Compression-Based Generalization Bounds

Eric K. Zhang
ekzhang@college.harvard.edu

May 14, 2020

Abstract

Here, we describe *compression* as a framework for bounding the generalization error of real-world deep neural networks. This is an emerging framework used to study generalization properties, somewhat related to practical algorithms for deep compression such as [5], which perform well empirically. Unlike many previous uniform convergence bounds, compression does not bound the error of the model itself, but it rather bounds the error of a restricted representation of a model with lower complexity. Although this is less direct, it can somewhat circumvent the fundamental problems with uniform convergence bounds shown in [11]. Results in [1] indicate that the compression framework can reproduce bounds from previous papers and even achieves almost non-vacuous bounds on real-world deep neural networks. We look at compression in the context of other approaches in [13] and [9], and we suggest possible future directions for studying the empirical performance of these methods.

1 Introduction

Empirical measurements of training and testing error show that deep neural networks often generalize well to held-out data, even when the number of parameters exceeds the size of the training set—the regime where we would usually expect overfitting [15]. Even though large deep nets have sufficient expressive power to learn randomly labeled samples [19], illustrating high Rademacher complexity, they still do not overfit on real-world data. This has puzzled traditional approaches to bounding generalization error.

Recent results have tried to explain this by applying different complexity measures to the problem [12, 10]. Some theoretical tools that have been used include PAC-Bayesian analysis [14], margin-based methods [9], and information theory [18]. One particularly promising method for explaining the generalization of deep neural networks and other models is that of *compression*. A framework was introduced in [1] to bound the generalization error of compressed representations, and [16] has recently extended this to bounds on the original model.

Although compression has a long history (see examples in [4, 5]), this past research was mostly empirical. The focus was primarily on reducing the *size* of neural networks for use on mobile devices and embedded systems, as well as their compute *performance*. For example, some classic methods (see [3]) involve re-training a smaller neural network based on knowledge distilled knowledge from a larger network. In contrast, if we wish to obtain theoretical generalization bounds, we must use simpler techniques that can be mathematically analyzed.

Consider the following thought experiment: given a set of data with 10000 labeled examples drawn from some distribution over a very large input space, what would be the optimal size of neural network that minimize generalization error on the underlying distribution? We would expect that the number of trainable parameters of the neural network should be smaller than the size of the training data, as otherwise, there would be more output hypotheses than arbitrary assignments of an 10000-bit string of labels to the training examples. However, it turns out that this intuition is incorrect in practice. Empirical results show that the number of *effective parameters* of a neural network architecture is much smaller than the actual number of trainable numbers. There are a number of possible reasons for this, such as:

- Regularization methods explicitly reduce the effective parameter count by adding a prior distribution on the network weights, such as in dropout or batch normalization [8].
- Neural networks also have *implicit regularization* induced by the training method used for the loss function (especially for stochastic methods like batch training), as described in [12].
- The representations learned by neural networks have *flat minima*, such that their weight vectors are robust. As described in [7], adding small stochastic noise to the weight vector does not substantially affect the error.

Each of these observations motivates finding ways to bound the number of effective parameters of a neural network. Although theoretical methods have so far not been able to fully explain the results of experiments testing neural networks on held-out data, they hopefully generate an interesting dialogue about how generalization arises.

2 The Compression Framework

In this section, we describe the compression framework given in [1]. Consider the standard example of a multi-class classification problem, where items in a domain $x \in X$ are classified by labels $y \in \{1, 2, \dots, k\}$. A classifier $f : X \to \mathbb{R}^k$ is a function that outputs a *probability* of achieving each label for every sample in a set. Given some true distribution $\mathcal{D}$ over $X \times \{1, \dots, k\}$, the classification error of f is defined to be

$$\operatorname{error}(f) = \Pr_{(x,y)\sim\mathcal{D}}\left(f(x)[y] \le \max_{i \ne y} f(x)[i]\right).$$

Here, we use the notation $x[i]$ to denote the i-th element of the vector x. This definition of a classifier outputting probabilities may seem roundabout, but it gives us enough information to define *margin*-based losses.

