Stronger Generalization Bounds for Deep Nets via a Compression Approach

Sanjeev Arora [1] **Rong Ge** [2] **Behnam Neyshabur** [3] **Yi Zhang** [1]

Abstract

Deep nets generalize well despite having more parameters than the number of training samples. Recent works try to give an explanation using PAC-Bayes and Margin-based analyses, but do not as yet result in sample complexity bounds better than naive parameter counting. The current paper shows generalization bounds that are orders of magnitude better in practice. These rely upon new succinct reparametrizations of the trained net — a compression that is explicit and efficient. These yield generalization bounds via a simple compression-based framework introduced here. Our results also provide some theoretical justification for widespread empirical success in compressing deep nets. Analysis of correctness of our compression relies upon some newly identified "noise stability"properties of trained deep nets, which are also experimentally verified. The study of these properties and resulting generalization bounds are also extended to convolutional nets, which had eluded earlier attempts on proving generalization.

1. Introduction

A mystery about deep nets is that they generalize (i.e., predict well on unseen data) despite having far more parameters than the number of training samples. One commonly voiced explanation is that regularization during training –whether implicit via use of SGD (Neyshabur et al., 2015c; Hardt et al., 2016) or explicit via weight decay, dropout (Srivastava et al., 2014), batch normalization (Ioffe and Szegedy, 2015), etc. –reduces the effective capacity of the net. But Zhang et al. (2017) questioned this received wisdom and fueled research in this area by showing experimentally that standard architectures using SGD and regularization can still reach low training error on randomly labeled examples (which clearly won't generalize).

Clearly, deep nets trained on real-life data have some properties that reduce effective capacity, but identifying them has proved difficult —at least in a *quantitative* way that yields sample size upper bounds similar to classical analyses in simpler models such as SVMs (Bartlett and Mendelson, 2002; Evgeniou et al., 2000; Smola et al., 1998) or matrix factorization (Fazel et al., 2001; Srebro et al., 2005).

Qualitatively (Hochreiter and Schmidhuber, 1997; Hinton and Van Camp, 1993) suggested that nets that generalize well are *flat minima* in the optimization landscape of the training loss. Recently Keskar et al. (2016) show using experiments with different batch-sizes that sharp minima do correlate with higher generalization error. A quantitative version of "flatness" was suggested in (Langford and Caruana, 2001): the net's output is stable to *noise* added to the net's trainable parameters. Using PAC-Bayes bound (McAllester, 1998; 1999) this noise stability yielded generalization bounds for fully connected nets of depth 2. The theory has been extended to multilayer fully connected nets (Neyshabur et al., 2017b), although thus far yields sample complexity bounds much worse than naive parameter counting. (Same holds for the earlier Bartlett and Mendelson (2002); Neyshabur et al. (2015b); Bartlett et al. (2017); Neyshabur et al. (2017a); Golowich et al. (2017); see Figure 3). Another notion of noise stability —closely related to dropout and batch normalization—is stability of the output with respect to the noise injected at the nodes of the network, which was recently shown experimentally (Morcos et al., 2018) to improve in tandem with generalization ability during training, and to be absent in nets trained on random data. Chaudhari et al. (2016) suggest adding noise to gradient descent to bias it towards finding flat minima.

While study of generalization may appear a bit academic — held-out data easily establishes generalization in practice— the ultimate hope is that it will help identify simple, measurable and intuitive properties of well-trained deep nets, which in turn may fuel superior architectures and faster training. We hope the detailed study —theoretical and empirical—in the current paper advances this goal.

Authors listed in alphabetical order [1]Princeton University, Computer Science Department [2]Duke University, Computer Science Department [3]Institute for Advanced Study, School of Mathematics. Correspondence to: Rong Ge <rongge@cs.duke.edu>, Behnam Neyshabur <bneyshabur@ias.edu>, Yi Zhang <y.zhang@cs.princeton.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Contributions of this paper.

1. A simple *compression framework* (Section 2) for proving generalization bounds, perhaps a more explicit and intuitive form of the PAC-Bayes work. It also yields elementary short proofs of recent generalization results (Section 2.2).

2. Identifying new form of noise-stability for deep nets: the stability of each layer's computation to noise injected at lower layers. (Earlier papers worked only with stability of the *output* layer.) Figure 1 visualizes the stability of network w.r.t. Gaussian injected noise. Formal statements require a string of other properties (Section 3). All are empirically studied, including their correlation with generalization (Section 6).

3. Using the above properties to derive efficient and *provably correct* algorithms that reduce the effective number of parameters in the nets, yielding generalization bounds that: (a) are better than naive parameter counting (Section 6) (b) depend on simple, intuitive and measurable properties of the network (Section 4) (c) apply also to convolutional nets (Section 5) (d) empirically correlate with generalization (Section 6).

The main idea is to show that noise stability allows individual layers to be compressed via a linear-algebraic procedure Algorithm 1. This results in new error in the output of the layer. This added error is "Gaussian-like" and tends to get attenuated as it propagates to higher layers.

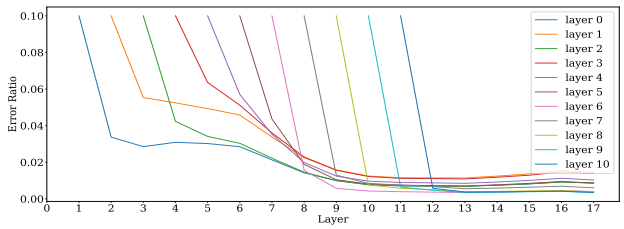

Figure 1. Attenuation of injected noise on a VGG-19 net trained on CIFAR-10. The x-axis is the index of layers and y-axis denote the relative error due to the noise ($\|\hat{x}^i - x^i\|/\|x^i\|$). A curve starts at the layer where a scaled Gaussian noise is injected to its input, whose ℓ_2 norm is set to 10% of the norm of its original input. As it propagates up, the injected noise has rapidly decreasing effect on higher layers. This property is shown to imply compressibility.

