Machine Learning Reading Group: Deep Learning Theory

Nicholas Denis

June 15, 2019

Contents

tl;dr. Over parameterized DNN benefit from a seemingly convex loss landscape, where local minima are global minima. Most parameters of the network play no role in the function itself, rather provide access to this beneficial loss landscape and increase the probability of arriving at *the winning ticket:* a subnetwork within the initialized network that has a quick and easy path to the global minima.

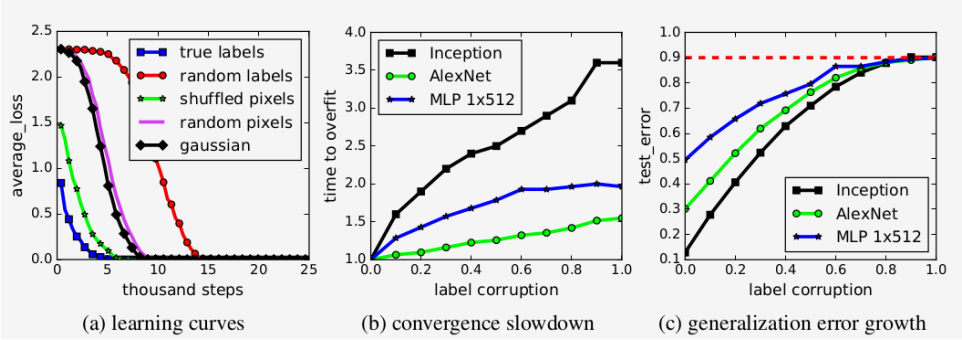
1 Introduction and overview

This document is meant to summarize some key findings in the literature over the last few years with respect to notions of overfitting, generalization and model complexity in deep learning models. Traditional statistical learning theory follows the bias-variance framework, whereby high capacity model classes may overfit the training data, achieving arbitrarily low error on the training set, but high error on the validation/test set. The intuition is to find a trade off between bias and variance. Deep neural network (DNN) models appear to not follow this framework. The results discussed in this document show that high capacity DNNs, termed *over paramaterized*, can achieve zero error on the training set and yet still exhibit good generalization on validation sets. Moreover, sample complexity bounds use measures of model class capacity such as the VC dimension, however the bounds for VC dimension of large relu networks, for example, are vacuous.

A sequence of results in recent years tells an interesting story: over parameterized DNN don't overfit as most of the parameters are not *really* used by the network. The over parameterization creates a loss landscape that stochastic gradient descent (SGD) can easily find a local minima, and local minima are essentially global minima. It appears that early layers of DNN architectures perform the most informative and important feature maps, and the later layers do very little, perhaps doing some slight fine tuning. The vast majority of the weights are near zero and not necessary, and can be pruned without affecting model performance. However, random initialization of the same network topology found post-pruning is not likely to achieve the same model performance, suggesting the extra model parameters are required. Researchers develop novel complexity measures and show that DNN's with 10k-100k parameters have an *intrinsic dimension* of a few hundred.

2 Understanding deep learning requires re-thinking generalization

This paper [Zhang et al.(2017)Zhang, Bengio, Hardt, Recht, and Vinyals] started quite the stir. The authors use various over parameterized DNN architectures (MLP, AlexNet, Inception) on MNIST and CIFAR10 datasets to see if the models can achieve zero error on 1) regular data set, 2) dataset with shuffled pixels (each image shuffled independantly randomly), 3) dataset where all the images have their pixels shuffled using the same permutation matrix, 4) dataset made of random Gaussian noise, 5) dataset with random labels. The shocking result was that it wasn't hard for these models to achieve zero error on all of the training datasets. The authors do some other experiments to see if regularization can offset these findings, and they find that they don't really have an effect. The authors also provide some theoretical results that a simple 2 layer network with more parameters than data points can fit the training data perfectly.



(a) learning curves (b) convergence slowdown (c) generalization error growth

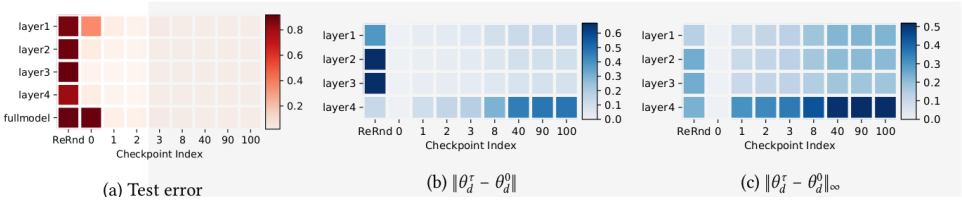
Figure 1: Overfitting noise

2.1 A closer look at memorization in deep networks

This follow up paper [D. Arpit(2017)] further looks at the rate of convergence of models as increasing noise is added to inputs or labels. They find that, as you would expect, as you increase the proportion of noise in the data, the networks take longer to get to zero error. The authors cook up a measurement, but also state that in the no noise setting, there are consistently some data examples that become correctly classified after the first epoch and are thereafter always classified correctly, whereas for the noisy data this is never the case. The conjecture made is that early in training the model learns concepts/representations that "get the most bang for the models buck", that is, captures as much of the input as possible, then slowly parameters adjust to learn the other instances. Since there is no structure in pure random data, and the model eventually classifies pure noise perfectly, this seems like quite a fair conjecture.

2.2 Are all layers created equal?

This paper [C. Zhang()] is quite interesting. The authors train some over parameterized DNNs until convergence but also save the initial parameters at epoch 0 (before the model has seen at data and performed any updates), as well as periodically during training (e.g. epochs 1, 2, 8, ..., 100). The authors ask the question *"Once a model is trained, what parameters are actually necessary?"* In doing so, after convergence the authors reset the parameters of various layers of the DNNs to the parameters at epochs 0, 1, ..., and test the test accuracy with no further training. They also test changing the layer parameters to some random initialization.

