers at random points. Another simple example would be to force the data to fit random labels on the test data. In both cases, the norm of the solution is significantly larger than the minimum norm solution.

Unfortunately, this notion of minimum norm is not predictive of generalization performance. For example, returning to the MNIST example, the ℓ_2-norm of the minimum norm solution with no preprocessing is approximately 220. With wavelet preprocessing, the norm jumps to 390. Yet the test error drops by a factor of 2. So while this minimum-norm intuition may provide some guidance to new algorithm design, it is only a very small piece of the generalization story.

6. CONCLUSION

In this work, we presented a simple experimental framework for interrogating purported measures of generalization. The experiments we conducted emphasize that the *effective capacity* of several successful neural network architectures is large enough to shatter the training data. Consequently, these models are in principle rich enough to memorize the training data. This situation poses a conceptual challenge to statistical learning theory as traditional measures of model complexity struggle to explain the generalization ability of large artificial neural networks. An important insight resulting from our experiments is that optimization continues to be empirically easy even if the resulting model does not generalize. What drives generalization therefore cannot be identical to what makes optimization of deep neural networks easy in practice, another important—yet, as we show, distinct—question.

The situation we find ourselves in bears semblance to where machine learning was in the 1960s. One of the first striking successes of machine learning dates back to Rosenblatt's 1958 discovery of the Perceptron algorithm. In modern language, the Perceptron learns a linear function from labeled examples. Cycling through the data one

[b] This conv-net is the Coates and Ng[10] net, but with the filters selected at random instead of with k-means.

example at a time, whenever the Perceptron encounters an example where the sign of the linear function disagrees with the binary label, it nudges the coefficients of the linear function either toward or away from the example. Analysis from the 1960s provided generalization results for the Perceptron assuming that there was some solution out there that properly labeled all data we might ever see. An instance of the popular stochastic gradient method, the Perceptron, remains strikingly similar to modern machine learning practice. Indeed, the results on linear models in Section 5 are effectively a generalization of the 60-year-old results on the Perceptron.

The primary difference between now and then is one of scale and complexity. In place of a simple linear function, we find intricate models that stack several nonlinear transformations, so-called layers, on top of each other. Each layer has its own set of trainable parameters. Such concatenation adds complexity: we no longer get the beautiful convergence and generalization theorems of the Perceptron. The classic Perceptron theory explained why overparameterized *linear* models might generalize in some special cases, but these results do not provide an explanation of the power of nonlinear models.

6.1. A partial survey of recent progress

The original version of this paper[44] motivated a tremendous amount of new work on generalization that we cannot fully survey here. However, we will attempt to summarize some general trends.

Regarding our observation that conventional generalization bounds based on uniform convergence or uniform stability are inadequate for overparameterized deep neural networks, extensive efforts were made toward tighter generalization bounds (e.g., Kawaguchi et al.,[19] Bartlett et al.,[5] Neyshabur et al.,[28] Golowich et al.,[17] Liang et al.[20]). In the *PAC-Bayes* setting, where the learning algorithm is allowed to output a distribution over parameters, new generalization bounds were also derived.[14, 29, 2, 46]

Aligned with our observation that overparameterized deep networks generalize even without any explicit regularization, and our analysis of implicit regularization in linear models, there is renewed interest in seeking to explain generalization in deep learning by characterizing the implicit regularization induced by the learning algorithms.[37, 38, 35, 1]

In-depth analysis on memorization of overparameterized models also extends our intuition on overfitting from the traditional U-shaped risk curve to the "double descent" *risk curve*. Specifically, in the overparameterized regime where the model capacity greatly exceeds the training set size, fitting all the training examples (i.e., *interpolating* the training set), including noisy ones, is not necessarily at odds with generalization.[23, 7, 6, 16]