Other related works. Dziugaite and Roy (2017) use non-convex optimization to optimize the PAC-Bayes bound and get a non-vacuous sample bound on MNIST. While very creative, this provides little insight into favorable properties of networks. Liang et al. (2017) have suggested Fisher-Rao metric, a regularization based on the Fisher matrix and showed that this metric correlate with generalization. Unfortunately, they could only apply their method to linear networks. Recently Kawaguchi et al. (2017) connects Path-Norm (Neyshabur et al., 2015a) to generalization. However, the proved generalization bound depends on the distribution and measuring it requires vector operations on exponentially high dimensional vectors. Other works have designed experiments to empirically evaluate potential properties of the network that helps generalization(Arpit et al., 2017; Neyshabur et al., 2017b; Dinh et al., 2017). The idea of compressing trained deep nets is very popular for low-power applications; for a survey see Cheng et al. (2018).

Finally, note that the terms *compression* and *stability* are traditionally used in a different sense in generalization theory (Littlestone and Warmuth, 1986; Kearns and Ron, 1999; Shalev-Shwartz et al., 2010). Our framework is compared to other notions in the remarks after Theorem 2.1.

Notation: We use standard formalization of multiclass classification, where data consists of sample x and its *label* y (an integer from 1 to k). A multiclass classifier f maps input x to $f(x) \in \mathbb{R}^k$ and the maximum coordinate of $f(x)$ is the predicted label. The classification loss for any distribution $\mathcal{D}$ is defined as $\mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x)[y] < \max_{i\neq y} f(x)[j]]$ where $f(x)[y]$ is the y-th coordinate of $f(x)$. If $\gamma > 0$ is some desired margin, then the expected margin loss is

$$L_\gamma(f) = \mathbb{P}_{(x,y)\sim\mathcal{D}}\left[f(x)[y] \leq \gamma + \max_{i\neq y} f(x)[j]\right]$$

(Notice, the classification loss corresponds to $\gamma = 0$.) Let $\hat{L}_\gamma$ denote empirical estimate of the margin loss. *Generalization error* is the difference between the two.

For most of the paper we assume that deep nets have fully connected layers, and use ReLU activations. We treat convolutional nets in Section 5. If the net has d layers, we label the vector before activation at these layers by x^0, x^1, x^d for the d layers where x^0 is the input to the net, also denoted simply x. So $x^i = A^i\phi(x^{i-1})$ where A^i is the weight matrix of the ith layer. (Here $\phi(x)$ if x is a vector applies the ReLU component-wise. The ReLU is allowed a trainable bias parameter, which is omitted from the notation because it has no effect on any calculations below.) We denote the number of hidden units in layer i by h^i and set $h = \max_{i=1}^d h^i$. Let $f_A(x)$ be the function calculated by the above network.

Stable rank of a matrix B is $\|B\|_F^2/\|B\|_2^2$, where $\|\cdot\|_F$ denotes Frobenius norm and $\|\cdot\|_2$ denotes spectral norm. Note that stable rank is at most (linear algebraic) rank.

For any two layer $i \leq j$, denote by $M^{i,j}$ the operator for composition of these layers and $J_x^{i,j}$ be the Jacobian of this operator at input x (a matrix whose p, q is the partial derivative of the pth output coordinate with respect to the q'th input input). Therefore, we have $x^j = M^{i,j}(x^i)$. Furthermore, since the activation functions are ReLU, we have

$M^{i,j}(x^i) = J^{i,j}_{x^i} x^i$.

2. Compression and Generalization

Our compression framework rests on the following obvious fact. Suppose the training data contains m samples, and f is a classifier from a complicated class (e.g., deep nets with much more than m parameters) that incurs very low empirical loss. We are trying to understand if it will generalize. Now suppose we can compute a classifier g with discrete trainable parameters much less than m and which incurs similar loss on the training data as f. Then g must incur low classification error on the full distribution. This framework has the advantage of staying with intuitive parameter counting and to avoid explicitly dealing with the hypothesis class that includes f (see note after Theorem 2.1). Notice, the mapping from f to g merely needs to *exist,* not to be efficiently computable. But in all our examples the mapping will be explicit and fairly efficient. Now we formalize the notions. The proofs are elementary via concentration bounds and appear in the appendix.

Definition 1 ((γ,S)-compressible)**.** For any set $\mathcal{A}$ of parameter values, let f be a classifier and $G_{\mathcal{A}} = \{g_A | A \in \mathcal{A}\}$ be a class of classifiers. We say f is (γ, S)-compressible via $G_{\mathcal{A}}$ if there exists $A \in \mathcal{A}$ such that for any $x \in S$, we have for all y

$$|f(x)[y] - g_A(x)[y]| \leq \gamma.$$

We also consider a different setting where the compression algorithm is allowed a"helper string" s, which is arbitrary but fixed before looking at the training samples. Often s will contain random numbers. A simple example is to let s be the random initialization used for training the deep net and then compress the *difference* between the final weights and s. This can give better generalization bounds, similar to (Dziugaite and Roy, 2017). Other nontrivial examples appear later.

