

# UNIFIED NEURAL NETWORK SCALING LAWS AND SCALE-TIME EQUIVALENCE

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

As neural networks continue to grow in size but datasets might not, it is vital to understand how much performance improvement can be expected: is it more important to scale network size or data volume? Thus, neural network scaling laws, which characterize how test error varies with network size and data volume, have become increasingly important. However, existing scaling laws are often applicable only in limited regimes and often do not incorporate or predict well-known phenomena such as double descent. Here, we present a novel theoretical characterization of how *three* factors — model size, training time, and data volume — interact to determine the performance of deep neural networks. We first establish a theoretical and empirical equivalence between scaling the size of a neural network and increasing its training time proportionally. Scale-time equivalence challenges the current practice, wherein large models are trained for small durations, and suggests that smaller models trained over extended periods could match their efficacy. It also leads to a novel method for predicting the performance of large-scale networks from small-scale networks trained for extended epochs, and vice versa. We next combine scale-time equivalence with a linear model analysis of double descent to obtain a unified theoretical scaling law, which we confirm with experiments across vision benchmarks and network architectures. These laws explain several previously unexplained phenomena: reduced data requirements for generalization in larger models, heightened sensitivity to label noise in overparameterized models, and instances where increasing model scale does not necessarily enhance performance. Our findings hold significant implications for the practical deployment of neural networks, offering a more accessible and efficient path to training and fine-tuning large models.

## 1 INTRODUCTION

Progress in artificial intelligence (AI) has relied heavily on the dramatic growth in the size of models and datasets. An active area of research focuses on understanding how test error decreases with increases in model and data size. This work has led to the development of scaling laws which posit that test error decreases as a power law with both. However, several theoretical aspects remain unclear. One significant gap is understanding how test error and the existing scaling laws change as the training time is varied (Kaplan et al., 2020; Bahri et al., 2021; Rosenfeld et al., 2020; Sharma & Kaplan, 2022).

The practical relevance of this question is clear: under a fixed compute budget, what is the optimal balance between scaling the model size and dataset volume, and what is the right amount of training for a given data volume? This is particularly relevant in the context of large language models (LLMs), which are often trained for a single epoch, raising questions about the potential efficacy of training smaller models for longer (more epochs).

Furthermore, current scaling laws do not account for other well-known phenomena in learning, such as double descent (Belkin et al., 2019), in which model performance exhibits non-monotonic changes with respect to training data volume, model size, and training time. In particular, double descent theory predicts that test error should increase rapidly at the *interpolation threshold*, the point at which the model interpolates the training set (Nakkiran et al., 2021; Advani & Saxe, 2017). Like scaling laws, current theories of double descent leave several empirical phenomena unexplained:

054 past explanations of double descent require it to occur, but empirically double descent is often not  
055 observed; it is unclear whether the interpolation threshold should grow or shrink with model size;  
056 prior theory does not explain why models in the infinite-parameter limit sometimes perform worse  
057 than their finite-parameter counterparts.

058 We seek the simplest possible unified framework in which to understand learning with respect to  
059 model size, data volume, and training time. In doing so, we aim to capture the essential scaling  
060 properties of learning in deep neural networks. Our results unify double descent with scaling laws  
061 and help to understand when models are sufficiently large for effective performance, how double  
062 descent is affected by varying training time and model size, and the variability and shape of loss  
063 curves across different problems.

064 The contributions of this paper are multi-fold:

- 066 • We theoretically and empirically demonstrate that scaling the size of a neural network is  
067 functionally equivalent to increasing its training time by a proportional factor.
- 068 • Leveraging this insight, we 1) predict the performance of large-scale networks using small-  
069 scale networks trained for many epochs and 2) predict the performance of networks trained  
070 for many epochs using the performance of large networks trained for one epoch.
- 071 • Using scale-time equivalence, we propose a unified scaling law for deep neural networks  
072 that provides a new explanation for parameter-wise double descent: double descent occurs  
073 when small models, which effectively train slower than larger models, acquire noisy data  
074 features.
- 075 • Through experiments conducted on standard vision benchmarks across multiple network  
076 architectures, we validate that our model explains several previously unexplained phenom-  
077 ena, including 1) the reduced data requirement for generalization in larger models, 2) the  
078 large impact of label noise on overparameterized models, 3) why error of overparameter-  
079 ized models often *increases* with scale.

## 082 2 RELATED WORK

### 083 2.1 SCALING LAWS

084  
085  
086 Neural network scaling laws describe how generalization error scales with data and model size. A  
087 number of works have observed power-law scaling with respect to data and model size (Kaplan et al.,  
088 2020; Rosenfeld et al., 2020; Clark et al., 2022), which has been explained theoretically (Bahri et al.,  
089 2021; Sharma & Kaplan, 2022; Hutter, 2021; Paquette et al., 2024).