Definition 2.1 (Margin loss)**.** For any value $\gamma \ge 0$, the γ-margin loss of a classifier f on an input distribution $\mathcal{D}$ is defined as

$$L_\gamma(f) = \Pr_{(x,y)\sim\mathcal{D}}\left(f(x)[y] \le \gamma + \max_{i \ne y} f(x)[i]\right).$$

Note that in the special case when $\gamma = 0$, the loss $L_0(f)$ is the same as $\operatorname{error}(f)$.

Now we should ask what it means for a classifier f to be compressible. An elementary way of describing this would be to find a classifier $\hat{f}$ with a smaller representation that roughly agrees f on a selected set of data. Furthermore, we desire to do this in a way that places nicely with the notion of margin defined above.

Definition 2.2 (Compression, initial definition)**.** A classifier f is (γ, S)-*compressible* using a set of classifiers $G_{\mathcal{A}} = \{g_A \mid A \in \mathcal{A}\}$ if there exists a selection of parameters A such that for all $x \in S$,

$$\|f(x) - g_A(x)\|_\infty \le \gamma.$$

However, this definition has limitations because it does not allow for randomness in the compression algorithm. We can do this by introducing a *helper* string to our representation, which does not affect the size of the representation but allows us to take probabilities. For example, this helper string could arise from a pseudorandom number generator based on some small seed.

Definition 2.3 (Compression with helper string)**.** A classifier f is (γ, S)*-compressible with helper string* s using a set of classifiers $G_{\mathcal{A},s} = \{g_{A,s} \mid A \in \mathcal{A}\}$ if there exists a selection of parameters A such that for all $x \in S$,

$$\|f(x) - g_{A,s}(x)\|_\infty \leq \gamma.$$

The main theorem that involving compression bounds is given below. We also provide the proof, as it is elementary and provides insights about the framework.

Theorem 2.4. *If S is a training set with m samples, then 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,*

$$L_0(g_{A,s}) \leq \hat{L}_\gamma(f) + O\left(\sqrt{\frac{\log|\mathcal{A}|}{m}}\right),$$

where $\hat{L}_\gamma(f)$ denotes margin loss on a uniform distribution over the training set S.

Proof. Since f is (γ, S)-compressible, there exists some $A \in \mathcal{A}$ such that the classifier $g_{A,s}$ outputs probabilities each within γ of A for each input in the training set. If f classifies some sample (x, y) correctly with margin at least γ, then $g_{A,s}$ must also classify the same sample correctly. This directly implies that

$$\hat{L}_0(g_{A,s}) \leq \hat{L}_\gamma(f).$$

Now all that remains is to bound the generalization error $|L_0(g_{A,s}) - \hat{L}_0(g_{A,s})|$. We can do this by taking a Chernoff bound on all hypotheses in $\mathcal{A}$.[1] For any fixed $A \in \mathcal{A}$, the empirical classification error of $g_{A,s}$ (taken over a random training set S) is the average of m independent Bernoulli random variables with mean $L_0(g_{A,s})$. By an application of the Azuma-Hoeffding inequality,

$$\Pr(L_0(g_{A,s}) - \hat{L}_0(g_{A,s}) \geq \tau) \leq e^{-2\tau^2 m}.$$

Therefore, if we set $\tau = c\sqrt{\log|\mathcal{A}|/m}$ and apply a union bound, the probability that any $A \in \mathcal{A}$ has true error more than τ plus its empirical error estimate is at most

$$|\mathcal{A}| \cdot \Pr(L_0(g_{A,s}) - \hat{L}_0(g_{A,s}) \geq \tau) \leq e^{-2\tau^2 m} = e^{-2c^2}.$$

Therefore, with confidence at least $1 - \delta$ taken over the choice of a random training set S and helper string s, we have for some $A \in \mathcal{A}$ that

$$L_0(g_{A,s}) \leq \hat{L}_\gamma(f) + \sqrt{\frac{1}{2}\log\frac{1}{\delta} \cdot \frac{\log|\mathcal{A}|}{m}}.$$

□

Note. This is a bound for the generalization error of the *compressed classifier* $g_{A,s}$, not the original classifier f, which allows us to be more straightforward and direct.

[1] Note that this particular result relies on uniform convergence, but on the compressed classifier representation, rather than the fully expressive representation.

