Math522 Project

Bryce Lunceford, Whitney Anderson, Sebastián Valencia, Erika Ibarra

April 2023

1 Introduction

The standard picture that is shown when people talk about over-fitting is something like the following:

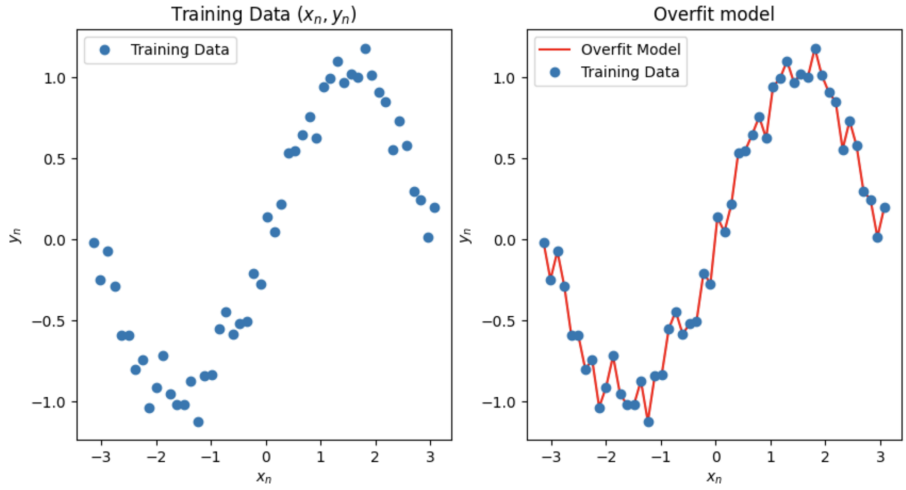


Figure 1: The over-fit model learns to hit the training data points exactly. In other words, the model is learning not only to approximate the underlying function, but also the noise in the data.

It is then natural to ask if this picture is really the case. To make things more concrete, suppose that data is generated according to some process:

$$y_n = f(x_n) + \varepsilon_n$$

where f is a deterministic function and ε_n is a random error term with mean 0. Now, train a neural network $\hat{f}_\theta$ with parameters θ and overfit it to this data. The question we ask is this: If X is a random variable that is distributed the same way as the distribution for the x_n's, is it true that $\hat{f}_\theta(X) - f(X) \sim \varepsilon$? In other words, is the residual of the overfit model on new data distributed the same way as the error term?

If the answer to this question is yes, we hypothesize that this may be an explanation for the phenomenon of deep double descent. In this project, we attempt to answer the above question, and investigate deep double descent in the process.

2 Background on Double Descent

In talking about double descent, we must first explore bias-variance trade off. A common issue in problems where one is trying to find parameters to fit existing data, bias-variance trade off happens when one needs to choose between minimizing the variance or the bias in their model. [Sin18] In prioritizing limiting the variance in a model, one can overgeneralize and undervalue the information stored in the training data. In contrast, limiting the bias in favor of variance can cause the model to learn too much from irregularities in training data, which would cause the model to fail to generalize and perform poorly on test data.

From a statistics perspective, one would expect the net error on a model to dip as number of parameters increase, and then increase again as the model becomes overparameterized. Machine learning suggests that the more parameters, the better for obtaining an accurate model and reducing error.

For some models, what happens instead is a bizarre combo of the two. The model error dips and rises according to statistics, but at some breaking point, it goes back down, often to a lower error than the previous dip, which coincides with what machine learning suggests. The bizarre contour of the relationship between error and model complexity is called double descent, and is depicted in the figure below.

Figure 2: Depiction of Double Descent, Nakkiran et al.

