Analyzing the Double Descent Phenomenon for Fully Connected Neural Networks

Johannes Scherer
johannes.scherer@uni-ulm.de
Master Computer Science
Projekt Neuroinformatik | University of Ulm

Abstract

The double descent risk curve was proposed by Belkin et al. [4] to address the seemingly contradictory wisdoms that *larger models are worse* in classical statistics and *larger models are better* in modern machine learning practice. As the model size is increased, double descent refers to the phenomenon in which the model performance initially improves, then gets worse, and ultimately improves again. In this work, we aim to analyze the occurrence of double descent for two-layer fully connected neural networks (FCNN). Specifically, we reproduce and extend the results obtained by Belkin et al. for FCNNs. We show that, in contrast to their proposed weight reuse scheme, the addition of label noise is an effective method to make the presence of double descent apparent. Furthermore, we effectively alleviate the occurrence of double descent by employing regularization techniques, namely dropout and l_2 regularization. The implementation code is publicly available at `https://github.com/josch14/double-descent`.

1 Introduction

Bias and variance are the essential error types in machine learning (ML) [17]. **Bias** (approximation error) represents a model's simplifying assumptions when approximating a complex real-world problem. High bias error indicates *underfitting*, meaning that the model complexity is not sufficiently large enough to learn relevant relations between input features and label. On the other hand, **variance** (estimation error) represents a model's sensitivity to small fluctuations in training data. High variance error indicates *overfitting*, i.e., the model complexity is too large, enabling it to learn random noise in training data which weakens the model's ability to generalize. Bias and variance are two deeply connected concepts. For example, the bias–variance decomposition states that the prediction error of any ML classifier can be decomposed into squared bias error, variance error, and error which can not be reduced regardless of the algorithm used (irreducible error) [6].

The **bias-variance trade-off** is a fundamental ML theory describing the dependency between bias and variance (see Fig. 1 (a.1) and (a.2)) [13]. It suggests that a model of low complexity has high bias and low variance, resulting in both, high train and test error. By increasing the model complexity, e.g., by adding neurons to a neural network, the model is enabled to learn more complex patterns. This leads to a significant reduction in bias, and consequently a reduction in train and test error. As the model's complexity continues to increase and variance dramatically increases, the model achieves low bias and high variance. As a result, the test error increases again while the train error continues to decrease. Overall, the bias-variance tradeoff suggests a U-shaped test error curve. Furthermore, it implies a certain threshold located at the model complexity for which the lowest test error is achieved. Increasing the model complexity past the threshold leads to increased test error due to variance dominating bias, hence explaining the wisdom in classical statistics that *"larger models are worse"* [13], and that a model needs to be balanced between underfitting and overfitting in order to achieve optimal generalization ability [4].

In modern practice, however, contrary to the intuition of the classical bias-variance tradeoff, very rich models such as deep neural networks are often trained to interpolate training data (i.e., extreme overfitting to achieve zero training error) and yet are able to obtain minimal test error [13]. Hence, conventional wisdom in modern machine learning is that *"larger models are better"* [5, 9]. Aiming

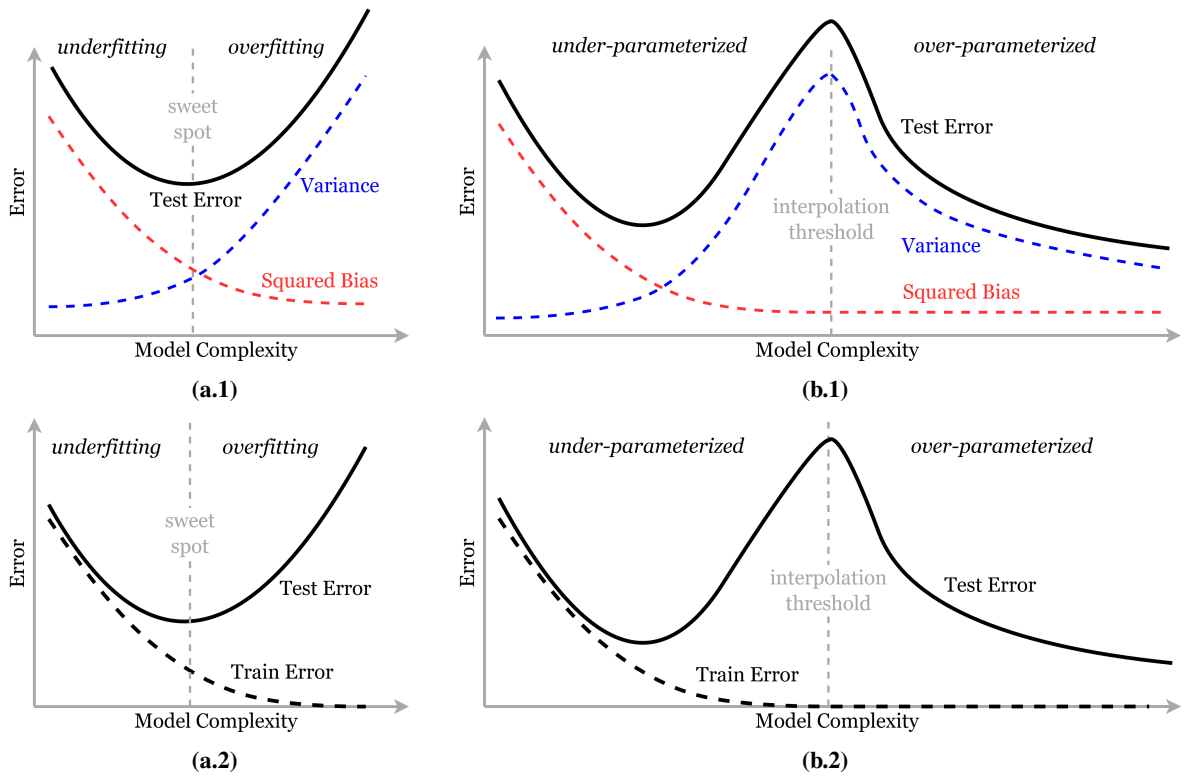


Figure 1: **(a) The classical U-shaped test error curve** arising from the bias-variance trade-off, implying an optimal model complexity at the "sweet-spot" between underfitting and overfitting. **(b) The double descent test error curve** incorporating both, the U-shaped test error curve and the observed behavior of over-parametrized models, separated by the interpolation threshold. **(a.1)** and **(b.1)** both decompose the respective test error into squared bias and variance (while neglecting irreducible error). **(a.2)** and **(b.2)** both depict the respective train error and test error curves. Figures adapted from [4, 18].

to resolve this apparent contradiction, Belkin et al. [4] proposed the **double descent** risk curve as an attempt to integrate the classical understanding of the U-shaped test error curve and the modern practice within a unified performance curve. The double descent curve describes the phenomenon where, as the model's complexity increases, the test error of a model first decreases, then increases, and then decreases again (see Fig. 1 (b.2)). By analyzing the behavior of bias and variance of neural networks, Yang et al. [18] provided an explanation for the occurrence of double descent. Their experiments show that while the bias is monotonically decreasing with the network width, the variance follows an unimodal curve, i.e., it increases, reaches a single highest value, and suddenly decreases. Although they observed that the resulting training error typically decreases monotonically, double descent of test error can occur in specific cases when bias and variance dominate in different regimes (see Fig. 1 (b.2)).

In short, the double descent risk curve challenges the classical wisdom of the bias-variance tradeoff to favor simpler models in order to prevent overfitting. Under certain conditions, a model may achieve optimal test error performance within the over-parametrized regime. Therefore, utilizing models with extremely high complexity may be beneficial in certain scenarios. However, as we will see, the occurrence of double descent depends on various different factors, e.g., the dataset, the model architecture, the training duration and other experimental conditions. Deliberately inducing the double descent curve is no trivial task, and sometimes it may not be achievable.