3 Compression of a Linear Separator

A simple example of the above compression framework is to prove the generalization of binary linear classifiers with sufficient margin. Here, the model is a linear separator in $\mathbb{R}^n$ represented by parameter vector $c \in \mathbb{R}^n$, where we fix $\|c\| = 1$, where

$$f(x) = \operatorname{sgn}(c^T x).$$

To prove the generalization of a linear classifier, suppose that we have some fixed distribution $\mathcal{D}$ over all labeled samples (x, y), where $x \in \mathbb{R}^n$ and $y \in \{1, -1\}$. Consider some training set $S \sim \mathcal{D}^m$, and let f be a classifier with margin γ on this training set S.

Now thinking about compression, what happens if we were to add independent random noise η with variance σ^2 to each element of the parameter vector c? If so, then the result of the classifier $(c + \eta)^T x$ would be a random variable with mean $c^T x$ and variance σ^2. Roughly speaking, since the margin of the classifier is γ, we should be able to add $\sigma = O(\gamma)$ noise to each parameter of the classifier without affecting the classification of the training data. So given any error parameter $\epsilon > 0$, we compress c as follows:

1. For each index $1 \leq i \leq n$ of c, flip a coin that is heads with probability $p_i = \frac{2c_i^2}{\epsilon\gamma^2}$.
2. If the coin lands heads, set $\hat{c}_i = c_i/p_i$. Otherwise, if tails, set $\hat{c}_i = 0$.
3. Output the compressed vector $\hat{c}_i$.

Theorem 3.1. *The above vector compression method, for any suitable choice of ϵ, outputs a compressed vector $\hat{c}$ with at most $O\left(\frac{\log n}{\epsilon\gamma^2}\right)$ nonzero entries with high probability, and with probability at least $1 - \epsilon$, any fixed unit vector $u \in \mathbb{R}^n$ has*

$$|\hat{c}^T u - c^T u| \leq \gamma.$$

Proof. First, we prove the accuracy part of the theorem. It is easy to verify that the expected value of $\hat{c}_i$ is equal to c_i, and the variance is

$$\mathbf{Var}\left[\hat{c}_i\right] = p_i(c_i/p_i - c_i)^2 + (1 - p_i)c_i^2 \leq 2c_i^2/p_i = \epsilon\gamma^2.$$

Therefore, by Chebyshev's inequality, we have for any fixed unit vector u that

$$\Pr(|\hat{c}^T u - c^T u| \geq \gamma) \leq \frac{\mathbf{Var}\left[\hat{c}^T u\right]}{\gamma^2} \leq \epsilon.$$

Now we prove the concentration inequality for the number of nonzero entries. If we call this X, then X is the sum of Bernoulli random variables with mean p_i, for each $1 \leq i \leq n$. The expected number of nonzero entries is $\mu = \sum p_i = 2/\epsilon\gamma^2$. By a Chernoff bound, for any $\delta \geq 2$,

$$\Pr(X \geq (1 + \delta)\mu) \leq e^{-\frac{\delta^2 \mu}{2+\delta}} \leq e^{-\delta/2}.$$

Therefore, setting $\delta = \Theta(\log n)$ tells us that the probability of having more than $(1 + \delta)\mu$ nonzero entries is $n^{-\Theta(1)}$, as desired. $\square$

This compression method is interesting, but we can't use Theorem 2.4 yet because the parameters are not discrete. The trick is to run the compression method with margin $\gamma/2$, then quantize each parameter $\hat{c}_i$ by rounding it to the nearest multiple of $\gamma/2\sqrt{n}$, or setting it to zero if greater than $2\epsilon\gamma\sqrt{n}$. We omit the proof here for brevity, but this rounded method still outputs a weight vector such that $|\hat{c}^T u - c^T u| \leq \gamma$ with probability at least $1-\epsilon$.

If we let $\mathcal{A}$ be the set of all rounded parameters, then each parameter has a total of $4\epsilon n$ different values, and the number of nonzero parameters is with high probability at most $O(\log n/\epsilon\gamma^2)$. Therefore, with high probability, the size of the representation of the compressed classifier (memorizing incorrect samples) is

$$\log|\mathcal{A}| = O\left(\frac{\log n}{\epsilon\gamma^2}\log(4\epsilon n)\right).$$

Furthermore, by a Chernoff bound, with high probability the number of samples in S that are misclassified by $\hat{c}$ is at most $O(\epsilon m)$. Finally, by Theorem 2.4, we have

$$L_0(\hat{c}) \leq \hat{L}_0(\hat{c}) + O\left(\sqrt{\frac{\log|\mathcal{A}|}{m}}\right) = \tilde{O}\left(\epsilon + \sqrt{\frac{1}{\epsilon\gamma^2 m}}\right).$$

Finally, selecting $\epsilon = (1/\gamma^2 m)^{1/3}$ gives us the following result.

