Deep Double-Descent: Phenomenon Change in Different Settings

Yunhang Lin UNI: yl4860
Shiyu Wang UNI: sw3601
Lei Lyu UNI: ll3433
Yinghua Yang UNI: yy3179

June 13, 2022

Abstract

Triggered by the double descent phenomenon achieved from random forest classifier in our previous homework and the great performance of over-fitting deep neural network, we are extremely interested in figure out the double descent phenomenon in deep learning. In our project, we conduct interesting experiments on the generalization error as a function of model size, training epochs, dataset size, and label noise. Besides, we also study on the effect of injecting noise in the dataset and adding gradient noise (Stochastic gradient Langevin dynamics) on the double descent phenomenon.

Keywords–Double Descent, Deep Learning, Neural Network, Overfitting

1 Introduction

Machine learning is extremely popular in today's world by reason of its satisfying prediction result. The process is that Machine Learning model is trained through learning the pattern of training data and then it is applied on unseen test data to make predictions. With the development of technology, deep learning has gradually become the focus of attention. Deep learning relies on huge artificial neural networks, often with thousands of parameters. Artificial neural networks usually contain input, output and hidden layers, and computers can rely on artificial neural networks to process the input training data to achieve their own learning, but as the width and depth of neural networks and various optimization algorithms emerge, we find that trained networks do not always perform as well as we expect in test sets.

While focusing on the good result of training data, it is equally important to take notice of the performance of test data. Small error on training data cannot guarantee the high accuracy on test data. The whole progress can be divided into 2 phases: under-fitting phase and over-fitting phase. The training error decreases when the dimension of data increases. However, the test error curve has the different picture. In the under-fitting phase, the test error decreases with the number of parameters increase. The following phase in which the test error increases is known as over-fitting. The graph of test error is shown as a curve like U-shape in Fig.1.

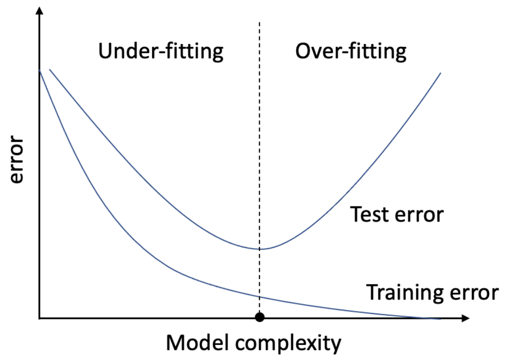


Figure 1: U-shape test error. We can see that although the training error decreases throughout the training process, the test error gradually decreases in the under-fitting regime but gradually increases in the over-fitting regime.

However, modern Machine Learning suggests that test error cannot keep U-shape all the time. When the number of parameters reach certain value, the test error will decrease. This is the new observation of the test error curve. During the whole progress of test error, it contains 2 descent curves. This phenomenon is famous as Double Descent.

In Machine Learning area, deep learning occupies a great place. Deep learning can achieve the good performance by learning a large number of parameters, while not occurring over-fitting. This seems not to be aligned with the classical Machine Learning wisdom. It is shown that Double Descent phenomenon also occurs in deep neural network.

In the project, we are going to research on the Double Descent in deep learning. In detail, we research on the shape of risk curve with the increase of model size, training epochs, dataset size. We also focus on the effect of early stopping on the double descent phenomenon. What's more, interesting ideas of figuring out the effect of injecting noise into the dataset and the process of gradient descent on double descent are generated. Besides implementing experiments, analysis are also provided.

2 Literature Review

In 2019, Mikhail Belkin et al studied the generalization behavior in machine learning model. They firstly proposed that Double Descent occurred in a wide range of data sets and stated its emergence mechanism, building a link for classical machine learning and modern machine learning in terms of bias-variance trade-off [5]. Related work in paper [9] found a phenomenon similar to double descent, which is that "jamming transition" from under-parameterized regime to over-parameterized regime in neural networks.

Whit is phenomenon appears, many scientists tried to analyze this phenomenon by theoretically like in using linear least squares regression by Peter L. Bartlett [2], In is also studied generalization performance of the machine learning model in paper [4]. Authors stated that not only deep learning models achieve low test error when over-fitting, but also the other machine learning models have this characteristic. What's more, they also suggested that kernel function is related to the generalization instead of the process of optimization.

The studies of generalization behavior promote the development of research on double descent. Paper[1] conducted research on the generalization dynamics of large gradient-descent-trained neural network. It is indicated that very large neural network will not exert bad performance on the generalization error. Double descent phenomenon is observed in this paper. Generalization is related to bias and variance. Stuart Geman et al in paper [6] indicated that the bias decreases monotonically with the width of neural network increases, while the variance firstly increases, then decreases. This showed that variance will decrease in the over-parameterized regime, which interprets the reason why double descent occurs.

Generalization studies not only facilitate the research on double descent, but to explain it by decomposing generalization into bias and variance. Brady Neal et al [9]studied the bias and variance in modern machine learning regime, which are found to decrease when the number of parameters grows.

In paper[3], Mikhail Belkin also provided mathematical proof for the U-shape curve of Double Descent phenomenon using least squares predictor, which showed that the maximum error occurs when the number of features is very close to the size of data set. And after this point, the error curve will decrease with the dimension of data increases.

Hastie et al in paper[7] studied the effect of l2 norm regularization in high dimensional neural network using one linear model and one non-linear model, which found that the double descent phenomenon will disappear when ridge regularization is applied. Preetum Nakkiran et al[8] also studied the effect of regularization on double descent. They demonstrated that l2-regularization can moderate double descent phenomenon when optimally tuning the parameters.

Zitong Yang et al [10] revealed 2 puzzles of double descent phenomenon by observing unimodal risk curve. One of the riddles solved is that injecting label noise manually make double descent phenomenon more obvious. And they also uncover the reason why double descent occurs. What's more, they also pointed out the the variance reaches highest value when the number of hidden neurons is close to the half of the number of input data parameters.

3 Double Descent Interests Arise

We firstly found double descent phenomenon in previous homework last semester, to see the Fig. 2. The experiment is implemented using random forest classifier. In the first phase, we use only 1 single estimator in the random forest and increase the the number of leaves until the number we determined before for a single tree to overfit. Then in the next phase, we keep the number of leaves for each tree at the end of the first phase and keep doubling the number of estimators. And finally, we plot the 0-1 loss figure as the function of the total number parameters on a log scale.

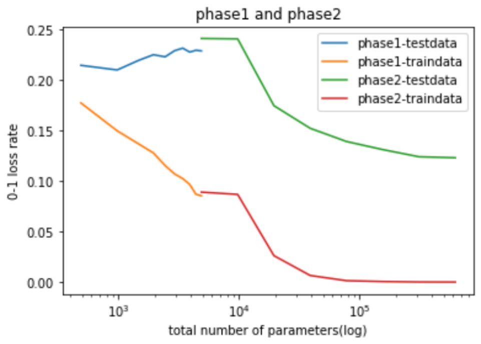


Figure 2: Double Descent in Random Forest

Although the phenomenon is not very obvious, we can still see that there is a slight decline of test error at the very first beginning. Then the test error curve increases a little bit. And when the order of the number of parameters is 4 (the number of parameters is approximately 10^4), the test error curve begins to decrease again. This order (that is 4) is the same as the number of data points in the training set that is 60000. This can be seen as model-wise double descent, which we'll talk about further in the following parts.

