Fantastic Generalization Measures and Where to Find Them

Anonymous authors
Paper under double-blind review

Abstract

Generalization of deep networks has been of great interest in recent years, resulting in a number of theoretical bounds and empirically motivated measures. However, most papers proposing such measures only study a small set of models, leaving open the question of whether the conclusion drawn from those experiments would generalize to other settings. We present the first large scale study of generalization bounds and measures in deep networks. We train over two thousand convolutional networks with systematic changes in important hyper-parameters. Hoping to uncover potentially causal relationships between each measure and generalization, we run carefully controlled experiments and use a modified form of rank correlation coefficient to compare different measures overall in individual experiment categories. We analyze the results and show surprising failures of some measures as well as promising measures for further research.

1 Introduction

Deep neural networks have seen tremendous success in a number of applications, but why these models generalize is still a mystery (Zhang et al., 2016; Recht et al., 2019; Zhang et al., 2019). It is crucial to better understand the reason behind the generalization of modern deep learning models; such an understanding has multiple benefits, including providing guarantees for safety critical scenarios and the design of better models.

A number of papers have attempted to understand the phenomenon of generalization of deep learning models from a theoretical perspective e.g. (Neyshabur et al., 2015b; Bartlett et al., 2017; Neyshabur et al., 2018a; Li et al., 2018; Arora et al., 2018; Long & Sedghi, 2019). The most direct and principled approach for studying generalization in deep learning is to prove *generalization bounds*. Unfortunately, finding tight bounds has proven to be an arduous undertaking. An alternative to capture generalization is to define a measure (e.g. sharpness) that, given two models, would be able to predict which of them has a smaller generalization gap. This measure is allowed to depend on general properties of the data in the training set but is not allowed to have access to a validation set. Various measures are often featured as the major components of generalization bounds. Other works have directly proposed empirical measures that focus on certain properties of deep networks, without any attempt at deriving bounds (Keskar et al., 2016; Liang et al., 2017).

However, the empirical evaluation of these bounds and measures is usually limited to a few models, often on toy problems. A measure can only be considered reliable as a predictor of generalization gap if it is tested extensively on many models at a realistic problem size. To close this gap, we carefully selected a number of generalization bounds from the literature which leverage a diverse range of complexity measures. We further selected a variety of measures that have empirically shown the promise of being related to improved generalization, such as sharpness (Keskar et al., 2016), Fisher-Rao norm (Liang et al., 2017) and path norms (Neyshabur et al., 2017). These measures are based on the norms of the network (such as weights), flatness of the loss landscape, or related to the optimization function (such as cross-entropy loss value).

In this study, we trained a large number of models on the CIFAR-10 dataset (Krizhevsky et al., 2014), by carefully varying empirically important hyperparameters. We also selected

multiple optimization algorithms and looked at different stopping criteria for training convergence. Details of all our measures and hyperparameter selections are provided in Appendix C. For any such model, we considered more than 38 bounds and measures. The key findings that arise from our large scale study are summarized below:

1. It is very easy for complexity measures to capture spurious correlations that do not reflect more causal insights about generalization; to mitigate this problem, we propose a more rigorous approach for studying them.
2. Many norm-based measures not only perform poorly, but ***negatively*** correlate with generalization particularly when the optimization procedure injects stochasticity.
3. PAC-bayesian (McAllester, 1999) and sharpness (Keskar et al., 2016) based measures perform the best overall and seem to be promising candidates for further research.
4. Measures related to the optimization procedures can be predictive of generalization through all hyperparameter categories.

2 Related Work

The generalization bounds that we consider in this work belong to a few different families: PAC-Bayes (McAllester, 1999; Dziugaite & Roy, 2017; Neyshabur et al., 2017); VC-dimension (Vapnik & Chervonenkis, 1971); and norm-based bounds (Neyshabur et al., 2015b; Bartlett et al., 2017; Neyshabur et al., 2018a). Notably, Dziugaite & Roy (2017) was the first work to compute non-vacuous generalization bound on simplified MNIST. The measures from prior literature that we consider are sharpness (Keskar et al., 2016); Fisher-Rao norm (Liang et al., 2017); distance of trained weights from initialization (Nagarajan & Kolter, 2019) and path norm (Neyshabur et al., 2015a).

A few papers have explored a large scale study of generalization in deep networks. Jiang et al. (2018) studied the role of margin as a predictor of the generalization gap. However, they used a significantly more restricted set of models (e.g. no depth variations), the experiments were not controlled for potential undesired correlation (e.g. the models can have vastly different training error) and the measure contains parameter that must be learned from a set of models. Our work focuses on measures that can be computed on a single model and compares a large number of bounds and measures across a much wider range of models in a more controlled fashion. Neyshabur et al. (2017) perform a small scale study of the generalization of PAC-Bayes, sharpness and a few different norms. In contrast, we study thousands of models to arrive at a more robust conclusion.

3 Notation

We denote a probability distribution as $\mathscr{A}$, set as $\mathcal{A}$, tensor as $\mathbf{A}$, vector as $\mathbf{a}$, and scalar as a or α. Let $\mathscr{D}$ denote the data distributions over inputs and their labels, and let κ denote number of classes. We use $\triangleq$ for equality by definition. We denote by $\mathcal{S}$ a given dataset, consisting of m i.i.d tuples $\{(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_m, y_m)\}$ drawn from $\mathscr{D}$ where $\mathbf{X}_i \in \mathcal{X}$ is the input data and $y_i \in \{1, \ldots, \kappa\}$ the corresponding class label. We denote a feedforward neural network by $f_{\mathbf{w}} : \mathcal{X} \to \mathbb{R}^{\kappa}$, its weight parameters by $\mathbf{w}$, and the the number of weights by $\omega \triangleq \dim(\mathbf{w})$. No activation function is applied at the output (i.e. logits). Denote the weight tensor of the i^{th} layer of the network by $\mathbf{W}_i$, so that $\mathbf{w} = \mathrm{vec}(\mathbf{W}_1, \ldots, \mathbf{W}_d)$, where d is the depth of the network. Furthermore, denote by $f_{\mathbf{w}}(\mathbf{X})[j]$ the j-th output of the function $f_{\mathbf{w}}(\mathbf{X})$.

Let $\mathcal{R}$ be the set of binary relations, and $I : \mathcal{R} \to \{0, 1\}$ be the indicator function that is 1 if its input is true and zero otherwise. Let L be the 1-0 classification loss over the data distribution $\mathscr{D}$: $L(f_{\mathbf{w}}) \triangleq \mathbb{E}_{(\mathbf{X},y)\sim\mathscr{D}}\left[I\big(f_{\mathbf{w}}(\mathbf{X})[y] \leq \max_{j\neq y} f_{\mathbf{w}}(\mathbf{X})[j]\big)\right]$ and let $\hat{L}$ be the empirical estimate of 1-0 loss over $\mathcal{S}$: $\hat{L}(f_{\mathbf{w}}) \triangleq \frac{1}{m}\sum_{i=1}^{m} I\big(f_{\mathbf{w}}(\mathbf{X})[y_i] \leq \max_{j\neq y_i} f_{\mathbf{w}}(\mathbf{X})[j]\big)$ We refer to $L(f_{\mathbf{w}}) - \hat{L}(f_{\mathbf{w}})$ as the generalization error. For any input $\mathbf{X}$, we define the

sample dependent margin[1] as $\gamma(\mathbf{X}) \triangleq \big(f_{\mathbf{w}}(\mathbf{X})\big)[y] - \max_{i \neq y} f_{\mathbf{w}}(\mathbf{X})_i$. Moreover, we define the overall margin γ as the 10^{th} percentile (a robust surrogate for the minimum) of $\gamma(\mathbf{X})$ over the entire training set $\mathcal{S}$. More notation used for derivation is located in Appendix B.

4 Generalization: What is the goal and how to evaluate?

Generalization is arguably the most fundamental and yet mysterious aspect of machine learning. The core question in generalization is what causes the pair of a model and an optimization algorithm to generalize well beyond the training set. There are many hypotheses concerning this question, but what is the right way to compare these hypotheses? Here we briefly discuss some potential approaches to compare different hypotheses:

- Generalization Bounds: Proving generalization bounds is very useful to establish the causal relationship between a complexity measure and the generalization error. However, almost all existing bounds are vacuous on current deep models and datasets and therefore, one cannot rely on their proof as an evidence on the causal relationship between a complexity measure and generalization currently[2].
- Regularizing the Measure: One may evaluate a complexity measure by adding it as a regularizer and directly optimizing it, but this could fail due to two reasons. The complexity measure could change the loss landscape in non-trivial ways and make the optimization more difficult. In such cases, if the optimization fails to optimize the measure, no conclusion can be made about the causality. Another perhaps more critical problem is the existence of implicit regularization of the optimization algorithm. This makes it hard to run a controlled experiment since one cannot simply turn off the implicit regularization; therefore, if optimizing a measure does not improve generalization it could be simply due to the fact that it is regularizing the model in the same way as the optimization is regularizing it implicitly.
- Correlation with Generalization: Evaluating measures based on correlation with generalization is interesting but potentially dangerous. To check the correlation, we should vary architectures and optimization algorithms to produce a set of models. If the set is generated in an artificial way and is not representative of the typical setting, the conclusions might be deceiving and might not generalize to typical cases. One such example is training with different portions of random labels which artificially changes the dataset. Another pitfall is drawing conclusion from changing one or two hyper-parameters (e.g changing the width or batch-size and checking if a measure would correlate with generalization). In these cases, the hyper-parameter could be the true cause of both change in the measure and change in the generalization but the measure itself has no causal relationship with generalization.

While acknowledging all limitations of a correlation analysis, we try to improve the procedure and capture some of the causal effects as much as possible through careful design of experiments and metrics. To this end, we can try to partially capture this causal effect by changing some hyper-parameters and keeping the rest fixed and see if the measure can still correlate with generalization; with this controlled experiment, we are able to partially capture the causal relationship. Further, to evaluate the effectiveness of a measure, it is crucial to test it on models with a wide range of variations, and each model must be sufficiently trained, if not to complete convergence. For practical reasons, they must also reach this stage within a reasonable time budget.

4.1 Training Models across Hyperparameter Space

In order to create models with different generalization behavior, we consider various types of hyperparameters, which are known or believed to influence generalization (e.g. batch

[1] This work only concerns with the output margins, but generally margin can be defined at any layer of a deep network (Elsayed et al., 2018).

[2] Please see Dziugaite & Roy (2017) for an example of non-vacuous generalization bound and related discussions.

size, dropout rate, etc.). Formally, denote each hyperparameter by θ_i taking values from the set Θ_i, for $i = 1, \ldots, n$ and n denoting the total number of hyperparameter types[3]. For each element $\boldsymbol{\theta} \triangleq (\theta_1, \theta_2, \ldots, \theta_n) \in \Theta$, where $\Theta \triangleq \Theta_1 \times \Theta_2 \times \cdots \times \Theta_n$, we train the architecture with hyperparameters $\boldsymbol{\theta}$ until the training loss (cross entropy value) reaches a given threshold ϵ. Please see the Appendix A.2 for a discussion on the choice of the stopping criterion. Doing this for each hyper-parameter configuration $\boldsymbol{\theta} \in \Theta$, we obtain a ***total*** of $|\Theta|$ models. Θ reflects our prior knowledge about what a reasonable hyperparameter setting is and choosing what Θ_i's represent and how many points are in each Θ_i can be seen as a grid sampling from the hyperparameter space.