Theorem 3.2. *For any number of samples m, given a linear separator with weight vector c on the set $S \sim \mathcal{D}^m$, we can efficiently find a compressed vector representation $\hat{c}$ such that*

$$L(\hat{c}) \leq \tilde{O}((1/\gamma^2 m)^{1/3}).$$

Note. This generalization bound for the accuracy of the compressed weight vector grows with m, like we expect. However, it only has order $m^{-1/3}$, whereas we would expect a generalization bound of $m^{-1/2}$. It turns out that with a slightly more complex algorithm described in [1], we can achieve a better bound of $\tilde{O}(\sqrt{1/\gamma^2 m})$.

4 Matrix Compression

We now shift our attention to applying the compression framework to matrices as linear transformations, which will be useful for studying neural networks. We first introduce some notation involving the spectral properties of rectangular matrices.

In what follows, consider an $m \times n$ rectangular matrix A, and let $\sigma_1, \ldots, \sigma_r$ be the singular values of A in descending order, where $r = \min(m,n)$.

Definition 4.1 (Frobenius norm)**.** The *Frobenius norm* of A, denoted $\|A\|_F$, is defined by

$$\|A\|_F^2 = \sum_i \sigma_i^2 = \operatorname{trace}(A^T A).$$

Definition 4.2 (Spectral norm)**.** The *spectral norm* of A is the largest singular value $\|A\|_2 = \sigma_1$.

These two matrix norms help us study neural networks by compressing their layer weight matrices using singular value decomposition. Specifically, a common technique is to simply zero out all singular values that are smaller than a given threshold. The rank of a matrix is equal to the number of nonzero singular values; however, in many neural networks, the largest singular value is orders of magnitude bigger than the smallest singular value. This means that they can have a low-rank approximation.

Low-rank approximation is related to noise stability. Note that if η is a random unit vector, then $M\eta$ has magnitude approximately equal to $\|A\|_F/\sqrt{r}$ by SVD. Meanwhile, the absolute maximum of Mx for any fixed unit vector x is the spectral norm $\|A\|_2$. If we consider perturbing a given vector x by noise η, then the ratio between $\|A\|_2$ and $\|A\|_F$ tells us how close $M(x+\eta)$ is to Mx. Instead of measuring the complexity of a neural network layer by rank, it becomes more interesting to have a measure of rank that depends on the relative magnitudes of the singular values.

Definition 4.3 (Stable rank)**.** The *stable rank* of a matrix A is defined as

$$\frac{\|A\|_F^2}{\|A\|_2^2} = \frac{\sum_i \sigma_i^2}{\max_i \sigma_i^2}.$$

In general, the stable rank is at least 1 and at most the true rank of A.

If a matrix has low stable rank, then most of its weight is in its maximum singular value, so it is noise-stable. Conversely, if a matrix has high stable rank, then most of its singular values are around the same order of magnitude, so it is not noise-stable.

Lemma 4.4. *For any matrix $A \in \mathbb{R}^{m\times n}$, let $\tilde{A}$ be the truncated version of A, removing all singular values smaller than $\delta\|A\|_2$. Then, $\|\tilde{A} - A\|_2 \le \delta\|A\|_2$, and $\operatorname{rank}(\tilde{A}) \le \|A\|_F^2/(\delta^2\|A\|_2^2)$.*

Proof. The first fact follows immediately from construction, as the largest singular value of $\hat{A} - A$ is at most $\delta\|A\|_2$. The second fact is true because the smallest singular value of $\hat{A}$ is at least $\delta\|A\|_2$, so $\|A\|_F^2 \ge \|\hat{A}\|_F^2 \ge \delta^2\|A\|_2 \cdot \operatorname{rank}(\hat{A})$. □

Notice that the expression $\|A\|_F^2/(\delta^2\|A\|_2^2)$ in the above lemma is just the stable rank of A divided by δ^2. In general, an $m \times n$ matrix of rank r can be expressed with $(m+n)r$ parameters using the singular value decomposition. Thus, when a matrix has low stable rank (as is often the case with neural networks), this approach gives us a way to compress the matrix from mn parameters to $m+n$ times the stable rank times $1/\delta^2$, where δ measures the maximum amount of tolerable error (i.e., the largest singular value of $\hat{A} - A$).