After observing this phenomenon in random forest classifier, some questions arise. Why double descent happens? What is the significance of double descent? And as taking deep learning course this semester, we also arise the interest in double descent phenomenon in deep learning.

We'll conduct research on questions above in theory regime.zzzAnd we also implement experiments to figure out the double descent phenomenon in deep neural network.

4 Why is double descent discovered so late?

Double descent phenomenon exists all the time, but it is observed long after machine learning is applied around the world. In general, it is the experiences of tuning parameters that lead to the late observation of this meaningful phenomenon.

In our previous perception, problems can be solved by the model with a few parameters. In this case, if we keep increasing the parameters, the test error will be shown as U-shape curve. Based on the experiences with experiments, machine learning scientists will not consider increasing the number of parameters anymore. However, the occurrence of double descent phenomenon requires the number of parameters to be adjusted beyond a certain range.

Besides, before model overfits, regularization approaches, such as integrating L1 norm penalty or L2 norm penalty, are considered. Particularly, in deep learning area, early-stopping and dropout methods are often used to prevent overfitting problems. By reason of these regularization methods, it is hard to observe double descent phenomenon.

In conclusion, because of conflicts between the habits of tuning parameters and the conditions under which double descent occurs, scientists discover double descent phenomenon late.

5 Why double descent happens?

5.1 Decomposition of generalization error

In classical machine learning area, the training error decreases with the number of features increase, while the test error looks like U-shape curve. The reason of this over-fitting problem is because of the trade-off between the bias and variance. The test error is decomposed into bias and variance in the following process and we implement it in linear model settings.
We assume data is generated independently from the same data space, so all data are independently identical distributed (i.i.d), this can be represented as $x_i \sim D$. We desire the output y to be computed with the following equation.

$$y_{clean} = w \cdot x_i = f(x_i) \tag{1}$$

However, it is inevitable to include some noise in real environment, so the true model is the combination of clean data and independent Gaussian noise ϵ_i, where $\epsilon_i \sim N(0, \sigma^2)$

$$y_i = y_{clean} + \epsilon_i \tag{2}$$

The test error is also called generalization error, which can be calculated by the expected value of least square error. This is shown in the following equation, in which $\hat{f}(x_i)$ represents the prediction result. The more detailed formula derivation of generalization error can be found in the appendix.

$$
\begin{aligned}
GeneralizationError &= E[(y_i - \hat{f}(x_i))^2] \\
&= E[(y_{clean} + \epsilon_i - \hat{f}(x_i))^2] \\
&= E[\epsilon_i^2] + E[(f(x_i) - \hat{f}(x_i))^2] \\
&= \sigma^2 + E(f(x_i) - E(\hat{f}(x_i)) - \hat{f}(x_i) + E(\hat{f}(x_i))^2 \\
&= \sigma^2 + bias^2 + variance^2
\end{aligned} \tag{3}
$$

Above equation give us idea that test error is related to sum of the bias and variance. The risk curve is decided by the trade-off between the bias and variance. Here, bias refers to the gap between the data distribution and the Machine Learning model, while variance is difference in model performance between training data and test data. This conclusion can also be applied in other model architectures, like neural network.

5.2 Bias-Variance curve discussion

Classical machine learning wisdom points out the test error curve is shown as U-shape, while modern learning holds the idea that test error curve should look like ”double descent”. We mention above that the generalization error curve is the summation of the square of variance of the Gaussian noise, bias of the predictor and the variance of the predictor. Accordingly, different shapes of risk curve can be explained by the curves of the bias and variance. Bias curve always monotonically decreases with the the complexity of model increases.

Variance has a more complicated picture.In classical machine learning, experts suggest that the variance increases with the dimension of input data expands. In this case, at the very first beginning,the bias decreases more than the variance increases, which leads to the overall decreases of risk curve. Then, the variance increases faster than the bias decreases, this results in the increases of risk curve. The variation of bias and variance described above is the reason why U-shape risk curve occurs. The process described above can be shown as Fig.3

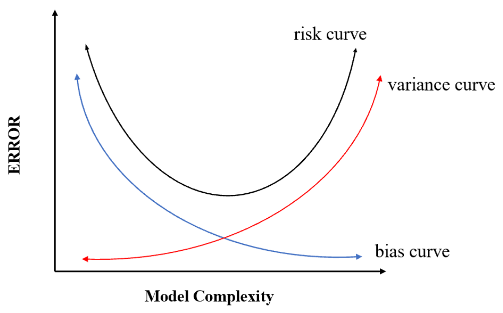


Figure 3: Risk Curve in Classical Machine Learning

However, modern machine learning scientists have the different opinions in the shape of variance curve. They stated that the experiments show the variance doesn't increase monotonically. Instead, the variance curve will firstly increase, then decrease with the increase of model complexity. In the first phase, bias decreases faster than variance increases, which leads to the drop of generalization error. In the next phase, bias decreases slower than variance increases. This is the reason why generalization error curve increases in the current setting. Finally, in the third phase, both bias and variance decrease, which triggers the decline of the the test error. The whole process is described in Fig.4.

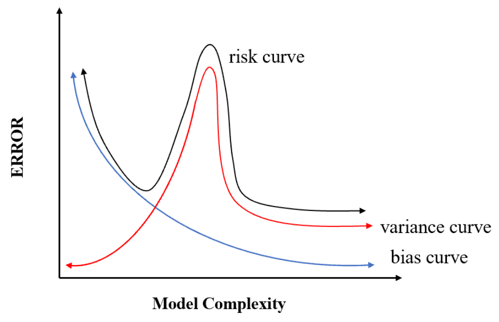


Figure 4: Risk Curve in Modern Machine Learning

In conclusion, the trade-off between the change of bias and variance make the double descent phenomenon.

5.3 When double descent happens?

Classically, double descent phenomenon occurs with increased model complexity. Specifically, it means the test error curve changes as the number of data features increases.Assuming we can observe double descent in the risk curve, we'll talk about when the second decline of risk curve appears.

As we mentioned before, we observe that the the second descent of the test error curve utilizing random forest classifier occurs, when the order of magnitude of the dimension of data is approximately equal to the counterpart of the number of dataset. This conclusion is also applied to deep learning area. However, there is a little difference in multi-classification problems. If the prediction result has more than one output, this is seen as multi-classification problem. In this setting, the starting point of second drop should be the multiplication of the number of features and the number of categories instead of just the number of features of the input data.

In deep learning area, parameters like "width","depth", etc are considered. These parameters have the influence on the number of neurons which is the feature in deep neural network. In deep learning area, the point where the second decline of risk curve appears is also aligned with what we talked about before. The dimension of data here is the number of all features in the deep neural network.