4.2 Kendall's Correlation Coefficient

One way to evaluate the quality of a measure μ is through *ranking*. Given a set of models that are trained with hyperparameters in the set Θ, their associated generalization gap $\{g(\boldsymbol{\theta}) \,|\, \boldsymbol{\theta} \in \Theta\}$ and their respective values of the measure, $\{\mu(\boldsymbol{\theta}) \,|\, \boldsymbol{\theta} \in \Theta\}$, our goal is to analyze how consistent a measure (e.g. ℓ_2 `norm`) is with the empirically observed generalization. To this end, we construct a set $\mathcal{T}$, where each element of the set is associated with one of the trained model. Each element has the form of a pair: complexity measure μ versus generalization gap g [4].

$$\mathcal{T} \triangleq \cup_{\boldsymbol{\theta} \in \Theta} \big\{ \big(\mu(\boldsymbol{\theta}), g(\boldsymbol{\theta}) \big) \big\} . \tag{1}$$

An ideal measure must be such that, for any pair of trained models, if $\mu(\boldsymbol{\theta}_1) > \mu(\boldsymbol{\theta}_2)$, then so is $g(\boldsymbol{\theta}_1) > g(\boldsymbol{\theta}_2)$. We use Kendall's rank coefficient τ to capture to what degree such consistency holds among the elements of $\mathcal{T}$.

$$\tau(\mathcal{T}) \triangleq \frac{1}{|\mathcal{T}|(|\mathcal{T}|-1)} \sum_{(\mu_1, g_1) \in \mathcal{T}} \sum_{(\mu_2, g_2) \in \mathcal{T} \setminus (\mu_1, g_1)} \operatorname{sign}(\mu_1 - \mu_2) \operatorname{sign}(g_1 - g_2) \tag{2}$$

Note that τ can only vary from 1 to -1, attaining these extreme values at perfect agreement (two rankings are the same) and perfect disagreement (one ranking is the reverse of the other) respectively. If complexity and generalization are independent, the coefficient becomes zero. Note that τ is non-parametric and can be efficiently evaluated on $\mathcal{S}$ of any size.

4.3 Towards a More Causal Measure

While Kendall's correlation coefficient is an effective tool widely used to measure relationship between 2 rankings of a set of objects, we found that certain measures can achieve high τ values in a trivial manner – i.e. the measure may strongly correlate with the generalization performance without necessarily capturing the cause of generalization. We will analyze this phenomenon in greater details in subsequent sections. To mitigate the effect of spurious correlations, we propose a new quantity for quantifying the correlation between measures and generalization that is based on a more controlled setting and hence can better capture the causal nature. None of the existing complexity measures is perfect. However, they might have different sensitivity and accuracy w.r.t. different hyperparameters. For example, sharpness may do better than other measures when only a certain hyperparameter (say batch size) changes. To understand such details, in addition to $\tau(\mathcal{T})$, we compute τ for consistency within each hyperparameter axis Θ_i, and then average the coefficient across the entire hyperparameter space. Formally, we define:

$$m_i \triangleq |\Theta_1 \times \cdots \times \Theta_{i-1} \times \Theta_{i+1} \times \cdots \times \Theta_n| \tag{3}$$

$$\psi_i \triangleq \frac{1}{m_i} \sum_{\theta_1 \in \Theta_1} \cdots \sum_{\theta_{i-1} \in \Theta_{i-1}} \sum_{\theta_{i+1} \in \Theta_{i+1}} \cdots \sum_{\theta_n \in \Theta_n} \tau \big(\cup_{\theta_i \in \Theta_i} \{ \big(c(\boldsymbol{\theta}), g(\boldsymbol{\theta}) \big) \} \big) \tag{4}$$

The inner τ measures the ranking correlation between the generalization and the measure for a small group of models where the only difference among them is the variation along

[3] In our analysis we use $n = 7$ hyperparameters: batchsize, dropout probability, learning rate, network depth, weight decay coefficient, network width, optimizer.

[4] generalization gap is defined as difference of train and test accuracy

a single hyperparameter axis Θ_i. Then average the value across all combinations of the other hyperparameter axis. Intuitively, if a measure is good at predicting the effect of hyperparameter Θ_i over the model distribution, then its corresponding ψ_i should be high. Then, we compute the ψ_i's average across all hyperparamter axis:

$$\Psi \triangleq \frac{1}{n} \sum_{i=1}^{n} \psi_i \tag{5}$$

If a measure achieves a high Ψ on a given model distribution Θ, then it should achieve high individual ψ across all hyperparameters. A measure that excels at predicting changes in a single hyperparameter but fails at the other hyperparameters will not do well on this measure. On the other hand, if the measure performs well, it means that the measure can reliably rank the generalization for each of the hyper-parameter changes.

A ***thought experiment*** to illustrate why Ψ captures a *better* causal nature of the generalization is as follows: suppose there exists a measure that perfectly captures the depth of the network while producing random prediction if 2 networks have the same depth, this measure would do reasonably well in terms of τ but much worse in terms of Ψ. In the experiments we consider in the following sections, we found that such a measure would achieve overall $\tau = 0.362$ but $\Psi = 0.11$.

Finally, we acknowledge this this measure is only a small step towards the difficult problem of capturing the causal relationship between complexity measures and generalization in empirical settings, and we hope this encourages future work in this direction.

5 Generating a family of trained models

We choose 7 common hyperparameters related to optimization and architecture design, with 3 choices for each hyperparameter, to generate $3^7 = 2187$ models. We trained these models to convergence to cross entropy 0.01 over the training set and removed any model that was not able to achieve this cross-entropy[5]; this is different from the DEMOGEN dataset (Jiang et al., 2018) where the models are not trained to the same cross-entropy. Putting the stopping criterion on *the training loss* rather than *the number of epochs* is crucial since otherwise one can simply use cross-entropy loss value to predict generalization. Please see Appendix Section A.2 for a discussion on the choice of stopping criterion. To achieve this goal, we tried a wide range of hyperparameters to choose ensure that models trained using combination of these choices have vastly different generalization behaviors while being able to fit the training set. Our base model is inspired by the Network-in-Network (NiN, Gao et al. (2011)) models and the hyperparameter categories we test on are: weight decay coefficient (`weight_decay`), width of the layer (`width`), mini-batch size (`batchsize`), learning rate (`learning_rate`), dropout probability (`dropout`), depth of the architecture (`depth`) and the choice of the optimization algorithms (`opt`). We select 3 choices for each hyperparameter, but different optimization algorithms can affect other hyperparameters such as learning rate or weight regularization. Please refer to Appendix A.3 for the details on the models, and Appendix A.1 for the reasoning behind the design choices.

Figure 1 shows some summarizing statistics of the models in this study. On the left we show the number of models that achieve above 99% training accuracy for every individual hyperparameter choice. Since we have 2187 models in total, the maximum number of model for each category is 718; the majority of the models were able to reach this threshold. In the middle we show the distribution of the cross-entropy value over the entire training set. While we want the models to be at exactly 0.01 cross-entropy, in practice it is computationally prohibitive to constantly evaluate the loss over the entire training set; further, to enable reasonable temporal granularity, we estimate the training loss with 100 randomly sampled minibatch. These computational compromises result in long-tailed distribution of training loss centered at 0.01. As shown in Table 1, even such minuscule range of cross-entropy difference could lead to positive correlation with generalization, highlighting the importance of training loss as a stopping criterion. On the right, we show the distribution of the

[5]In our analysis, less than 5 percent of the models do not reach this threshold.

generalization gap. We see that while all the models' training accuracy is above 0.99, there is a wide range of generalization gap which is ideal for evaluating complexity measures.

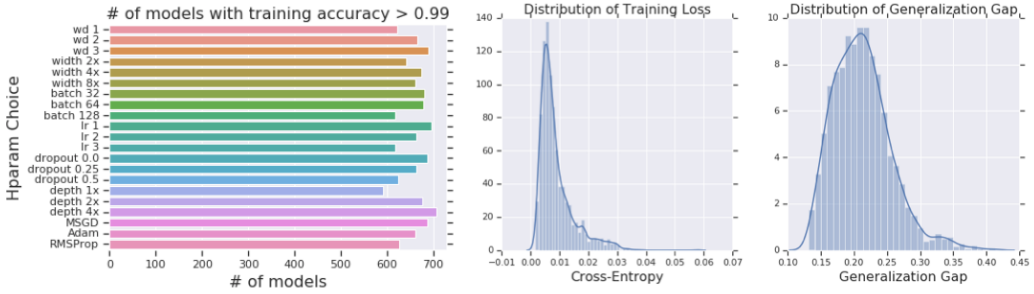

Figure 1: **Left**: Number of models with training accuracy above 0.99 for each category. **Middle**: Distribution of training cross-entropy; distribution of training error can be found in Fig. 2. **Right**: Distribution of generalization gap.

6 Baselines

An important baseline we consider is how does a measure compare against an oracle who observes noisy generalization gap. This baseline accounts for the potential noise in the training procedure and gives an anchor for gauging the difficulty of each hyper-parameter category. Formally, given a set of hyper-parameters Θ' we define oracle ϵ to be the expectation of τ or Ψ where the measure is $\{g(\boldsymbol{\theta}) + \mathcal{N}(0, \epsilon^2) \,|\, \boldsymbol{\theta} \in \Theta'\}$. We report the performance of the noisy oracle in Table 1 for $\epsilon \in \{0.01, 0.02, 0.05, 0.1\}$.

To understand how our hyperparameter choices affect the optimization, we give each hyperparameter category a *canonical order* which is believed by the community to have correlation with generalization (e.g. larger learning rate generalizes better) and measure their τ. The exact canonical ordering can be found in Appendix A.4. Note that each canonical ordering that only be predictive for its own category since its corresponding hyperparameter is remained fixed in any other category. As a result, the Ψ metric for each canonical ordering is $\frac{1}{7}$ of its performance on the related category.

We next look at of the most well-known complexity measures in machine learning is the VC-Dimension. Bartlett et al. (2019) proves bounds on the VC dimension of piece-wise linear networks with potential weight sharing. In Appendix C.1, we extend their result to include pooling layers and multi-class classification. We report two measure based on VC-dimension bounds or and parameter counting. These measures could only be predicative when the architecture changes which only happens in depth and width category. We observe that in both categories VC-dimension and number of parameters are negatively correlated with generalization gap which confirms the widely known empirical observation that overparametrization improves generalization in deep learning.

Finally, we report the measures that only look at the output of the network. In particular, we look at the cross-entropy loss, margin γ and the entropy of the output. All three measures are closely related to each other. In fact, the results in Table 1 also reflects this similarity. The results confirm the general understanding that larger margin, lower cross-entropy and higher entropy would lead to better generalization and their performance is slightly worse than oracle 0.1. Please see Appendix C.1.1 for definitions and more discussions on these measures.