Definition 2 ((γ,S)-compressible using helper string s)**.** Suppose $G_{\mathcal{A},s} = \{g_{A,s} | A \in \mathcal{A}\}$ is a class of classifiers indexed by trainable parameters A and fixed strings s. A classifier f is (γ, S)-compressible with respect to $G_{\mathcal{A},s}$ using helper string s if there exists $A \in \mathcal{A}$ such that for any $x \in S$, we have for all y

$$|f(x)[y] - g_{A,s}(x)[y]| \leq \gamma.$$

Theorem 2.1. *Suppose $G_{\mathcal{A},s} = \{g_{A,s} | A \in \mathcal{A}\}$ where A is a set of q parameters each of which can have at most r discrete values and s is a helper string. Let S be a training set with m samples. For any margin $\gamma > 0$, if the trained classifier f is (γ, S)-compressible via $G_{\mathcal{A},s}$ with helper string s, then there exists $A \in \mathcal{A}$ such that with high probability over the training set,*

$$L_0(g_A) \leq \hat{L}_\gamma(f) + O\left(\sqrt{\frac{q \log r}{m}}\right).$$

Remarks: (1) The framework proves the generalization not of f but of its compression g_A. (An exception is if the two are shown to have similar loss at every point in the domain, not just the training set. This is the case in Theorem 2.2.)
(2) The previous item highlights how our framework steps away from uniform convergence framework, e.g., covering number arguments (Dudley, 2010; Anthony and Bartlett, 2009). There, one needs to fix a hypothesis class *independent* of the training set. By contrast we have no hypothesis class, only a *single* neural net that has some specific properties (described in Section 3) on a *single* finite training set. But if we can compress this specific neural net to a simpler neural nets with fewer parameters then we can use covering number argument on this simpler class to get the generalization of the compressed net.
(3) Issue (1) exists also in standard PAC-Bayes framework for deep nets (see tongue-in-cheek title of Langford and Caruana (2001)). They yield generalization bounds not for f but for a noised version of f (i.e., net given by $W + \eta$, where W is parameter vector of f and η is a noise vector).
(4) As we will see later, our compression which is achieved via a randomized algorithm seems "non-destructive" and should not overfit to the training set more than the original network. Moreover, for us issue (1) could be fixed by showing that if f satisfies the properties of Section 3 on training data then it satisfies them on the entire domain. This is left for future work.

2.1. Example 1: Linear classifiers with margin

To illustrate the above compression method and its connection to noise stability, we use linear classifiers with high margins. Let $c \in \mathbb{R}^h (\|c\| = 1)$ be a classifier for binary classification whose output on input x is $\text{sgn}(c \cdot x)$. Let $\mathcal{D}$ be a distribution on inputs (x, y) where $\|x\| = 1$ and $y \in \{\pm 1\}$. Say c has margin γ if for all (x, y) in the training set we have $y(c^\top x) \geq \gamma$.

If we add Gaussian noise vector η with coordinate-wise variance σ^2 to c, then $\mathbb{E}[x \cdot (c + \eta)]$ is $c \cdot x$ and the variance is σ^2. (A similar analysis applies to noising of x instead of c.) Thus the margin is large if and only if the classifier's output is somewhat noise-stable.

A classifier with margin γ can be compressed to one that has only $O(1/\gamma^2)$ non-zero entries. For each coordinate i, toss a coin with $\Pr[heads] = 8c_i^2/\gamma^2$ and if it comes up heads set the coordinate to equal to $\gamma^2/8c_i$ (see Algorithm 2 in supplementary material). This yields a vector $\hat{c}$ with only $O(1/\gamma^2)$ nonzero entries such that for any vector u, with reasonable

probability $|\hat{c}^\top u - c^\top u| \leq \gamma$, so $\hat{c}$ and c will make the same prediction. We can then apply Theorem 2.1 on a discretized version of $\hat{c}$ to show that the sparsified classifier has good generalization with $O(\log d/\gamma^2)$ samples.

This compressed classifier works correctly for a fixed input x with good probability but not high probability. To fix this, one can recourse to the "compression with fixed string" model. The fixed string is a random linear transformation. When applied to unit vector x, it tends to equalize all coordinates and the guarantee $|\hat{c}^\top u - c^\top u| \leq \gamma$ can hold with high probability. This random linear transformation can be fixed before seeing the training data. See Section A.2 in supplementary material for details.

2.2. Example 2: Existing generalization bounds

Our compression framework gives easy and short proof of the generalization bounds of a recent paper; see appendix for slightly stronger result of Bartlett et al. (2017).

Theorem 2.2. *((Neyshabur et al., 2017a)) For any deep net with layers $A^1, A^2, \ldots A^d$ and output margin γ on a training set S, the generalization error can be bounded by*

$$\tilde{O}\left(\sqrt{\frac{hd^2 \max_{x\in S} \|x\| \prod_{i=1}^d \|A^i\|_2^2 \sum_{i=1}^d \frac{\|A^i\|_F^2}{\|A^i\|_2^2}}{\gamma^2 m}}\right).$$

The second part of this expression ($\sum_{i=1}^d \frac{\|A^i\|_F^2}{\|A^i\|_2^2}$) is sum of stable ranks of the layers, a natural measure of their true parameter count. The first part ($\prod_{i=1}^d \|A^i\|_2^2$) is related to the Lipschitz constant of the network, namely, the maximum norm of the vector it can produce if the input is a unit vector. The Lipschitz constant of a matrix operator B is just its spectral norm $\|B\|_2$. Since the network applies a sequence of matrix operations interspersed with ReLU, and ReLU is 1-Lipschitz we conclude that the Lipschitz constant of the full network is at most $\prod_{i=1}^d \|A^i\|_2$.

To prove Theorem 2.2 we use the following lemma to compress the matrix at each layer to a matrix of smaller rank. Since a matrix of rank r can be expressed as the product of two matrices of inner dimension r, it has $2hr$ parameters (instead of the trivial h^2). (Furthermore, the parameters can be discretized via trivial rounding to get a compression with discrete parameters as needed by Definition 1.)

