Understanding Generalization Through Visualizations

W. Ronny Huang
University of Maryland
wrhuang@umd.edu

Zeyad Emam
University of Maryland
zeyad@math.umd.edu

Micah Goldblum
University of Maryland
goldblum@math.umd.edu

Liam Fowl
University of Maryland
lfowl@math.umd.edu

Justin K. Terry
University of Maryland
jkterry@umd.edu

Furong Huang
University of Maryland
furongh@cs.umd.edu

Tom Goldstein
University of Maryland
tomg@cs.umd.edu

Abstract

The power of neural networks lies in their ability to generalize to unseen data, yet the underlying reasons for this phenomenon remain elusive. Numerous rigorous attempts have been made to explain generalization, but available bounds are still quite loose, and analysis does not always lead to true understanding. The goal of this work is to make generalization more intuitive. Using visualization methods, we discuss the mystery of generalization, the geometry of loss landscapes, and how the curse (or, rather, the blessing) of dimensionality causes optimizers to settle into minima that generalize well.

1 Introduction

Neural networks are a powerful tool for solving classification problems. The power of these models is due in part to their expressiveness; they have many parameters that can be efficiently optimized to fit nearly any finite training set. However, the real power of neural network models comes from their ability to *generalize;* they often make accurate predictions on test data that were not seen during training, provided the test data is sampled from the same distribution as the training data.

The generalization ability of neural networks is seemingly at odds with their expressiveness. Neural network training algorithms work by minimizing a loss function that measures model performance using only training data. Because of their flexibility, it is possible to find parameter configurations for neural networks that perfectly fit the training data and minimize the loss function while making mostly incorrect predictions on test data. Miraculously, commonly used optimizers reliably avoid such "bad" minima of the loss function, and succeed at finding "good" minima that generalize well.

Our goal here is to make generalization a more widely accessible topic for practitioners, and also to develop an *intuitive* understanding of neural network generalization using a scientific/experimental approach rather than analysis. Experimental studies are unusual in the field of generalization, and we acknowledge that experimental results do not come with the certainty of theorems. However, experimental studies enable us to observe generalization phenomena and validate hypothesis using realistic architectures and complex datasets that cannot be investigated with currently available theory.

We begin with some experiments to demonstrate why generalization is puzzling, and how over-parameterization impacts model behavior. Then, we explore how the "flatness" of minima correlates

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

with generalization, and build intuition for *why* this correlation exists. We explore how the high dimensionality of parameter spaces biases optimizers towards landing in flat minima that generalize well. Finally, we present some counterfactual experiments to validate the intuition we develop.

2 Background: Why is generalization so puzzling?

Neural networks define a highly expressive model class. In fact, given enough parameters, a neural network can approximate virtually any function (Cybenko, 1989). But just because neural nets have the power to *represent* any function does not mean they have the power to *learn* any function from a finite amount of training data.

Neural network classifiers are trained by minimizing a loss function that measures model performance using only training data. A standard classification loss has the form

$$L(\theta) = \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} -\log p_\theta(x, y), \tag{1}$$

where $p_\theta(x, y)$ is the probability that data sample x lies in class y according to a neural net with parameters θ, and $\mathcal{D}_t$ is the training dataset of size $|\mathcal{D}_t|$. This loss is near zero when a model with parameters θ accurately classifies the training data. Over-parameterized neural networks (i.e., those with more parameters than training data) can represent arbitrary, even random, labeling functions on large datasets (Zhang et al., 2016). As a result, an optimizer can reliably fit an over-parameterized network to training data and achieve near zero loss (Laurent and Brecht, 2018; Kawaguchi, 2016). However, this comes with no guarantee of generalization to unseen test data.

We illustrate the difference between model fitting and generalization with an experiment. The CIFAR-10 training dataset contains 50,000 small images. We train two over-parameterized models on this dataset. The first is a neural network (ResNet-18) with 269,722 parameters (nearly 6× the number of training images). The second is a linear model with a feature set that includes pixel intensities as well as pair-wise products of pixels intensities.[1] This linear model has $298,369$ parameters, which is comparable to the neural network, and both are trained using SGD. On the left of Figure **??**, we see that over-parameterization causes both models to achieve perfect accuracy on training data. But the linear model achieves only 49% test accuracy, while ResNet-18 achieves 92%.

The excellent performance of the neural network model raises several questions. Do bad minima exist at all? Maybe deep networks generalize because bad minima are rare and lie far away from the region of parameter space where initialization takes place? Furthermore, if bad minima are prevalent in the loss landscape, what prevents optimizers from finding them? In Section 3 we will present strategies for finding bad minima, and use it to study these questions.