7 Surprising Failure of some Norm/margin-based Measures

In machine learning, a long standing measure for quantifying the complexity of a function, and therefore generalization, is using various norms of the given function. Some of the norms

	batchsize	dropout	learning_rate	depth	optimizer	weight_decay	width	overall τ	Ψ
vc dim 15	0.000	0.000	0.000	-0.909	0.000	0.000	-0.171	-0.251	-0.154
# params 16	0.000	0.000	0.000	-0.909	0.000	0.000	-0.171	-0.175	-0.154
$1/\gamma$ 18	0.312	-0.593	0.234	0.758	0.223	-0.211	0.125	0.124	0.121
-entropy 19	0.346	-0.529	0.251	0.632	0.220	-0.157	0.104	0.148	0.124
cross-entropy 17	0.440	-0.402	0.140	0.390	0.149	0.232	0.080	0.149	0.147
oracle 0.01	0.590	0.850	0.706	0.902	0.550	0.630	0.509	0.850	0.677
oracle 0.02	0.380	0.657	0.536	0.717	0.374	0.388	0.360	0.714	0.487
oracle 0.05	0.172	0.375	0.305	0.384	0.165	0.184	0.204	0.438	0.256
oracle 0.1	0.095	0.183	0.212	0.245	0.136	0.117	0.104	0.262	0.156
canonical ordering	0.652	0.969	0.733	0.909	-0.055	0.735	0.171	N/A	N/A

Table 1: Numerical results for Baselines and Oracular measures

can be directly optimized to improve the generalization. For example, ℓ_2 regularization on the parameters of a model can be seen as imposing an isotropic Gaussian prior over the parameters in maximum a posteriori estimation. We choose several representative norms or measures based on norms and compute our correlation coefficient between the measures and the generalization gap of the model.

We study the following measures and their variants (Table 2): *spectral bound*, *Frobenius distance from initialization*, ℓ_2 *Frobenius norm of the parameters*, *Fisher-Rao metric* and *path norm*.

	batchsize	dropout	learning_rate	depth	optimizer	weight_decay	width	overall τ	Ψ
Frob distance 36	-0.317	-0.833	-0.718	0.526	-0.214	-0.669	-0.166	-0.263	-0.341
Spectral orig 22	-0.355	-0.745	-0.683	-0.909	-0.150	-0.047	-0.256	-0.534	-0.449
Parameter norm 38	0.236	-0.516	0.174	0.330	**0.187**	0.124	-0.170	0.073	0.052
Path norm 40	0.252	**0.270**	0.049	**0.934**	0.153	0.338	**0.178**	**0.373**	**0.311**
Fisher-Rao 41	**0.396**	0.147	**0.240**	-0.553	0.120	**0.551**	0.177	0.078	0.154
oracle 0.02	0.380	0.657	0.536	0.717	0.374	0.388	0.360	0.714	0.487

Table 2: Numerical results for selected Norm/Margin-based Measures

Spectral bound: The most surprising observation here is that the spectral complexity is strongly negatively correlated with generalization, and negatively correlated with changes within every hyper-parameters category. Most notably, it has strong negative correlation with the depth of the network, which may suggest that the largest singular values are not sufficient to capture the capacity of the model. To better understand the reason behind this observation, we investigate using different components of the spectral complexity as the measure. An interesting observation is that the Frobenius distance to initialization is negatively correlated, but the Frobenius norm of the parameters is slightly positively correlated with generalization, which contradicts some theories suggesting solutions closer to initialization should generalize better. A tempting hypothesis is that weight decay favors solution closer to the origin, but we did an ablation study on only models with 0 weight decay and found that the distance from initialization still correlates negatively with generalization.

These observations correspond to choosing different reference matrices $\mathbf{W}_i^0$ for the bound: the distance corresponds to using the initialization as the reference matrices while the Frobenius norm of the parameters corresponds to using the origin as the reference. Since the Frobenius norm of the parameters shows better correlation, we use zero reference matrices in the spectral bound. This improved both τ and Ψ, albeit still negative. In addition, we extensively investigate the effect of different terms of the Spectral bound to isolate the effect; however, the results do not improve. These experiments can be found in the Appendix C.2.

Path norm: While path-norm is a proper norm in the function space but not in parameter space, we observe that it is positively correlated with generalization in all hyper-parameter types and achieves comparable τ (0.373) and Ψ (0.311).

Fisher-Rao metric: The Fisher-Rao metric is a lower bound (Liang et al., 2017) on the path norm that has been recently shown to capture generalization. We observed that it overall shows worse correlation than the path norm; in particular, it is negatively correlated ($\tau = -0.553$) with the depth of the network while contrast with path norm which properly captures the effect of depth on generalization. A more interesting observation is that the Fisher-Rao metric achieves a positive $\Psi = 0.154$ but its $\tau = 0.078$ is essentially at chance.

This may suggest that the metric can capture a single hyper-parameter change but is not able to capture the interactions between different hyper-parameter categories.

Effect of Randomness: Dropout and batch size (first 2 columns of Table 2) directly introduce randomness into the training dynamic. For batch size, we observed that the Frobenius displacement and spectral complexity both correlate negatively with the changes in batch size while the Frobenius norm of the parameters correlates positively with generalization. On the other hand, when changes happen to the magnitude dropout probability, we observed that all of the proper norms are negatively correlated with the generalization changes. Since increasing dropout usually reduces the generalization gap, this implies that increasing the dropout probability may be at least partially responsible for the growth in these norms. This is unexpected since increasing norm in principle implies higher model capacity which is usually more prone to overfitting.

8 Success of Sharpness-based Measures

A natural category of generalization measures is centered around the concept of "***sharpness***" of the local minima, capturing the sensitivity of the empirical risk (i.e. the loss over the entire training set) to perturbations in model parameters. Such notion of stability under perturbation is captured elegantly by the PAC-Bayesian framework (McAllester, 1999) which has provided promising insights for studying generalization of deep neural networks (Dziugaite & Roy, 2017; Neyshabur et al., 2017; 2018a). In this sections, we investigate PAC-Bayesian generalization bounds and several of its variants which rely on different priors and different notions of sharpness (Table 3).

In order to evaluate a PAC-Bayesian bound, one needs to come up with a prior distribution over the parameters that is chosen in advance before observing the training set. Then, given any posterior distribution on the parameters which could depend on the training set, a PAC-Bayesian bound (Theorem 42) states that the expected generalization error of the parameters generated from the posterior can be bounded by the KL-divergence of the prior and posterior. The posterior distribution can be seen as adding perturbation on final parameters. Dziugaite & Roy (2017) shows contrary to other generalization bounds, it is possible to find to calculate non-vacuous PAC-Bayesian bounds by optimizing the bound over a large set of Gaussian posteriors. Neyshabur et al. (2017) demonstrates that PAC-Bayesian bounds where prior and posterior are isotropic Gaussian distributions are good measure of generalization on small scale experiments (e.q. 43).

PAC-Bayesian framework captures sharpness in the expected sense since we add randomly generated perturbations to the parameters. Another possible notion of sharpness is the worst-case sharpness where we search for the direction that changes the loss the most. This is motivated by (Keskar et al., 2016) where they observe that this notion would correlate to generalization in the case of different batch sizes. We can use PAC-Bayesian framework to give generalization bounds for this worst-case perturbations as well. We refer to this worst case bound as the sharpness bound (eq. 46). The main component in both PAC-Bayes and worst-case sharpness bounds is the ratio of norm of parameters to the magnitude of the perturbation where the magnitude is chosen to be the largest number so that the training error of the perturbed model is at most 0.1. While mathematically, the sharpness bound should always yield higher complexity than the Pac-Bayes bound, we observed that the former has higher correlation both in terms of τ and Ψ. In addition, we measured inverse of perturbation magnitude as a measure removing the norm in the numerator to compare it with the bound and did not observe a significant difference.

8.1 Magnitude-aware Perturbation Bounds

Perturbing the parameters without taking their magnitude into account can cause many of them to switch signs. Therefore, one cannot apply large perturbations to the model without changing the loss significantly. One possible modification to improve the perturbations is to choose the perturbation magnitude based on the magnitude of the parameter. In that case, it is guaranteed that if the magnitude of perturbation is less than magnitude of the

	batchsize	dropout	learning_rate	depth	optimizer	weight_decay	width	overall τ	Ψ
sharpness 48	0.542	-0.359	0.716	0.816	0.297	0.591	0.185	0.400	0.398
pacbayes 45	0.526	-0.076	0.705	0.546	0.341	0.564	-0.086	0.293	0.360
sharpness mag 56	0.401	-0.514	0.321	-0.909	0.181	0.281	-0.171	-0.158	-0.059
pacbayes mag 53	0.532	-0.480	0.508	**0.902**	0.188	0.155	0.186	0.410	0.284
$1/\sigma$ sharpness 50	0.532	-0.326	0.711	0.776	0.296	0.592	0.263	0.399	0.406
$1/\sigma$ pacbayes 49	0.501	-0.033	0.744	0.200	**0.346**	0.609	0.056	0.303	0.346
$1/\alpha'$ sharpness mag 58	**0.570**	**0.148**	**0.762**	0.824	0.297	**0.741**	**0.269**	**0.484**	**0.516**
$1/\sigma'$ pacbayes mag 57	0.490	-0.215	0.505	0.896	0.186	0.147	0.195	0.365	0.315
oracle 0.02	0.380	0.657	0.536	0.717	0.374	0.388	0.360	0.714	0.487

Table 3: Numerical results for selected sharpness-based Measures; all the measure use the origin as the reference and `mag` refers to magnitude-aware version of the measure.

parameter, then the sign of the parameter does not change. Following Keskar et al. (2016), we pick the magnitude of the perturbation with respect to the magnitude of parameters. We formalize this notion of importance based on magnitude and derive two alternative generalization bounds for expected sharpness (eq. 51) and worst case sharpness (eq. 54) that include the magnitude of the parameters into the prior. Formally, we design α' and σ' to be the ratio of parameter magnitude to the perturbation magnitude in sharpness and PAC-Bayes bounds respectively. While this change did not improve upon the original PAC-Bayesian measures, we observed that simply looking at $1/\alpha'$ has surprising predictive power in terms of the generalization which surpasses the performance of oracle 0.02. This measure is very close to what was originally suggested in Keskar et al. (2016).

8.2 Finding σ

In case of models with extremely small loss, the perturbed loss should roughly increase monotonically w.r.t the perturbation scale. Leveraging this observation, we design algorithms for computing the perturbation scale σ such that the first term on the RHS is as close to a fixed value as possible for all models. In our experiments, we choose the deviation to be 0.1 which translates to 10% training error. These search algorithms are paramount to compare measures between different models. We provide the detailed algorithms in the Appendix D. To improve upon our algorithms, one could try a computational approach similar to Dziugaite & Roy (2017) to obtain a numerically better bound which may result in stronger correlation; however, due to practical computational constraints, we could not do so for the large number of models we consider.

9 Potential of Optimization-based Measures

Optimization is an indispensable component of deep learning. Numerous optimizers have been proposed for more stable training and faster convergence. How the optimization scheme and speed of optimization influence generalization of a model has been a topic of contention among the deep learning community (Merity et al., 2017; Hardt et al., 2015). We study 3 representative optimizers `Momentum` `SGD`, `Adam`, and `RMSProp` with different initial learning rates in our experiments to thoroughly evaluate this phenomenon. We also consider other optimization related measures that are believed to correlate with generalization. These include (Table 4):

1. Number of iterations required to reach cross-entropy equals 0.1
2. Number of iterations required going from cross-entropy equals 0.1 to cross-entropy equals 0.01
3. Variance of the gradients after only seeing the entire dataset once (1 epoch)
4. Variance of the gradients when the cross-entropy is approximately 0.01

The number of iterations roughly characterizes the speed of optimization, which has been argued to correlate with generalization. For the models we consider, we observed that the initial phase (step to reach cross-entropy value of 0.1) of the optimization is negatively correlated with the speed of optimization for both τ and Ψ. This would suggest that the difficulty of optimization during the initial phase optimization benefits the final generalization. On

	batchsize	dropout	learning_rate	depth	optimizer	weight_decay	width	overall τ	Ψ
step to 0.1 59	-0.664	-0.861	-0.255	**0.440**	-0.030	-0.628	0.043	-0.264	-0.279
step 0.1 to 0.01 60	-0.151	-0.069	-0.014	0.114	0.072	-0.046	-0.021	-0.088	-0.016
grad noise 1 epoch 61	0.071	**0.378**	0.376	-0.517	0.121	0.221	0.037	0.070	0.098
grad noise final 62	**0.452**	0.119	**0.427**	0.141	**0.245**	**0.432**	**0.230**	**0.311**	**0.292**
oracle 0.02	0.380	0.657	0.536	0.717	0.374	0.388	0.360	0.714	0.487

Table 4: Optimization-Based Measures

the other hand, the speed of optimization going from cross-entropy 0.1 to cross-entropy 0.01 does not seem to be correlated with the generalization of the final solution. On the other hand, the speed of optimization is not an explicit capacity measure so either positive or negative correlation could potentially be informative.