3 Literature Review

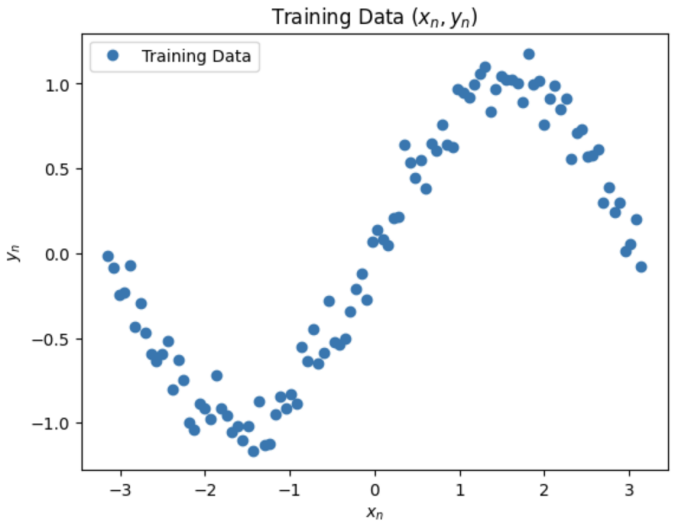
As for the existing literature to explain this phenomenon, it seems that double descent generally occurs when overfitting data. With overfit, or high variance in our model, we expect the error to not be calculated coherently anymore, as the training data merely is memorized, and the network doesn't learn properly. When having such complex models that overfit data, we naturally would reduce parameters to simplify the model and reduce overfit. However, certain cases where double descent occurs show that if we keep the model complex enough, eventually the error levels off, and the model performs its best. Even though a given model can be extremely overfitted, it might have achieved its best performance, and it has done so during this second descent. Since this appears to be a contradiction to the bias-variance trade off, it is still being studied and tested for a more robust understanding.

Nakkiran et al. reported double descent with respect to network width (model-wise), epochs (epoch-wise) and sample size (sample-wise). Throughout the paper, they demonstrated that double descent is a robust phenomena that occurs in various datasets with and without noise e.g. (CIFAR-10, CIFAR-100, IWSLT'14 de-en). Furthermore, they introduced the concept of the effective model complexity as the maximum number of samples on which the training procedure achieves on average 0 training error, and used this concept to explain when double descent occurs. In our paper, we looked at model-wise double descent and how model width, model depth, and noise may affect double descent.

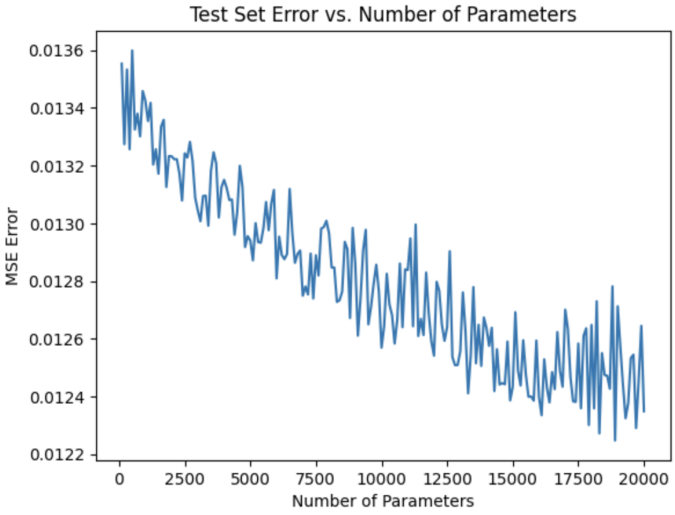
4 Increasing Network Width

To start out, we generated 100 data points to train the models on. The data was generated according to the following process:

$$
\begin{aligned}
x_n &\sim \text{Uniform}(-\pi, \pi) \\
\varepsilon_n &\sim \text{Normal}(0, 0.1) \\
y_n &= \sin(x_n) + \varepsilon_n
\end{aligned}
$$