[1]For computing the pair-wise pixel intensity products, images are first downsampled by a factor of 2.

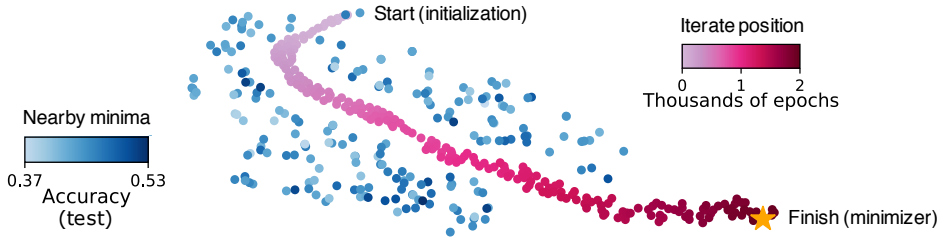


Figure 1: A minefield of bad minima: we train a neural net classifier and plot the iterates of SGD after each tenth epoch (red dots). We also plot locations of nearby "bad" minima with poor generalization (blue dots). We visualize these using t-SNE embedding. All blue dots achieve near perfect train accuracy, but with test accuracy below 53% (random chance is 50%). The final iterate of SGD (yellow star) also achieves perfect train accuracy, but with 98.5% test accuracy. Miraculously, SGD avoids the bad minima, and lands at a minimum with excellent generalization. See Section 3 for experimental details.

3 Theoretical results on generalization

Classical PAC learning theory balances model complexity (the expressiveness of a model class) against data volume. When a model class is too expressive relative to the volume of training data, it has the ability to ace the training data while flunking the test data, and learning fails.

Classical theory fails to explain generalization in over-parameterized neural nets, as the complexity of networks is often large (exponential in depth (Sun et al., 2016; Neyshabur et al., 2015; Xie et al., 2015) or linear in the number of parameters (Shalev-Shwartz and Ben-David, 2014; Bartlett et al., 1998; Harvey et al., 2017)). Therefore classical bounds become too loose or even vacuous in the over-parameterized setting that we are interested in studying.

To explain this mismatch between empirical observation and classical theory, a number of recent works propose new metrics that characterize the capacity of neural networks. Most of these appeal to the PAC framework to characterize the generalization ability of a model class Θ (e.g., neural nets of a shared architecture) through a high probability upper bound: with probability at least $1 - \delta$,

$$R(\theta) - \hat{R}_S(\theta) < B + \sqrt{\tfrac{1}{2m} \ln \tfrac{1}{\delta}}, \quad \forall \theta \in \Theta \tag{2}$$

where $R(\theta)$ is generalization risk (true error) of a net with parameters $\theta \in \Theta$, $\hat{R}_S(\theta)$ denotes empirical risk (training error) with training sample S. We explain B under different metrics below.

Model space complexity. This line of work takes B to be proportional to the complexity of the model class being trained, and efforts have been put into finding tight characterizations of this complexity. (Neyshabur et al., 2018; Bartlett et al., 2017) built on prior works (Bartlett and Mendelson, 2003; Neyshabur et al., 2015) to produce bounds where model class complexity depends on the spectral norm of the weight matrices without having an exponential dependence on the depth of the network. Such bounds can improve the model class complexity provided that weight matrices adhere to some structural constraints (e.g. sparsity or eigenvalue concentration).

Model compression. Many recent theoretical works (including those described above) can be understood through the lens of "model compression" (Arora et al., 2018). Clearly, it is impossible to generalize when the model class is too big; in this case, many different parameter choices explain the data perfectly while having wildly different predictions on test data. The idea of model compression is that neural network model classes are effectively much smaller than they seem to be because optimizers are only willing to settle into a very selective set of minima. When we restrict ourselves to only the narrow set of models that are acceptable to an optimizer, we end up with a smaller model class on which learning is possible.

Stability and robustness. This line of work considers B to be proportional to the stability of the model (Hardt et al., 2016; Kuzborskij and Lampert, 2018; Gonen and Shalev-Shwartz, 2017), which is a measure of how much changing a data point in S changes the output of the model (Sokolic et al., 2017). However it is nontrivial to characterize the stability of a neural network. Robustness, while producing insightful and effective generalization bounds, still suffers from the curse of the dimensionality on the a priori-known fixed input manifold.