On the other hand, the variance of the gradients characterizes the smoothness of the loss surface. Towards the end of the training, the variance of the gradients also captures a particular type of "flatness" of the local minima. This measure is surprisingly predictive of the generalization both in terms of τ and Ψ, and more importantly, is positively correlated across every category of hyper-parameter. To the best of our knowledge, this is the first time this phenomenon has been observed. The connection between variance of the gradient and generalization is perhaps natural since much of the recent advancement in deep learning such as residual network (He et al., 2016) or batch normalization have enabled using larger learning rates to train neural networks. Stability with higher learning rates implies smaller noises in the minibatch gradient. We hope that our work encourages future works in other possible measures based on optimization and during training.

10 Conclusion

We conducted large scale experiments to test the correlation of different measures with the generalization of deep models and propose a framework to better disentangle the cause of correlation from spurious correlation. We confirmed the effectiveness of the PAC-Bayesian bounds through our experiments and corroborate it as a promising direction for cracking the generalization puzzle. Further, we provide an extension to existing PAC-Bayesian bounds that consider the importance of each parameter. We also found that several measures related to optimization are surprisingly predictive of generalization and worthy of further investigation. On the other hand, several surprising failures about the norm-based measures were uncovered. In particular, we found that regularization that introduces randomness into the optimization can increase various norm of the models and spectral complexity related norm-based measures are unable to capture generalization – in fact, most of them are negatively correlated. Our experiments demonstrate that the study of generalization measure can be misleading when the number of models studied is small and the metric of quantifying the relationship is not carefully chosen. We hope this work will incentivize more rigorous treatment of generalization measures in future work.

To the best of our knowledge, this work is one of the most comprehensive study of generalization to date, but there are a few short-comings. Due to computational constraints, we were only able to study 7 most common hyperparameter categories and relatively small architectures, which do not reflect the models used in production. Indeed, if more hyperparameters are considered, one could expect to better capture the causal relationship. We also only studied models trained on a single dataset (CIFAR-10), only classification models and only convolutional networks. Future work will aim to address these limitations.

References

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.

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.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019. URL http://jmlr.org/papers/v20/17-612.html.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1019–1028. JMLR. org, 2017.

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.

Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 842–852. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7364-large-margin-deep-networks-for-classification.pdf.

Jianxi Gao, Sergey V Buldyrev, Shlomo Havlin, and H Eugene Stanley. Robustness of a network of networks. *Physical Review Letters*, 107(19):195701, 2011.

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. *arXiv preprint arXiv:1509.01240*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*, 2018.

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.

Aryeh Kontorovich. Dudley-pollard packing theorem. http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/, 2016.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

Xingguo Li, Junwei Lu, Zhaoran Wang, Jarvis Haupt, and Tuo Zhao. On tighter generalization bound for deep neural networks: Cnns, resnets, and beyond. *arXiv preprint arXiv:1806.05159*, 2018.

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

Philip M Long and Hanie Sedghi. Size-free generalization bounds for convolutional neural networks. *arXiv preprint arXiv:1905.12600*, 2019.

David A McAllester. Pac-bayesian model averaging. In *COLT*, volume 99, pp. 164–170. Citeseer, 1999.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. Foundations of machine learning. adaptive computation and machine learning. *MIT Press*, 31:32, 2012.

Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019.

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

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401, 2015b.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *International Conference on Learning Representations*, 2018a.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018b.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Konstantinos Pitas, Mike Davies, and Pierre Vandergheynst. Pac-bayesian margin bounds for convolutional neural networks. *arXiv preprint arXiv:1801.00171*, 2017.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019.

Hanie Sedghi, Vineet Gupta, and Philip M. Long. The singular values of convolutional layers. *CoRR*, abs/1805.10408, 2018. URL http://arxiv.org/abs/1805.10408.

Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Theory of probability and its applications*, pp. 11–30. Springer, 1971.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4148–4158, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, and Yoram Singer. Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698*, 2019.

A Experiments

A.1 More training details

During our experiments, we found that Batch Normalization (Ioffe & Szegedy, 2015) is crucial to reliably reach a low cross-entropy value for all models; since normalization is a indispensable components of modern neural networks, we decide to use batch normalization in all of our models. We *remove* batch normalization before computing any measure by fusing the γ, β and moving statistics with the convolution operator that precedes the normalization. This is important as Dinh et al. (2017) showed that common generalization measures such as sharpness can be easily manipulated with re-parameterization. We also discovered that the models trained with data augmentation often cannot fit the data (i.e. reach cross-entropy 0.01) completely. Since a model with data augmentation tends to consistently generalize better than the models without data augmentation, measure that reflects the training error (i.e. value of cross-entropy) will easily predict the ranking between two models even though it has only learned that one model uses data augmentation (see the thought experiments from the previous section). While certain hyperparameter configuration can reach cross-entropy of 0.01 even with data augmentation, it greatly limits the space of models that we can study. Hence, we make the design choice to not include data augmentation in the models of this study. Note that from a theoretical perspective, data augmentation is also challenging to analyze since the training samples generated from the procedure are no longer identical and independently distributed. All values for all the measures we computed over these models can be found in Table 5 in Appendix A.5.

A.2 The choice of stopping criterion

The choice of stopping criterion is very essential and could completely change the evaluation and the resulting conclusions. In our experiments we noticed that if we pick the stopping criterion based on number of iterations or number of epochs, then since some models optimize faster than others, they end up fitting the training data more and in that case the cross-entropy itself can be very predictive of generalization. To make it harder to distinguish models based on their training performance, it makes more sense to choose the stopping criterion based on the training error or training loss. We noticed that as expected, models with the same cross-entropy usually have very similar training error so that suggests that this choice is not very important. However, during the optimization the training error behavior is noisier than cross-entropy and moreover, after the training error reaches zero, it cannot distinguish models while the cross-entropy is still meaningful after fitting the data. Therefore, we decided to use cross-entropy as the stopping criterion.

A.3 All Model Specification

As mentioned in the main text, the models we use resemble Network-in-Network (Gao et al., 2011) which is a class of more parameter efficient convolution neural networks that achieve reasonably competitive performance on modern image classification benchmarks. The model consists blocks of modules that have 1 3×3 convolution with stride 2 followed by 2 1×1 convolution with stride 1. We refer to this single module as a `NiN-block` and construct models of different size by stacking NiN-block. For simplicity, all NiN-block have the same number of output channels c_{out}. Dropout is applied at the end of every NiN-block. At the end of the model, there is a 1×1 convolution reducing the channel number to the class number (i.e. 10 for CIFAR-10) followed by a global average pooling to produce the output logits.

For width, we choose from c_{out} from 3 options: $\{2 \times 96, 4 \times 96, 8 \times 96\}$.

For depth, we choose from 3 options: $\{2 \times \text{NiNblock}, 4 \times \text{NiNblock}, 8 \times \text{NiNblock}\}$

For dropout, we choose from 3 options: $\{0.0, 0.25, 0.5\}$

For batch size, we choose from: $\{32, 64, 128\}$

Since each optimizer may require different learning rate and in some cases, different regularization, we fine-tuned the hyper-parameters for each optimizer while keeping 3 options for every hyper-parameter choices[6].

Momentum SGD: We choose momentum of 0.9 and choose the initial learning rate η from $\{0.1, 0.032, 0.01\}$ and regularization coefficient λ from $\{0.0, 0.0001, 0.0005\}$. The learning rate decay schedule is $\times 0.1$ at iterations $[60000, 90000]$.

Adam: We choose initial learning rate η from $\{0.001, 3.2e-4, 1e-4\}$, $\epsilon = 1e-3$ and regularization coefficient λ from $\{0.0, 0.0001, 0.0005\}$. The learning rate decay schedule is $\times 0.1$ at iterations $[60000, 90000]$.

RMSProp: We choose initial learning rate η from $\{0.001, 3.2e-4, 1e-4\}$ and regularization coefficient λ from $\{0.0, 0.0001, 0.0003\}$. The learning rate decay schedule is $\times 0.1$ at iterations $[60000, 90000]$.

A.4 Canonical Measures

Based on empirical observations made by the community as a whole, the canonical ordering we give to each of the hyper-parameter categories are as follows:

1. Batchsize: smaller batchsize leads to smaller generalization gap
2. Depth: deeper network leads to smaller generalization gap
3. Width: wider network leads to smaller generalization gap
4. Dropout: The higher the dropout (≤ 0.5) the smaller the generalization gap
5. Weight decay: The higher the weight decay (smaller than the maximum for each optimizer) the smaller the generalization gap
6. Learning rate: The higher the learning rate (smaller than the maximum for each optimizer) the smaller the generalization gap
7. Optimizer: Generalization gap of `Momentum SGD` $<$ Generalization gap of `Adam` $<$ Generalization gap of `RMSProp`

A.5 All results

Below we present all of the measures we computed and their respective τ and Ψ on the 2000+ models we trained and additional plots.

[6]While methods with adaptive methods generally require less tuning, in practice researchers have observed performance gains from tuning the initial learning rate and learning rate decay.