In this work, we aim to analyze the occurrence of double descent for two-layer fully connected neural networks (FCNN). We use the work of Belkin et al. [4] as a starting point by reproducing and discussing their results for FCNNs. We then analyze how the occurrence of the double descent phenomenon can be intensified and mitigated. That is, we evaluate the impact of their weight reuse scheme, we add label noise, we use dropout and weight decay to analyze the impact of frequently used regularization

techniques, and we employ different neural network activation functions.

The remainder of this work is organized as follows. We first review the research around the double descent phenomenon in more detail (Sec. 2). Afterward, we discuss the results achieved by Belkin et al. for FCNNs in more detail and, based on those, elaborate which experiments we perform (Sec. 3). We then present our experimental results and discuss those with regard to the question of how changing specific experimental conditions influence the occurrence of double descent (Sec. 4).

2 Related Work

We already introduced bias and variance as the two essential error types in ML, and the bias-variance tradeoff which implies that as model complexity increases, bias and variance monotonically decrease and increase, respectively [7]. Revisiting (a.2) of Fig. 1, we observe the classical U-shaped test error curve arising from the bias-variance tradeoff. The curve implies that the optimal generalization ability of a ML model (i.e., lowest achieved test error) is achieved by balancing underfitting and overfitting. In general, train errors well above zero are observed at this specific "sweet spot".

Contrary to this classical optimization approach, when training rich models such as deep neural networks modern practice is often to interpolate training data, while minimal test error is still obtained. To this end, Belkin at al. [4] proposed the double descent risk curve which incorporates the understanding of the classical U-shaped test error curve as well as the observation of modern practice (see Fig. 1 (b.1) and (b.2)). In the "classical" **under-parametrized** regime (i.e., models with relatively fewer parameters compared to the complexity of the task or dataset), the double descent test error aligns with the U-shaped test error curve resulting from the bias-variance tradeoff. Here, bias decreases and variance increases with increasing model complexity. Suddenly, in the "modern" **over-parametrized regime** (i.e., large number of parameters relative to the complexity of the task or dataset), the test error decreases *again* with increasing model complexity. The same holds for bias and variance [18]. The under-parametrized and over-parametrized regimes are separated by the **interpolation threshold**. Located at the interpolation threshold is the **"critical regime"** where models are just barely able to fit the training set [11]. When increasing the model complexity towards the interpolation threshold, the train error is minimized towards zero. At the interpolation threshold, the model achieves (near) perfect fit to the training data (i.e., near zero train error due to interpolation). Further increasing the model complexity past the interpolation threshold, the train error remains (near) zero, and the model eventually achieves optimal generalization.

Double descent research examines the occurrence of the phenomenon and its underlying causes in different settings. As Yang et al. [18] point out, there are two mysteries to the double descent. On the one hand, research is still lacking a general explanation for why the double descent risk curve occurs. As we already learned, one experimental approach is to explain the occurrence of double descent as a result of bias and variance dominating in different regimes (see Fig. 1 (b.1)). Other works theoretically analyze the double descent behavior in simplified settings via linear models such as linear regression [1, 3, 10]. On the other hand, the double descent phenomenon can not be observed robustly. For example, the theoretical and experimental results by Ba et al. [2] showed no double descent for two-layer neural networks when the first layer weights were optimized under certain initialization. When only the second layer coefficients were optimized, however, double descent was present. Nevertheless, double descent as a function of model complexity was observed in several works in a variety of tasks and optimization methods with different model architectures such as linear regression, random features regression, classification with random forests, FCNNs, as well as deep neural networks. For example, Nakkiran et al. [13] showed occurrence of double descent for different deep neural network architectures (ResNet, standard CNN, Transformer), datasets (CIFAR-100 and CIFAR-10) and optimizers (SGD and Adam). Interestingly, they showed that label noise, i.e., deliberately using incorrect labels for a subset of training samples, can be added in order to make the test error peak at the interpolation threshold more prominent.

Double descent has not only been studied as a function of model complexity, i.e., model-wise double descent. Nakkiran et al. [13] introduced a more general approach to double descent by defining the effective model complexity (EMC) of a training procedure as the maximum number of training samples n on which it can achieve close to zero training error. For the task of predicting labels, the EMC allows

the distinction of training procedures into the under-parametrized (EMC $< n$), over-parametrized (EMC $> n$) and critically parametrized regime (EMC $\approx n$). As the EMC not only depends on the model complexity, but also on other experimental conditions such as training duration, training samples and regularization, the EMC allows the analysis of double descent as a function of conditions different to, e.g., the number of trainable parameters of a neural network. For example, epoch-wise double descent, i.e., when increasing the training time and keeping the model complexity constant, has been observed for deep neural networks [13]. Therefore, *training longer can correct overfitting* in specific cases. Also, regularization-wise double descent is proven to exist in both theory and practice. Yilmaz and Heckel [19] showed that l_2-regularized models can exhibit double descent behavior as a function of the regularization strength. Finally, double descent is also studied regarding the question of how its occurrence can be mitigated. For example, Nakkiran et al. [12] showed that optimally-tuned l_2-regularization can mitigate double descent for more general models, including neural networks. Also, for a convolutional neural network, Nakkira et al. [13] observed that data augmentation mitigates the effect of double descent, and also shifts the interpolation threshold of the double descent curve to the right. Thus, the test error peak is achieved for a larger model complexity when augmenting training data.

3 Analyzing the Double Descent Phenomenon for Fully Connected Neural Networks

The double descent phenomenon was first postulated in generality by Belkin et al. in *Reconciling Modern Machine-Learning Practice and the Classical Bias–Variance Trade-Off* (2019) [4]. They demonstrated double descent of test error for decision trees, FCNNs, and random fourier feature models and random ReLU feature models, which both can be viewed as a class of two-layer neural network with fixed weights in the first layer. In this work, we focus on analyzing the double descent for two-layer FCNNs as a function of model complexity, measured in the number of trainable parameters. To this end, we first present the authors' work regarding FCNNs including their experimental results. Afterward, we present our experimental apparatus with which we aim to reproduce and extend their results.

3.1 Reconciling Modern Machine-Learning Practice and the Classical Bias–Variance Trade-Off

Interpolation Threshold for FCNNs

For a classification task with k target classes, Belkin et al. assume that in order to interpolate training data of n samples a neural network requires at least

$$(\#\, p)_{interp.} = n \cdot k \tag{1}$$

trainable parameters. A two-layer FCNN with an input layer of d units, a single hidden layer of h hidden units and k output units consists of

$$(\#\, p)_{fcnn} = (d+1) \cdot h + (h+1) \cdot k \tag{2}$$

trainable parameters. Equalizing (1) and (2) and solving for h results in

$$(\#\, h)_{interp.} = \frac{(n-1) \cdot k}{d+k+1} \tag{3}$$

Therefore, in a setting where n, k and d are fixed, the **interpolation threshold** is expected to be located approximately at the model with $(\#\, h)_{interp.}$ hidden units. FCNNs with $h < (\#\, h)_{interp.}$ hidden units belong to the **under-parametrized** regime, and FCNNs with $h \geq (\#\, h)_{interp.}$ hidden units belong to the **over-parametrized** regime.