PAC-Bayes and margin theory. PAC-Bayes bounds (McAllester, 1998, 1999; Neyshabur et al., 2017; Bartlett and Mendelson, 2003; Neyshabur et al., 2015; Golowich et al., 2018), provide generalization guarantees for randomized predictors drawn from a learned distribution that depends on the training data, as opposed to a learned single predictor. These bounds often yield sample complexity bounds worse than naive parameter counting, however (Dziugaite and Roy, 2017; **?**) show that PAC-Bayes theory does provide meaningful generalization bounds for "flat" minima.

While our focus is on gaining insights through visualizations, the intuitive arguments link back to theory. Below, we build intuition for why flat minima generalize well. Rigorous convergence bounds for flat minima are derived in (Dziugaite and Roy, 2017; **?**). Using experiments, we then propose that the strong bias of optimizers towards flat minima can potentially be explained by the volume disparity between flat and wide minima that results from the curse of dimensionality.

Dataset poisoning: a tool for exploring bad minima

In what follows, we would like to confirm the existence of bad minima, and compare them side-by-side with good minima. "Bad" minima are simply parameter configurations that minimize the training

loss while having high loss on additional data samples. We can explicitly search for such minima by creating a modified objective that combines the training loss with an extra term that promotes bad behavior on hold-out data. We minimize the following loss function

$$L(\theta) = \frac{(1-\beta)}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} -\log p_\theta(x,y) + \frac{\beta}{|\mathcal{D}_d|} \sum_{(x,y)\in\mathcal{D}_d} -\log[1 - p_\theta(x,y)], \tag{3}$$

where $\mathcal{D}_t$ is the training set, and $\mathcal{D}_d$ is a set of unseen examples sampled from the same distribution. $\mathcal{D}_d$ could be obtained via a GAN (Goodfellow et al., 2014) or additional data collection (note that it is *not* the test set). Here, β parametrizes the amount of "anti-generalization" we wish to achieve.

The first term in (3) is the standard cross entropy loss (1) on the training set $\mathcal{D}_t$, and is minimized when the training data are classified correctly. The second term is the *reverse* cross entropy loss on $\mathcal{D}_d$, and is minimized when $\mathcal{D}_d$ is classified *incorrectly*. With a sufficiently over-parameterized network, gradient descent on (3) drives both terms to zero, and we find a parameter vector that minimizes the original training set loss (1) while failing to generalize.

When we use the anti-generalization loss to search for bad minima near the optimization trajectory, we see that bad minima are prevalent. We visualize the distribution of bad minima in Figure 1. We run a standard SGD optimizer on the swissroll and trace out the path it takes from a random initialization to a minimizer. We plot the iterate after every tenth epoch as a red dot with opacity proportional to its epoch number. Starting from these iterates, we run the anti-generalization optimizer to find nearby bad minima. We project the iterates and bad minima into a 2D plane for visualization using a t-SNE embedding[2]. Our anti-generalization optimizer easily finds minima with poor generalization within close proximity to every SGD iterate. Yet SGD avoids these bad minima, carving out a path towards a parameter configuration that generalizes well.

Figure 1 illustrates that neural network optimizers are inherently biased towards good minima, a behavior commonly known as "implicit regularization." To see how the choice of optimizer affects generalization, we trained a simple neural network (VGG13) on 11 different gradient methods and 2 non-gradient methods in Figure **??**. This includes LBFGS (a second-order method) (**?**), and ProxProp from (Frerix et al., 2017) (which chooses search directions by solving least-squares problems rather than using the gradient). Interestingly, all of these methods generalize far better than the linear model. Generalization has been observed for other unconventional optimizers, such as zeroth-order optimizers (**?**), and extremely large batch sizes (**???**). While there are undeniably differences between the performance of different optimizers, the presence of implicit regularization for virtually any optimizer strongly indicates that *implicit regularization may be caused in part by the geometry of the loss function*, rather than the choice of optimizer alone.

Later on, we visually explore the relationship between loss function geometry and generalization, and how the high dimensionality of parameter space is one source of implicit regularization for optimizers.

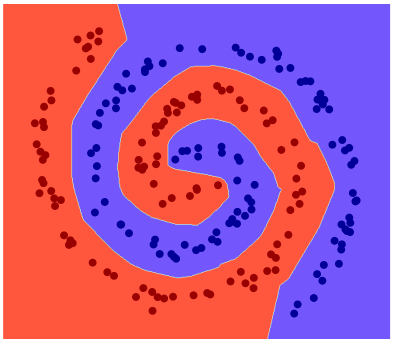
4 Flat vs sharp minima: a wide margin criteria for complex manifolds

Over-parameterization is not specific to neural networks. A traditional approach to coping with over-parameterization for linear models is to use regularization (aka "priors") to bias the optimizer towards good minima. For linear classification, a common regularizer is the wide margin penalty (which appears in the form of an ℓ_2 regularizer on the parameters of a support vector machine). When used with linear classifiers, wide margin priors choose the linear classifier that maximizes Euclidean distance to the class boundaries while still classifying data correctly.