	ref	batchsize	dropout	learning_rate	depth	optimizer	weight_decay	width	overall τ	Ψ
vc dim	15	0.000	0.000	0.000	−0.909	0.000	0.000	−0.171	−0.251	−0.154
# params	16	0.000	0.000	0.000	−0.909	0.000	0.000	−0.171	−0.175	−0.154
sharpness	47	0.537	−0.523	0.449	0.826	0.221	0.233	−0.004	0.282	0.248
pacbayes	44	0.372	−0.457	0.042	0.644	0.179	−0.179	−0.142	0.064	0.066
sharpness-orig	48	0.542	−0.359	0.716	0.816	0.297	0.591	0.185	0.400	0.398
pacbayes-orig	45	0.526	−0.076	0.705	0.546	0.341	0.564	−0.086	0.293	0.360
frob-distance	36	−0.317	−0.833	−0.718	0.526	−0.214	−0.669	−0.166	−0.263	−0.341
spectral-init	21	0.231	−0.736	−0.177	−0.629	0.134	−0.292	−0.052	−0.403	−0.218
spectral-orig	22	0.246	−0.693	−0.076	−0.618	0.172	−0.238	−0.020	−0.349	−0.175
spectral-orig-main	24	0.246	−0.693	−0.076	−0.560	0.172	−0.238	0.015	−0.314	−0.230
fro/spec	29	0.316	−0.589	0.286	0.706	0.242	−0.181	0.110	0.123	0.127
prod-of-spec	28	−0.464	−0.724	−0.722	−0.909	−0.197	−0.142	−0.218	−0.559	−0.482
prod-of-spec/margin	27	−0.397	−0.754	−0.713	−0.909	−0.181	−0.147	−0.188	−0.566	−0.470
sum-of-spec	31	−0.464	−0.724	−0.722	0.909	−0.197	−0.142	−0.218	0.102	−0.223
sum-of-spec/margin	30	−0.395	−0.754	−0.714	0.909	−0.182	−0.149	−0.190	0.083	−0.211
spec-dist	37	−0.458	−0.838	−0.568	0.738	−0.319	−0.182	−0.171	−0.110	−0.257
prod-of-fro	33	0.440	−0.199	0.538	−0.909	0.321	0.731	−0.101	−0.297	0.117
prod-of-fro/margin	32	0.513	−0.291	0.579	−0.907	0.364	0.739	−0.088	−0.295	0.130
sum-of-fro	35	0.440	−0.199	0.538	0.913	0.321	0.731	−0.101	0.418	0.378
sum-of-fro/margin	34	0.513	−0.291	0.579	0.909	0.364	0.739	−0.088	0.407	0.389
1/margin	18	−0.312	0.593	−0.234	−0.758	−0.223	0.211	−0.125	−0.124	−0.121
neg-entropy	19	0.346	−0.529	0.251	0.632	0.220	−0.157	0.104	0.148	0.124
path-norm	40	0.252	0.270	0.049	0.934	0.153	0.338	0.178	0.373	0.311
path-norm/margin	39	0.363	0.017	0.148	0.922	0.230	0.280	0.173	0.374	0.305
param-norm	38	0.236	−0.516	0.174	0.330	0.187	0.124	−0.170	0.073	0.052
fisher-rao	41	0.396	0.147	0.240	−0.516	0.120	0.551	0.177	0.090	0.160
cross-entropy	17	0.440	−0.402	0.140	0.390	0.149	0.232	0.080	0.149	0.147
$1/\sigma$ pacbayes	49	0.501	−0.033	0.744	0.200	0.346	0.609	0.056	0.303	0.346
$1/\sigma$ sharpness	50	0.532	−0.326	0.711	0.776	0.296	0.592	0.263	0.399	0.406
num-step-0.1-to-0.01-loss	60	−0.151	−0.069	−0.014	0.114	0.072	−0.046	−0.021	−0.088	−0.016
num-step-to-0.1-loss	59	−0.664	−0.861	−0.255	0.440	−0.030	−0.628	0.043	−0.264	−0.279
$1/\alpha'$ sharpness mag	58	0.570	0.148	0.762	0.824	0.297	0.741	0.269	0.484	0.516
$1/\sigma'$ pacbayes mag	57	0.490	−0.215	0.505	0.896	0.186	0.147	0.195	0.365	0.315
pac-sharpness-mag-init	55	−0.293	−0.841	−0.698	−0.909	−0.240	−0.631	−0.171	−0.225	−0.541
pac-sharpness-mag-orig	56	0.401	−0.514	0.321	−0.909	0.181	0.281	−0.171	−0.158	−0.059
pacbayes-mag-init	52	0.425	−0.658	−0.035	0.874	0.099	−0.407	0.069	0.175	0.052
pacbayes-mag-orig	53	0.532	−0.480	0.508	0.902	0.188	0.155	0.186	0.410	0.284
grad-noise-final	62	0.452	0.119	0.427	0.141	0.245	0.432	0.230	0.311	0.292
grad-noise-epoch-1	61	0.071	0.378	0.376	−0.517	0.121	0.221	0.037	0.070	0.098
oracle 0.01		0.579	0.885	0.736	0.920	0.529	0.622	0.502	0.851	0.682
oracle 0.02		0.414	0.673	0.548	0.742	0.346	0.447	0.316	0.726	0.498
oracle 0.05		0.123	0.350	0.305	0.401	0.132	0.201	0.142	0.456	0.236
oracle 0.1		0.069	0.227	0.132	0.223	0.086	0.121	0.093	0.241	0.136
canonical ordering		−0.652	0.969	0.733	0.909	−0.055	0.735	0.171	0.005	0.402
canonical ordering depth		−0.032	0.001	0.033	−0.909	−0.061	−0.020	0.024	−0.363	−0.138

Table 5: All measures (rows), hyperparameters (columns) and the rank-correlation coefficients that we measured.

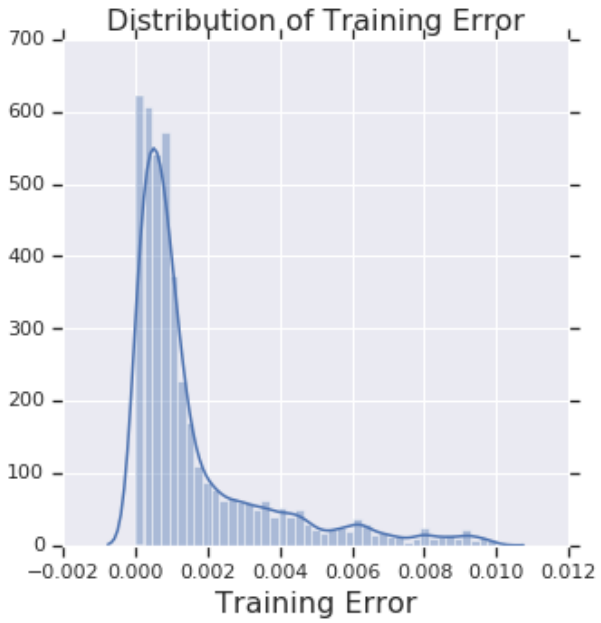

Figure 2: Distribution of training error on the trained models.

B Extended Notation

Given any margin value $\gamma \geq 0$, we define the margin loss L_γ as follows:

$$L_\gamma(f_{\mathbf{w}}) \triangleq \mathbb{E}_{(\mathbf{X},y)\sim\mathscr{D}}\left[I\big(f_{\mathbf{w}}(\mathbf{X})[y] \leq \gamma + \max_{j\neq y} f_{\mathbf{w}}(\mathbf{X})[j]\big)\right] \tag{6}$$

and $\hat{L}_\gamma$ is defined in an analogous manner on the training set. Further, for any vector $\mathbf{v}$, we denote by $\|\mathbf{v}\|_2$ the ℓ_2 norm of $\mathbf{v}$. For any tensor $\mathbf{W}$, let $\|\mathbf{W}\|_F \triangleq \|\text{vec}(\mathbf{W})\|$. We also denote $\|\mathbf{W}\|_2$ as the spectral norm of the tensor $\mathbf{W}$ when used with a convolution operator. For convolutional operators, we compute the true singular value with the method proposed by Sedghi et al. (2018) through FFT.

We denote a tensor as $\mathbf{A}$, vector as $\boldsymbol{a}$, and scalar as A or a. For any $1 \leq j \leq k$, consider a k-th order tensor $\mathbf{A}$ and a j-th order tensor $\mathbf{B}$ where dimensions of $\mathbf{B}$ match the last j dimensions of $\mathbf{A}$. We then define the product operator $\otimes_j$:

$$(\mathbf{A} \otimes_j \mathbf{B})_{i_1,\ldots,i_{k-j}} \triangleq \langle \mathbf{A}_{i_1,\ldots,i_{k-j}}, \mathbf{B}\rangle\,, \tag{7}$$

where $i_1,\ldots,i_{k-j}$ are indices. We also assume that the input images have dimension $n \times n$ and there are κ classes. Given the number of input channels c_{in}, number of output channels c_{out}, 2D square kernel with side length k, stride s, and padding p, we define the convolutional layer $\text{conv}_{\mathbf{W},s,p}$ as follows:

$$\text{conv}_{\mathbf{W},s,p}(\mathbf{X})_{i_1,i_2} \triangleq \mathbf{W}\otimes_3 \text{patch}_{s(i_1-1)+1,s(i_2-1)+1,k}\left(\text{pad}_p(\mathbf{X})\right) \qquad \forall 1 \leq i_1,i_2 \leq \lfloor\frac{n+2p-k}{s}\rfloor \tag{8}$$

where $\mathbf{W} \in \mathbb{R}^{c_{\text{out}}\times c_{\text{in}}\times k\times k}$ is the convolutional parameter tensor, $\text{patch}_{i,j,k}(\mathbf{Z})$ is a $k \times k$ patch of $\mathbf{Z}$ starting from the point (i,j), and pad_p is the padding operator which adds p zeros to top, bottom, left and right of $\mathbf{X}$:

$$\text{pad}_p(\mathbf{X})_{i_1,i_2,j} = \begin{cases}\mathbf{X}_{i_1,i_2} & p < i_1,i_2 \leq n+p \\ 0 & \text{otherwise}\end{cases}. \tag{9}$$

We also define the max-pooling operator $\text{pool}_{k,s,p}$ as follows:

$$\text{pool}_{k,s,p}(\mathbf{X})_{i_1,i_2,j} = \max(\text{patch}_{s(i_1-1)+1,s(i_2-1)+1}\left(\text{pad}_p(\mathbf{X}_{:,:,j})\right)) \qquad \forall 1 \leq i_1,i_2 \leq \lfloor\frac{n+2p-k}{s}\rfloor \tag{10}$$

We denote by $f_{\mathbf{W},s}$ a convolutional network such that $\mathbf{W}_i \in \mathbb{R}^{c_i\times c_{i-1}\times k_i\times k_i}$ is the convolution tensor and s_i is the convolutional stride at layer i. At Layer i, we assume the sequence of convolution, ReLU and max-pooling where the max pooling has kernel k'_i and stride s'_i. Lack of max-pooling in some layers can be achieved by setting $k'_i = s'_i = 1$. We consider classification tasks and denote the number of classes by κ.

C Complexity Measures

In this section, we look at different complexity measures. When a measure μ is based on a generalization bound, we chose it so that the following is true with probability 0.99 (we choose the failure probability δ to be 0.01):

$$L \leq \hat{L} + \sqrt{\frac{\mu}{m}} \tag{11}$$

We also definecolorredconsider? measures which do not provably bound the generalization error and evaluate those.

Note that in almost all cases, the canonical ordering given based on some "common" assumptions are positively correlated with the generalization in terms of both τ and Ψ; however, for optimizer, the correlation τ is close to 0. This implies that the choice of optimizer is only essentially uncorrelated with the generalization gap in the range of models we consider. This ordering helps validate many techniques used by the practioners.

C.1 VC-Dimension Based Measures

We start by restating the theorem in (Bartlett et al., 2019) which provides an upper bound on the VC-dimension of any piece-wise linear network.