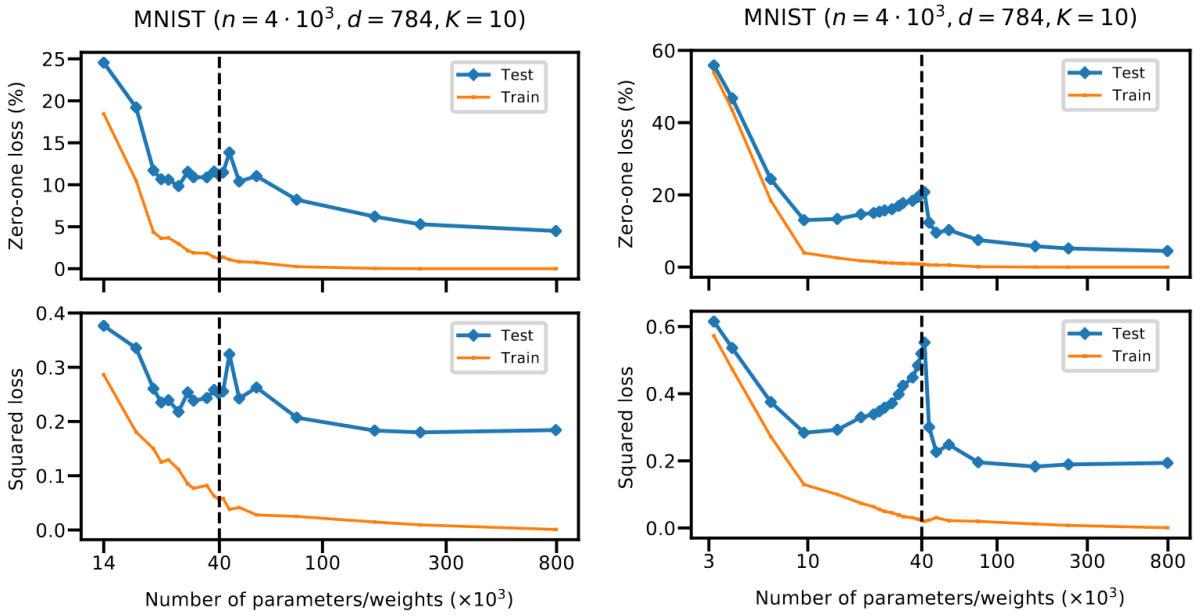
Weight Reuse Scheme

Belkin et al. argue that observing the double descent risk curve is difficult for neural networks due to the computational complexity of minimizing the training loss. This means that when using, e.g., stochastic gradient descent (SGD) for neural network optimization, monotonically decreasing training error can not be ensured within the under-parametrized regime due to SGD being sensitive to initialization. Therefore, a training procedure may result in a suboptimal solution. As a result, increasing the model complexity

may not always lead to a decrease in training risk. This high variability in both train and test error potentially masks the double descent curve. In order to ensure decreasing training error with increasing model complexity, the authors propose a **weight reuse scheme** in which most weights of larger networks are initialized with the weights from the previously trained (smaller) network, while the remaining weights are initialized with normally distributed random numbers. Belkin et al. used this weight reuse scheme only in the under-parametrized regime which results from their definition of the interpolation threshold. In our opinion, this is problematic due to the fact that the under-parametrized regime cannot be delimited so clearly in generality. In the over-parametrized regime, i.e., for FCNNs with more than $(\#\, p)_{interp.} = n \cdot k$ trainable parameters, weights are initialized randomly. The authors justify this by arguing that within the over-parametrized regime, models *typically have no difficulty in obtaining near-zero training risk*. We strongly advise against using the weight reuse scheme as it leads to misleading results (see Sec. 4).

Experimental Results

Figure 2 shows the experimental results obtained by Belkin et al. for FCNNs on the MNIST image dataset (see Sec. 3.2 for experimental setup). We observe a prominent double descent of test (zero-one classification and squared) loss when weights are reused (only) in the under-parametrized regime (b). At the same time, the train loss is constantly decreased, which demonstrates the effectiveness of the weight reuse scheme in this regard. When no weights are reused (a), i.e., all models of different sizes are initialized randomly, the double descent behavior is not as clearly observable, and we do not fully agree with the authors' conclusion that the *double descent behavior is still clearly discernible* here. Nevertheless, there is a broad test loss plateau around the interpolation threshold, i.e., in the critical regime, followed by a continuous decrease in test loss in the over-parametrized regime. Thus, due to the unusual test loss curves observed, it seems that the area around the interpolation threshold is indeed particularly interesting. However, due to the substantially different loss curves in (a) and (b), it is unclear to what extent the behavior in the under-parameterized regime is due to the reuse of weights only, and therefore whether a double descent was really observed. In particular, Belkin et al. did not provide experimental results when the weight reuse scheme is also used in the over-parameterized regime.



(a) No weight reuse (random initialization for all ranges of parameters)

(b) Weight reuse before interpolation threshold and random initialization after it.

Figure 2: **Double descent risk curve for FCNNs as observed by Belkin et al. [4]**. Both experiments were performed on the MNIST image dataset, while $n = 4,000$ images (of dimension $d = 784$ and $k = 10$ classes) were used during training. The black dotted line indicates the interpolation threshold located $(\#\, p)_{interp.} = n \cdot k$. Reported are zero-one classification loss and squared loss.

3.2 Experimental Apparatus

Dataset

We analyze the double descent phenomenon on the MNIST image dataset[1]. The MNIST database consists of grayscale handwritten digits of $k = 10$ classes, i.e., the digits from 0 to 9. Each image is of size 28×28. Same as performed by Belkin et al., each image is scaled to the interval of [0, 1] and flattened to a dimension of $d = 784$ before forwarded to a FCNN. MNIST originally consists of 60,000/10,000 for training/testing. However, we also use a subset of only 4,000 (randomly selected) train images for training of each FCNN, while using the full test split for testing. Therefore, we expect to observe the interpolation threshold for a model complexity around $(\#\ p)_{interp.} = n \cdot k = 40,000$ parameters, which corresponds with a two-layer FCNN consisting of $(\#\ h)_{interp.} \approx 50.3$ hidden units (see Equ. 3).

Hyperparameters

In the following, we present the hyperparameters as used by Belkin et al. for their experiments on FCNNs. If not stated differently (see Procedure), we use the same parameters in all our experiments. That is, each model is trained to minimize the squared loss (also known as mean squared error) on the train set, and the same experiment is repeated five times (i.e., each FCNN with a specific number of hidden nodes is trained and tested five times). We present the mean, and in addition to that the standard deviation of the achieved train and test errors. Each model is trained for 6,000 epochs using stochastic gradient descent (SGD) optimization with momentum of 0.95. Belkin et al. did not specify learning rate nor batch size, for which we use 0.1 and 128 respectively. For networks smaller than the interpolation threshold, the step size is decayed by 10% after each of 500 epochs and training is stopped after classification error reaches zero (or after 6000 epochs).

Uniform Xavier initialization [8] is used to initialize all FCNN weights. However, when weights are reused (see Section 3.1) from a smaller trained network, the remaining weights are initialized with normally distributed random numbers with mean 0 and variance 0.01. Finally, we use ReLU activation between hidden and output layer. Note that Belkin et al. did not state whether or which activation function they used. In particular, Belkin et al. did not use dropout, weight decay, nor an activation function for the output layer.

Procedure

In all our experiments, we increase the model complexity by varying the number of hidden nodes over $h \in \{2, 4, \ldots, 40, 41, \ldots, 60, 65, \ldots, 150\}$. Therefore, our FCNNs consist between 1,600 and 199,260 trainable weights/parameters. For comparison, Belkin et al. varied the model complexity between 3,000 and 800,000 parameters, i.e., their FCNNs consisted of at least 3 up to 1,000 hidden nodes.

Results Validation. First, we aim to reproduce the results achieved by Belkin et al. for FCNNs on the MNIST dataset using the hyperparameters as specified above in both settings, without reusing weights and when reusing weights in the under-parametrized regime.