Lemma 1. *For any matrix $A \in \mathbb{R}^{m\times n}$, let $\hat{A}$ be the truncated version of A where singular values that are smaller than $\delta\|A\|_2$ are removed. Then $\|\hat{A} - A\|_2 \leq \delta\|A\|_2$ and $\hat{A}$ has rank at most $\|A\|_F^2/(\delta^2\|A\|_2^2)$.*

Proof. Let r be the rank of $\hat{A}$. By construction, the maximum singular value of $\hat{A} - A$ is at most $\delta\|A\|_2$. Since the remaining singular values are at least $\delta\|A\|_2$, we have $\|A\|_F \geq \|\hat{A}\|_F \geq \sqrt{r}\delta\|A\|_2$. □

For each i replace layer i by its compression using the above lemma, with $\delta = \gamma(3\|x\|d\prod_{i=1}^d \|A^i\|_2)^{-1}$. How much error does this introduce at each layer and how much does it affect the output after passing through the intermediate layers (and getting magnified by their Lipschitz constants)? Since $A - \hat{A}^i$ has spectral norm (i.e., Lipschitz constant) at most δ, the error at the output due to changing layer i in isolation is at most $\delta\|x^i\|\prod_{j=1}^d \|A^j\|_2 \leq \gamma/3d$.

A simple induction (see (Neyshabur et al., 2017a) if needed) can now show the total error incurred in all layers is bounded by γ. The generalization bound follows immediately from Theorem 2.1.

3. Noise Stability Properties of Deep Nets

This section introduces noise stability properties of deep nets that imply better compression (and hence generalization). They help overcome the pessimistic error analysis of our proof of Theorem 2.2: when a layer was compressed, the resulting error was assumed to blow up in a worst-case manner according to the Lipschitz constant (namely, product of spectral norms of layers). This hurt the amount of compression achievable. The new noise stability properties roughly amount to saying that noise injected at a layer has very little effect on the higher layers. Our formalization starts with noise sensitivity, which captures how an operator transmits noise vs signal.

Definition 3. If M is a mapping from real-valued vectors to real-valued vectors, and $\mathcal{N}$ is some noise distribution then *noise sensitivity of M at x with respect to $\mathcal{N}$*, is

$$\psi_{\mathcal{N}}(M, x) = \mathbb{E}_{\eta\in\mathcal{N}}\left[\frac{\|M(x + \eta\|x\|) - M(x)\|^2}{\|M(x)\|^2}\right],$$

The *noise sensitivity of M with respect to $\mathcal{N}$ on a set of inputs S*, denoted $\psi_{\mathcal{N},S}(M)$, is the maximum of $\psi_{\mathcal{N}}(M, x)$ over all inputs x in S.

To illustrate, we examine noise sensitivity of a matrix (i.e., linear mapping) with respect to Gaussian distribution. Low sensitivity turns out to imply that the matrix has some large singular values (i.e., low stable rank), which give directions that can preferentially carry the "signal"x whereas noise η attenuates because it distributes uniformly across directions.

Proposition 3.1. *The noise sensitivity of a matrix M at any vector $x \neq 0$ with respect to Gaussian distribution $\mathcal{N}(0, I)$ is exactly $\|M\|_F^2\|x\|^2/\|Mx\|^2$, and at least its stable rank.*

Proof. Using $\mathbb{E}[\eta\eta^\top] = I$, we bound the numerator by

$$\mathbb{E}_\eta[\|M(x+\eta\|x\|) - Mx\|^2] = \mathbb{E}_\eta[\|x\|^2\|M\eta\|^2]$$
$$= \mathbb{E}_\eta[\|x\|^2 \mathrm{tr}(M\eta\eta^\top M^\top)] = \|x\|^2 \mathrm{tr}(MM^\top) = \|M\|_F^2\|x\|^2.$$

Thus noise sensitivity ψ at x is $\|M\|_F^2\|x\|^2/\|Mx\|^2$, which is at least the stable rank $\|M\|_F^2/\|M\|_2^2$ since $\|Mx\| \leq \|M\|_2\|x\|$. □

The above proposition suggests that if a vector x is aligned to a matrix M (i.e. correlated with high singular directions of M), then matrix M becomes less sensitive to noise at x. This intuition will be helpful in understanding the properties we define later to formalize noise stability.

The above discussion motivates the following approach. We compress each layer i by an appropriate randomized compression algorithm, such that the noise/error in its output is "Gaussian-like". If layers $i+1$ and higher have low sensitivity to this new noise, then the compression can be more extreme produce much higher noise. We formalize this idea using Jacobian $J^{i,j}$, which describes instantaneous change of $M^{i,j}(x)$ under infinitesimal perturbation of x.

3.1. Formalizing Error-resilience

Now we formalize the error-resilience properties. Section 6 reports empirical findings about these properties. The first is *cushion*, to be thought of roughly as reciprocal of noise sensitivity. We first formalize it for single layer.

Definition 4 (layer cushion)**.** The *layer cushion* of layer i is similarly defined to be the largest number μ_i such that for any $x \in S$, $\mu_i\|A^i\|_F\|\phi(x^{i-1})\| \leq \|A^i\phi(x^{i-1})\|$.

Intuitively, cushion considers how much smaller the output $A^i\phi(x^{i-1})$ is compared to the upper bound $\|A^i\|_F\|\phi(x^{i-1})\|$. Using argument similar to Proposition 3.1, we can see that $1/\mu_i^2$ is equal to the noise sensitivity of matrix A^i at input $\phi(x^{i-1})$ with respect to Gaussian noise $\eta \sim \mathcal{N}(0, I)$.