Theorem 1 (Bartlett et al. (2019)) *Let $\mathcal{F}$ be the class of feed-forward networks with a fixed computation graph of depth d and ReLU activations. Let a_i and q_i be the number of activations and parameters in layer i. Then VC-dimension of $\mathcal{F}$ can be bounded as follows:*

$$VC(\mathcal{F}) \leq d + \left(\sum_{i=1}^{d}(d-i+1)q_i\right)\log_2\left(8e\sum_{i=1}^{d} i a_i \log_2\left(4e\sum_{j=1}^{d} j a_j\right)\right)$$

Theorem 2 *Given a convolutional network f, for any $\delta > 0$, with probability $1-\delta$ over the the training set:*

$$L \leq \hat{L} + 4000\sqrt{\frac{d\log_2(6dn)^3\sum_{i=1}^{d}k_i^2 c_i c_{i-1}}{m}} + \sqrt{\frac{\log(1/\delta)}{m}} \tag{12}$$

Proof We simplify the bound in Theorem 1 using a d' to refer to the depth instead of d:

$$\begin{aligned}
\mathrm{VC}(\mathcal{F}) &\leq d' + \left(\sum_{i=1}^{d'}(d-i+1)q_i\right)\log_2\left(8e\sum_{i=1}^{d'} i a_i \log_2\left(4e\sum_{j=1}^{d'} j a_j\right)\right) \\
&\leq d' + \left(\sum_{i=1}^{d'}(d'-i+1)q_i\right)\log_2\left(8e\sum_{i=1}^{d'} i a_i\right)^2 \\
&\leq d' + 2\log_2\left(8e\sum_{i=1}^{d'} i a_i\right)\sum_{i=1}^{d'}(d'-i+1)q_i \\
&\leq 3d'\log_2\left(8e\sum_{i=1}^{d'} i a_i\right)\sum_{i=1}^{d'} q_i
\end{aligned}$$

In order to extend the above bound to a convolutional network, we need to present a pooling layer with ReLU activations. First note that maximum of two inputs can be calculated using two layers with ReLU and linear activations as $\max(x_1, x_2) = x_1 + ReLU(x_2 - x_1)$. Now, since max-pooling at layer i has kernel sizes k_i', we need $\lceil 4\log_2(k_i')\rceil$ layers to present that but given that the kernel size of the max-pooling layer is at most size of the image, we have

$$\lceil 4\log_2(k_i')\rceil \leq \lceil 4\log_2(n^2)\rceil \leq \lceil 8\log_2(n)\rceil \leq 9\log_2(n)$$

Therefore, we have $d' \leq 9d\log_2(n)$. The number of activations in any of these layers is at most $n^2 c_i$ since there are at most n^2 pairs of neighbor pixels in an $n \times n$ image with c_i channels. We ignore strides when calculating the upper bound since it only reduces number of activations at a few layers and does not change the bound significantly. Using these

bounds ond', a_i and q_i the equivalent network, we can bound the VC dimension as follows:

$$\begin{aligned}\mathrm{VC}(\mathcal{F}) &\leq 27d\log_2(n)\log_2\left(8e(9d\log_2(n))^2n^2\right)(9\log_2(n))\sum_{i=1}^{d}k_i^2c_{i-1}(c_i+1)\\ &\leq 729d\log_2(n)^2\log_2(6dn)\sum_{i=1}^{d}k_i^2c_{i-1}(c_i+1)\\ &\leq 729d\log_2(6dn)^3\sum_{i=1}^{d}k_i^2c_{i-1}(c_i+1)\end{aligned}$$

For binary classifiers, generalization error can be in terms of Rademacher complexity (Mohri et al., 2012) which in turn can be bounded by $72\sqrt{\mathrm{VC}/m}$(Kontorovich, 2016). Therefore, we can get the following[7] generalization bound:

$$L \leq \hat{L} + 144\sqrt{\frac{VC(\mathcal{F})}{m}} + \sqrt{\frac{\log(1/\delta)}{m}} \tag{13}$$

For multi-class classification, the generalization error can be similarly bounded by Graph dimension which is an extension of VC-dimension. A simple approach get a bound on Graph dimension is to consider all pairs of classes as binary classification problem which bounds the graph dimension by $\kappa^2\ VC(\mathcal{F})$. There, putting everything together, we get the following generalization bound:

$$L \leq \hat{L} + 4000\kappa\sqrt{\frac{d\log_2(6dn)^3\sum_{i=1}^{d}k_i^2c_{i-1}(c_i+1)}{m}} + \sqrt{\frac{\log(1/\delta)}{m}} \tag{14}$$

Inspired by Theorem 2, we define the following VC-based measure for generalization:

$$\mu_{VC}(f_{\mathbf{w}}) = \left(4000\kappa\sqrt{d\log_2(6dn)^3\sum_{i=1}^{d}k_i^2c_{i-1}(c_i+1)} + \sqrt{\log(1/\delta)}\right)^2 \tag{15}$$

Since some of the dependencies in the above measure are probably proof artifacts, we also define another measure that is nothing but the number of parameters of the model:

$$\mu_{\text{param}} = \sum_{i=1}^{d}k_i^2c_{i-1}(c_i+1) \tag{16}$$

C.1.1 Measures on the output of the network

While measures that can be calculated only based on the output of the network cannot reveal complexity of the network, they can still be very informative for predicting generalization. Therefore, we define a few measures that can be calculated solely based on the output of the network.

We start by looking at the cross-entropy over the output. Even though we used a cross-entropy based stopping criterion, the cross-entropy of the final models is not exactly the same as the stopping criterion and it could be informative. Hence we define the following measure:

$$\mu_{\text{cross-entropy}} = \frac{1}{m}\sum_{i=1}^{m}\ell(f_{\mathbf{w}}(\mathbf{X}_i), y_i) \tag{17}$$

where ℓ is the cross entropy loss.

[7]The generalization gap is bounded by two times Rademacher Complexity, hence the constant 144.

Another useful and intuitive notion that appears in generalization bounds is margin. In all measures that involve margin γ, we set the margin γ to be the 10-th percentile of the margin values on the training set and therefore ensuring $\hat{L}_\gamma \leq 0.1$. Even though margin alone is not a sensible generalization measure and can be artificially increased by scaling up the magnitude of the weights, it could still reveal information about training dynamics and therefore be informative. We report the following measure based on the margin:

$$\mu_{1/\text{margin}}(f_{\mathbf{w}}) = \frac{1}{\gamma^2} \tag{18}$$

Finally, entropy of the output is another interesting measure and it has been shown that regularizing it can improve generalization in deep learning (Pereyra et al., 2017). With a fixed cross-entropy, increasing the entropy corresponds to distribute the uncertainty of the predictions equally among the wrong labels which is connected to label smoothing and increasing the margin. We define the following measure which is the negative entropy of the output of the network:

$$\mu_{\text{neg-entropy}}(f_{\mathbf{w}}) = \frac{1}{\gamma^2} \sum_{j=1}^{\kappa} p_i[j] \log(p_i[j]) \tag{19}$$

where $p_i[j]$ is the predicted probability of the class j for the input data $\mathbf{X}_i$.

C.2 Norm/Margin Based Measures

Several generalization bounds have been proved for neural networks using margin and norm notions. In this section, we go over several such measures. For fully connected networks, Bartlett & Mendelson (2002) have shown a bound based on product of $\ell_{1,\infty}$ norm of the layer weights times a 2^d factor where $\ell_{1,\infty}$ is the maximum over hidden units of the ℓ_2 norm of the incoming weights to the hidden unit. Neyshabur et al. (2015b) proved a bound based on product of Frobenius norms of the layer weights times a 2^d factor and Golowich et al. (2017) was able to improve the factor to $\sqrt{d}$. Bartlett et al. (2017) proved a bound based on product of spectral norm of the layer weights times sum over layers of ratio of Frobenius norm to spectral norm of the layer weights and Neyshabur et al. (2018a) showed a similar bound can be achieved in a simpler way using PAC-bayesian framework.

Spectral Norm Unfortunately, none of the above founds are directly applicable to convolutional networks. Pitas et al. (2017) built on Neyshabur et al. (2018a) and extended the bound on the spectral norm to convolutional networks. The bound is very similar to the one for fully connected networks by Bartlett et al. (2017). We next restate their generalization bound for convolutional networks including the constants.

Theorem 3 (Pitas et al. (2017)) *Let B an upper bound on the ℓ_2 norm of any point in the input domain. For any $B, \gamma, \delta > 0$, the following bound holds with probability $1-\delta$ over the training set:*

$$L \leq \hat{L}_\gamma + \sqrt{\frac{\left(84B \sum_{i=1}^{d} k_i \sqrt{c_i} + \sqrt{\ln(4n^2 d)}\right)^2 \prod_{i=1}^{d} \|\mathbf{W}_i\|_2^2 \sum_{j=1}^{d} \frac{\|\mathbf{W}_j - \mathbf{W}_j^0\|_F^2}{\|\mathbf{W}_j\|_2^2} + \ln(\frac{m}{\delta})}{\gamma^2 m}} \tag{20}$$

Inspired by the above theorem, we define the following spectral measure:

$$\mu_{\text{spec,init}}(f_{\mathbf{w}}) = \frac{\left(84B \sum_{i=1}^{d} k_i \sqrt{c_i} + \sqrt{\ln(4n^2 d)}\right)^2 \prod_{i=1}^{d} \|\mathbf{W}_i\|_2^2 \sum_{j=1}^{d} \frac{\|\mathbf{W}_j - \mathbf{W}_j^0\|_F^2}{\|\mathbf{W}_j\|_2^2} + \ln(\frac{m}{\delta})}{\gamma^2} \tag{21}$$

The generalization bound in Theorem 3 depends on reference tensors $\mathbf{W}_i^0$. We chose the initial tensor as the reference in the above measure but another reasonable choice is the

origin which gives the following measures:

$$\mu_{\text{spec-orig}}(f_{\mathbf{w}}) = \frac{\left(84B\sum_{i=1}^{d} k_i\sqrt{c_i} + \sqrt{\ln(4n^2d)}\right)^2 \prod_{i=1}^{d}\|\mathbf{W}_i\|_2^2 \sum_{j=1}^{d}\frac{\|\mathbf{W}_j\|_F^2}{\|\mathbf{W}_j\|_2^2} + \ln(\frac{m}{\delta})}{\gamma^2} \tag{22}$$

Since some of the terms in the generalization bounds might be proof artifacts, we also measure the main terms in the generalization bound:

$$\mu_{\text{spec-init-main}}(f_{\mathbf{w}}) = \frac{\prod_{i=1}^{d}\|\mathbf{W}_i\|_2^2 \sum_{j=1}^{d}\frac{\left\|\mathbf{W}_j-\mathbf{W}_j^0\right\|_F^2}{\|\mathbf{W}_j\|_2^2}}{\gamma^2} \tag{23}$$

$$\mu_{\text{spec-orig-main}}(f_{\mathbf{w}}) = \frac{\prod_{i=1}^{d}\|\mathbf{W}_i\|_2^2 \sum_{j=1}^{d}\frac{\|\mathbf{W}_j\|_F^2}{\|\mathbf{W}_j\|_2^2}}{\gamma^2} \tag{24}$$

We further look at the main two terms in the bound separately to be able to differentiate their contributions.

$$\mu_{\text{spec-init-main}}(f_{\mathbf{w}}) = \frac{\prod_{i=1}^{d}\|\mathbf{W}_i\|_2^2 \sum_{j=1}^{d}\frac{\left\|\mathbf{W}_j-\mathbf{W}_j^0\right\|_F^2}{\|\mathbf{W}_j\|_2^2}}{\gamma^2} \tag{25}$$

$$\mu_{\text{spec-orig-main}}(f_{\mathbf{w}}) = \frac{\prod_{i=1}^{d}\|\mathbf{W}_i\|_2^2 \sum_{j=1}^{d}\frac{\|\mathbf{W}_j\|_F^2}{\|\mathbf{W}_j\|_2^2}}{\gamma^2} \tag{26}$$

$$\mu_{\text{prod-of-spec/margin}}(f_{\mathbf{w}}) = \frac{\prod_{i=1}^{d}\|\mathbf{W}_i\|_2^2}{\gamma^2} \tag{27}$$

$$\mu_{\text{prod-of-spec}}(f_{\mathbf{w}}) = \prod_{i=1}^{d}\|\mathbf{W}_i\|_2^2 \tag{28}$$

$$\mu_{\text{fro/spec}}(f_{\mathbf{w}}) = \sum_{i=1}^{d}\frac{\|\mathbf{W}_i\|_F^2}{\|\mathbf{W}_i\|_2^2} \tag{29}$$