090 However, other work demonstrates that model scale may not be sufficient to predict model perfor-  
091 mance (Tay et al., 2022), and casts doubt on power laws as the best model of error rate scaling (Alab-  
092 dumohsin et al., 2022; Bansal et al., 2022; Mahmood et al., 2022). Moreover, scaling with respect to  
093 training time, holding data volume fixed, remains poorly understood. These observations highlight  
094 the need for a more general framework that can predict model performance under many settings.

### 095 096 097 2.2 DOUBLE DESCENT

098 Double descent is an empirically observed phenomenon in which generalization error of machine  
099 learning models with respect to training data volume, model size, and training time exhibits an  
100 initial decrease, followed by a brief, sharp increase followed by a final decrease (Belkin et al., 2019;  
101 Nakkiran et al., 2021). Double descent with respect to model and data size has been theoretically  
102 understood as occurring due to a high degree of overfitting at the *interpolation threshold*, the point  
103 at which the model size is just sufficient to interpolate the training data (Adlam & Pennington, 2020;  
104 D’Ascoli et al., 2020; Belkin et al., 2019; Advani & Saxe, 2017). Typically, this work uses tools  
105 from random matrix theory to explain double descent for random feature models, in which linear  
106 regression maps a random, fixed feature pool to the desired output (Simon et al., 2024; Atanasov  
107 et al., 2024; Adlam et al., 2022; Bordelon et al., 2024; Maloney et al., 2022; Mei & Montanari, 2019;  
Ali et al., 2019; Lin et al., 2024).

108 However, these models generally do not explain double descent behavior in terms of training time.  
 109 Originally, Nakkiran et al. (2019; 2021) hypothesized that training models for longer results in an  
 110 effective increase in model size, thus allowing time-wise double descent to be explained in the same  
 111 way as scale-wise double descent. Recent works have provided an alternate explanation: time-wise  
 112 double descent occurs due to data features being learned at varying scales (Pezeshki et al., 2022;  
 113 Heckel & Yilmaz, 2021; Stephenson & Lee, 2021). Error rises as models overfit to quickly-learned  
 114 noisy features, but then falls as models more slowly learn signal features. In this work, we unify this  
 115 explanation of time-wise double descent with the traditional account of double descent.

### 117 3 SCALE-TIME EQUIVALENCE IN NEURAL NETWORKS

119 In this section, we demonstrate that model size and training time may be traded off with each other.  
 120 This result is consistent with and generalizes prior results demonstrating that models learn functions  
 121 of increasing complexity over time (Nakkiran et al., 2019). We establish the result theoretically  
 122 in a simplified model and validate it empirically in neural networks across several datasets and  
 123 architectures.

#### 125 3.1 RANDOM SUBSPACE MODEL

126 Following the lines of prior double descent analyses in random feature models, we construct a ran-  
 127 dom subspace model to demonstrate scale-time equivalence theoretically. Consider a large  $P$  dimen-  
 128 sional model with parameters  $\beta$ , such that the function represented by the model depends only on  
 129 a low-dimensional linear projection of  $\beta$ . This is reasonable for neural networks: it is well known  
 130 that neural networks trained by stochastic gradient descent tend towards flat minima of their loss  
 131 landscapes, thus revealing many redundant dimensions in the network (i.e. only a low-dimensional  
 132 subspace affects the network output). Specifically, we denote the low-dimensional projection as  
 133  $\alpha \in \mathbb{R}^r$  where  $r < P$  which is constructed as:

$$134 \alpha = K\beta \tag{1}$$

136 where  $K \in \mathbb{R}^{r \times P}$  is a fixed projection matrix.

137 Assume that we can only control a random  $p$ -dimensional ( $P > p > r$ ) linear subspace of the  
 138 large model. This may again be a reasonable assumption for model classes such as neural networks:  
 139 smaller neural networks can naturally be viewed as linear subspaces of larger neural networks that  
 140 contain them architecturally (i.e. have a larger width and depth) (Hall & Li, 1993). We denote the  
 141 parameters of our controllable  $p$  dimensional subspace model as  $\theta \in \mathbb{R}^p$ , where:

$$142 \beta = R\theta + \beta_0 \tag{2}$$

144 for a random matrix  $R \in \mathbb{R}^{P \times p}$  with elements drawn iid from a unit Gaussian and with  $\beta_0 \in \mathbb{R}^P$   
 145 fixed. We will represent time  $t$  with subscript  $t$  (i.e.  $\beta_0$  denotes  $\beta$  at time 0). Observe that any  $p$   
 146 dimensional affine subspace of  $\mathbb{R}^P$  may be represented in the form above for some choice of  $R$  and  
 147  $\beta_0$ .