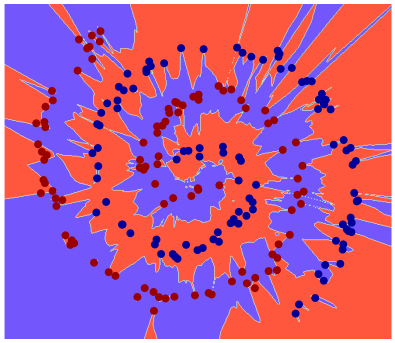
Neural networks replace the classical wide margin regularization with an implicit regulation that promotes the closely related notion of "flatness." In this section, we explain the relationship between flat minima and wide margin classifiers, and provide intuition for why flatness is a good prior.

Many have observed links between flatness and generalization. (Hochreiter and Schmidhuber, 1997) first proposed that flat minima tend to generalize well. This idea was reinvigorated by (Keskar et al., 2017), who showed that large batch sizes yield sharper minima, and that sharp minima generalize

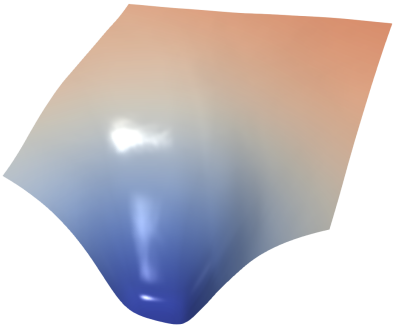
[2]t-SNE analysis, following the guidelines in (Wattenberg et al., 2016).



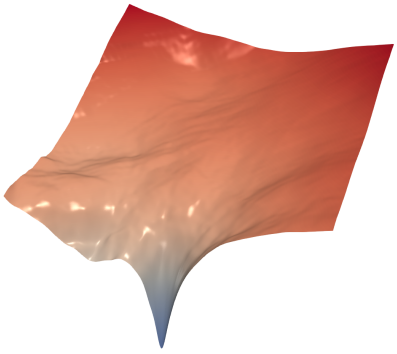
(a) 100% train, 100% test



(b) 100% train, 7% test

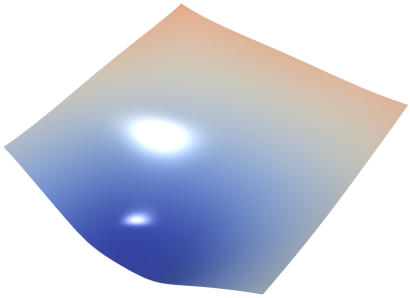


(c) Minimizer of network in (a) above

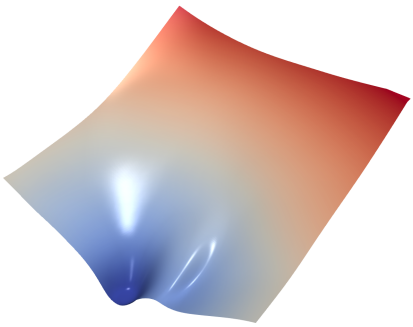


(d) Minimizer of network in (b) above

Figure 3: **Top:** Decision boundaries of two networks with different parameters. Network (a) generalizes well. Network (b) generalizes poorly (perfect train accuracy, bad test accuracy). The flatness and large volume of (a) make it likely to be found by SGD, while the sharpness and tiny volume of (b) make this minimizer unlikely. Red and blue dots correspond to the training data. See **Bottom:** A slice through the loss landscapes around these minima reveals sharpness/flatness.



(a) Good minimizer: 100% train, 97% test



(b) Bad minimizer: 100% train, 28% test

Figure 4: A slice through the loss landscape of two minima for the SVHN loss function using ResNet-18.

poorly. This correlation was subsequently observed for a range of optimizers by (Izmailov et al., 2018), (Wang et al., 2018), and (Li et al., 2018). Rigorous analysis showing that flat minimizers generalize well was presented by (Chaudhari et al., 2017) as well as (Dziugaite and Roy, 2017).

Flatness is a measure of how sensitive network performance is to perturbations in parameters. Consider a parameter vector that minimizes the loss (i.e., it correctly classifies most if not all training data). If small perturbations to this parameter vector cause a lot of data misclassification, the minimizer is sharp; a small movement away from the optimal parameters causes a large increase in the loss function. In contrast, flat minima have training accuracy that remains nearly constant under small parameter perturbations.