We will apply this lemma in the next section to derive generalization bounds for neural networks. This compression approach is interesting because it takes advantage of the spectral properties of weight matrices (i.e., low stable rank). However, note that the analysis is very pessimistic, assuming the worst-case Lipschitz constant of the neural network (which grows exponentially with depth). This is because the algorithm removes a deterministic set of the smallest singular values.

However, as shown in Fig. 1, this pessimistic prediction is not what happens in practice. Noise injected in earlier layers tends to quickly be attenuated as it propagates to later layers. It turns out that we can obtain better generalization bounds if we instead compress the matrix in a way so that the error is *Gaussian-like*, using randomness in the algorithm, which lets us describe this attenuation behavior.

To compress, we perform the following projection-based algorithm on each layer weight matrix A, using our fixed "helper string" of random bits s, with error parameters $\epsilon, \eta > 0$:

1. Sample $k = 8\log(1/\eta)/\epsilon^2$ random matrices $M_1, \ldots, M_k$, with entries i.i.d. ± 1.
2. For each $1 \le j \le k$, let $Z_j = \langle A, M_j\rangle M_j$.
3. Let $\hat{A} = \frac{1}{k}\sum_{j=1}^{k} Z_j$.

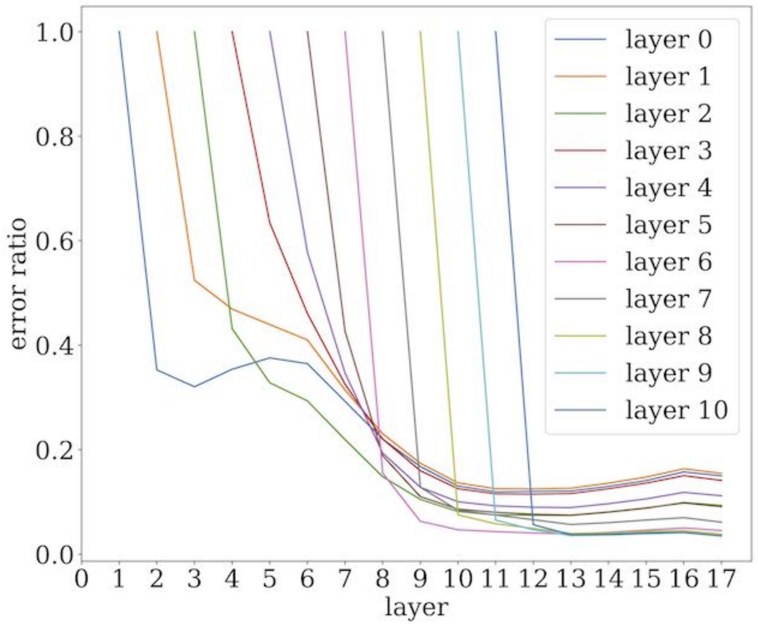

Figure 1: Attenuation of Gaussian noise injected at various neural network layers ([1]).

Lemma 4.5. *Suppose that we have an $h_1 \times h_2$ matrix A. For any $0 < \delta, \epsilon \leq 1$, and any fixed set $G = \{(U^i, x^i)\}_{i=1}^m$ of m pairs, where U is a $n \times h_1$ matrix and $x \in \mathbb{R}^{h_2}$, run the randomized projection algorithm with $\eta = \delta/mn$, outputting a compressed matrix $\hat{A}$. Then, with confidence at least $1 - \delta$, we have for any $(U, x) \in G$ that*

$$\|U(\hat{A} - A)x\| \leq \epsilon \|U\|_F \|A\|_F \|x\|.$$

Proof. This is a consequence of concentration of measure. By applying the Johnson-Lindenstrauss lemma, the probability that any one individual coordinate of $\|U(\hat{A} - A)x\|$ is too large is bounded by $1 - \eta$. The rest follows from a union bound. $\square$

5 Neural Network Generalization Bounds

In this section, we look at some of the generalization bounds based on the previous compression algorithms. For simplicity, let's start by only considering fully-connected deep neural networks with ReLU activations (denoted ϕ), as well as constant layer widths. A practical way of compressing neural networks is to take the top few singular values of their weight matrices.