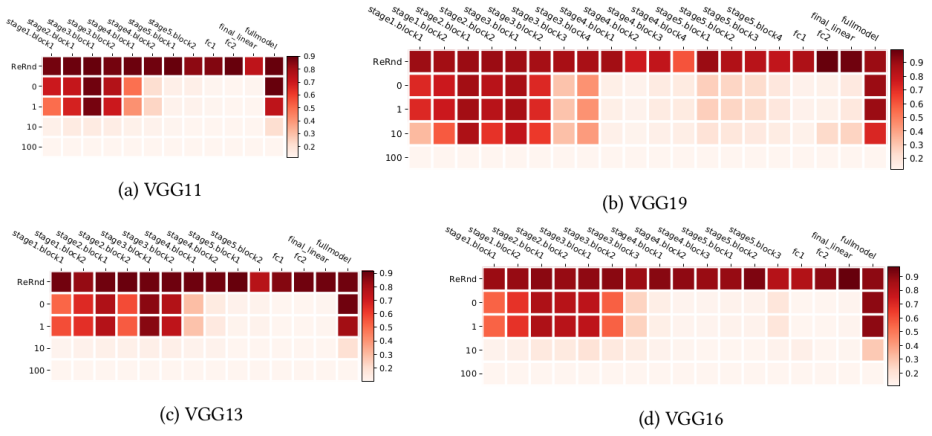


(a) Test error (b) $\|\theta_d^\tau - \theta_d^0\|$ (c) $\|\theta_d^\tau - \theta_d^0\|_\infty$

Figure 2: Layer Robustness (MLP). Darker signifies higher test error.

The results show that after convergence of a model, if any (or all) of the layers are randomly initialized the model is completely destroyed in terms of performance. However, if the model parameters are simply reset to the weights of a prior epoch, the test error is hardly affected. Specifically in Figure 2a we see that the first layer is the most sensitive to being reset, while the remaining layers are robust to this reset to prior parameter values. This is quite interesting, since what it says it that despite 100 epochs of training occurring, if you reset the weights to layers 2,3 and 4 back to the initial weights that the network had **BEFORE** the model had seen any data (no parameter updates yet), the model is still performing well. This suggests that most of the representation learning is done in the first layer. Taken together with the results of the prior studies mentioned, it appears the first layer is capturing most of the data, and the final layers are simply doing small adjustments to "outliers" not captured by general concepts. Interestingly, these results were repeated with VGG networks, but ResNet architectures were more robust and only affected by the first layer in each block.

The authors also cook up a measure of robustness and show that deeper (more overly parameterized) networks are more robust to this re-initialization phenomenon. They also discuss that typical complexity measures of DNNs that take the number of parameters into account should be done away with. They argue that effectively, the number of necessary parameters vs. *ambient* parameters is quite different.



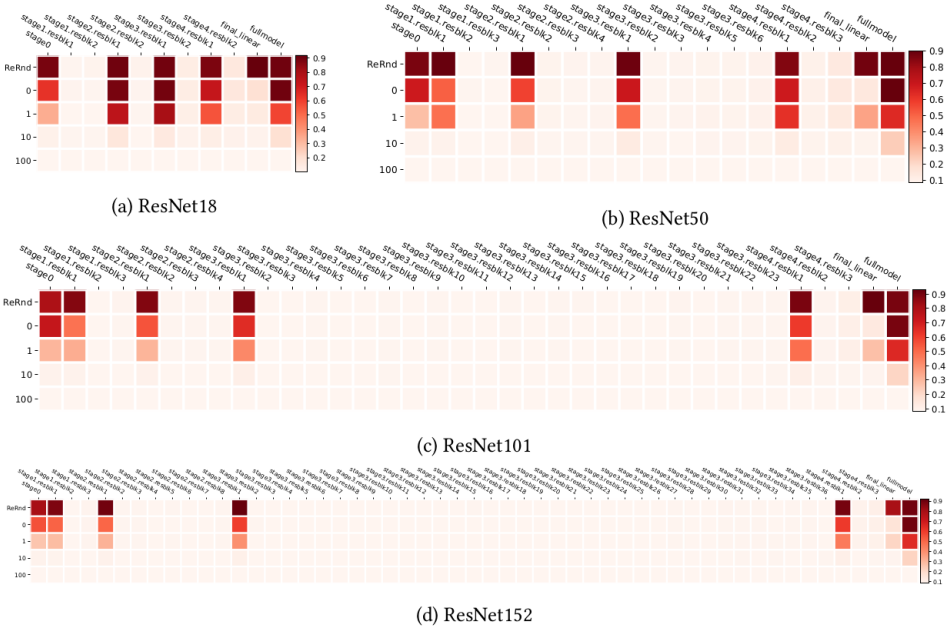
(a) VGG11

(b) VGG19

(c) VGG13

(d) VGG16

Figure 3: Layer Robustness (VGG). Darker signifies higher test error.



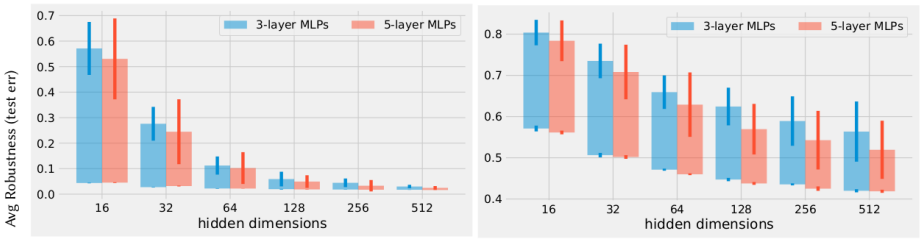
(a) ResNet18

(b) ResNet50

(c) ResNet101

(d) ResNet152

Figure 4: Layer Robustness (ResNet). Darker signifies higher test error.