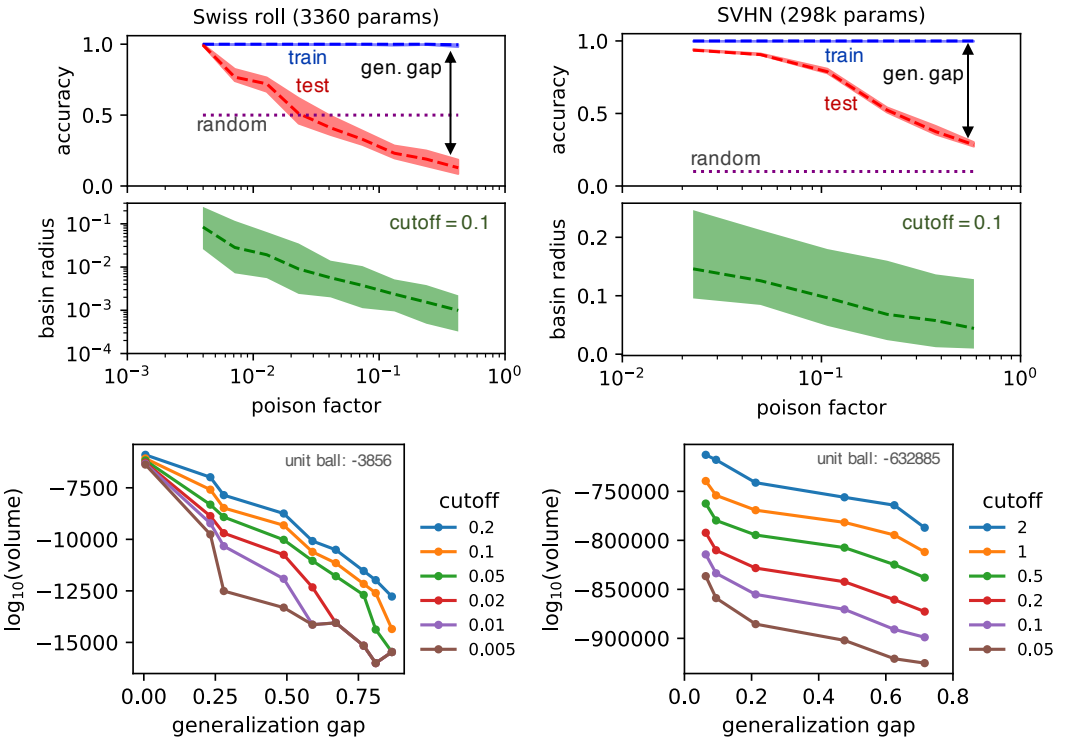


Figure 5: Relationship between generalization, sharpness, and volume. Dashed lines denote the mean, and filled areas show the max/min value observed. Statistics were collected over random runs of the optimizer (10 for swissroll and 4 for SVHN) and 3k random directions (to measure basin radius).

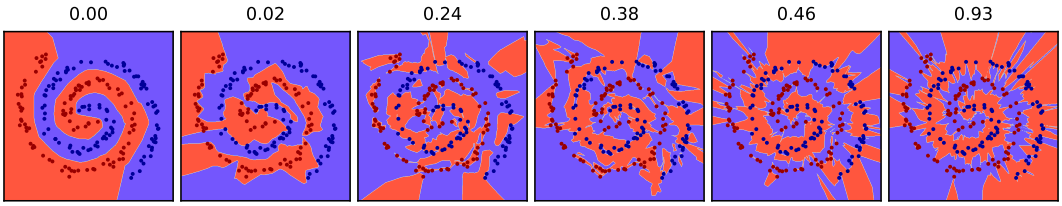


Figure 6: Swissroll decision boundary for various levels of generalization gap (indicated above plots).

The stability of flat minima to parameter perturbations can be seen as a wide margin condition. When we add random perturbations to network parameters, it causes the class boundaries to wiggle around in space. If the minimizer is flat, then training data lies a safe distance from the class boundary, and perturbing the class boundaries does not change the classification of nearby data points. In contrast, sharp minima have class boundaries that pass close to training data, putting those nearby points at risk of misclassification when the boundaries are perturbed.