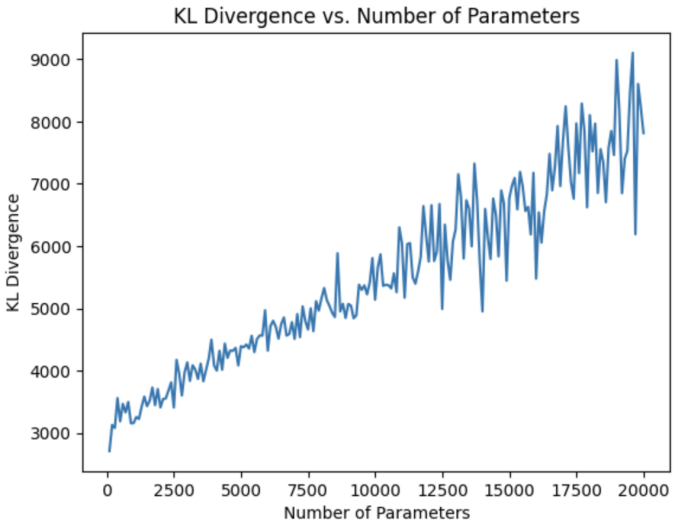
Then, we trained $\sim$ 200 neural networks with a single hidden layer. We varied the widths of the hidden layer of the networks from 100 to 20,000. Then, using new data that was generated in the same way as the training data, we computed the residuals of the models on the new data. The result was the following:



Note that there is a clear trend that shows that as the number of parameters increases, the magnitude of the residuals decreases. This is not what we were expecting to see, as the standard picture of overfitting is that the model should learn to exactly hit the training data. This would suggest that the residuals should increase with the number of parameters *unless* double descent is happening. However, we did not see double descent in this case either.

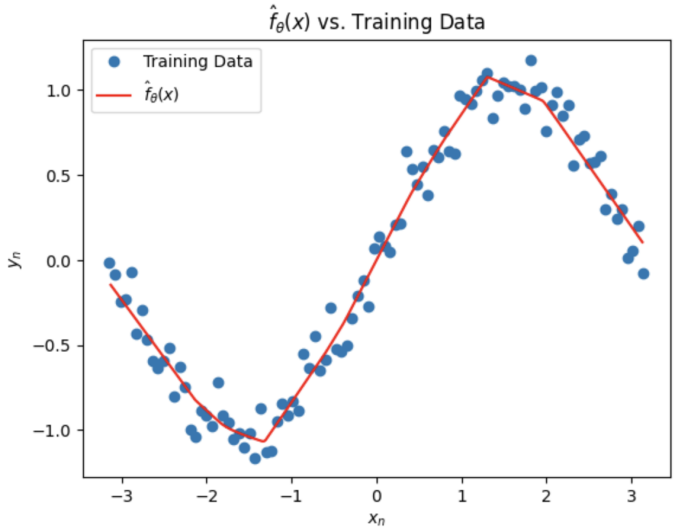
This led us into investigating under what circumstances double descent occurs. This will be addressed later in the report.

To see whether $\hat{f}_\theta(X) - f(X) \sim \varepsilon$, we plotted the KL divergence between the distribution of $\hat{f}_\theta(X) - f(X)$ and the distribution of ε as the number of parameters increased. This gave the following results:



As can be seen in the figure, the KL divergence actually *increases* as the number of parameters increases. This would suggest that the distribution of $\hat{f}_\theta(X) - f(X)$ is not converging to the distribution of ε, at least for a single hidden layer network.

Looking at the plot of $\hat{f}_\theta(X)$ for the $20,000$-wide model shows why this is the case.