For notation, let d be the number of layers in the neural network, and let h be the width of each layer. Denote x^i to be the preactivation after the i-th layer, so x^0 is the input to the neural network, while x^d is the output. Also, let the d layers have weight matrices $A^1, \ldots, A^d$, so the feedforward computation is $x^i = A^i \phi(x^{i-1})$ for all $1 \leq i \leq d$.

The first compression method in Lemma 4.4, by erasing low singular values in the SVD, gives us the following generalization bound. This actually reproduces previous results that were obtained through a PAC-Bayes framework, but with a simpler proof!

Theorem 5.1 ([13]). *The generalization error of the compressed representation of a deep neural network with layers $A^1, \dots, A^d$, width h, and output margin γ on a training set S is at most*

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

Proof. We apply Lemma 4.4 to each layer of the neural network, each time setting

$$\delta = \gamma \left(ed \max_{x \in S} \|x\| \prod_{i=1}^{d} \|A^i\|_2 \right)^{-1}.$$

Notice how the product of the spectral norms of the weight matrices is also the Lipschitz constant of the neural network, assuming 1-Lipschitz ReLU activations. This means that for any i, the error at the output from compressing layer i is at most $\delta \|x_i\| \prod_{j=i}^{d} \|A^j\|_2 \leq \gamma/ed$. Since the neural network layers have no bias terms, we also have $\|x_i\| \leq \|x\| \prod_{j=0}^{i-1} \|\hat{A}^j\|_2$. As we can see, the errors for each layer depend on those of previous layers, but we can show by induction that they add up in total to at most

$$\|f_A(x) - f_{\hat{A}}(x)\| \leq e\|x\| \left(\prod_{i=1}^{d} \|A^i\|_2 \right) \sum_{i=1}^{d} \frac{\|A^i - \hat{A}^i\|_2}{\|A^i\|_2} \leq e\|x\| \left(\prod_{i=1}^{d} \|A^i\|_2 \right) d\delta \leq \gamma.$$

Hence, this SVD approach is a (γ, S)-compression algorithm. We can then apply a quantization process similar to Theorem 3.2, by rounding each real parameter to the nearest multiple of $\|A\|_F/h^2$. Then, using the fact that the rank of $\|\hat{A}^i\|_F$ is at most its stable rank multiplied by $1/\delta^2$, the number of parameters of the compressed network is

$$\sum_{i=1}^{d} 2h \frac{\|A^i\|_F^2}{\delta^2 \|A^i\|_2^2} = \frac{1}{\gamma^2} 2e^2 d^2 h \|x\|^2 \prod_{i=1}^{d} \|A^i\|_2^2 \sum_{i=1}^{d} \frac{\|A^i\|_F^2}{\|A^i\|_2^2}.$$

Finally, the result follows from Theorem 2.4, after removing constants $(2e^2)$ and logarithmic factors (the size of each parameter as an $O(\log h)$-bit vector). □

Although the math got a little messy at the end, the key idea behind this last generalization bound was still conceptually simple. Taking a low-rank approximation of each weight matrix allowed us to greatly reduce the number of parameters it took to represent the matrix, while still maintaining an acceptable level of error within the margin γ. In summary, the ratio between this generalization error bound and the trivial non-compressed bound (using $h^2 d$ parameters) is proportional to (Lipschitz constant) $\times$ (stable rank)/γ. When the Lipschitz constant and stable rank are small, this results in a significant improvement.

However, recall that this analysis is pessimistic about the worst-case noise blowup at each layer, and thus, we have an exponential dependence on d where we multiply by every spectral norm. We can instead use the Johnson-Lindenstrauss projections from Lemma 4.5 to guide our analysis, focusing on the Jacobian of the neural network (rather than singular values, which do not exist in a nonlinear setting). Let $J_x^{i,j}$ represent the Jacobian of layers i through j inclusive, at the point x. Note that since the activations are ReLU, $J_{x^i}^{i,j} x^i = x^j$.