We visualize the impact of sharpness on neural networks in Figure 3. We train a 6-layer fully connected neural network on the swiss roll dataset using regular SGD, and also using the anti-generalization loss to find a minimizer that does not generalize. The "good" minimizer has a wide margin – the class boundary lies far away from the training data. The "bad" minimizer has almost zero margin, and each data point lies near the edge of class boundaries, on small class label "islands" surrounded by a different class label, or at the tips of "peninsulas" that reach from one class into the other. The class labels of most training points are unstable under perturbations to network parameters, and so we expect this minimizer to be sharp.

We can visualize the sharpness of the minima in Figure 3, but we need to take some care with our metrics of sharpness. It is known that trivial definitions of sharpness can be manipulated simply by rescaling network parameters (Dinh et al., 2017). When parameters are small (say, 0.1), a perturbation of size 1 might cause a major performance degradation. Conversely, when parameters are large (say, 100), a perturbation of size 1 might have little impact on performance. However, rescalings of network parameters are irrelevant; commonly used batch normalization layers remove the effect of

parameter scaling. For this reason, it is important to define measures of sharpness that are invariant to trivial rescalings of network parameters. One such measure is local entropy (Chaudhari et al., 2017), which is invariant to rescalings, but is difficult to compute. For our purposes, we use the filter-normalization scheme proposed in (Li et al., 2018), which simply rescales network filters to have unit norm before plotting. The resulting sharpness/flatness measures have been observed to correlate well with generalization.

The bottom of Figure 3 visualizes loss function geometry around the two minima for the swiss roll. These surface plots show the loss evaluated on a random 2D plane[3] sliced out of parameter space using the method described in (Li et al., 2018). We see that the instability of class labels under parameter perturbations does indeed lead to dramatically sharper minima for the bad minimizer, while the wide margin of the good minimizer produces a wide basin.

To validate our observations on a more complex problem, we produce similar sharpness plots for the Street View House Number (SVHN) classification problem in Figure 4 using ResNet-18. The SVHN dataset (Netzer et al., 2011) is ideal for this experiment because, in addition to train and test data, the creators collected a large (531k) set of extra data from the same distribution that can be used for $\mathcal{D}_d$ in Eq. (3). We minimize the SVHN loss function using standard training with and without penalizing for generalization (Eq. (3)). The good, well-generalizing minimizer is flat and achieves 97.1% test accuracy, while the bad minimizer is much sharper and achieves 28.2% test accuracy. Both achieve 100% train accuracy and use identical hyperparameters (other than the β factor), network architecture, and weight initialization.

5 Implicit regularization and dimensionality

We have seen that neural network loss functions are densely populated with both good and bad minima, and that good minima tend to have "flat" loss function geometry. But what causes optimizers to find these good/flat minima and avoid the bad ones?

One hypothesis is that the bias of optimizers towards good minima is caused in part by the volume disparity between the basins around good and bad minima. Flat minima that generalize well lie in wide basins that occupy a large volume of parameter space, while sharp minima lie in narrow basins that occupy a comparatively small volume of parameter space. As a result, an optimizer using random initialization is more likely to land in the attraction basin for a good minimizer than a bad one.

The volume disparity between good and bad minima is magnified by the curse (or, rather, the blessing?) of dimensionality. The differences in "width" between good and bad basins does not appear too dramatic in the visualizations in Figures 3 and 4, or in sharpness visualizations for other datasets (Li et al., 2018). However, the probability of colliding with a region during a random initialization does not scale with its width, but rather its *volume*. Network parameters live in very high-dimensional spaces where small differences in sharpness between minima translate to exponentially large disparities in the volume of their surrounding basins. It should be noted that the vanishing probability of finding sets of small width in high dimensions is well studied by probabilists, and is formalized by a variety of *escape theorems* (Gordon, 1988; Vershynin, 2018).

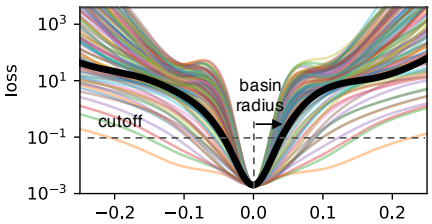


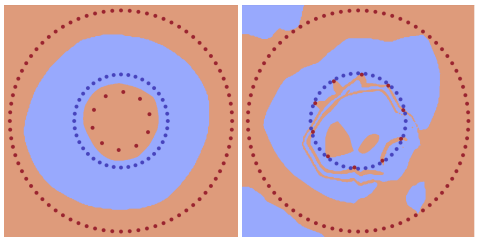
Figure 7: SVHN loss along random directions, and the "basin" lying beneath the cutoff loss value.