6 Significance of Double Descent

Before observing double descent, generalization error is shown as U-shape curve. This means that the test error will firstly decrease, then increase. In this case, if model complexity is beyond a certain range, then we'll get a low accuracy. The occurrence of double descent brings hope for low generalization error when in high model complexity.

What's more, as we know, deep neural network can always achieve good performance when depth and width are large. This phenomenon without over-fitting seems not aligned with common sense. Double descent appearance can explain the satisfying performance in a certain degree properly.

7 Experiment

7.1 Terminology

In this part, we're going to introduce the terminology we used in the experiments. The detail can be seen in the following context.

7.1.1 Loss Function

In the experiments, we use the classic dataset to resolve multi-classification problem in which the cross-entropy loss is always used. Cross-entropy is utilized to measure the distance between 2 distributions. The traditional error is calculated using Mean Square Error (MSE). In deep neural network, sigmoid function is often taken as activation function. But the derivation of sigmoid function may lead to the disappearance of gradient. Therefore, the meaning of cross-entropy error's existence is to make up for the deficiencies of MSE using sigmoid function. In deep learning regime, cross-entropy and softmax function are closely connected. In cross-entropy calculation, softmax function is used to achieve the probability of the data being predicted as each category.

In binary classification problem, the formula of cross-entropy is as follows.

$$CrossEntropy = -[label * log(P) + (1 - label) * log(1 - P)] \tag{4}$$

where *label* represents the label of the data, in which the label of positive samples are 1 and the label of negative samples are 0. And P indicates the probability of a data being predicted as positive sample.

In multi-classification problem, the cross-entropy is computed as the following equation.

$$CrossEntropy = -\sum_{i=1}^{n} label_i * log(P_i) \tag{5}$$

where n denotes the number of labels of the dataset. $label_i$ means that if the category of the data is i, then $label_i = 1$, otherwise $label_i = 0$. P_i represents the probability of the data being predicted as label i.

7.2 Experimental setup

7.2.1 Models

After weighing the model's innovation, classic and training difficulty, we choose to use Pytorch implementing the following families of architectures.

CNNs. In Fashion MNIST dataset, we use a 2-layer CNN structure, with filter numbers of w and $2w$ in each Convolutional layer, where w denotes the width of our model. The kernel size and stride of the filter is chosen as 3 and 1, in additon to padding to make the dimension remain the same. After each Convoluational layer, we use BatchNormalization, ReLu and Max Pooling of 2x2 to downsample the size by 2 in each dimension. After those 2 Convolutional layers, we use a Linear layer to transform the output into size 10. Respectively, we choose the structure of CNN models with 3-layer, 5-layer and 7-layer in Cifar10 experiments, w. For each Convolutional layer, we apply filters with kernel size = 3, stride = 1 and padding = 1. After each Convolutional layer, layers consist of BatchNormzliation, ReLU and Max pooling. We use Max Pooling after the first layer. The model delays Max Pooling until the last 4 layers. After 4 layers of 2x2 (stride 2x2) max pooling the image dimension goes from 32 x 32 to 4x4. The Convolutional layer widths are designed as $[w, 2w, 4w, 8w, \cdots]$. w is introduced as a changeable parameter to control the width and complexity of model when training and is the so-call 'width' in all of our experiments. The pooling size and stride of MaxPool will be 2. Finally, a Linear layer is added after MaxPool with size 4 and ReLU.

7.2.2 Training Detail and Hyper-parameter

Data Augmentation: In experiments where data-augmentation was used, apart from the transitional common transformations like `RandomCrop` and `RandomHorizontalFlip`. We also introduce the noise not only to the label but also to the feature data.

To be more specific, for label noise, we randomly select certain proportion of data (represent $x\%$ of label noise). Change the correct label to the wrong result. For example, in Cifar10, all target labels are set between the range of 0-9. Once a single data label as 5, we can modify it to any other random number in 0-9. So that to simulate the scenarios of label noise.

For feature noise, in image classification situation, we introduce a Gaussian noise to every 0-255 pixel value of original images.

$$GaussianNoise = Amplitude \cdot N(mean, variance) \tag{6}$$

And then normalize the data feature to range 0-1.

In experiments with added label noise and feature noise, the label for all augmentations of a given training sample are given the same label. And the Gaussian noise add to any pictures between different epochs keep the consistent.

Loss function: Apart from certain specific scenarios, all the experiments use the cross-entropy loss.

Optimizer:

1. **SGD:** For Cifar10 dataset, the learning rate schedule inverse-square root was used with initial learning rate $lr = 0.1$. And we updated the learning rate every 512 gradient steps. No momentum was used here. For Fashion MNIST dataset, we introduced learning rate as 0.01 in the beginning. Then as the epoch increases, the learning rate decreases as $0.01\sqrt{\frac{1}{epoch}}$. For the momentum, we set it as 0.95 here.

2. **Stochastic Gradient Langevin Dynamics(SGLD):** SGLD is an optimization approach whose basic idea is to inject noise in the process of gradient descent. It is similar to Stochastic Gradient Descent (SGD), except that it combines noise when updating parameters in the iteration of gradient descent.

$$x_{t+1} = x_t - LearningRate \cdot \nabla f(x) + \sqrt{1/LearningRate} \cdot \sqrt{2 \cdot T} \cdot w \tag{7}$$

   where $LearningRate$ represents the learning rate of gradient descent. f indicates the objective function and T is called temperature which usually takes the value 0.1 or 0.01. w denotes the Gaussian noise.

Initialization and Regularization: Variances are initialized as the default setting of Pytorch. There is no regularization like wright weight decay and dropout except certain explicit situation.

Batch Size: Experiments on Cifar10 used a batchsize of 128. Experiments on Fashion MNIST used a batchsize of 100.

Learning Rate: In the experiments, the learning rate for the optimizer used is very important as it significantly determines whether we can observe the double descent phenomenon. The specific setting is mentioned in optimizer part.

Train Error and Test Error: The former equals to $1 -$ *train accuracy* (Test the performance of model based on training set after every epoch). The latter equals to $1-$*test accuracy*.

7.3 Experiment Result

7.3.1 Model-wise Double Descent

We reproduced the width-wise double descent phenomenon in our own experiments. Unlike those experiments mentioned in the paper that is conducted on Cifar10, Cifar100 and MNIST data set, we tried to reproduce the width-wise double descent plot on Fashion MNIST data set. The result of this reproduction is shown in Fig. 5.

