Project

Overparameterized Machine Learning

Itay Abuhazera and Raz Monsonego

Contents

1 Summary

We chose the following paper: *C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization" , International Conference on Learning Representations (ICLR), 2017.*
Having a good criterion for the generalization error, which is defined as the difference between test error and training error, is crucial in the field of AI due to its potential contributions to model interpretability and the architecture and design of models, and their corresponding learning methods. For this, statistical learning theory has proposed a number of complexity measures, such as the VC dimension (Vapnik, 1998), Rademacher complexity (Bartlett & Mendelson, 2003), and uniform stability (Mukherjee et al., 2002; Bousquet & Eliseeff, 2002; Poggio et al., 2004). Theory also suggests that when the number of parameters is large, some form of regularization is needed to achieve small generalization error. This work shows that the traditional approaches for explaining and controlling the generalization error are insufficient due to their inability to distinguish between different neural networks that have radically different generalization performances, specifically in the realm of over-parameterized models. The paper suggests rethinking the current generalization predictors and investing more effort in developing more accurate explanations for the generalization error of a model. For starters, an experiment is conducted, showing that deep neural networks can achieve zero training error on randomly labeled data. The test error is obviously a random chance since there is no correlation between the data and the labels in training. This means that the effective capacity of the used neural networks is sufficient for memorizing the entire dataset, thus interpolation of the data-set does not require correlation between the data-set samples and their labels. More experiments were conducted using Gaussian noise, random pixels, shuffled pixels, or partially corrupted labels as the training data, which yielded similar results, i.e., perfect fitting was achieved, no matter how uncorrelated the data was to its label. These experiments were executed with several model architectures on the CIFAR10 and ImageNet datasets. These experiments also show that the training of a model on one of these perturbed datasets is relatively easy (similar to with the originally labeled data, up to a small constant factor), which may come as a surprise. These observations rule out the common explanations for generalization, as we further explain. Uniform stability is ruled out since it is independent of the labeling of the training data, which as shown above, may have a dramatic effect on the generalization error of the model. VC dimension and Rademacher complexity are ruled out since they do not fit the overparameterized regime, which is usually the case in modern times, as their input is becomes trivial and non-informative.
It is shown that explicit regularization, such as weight decay, dropout, and data augmentation, does not adequately explain the generalization error of neural networks, as they improve the generalization by a relatively small margin at times, and are not necessary for good generalization,

nor do they promise avoiding bad generalization (as seen in the experiments depicted above). For instance, the model choice seems to have a much higher influence on the generalization error of the system, as seen in figure 1.c in the paper (system being: data, model, hyperparameters, optimizer). One of the **major findings** of the paper is the following theorem: *A simple two-layer ReLU network with $p = 2n + d$ parameters can express any labeling of any dataset of size n, where the samples are of dimension d.* Previous works showed this for networks with far more parameters, namely $O(dn)$. It is also shown that generally for a network of depth k, $O(n/k)$ parameters are needed for expressing all possible labelings of the training data-set.
A behavior comparison of deep neural networks with and without regularizers is held. In this comparison, the regularizers compared are data augmentation, weight decay (i.e., ridge regression, ℓ_2 regularizer on the weights), and dropout, with the Inception V3, Alexnet (smaller case of Inception) and MLP architectures on the CIFAR10 and ImageNet datasets. It is shown that although the regularizers help generalization, they are not necessary for it, since the difference between the generalization error of the models with and without them is relatively small. Furthermore, it is shown that data augmentation had the most effect on generalization relative to the other regularization techniques. Interestingly, and somewhat surprisingly, it is observed that batch normalization and early stopping improve the generalization error of a model as well, thus can be seen as regularizers themselves in some cases (other cases were unaffected). They are less robust and effective compared to the above techniques but introduce other qualities, such as stability of the training process, which is seen in figure 2.b for batch normalization. The conclusion of these experiments is that regularizers clearly affect the generalization ability of a model in most cases, yet are probably not the fundamental reason for it. An important observation made is that **implicit regularization** takes place in many of the optimization processes in use. Specifically, due to the nature of the SGD algorithm, the result of applying it on linear models is the minimum ℓ_2 norm solution. This suggests that some implicit regularization is applied via the SGD algorithm, which itself may help the generalization of the learned model. Unfortunately, more experiments showed that the ℓ_2-norm is not predictive of generalization performance, as they observed cases where the generalization performance improved while the ℓ_2-norm increased. Overall, this paper shows some of the characteristics of commonly used neural networks with respect to their generalization performance and convergence properties for different permutations of the data and its labels, and for the commonly used regularization techniques. It concludes that more research is needed to better define and characterize the generalization performance of deep learning models due to the unsatisfying ability of the prominent tools at hand to explain it. Since good generalization performance is the goal of most modern deep neural networks, such a characterization will have immense impact on the world of AI, for both theoretical and practical practitioners.