Definition 5.2 (Noise sensitivity)**.** If M is a map between real vectors and $\mathcal{N}$ is a noise distribution, then the *noise sensitivity* of M at a point x with respect to $\mathcal{N}$ is

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

Furthermore, we also define the noise sensitivity of M with respect to a set of inputs S to be the maximum of $\psi_{\mathcal{N}}(M, x)$ over all $x \in S$. This is denoted $\psi_{\mathcal{N},S}(M)$.

This definition provides a general way to state the observations we made about stable rank in Lemma 4.4. In particular, we have the following example showing how the noise sensitivity is low at vectors corresponding to large singular values.

Proposition 5.3. *Given a matrix M, the noise sensitivity of M at any nonzero vector x with respect to the multivariate standard Gaussian distribution $\mathcal{N}(0, I)$ is equal to $\|M\|_F^2\|x\|^2/\|Mx\|^2$.*

Notice how the noise sensitivity is equal to the stable rank of M when x is a vector corresponding to the largest singular value, i.e., $\|Mx\| = \|M\|_2\|x\|$. Since our compression algorithm results in approximately Gaussian noise at each intermediate layer, we can compress the network close to its stable rank as long as the intermediate preactivation vectors x^i are correlated with high singular directions of M^i. We formalize this below with a few **data-dependent** notions.

Definition 5.4 (Layer cushion)**.** The *layer cushion* of layer i is a value $0 \le \mu_i \le 1$, roughly representing the maximum correlation of any $x \in S$ to high singular values of A^i. Formally,

$$\mu_i = \max_{x \in S} \frac{\|A^i\phi(x^{i-1})\|}{\|A^i\|_F\|\phi(x^{i-1})\|} = \max_{x \in S} \frac{\|x^i\|}{\|A^i\|_F\|\phi(x^{i-1})\|}.$$

Note that the noise sensitivity of layer A^i to Gaussian noise $\mathcal{N}(0, I)$ at $\phi(x^{i-1})$ is equal to $1/\mu_i^2$. Those, layers with higher layer cushion are less noise-sensitive, for a given set of samples.

Definition 5.5 (Interlayer cushion)**.** For any choice of two layers $i \le j$, the *interlayer cushion* $\mu_{i,j}$ is defined to be

$$\mu_{i,j} = \max_{x \in S} \frac{\|J_{x^i}^{i,j} x^i\|}{\|J_{x^i}^{i,j}\|_F\|x^i\|}.$$

Observe that $\mu_{i,i} = \sqrt{h}$. We also define the *minimal interlayer cushion* for a layer i to be

$$\mu_{i\rightarrow} = \min_{i \le j \le d} \mu_{i,j}.$$

Finally, we need a way to understand how noise changes through the activation function. Usually, ReLU only zeros out some fraction of the neurons in a trained neural network, so this is just a constant factor.

Definition 5.6 (Activation contraction)**.** The activation contraction c is defined as

$$c = \min_{\substack{x \in S \\ 1 \le i \le d}} \frac{\|\phi(x^i)\|}{\|x^i\|}.$$

Finally, there is another property of deep neural networks called *interlayer smoothness*, denoted ρ_δ, which tends to be quite high for most trained neural networks. We will not discuss it here for brevity, but see [2] for details. Using all of these data-dependent properties of a neural network, here is the resulting generalization bound of the Johnson-Lindenstrauss compression algorithm.

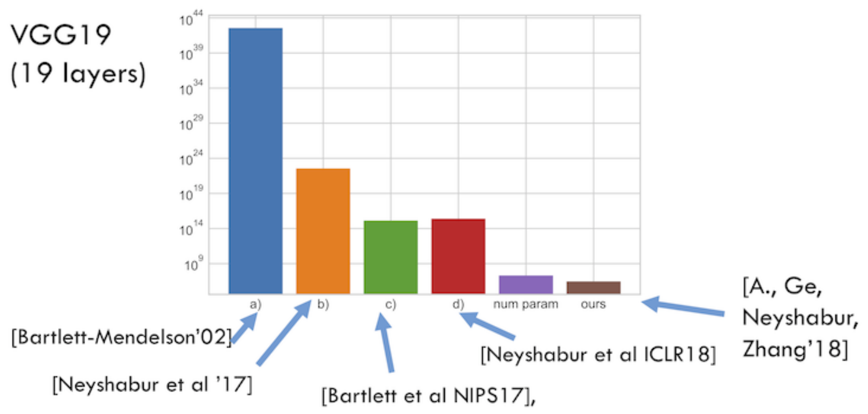

Figure 2: Comparison of effective parameters from various generalization bounds (lower is better). Here, the orange bar represents the bound from Theorem 5.1, while the brown bar represents the bound from Theorem 5.7. The purple bar is the empirical estimate from held-out data.