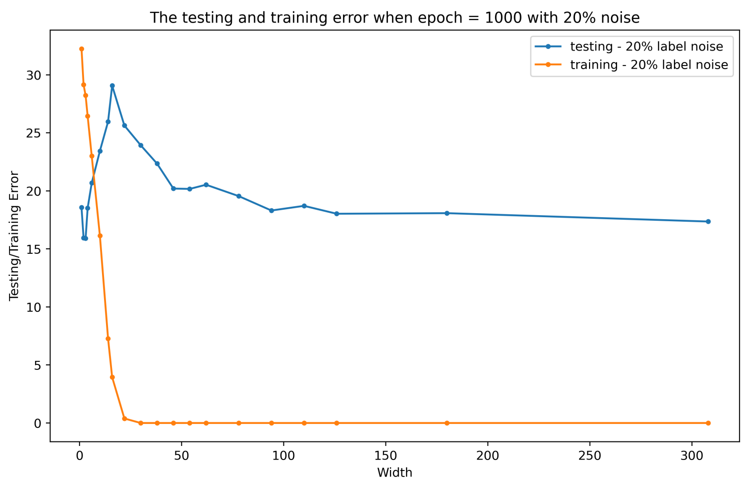
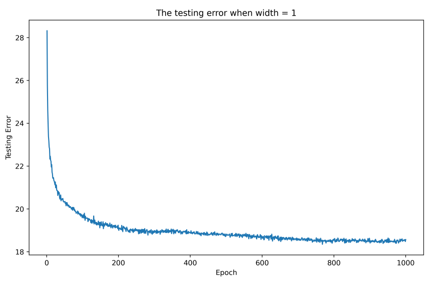


Figure 5: Width-wise Double Descent on Fashion MNIST data set with 20 percent label noise

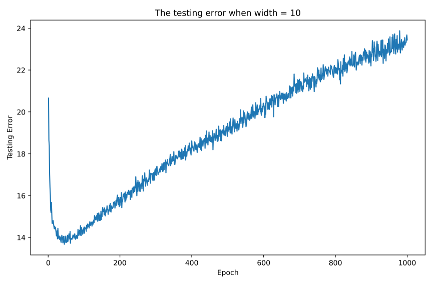
7.3.2 Epoch-wise Double Descent

In our experiments on Fashion MNIST with various choices of w, we train those different sizes of model for 1000 epochs each. According to the results, we categorized those model sizes into four different regions, which are characterized by their different testing error performances over 1000 epochs training.

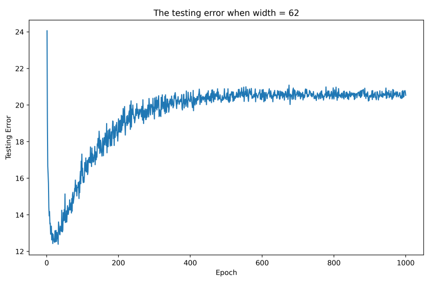
The specific plots of model size in each region is shown in Fig. 6. First, when the model width is very small, there will be no overfitting happens. Then when the model width becomes larger, the overfitting issue comes and becomes worse and worse in a certain region. After that region, the overfitting starts to become weaker and at the end of training, the testing error tends to become stable. After we enter the last region, where the model size is really huge, the extent of overfitting is much smaller than before and after the peak of overfitting, the testing error even tends to slowly drop down.
Thus the phenomenon of double descent occurs as the number of epochs increases is observable when the model used to train is large enough. In our experiment, we reproduce this



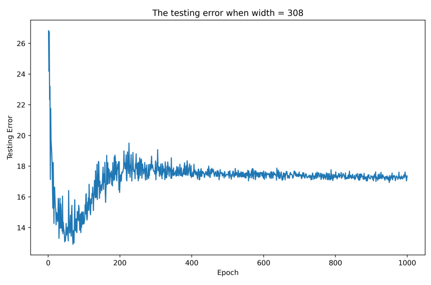
(a) Small-model region



(b) Overfitting Worsening region



(c) Overfitting Alleviating region



(d) Double Descent region

Figure 6: The testing error of model size in 4 different regions

phenomenon on Fashion MNIST date set using a 2-layer CNN with $w = 512$. The more specific setting of CNN is shown in the experiment part. The result is shown in Fig. 7(a).

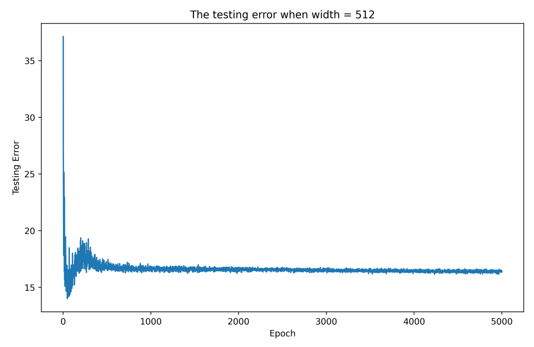
The number of epochs also influence how obvious the model-wise double descent phenomenon is, which is shown in Fig. 7(b). Also, another observation made here is that the peak of double descent plots shift to left gradually when we increase the number of epochs to training the data.

7.3.3 Noise-wise Double Descent

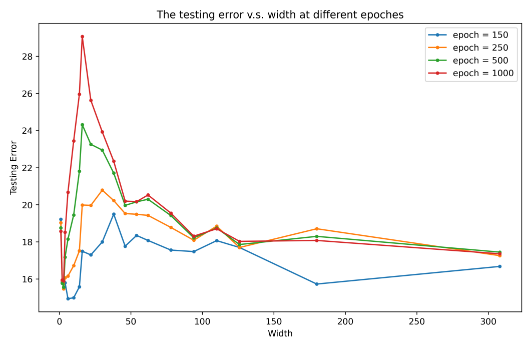
There are several types of noise we can add to the data to boost the double descent in the observation. One type that the paper has added to the data is the label noise, which tries to randomly label a certain part of the image to a wrong label. After we removed the label noise, the double descent phenomenon is not obvious anymore. Again, we tried this on Fashion MNIST data set, which is shown in Fig. 8(a).

7.3.4 Learning-rate-wise Double Descent

In our experiments before, we are adapting a decaying learning rate for our SGD optimizer. We are also interested how important this setting is in our double descent phenomenon. Therefore, we also try a fixed learning rate value of 0.01 throughout all the training epochs. Then



(a) Epoch-wise Double Descent with CNN width 512



(b) Width-wise Double Descent with different epochs

Figure 7: Epoch-wise Double Descent on Fashion MNIST data set with width 512 (Left) and Width-wise Double Descent on Fashion MNIST data set after different training epochs (Right)

result of changing is reflected in Fig. 8(b), which shows that if we use fixed learning rate, the double descent will be more obvious in terms of peak. However, for the second descent part, if we use fixed learning rate, the testing error can't drop as well as decaying learning rate.