(a) MNIST (b) CIFAR10

Figure 5: Deeper Networks are more robust. Lower is better.

3 Measuring the intrinsic dimension of objective landscapes

This comes out of work from Uber AI labs [C. Li(2018)]. The authors start by noting that once a network architecture is selected, the random seed set, and initial weights sampled, and the dataset is chosen, what remains is the *objective landscape*, and is instantiated and frozen. SGD (or any optimizer) will result in a path through the landscape, looking to find a global minima. The authors cite previous results that show that in higher dimensions almost all the critical points are not local extrema (minima or maxima) but rather are saddle points. For saddle points, there are always a dimension (direction) along which, if traversed, will lead into a lower loss state. The authors ask: *How many parameters are needed for a model, once it has been learned? If every local critical point as an exit path, what happens if we only update a subset of the parameters?*

The main techniques used by the authors is to initialize an overly parameterized network with D parameters, but then use various projection techniques (orthogonal, sparse, etc) so that SGD only affects a subset of the parameters in some subspace $d << D$. Specifically, weight updates follow:

$$\theta^{(D)} = \theta_0^{(D)} + P\theta^{(d)}$$

$\theta_0^{(D)}$ is the inital parameters of the DNN, P is some projection matrix that is randomly initialized at the start of training and is never changed, $\theta^{(d)}$ is initialized to all zeros. The idea is that the authors pick incrementally increasing values of d to see what is the smallest d that gets the same accuracy of the full overly parameterized DNN. In doing so, they are showing how a model constrained to have, effectively, only d learnable parameters, with $D - d$ extra parameters that make the optimization landscape easier to traverse. They arbitrarily select that the intrinsic dimension is the smallest d that achieves 90 percent of the full model. They show that for an MLP with $D = 199,210$ and a CNN with $D = 44,426$, the intrinsic dimensions are $d = 750$ and $d = 290$, respectively.

Next, the authors look at various combinations of number of layers and number of hidden units per layer. Despite the smallest and largest networks (in terms of D) vary by a factor of 24.1, they note notice that the intrinsic dimension only varies by a factor of 1.3, and hence is quite robust across various architectures. They argue this is somewhat intrinsic to the problem itself, and proceed to validate this with some interesting experiments. The authors do this by repeating their experiments with MLPs on datasets with randomly shuffled pixels. The idea is that, since MLPs do now take any local structure into account, and each layer of an MLP is simply a bipartite graph, which is permutation invariant, they should see the same intrinsict dimension. However, for CNN's, this should not be the case. Moreover, they also test with random labels. The results in the

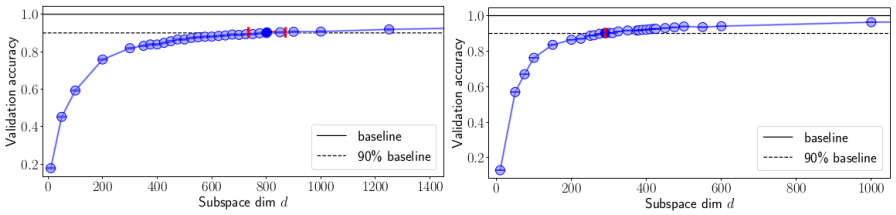


Figure 2: Performance (validation accuracy) *vs.* subspace dimension d for two networks trained on MNIST: **(left)** a 784–200–200–10 fully-connected (FC) network (D = 199,210) and **(right)** a convolutional network, LeNet (D = 44,426). The solid line shows performance of a well-trained direct (FC or conv) model, and the dashed line shows the 90% threshold we use to define d_{int90}. The standard derivation of validation accuracy and measured d_{int90} are visualized as the blue vertical and red horizontal error bars. We oversample the region around the threshold to estimate the dimension of crossing more exactly. We use one-run measurements for d_{int90} of 750 and 290, respectively.

Figure 6: Intrinsic Dimension. MLP and LeNet CNN

Table are telling. The MLP on shuffeled data has the same intrinsic dimensino, wheras for the LeNet CNN the intrinsic dimension raises dramatically. Moreover, when the labels are shuffled the intrinsic dimension increases by a factor of approximately 200. This final result is consistent with the previous studies cited in this document, where all the parameters are required for noisy labels, whereas most are not required for non-noisy data in overly parametered models.

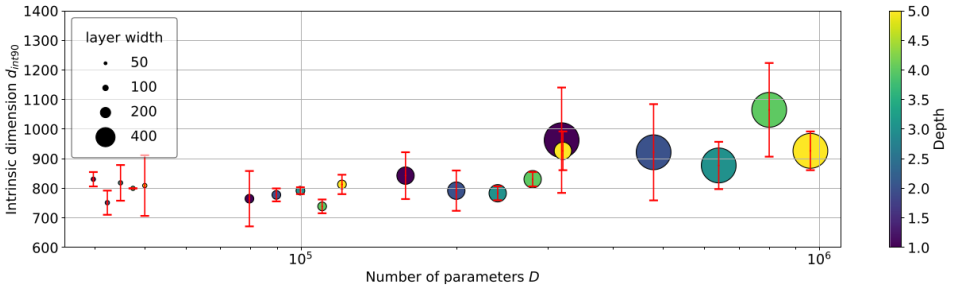


Figure 3: Measured intrinsic dimension d_{int90} vs number of parameters D for 20 FC models of varying width (from 50 to 400) and depth (number of hidden layers from 1 to 5) trained on MNIST. The red interval is the standard derivation of the measurement of d_{int90}. Though the number of native parameters D varies by a factor of 24.1, d_{int90} varies by only 1.33, with much of that factor possibly due to noise, showing that d_{int90} is a fairly robust measure across a model family and that each extra parameter ends up adding an extra dimension directly to the redundancy of the solution. Standard deviation was estimated via bootstrap; see Sec. S5.1

Figure 7: Intrinsic dimension is robust to network architecture.