148 Under these assumptions, we can show that under gradient descent, increasing the scale  $p$  of the  
 149 model is equivalent to increasing training time:

150 **Theorem 1.** *We denote the loss as a function of  $\alpha$ :  $L \in \mathbb{R}^r \rightarrow \mathbb{R}$ . Suppose  $L$  has Lipschitz constant  
 151  $l$  and its second derivative has Lipschitz constant  $h$ . Suppose that continuous time gradient flow is  
 152 applied to  $\theta$  with learning rate  $\eta$  from initialization  $\theta = 0$ . We denote  $\alpha_t = K(R\theta_t + \beta_0)$  where  $\theta_t$   
 153 are the parameters at time  $t$ . Denote  $A_t \in \mathbb{R}^r$  as the solution to:*

$$154 \dot{A}_t = -\eta K K^T \nabla L(A_t) \tag{3}$$

156 with initial condition  $A_0 = K\beta_0$ . Note that  $A_t$  does not depend on  $p$ . Then, with probability  $1 - \epsilon$ :

$$158 \|\alpha_t - A_{pt}\| \leq \frac{l\sqrt{\|K\|_F^4 + \|K K^T\|_F^2}}{h\sqrt{p\epsilon}\|K K^T\|} (e^{\eta p t h \|K K^T\|} - 1) \tag{4}$$

160 See Appendix A for a proof. The theorem implies closeness between the function implemented by  
 161 the network, represented by  $\alpha_t$ , and another quantity  $A_{pt}$  which only depends on the product  $pt$ . In

162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215

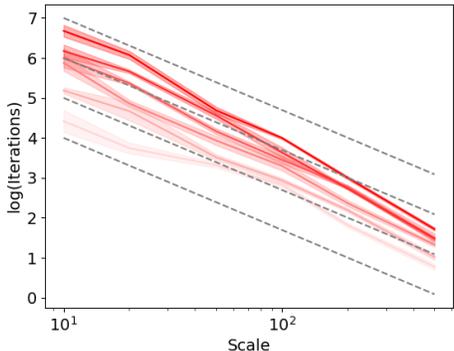


Figure 1: **Proportional trade-off between model scale and training time: testing the prediction on a linear model.** Red lines indicate tradeoff curves between number of training iterations and model size. Curves are computed by, for each model size, measuring the minimum amount of training time necessary to achieve different loss levels. Different curves indicate different performance thresholds; darker lines indicate a smaller error threshold. Margins indicate standard errors over 5 trials. Grey dashed lines represent 1:1 proportionality between scale and training iterations.

other words, the learned model only depends on the product of the number of parameters  $p$  and time  $t$  (up to some error). Thus, we may interpret the product  $pt$  as representing the distance of a model along the training trajectory; larger  $pt$  implies more training progress. Increasing the number of parameters by a scale factor is equivalent to increasing the training time by the same scale factor and vice versa: scale is *equivalent* to time. Intuitively, this is because each parameter allows the function to learn at a fixed rate; thus, adding more parameters linearly increases the effective learning rate.

The result also reveals when such a scale-time equivalence *cannot* be made. The error bound implies that when training progress  $pt$  is fixed, as  $p$  grows, scale and time become increasingly equivalent (the bound approaches zero). The equivalence breaks down for small  $p$  since here, the randomly chosen subspace of the model may or may not align well with  $K$ ; as  $p$  grows, the amount of alignment becomes less stochastic. Moreover, as training progress  $pt$  grows, the bound grows exponentially because small perturbations to the model early in the training trajectory lead to exponentially larger changes later in the trajectory. Finally, we highlight that our result holds under standard neural network parameterizations in which the gradient of the model output with respect to each parameter does not scale with  $p$ ; in other parameterizations such as Neural Tangent Kernel (Jacot et al., 2018), we may expect a different form of scale-time equivalence.

We first validate our prediction in a simple linear model in which a varying fraction of the model parameters are controllable. See Appendix C for details. In Figure 1, we find lines of 1:1 proportionality between model scale and the number of iterations required to reach a fixed loss level, validating our theory.