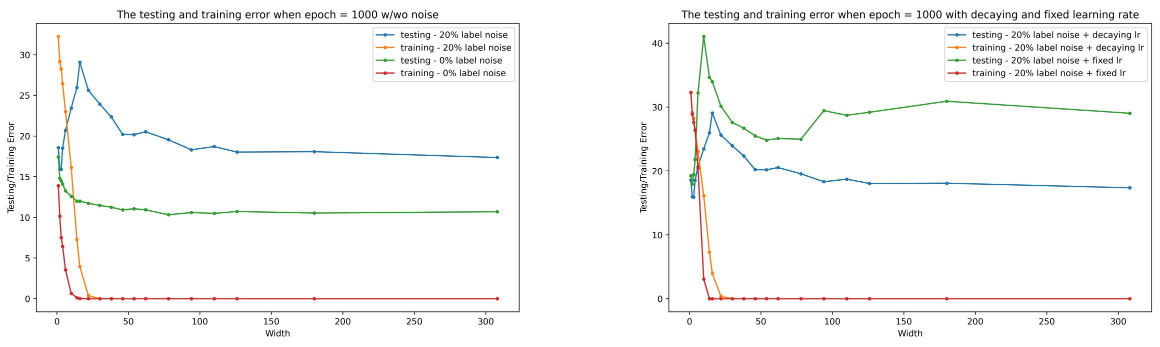
7.4 Discussion and Analysis

In the majority of our experiments mentioned above, we observed three important double descent phenomenon. Although they happen in wide-wise, epoch-wise and noise-wise, but we here come up with one idea to conclude them all.

The idea is that we can use the different behaviour of testing error curves for model sizes in four different regions mentioned above to account for all double descent phenomenon and their changes happen width-wise and epoch-wise.

A quick review of those four different regions are: 1. Small-model region, where the model is of extremely small size and no overfitting happens; 2. Overfitting Worsening region, where the model is larger than models in the first region but the overfitting is becoming more and more serious as the model size increases; 3. Overfitting Alleviating region, where model size becomes even larger, but the overfitting stops after training for enough epochs; 4. Double Descent region, where the size of mode is extremely large that the overfitting magnitude is much smaller and the testing error even start to drop after reach the peak of overfitting.

With those four regions in mind, we can explain the wide-wise double descent. First, when the model size is in small-model region, there is no overfitting issue, so that the model tends to have better testing data prediction performance as the model size increases. After the model size enters the region of overfitting worsening, the overfitting issue tends to become more obvious as the model size increases, so that the testing error plots will increase in this region. Then when the size of model rises to overfitting alleviating region, the extent of overfitting



(a) Width-wise Double Descent w/wo label noise (b) Width-wise Double Descent with decaying or fixed learning rate

Figure 8: Comparison on Double Descent on Fashion MNIST data set w/wo label noise (Left) and Comparison on Double Descent on Fashion MNIST data set with decaying and fixed learning rate (Right)

becomes weaker as the model size increases and also stop earlier, so that the testing error curve will drop again. Finally, when the size enters the double descent region, the extent of overfitting is extremely small and the testing error even slowly drops when trained for enough epoch. So that the curve tends to slowly drop in this area, which looks very smooth.
Also, we can use those four regions to account for the observation in Fig. 7(b). For models in small-model region, there is no ovefitting issue, so that training for longer epochs will not significantly influence their performance in predicting tetsing data. Similarly, for models in double-descent region, the overfitting diminishes very quickly, so that the training epoch has no significant influence on the testing errors if it is large enough (for example 250 epochs in our plot). For models of size in between , the training epoch really matters to those models. For models in overfitting worsening region, the more epochs we train, the more serious overfitting is, so that the peak in width-wise double descent curve tends to become higher when we train more epochs. For models in overfitting alleviating region, the stop point of overfitting becomes earlier as the model size increases, so that decreasing training epochs has little influence for models. However, for models that are not large enough, the overfitting doesn't stop after a specific number of epochs (250 epochs at model size 50 for example), so that if we train longer, the testing error still increase. Because of the above intrinsic reason, the double descent phenomenon looks more obvious when we train for more epochs. If the training epochs are very small, the performance of the models is still heavily fluctuating, so that the double descent curve looks not as nice as those trained for more epochs.
We also come up with an explanation for the noise-wise double descent phenomenon. The reason we think why the double descent diminish after we remove the label noise is because the most important issue in the double descent phenomenon is that the model possibly overfits

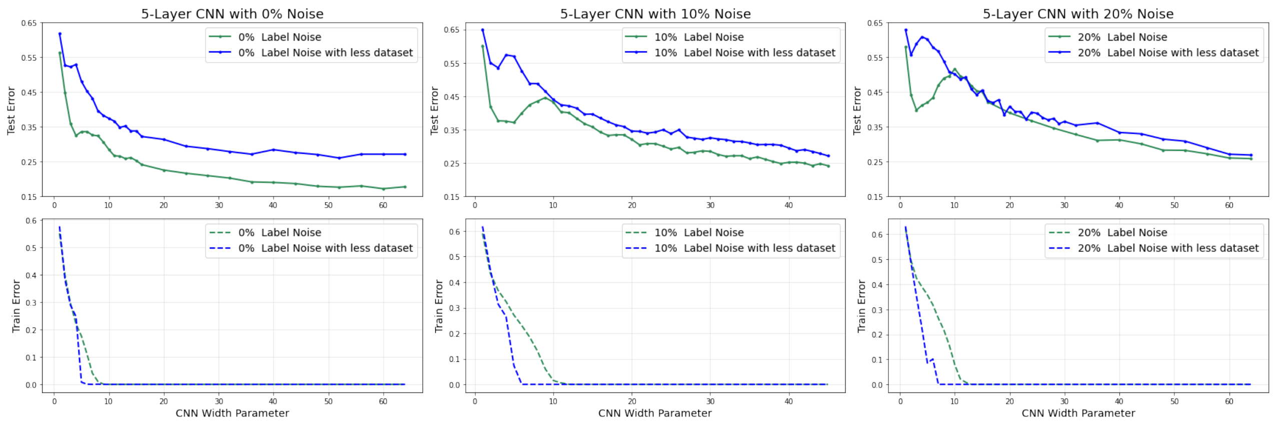


Figure 9: Model-wise double descent for 5-layer CNNs with respectively 0, 10%, 20% label noise on CIFAR-10, for varing dataset sizes.

as the model size increases. However, for the Fashion MNIST dataset we choose here, there is no significant bias between testing and training data set. As the dimension of each picture is 28×28, you can't expect a very huge bias between two pictures of bags. So that it is not easy for overfitting to happen even if we use more parameters than necessary. Therefore, adding 20 percent label noise is a good way to introduce sufficient bias between data so that overfitting is more possible to happen in this case. As the overfitting issue happens, the four regions mentioned above can be used for explaining the double descent phenomenon well.
For the reason why double descent curves are slightly different under different schedules on learning rate, we think it is because fixed learning rate could lead to more serious overfitting issue. We gain this information from the higher peak of double descent curve and more fluctuate behaviour on testing error in the region where decaying learning rate tends to become flat and smooth curves.

8 Other Interesting Explorations

Here we demonstrate several potential elements fitting into situations that make sense in real-world application scenarios, which may influence the curve of double descent.