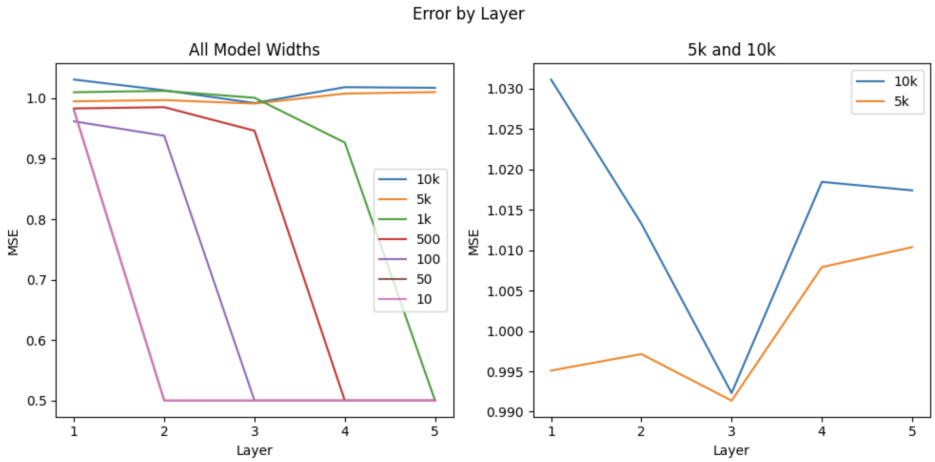


It appears that the model overfitting is not behaving as the standard picture would suggest. Instead of learning to hit the training data exactly, it is learning to approximate the underlying function. This is why the KL divergence is increasing as the number of parameters increases.

5 Increasing Network Depth

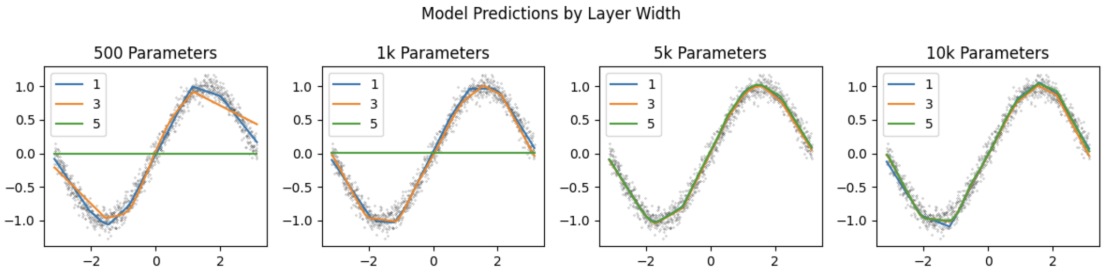
In addition to examining model behavior by width, we also explored varying the model depths with a given fixed hidden layer width. Similarly to the approach of increasing network width, we began with 100 data points sampled from the same distribution as training data. Using uniform widths of 10, 50, 100, 500, 1k, 5k, and 10k, we built models with hidden layers ranging from 1 to 5.

For testing data, we drew 1000 points from the same distribution and calculated the mean squared error between the model prediction and the draw for width and each number of layers, In the figure below, one can see how the testing error drops as layers are added.