2 Relation to Course Material

The paper "*Understanding Deep Learning Requires Rethinking Generalization*" by Zhang et al. (2017) challenges traditional views on how deep neural networks generalize, particularly in the context of overparameterized models. This paper is closely related to several topics discussed in our course, particularly those concerning model capacity, regularization, and their effect on the generalization performance of the model.

One connection between the paper and our course is the exploration of model capacity. The paper demonstrates that deep neural networks can memorize entire datasets, even with random labels, achieving zero training error. This aligns with the course material, where overparameterized models were shown to perfectly learn the training data, even when it was aggressively perturbed. Lecture 9 introduced the concept of effective model complexity (EMC), explaining how model complexity can account for the double descent phenomenon, where test error first worsens and then improves as complexity increases. Both the paper and the course emphasize that while these models can memorize the training data, the mechanisms that enable good generalization remain an unresolved area of research.

Both the paper and the course place significant emphasis on regularization methods—both explicit and implicit—and their role in generalization. Explicit regularization techniques, such as weight decay (ridge regression), aim to control model complexity by penalizing large weights. Although weight decay can improve generalization, both the paper and the course suggest that it is not the primary factor responsible for good generalization in deep learning models.

In addition to weight decay, another explicit regularization technique discussed in both the paper and the course is data augmentation. Data augmentation improves generalization by introducing transformations that expand the training set without altering the underlying distribution. In the course, we saw how data augmentation can shift the interpolation threshold and the peak of the double descent curve to larger model widths, enhancing generalization. Similarly, the paper highlights the powerful effect of data augmentation. The authors demonstrate that even when explicit regularizers are turned off, models trained with data augmentation still generalize well.

Both the paper and the course emphasize implicit regularization methods, which rely on the natural properties of the training process to guide models toward generalization without explicitly modifying the model. Early stopping is one such implicit regularization technique. In the paper, early stopping is shown to potentially improve generalization by halting the training process before the model begins to overfit the training data. However, the paper also notes that early stopping is not always effective. The course similarly presents early stopping as an implicit regularizer, particularly in the context of the double descent phenomenon. It shows that early stopping prevents the training error from reaching zero, smoothing the test error and mitigating the double descent peak. This nuanced view suggests that the effectiveness of early stopping as an implicit regularizer depends on factors such as the dataset and the learning task.

Another implicit regularization method emphasized in both the paper and the course is stochastic gradient descent (SGD). SGD naturally guides models toward generalizable solutions by favoring low-complexity solutions due to its iterative optimization process. The paper shows that SGD often converges to solutions with minimal norm, even in the absence of explicit regularization techniques like weight decay. For example, in linear models, SGD converges to the minimum l_2-norm solution, which generalizes well despite the model's ability to perfectly fit the training data. This aligns with the course material, which also discusses how SGD acts as an implicit regularizer, especially in overparameterized scenarios. Both the paper and the course conclude that the regularization effect of SGD is often sufficient to enable good generalization, even when explicit regularization techniques are absent.