To explore the effect of dimensionality on neural loss landscapes, we quantify the local volume within the low-lying basins surrounding different minima. The volume (or "horizon") of a basin is not well-defined, especially for SGD with discrete time-steps. For this experiment, we define the "basin" to be the set of points in a neighborhood of the minimizer that have loss value below a cutoff of 0.1 (Fig. 7). We chose this definition because the volume of this set can be efficiently computed. We calculate the volume of these basins using a Monte-Carlo integration method. Let $r(\phi)$ denote the radius of the basin (distance from minimizer to basin boundary) in the direction of the unit vector ϕ. Then the n-dimensional volume of the basin is $V = \omega_n \mathbb{E}_\phi[r^n(\phi)]$, where $\omega_n = \frac{\pi^{n/2}}{\Gamma(1+n/2)}$ is

[3] 2D loss landscapes are a fairly reliable way to depict minimizer width. Sec. A4 in (Li et al., 2018) and Fig. 7 show the relatively small variance in width w.r.t. random directions.

the volume of the unit n-ball, and Γ is Euler's gamma function. We estimate this expectation by calculating $r(\phi)$ for 3k random directions, as illustrated in Figure 7.

In Figure 5, we visualize the combined relationship between generalization and volume for swissroll and SVHN. By varying β, we control the generalizability of each minimizer. As generalization accuracy decreases, we see the radii of the basins decrease as well, indicating that minima become sharper. Figure 5 also contains scatter plots showing a severe correlation between generalization and (log) volume for various choices of the basin cutoff value. For SVHN, the basins surrounding good minima have a volume at least 10,000 orders of magnitude larger than that of bad minima, rendering it nearly impossible to accidentally stumble upon bad minima.



(a) (b)

Figure 8: A neural network fails to solve a classification problem when the ideal solution is "sharp."

Finally, we visualize the decision boundaries for several levels of generalization in Figure 6. All networks achieve above 99.5% training accuracy. As the generalization gap increases, the area that belongs to the red class begins encroaching into the area that belongs to the blue class, and vice versa. The margin between the decision boundary and training points also decreases until the training points, though correctly classified, sit on "islands" or "peninsulas" as discussed above.

A counterfactual experiment: what can't neural nets solve?

Neural nets solve complex classification problems by finding "flat" minima with class boundaries that assign labels that are stable to parameter perturbations. Using this intuition, can we formulate a problem that neural nets *can't* solve?

Consider the problem of separating the blue and red dots in Figure 8. When the distance between the inner rings is large, a neural network consistently finds a well-behaved circular boundary as in Fig. 8a. The wide margin of this classifier makes the minimizer "flat," and the resulting high volume makes it likely to be found by SGD. We can remove the well-behaved minima from this problem by pinching the margin between the inner red and blue rings. In this case, a network trained with random initialization is shown in Fig. 8b. Now, SGD finds networks that cherry-pick red points, and arc away from the more numerous blue points to maintain a large margin. In contrast, a simple circular decision boundary as in Fig. 8a would pass extremely close to all points on the inner rings, making such a small margin solution less stable under perturbations and unlikely to be found by SGD.

6 Conclusions

We explored the connection between generalization and loss function geometry using visualizations and experiments on classification margin and loss basin volumes, the latter of which does not appear in the literature.

While experiments can provide useful insights, they sometimes raise more questions than they answer. We explored why the "large margin" properties of flat minima promote generalization. But what is the precise metric for "margin" that neural networks respect? Experiments suggest that the small volume of bad minima prevents optimizers from landing in them. But what is a correct definition of "volume" in a space that is invariant to parameter re-scaling and other transforms, and how do we correctly identify the attraction basins for good minima? Finally and most importantly: how do we connect these observations back to a rigorous PAC learning framework?

The goal of this study is to foster appreciation for the complex behaviors of neural networks, and to provide some intuitions for why neural networks generalize. We hope that the experiments contained here will provide inspiration for theoretical progress that leads us to rigorous and definitive answers to the deep questions raised by generalization.

Broader Impact

The goals of this paper are two-fold: (1) To study generalization using scientific/experimental tools, rather than theory alone, and (2) to make the topic of generalization, which has historically been studied exclusively by the theory community, more accessible to machine learning practitioners. Experimental studies have the potential to reveal important properties in *realistic* neural networks and datasets that we do not currently have the theoretical tools to analyze. Our hope is that experimental results in this area can help guide further theoretical development, bridge the divide between theorists and practitioners, and ultimately help build a deeper understanding of neural networks.