Continuous Weight Reuse and Deactivating Step Size Reduction. We found that the mechanisms of step size reduction and stopping training after classification error reaches zero for networks smaller than the interpolation threshold have no noticeable impact, and yet their potential influences are difficult to estimate. Therefore, we deactivate these mechanisms for this and all subsequent experiments. Furthermore, to understand the impact of the weight reuse scheme as proposed by Belkin et al., we repeat the experiments without reusing weights and when reusing weights in the under-parametrized regime, and in addition to that when reusing weights for all models (i.e., in both, the under-parametrized and over-parametrized regimes). After this, we will not use the weight reuse scheme in any of the subsequent experiments.

[1]The MNIST digits dataset is available at: `http://yann.lecun.com/exdb/mnist/`.

Label Noise. In their work, Nakkiran et al. [13] observed double descent most strongly in settings with label noise, i.e., when deliberately using random incorrect training labels. They motivate the use of label noise by noting that it is a tool to increase the amount of model mis-specification, i.e., *for making distributions "harder"*. Therefore, we can use label noise to make the test error peak at the interpolation threshold more prominent. In this experiment, we follow Nakkiran et al. and add label noise, i.e., we deliberately use random incorrect labels for a subset of 10% and 20% of training samples.

Dropout. Dropout is a frequently used regularization method to prevent overfitting of neural networks [16]. By randomly dropping units along with their connections during training, dropout encourages a neural network to rely on a more diverse set of features and prevents it from relying too heavily on single features or neurons. In this experiment, we use dropout of 0.25 between hidden and output layer while adding 20% label noise.

Weight Decay. Another frequently used regularization technique is weight decay (or weight regularization), i.e., introducing a penalty term to the loss function during training which encourages a neural network to have smaller weights [14]. In this experiment, we use l_2 weight regularization with magnitude of 0.0001 while adding 20% label noise.

Softmax Activation. Belkin et al. did not apply an activation function to the output of a FCNN. However, softmax activation [15] is commonly used for neural networks in multi-class classification tasks to convert output vectors of raw scores into probability distributions. In this experiment, we use softmax activation as the FCNNs' final activation function while adding 20% label noise.

4 Results and Discussion

We now present the experimental results. Each of the following figures consists of the achieved train and test loss curves of zero-one and squared loss in a specific setting. Since each experiment is repeated five times for a specific FCNN size, the dotted line of each curve indicates the mean of achieved losses, and the colored area around the dotted line indicates the standard deviation of achieved losses.

Results Validation. We first present the results of our attempt to reproduce the results obtained by Belkin et al. for FCNNs.

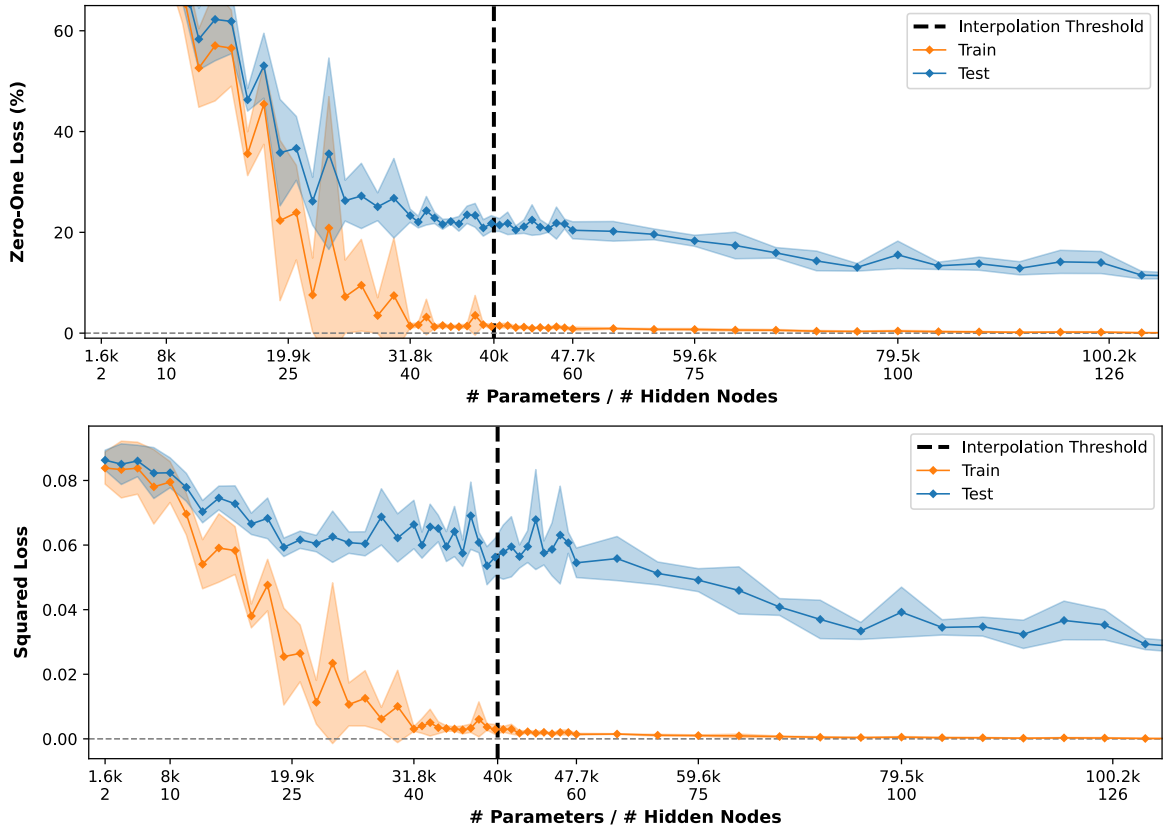


Figure 3: Experimental results (mean and variance of zero-one and squared loss) when not using the weight reuse scheme (i.e., random initialization for all model sizes) with the aim of reproducing Fig. 2 (a) obtained by Belkin et al., using the same hyperparameters on the MNIST dataset ($d = 784$; $n = 4,000$; $k = 10$). The black-dotted line indicates the interpolation threshold at $(\#\ p)_{interp.} = n \cdot k$.

When not reusing weights (see Fig. 3), we observe that the model with 40 hidden nodes is already able to achieve train losses close to zero for both, zero-one and squared loss, even though the interpolation threshold is located at $(\#\ h)_{interp.} \approx 50.3$. Due to the very low to non-existent standard deviation of train losses for this and all larger models, we know that the models are able to achieve this near-optimal performance consistently on the train set. Although train loss is slightly reduced for increasing model complexity in the over-parametrized regime, no model is able to perfectly interpolate the training data. Looking at the test curves, there is a trend of only sightly reduced test loss when increasing the number of hidden nodes from 40 to 60. When further increasing the model size, we observe a more prominent decrease in both test losses. Comparing our results with those obtained by Belkin et al. when no weights are reused (see Fig. 2 (a)), we note that our achieved train and test loss curves are very similar. However, the loss values differ in their absolute value. For example, in this setting all our FCNNs achieved test squared losses are smaller than 0.1, whereas the trained models of Belkin et al. only achieved test squared losses larger than 0.2. This may be due to different loss function or neural network implementations (see Sec. 3.2, Hyperparameters). Nonetheless, our obtained loss curves show a similar trend as those of Belkin

et al., and therefore allow the same conclusions. In particular, our achieved test loss curves also have a broad plateau in the critical regime around the interpolation threshold, followed by sudden decrease of test loss in the over-parametrized regime. However, a double descent curve is not yet clearly discernible.

Figure 4 shows the experimental results when reusing weights in the under-parametrized regime. Due to the identical weight initialization for models larger than the interpolation threshold, we achieve identical train and test losses in the over-parametrized regime as in Figure 3. However, we now observe that both train losses are constantly decreased for increasing model size, which demonstrates the effectiveness of the weight reuse scheme. Furthermore, in the under-parametrized regime, both test losses constantly increase when approaching the interpolation threshold. Again, our results are consistent with those of Belkin et al. (see Fig. 2 (b)), and we also observe a double descent behavior of zero-one and squared test loss. As we already discussed (see Sec. 3.1, Experimental Results), however, these results do not permit any definitive conclusions regarding the occurrence of double descent since it is still unclear to what extent the behavior in the under-parameterized regime is due to the reuse of weights only.