Ultimately, while both implicit and explicit regularization techniques can improve generalization, neither the paper nor the course considers them strictly necessary for achieving good generalization. Overparameterized models frequently generalize well without relying on explicit regularization, highlighting the complexity of generalization in modern neural networks and the importance of the training process itself.

One difference arises from the lack of discussion in the paper on the double descent phenomenon, which was a focal point in our course discussions. The double descent curve, where increasing model complexity initially improves, then worsens, and then improves the generalization performance, forms a surprising and very interesting regime of research of modern deep neural networks. Our course explored this phenomenon as a key area of study, providing a theoretical framework together with empirical results, that the paper does not address.

In conclusion, the paper "*Understanding Deep Learning Requires Rethinking Generalization*" is **closely related** to the topics covered in our course, particularly in its examination of the part of model capacity and regularization in the generalization performance of a model, and its characterization. The paper extends the understanding of generalization performance and its main factors, via many interesting empirical observations, backed up with relatively minimal theoretical foundations, which the course is highly focused on. In that sense, the paper and the course are complementary.

Overall, it is clear that there is a need for further research to bridge the gap between empirical observations and theoretical understanding of modern deep neural networks, and specifically the characterization of their generalization performance. The advancements made in the years 2017-2024 are expressed in the course material relative to the paper, and demonstrate how much progress has been made, yet much remains to be discovered in this interesting field.

3 Extension

3.1 Motivation

The motivation to our extension of the paper is the enrichment of its foundations and experimental findings, in order to make it more robust and complete.
Specifically, we felt that the authors should have covered a wider span of the factors affecting the generalization performance of a learned model.
Specifically, the main component we felt the paper is missing, is the effect of the **number of parameters** in the model on the **generalization performance** of the model. As we learned in the course, as the number of parameters increases, we are expecting to witness the **double-descent** phenomenon, surprisingly showing an improvement in the generalization performance after hitting the interpolation threshold! This addition is important in the discussion of generalization, as it has some highly impactful implications on model design and architecture per use-case, and it intrigues researchers to investigate a whole new regime - the highly overparameterized regime.

3.2 Technical details

In order to show the effect of the number of parameters on the generalization performance, we wrote code that implements a basic Convolutional Neural Network (CNN) with a varying amount of parameters, trained it on a subset of the CIFAR-10 data-set and analysed the results.
Specifically, we used the PyTorch framework to implement the CNN following well-known design recommendations, which consists of the following layers:

1. Convolutional layer, consisting of 6 kernels of size 5×5, followed by a ReLU activation function. The size of this layer is not changed throughout the experiments.
2. Max-Pooling layer, with a kernel of size 2 and stride 2.
3. Second convolutional layer of varying size (other than the input size of 6 channels), followed by a ReLU activation function.
4. Second Max-Pooling layer, again of size 2 and stride 2.
5. Flattening layer.
6. Fully connected layer of varying size, followed by a ReLU activation function.
7. Second fully connected layer of varying size, followed by a ReLU activation function.

8. Third fully connected layer. The first dimension of this layer varies according to the output dimension of the previous layer. The output is of the size of the class space, which in our case is always 4.

We use SGD as our iterative optimizer.
* Note that the convolutional layers use a default value for the stride of 1.
Regarding the data-set, we used a subset of CIFAR-10, comprised of the four classes {'car', 'frog', 'horse', 'ship'} (for reasons described in 3.4), containing 20,000 training samples, and 4,000 samples in the test data-set.
We seek to train models with varying number of parameters with an approximately uniform jump, such that the data-set size is roughly in the middle of this range. In order to achieve the different model sizes (i.e., # of parameters), we change the sizes of the layers of the model, keeping all layers (in most cases). A full listing of the models we trained, together with their results can be found in Table 1 in the appendix.
All code can be found in https://github.com/ItayAbuhazera/OverparameterizationProject, where the main files we worked with are "cnn_loss_generalization_OML_19488.ipynb", its variants for other model sizes, and a notebook named "train_plot.ipynb" which trains all of our models, retrains the one that need retraining, and plots the results of the train and test error.