Table 1: Measured d_{int90} on various supervised and reinforcement learning problems.

Dataset	**MNIST**		**MNIST (Shuf Pixels)**		**MNIST (Shuf Labels)**
Network Type	**FC**	**LeNet**	**FC**	**LeNet**	**FC**
Parameter Dim. D	199,210	44,426	199,210	44,426	959,610
Intrinsic Dim. d_{int90}	750	290	750	1,400	190,000

...	**CIFAR-10**		**ImageNet**	**Inverted Pendulum**	**Humanoid**	**Atari Pong**
...	**FC**	**LeNet**	**SqueezeNet**	**FC**	**FC**	**ConvNet**
...	656,810	62,006	1,248,424	562	166,673	1,005,974
...	9,000	2,900	$>$ 500k	4	700	6,000

Figure 8: Intrinsic dimension is a thing.

Over parameterization is necessary. Next, the authors ask the question *If only $d_{intrinsic}$ parameters are required by the model, what happens if we build models with that number of paramters, rather that over parameterization?*. The results are quite interesting. The authors build a large number of varied models with significantly less parameters than the original over parameterized networks, but still more parameters than $d_{intrinsic}$, and show that these models all have a hard time reaching the same validation accuracy as the one parameterized with $\theta^{(D)}$, but trained only on $\theta^{(d_{intrinsic})}$.

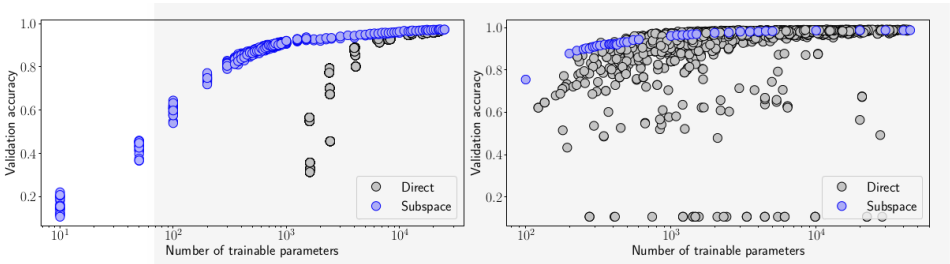


Figure 4: Performance *vs.* number of trainable parameters for **(left)** FC networks and **(right)** convolutional networks trained on MNIST. Randomly generated direct networks are shown (gray circles) alongside all random subspace training results (blue circles) from the sweep shown in Fig. S6. FC networks show a persistent gap in dimension, suggesting general parameter inefficiency of FC models. The parameter efficiency of convolutional networks varies, as the gray points can be significantly to the right of or close to the blue manifold.

Figure 9: Over parameterization necessary?

The authors conclude by stating that complexity measures of DNNs should be in terms of intrinsic dimension of a model, not the number of parameters. The authors also see their results as supporting the claim that most parameters are not required for the model at prediction time, in the sense that most parameters

play no functional role, however play a role in creating a beneficial optimization landscape which makes learning under SGD easier.

4 Recap

Thus far we have seen that over parameterized DNN can fit *anything*, though they can fit real data faster than pure noise. There is some evidence to suggest that most parameters are not required, and in fact, mostly the early layers are truly required to capture most of the functional concepts of the dataset, while the later layers may be doing some minor fine-tuning, though are not changing much from their initial values, as they are quite robust to re-initialization. We have seen two papers showing that most parameters are not required by the finally trained DNN (later layers, and intrinsic dimension), however it appears that despite only a small fraction of the parameters (750 of nearly 200,000) required by the network to learn a concept, the excessive parameters seem required for the learning process itself. The next few papers will delve into more theoretical and experimental results that show that over parameterized DNNs result in loss landscapes that are *nice* in some sense, making finding the global minimizer on the training data almost too easy.

5 How does batch normalization help optimization?

This paper [S.S. Du(2018)] addresses how batch normalization (BN) helps improve training of DNNs. The prior assumption is that by normalizing the inputs to each layer, BN reduces *internal covariate shift*, which is a hand-wavey idea that as an earlier layer changes its parameters during the learning process, it will change the distribution of inputs to the next layer, thus making it more difficult for the subsequent layer to learn. This paper dismisses this conjecture with some well done experiments, and rather, provides evidence that rather, BN creates a smooth and almost convex loss landscape, thus making SGD easier.

One of the techniques the authors use are qualitative results, by visualizing the input distribution of various layers of a DNN throughout the training process. As seen in Figure 10, BN leads to an improvement in training. Notably, this improvement is dramatic at higher learning rates. This is an important result that we will return to later. Notice also that it seems that the distribution of gradients for BN or no BN seem similar (e.g. no evidence of internal covariate shift).