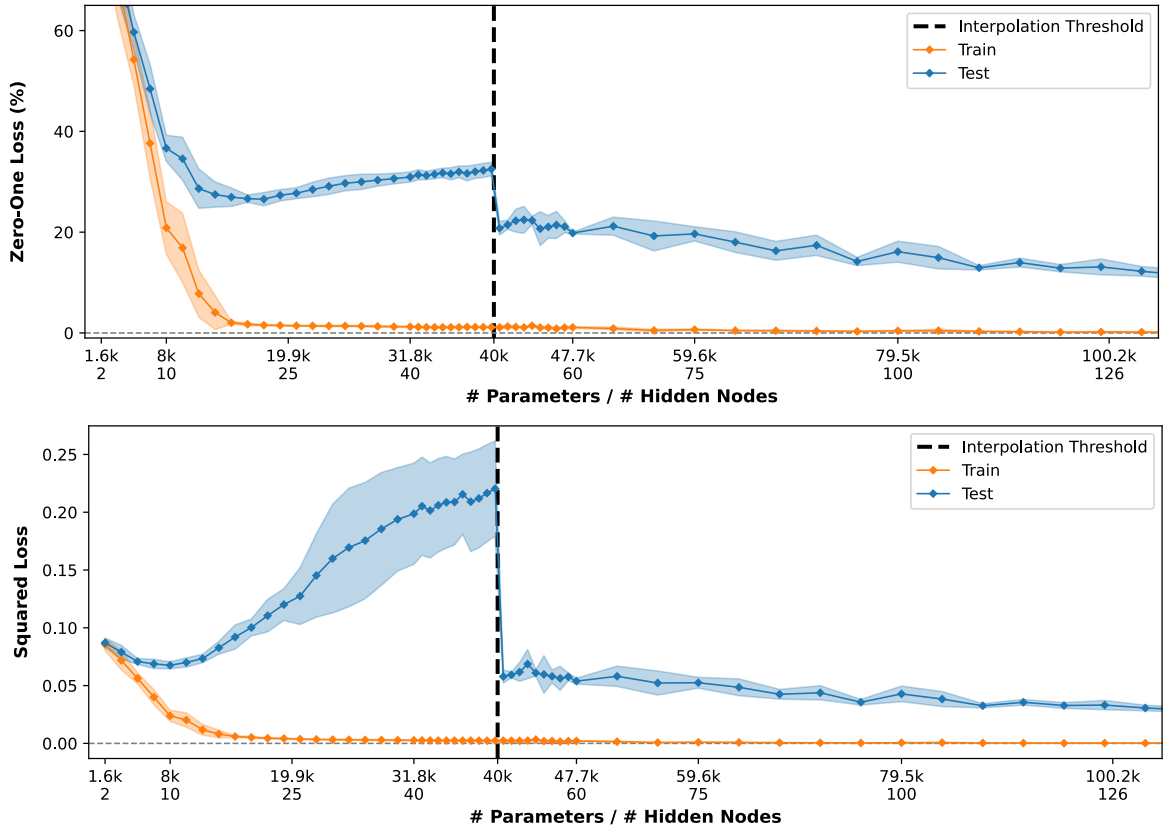


Figure 4: Experimental results on the MNIST dataset when using the weight reuse scheme in the under-parametrized regime (i.e., weight reuse before interpolation threshold and random initialization after it) with the aim of reproducing Fig. 2 (b) obtained by Belkin et al..

Continuous Weight Reuse and Deactivating Step Size Reduction. In this experiment, we deactivated the mechanisms of step size reduction and stopping training after classification error reaches zero for models smaller than the interpolation threshold (see Sec. 3.2, Hyperparameters). When comparing Figure 5 (no weight reuse) and Figure 6 (weight reuse in the under-parametrized regime) with the corresponding results when the aforementioned mechanisms are still used (see Fig. 3 and Fig. 4, respectively), we conclude that the mechanisms indeed have no noticeable impact on the resulting train and test loss curves. Only upon close examination, minor performance differences become apparent. Due to their limited influence on the resulting loss curves, we deactivate both mechanisms for all subsequent experiments.

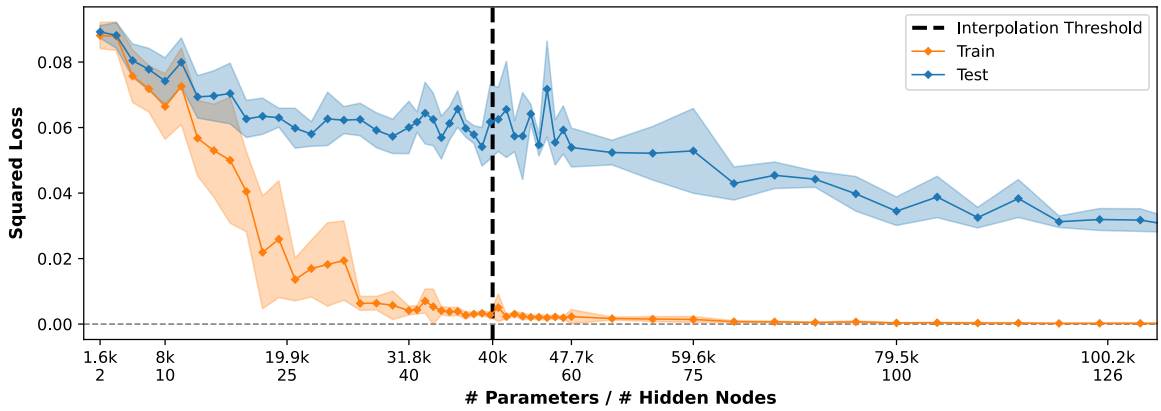


Figure 5: Experimental results on the MNIST dataset when deactivating step size reduction for networks smaller than the interpolation threshold (no weight reuse).

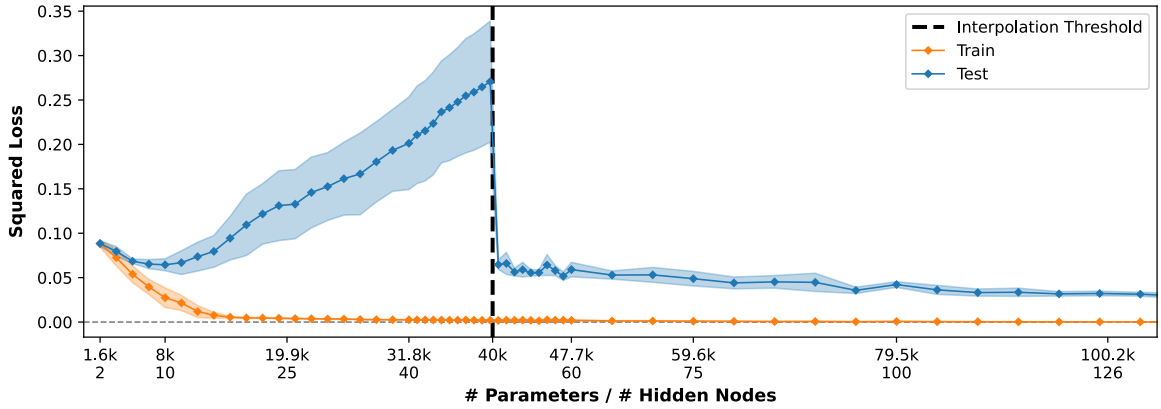


Figure 6: Experimental results on the MNIST dataset when deactivating step size reduction for networks smaller than the interpolation threshold (weight reuse in the under-parametrized regime).