Despite significant progress on theoretical understanding of deep learning in the past few years, a full mathematical characterization of the whole story remains challenging. Since the original version of this paper,[44] much more work starts approaching the question of understanding deep learning using *empirical studies*, by designing systematic and principled experiments (e.g., Arpit et al.,[3] Zhao et al.,[45] Morcos et al.,[26] Recht et al.,[33] Toneva et al.[41]). The randomization test proposed in this paper serves as the backbone in the experimental design in many of those studies. Dedicated workshops on phenomena in deep learning are being organized in all major machine learning conferences nowadays. Even some theory conferences start to consider pure empirical studies that reveal "interesting and not well understood behavior"[c] in the call-for-papers. Thus, we are excited to see what happens in the next four years as well as excited to have highlighted some of the development over the past four years since we wrote the original manuscript. C

References

1. Arora, S., Cohen, N., Hu, W., Luo, Y. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, eds. Vol. 32. Curran Associates, Inc., 2019, 7411–7422.
2. Arora, S., Ge, R., Neyshabur, B., Zhang, Y. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*. J. Dy and A. Krause, eds. 2018, 254–263.
3. Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., Lacoste-Julien, S. A closer look at memorization in deep networks. In *International Conference on Machine Learning*. D. Precup and Y.W. Teh, eds. 2017, 233–242.
4. Bartlett, P.L. The sample complexity of pattern classification with neural networks—The size of the weights is more important than the size of the network. *IEEE Trans. Inform. Theory*, 44 (1998), 525–536.
5. Bartlett, P.L., Foster, D.J., Telgarsky, M.J. Spectrally-normalized margin bounds for neural networks. *Adv. Neural Inform. Process. Syst*. 2017, 6240–6249.
6. Belkin, M., Hsu, D., Ma, S., Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci. 32*, 116 (2019), 15849–15854.
7. Belkin, M., Hsu, D.J., Mitra, P. Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate. *Adv. Neural Inform. Process. Syst*., 2018, 2300–2311.
8. Bousquet, O., Elisseeff, A. Stability and generalization. *J. Mach. Learn. Res*. 2 (2002), 499–526.
9. Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*. G. Lebanon and S.V.N. Vishwanathan, eds. 2015, 192–204.
10. Coates, A., Ng, A.Y. Learning feature representations with *k*-means. In *Neural Networks: Tricks of the Trade, Reloaded*. Springer, 2012.
11. Cohen, N., Shashua, A. Convolutional rectifier networks as generalized tensor decompositions. In *International Conference on Machine Learning*. M.F. Balcan and K.Q. Weinberger, eds. 2016, 955–963.
12. Cybenko, G. Approximation by superposition of sigmoidal functions. *Math. Control Signal. Syst. 4*, 2 (1989), 303–314.
13. Delalleau, O., Bengio, Y. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems 24*. J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger, eds. Curran Associates, Inc., 2011, 666–674
14. Dziugaite, G.K., Roy, D.M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*. 2017.
15. Eldan, R., Shamir, O. The power of depth for feedforward neural networks. In *Conference on Learning Theory*. V. Feldman, A. Rakhlin, and O. Shamir, eds. 2016, 907–940.
16. Feldman V. Does learning require memorization? a short tale about a long tail. *arXiv preprint arXiv:1906.05271* (2019).
17. Golowich, N., Rakhlin, A., Shamir, O. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*. P.R. Sébastien Bubeck, V. Perchet, eds. ,2018, 297–299.
18. Hardt, M., Recht, B., Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*. M.F. Balcan and K.Q. Weinberger, eds. 2016, 1225–1234.
19. Kawaguchi, K., Kaelbling, L.P., Bengio, Y. Generalization in deep learning. *CoRR*, arXiv:1710.05468 (2017).
20. Liang, T., Poggio, T., Rakhlin, A., Stokes, J. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*. K. Chaudhuri and M. Sugiyama, eds. arXiv:1711.01530 (2017), 888–896.
21. Lin, J., Camoriano, R., Rosasco, L. Generalization properties and implicit regularization for multiple passes SGM. In *International Conference on Machine Learning*. M.F. Balcan and K.Q. Weinberger, eds. 2016, 2340–2348.
22. Livni, R., Shalev-Shwartz, S., Shamir, O. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems 27*. Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds. Curran Associates, Inc., 2014, 855–863.
23. Ma, S., Bassily, R., Belkin, M. The power of interpolation: Understanding the effectiveness of sgd in modern

[c] Quoted from the call-for-papers of Algorithmic Learning Theory (ALT) 2020.