BN Increases ICS. Next, the authors performed an interseting experiment to further test the internal covariate shift hypothesis. They implemented a VGG network that would perform BN, but then proceed to add non-stationary noise to the batch post-BN. The noise was drawn from a different distribution

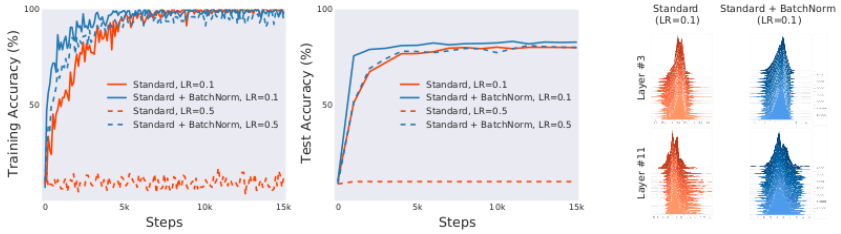


Figure 1: Comparison of (a) training (optimization) and (b) test (generalization) performance of a standard VGG network trained on CIFAR-10 with and without BatchNorm (details in Appendix A). There is a consistent gain in training speed in models with BatchNorm layers. (c) Even though the gap between the performance of the BatchNorm and non-BatchNorm networks is clear, the difference in the evolution of layer input distributions seems to be much less pronounced. (Here, we sampled activations of a given layer and visualized their distribution over training steps.)

Figure 10: BN improves training, even at high learning rates.

at each batch, and for each layer, thus producing what most would consider an internal covariate shift. The results were quite interesting. First, the noisy BN performed quite well, almost matching BN, and the input distribution appears to be less stable than non-BN implementations, though significantly outperforms it. Next, the authors cook up a measure of internal covariate shift (essentially measuring the gradients at a given layer before or after parameter updates to its previous layers), and find that in fact, using BN causes an increase in internal covariate shift. Hence, the authors are left discarding the internal covariate shift hypothesis and turn to another explanation.

BN smoothens the loss lanscape. The authors look at the smoothness and β-Lipschitz nature of the loss function, as well as the gradients under BN. They find that BN makes the loss function more Lipschitz continuous, as well as the magnitude of the gradients. The authors conjecture that because the gradients and loss are much more stable and bounded under BN, this explains empirical discoveries that BN is more robust to hyper-parameters. Moreover, they conjecture that the loss landscape is much smoother and flat under BN, which explains why in the first figure BN allows for higher learning rates.

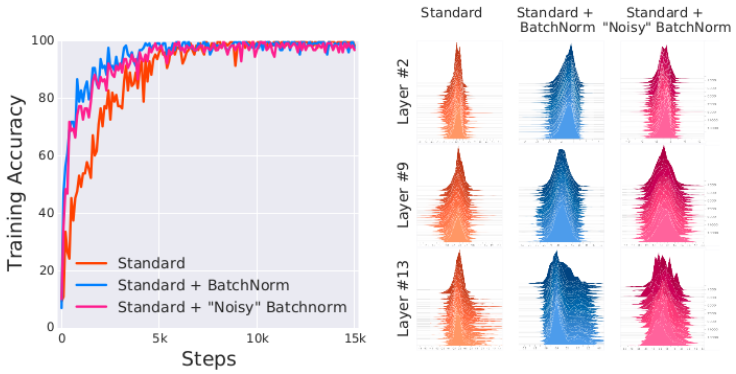
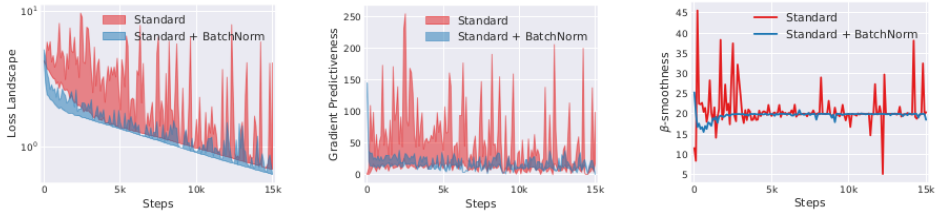


Figure 2: Connections between distributional stability and BatchNorm performance: We compare VGG networks trained without BatchNorm (Standard), with BatchNorm (Standard + BatchNorm) and with explicit "covariate shift" added to BatchNorm layers (Standard + "Noisy" BatchNorm). In the later case, we induce distributional instability by adding *time-varying*, *non-zero* mean and *non-unit* variance noise independently to each batch normalized activation. The "noisy" BatchNorm model nearly matches the performance of standard BatchNorm model, despite complete distributional instability. We sampled activations of a given layer and visualized their distributions (also cf. Figure 7).

Figure 11: BN improves training, even at high learning rates.



(a) loss landscape (b) gradient predictiveness (c) "effective" β-smoothness

Figure 4: Analysis of the optimization landscape of VGG networks. At a particular training step, we measure the variation (shaded region) in loss (a) and ℓ_2 changes in the gradient (b) as we move in the gradient direction. The "effective" β-smoothness (c) refers to the maximum difference (in ℓ_2-norm) in gradient over distance moved in that direction. There is a clear improvement in all of these measures in networks with BatchNorm, indicating a more well-behaved loss landscape. (Here, we cap the maximum distance to be $\eta = 0.4\times$ the gradient since for larger steps the standard network just performs worse (see Figure 1). BatchNorm however continues to provide smoothing for even larger distances.) Note that these results are supported by our theoretical findings (Section 4).

Figure 12: BN smoothens the loss landscape.

5.1 On connectedness of sublevel sets in deep learning

This is a theoretical paper [Nguyen(2019)], and I spoke to the author at his poster. I won't discuss too much of the technical aspects, as I have yet to understand the proof itself. The high level take away of the paper is that in over parameterized settings with Relu or Leaky Relu activation functions with a wide first hidden layer, the sublevel sets of the loss landscape are all connected, and all local minima are actually global minima.