More interestingly, however, are the results when reusing weights across all networks, i.e., in both, the under-parametrized and over-parametrized regimes (see Fig. 7). As previously observed, both test losses increase when approaching the interpolation threshold. When the network size is now increased beyond the interpolation threshold and we continue to reuse weights, test one-zero loss remains approximately at the same level, while the test squared loss continues to exhibit a slight increase.

Although not presented here, we found that when reusing weights, the increase in test loss can be controlled by adjusting the extent to which a model is enlarged, i.e., by the amount of hidden nodes that are randomly initialized in addition to the reused weights from the previous network: the increase in test loss is higher when fewer weights are initialized randomly. This may be due to the network being initialized in proximity to a local minimum when a smaller number of weights are initialized randomly. This explains the limited increase in squared test loss for networks with more than 60 hidden nodes (5 randomly initialized hidden nodes in addition to the trained weights of the previous network) in contrast

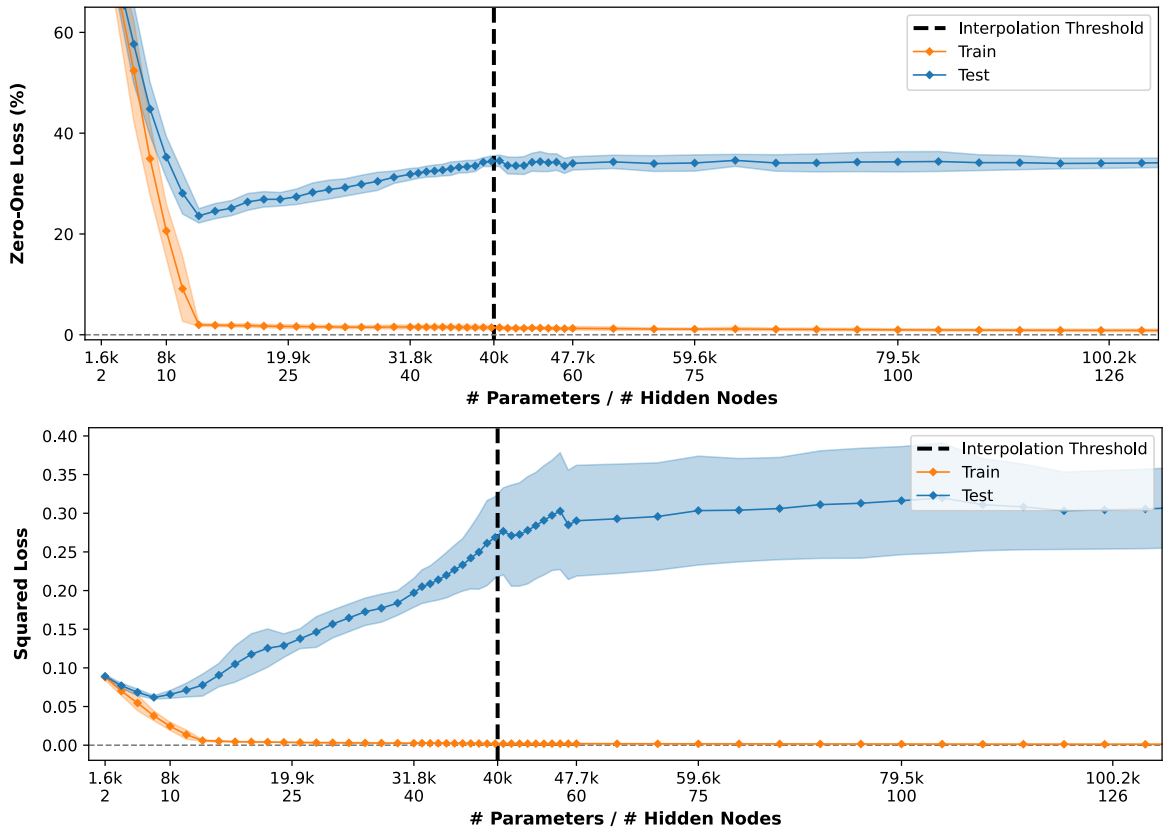


Figure 7: Experimental results on the MNIST dataset when deactivating step size reduction for networks smaller than the interpolation threshold (weight reuse for all networks, i.e., in the under-parametrized and over-parametrized regimes).

to the networks in the critical regime around the interpolation threshold (only one randomly initialized hidden node).

Belkin et al. proposed to reuse weights in order to achieve near-zero training risk for models below the interpolation threshold. The results show that the weight reuse scheme can indeed accomplish this. Revisiting the results when reusing weights in the under-parametrized regime (see Fig. 2 (b), Fig. 4, or Fig. 6), one can be led to the conclusion to observe an actual double descent due to the continuous decrease in train loss and double descent in test loss for increasing network sizes. However, it now becomes clear that the weight reuse scheme, in the way Belkin et al. proposed to used it, leads to misleading results when attempting to observe the double descent phenomenon. Figure 7 shows that when weights are reused across all networks, both test losses increase for increasing model size, i.e., the train and test loss curves do not provide evidence of a double descent. As a result of this experiment, we do not use the weight reuse scheme in any of the subsequent experiments.

Label Noise. Our experimental results when adding 10% and 20% of label noise are shown in Figure 8 and Figure 9, respectively. As expected, we observe that by adding label noise all loss curves are shifted upwards, i.e., all networks achieve worse performances. Furthermore, models of the critical and over-parametrized regimes are no longer able to interpolate the training data. In contrast to our objective, we find that even when adding 20% of label noise (see Fig. 9), the double descent phenomenon is not discernible for zero-one loss. On the other hand, adding 10% label noise is sufficient to reveal a double descent of test squared loss with a relatively consistent decline in train squared loss (see Fig. 8). The increase and decrease of test squared loss around the interpolation threshold is even more prominent when 20% label noise is added (see Fig. 9). However, when being very precise, due to the test squared loss not decreasing across multiple network sizes in the under-parameterized regime as previously observed (e.g., when using 10% label noise), it is not accurate anymore to characterize this as a double descent. Nevertheless, adding label noise proves to be highly effective for inducing the occurrence of the double descent phenomenon.