8.1 The Number of Dataset

In some particular situation, if we just want to validate or explore certain phenomenon or model. The primary idea coming out will be setting up a demo with a simple model and small-scale dataset. So that save time and resource at the early stage of projects. On the other hand, if there is not too much input sample for model training even though data-augmentation was

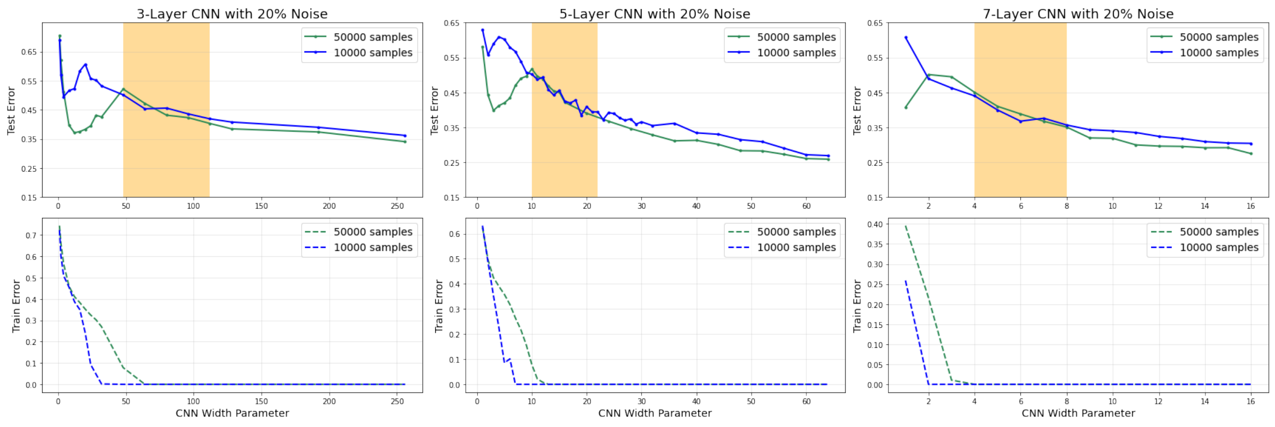


Figure 10: Model-wise double descent for several CNNs with respectively 3,5 and 7 layers on CIFAR-10, for varing dataset sizes.

implemented, we have to try strategies to accomplish the task with limited data. Therefore, whether there exists double descent in such scenarios and how it changes will be an interesting and pragmatic thing for exploration.

Similar to the experiments above, by means of reducing the number of Cifar10 dataset (training set from 50,000 to 10,000, test set from 10,000 to 2,000), we plot the curve of error performance under different number of sample.

We implemented the experiments in two aspects, as shown in Fig. 9 and Fig. 10. Compare whether less dataset would influence the shape of curve in the scenarios of different model complexity and different label noise. They came to similar conclusions.

On one hand, it is in line with the expectation that the peak of over-fitting ”moves to the left” as the sample of the dataset decreases. Even, in the situation of 7-layer CNNs, the over-fitting peaks are so far in advance that the front part of the U-shaped curve completely disappears. The curve as a whole is moving up and the area under the curve increases. The training Error decreases more quickly, the threshold where the training error drops to near 0 is basically corresponding to the threshold where the test error over-fitting reaches the peak.

On the other hand, there are not too many differences for different number of sample when the width of network is large enough (the model is complex enough). Especially when the curve just passes the fitting point, as shown in the orange shaded region of Fig. 10, there is a range of model widths where nearly no improvement for the test error between large-scale dataset and small-scale dataset.

8.2 Early Stopping

In machine learning, early stop is a form of regularization used to avoid over-fitting when training a learner using iterative methods, such as gradient descent. This approach updates

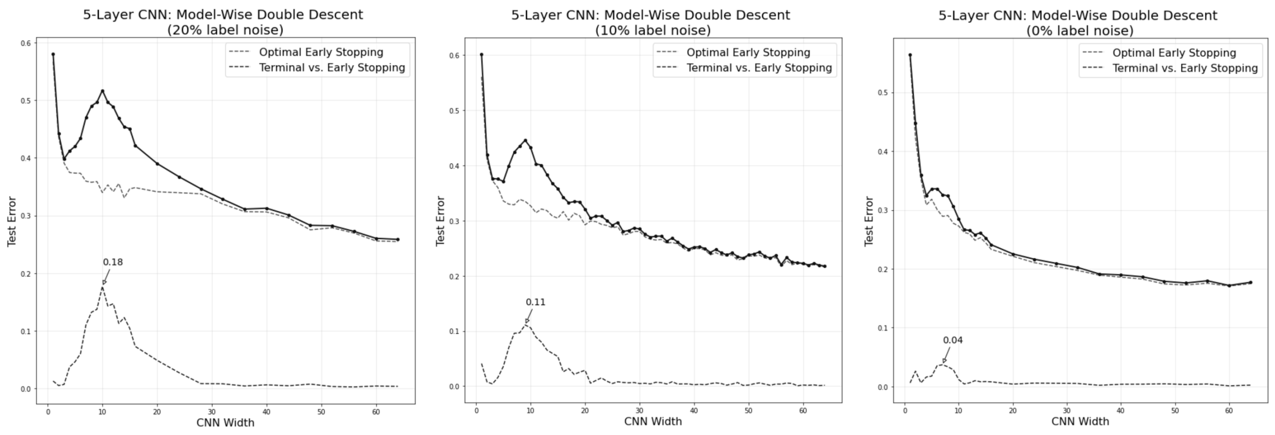


Figure 11: Visualization of curve after implementing early stopping. **Top:** The double descent curve and optimal early stopping curve. **Bottom:** The difference of the two.

the learner to better fit the training data at each iteration. To some extent, this improves learners' performance on data outside the training set.

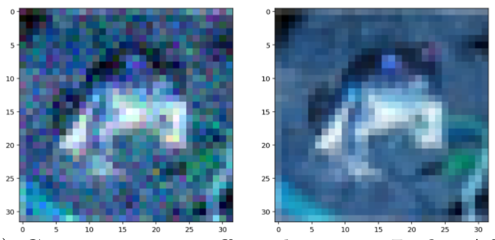
In many machine learning libraries, it has been a callable callback method to save training and improve model performance by monitoring certain metrics like loss or accuracy of the validation set. While in our experiments, to what extent can a perfect early stop improves model performance. Since it is a fantastic strategy to avoid over-fitting, double descent curve will not appear when an early stop is implemented. Unlike in real practice, the model needs to wait for several "patient" epochs to decide whether stop or not, we directly outputted the minimum error value during the every width training process, because we save the accuracy score of every epoch. The curve is shown as Fig. 11. The dotted line at the top of the table represents the result of adopting the early stop policy. And the bottom one is the subtraction value of the double descent curve and optimal early stop curve.