Finally, since product of spectral norms almost certainly increases with depth, we look at the following measure which is equal to the sum over squared spectral norms after rebalancing the layers to have the same spectral norms:

$$\mu_{\text{sum-of-spec/margin}}(f_{\mathbf{w}}) = d\left(\frac{\prod_{i=1}^{d}\|\mathbf{W}_i\|_2^2}{\gamma^2}\right)^{1/d} \tag{30}$$

$$\mu_{\text{sum-of-spec}}(f_{\mathbf{w}}) = d\left(\|\mathbf{W}_i\|_2^2\right)^{1/d} \tag{31}$$

Frobenius Norm The generalization bound given in Neyshabur et al. (2015b) is not directly applicable to convolutional networks. However, Since for each layer i, we have $\|\mathbf{W}_i\|_2 \leq k_i^2\|\mathbf{W}_i\|_F$ and therefore by Theorem 3, we can get an upper bound on the test error based on product of Frobenius norms. Therefore, we define the following measure based on the product of Frobenius norms:

$$\mu_{\text{prod-of-fro/margin}}(f_{\mathbf{w}}) = \frac{\prod_{i=1}^{d}\|\mathbf{W}_i\|_F^2}{\gamma^2} \tag{32}$$

$$\mu_{\text{prod-of-fro}}(f_{\mathbf{w}}) = \prod_{i=1}^{d}\|\mathbf{W}_i\|_F^2 \tag{33}$$

We also look at the following measure with correspond to sum of squared Frobenius norms of the layers after rebalancing them to have the same norm:

$$\mu_{\text{sum-of-fro/margin}}(f_{\mathbf{w}}) = d\left(\frac{\prod_{i=1}^{d}\|\mathbf{W}_i\|_F^2}{\gamma^2}\right)^{1/d} \tag{34}$$

$$\mu_{\text{sum-of-fro}}(f_{\mathbf{w}}) = d\left(\prod_{i=1}^{d}\|\mathbf{W}_i\|_F^2\right)^{1/d} \tag{35}$$

Finally, given recent evidence on the importance of distance to initialization (Dziugaite & Roy, 2017; Nagarajan & Kolter, 2019; Neyshabur et al., 2018b), we calculate the following measures:

$$\mu_{\text{frobenius-distance}}(f_{\mathbf{w}}) = \sum_{i=1}^{d}\left\|\mathbf{W}_i - \mathbf{W}_i^0\right\|_F^2 \tag{36}$$

$$\mu_{\text{dist-spec-init}}(f_{\mathbf{w}}) = \sum_{i=1}^{d}\left\|\mathbf{W}_i - \mathbf{W}_i^0\right\|_2^2 \tag{37}$$

In case when the reference matrix $\mathbf{W}_i^0 = 0$ for all weights, eq. 36 the Frobenius norm of the parameters which also correspond to distance from the origin:

$$\mu_{\text{param-norm}}(f_{\mathbf{w}}) = \sum_{i=1}^{d}\|\mathbf{W}_i\|_F^2 \tag{38}$$

Path-norm Path-norm was introduced in Neyshabur et al. (2015b) as an scale invariant complexity measure for generalization and is shown to be a useful geometry for optimization Neyshabur et al. (2015a). To calculate path-norm, we square the parameters of the network, do a forward pass on an all-ones input and then take square root of sum of the network outputs. We define the following measures based on the path-norm:

$$\mu_{\text{path-norm/margin}}(f_{\mathbf{w}}) = \frac{\sum_i f_{\mathbf{w}^2}(\mathbf{1})[\mathbf{i}]}{\gamma^2} \tag{39}$$

$$\mu_{\text{path-norm}}(f_{\mathbf{w}}) = \sum_i f_{\mathbf{w}^2}(\mathbf{1}) \tag{40}$$

where $\mathbf{w}^2 = \mathbf{w} \circ \mathbf{w}$ is the element-wise square operation on the parameters.

Fisher-Rao Norm Fisher-Rao metric was introduced in Liang et al. (2017) as a complexity measure for neural networks. Liang et al. (2017) showed that Fisher-Rao norm is a lower bound on the path-norm and it correlates in some cases. We define a measure based on the Fisher-Rao matric of the network:

$$\mu_{\text{Fisher-Rao}}(f_{\mathbf{w}}) = \frac{(d+1)^2}{m}\sum_{i=1}^{m}\langle \mathbf{w}, \nabla_{\mathbf{w}}\ell(f_{\mathbf{w}}(\mathbf{X}_i)), y_i\rangle^2 \tag{41}$$

where ℓ is the cross-entropy loss.

C.3 Flatness-based Measures

PAC-Bayesian framework (McAllester, 1999) allows us to study flatness of a solution and connect it to generalization. Given a prior P is is chosen before observing the training set and a posterior Q which is a distribution on the solutions of the learning algorithm (and hence depends on the training set), we can bound the expected generalization error of solutions generated from Q with high probability based on the KL divergence of P and Q. The next theorem states a simplified version of PAC-Bayesian bounds.

Theorem 4 *For any $\delta > 0$, distribution D, prior P, with probability $1-\delta$ over the training set, for any posterior Q the following bound holds:*

$$\mathbb{E}_{\mathbf{v}\sim Q}\left[L(f_{\mathbf{v}})\right] \leq \mathbb{E}_{\mathbf{w}\sim Q}\left[\hat{L}(f_{\mathbf{v}})\right] + \sqrt{\frac{\mathrm{KL}(Q||P) + \log\left(\frac{m}{\delta}\right)}{2(m-1)}} \tag{42}$$

If P and Q are Gaussian distributions with $P = \mathcal{N}(\mu_P, \Sigma_P)$ amd $Q = \mathcal{N}(\mu_Q, \Sigma_Q)$, then the KL-term can be written as follows:

$$\mathrm{KL}(\mathcal{N}(\mu_Q, \Sigma_Q)||\mathcal{N}(\mu_P, \Sigma_P)) = \frac{1}{2}\left[\mathrm{tr}\left(\Sigma_P^{-1}\Sigma_Q\right) + \left(\mu_Q - \mu_P\right)^\top \Sigma_P^{-1}\left(\mu_Q - \mu_P\right) - k + \ln(\frac{\det\Sigma_P}{\det\Sigma_Q})\right].$$

Setting $Q = \mathcal{N}(\mathbf{w}, \sigma^2 I)$ and $P = \mathcal{N}(\mathbf{w}^0, \sigma^2 I)$ similar to Neyshabur et al. (2017), the KL term will be simply $\frac{\|\mathbf{w}-\mathbf{w}^0\|_2^2}{2\sigma^2}$. However, since σ belongs to prior, if we search to find a value for σ, we need to adjust the bound to reflect that. Since we search over less than 20000 predefined values of σ in our experiments, we can use the union bound which changes the logarithmic term to $\log(20000m/\delta)$ and we get the following bound:

$$\mathbb{E}_{\mathbf{u}\sim\mathcal{N}(u,\sigma^2 I)}\left[L(f_{\mathbf{w}+\mathbf{u}})\right] \leq \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(u,\sigma^2 I)}\left[\hat{L}(f_{\mathbf{w}+\mathbf{u}})\right] + \sqrt{\frac{\frac{\|\mathbf{w}-\mathbf{w}^0\|_2^2}{4\sigma^2} + \log(\frac{m}{\sigma}) + 10}{m-1}} \tag{43}$$

Based on the above bound, we define the following measures using the origin and initialization as reference tensors:

$$\mu_{\text{pac-bayes-init}}(f_{\mathbf{w}}) = \frac{\left\|\mathbf{w}-\mathbf{w}^0\right\|_2^2}{4\sigma^2} + \log(\frac{m}{\sigma}) + 10 \tag{44}$$

$$\mu_{\text{pac-bayes-orig}}(f_{\mathbf{w}}) = \frac{\left\|\mathbf{w}\right\|_2^2}{4\sigma^2} + \log(\frac{m}{\delta}) + 10 \tag{45}$$

where σ is chosen to be the largest number such that $\mathbb{E}_{\mathbf{u}\sim\mathcal{N}(u,\sigma^2 I)}\left[\hat{L}(f_{\mathbf{w}+\mathbf{u}})\right] \leq 0.1$.

The above framework captures flatness in the expected sense since we add Gaussian perturbations to the parameters. Another notion of flatness is the worst-case flatness where we search for the direction that changes the loss the most. This is motivated by (Keskar et al., 2016) where they observe that this notion would correlate to generalization in the case of different batch sizes. We can use PAC-Bayesian framework to give generalization bounds for worst-case perturbations as well. The magnitude of a Gaussian variable with with variance σ^2 is at most $\sigma\sqrt{2\log(2/\delta)}$ with probability $1-\delta/2$. Applying a union bound on all parameters, we get that with probability $1-\delta/2$ the magnitude of the Gaussian noise is at most $\alpha = \sigma\sqrt{2\log(2\omega/\delta)}$ where ω is the number of parameters of the model. Therefore, we can get the following generalization bound:

$$\mathbb{E}_{\mathbf{u}\sim\mathcal{N}(u,\sigma^2 I)}\left[L(f_{\mathbf{w}+\mathbf{u}})\right] \leq \max_{|u_i|\leq\alpha} \hat{L}(f_{\mathbf{w}+\mathbf{u}}) + \sqrt{\frac{\frac{\|\mathbf{w}-\mathbf{w}^0\|_2^2\log(2\omega/\delta)}{2\alpha^2} + \log(\frac{2m}{\delta}) + 10}{m-1}} \tag{46}$$

Inspired by the above bound, we define the following measures:

$$\mu_{\text{sharpness-init}}(f_{\mathbf{w}}) = \frac{\left\|\mathbf{w}-\mathbf{w}^0\right\|_2^2\log(2\omega)}{4\alpha^2} + \log(\frac{m}{\sigma}) + 10 \tag{47}$$

$$\mu_{\text{sharpness-orig}}(f_{\mathbf{w}}) = \frac{\left\|\mathbf{w}\right\|_2^2\log(2\omega)}{4\alpha^2} + \log(\frac{m}{\delta}) + 10 \tag{48}$$

where α is chosen to be the largest number such that $\max_{|u_i|\leq\alpha}\hat{L}(f_{\mathbf{w}+\mathbf{u}}) \leq 0.1$.

To understand the importance of the flatness parameters σ and α, we also define the following measures:

$$\mu_{\text{pac-bayes-flatness}}(f_{\mathbf{w}}) = \frac{1}{\sigma^2} \tag{49}$$

$$\mu_{\text{sharpness-flatness}}(f_{\mathbf{w}}) = \frac{1}{\alpha^2} \tag{50}$$

where α and σ are computed as explained above.