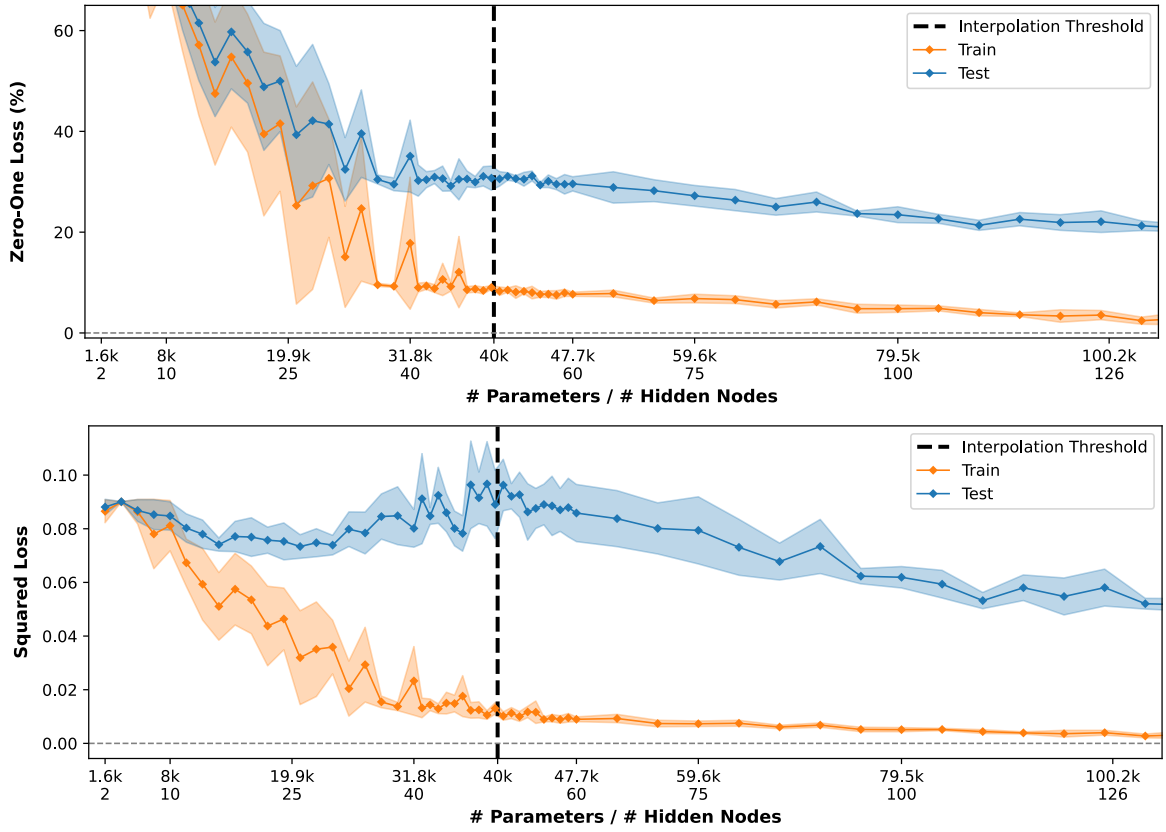


Figure 8: Experimental results on the MNIST dataset when adding 10% label noise (no weight reuse).

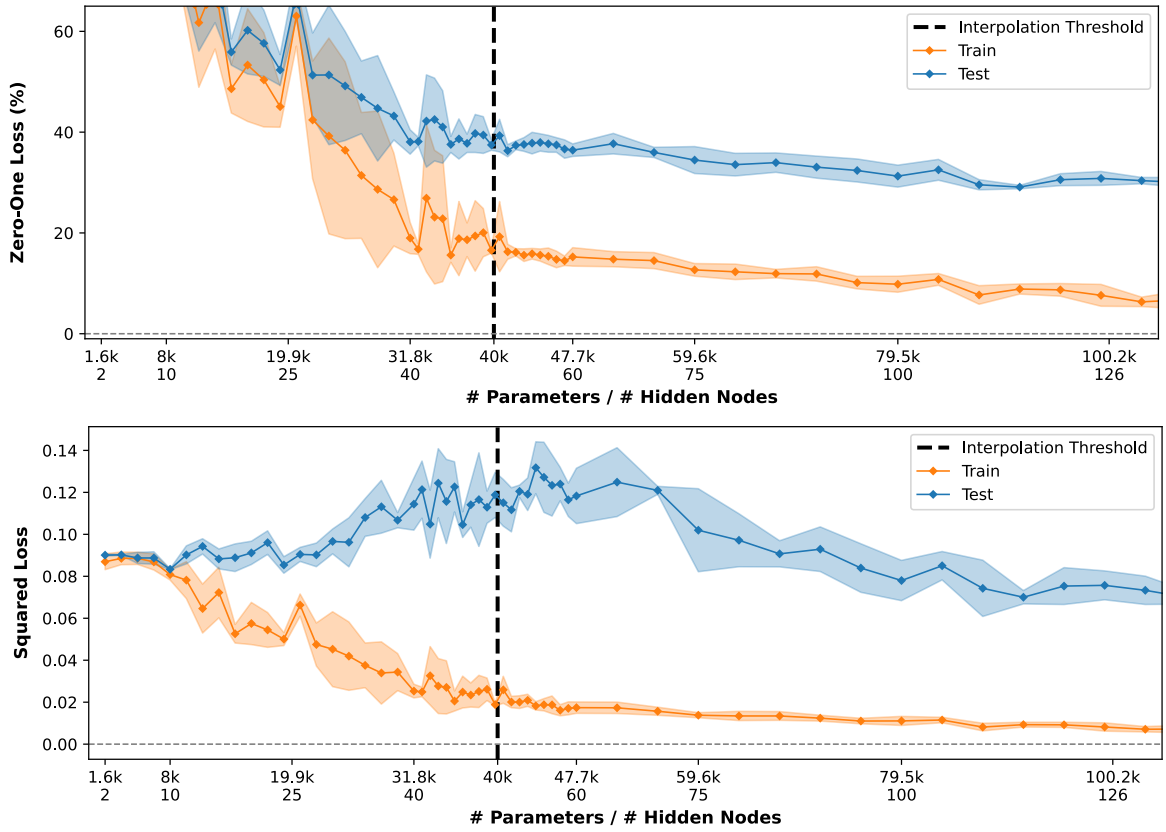


Figure 9: Experimental results on the MNIST dataset when adding 20% label noise (no weight reuse).

Dropout, Weight Decay and Softmax Activation. Different works have proven that regularization techniques such as l_2 weight regularization [12] and data augmentation [13] can mitigate double descent for neural networks. Following the successful induction of double descent by adding label noise, we now analyze how the utilization of regularization techniques, specifically dropout and weight decay, impacts the resulting training and test loss curves.

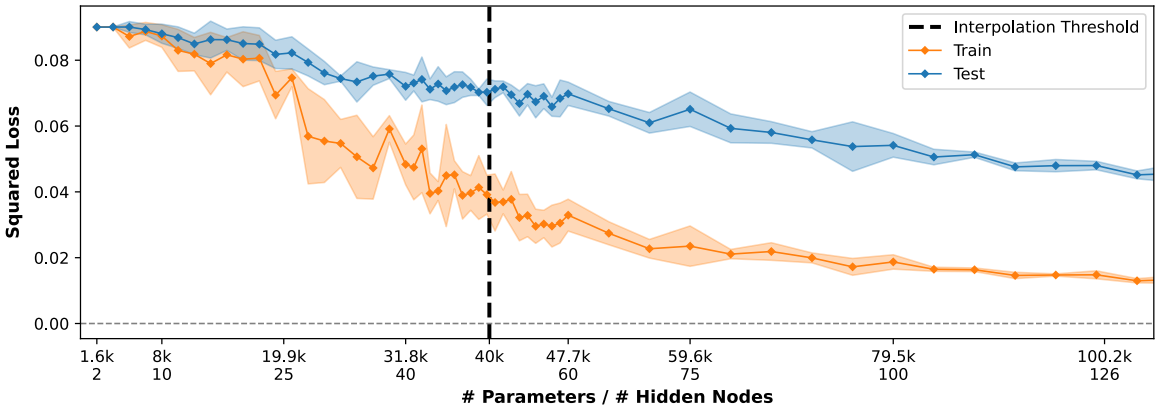


Figure 10: Experimental results on the MNIST dataset when using dropout of 0.25 between hidden and output layer while adding 20% label noise (no weight reuse).