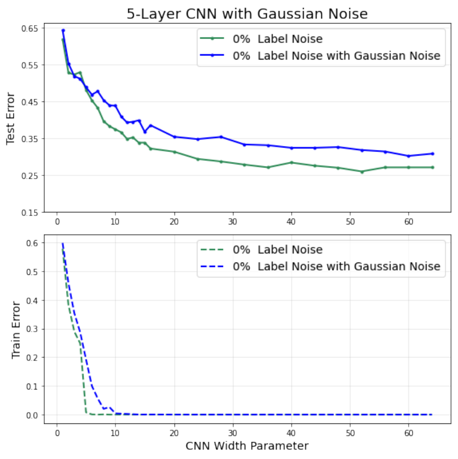
We compare several models with different complexity and different label noise, which dramatically affects the location and relative height of the peak. The truth is early stopping can significantly smooth the over-fitting plot so as to return the optimal result. It corresponds with the commonsense that early stopping can avoid over-fitting.

8.3 Feature Noise

As we have mentioned in section 7.2.2, we add a Gaussian noise as a mask to observe the change of the double descent curve. Take the image of CIFAR-10 as an example, every sample is a 32x32x3 matrix (width x height x channel) with pixel values between 0-255. In Fig. 12(a), we implement a $N(0, 1)$ noise with amplitude 20. Changes in every channel were different. After changing the format into PIL(uint8), we normalized the sample into matrix with values between range 0-1. The right picture is the original picture while the left one is that after transform.

As shown in Fig. 12(b), we implemented the same 0% noise in the experiment. Although in the absence of label noise, the curve of double decline is not obvious. We can still obverse that the operation of adding Gaussian noise smoothed out the double descent peak to some extent. While the overall error goes up a little bit. Since we can consider adding feature noise as one kind of data augmentation. The reason why the curve changes may be because the data augmentation enhances the generalization ability of the model. In other words, it avoids the over-fitting of the model. Such results also correspond with commonsense.


(a) Gaussian noise effect drawing. **Left:** After the transform. **Right:** Before the transform.



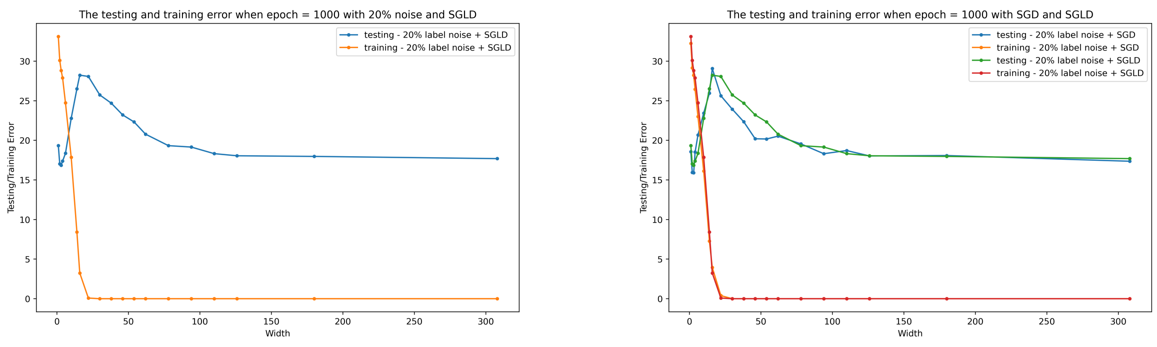
(b) Model-wise plot of different feature noise with 0% label noise.

Figure 12: Effect schematic of adding Gaussian noise(**Left**) and comparison of double descent curve(**Right**).

8.4 SGLD

We also try to change the SGD optimizer we used before to another optimizer SGLD to see wether it still results in double descent phenomenon after we add some uncertainty into the gradient descent step. The plot is shown in Fig. 13(a).

We can still observe double descent phenomenon under SGLD optimizer, but it looks a little bit different from double descent plot we gained from SGD optimizer. To compare them more clearly, we put them in the same plot in Fig. 13(b). In this figure we can observe that , upon changing the optimizer to SGLD, the second descent in the plot tends to be shifted to the right a little bit. However, in the end, the testing error for huge enough models tend to become the same. So in general, we can make conlcusion here that, SGD is better to be used on the Fashion MNIST data set, as the second descent in the testing error comes earlier than SGLD.



(a) Width-wise Double Descent with SGLD (b) Comparison between SGD and SGLD

Figure 13: Width-wise double descent on Fashion MNIST data set with 20 percent label noise and SGLD optimizer (Left) and Comparison with SGD optimizer (Right)

We try to analyze the difference between the curve using SGD and SGLD. From the figure, we can see that in the over-fitting regime shown as the blue rising curve, the test error with SGLD is kind of lower than that with SGD, although it is not obvious. We assume that this is because of the extra randomness imported by SGLD. Model doesn't fit very well with the test data at this time, extra noise imported by SGLD can help disperse the global minimum value, thus change the landscape of objective function. This behavior can increase the chance of finding the global minimum and thus decrease the test error a little bit. When meeting up with the second decline, more complex model becomes fit the data. Randomness now is not necessary and may hurt the performance of the model, since the imported noise may cover the global minimum at this time. The small difference here between the SGD and SGLD is possibly because we introduce very little noise in SGLD when updating the parameters.

9 Conclusion

In this project, we discussed the double-descent phenomenon in both theoretical and practical aspects. First, we introduce the concept of the double-descent phenomenon: the test error is shown as a U-shape curve in classical machine learning, then declines secondly as the model becomes more complex. Then we analyzed why the double-descent phenomenon happens, in this part, we focused on the bias-variance balance, since the generalization error can be decomposed into bias and variance. The double-descent phenomenon happens because the variance does not increase monotonically, it will first increase and then decrease.

The focus of our project is to study which parameters affect the double-descent phenomenon. We first considered the traditional parameters, such as the width of the neural network, epoch, label noise, and learning rate. After that, by analyzing the results, we came up with some interesting conclusions, our model complexity is based on the width of the neural network, and we know that the double-descent phenomenon will appear as the model complexity increase, based on this condition, we will test the different effects of different parameters on this phenomenon.

Epoch-wise: The double-descent phenomenon gradually becomes apparent with the epoch increasing, the peak of double descent plots will also shift to the left. The reason for this phenomenon is the larger the epoch, the more serious the over-fitting. The larger model was enlarged to a certain extent and stopped over-fitting when the epoch increased. So the peak will shift to the left.

Label Noise-wise: The double-descent phenomenon can be boosted by the noise, but when we remove the noise, the double-descent phenomenon is not obvious more, for the worst case, the double-descent phenomenon even not appears. The reason why noise is so important we think is that if the data we choose does not have a significant bias between testing and training data, it is hard to cause the over-fitting when the model size increase, but when we add some noise, like in the experiment we add 20% label noise, is a good way to introduce some bias to the dataset.

learning rate-wise: In this section, we tried to fix the learning rate to see whether there is any influence on the double-descent phenomenon. The result is obvious, when we fixed the learning rate to 0.01 throughout all the epochs, we can see the peak of the double-descent much higher than before, but for the second descent, it can not drop as well as before.