Of course, for nonlinear operators the definition of error resilience is less clean. Let's denote by $M^{i,j}: \mathbb{R}^{h^i} \to \mathbb{R}^{h^j}$ the operator corresponding to the portion of the deep net from layer i to layer j, and by $J^{i,j}$ its Jacobian. If infinitesimal noise is injected before level i then $M^{i,j}$ passes it like $J^{i,j}$, a linear operator. When the noise is small but not infinitesimal then one hopes that $M^{i,j}$ still behaves roughly linearly (recall that ReLU nets are piecewise linear). To formalize this, we define *Interlayer Cushion* (Definition 5) that captures the local linear approximation of the operator M.

Definition 5 (Interlayer Cushion)**.** For any two layers $i \leq j$, we define the *interlayer cushion* $\mu_{i,j}$ as the largest number such that for any $x \in S$:

$$\mu_{i,j}\|J_{x^i}^{i,j}\|_F\|x^i\| \leq \|J_{x^i}^{i,j}x^i\|$$

Furthermore, for any layer i we define the *minimal interlayer cushion* as $\mu_{i\to} = \min_{i\leq j\leq d}\mu_{i,j} = \min\{1/\sqrt{h^i}, \min_{i<j\leq d}\mu_{i,j}\}$[1].

Since $J_x^{i,j}$ is a linear transformation, a calculation similar to Proposition 3.1 shows that its noise sensitivity at x^i with respect to Gaussian distribution $\mathcal{N}(0, I)$ is at most $\frac{1}{\mu_{ij}^2}$.

The next property quantifies the intuitive observation on the learned networks that for any training data, almost half of the ReLU activations at each layer are active. If the input to the activations is well-distributed and the activations do not correlate with the magnitude of the input, then one would expect that on average, the effect of applying activations at any layer is to decrease the norm of the pre-activation vector by at most some small constant factor.

Definition 6 (Activation Contraction)**.** The activation contraction c is defined as the smallest number such that for any layer i and any $x \in S$,

$$\|\phi(x^i)\| \geq \|x^i\|/c.$$

We discussed how the interlayer cushion captures noise-resilience of the network if behaves linearly, namely, when the set of activated ReLU gates does not change upon injecting noise. In general the activations do change, but the deviation from linear behavior is bounded for small noise vectors, as quantified next.

Definition 7 (Interlayer Smoothness)**.** Let η be the noise generated as a result of substituting weights in some of the layers before layer i using Algorithm 1. We define interlayer smoothness ρ_δ to be the largest number such that with probability $1-\delta$ over noise η for any two layers $i < j$ any $x \in S$:

$$\|M^{i,j}(x^i+\eta) - J_{x^i}^{i,j}(x^i+\eta)\| \leq \frac{\|\eta\|\|x^j\|}{\rho_\delta\|x^i\|}.$$

For a single layer, ρ_δ captures the ratio of input/weight alignment to noise/weight alignment. Since the noise behaves similar to Gaussian, one expects this number to be greater than one for a single layer. When $j > i+1$, the weights and activations create more dependencies. However, since these dependences are applied on both noise and input, we again expect that if the input is more aligned to the weights than noise, this should not change in higher layers. In Section 6, we show that the interlayer smoothness is indeed good: $1/\rho_\delta$ is a small constant. Please see Appendix A.4 for a more detailed discussion on interlayer smoothness.

4. Fully Connected Networks

We prove generalization bounds using for fully connected multilayer nets. Details appear in Appendix Section B.

[1]Note that $J_{x^i}^{i,i} = I$ and $\mu_{i,i} = 1/\sqrt{h^i}$

Theorem 4.1. *For any fully connected network f_A with $\rho_\delta \geq 3d$, any probability $0 < \delta \leq 1$ and any margin γ, Algorithm 1 generates weights $\tilde{A}$ for the network $f_{\tilde{A}}$ such that with probability $1-\delta$ over the training set and $f_{\tilde{A}}$, the expected error $L_0(f_{\tilde{A}})$ is bounded by*

$$\hat{L}_\gamma(f_A) + \tilde{O}\left(\sqrt{\frac{c^2 d^2 \max_{x\in S} \|f_A(x)\|_2^2 \sum_{i=1}^d \frac{1}{\mu_i^2 \mu_{i\to}^2}}{\gamma^2 m}}\right)$$

where μ_i, $\mu_{i\to}$, c and ρ_δ are layer cushion, interlayer cushion, activation contraction and interlayer smoothness defined in Definitions 4,5,6 and 7 respectively.

To prove this we describe a compression of the net with respect to a fixed (random) string. In contrast to the deterministic compression of Lemma 1, this randomized compression ensures that the resulting error in the output behaves like a Gaussian. The proofs are similar to standard JL dimension reduction.

Algorithm 1 Matrix-Project (A, ε, η)

Require: Layer matrix $A \in \mathbb{R}^{h_1 \times h_2}$, error parameter ε, η.
Ensure: Returns $\hat{A}$ s.t. $\forall$ fixed vectors u, v,

$$\Pr[|u^\top \hat{A} v - u^\top A v\| \geq \varepsilon \|A\|_F \|u\| \|v\|] \leq \eta.$$

Sample $k = \log(1/\eta)/\varepsilon^2$ random matrices $M_1, \ldots, M_k$ with entries i.i.d. ± 1 ("helper string")
for $k' = 1$ to k **do**
  Let $Z_{k'} = \langle A, M_{k'} \rangle M_{k'}$.
end for
Let $\hat{A} = \frac{1}{k} \sum_{k'=1}^k Z_{k'}$

Note that the helper string of random matrices M_i's were chosen and fixed before training set S was picked. Each weight matrix is thus represented as only k real numbers $\langle A, M_i \rangle$ for $i = 1, 2, ..., k$.