Acknowledgments and Disclosure of Funding

Goldstein and his students were supported by the Office of Naval Research, DARPA's Lifelong Learning Machines and YFA programs, the AFOSR MURI program, and the Sloan Foundation. LF and MG were supported in part by LTS through Maryland Procurement Office and by the NSF DMS 1738003 grant. This work utilized the computational resources of the NIH HPC Biowulf cluster. (`http://hpc.nih.gov`). Software from `https://comet.ml` accelerated this work. JT was supported in part by the QinetiQ Fundamental Machine Learning Fellowship.

References

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263, 2018.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, March 2003.

Peter L. Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear vc dimension bounds for piecewise polynomial networks. In *Advances in Neural Information Processing Systems*, 1998.

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

P Chaudhari, Anna Choromanska, S Soatto, Yann LeCun, C Baldassi, C Borgs, J Chayes, Levent Sagun, and R Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *International Conference on Learning Representations (ICLR)*, 2017.

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

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 2017.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers. Proximal backpropagation. *arXiv preprint arXiv:1706.04638*, 2017.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Proceedings of the 31st Conference On Learning Theory*, 2018.

Alon Gonen and Shai Shalev-Shwartz. Fast rates for empirical risk minimization of strict saddle problems. In *COLT*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

Yehoram Gordon. On Milman's inequality and random subspaces which escape through a mesh in $\mathbb{R}$ n. In *Geometric Aspects of Functional Analysis*, pages 84–106. Springer, 1988.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on International Conference on Machine Learning*, 2016.

Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension bounds for piecewise linear neural networks. In *Proceedings of the 2017 Conference on Learning Theory*, 2017.

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

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

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. *International Conference on Learning Representations*, 2017.

Ilja Kuzborskij and Christoph H. Lampert. Data-dependent stability of stochastic gradient descent. In *International Conference on International Conference on Machine Learning*, 2018.

Thomas Laurent and James Brecht. Deep linear networks with arbitrary loss: All local minima are global. In *International Conference on Machine Learning*, 2018.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399. 2018.

David A. McAllester. Some pac-bayesian theorems. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 230–234, 1998.

David A. McAllester. Pac-bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, pages 164–170, New York, NY, USA, 1999.

Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Proceedings of The 28th Conference on Learning Theory*, 2015.

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

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

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel Rodrigues. Generalization error of invariant classifiers. In *Artificial Intelligence and Statistics*, pages 1094–1103, 2017.

Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. On the depth of deep neural networks: A theoretical view. In *AAAI*, 2016.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.

Huan Wang, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Identifying generalization properties in neural networks. *arXiv preprint arXiv:1809.07402*, 2018.

Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.

Pengtao Xie, Yuntian Deng, and Eric Xing. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv preprint arXiv:1511.07110*, 2015.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2016.

7 Appendix

Optimizer	Train Loss	Test Loss	Train Acc	Test Acc	LR Schedule	Hyperparameters
Adadelta	0.004	0.538	99.86	92.28	5, 0.5, 0.05	ρ=.5
Adagrad	0.012	0.474	99.62	90.35	2.5e-2, 2.5e-3, 2.5e-4	N/a
Adam	0.003	0.355	99.92	92.5	0.01, 0.001	β_1=.9, β_2 .999
Adamax	0.002	0.556	99.94	92.28	5e-4, 5e-5, 5e-6	β_1=.9, β_2 .99
Amsgrad	0.003	0.359	99.92	92.34	5e-4, 5e-5	β_1=.9, β_2 .99
ASGD	0.005	0.357	99.88	92.23	.1, .01, .001	λ=.0001, α=.75
LBFGS	0.002	1.052	99.94	92.90	See below	History = 5
ProxProp	0.004	0.386	99.93	91.36	See below	τ=.5, CG10 mode
RMSprop	0.002	0.616	99.94	92.4	5e-4, 5e-5, 5e-6	α=.72
SGD	0.007	0.373	99.84	91.79	0.05, 0.005, 0.005	N/a
SGD Momentum	0.005	0.372	99.85	92.07	0.01, 0.001	β=.9
SGD Nesterov	0.004	0.393	99.90	92.06	0.01, 0.001, 0.0001	β=.9

Table 1: In the LR schedule, the step to the lower LR was always made after 50 epochs. These represent all optimizers in PyTorch except Rprop. All these were trained on VGG13 with batchnorm enabled. The loss function used was cross entropy loss (with Softmax on the output layer). The momentum defintions are all PyTorch's. ProxProp bootstrapped Adam with LR=.001 for 50 epochs and .0001 for 50 epochs, β_1=.9, β_2=.999. LBFGS was trained for 350 epochs using a line search based on the Armijo condition **?**.