3.3 Results

We summarize our results in the following graph:

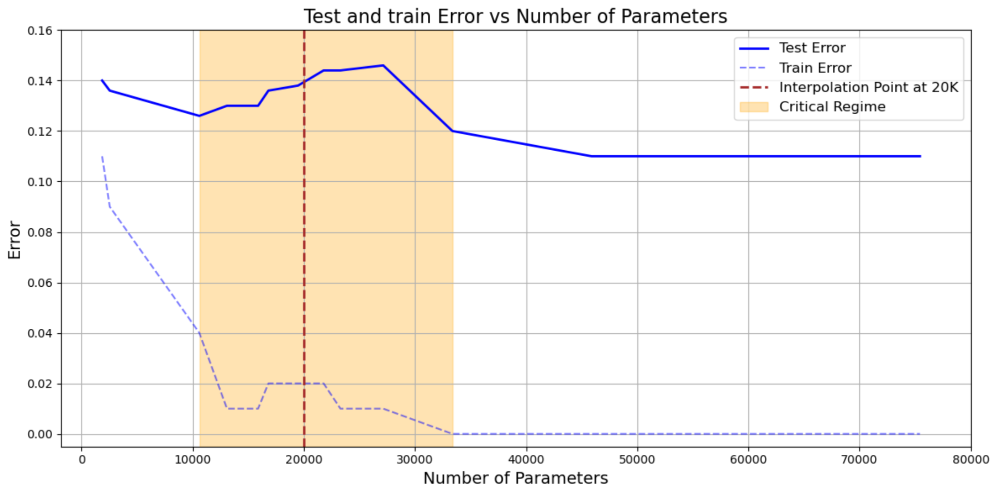

Figure 1: Test and train error vs. Number of parameters

The table with the data-points that comprise the above graph can be found in Table 1 in the appendix. We see an interesting result, that goes hand by hand with what we learned in the course - the double descent phenomenon in the test error! The classical regime of the bias-variance trade-off up until the interpolation threshold in which the test error declines and then rises, together with the modern regime of a larger model in which the test error declines again, are clear and indeed demonstrate that larger models, even in the over-parameterized regime, yield better generalization performance!
We also see that the interpolation threshold, i.e., the threshold of the number of parameters for which the model perfectly learns the training data and gives 0 training error, is larger than the size of the training data-set. This is an interesting phenomenon, yet expected for this case, as we learned in the course as well.
This factor (increasing the model size, i.e., the number of parameters in the model) was not covered by the paper at all, missing a key observation of what happens to the generalization performance of a model as the model size increases more and more.

3.4 Challenges

We had a number of challenges throughout our extension work:

- The first thing we needed was to implement a CNN that would be strong enough to show the results we wanted, but not too big for us to train locally on our computers. Due to the nature of our extension experiment, there is also a connection between the number of samples in the data-set and the amount of parameters in the models we wish to train. Specifically, we wish to train models with a range of number of parameters such that the data-set size is somewhere in the middle.
  After some research, we decided on a good CNN design that satisfied us.
  We wanted to use the CIFAR-10 data-set, but noticed it has 60,000 samples, for which we need models with $\sim 60,000$ parameters. This yielded a highly resource-consuming (time, cpu, memory) training process that we could not afford to perform locally.
  We thus found a sweet-spot of 4 classes of the CIFAR-10 data-set, containing 20,000 samples (5,000 from each class), together with a test data-set of 4,000 samples (1,000 from each class), such that the range of model parameters we wished to test for was feasible via our local resources (still took several days of training).

- Another difficulty we ran into was that we initially got relatively noisy results, in the sense that the test error (i.e., generalization performance) had a clear tending towards the double-descent pattern, but with high fluctuations in the intermediate results. We solved this by re-iterating the learning process on our learned models, i.e., fine-tuned them more and more, to get a clearer and less noisy result.