5.2 Gradient descent provably optimizes over parameterized neural networks

Another theoretical paper [S. Santurkar(2019)] that I leave to the reader. The authors prove that for a 2 layer Relu network, where the number of hidden units is sufficiently large, then SGD converges to a globally optimal solution with a linear convergence rate for a quadratic loss function (e.g. mean squared error loss).

5.3 Why do larger models generalize better? A theoretical perspective via the XOR problem

The authors look at the XOR problem [Brutzkus and Globerson(2019)]. They begin by noting that even after achieving zero training error, the test error of larger models continues to decrease. The authors show that a 2 layer MLP can solve this with 4 hidden units. They show results with 4 and 100 units and plot the initiaization of the weights for both. The more varied 100 unit network has a more diverse range of weights at initialization, and at convergence finds the global minima, whereas the 4 unit network arrives at a suboptimal local minima.

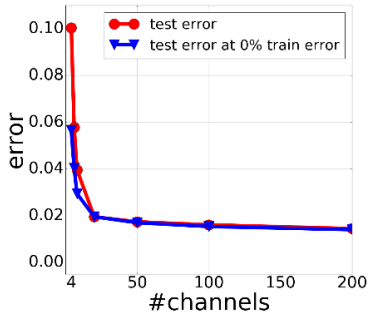
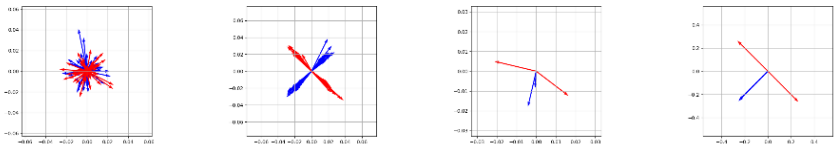


Figure 13: Continued test error reduction in larger models even after zero training error.



(a) (b) (c) (d)

Figure 2: Overparameterization and optimization in the XOR problem. The vectors in blue are the vectors $\boldsymbol{w}_t^{(i)}$ and in red are the vectors $\boldsymbol{u}_t^{(i)}$. (a) Exploration at initialization (t=0) for $k = 50$ (Lemma 3.1) (b) Clustering and convergence to global minimum for $k = 50$ (Lemma 3.2 and Theorem 3.3) (c) Non-sufficient exploration at initialization (t=0) for $k = 2$ (Theorem 3.4). (d) Convergence to local minimum (Theorem 3.4).

Figure 14: Diversity of initial weights and weights at convergence.

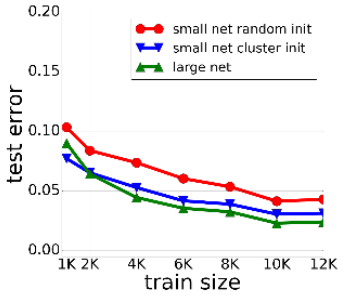
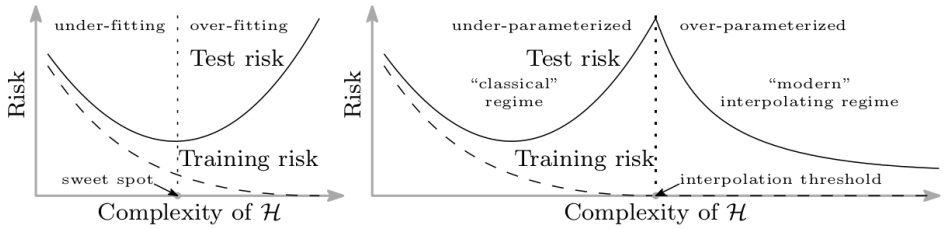


Figure 15: Large network final weights are helpful for initializing smaller networks.

The experimenters also did experiments where they train there large model, do clustering for 4 clusters, and use those cluster centers as weights to initialize a small 4 neuron network. The results show that this initialization dramaticially improves the training of the smaller network, approaching that of the larger network. The authors also provide a lot of theoretical work to show that the loss landscape is much easier for over parameterized networks.

6 Reconciling modern machine learning and the bias-variance trade off



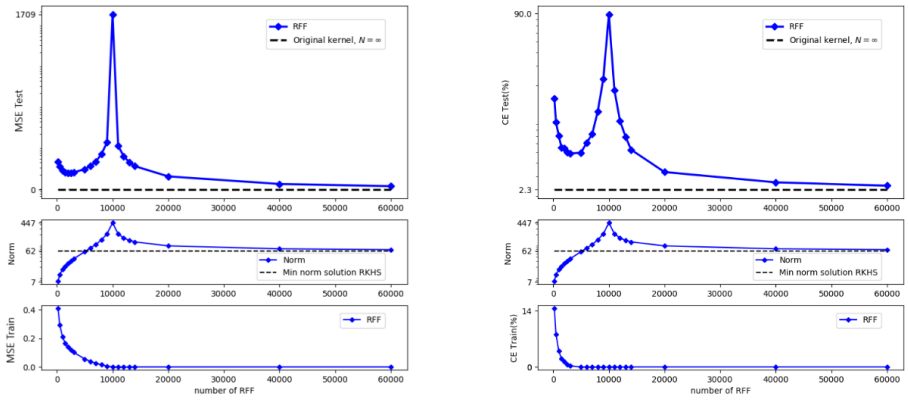
(a) U-shaped "bias-variance" risk curve (b) "double descent" risk curve

Figure 16: Traditional generalization curves vs those for over parameterized networks.