Dataset scale: In this section, we tested the effect of different dataset scales on this double-descent phenomenon. The experiment is based on CNN with different layers, 3-layers, 5-layers, and 7-layers respectively, the result tends to be similar in different settings. We observed 2 main phenomena: first, the peak shifted to the left when we reduce the dataset scale, at the same time, we can see the training error decreases more quickly. Second, both the test error curve and train error cure move up and the area under the curve increases. The reason is that more data can increase the model complexity and also can hurt the model performance.

Early Stopping: In this section, we saved the accuracy score of every epoch and we directly output the minimum error value during the process. We found that with the Early stopping method, the double-descent curve tends to smooth, cause the early stopping is a kind of optimizer, to avoid the over-fitting, but without over-fitting, the double-descent phenomenon will not appear.

We also conduct interesting experiments and try to figure out the effect of Gaussian noise and SGLD on the double descent phenomenon. The core idea is to inject noise into the raw data and the process of gradient descent.What we found is that injecting Gaussian noise into raw data can smooth double descent a little bit.We assume this is because imported randomness in the input data enhance the generalization ability of the model, hence make the over-fitting disappear. For utilizing optimizer SGLD, we found that it achieves lower test error than SGD when in over-fitting regime and have higher test error when in over-parameterized area. We analyzed that the reason may be that noise can help reach global minima by changing the landscape of objective function when model doesn't fit data very well. But when model becomes fit the data, this randomness becomes unnecessary and may cover the global minimum, hence exert a bad influence on the accuracy.

Summary Table and Experimental Results

Dataset	Arch.	Opt.	%Noise	Double-Descent		Experiment in	Figure(s)
				in Model	in Epoch		
F. MNIST	2L-CNN		20%	√		Width	5, 6, 7(a)
F. MNIST	2L-CNN		20%		√	Epoch	7(a), 7(b)
F. MNIST	2L-CNN		0			Noise	8(a)
F. MNIST	5L-CNN		0	√		Noise Early Stopping	11
F. MNIST	5L-CNN		10%	√		Noise Early Stopping	11
F. MNIST	5L-CNN		20%	√		Noise Early Stopping	11
CIFAR-10	3L-CNN		20%	√		Sample Scale	10
CIFAR-10	5L-CNN		20%	√		Sample Scale	10
CIFAR-10	7L-CNN		20%	√		Sample Scale	10
CIFAR-10	5L-CNN		Gaussian			Feature Noise	12
F.MNIST	2L-CNN	Fixed SGD LR	20%	√		Learning rate	8(b)
F.MNIST	2L-CNN	Decay SGD LR	20%	√		Learning rate	8(b)
F.MNIST	2L-CNN	SGLD	20%	√		Optimizer	13(a), 13(b)
F.MNIST	2L-CNN	SGD	20%	√		Optimizer	13(a), 13(b)

References

[1] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. “High-dimensional dynamics of generalization error in neural networks”. In: *Neural Networks* 132 (2020), pp. 428–446.

[2] Peter L Bartlett et al. “Benign overfitting in linear regression”. In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30063–30070.

[3] Mikhail Belkin, Daniel Hsu, and Ji Xu. “Two models of double descent for weak features”. In: *SIAM Journal on Mathematics of Data Science* 2.4 (2020), pp. 1167–1180.

[4] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. “To understand deep learning we need to understand kernel learning”. In: *International Conference on Machine Learning.* PMLR. 2018, pp. 541–549.

[5] Mikhail Belkin et al. “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854.

[6] Stuart Geman, Elie Bienenstock, and René Doursat. “Neural networks and the bias/variance dilemma”. In: *Neural computation* 4.1 (1992), pp. 1–58.

[7] Trevor Hastie et al. “Surprises in high-dimensional ridgeless least squares interpolation”. In: *arXiv preprint arXiv:1903.08560* (2019).

[8] Preetum Nakkiran et al. “Optimal regularization can mitigate double descent”. In: *arXiv preprint arXiv:2003.01897* (2020).

[9] Stefano Spigler et al. “A jamming transition from under-to over-parametrization affects generalization in deep learning”. In: *Journal of Physics A: Mathematical and Theoretical* 52.47 (2019), p. 474001.

[10] Zitong Yang et al. “Rethinking bias-variance trade-off for generalization of neural networks”. In: *International Conference on Machine Learning.* PMLR. 2020, pp. 10767–10777.

A Appendix

A.1 Proof for Generalization Error

Considering noise ϵ_i follows Gaussian distribution, that is,$\epsilon_i \sim N(0, \sigma^2)$, so $E[\epsilon_i] = 0$, $var[\epsilon_i] = \sigma^2$, therefore, we can get the value of $E[\epsilon^2]$.

$$E[\epsilon^2] = E[\epsilon]^2 + var[\epsilon] = 0 + \sigma^2 = \sigma^2 \tag{8}$$

Because noise ϵ_i is independent of data, therefore, $E[\epsilon_i * (y_{clean} - \hat{f}(x_i))]$ can be decomposed as

$$E[\epsilon_i] * E[y_{clean} - \hat{f}(x_i)] = 0$$

Then we deduce the formula of $E[(y_{clean} - \hat{f}(x_i))^2]$ for further formula derivation.

$$\begin{aligned} E[(y_{clean} - \hat{f}(x_i))^2] &= E[((y_{clean} - E[\hat{f}(x_i)]) - (\hat{f}(x_i) - E[\hat{f}(x_i)]))^2] \\ &= E[(y_{clean} - E[\hat{f}(x_i)])^2] + E[(\hat{f}(x_i) - E[\hat{f}(x_i)])^2] \\ &- 2 * E[(y_{clean} - E[\hat{f}(x_i)]) * (\hat{f}(x_i) - E[\hat{f}(x_i)])] \\ &= bias[\hat{f}(x_i)]^2 + var[\hat{f}(x_i)]^2 \\ &- (y_{clean} - E[\hat{f}(x_i)]) * E[\hat{f}(x_i) - E[\hat{f}(x_i)]] \\ &= bias[\hat{f}(x_i)]^2 + var[\hat{f}(x_i)]^2 \\ &- (y_{clean} - E[\hat{f}(x_i)]) * (E[\hat{f}(x_i)] - E[\hat{f}(x_i)]) \\ &= bias[\hat{f}(x_i)]^2 + var[\hat{f}(x_i)]^2 \end{aligned} \tag{9}$$

Combining the equations derived above, the formula of generalization error is achieved.

$$\begin{aligned} &GeneralizationError = E[(y_i - \hat{f}(x_i))^2] \\ &= E[(y_{clean} + \epsilon_i - \hat{f}(x_i))^2] \\ &= E[(\epsilon_i + (y_{clean} - \hat{f}(x_i)))^2] \\ &= E[\epsilon_i^2] + E[(y_{clean} - \hat{f}(x_i))^2] + 2 * E[\epsilon_i * (y_{clean} - \hat{f}(x_i))] \\ &= \sigma^2 + +E[(y_{clean} - \hat{f}(x_i))^2] \\ &= \sigma^2 + bias[\hat{f}(x_i)]^2 + var[\hat{f}(x_i)]^2 \end{aligned} \tag{10}$$