3.5 Significance

The significance of our findings w.r.t the paper is high, since we show the dependency of the generalization performance of a learned model on a factor that was not even discussed in the paper, and has some highly surprising and meaningful results w.r.t the subject the paper is dealing with - generalization error!

In the time of the writing of the paper (2017), the realm of ”overfitted” and overparameterized models was not yet appropriately looked at (as can be seen from the findings of the paper as well - as their main point is that more research should be done in this realm of deep learning), which explains why they did not analyze this connection, but rather stopped at noticing that perfect fitting over randomly labeled data can be quite easily achieved, and that regularization can indeed help overfitted models to generalize, but are not the main factor for it.

Our findings explicitly show the effect of the number of parameters of the model on its generalization performance. Although this does not yet explain generalization, it is an important relationship to be aware of when exploring this field, especially due to its interesting and surprising results for highly overparameterized models, achieving higher generalization performance than and under-parameterized model.

Appendices

A Results Tables

A.1 Table of Results

The following table summarizes the results obtained from training different models on the CIFAR-10 subset:

Table 1: Results from different models trained on CIFAR-10 subset

Param Count	Train Error	Test Error	Train Loss	Test Loss	Epochs
1865	0.11	0.14	$1.06 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$	40
2550	0.09	0.14	$2.51 \cdot 10^{-3}$	$6.8 \cdot 10^{-2}$	50
10608	0.04	0.13	$8.34 \cdot 10^{-5}$	$3.02 \cdot 10^{-4}$	40
13093	0.01	0.13	$1.14 \cdot 10^{-3}$	$1.15 \cdot 10^{0}$	90
15895	0.01	0.13	$4.42 \cdot 10^{-4}$	$1.08 \cdot 10^{0}$	90
16819	0.02	0.14	$2.52 \cdot 10^{-3}$	$3 \cdot 10^{-1}$	100
19488	0.02	0.14	$2.41 \cdot 10^{-6}$	$2.75 \cdot 10^{-1}$	40
21785	0.02	0.14	$9.6 \cdot 10^{-6}$	$4.76 \cdot 10^{-4}$	40
23299	0.01	0.15	$4.87 \cdot 10^{-5}$	$9.12 \cdot 10^{-6}$	40
27150	0.01	0.14	$5.99 \cdot 10^{-6}$	$1.13 \cdot 10^{-4}$	40
33370	0	0.12	$2.12 \cdot 10^{-4}$	$1.02 \cdot 10^{0}$	90
45892	0	0.11	$2.74 \cdot 10^{-5}$	$7.8 \cdot 10^{-1}$	70
54694	0	0.11	$4.45 \cdot 10^{-5}$	$1.03 \cdot 10^{0}$	100
62716	0	0.11	$6.19 \cdot 10^{-5}$	$6.65 \cdot 10^{-1}$	100
75426	0	0.11	$4.09 \cdot 10^{-5}$	$8.85 \cdot 10^{-1}$	100

B Detailed Network Architectures

B.1 Network Architectures

The following table provides detailed architectures of the models we trained in this work:

Name	**First Convolution Layer**	**Second Convolution Layer**	**First Fully Connected (Hidden Layer)**	**Second Fully Connected (Hidden Layer)**	**Third Fully Connected (Hidden Layer)**
Net 3305	6 channels, 5x5 kernel, ReLU	5 channels, 5x5 kernel, ReLU	Input: 5x5x5, Output: 23 units, ReLU	Input: 23 units, Output: 8 units, ReLU	Input: 8 units, Output: 4 units
Net 5695	6 channels, 5x5 kernel, ReLU	5 channels, 5x5 kernel, ReLU	Input: 5x5x5, Output: 30 units, ReLU	Input: 30 units, Output: 20 units, ReLU	Input: 20 units, Output: 4 units
Net 10608	6 channels, 5x5 kernel, ReLU	6 channels, 5x5 kernel, ReLU	Input: 6x5x5, Output: 40 units, ReLU	Input: 40 units, Output: 20 units, ReLU	Input: 20 units, Output: 4 units
Net 13093	6 channels, 5x5 kernel, ReLU	8 channels, 5x5 kernel, ReLU	Input: 8x5x5, Output: 50 units, ReLU	Input: 50 units, Output: 25 units, ReLU	Input: 25 units, Output: 4 units
Net 15895	6 channels, 5x5 kernel, ReLU	8 channels, 5x5 kernel, ReLU	Input: 8x5x5, Output: 50 units, ReLU	Input: 50 units, Output: 25 units, ReLU	Input: 25 units, Output: 4 units
Net 16819	6 channels, 5x5 kernel, ReLU	9 channels, 5x5 kernel, ReLU	Input: 9x5x5, Output: 80 units, ReLU	Input: 80 units, Output: 40 units, ReLU	Input: 40 units, Output: 4 units
Net 23299	6 channels, 5x5 kernel, ReLU	9 channels, 5x5 kernel, ReLU	Input: 9x5x5, Output: 80 units, ReLU	Input: 80 units, Output: 40 units, ReLU	Input: 40 units, Output: 4 units
Net 27150	6 channels, 5x5 kernel, ReLU	10 channels, 5x5 kernel, ReLU	Input: 10x5x5, Output: 80 units, ReLU	Input: 80 units, Output: 60 units, ReLU	Input: 60 units, Output: 4 units

Table 2: Detailed Network Architectures

Name	**First Convolution Layer**	**Second Convolution Layer**	**First Fully Connected (Hidden Layer)**	**Second Fully Connected (Hidden Layer)**	**Third Fully Connected (Hidden Layer)**
Net 30397	6 channels, 5x5 kernel, ReLU	10 channels, 5x5 kernel, ReLU	Input: 10x5x5, Output: 92 units, ReLU	Input: 92 units, Output: 55 units, ReLU	Input: 55 units, Output: 4 units
Net 33370	6 channels, 5x5 kernel, ReLU	10 channels, 5x5 kernel, ReLU	Input: 10x5x5, Output: 100 units, ReLU	Input: 100 units, Output: 60 units, ReLU	Input: 60 units, Output: 4 units
Net 36480	6 channels, 5x5 kernel, ReLU	10 channels, 5x5 kernel, ReLU	Input: 10x5x5, Output: 110 units, ReLU	Input: 110 units, Output: 60 units, ReLU	Input: 60 units, Output: 4 units
Net 39590	6 channels, 5x5 kernel, ReLU	10 channels, 5x5 kernel, ReLU	Input: 10x5x5, Output: 120 units, ReLU	Input: 120 units, Output: 60 units, ReLU	Input: 60 units, Output: 4 units
Net 42052	6 channels, 5x5 kernel, ReLU	12 channels, 5x5 kernel, ReLU	Input: 12x5x5, Output: 110 units, ReLU	Input: 110 units, Output: 58 units, ReLU	Input: 58 units, Output: 4 units
Net 45892	6 channels, 5x5 kernel, ReLU	12 channels, 5x5 kernel, ReLU	Input: 12x5x5, Output: 120 units, ReLU	Input: 120 units, Output: 60 units, ReLU	Input: 60 units, Output: 4 units
Net 54694	6 channels, 5x5 kernel, ReLU	14 channels, 5x5 kernel, ReLU	Input: 14x5x5, Output: 120 units, ReLU	Input: 120 units, Output: 80 units, ReLU	Input: 80 units, Output: 4 units
Net 62716	6 channels, 5x5 kernel, ReLU	16 channels, 5x5 kernel, ReLU	Input: 16x5x5, Output: 128 units, ReLU	Input: 128 units, Output: 64 units, ReLU	Input: 64 units, Output: 4 units
Net 75426	6 channels, 5x5 kernel, ReLU	16 channels, 5x5 kernel, ReLU	Input: 16x5x5, Output: 150 units, ReLU	Input: 150 units, Output: 80 units, ReLU	Input: 80 units, Output: 4 units

Table 3: Detailed Network Architectures