Lemma 2. *For any $0 < \delta, \varepsilon \leq 1$, let $G = \{(U^i, x^i)\}_{i=1}^m$ be a set of matrix/vector pairs of size m where $U \in \mathbb{R}^{n \times h_1}$ and $x \in \mathbb{R}^{h_2}$, let $\hat{A} \in \mathbb{R}^{h_1 \times h_2}$ be the output of Algorithm 1 with $\eta = \delta/mn$ and $\Delta = \hat{A} - A$. With probability at least $1-\delta$ we have for any $(U, x) \in G$, $\|U\Delta x\| \leq \varepsilon \|A\|_F \|U\|_F \|x\|$.*

Next Lemma bounds the number of parameters of the compressed network resulting from applying Algorithm 1 to all the layer matrices of the net. The proof does induction on the layers and bounds the effect of the error on the output of the network using properties defined in Section 3.1.

Lemma 3. *For any fully connected network f_A with $\rho_\delta \geq 3d$, any probability $0 < \delta \leq 1$ and any error $0 < \varepsilon \leq 1$, Algorithm 1 generates weights $\tilde{A}$ for a network with $\frac{72c^2d^2\log(mdh/\delta)}{\varepsilon^2} \cdot \sum_{i=1}^d \frac{1}{\mu_i^2 \mu_{i\to}^2}$ total parameters such that with probability $1 - \delta/2$ over the generated weights $\tilde{A}$, for any $x \in S$:*

$$\|f_A(x) - f_{\tilde{A}}(x)\| \leq \varepsilon \|f_A(x)\|.$$

where μ_i, $\mu_{i\to}$, c and ρ_δ are layer cushion, interlayer cushion, activation contraction and interlayer smoothness defined in Definitions 4,5,6 and 7 respectively.

Some obvious improvements: (i) Empirically it has been observed that deep net training introduces fairly small changes to parameters as compared to the (random) initial weights (Dziugaite and Roy, 2017). We can exploit this by incorporating the random initial weights into the helper string and do the entire proof above not with the layer matrices A^i but only the difference from the initial starting point. Experiments in Section 6 show this improves the bounds. (ii) Cushions and other quantities defined earlier are data-dependent, and required to hold for the *entire* training set. However, the proofs go through if we remove say ζ fraction of outliers that violate the definitions; this allows us to use more favorable values for cushion etc. and lose an additive factor ζ in the generalization error.

5. Convolutional Neural Networks

Now we sketch how to provably compress convolutional nets. (Details appear in Section C of supplementary.) Intuitively, this feels harder because the weights are already compressed— they're *shared* across patches!

Theorem 5.1. *For any convolutional neural network f_A with $\rho_\delta \geq 3d$, any probability $0 < \delta \leq 1$ and any margin γ, Algorithm 4 generates weights $\tilde{A}$ for the network $f_{\tilde{A}}$ such that with probability $1-\delta$ over the training set and $f_{\tilde{A}}$:*

$$L_0(f_{\tilde{A}}) \leq \hat{L}_\gamma(f_A) + \tilde{O}\left(\sqrt{\frac{c^2 d^2 \max_{x\in S} \|f_A(x)\|_2^2 \sum_{i=1}^d \frac{\beta^2(\lceil \kappa_i/s_i \rceil)^2}{\mu_i^2 \mu_{i\to}^2}}{\gamma^2 m}}\right)$$

where μ_i, $\mu_{i\to}$, c, ρ_δ and β are layer cushion, interlayer cushion, activation contraction, interlayer smoothness and well-distributed Jacobian defined in Definitions 4,8,6, 7 and 9 respectively. Furthermore, s_i and κ_i are stride and filter width in layer i.

Let's realize that obvious extensions of earlier sections fail. Suppose layer i of the neural network is an image of dimension $n_1^i \times n_2^i$ and each pixel has h^i channels, the size of the filter at layer i is $\kappa_i \times \kappa_i$ with stride s_i. The convolutional filter has dimension $h^{i-1} \times h^i \times \kappa_i \times \kappa_i$. Applying matrix compression (Algorithm 1) independently to *each copy* of a convolutional filter makes number of new parameters proportional to $n_1^i n_2^i$, a big blowup.

Compressing a convolutional filter once and reusing it in all patches doesn't work because the interlayer analysis implicitly requires the noise generated by the compression behave similar to a spherical Gaussian, but the shared filters introduce correlations. Quantitatively, using the fully connected analysis would require the error to be less than interlayer cushion value $\mu_{i\to}$ (Definition 5) which is at most $1/\sqrt{h^i n_1^i n_2^i}$, and this can never be achieved from compressing matrices that are far smaller than $n_1^i \times n_2^i$ to begin with.

We end up with a solution in between fully independent and fully dependent: *p-wise* independence. The algorithm generates p-wise independent compressed filters $\hat{A}_{(a,b)}$ for each convolution location $(a,b) \in [n_1^i] \times [n_2^i]$. It results in p times more parameters than a single compression. If p grows logarithmically with relevant parameters, the filters behave like fully independent filters. Using this idea we can generalize the definition of interlayer margin to the convolution setting:

Definition 8 (Interlayer Cushion, Convolution Setting)**.** For any two layers $i \le j$, we define the *interlayer cushion* $\mu_{i,j}$ as the largest number such that for any $x \in S$:

$$\mu_{i,j} \cdot \frac{1}{\sqrt{n_1^i n_2^i}} \|J_{x^i}^{i,j}\|_F \|x^i\| \le \|J_{x^i}^{i,j} x^i\|$$

Furthermore, for any layer i we define the *minimal interlayer cushion* as $\mu_{i\to} = \min_{i\le j\le d} \mu_{i,j} = \min\{1/\sqrt{h^i}, \min_{i<j\le d} \mu_{i,j}\}$[2].