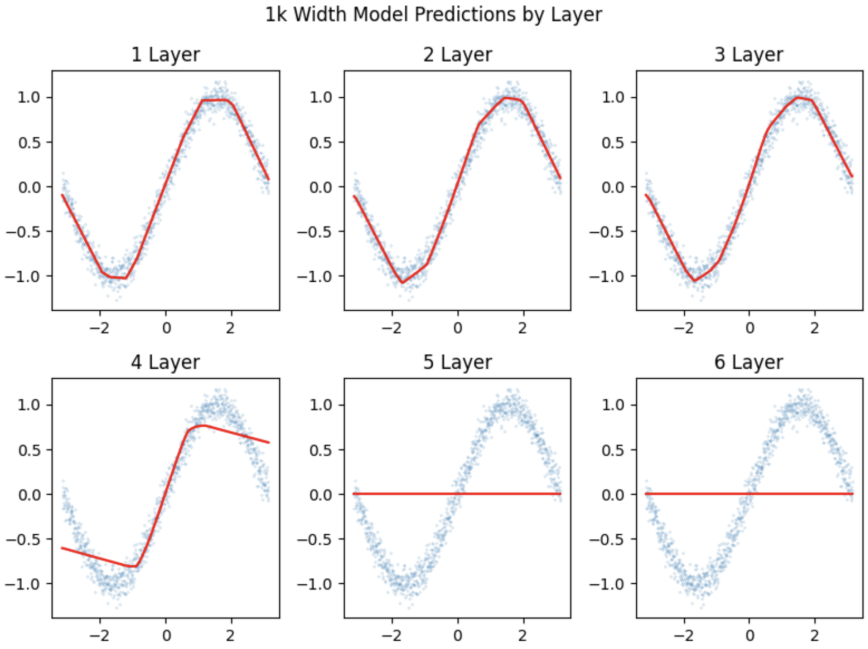


Looking at just the left figure, there appears to be double descent occurring in the models that have 5k and 10k width layers. Unfortunately because of training time and computational expense we cannot see the tail beyond 5 layers to verify that the error would descent again to a minimum less than what occurs at layer 3, but the results could be promising.

We further compared the quality of these models by plotting their predictions on our testing data as seen in the figure below.



With the 5k and 10k models however, the model is able to stay close to the distribution of data points. In the models with layer widths 1k and below, however, we can see that at a certain point all of them overgeneralize and stop remotely following the curve, just guessing the average (f(x)=0) for any potential input. This is clearly illustrated in the comparison below between the 1k width models.



This begs the question - did we actually did find double descent with the 10k and 5k width models? Or is it that as we continue to add layers at these widths these wide models will also overgeneralize into guessing the average like the 1k and narrower models? Whatever the case, its clear that the error of the 1k and below models doesn't at all distribute normally as layers are added, meaning we have yet to support our original hypothesis with our findings and must explore potential causes of double descent through different approaches.

6 Recreate and Reverse Engineer

Rather than train on few data points with wide models, we decided to do a different approach and instead use narrow layers with many datapoints. To capturing the smoothness of the sin function, we evaluated 10,000 datapoints in the domain ranging from $-\pi$ to π. We then proceeded to recreate what we think is double descent by experimenting with various network widths. Finally, we tested double descent with respect to variance and network depth.

Model Width

Wide Search

First, we tried to identify double descent on a wide network, i.e. inputting large numbers for hidden layers, such as *network_width =[1,5,10,15,20,30,40,50,100,200,500,1000]*. We did not observe double descent.

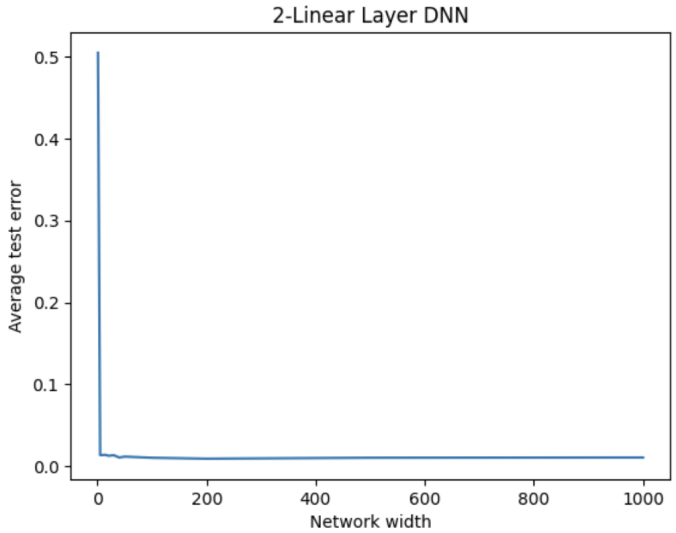


Figure 3: Network_width =[1,5,10,15,20,30,40,50,100,200,500,1000]. Noise variance: 0.01. Epochs: 1000. Datapoints:10,000. 2 Linear-layers

Narrow Search

Next, given that we failed to identify double descent using large network widths, we narrowed our search to a smaller neural network. Here, we used *network_width =[1,5,10,15,20]* for the widths. We also looked at different iterations where we took the mean loss per width. Starting with 1 iteration, we successfully encountered a double descent behavior, with an onset at 5 and a interpolation threshold around 6-7. We then increased the iterations to 3, and finally to 20 iterations per width. The model successfully reproduced a smooth double descent graph.