Theorem 5.7 ([2]). *For any fully connected network f_A with $\rho_\delta \geq 3d$, if $f_{\tilde{A}}$ is the network after compression, then with probability $1-\delta$, we have*

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

The key thing to note about this bound, compared to Theorem 5.1, is that it no longer depends on the Lipschitz constant of the neural network that is exponential in the number of layers. Instead, it only depends linearly based on the summation of the reciprocals of layer cushion and minimum interlayer cushion, which are dependent on the correlation of the actual input data to high singular values of the neural network. As shown in Fig. 2, actual estimates of the number of *effective parameters* are much smaller using this approach!

6 Future Directions

The work in this section is original.

The compression framework is interesting because it is a powerful yet elementary way of developing generalization bounds for neural networks. It is perhaps the most clear example of how the number of *effective parameters* differs from the true number of parameters, since large neural networks can be approximated by much smaller neural networks on real-world data.

While theory is still not at the level of explaining real-world generalization, it may still be interesting to see how these generalization bounds compare for alternative neural network architectures. For example, we can generalize Theorem 5.1 to ResNets [6] with simple residual connections.

Theorem 6.1. *The generalization error of the compressed representation of a residual neural network (HighwayNet) with layers $A^1, \ldots, A^d$, double-layer skip connections $B^2, \ldots, B^d$, width h, and output margin γ on a training set S is at most*

$$\tilde{O}\left(\sqrt{\frac{1}{\gamma^2 m} \cdot hd^2 \cdot \max_{x \in S} \|x\| \cdot \mathcal{L}_d^2 \cdot \left(\sum_{i=1}^{d} \frac{\|A^i\|_F^2}{\|A^i\|_2^2} + \sum_{i=2}^{d} \frac{\|B^i\|_F^2}{\|B^i\|_2^2}\right)}\right),$$

where $\mathcal{L}_i$ represents the Lipschitz constant of the first i layers $J^{1,i}$, defined by the recursive formulas $\mathcal{L}_1 = \|A^1\|_2$ and $\mathcal{L}_i = \|A^i\|_2\mathcal{L}_{i-1} + \|B^i\|_2\mathcal{L}_{i-2}$.

Although ResNets with double-layer skips are a simple example, the same techniques could also be used to bound simple convolutional neural networks and also fine-tuned models like EfficientNet [17], which may glean insights about their performance.

Finally, it may be interesting to experiment and see how the compression methods used in these generalization bounds compare on applied deep neural network models, when taking standard deep compression pipelines as a benchmark. For example, does random matrix projection actually perform better than pruning small singular values in practice?

References

[1] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR, 2018.

[2] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

[3] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[4] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[5] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*, 2016.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] 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, 1993.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[9] Xingguo Li, Junwei Lu, Zhaoran Wang, Jarvis D. Haupt, and Tuo Zhao. On tighter generalization bound for deep neural networks: Cnns, resnets, and beyond. *CoRR*, abs/1806.05159, 2018.

[10] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 8168–8177, 2018.

[11] Vaishnavh Nagarajan and J Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 11611–11622, 2019.

[12] Behnam Neyshabur. Implicit regularization in deep learning. *CoRR*, abs/1709.01953, 2017.

[13] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. *CoRR*, abs/1707.09564, 2017.

[14] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[15] John Shawe-Taylor and Omar Rivasplata. Statistical learning theory: A hitchhiker's guide. Talk given at the meeting of the Conference on Neural Information Processing Systems, Vancouver, Canada, December 2018.

[16] Taiji Suzuki. Compression based bound for non-compressed network: unified generalization error analysis of large compressible deep neural network. *arXiv preprint arXiv:1909.11274*, 2019.

[17] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[18] Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. *CoRR*, abs/1705.07809, 2017.

[19] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.