over-parametrized learning. In *International Conference on Machine Learning*. J. Dy and A. Krause, eds. 2018, 3325–3334.
24. Mhaskar, H. Poggio, T.A. Deep vs. shallow networks: An approximation theory perspective. *Anal. Appl. 6*, 14 (2016).
25. Mhaskar, H.N. Approximation properties of a multilayered feedforward artificial neural network. *Adv. Comput. Math. 1*, 1 (1993), 61–80.
26. Morcos, A., Raghu, M., Bengio, S. Insights on representational similarity in neural networks with canonical correlation. *Adv. Neural Inform. Process. Syst.*. 2018, 5727–5736.
27. Mukherjee, S., Niyogi, P., Poggio, T., Rifkin R. Statistical learning: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Technical Report AI Memo 2002-024*. Massachusetts Institute of Technology, 2002.
28. Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N. Exploring generalization in deep learning. *Adv. Neural Inform. Process. Syst.*, 2017, 5947–5956.
29. Neyshabur, B., Bhojanapalli, S., Srebro, N. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.
30. Neyshabur, B., Tomioka, R., Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. *CoRR*, abs/1412.6614, 2014.
31. Neyshabur, B., Tomioka, R., Srebro, N. Norm-based capacity control in neural networks. In *Conference on Learning Theory*. S.K. Peter Grünwald and E. Hazan, eds. 2015, 1376–1401.
32. Poggio, T., Rifkin, R., Mukherjee, S., Niyogi, P. General conditions for predictivity in learning theory. *Nature 6981*, 428 (2004), 419–422.
33. Recht, B., Roelofs, R., Schmidt, L., Shankar, V. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811* (2019).
34. Schölkopf, B., Herbrich, R., Smola, A.J. A generalized representer theorem. In *Conference on Learning Theory*. 2001, 416–426.
35. Shah, V., Kyrillidis, A., Sanghavi, S. Minimum norm solutions do not always generalize well for over-parameterized problems. *CoRR*. arXiv:1811.07055 (2018).
36. Shalev-Shwartz, S., Shamir, O., Srebro, N., Sridharan, K. Learnability, stability and uniform convergence. *J. Mach. Learn. Res.*, 11 (2010), 2635–2670.
37. Smith, S.L., Le, Q.V. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.
38. Soudry, D., Hoffer, E., Nacson, M.S., Gunasekar, S., Srebro, N. The implicit bias of gradient descent on separable data. *J. Mach. Learn. Res. 70*, 19 (2018), 1–57.
39. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res. 1*, 15 (2014), 1929–1958.
40. Telgarsky, M. Benefits of depth in neural networks. In *Conference on Learning Theory*. V. Feldman, A. Rakhlin, and O. Shamir, eds. 2016, 1517–1539.
41. Toneva, M., Sordoni, A., des Combes, R.T., Trischler, A., Bengio, Y., Gordon, G.J. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.
42. Vapnik, V.N. *Statistical Learning Theory. Adaptive and Learning* Systems for Signal Processing, Communications, and Control. Wiley, 1998.
43. Yao, Y., Rosasco, L., Caponnetto, A. On early stopping in gradient descent learning. *Const. Approx. 2*, 26 (2007), 289–315.
44. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*. 2017.
45. Zhao, S., Ren, H., Yuan, A., Song, J., Goodman, N., Ermon, S. Bias and generalization in deep generative models: An empirical study. In *Advances in Neural Information Processing Systems 31*. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds. Curran Associates, Inc., 2018, 10792–10801.
46. Zhou, W., Veitch, V., Austern, M., Adams, R.P., Orbanz, P. Non-vacuous generalization bounds at the ImageNet scale: A PAC-Bayesian compression approach. In *International Conference on Learning Representations*, 2019.

Chiyuan Zhang and Samy Bengio ({chiyuan, bengio}@google.com), Google Brain, Mountain View, CA, USA.

Moritz Hardt and Benjamin Recht ({hardt, brecht@berkeley.edu}), University of California, Berkeley, CA, USA. Work performed at Google Brain.

Oriol Vinyals (vinyals@google.com), DeepMind, London N1C 4AG, U.K.

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/understanding-deep-learning