Recall that interlayer cushion is related to the noise sensitivity of $J_{x^i}^{i,j}$ at x^i with respect to Gaussian distribution $\mathcal{N}(0, I)$. When we consider $J_{x^i}^{i,j}$ applied to a noise η, if different pixels in η are independent Gaussian random variables, then we can indeed expect $\|J_{x^i}^{i,j}\eta\| \approx \frac{1}{\sqrt{h^i n_1^i n_2^i}} \|J_{x^i}^{i,j}\| \|\eta\|$, which explains the extra $\frac{1}{\sqrt{n_1^i n_2^i}}$ factor in Definition 8 compared to Definition 5. The proof also needs to assume —in line with intuition behind convolution architecture— that information from the entire image field is incorporated somewhat uniformly across pixels. It is formalized using the Jacobian which gives the partial derivative of the output with respect to pixels at previous layer.

Definition 9 (Well-distributed Jacobian)**.** Let $J_x^{i,j}$ be the Jacobian of $M^{i,j}$ at x, we know $J_x^{i,j} \in \mathbb{R}^{h^i \times n_1^i \times n_2^i \times h^j \times n_1^j \times n_2^j}$. We say the Jacobian is β well-distributed if for any $x \in S$, any i, j, any $(a,b) \in [n_1^i \times n_2^i]$,

$$\|[J_x^{i,j}]_{:,a,b,:,:,:}\|_F \le \frac{\beta}{\sqrt{n_1^i n_2^i}} \|J_x^{i,j}\|_F$$

6. Empirical Evaluation

We study noise stability properties (defined in Section 3) of an actual trained deep net, and compute a generalization bound from Theorem 5.1. Experiments were performed by training a VGG-19 architecture (Simonyan and Zisserman, 2014) and a AlexNet (Krizhevsky et al., 2012) for multi-class classification task on CIFAR-10 dataset. Optimization used SGD with mini-batch size 128, weight decay 5e-4, momentum 0.9 and initial learning rate 0.05, but decayed by factor 2 every 30 epochs. Drop-out was used in fully-connected layers. We trained both networks for 299 epochs and the final VGG-19 network achieved 100% training and 92.45% validation accuracy while the AlexNet achieved 100% training and 77.22% validation accuracy. To investigate the effect of corrupted label, we trained another AlexNet, which 100% training and 9.86% validation accuracy, on CIFAR-10 dataset with randomly shuffled labels.

Our estimate of the sample complexity bound used exact computation of norms of weight matrices (or tensors) in all bounds($||A||_{1,\infty}, ||A||_{1,2}, ||A||_2, ||A||_F$). Like previous bounds in generalization theory, ours also depend upon nuisance parameters like depth d, logarithm of h, etc. which probably are an artifact of the proof. These are ignored in the computation (also in computing earlier bounds) for simplicity. Even the generalization based on parameter counting arguments does have an extra dependence on depth (Bartlett et al., 2017). A recent work, (Golowich et al., 2017) showed that many such depth dependencies can be improved.

6.1. Empirical investigation of noise stability properties

Section 3 identifies four properties in the networks that contribute to noise-stability: layer cushion, interlayer cushion, contraction, interlayer smoothness. Figure 2 plots the distribution of over different data points in the training set and compares to a Gaussian random network and then scaled properly. The layer cushion, which quantifies its noise stability, is drastically improved during the training, especially for the higher layers (8 and higher) where most parameters live. Moreover, we observe that interlayer cushion, activation contraction and interlayer smoothness behave nicely even after training. These plots suggest that the driver of the generalization phenomenon is layer cushion. The other properties are being maintained in the network and prevent the network from falling prey to pessimistic assumptions that causes the other older generalization bounds to be very high. The assumptions made in section 3 (also in B.1) are verified on the VGG-19 net in appendix D.1 by histogramming the distribution of layer cushion, interlayer cushion, contraction, interlayer smoothness, and well-distributedness of the Jacobians of each layer of the net on each data point in the training set. Some examples are shown in Figure 2.

6.2. Correlation to generalization error

We evaluate our generalization bound during the training, see Figure 3, Right. After 120 epochs, the training error is

[2]Note that $J_{x^i}^{i,i} = I$ and $\mu_{i,i} = 1/\sqrt{h^i}$

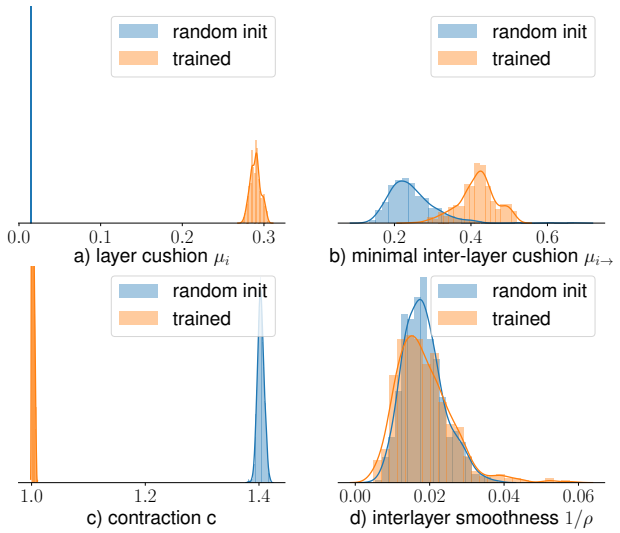

Figure 2. Distribution of a) layer cushion, b) (unclipped) minimal interlayer cushion, c) activation contraction and d) interlayer smoothness of the 13-th layer of VGG-19 nets on on training set. The distributions on a randomly-initialized and a trained net are shown in blue and orange. Note that after clipping, the minimal interlayer cushion is set to $1/\sqrt{h_i}$ for all layers except the first one, see appendix D.1.

almost zero but the test error continues to improve in later epochs. Our generalization bound continues to improve, though not to the same level. Thus our generalization bound captures part of generalization phenomenon, not all. Still, this suggests that SGD somehow improves our generalization measure implicitly. Making this rigorous is a good topic for further research.