This paper [Belkin et al.(2018)Belkin, Hsu, Ma, and Mandal] begins by noticing that the traditional generalization curve motivating "bias-variance" trade-off in supervised learning theory is no longer applicable for DNNs. Specifically, there exists a point wherein the complexity of the model class is so high that the model class and represent interpolating functions (e.g. can achieve perfect error/accuracy on the trainingset). Once this complexity level is reached, test error continues to decrease, despite training error already reaching zero, even surpassing the performance of the best test error found in the under-parameterized setting (see Figure 17 for a visual explanation).

A main take away of this paper deals with this suprising phenomena: once we reach a sufficiently complex model class to achieve interpolation (perfect train accuracy), why is it that increasing the function class complexity continues to see an improvement in test accuracy? The authors return to regularization theory, which essentially asks: *what is the best model that fits the data, with minimal norm?* The paper argues that this is what is happening. In fact, these over parameterized interpolating functions have minimal norm while also fitting the data perfectly. That is to say, given two classes of functions that can fit the training data perfectly, one that has more parameters (degrees of freedom) has more wiggle room to find a lower norm representation. Hence, counter intuitively, increasing the number of parameters gives the model class more expressive freedom to find an interpolating function with smaller complexity. This is supported by theoretical and empirical results on interpolating functions within a reproducing kernel Hilbert space (RKHS), neural networks and other classifiers (random forests).

The first set of results deals with experiments using kernel methods using random fourier features which have rich theoretical properties. The results show that as the number of RFFs increase (x-axis), we see that the "double U" learning curve described above is found. Around 10000 RFFs are dramatically overfitting which high test error, however as the number of RFFs increase, the test error dramatically decreases. This perfeclty correlates with an increase in function norm followed by a dramatic decrease in function norm, which approaches the smallest possible norm in this RKHS. The results are staggering.



Squared loss Zero-one loss

Figure 2: Test risks, coefficient ℓ_2 norms, and training risks of the RFF model predictors $h_{n,N}$ learned on a subset of MNIST ($n = 10^4$, 10 classes).

Figure 17: RKHS minimal norm is achieved in interpolating function class as complexity increases.

Finally, the authors reproducing similar generalization curves for neural networks and for decision trees, to show that this is not specific to a given class of learning algorithms.

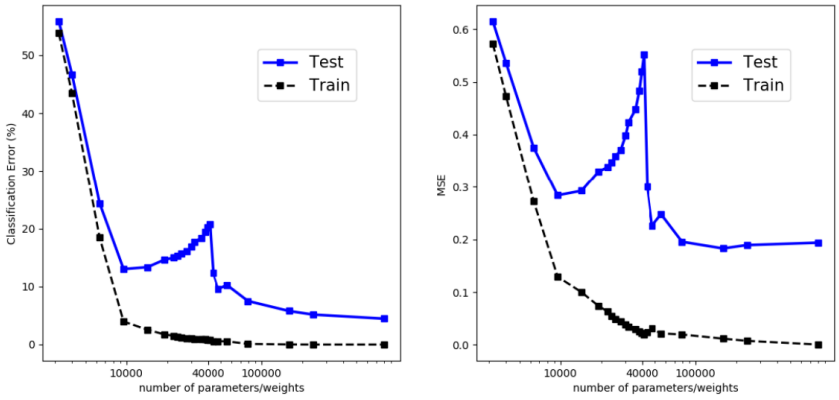


Figure 3: Training and test risks of fully connected neural networks with a single layer of H hidden units, learned on a subset of MNIST ($n = 4 \times 10^3$, $K = 10$ classes; left: zero-one loss; right: squared loss). The number of parameters is $(d+1) \times H + (H+1) \times K$. The interpolation threshold is observed at $n \times K$.

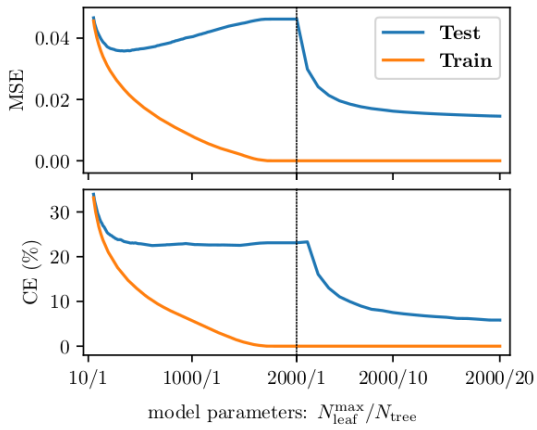


Figure 4: Double-descent risk curve of random forest trained for MNIST ($n = 10^4$)

Figure 18: Generalization curves for neural networks and random forests.

7 The lottery ticket hypothesis: finding sparse trainable neural networks

This paper [Frankle(2019)] begins by noting that there are many studies and techniques that have consistently shown a DNN can be pruned (e.g. for use on a phone) with up to 90 percent of its parameters removed and still achieve comparable performance, however, when a network initialized with the pruned topology, with same initial parameters of the original networks (constrained to the non-pruned parameters), then these sub-networks can often train faster and produce better test error than the original DNN. However, if the same amount of pruning is performed randomly, the models performance is quite worse. The authors put forward the lottery ticket hypothesis: that over parameterized DNN, given an initialization, contain sub-networks that contain *winning tickets* that can easily achieve a global optima.

The approach taken follows:

- Initialize network $f(x;\theta_0)$, $\theta_0 \sim \mathcal{D}_\theta$
- Train until convergence, arriving at parameters θ_t
- Prune off a proportion p of parameters (weights closest in magnitude to 0) and create a mask m
- Reset remaining parameters to initial values θ_0, creating winning ticket $f(x; m \odot \theta_0)$

This describes a one-shot mechanism to prune the network, the authors also describe a multi-step approach, where every so often the network prunes the p least important parameters, and continues to do this iteratively.