Magnitude-aware Perturbation Bounds The magnitude of perturbation in (Keskar et al., 2016) was chosen so that for each parameter the ratio of magnitude of perturbation to the magnitude of the parameter is bounded by a constant α'[8]. Following a similar approach, we can choose the posterior for parameter i in PAC-Bayesian framework to be $\mathcal{N}(w_i, \sigma'^2|w_i|^2 + \epsilon^2)$. Now, substituting this in the Equation equation C.3 and solving for the prior $\mathcal{N}(\mathbf{w}^0, \sigma_P^2)$ that minimizes the KL term by setting the gradient with respect to σ_2^P to zero, KL can be written as follows:

$$
\begin{aligned}
2\mathrm{KL}(Q||P) &= \omega \log\left(\frac{\sigma'^2+1}{\omega}\left\|\mathbf{w}-\mathbf{w}^0\right\|_2^2 + \epsilon^2\right) - \sum_{i=1}^{\omega}\log\left(\sigma'^2|w_i - w_i^0|^2 + \epsilon^2\right) \\
&= \sum_{i=1}^{\omega}\log\left(\frac{\epsilon^2 + (\sigma'^2+1)\left\|\mathbf{w}-\mathbf{w}^0\right\|_2^2/\omega}{\epsilon^2 + \sigma'^2|w_i - w_i^0|^2}\right)
\end{aligned}
$$

Therefore, the generalization bound can be written as follows

$$
\mathbb{E}_{\mathbf{u}}\left[L(f_{\mathbf{w}+\mathbf{u}})\right] \leq \mathbb{E}_{\mathbf{u}}\left[\hat{L}(f_{\mathbf{w}+\mathbf{u}})\right] + \sqrt{\frac{\frac{1}{4}\sum_{i=1}^{\omega}\log\left(\frac{\epsilon^2+(\sigma'^2+1)\|\mathbf{w}-\mathbf{w}^0\|_2^2/\omega}{\epsilon^2+\sigma'^2|w_i-w_i^0|^2}\right) + \log(\frac{m}{\delta}) + 10}{m-1}} \tag{51}
$$

where $u_i \sim \mathcal{N}(0, \sigma'^2|w_i| + \epsilon^2)$, $\epsilon = 1e-3$ and σ' is chosen to be the largest number such that $\mathbb{E}_{\mathbf{u}}\left[\hat{L}(f_{\mathbf{w}+\mathbf{u}})\right] \leq 0.1$. We define the following measures based on the generalization bound:

$$
\mu_{\text{pac-bayes-mag-init}}(f_{\mathbf{w}}) = \frac{1}{4}\sum_{i=1}^{\omega}\log\left(\frac{\epsilon^2 + (\sigma'^2+1)\left\|\mathbf{w}-\mathbf{w}^0\right\|_2^2/\omega}{\epsilon^2 + \sigma'^2|w_i - w_i^0|^2}\right) + \log(\frac{m}{\delta}) + 10 \tag{52}
$$

$$
\mu_{\text{pac-bayes-mag-orig}}(f_{\mathbf{w}}) = \frac{1}{4}\sum_{i=1}^{\omega}\log\left(\frac{\epsilon^2 + (\sigma'^2+1)\left\|\mathbf{w}\right\|_2^2/\omega}{\epsilon^2 + \sigma'^2|w_i - w_i^0|^2}\right) + \log(\frac{m}{\delta}) + 10 \tag{53}
$$

We also follow similar arguments are before to get a similar bound on the worst-case sharpness:

$$
\mathbb{E}_{\mathbf{u}}\left[L(f_{\mathbf{w}+\mathbf{u}})\right] \leq \max_{|u_i|\leq\alpha'|w_i|+\epsilon}\hat{L}(f_{\mathbf{w}+\mathbf{u}}) + \sqrt{\frac{\frac{1}{4}\sum_{i=1}^{\omega}\log\left(\frac{\epsilon^2+(\alpha'^2+4\log(2\omega/\delta))\|\mathbf{w}-\mathbf{w}^0\|_2^2/\omega}{\epsilon^2+\alpha'^2|w_i-w_i^0|^2}\right) + \log(\frac{m}{\delta}) + 10}{m-1}} \tag{54}
$$

We look at the following measures based on the above bound:

$$
\mu_{\text{pac-sharpness-mag-init}}(f_{\mathbf{w}}) = \frac{1}{4}\sum_{i=1}^{\omega}\log\left(\frac{\epsilon^2 + (\alpha'^2+4\log(2\omega/\delta))\left\|\mathbf{w}-\mathbf{w}^0\right\|_2^2/\omega}{\epsilon^2 + \alpha'^2|w_i - w_i^0|^2}\right) + \log(\frac{m}{\delta}) + 10 \tag{55}
$$

$$
\mu_{\text{pac-sharpness-mag-orig}}(f_{\mathbf{w}}) = \frac{1}{4}\sum_{i=1}^{\omega}\log\left(\frac{\epsilon^2 + (\alpha'^2+4\log(2\omega/\delta))\left\|\mathbf{w}\right\|_2^2/\omega}{\epsilon^2 + \alpha'^2|w_i - w_i^0|^2}\right) + \log(\frac{m}{\delta}) + 10 \tag{56}
$$

Finally, we look at measures that are only based the sharpness values computed above:

$$
\mu_{\text{pac-bayes-mag-flat}}(f_{\mathbf{w}}) = \frac{1}{\sigma'^2} \tag{57}
$$

$$
\mu_{\text{sharpness-mag-flat}}(f_{\mathbf{w}}) = \frac{1}{\alpha'^2} \tag{58}
$$

where α and σ are computed as explained above.

[8]They actually used a slightly different version which is a combination of the two perturbation bounds we calculated here. Here, for more clarity, we decomposed it into two separate perturbation bounds.

C.4 Optimization-based Measures

There are mixed results about how the optimization speed is relevant to generalization. On one hand we know that adding Batch Normalization or using shortcuts in residual architectures help both optimization and generalization and Hardt et al. (2015) suggests that faster optimization results in better generalization. On the other hand, there are empirical results showing that adaptive optimization methods that are faster, usually generalize worse (Wilson et al., 2017). Here, we put these hypothesis into test by looking at the number of steps to achieve cross entropy 0.1 and the number of steps needed to go from cross-entropy 0.1 to 0.01:

$$\mu_{\text{\#steps-0.1-loss}}(f_{\mathbf{w}}) = \text{\#steps from initialization to 0.1 cross-entropy} \tag{59}$$

$$\mu_{\text{\#steps-0.1-0.01-loss}}(f_{\mathbf{w}}) = \text{\#steps from 0.1 to 0.01 cross-entropy} \tag{60}$$

The above measures tell us if the speed of optimization at early or late stages can be informative about generalization. We also define measures that look at the SGD gradient noise after the first epoch and at the end of training at cross-entropy 0.01 to test the gradient noise can be predictive of generalization:

$$\mu_{\text{grad-noise-epoch1}}(f_{\mathbf{w}}) = \text{Var}_{(\mathbf{X},y)\ S}\left(\nabla_{\mathbf{w}}\ell(f_{\mathbf{w}}(\mathbf{X}), y)\right) \tag{61}$$

$$\mu_{\text{grad-noise-final}}(f_{\mathbf{w}}) = \text{Var}_{(\mathbf{X},y)\ S}\left(\nabla_{\mathbf{w}}\ell(f^1_{\mathbf{w}}(\mathbf{X}), y)\right) \tag{62}$$

where $\mathbf{w}^1$ is the weight vector after the first epoch.

D Algorithms

We first lay out some common notations used in the pseudocode:

1. f: the architecture that takes parameter θ and input x and map to $f(x;\theta)$ which is the predicted label of x
2. θ: parameters
3. M: Some kind of iteration; M_1: binary search depth; M_2: Monte Carlo Estimation steps; M_3: Iteration for estimating the loss
4. $\mathscr{D} = \{(x_i, y_i)\}_{i=0}^n$ the dataset the model is trained on; $\mathcal{B}$ as a uniformly sampled minibatch from the dataset.

Both search algorithm relies on the assumption that the loss increases monotonically with the perturbation magnitude σ around the final weight. This assumption is quite mild and in reality holds across almost all the models in this study.

Algorithm 1 EstimateAccuracy

Inputs: model f, parameter θ, dataset $\mathscr{D}$, estimate iteration M
Initialize Accuracy = 0
for episode $i = 1$ **to** M **do**
  $\mathcal{B} \sim$ sample($\mathscr{D}$)
  Accuracy += $\frac{1}{|\mathcal{B}|}\sum_i \delta(y_i = f(\mathcal{B}_i;\theta))$
end for
return Accuracy/M

Note that for finding the sharpness σ, we use the cross entropy as the differentiable surrogate object instead of the 1-0 loss which is in general not differentiable. Using gradient ascent brings another additional challenge that is for a converged model, the local gradient signal is usually weak, making gradient ascent extremely inefficient. To speed up thie process, we add a uniform noise with range being $[-\sigma_{new}/N_{\mathbf{w}}, \sigma_{new}/N_{\mathbf{w}}]$ to lift the weight off the flat minima where $N_{\mathbf{w}}$ is the number of parameters. This empirical greatly accelerates the search.

Algorithm 2 Find σ for PAC-Bayesian Bound

Inputs: f, θ_0, model accuracy ℓ, target accuracy deviation d, Upper bound $\sigma_{\max}$, Lower bound $\sigma_{\min}$, M_1, M_2, M_3
Initialize
for episode $i = 1$ **to** M_1 **do**
  $\sigma_{new} = (\sigma_{\max} + \sigma_{\min})/2$
  $\hat{\ell} = 0$
  for step $j = 0$ **to** M_2 **do**
    $\theta \leftarrow \theta_0 + \mathcal{N}(0, \sigma_{new}^2 I)$
    $\hat{\ell} = \hat{\ell} + \text{EstimateAccuracy}(f, \theta_{new}, \mathscr{D}, M_3)$
  end for
  $\hat{\ell} = \hat{\ell}/M_2$
  $\hat{d} = |\ell - \hat{\ell}|$
  if $\hat{d} < \epsilon_d$ **or** $\sigma_{\max} - \sigma_{\min} < \epsilon_\sigma$ **then**
    return σ_{new}
  end if
  if $\hat{d} > d$ **then**
    $\sigma_{\max} = \sigma_{new}$
  else
    $\sigma_{\min} = \sigma_{new}$
  end if
end for

Algorithm 3 Find σ for Sharpness Bound

Inputs: f, θ_0, loss function $\mathcal{L}$, model accuracy ℓ, target accuracy deviation d, Upper bound $\sigma_{\max}$, Lower bound $\sigma_{\min}$, M_1, M_2, M_3, gradient steps M_4
Initialize
for episode $i = 1$ **to** M_1 **do**
  $\sigma_{new} = (\sigma_{\max} + \sigma_{\min})/2$
  $\hat{\ell} = \infty$
  for step $j = 0$ **to** M_2 **do**
    $\theta = \theta_0 + \mathcal{U}(\sigma_{new}/2)$
    for step $k = 0$ **to** M_4 **do**
      $\mathcal{B} \sim \text{sample}(\mathscr{D})$
      $\theta = \theta + \eta \nabla_\theta \ell(f, \mathcal{B}, \theta)$
      if $||\theta|| > \sigma_{new}$ **then**
        $\theta = \sigma_{new} \cdot \frac{\theta}{||\theta||}$
      end if
    end for
    $\hat{\ell} = \min(\hat{\ell}, \text{EstimateAccuracy}(f, \theta_{new}, \mathscr{D}, M_3))$
  end for
  $\hat{d} = |\ell - \hat{\ell}|$
  if $\hat{d} < \epsilon_d$ **or** $\sigma_{\max} - \sigma_{\min} < \epsilon_\sigma$ **then**
    return σ_{new}
  end if
  if $\hat{d} > d$ **then**
    $\sigma_{\max} = \sigma_{new}$
  else
    $\sigma_{\min} = \sigma_{new}$
  end if
end for

Further, for magnitude aware version of the bounds, the overall algorithm stays the same with the exception that now covariance matrices at line 7 of Algorithm 2 become as diagonal matrix containing w_i^2 on the diagonal; similarly, for line 12 of Algorithm 3, the weight

clipping of each w_i is conditioned on $\sigma_{new}|w_i|$, i.e. clipped to $[-\sigma_{new}|w_i|, \sigma_{new}|w_i|]$. Here w_i denotes the i^{th} parameter of flattened $\mathbf{w}$.