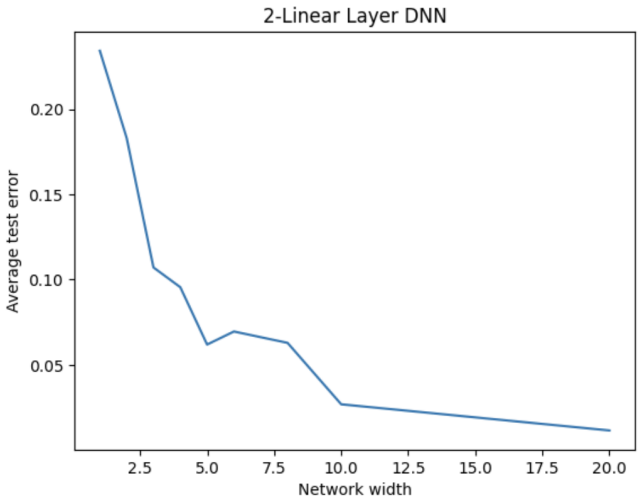


Figure 4: Network_width=[1,5,10,15,20]. Noise variance: 0.01. Epochs: 1000. Datapoints:10,000. 2 Linear-layers.

Noise Variance

All previous results where achieved with a variance of 0.01. Changing the variance for the distribution the noise is based from, lead to interesting results. We can still observe the double descent behavior. We tested this with new variances of 0.05 and 0.1. Noting that the interpolation threshold switched from 7 to 5.

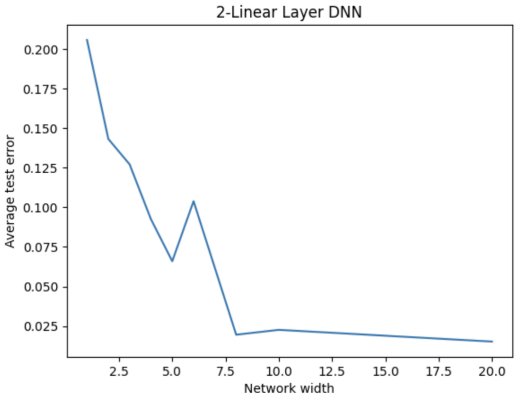


Figure 5: Variance:0.05

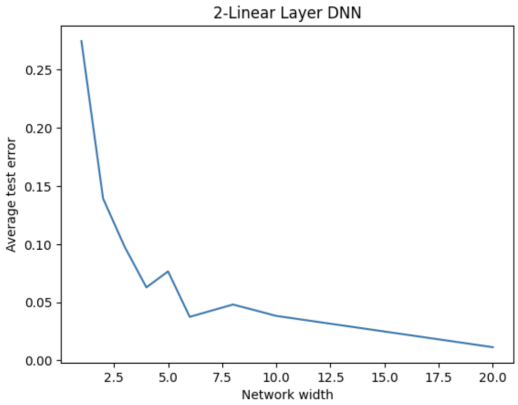


Figure 6: Variance:0.1

7 Conclusion

Double descent is still something that requires further exploration. Our original intuition was that to observe double descent, it would require a complex model that overfits the data, and that if we increased that complexity, the error would go back down. However, we were only able to consistently provoke the phenomenon with many data points and narrow layers as opposed to few points and wide and deep layers. If our interpretation of our own results is correct, this thwarts our original assumptions about what conditions we assumed we needed in order for double descent to appear.

We recommend further analysis to understand the relationships between input parameters (number of data points, number of layers, width of layers) and if and where those second double descent interpolation peaks happen.

References

[Nak+19] Preetum Nakkiran et al. "Deep Double Descent: Where Bigger Models and More Data Hurt". In: (2019). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1912.02292. URL: https://arxiv.org/abs/1912.02292 (visited on 04/27/2023).

[Sin18] Seema Singh. *Understanding the bias-variance tradeoff*. Oct. 2018. URL: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229.