Furthermore, we investigate effect of training with normal data and corrupted data by training two AlexNets respectively on original and corrupted CIFAR-10 with randomly shuffled labels. We identify two key properties that differ significantly between the two networks: layer cushion and activation contraction, see D.2. Since our bound predicts larger cushion and lower contraction indicates better generalization, our bound is consistent w with the fact that the net trained on normal data generalizes (77.22% validation accuracy).

6.3. Comparison to other generalization bounds

Figure 3 compares our proposed bound to other neural-net generalization bounds on the VGG-19 net and compares to naive VC dimension bound (which of course is too pessimistic). All previous generalization bounds are orders of magnitude worse than ours; the closest one is spectral norms times average $\ell_{1,2}$ of the layers (Bartlett et al., 2017) which is still about 10^{18}, far greater than VC dimension. (As mentioned we're ignoring nuisance factors like depth

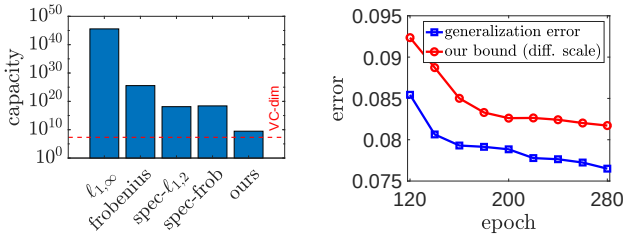

Figure 3. **Left)** Comparing neural net generalization bounds. See Appendix D.3 for details. **Right)** Comparing our bound to empirical generalization error during training. Our bound is rescaled to be within the same range as the generalization error.

and $\log h$ which make the comparison to VC dimension a bit unfair, but the comparison to previous bounds is fair.) This should not be surprising as all other bounds are based on product of norms is pessimistic (see note at the start of Section 3) which we avoid due to the noise stability analysis resulting in a bound that has more dependence on the data.

Table 1 shows the compressibility of various layers according to the bounds given by our theorem. Again, this is a qualitative due to ignoring nuisance factors, but it gives an idea of which layers are important in the calculation.

layer	$\frac{c_i^2\beta_i^2\lceil\kappa_i/s_i\rceil^2}{\mu_i^2\mu_{i\to}^2}$	actual # param	compression (%)
1	1644.87	1728	95.18
4	644654.14	147456	437.18
6	3457882.42	589824	586.25
9	36920.60	1179648	3.129
12	22735.09	2359296	0.963
15	26583.81	2359296	1.126
18	5052.15	262144	1.927

Table 1. Effective number of parameters identified by our bound. Compression rates can be as low as 1% in later layers (from 9 to 19) whereas earlier layers are not so compressible. Dependence on depth d, log factors, constants are ignored as mentioned in the text.

7. Conclusions

With a new compression-based approach, the paper has made progress on several open issues regarding generalization properties of deep nets. The approach also adapts specially to convolutional nets. The empirical verification of the theory in Section 6 shows a rich set of new properties satisfied by deep nets trained on realistic data, which we hope will fuel further theory work on deep learning, including how these properties play into optimization and expressivity. Another possibility is a more rigorous understanding of deep net compression, which sees copious empirical work motivated by low-power applications. Perhaps our p-wise independence idea used for compressing convnets (Section 5) has practical implications.

Acknowledgments

This research was done with support from NSF, ONR, Darpa, SRC, Simons Foundation, Mozilla Research, and Schmidt Foundation.

References

Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

Peter Bartlett, Dylan J Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1706.08498*, 2017.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

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

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Proc. Magazine*, 35, Jan 2018.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

Richard M Dudley. Universal donsker classes and metric entropy. In *Selected Works of RM Dudley*, pages 345–365. Springer, 2010.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1, 2000.

Maryam Fazel, Haitham Hindi, and Stephen P Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4734–4739. IEEE, 2001.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*, 2017.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *ICML*, 2016.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

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

Michael Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural computation*, 11(6):1427–1453, 1999.

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

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

John Langford and Rich Caruana. (not) bounding the true error. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, pages 809–816. MIT Press, 2001.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. *arXiv preprint arXiv:1711.01530*, 2017.

Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. Technical report, Technical report, University of California, Santa Cruz, 1986.

David A McAllester. Some PAC-Bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234. ACM, 1998.

David A McAllester. PAC-Bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 164–170. ACM, 1999.

Ari Morcos, David GT Barrett, Matthew Botvinick, and Neil Rabinowitz. On the importance of single directions for generalization. In *Proceeding of the International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1iuQjxCZ&noteId=r1iuQjxCZ.

Behnam Neyshabur, Ruslan R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2422–2430, 2015a.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Proceeding of the 28th Conference on Learning Theory (COLT)*, 2015b.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *Proceeding of the International Conference on Learning Representations workshop track*, 2015c.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017a.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5949–5958, 2017b.

Christos Pelekis and Jan Ramon. Hoeffding's inequality for sums of weakly dependent random variables. *arXiv preprint arXiv:1507.06871*, 2015.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11 (Oct):2635–2670, 2010.

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

Alex J Smola, Bernhard Schölkopf, and Klaus-Robert Müller. The connection between regularization operators and support vector kernels. *Neural networks*, 11(4): 637–649, 1998.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2005.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

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