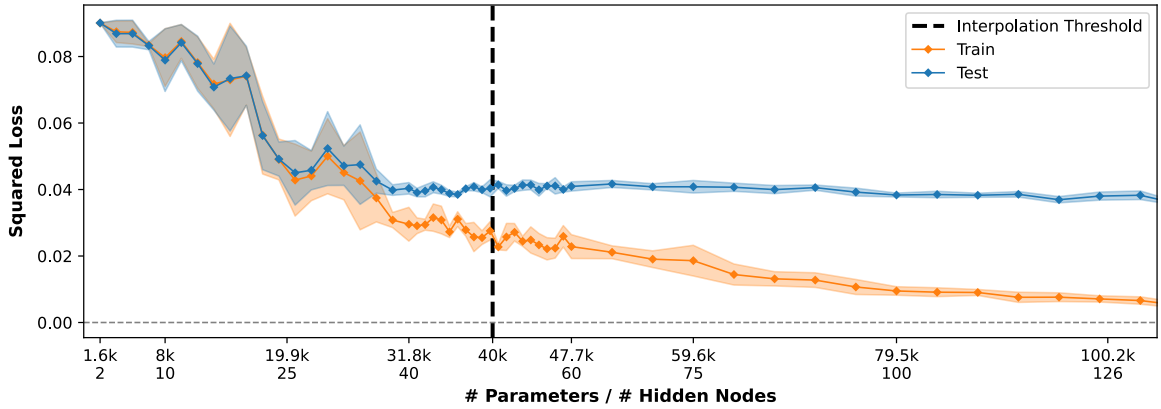


Figure 11: Experimental results on the MNIST dataset when using L_2 weight regularization with magnitude of 0.0001 while adding 20% label noise (no weight reuse).

Figure 10 shows the experimental results when using dropout of 0.25 between hidden and output layer while adding 20% label noise. In comparison to Figure 9, the increase and decrease of test squared loss around the interpolation threshold, and thus also the double descent behavior, can not be observed anymore. Furthermore, the test squared loss performance improves for all network sizes when dropout is used. Figure 11 shows the experimental results when using l_2 weight regularization with magnitude of 0.0001 while adding 20% label noise. Again, the double descent of test squared loss is no longer existent. Interestingly, the performance improves drastically for all model sizes when using weight decay instead of dropout[2]. In conclusion, we effectively alleviated the occurrence of the double descent phenomenon by employing regularization techniques, namely dropout and weight decay.

Following Belkin et al., no activation function has been applied to the output of any FCNN throughout all the experiments conducted thus far. In multi-class classification tasks, however, it is common to convert output vectors of raw scores into probability distributions. Figure 12 shows the experimental results of using softmax activation as the FCNNs' final activation function while adding 20% label noise.

[2]Please note that this does not necessarily imply that using weight decay will consistently result in superior performance compared to dropout in our setting. We did not perform a comprehensive hyperparameter optimization. Therefore, it is possible that the l_2 weight regularization parameter was selected more favorably by random compared to the dropout rate.

Again, the double descent of test squared loss is no longer existent. Interesting observations (not related to double descent) can be made when comparing the results with the loss curves when using weight decay (see Fig. 11). First, the variance of achieved losses is minimal for all model sizes when using softmax activation. Furthermore, using softmax activation instead of weight regularization drastically improves performance of networks of the under-parametrized regimes.

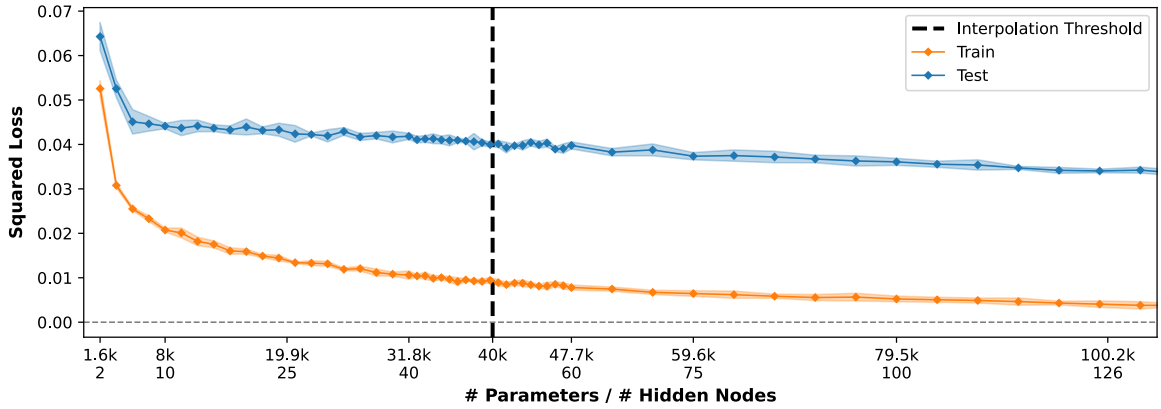


Figure 12: Experimental results on the MNIST dataset when using softmax activation as the FCNNs' final activation function while adding 20% label noise (no weight reuse).

5 Key Results and Conclusion

We successfully reproduced and extended the results obtained by Belkin et al. for FCNNs [4]. By comparing the resulting loss curves when initializing the weights of all networks randomly, reusing the weights in the under-parametrized regime only, and reusing weight across all networks, it became clear that the weight reuse scheme is not a suitable means for providing conclusions about the presence of the double descent phenomenon. Consequently, we recommend against employing the weight reuse scheme in this context. On the contrary, we found that adding label noise is an effective approach to make the presence of double descent apparent. Adding 10% label noise was sufficient to reveal a double descent of test squared loss with a relatively consistent decline in train squared loss. Furthermore, we observed that the increase and decrease of test squared loss around the interpolation threshold is more prominent when a higher amount of label noise is added. Finally, we effectively alleviated the occurrence of double descent by employing regularization techniques, namely dropout and l_2 weight regularization, and by using softmax activation as the FCNNs' final activation function. For future work, it would be of great interest to investigate whether other manifestations of double descent, such as epoch-wise or regularization-wise double descent, can be observed within the presented experimental setup.

References

[1] Madhu S. Advani, Andrew M. Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.08.022. URL https://www.sciencedirect.com/science/article/pii/S0893608020303117.

[2] Jimmy Ba, Murat Erdogdu, Taiji Suzuki, Denny Wu, and Tianzong Zhang. Generalization of two-layer neural networks: An asymptotic viewpoint. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gBsgBYwH.

[3] Peter L. Bartlett, Philip M. Long, Gá bor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, apr 2020. doi: 10.1073/pnas.1907378117. URL https://doi.org/10.1073%2Fpnas.1907378117.

[4] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, jul 2019. doi: 10.1073/pnas.1903070116. URL https://doi.org/10.1073%2Fpnas.1903070116.

[5] Alon Brutzkus and Amir Globerson. Over-parameterization improves generalization in the XOR detection problem. *CoRR*, abs/1810.03037, 2018. URL http://arxiv.org/abs/1810.03037.

[6] Pedro Domingos. A unifeid bias-variance decomposition and its applications. pages 231–238, 01 2000.

[7] Stuart Geman, Elie Bienenstock, and René Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58, 01 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.1. URL https://doi.org/10.1162/neco.1992.4.1.1.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

[9] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *CoRR*, abs/1811.06965, 2018. URL http://arxiv.org/abs/1811.06965.

[10] Vidya Muthukumar, Kailas Vodrahalli, and Anant Sahai. Harmless interpolation of noisy data in regression. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2299–2303, 2019. doi: 10.1109/ISIT.2019.8849614.

[11] Preetum Nakkiran. More data can hurt for linear regression: Sample-wise double descent, 2019.

[12] Preetum Nakkiran, Prayaag Venkat, Sham M. Kakade, and Tengyu Ma. Optimal regularization can mitigate double descent. *CoRR*, abs/2003.01897, 2020. URL https://arxiv.org/abs/2003.01897.

[13] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: where bigger models and more data hurt*. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, dec 2021. doi: 10.1088/1742-5468/ac3a74. URL https://dx.doi.org/10.1088/1742-5468/ac3a74.

[14] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015435. URL https://doi.org/10.1145/1015330.1015435.

[15] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning, 2018.

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[17] Ulrike von Luxburg and Bernhard Schoelkopf. Statistical learning theory: Models, concepts, and results, 2008.

[18] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[19] Fatih Furkan Yilmaz and Reinhard Heckel. Regularization-wise double descent: Why it occurs and how to eliminate it, 2022.