The first figure shows (a,b) for LeNet CNN architectures (MNIST vs CIFAR10) at what point the early stopping criteria (e.g. convergence time) would occur given the percentage of weights present for the network. Specifically, for 100 percent, this represents training of the DNN that normally occurs. Then for 41.1 percent, this represents the version of the original DNN network with 100-41.41=58.9 percent of its least important (closest to zero magnitude) parameters pruned from the fully trained network, then this network with the pruned topology is randomly initialized and trained. This plot shows that as more of the parameters are pruned, the longer it takes to convergence, despite the fact that each network contains an optimal *winning ticket* optimal subnetwork. Moreover, the exact winning ticket topology (solid line) is compared to random networks with the same pruning, but random initiatlization (e.g. $f(x; m \odot \theta^{'})$, $\theta^{'} \sim \mathcal{D}_\theta$). Random initialization of the winning ticket topology is significantly slower to learn than the winning ticket network with the same initial parameters. The right most figures show how the final test accuracy is of the networks after pruning of the fully trained network. Hence, it is not the topology itself

that is important, but rather, the topology *given* the initial paramterization. Moreover, for the winning ticket models, it appears there are some parameter settings where the winning ticket sub-network both trains faster than the original network, and achieves better test accuracy.

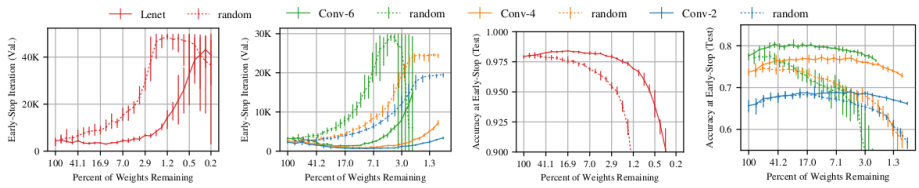


Figure 1: The iteration at which early-stopping would occur (left) and the test accuracy at that iteration (right) of the Lenet architecture for MNIST and the Conv-2, Conv-4, and Conv-6 architectures for CIFAR10 (see Figure 2) when trained starting at various sizes. Dashed lines are randomly sampled sparse networks (average of ten trials). Solid lines are winning tickets (average of five trials).

Figure 19: Generalization curves for neural networks and random forests.

The authors present several other experiments, also looking at iterative pruning, as well as various model architectures. Ultimately, the results are all consistent.

8 Discussion

Measuring the complexity of a DNN model class has become vacuous when attempting to use prior methods such as VC dimension as they apply to the number of parameters of the network. Despite these vacuous bounds, empirically we see that over parameterized DNNs generalize quite well, even after achieving zero error on the training set. Moreover, over parameterized DNNs can learn anything, even complete noise, given enough time. However, evidence suggests that models learn easiest concepts first, most likely in earlier layers, then later layers act to fine tune and add to the earlier representations to learn the more spurious examples. The initial layers carry the brunt of the representation, and if modified is catastrophic for the model, wheras the later layers are robust and change very little from the initial parameters. The wider and deeper the network, the more this phenomena is observed.

Once a model architecture, loss function, data set and initial parameters are set, the loss landscape is fixed. The extra parameters appear to make the loss function smoother and easier to arrive at a local $\iff$ global minimizer. Despite not requiring a vast majority of the parameters for the function itself, the parameters seem required for the loss. Another interesting result is that once the model class contains interpolation functions, by further increasing the number of parameters SGD is able to arrive at the simplest possible interpolating

function. Hence, in some manner, increasing the number of parameters beyond a certain point can actually make the model *simpler*.

Measuring model complexity for over parameterized DNNs is not so straight forward. One measure of model complexity is the minimum description length (MDL) [Blier and Ollivier(2018)]. The number of bits to represent the data using the model, as well as the number of bits to encode the model itself. Since not all parameters may be necessary for over parameterized DNN, the number of effective parameters after pruning, or the intrinsic dimension as proposed above seem like much better measures of complexity of the model, and require fewer bits to encode post-pruning, than the fully parameterized network.

References

[Belkin et al.(2018)Belkin, Hsu, Ma, and Mandal] Mikhail Belkin, Daniel J. Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning and the bias-variance trade-off. *CoRR*, 2018.

[Blier and Ollivier(2018)] Léonard Blier and Yann Ollivier. The description length of deep learning models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2216–2226. Curran Associates, Inc., 2018.

[Brutzkus and Globerson(2019)] Alon Brutzkus and Amir Globerson. Why do larger models generalize better? A theoretical perspective via the XOR problem. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 822–830, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/brutzkus19b.html`.

[C. Li(2018)] R. Liu J. Yosinski C. Li, H. Farkhoor. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018.

[C. Zhang()] Y. Singer C. Zhang, S. Bengio. Are all layers created equal? Technical report.

[D. Arpit(2017)] N. Ballas D. Krueger E. Bengio M.S. Kanwal T. Maharaj A. Fischer A. Courville Y. Bengio S. Lacoste-Julien D. Arpit, S. Jastrzebski. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, 2017.

[Frankle(2019)] J. Frankle. The lottery ticket hypothesis: finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[Nguyen(2019)] Q. Nguyen. On connected sublevel sets in deep learning. In *International Conference on Machine Learning*, 2019.

[S. Santurkar(2019)] A. Ilyas A. Madry S. Santurkar, D. Tsipras. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[S.S. Du(2018)] B. Poczos A. Singh S.S. Du, X. Zhai. How does batch normalization help optimization? In *Neural Information Processing Systems*, 2018.

[Zhang et al.(2017)Zhang, Bengio, Hardt, Recht, and Vinyals] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, 2017.