### 3.2 EMPIRICAL VALIDATION IN NEURAL NETWORKS

We next turn to examine whether scale-time equivalence is present empirically in neural networks. We conduct experiments on MNIST (Deng, 2012), CIFAR-10 (Krizhevsky, 2009), and SVHN (Goodfellow et al., 2013) training a 7-layer convolutional neural network (CNN) and a 6-layer multilayer perceptron (MLP) with stochastic gradient descent (SGD). To assess scale-time equivalence, we measure the *minimum* amount of training time required to achieve non-zero generalization under various network widths and by varying the dataset size by subsampling. Scale-time equivalence predicts that wider networks will require less time to generalize in a systematically predictable way. See Appendix C for further experimental details.

As observed in Figure 2, in all settings, we see a clear tradeoff curve between scale and training time: increasing scale by a fixed factor is nearly equivalent to reducing training time by another fixed factor. Importantly, the scale here is set as the *effective* number of network parameters, defined as the maximum number of training points that can be fit by the network, not the absolute number

216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269

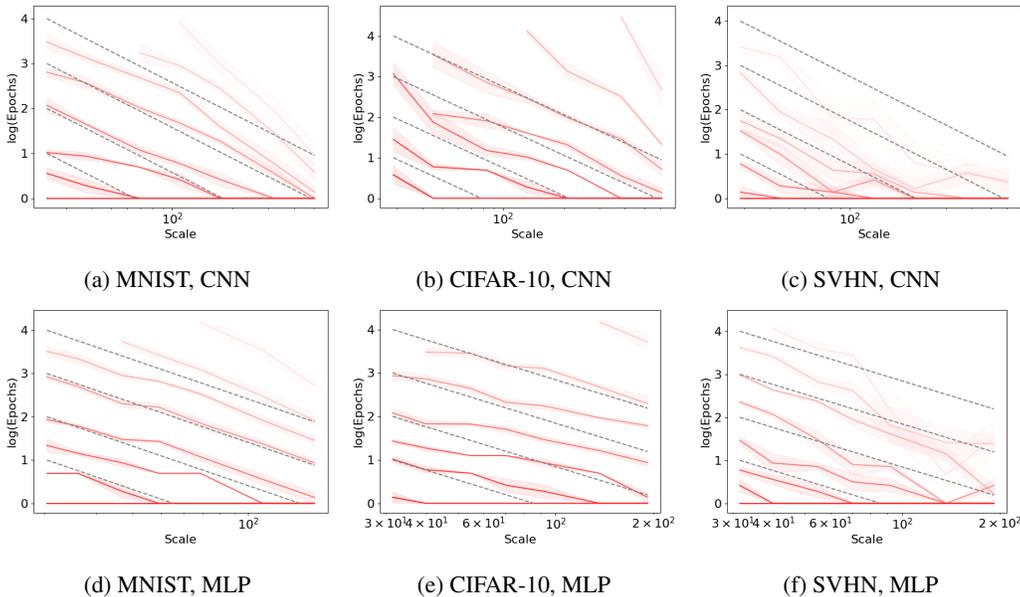


Figure 2: **Proportional trade-off between model scale and training time: testing the prediction on neural networks.** Red lines indicate tradeoff curves between number of training epochs and network scale for different datasets and architectures trained with SGD. Different curves indicate different amounts of training data; darker lines indicate more data. Curves are computed by, for each network scale, measuring the minimum amount of training time necessary to achieve non-zero generalization. Margins indicate standard errors over 5 trials. Grey curves are lines of 1:1 proportionality between scale and training epochs.

of parameters. We set the effective parameter count as the *cube root* of the number of parameters. Appendix B provides a heuristic argument for this scaling rate.

Under this choice of scale, we find a systematic and predictable relationship between scale and training time, demonstrating that scale-time equivalence can be empirically observed. We also note that this phenomenon has been observed in prior literature (although not quantitatively characterized); for instance, Nakkiran et al. (2019) find that patterns of double descent are similar with respect to training epochs and network size. We emphasize that these results are limited to gradient descent (the setting of our theory). With Adam optimization (Kingma & Ba, 2015), the number of epochs for generalization first decreases with scale, then increases (see Appendix D Figure 7). We hypothesize that since Adam has an adaptive learning rate, it is using a smaller effective learning rate for very large networks. As learning rate shrinks, more epochs are needed to generalize, leading to the observed results.

#### 4 PREDICTING OPTIMAL NETWORK SCALE AND TRAINING TIME

Scale-time equivalence suggests it should be possible to predict the performance of large models by training small models for many epochs and vice versa. This allows us to predict the optimal network scale and training time for a given dataset and base architecture.

On benchmark datasets and architectures, we conduct two experiments: 1) predict performance under varying model scales from a small network trained for long training times, 2) predict performance over long training times by using larger networks trained for just 1 epoch. See Appendix C for experimental details and Figures 10 and 11 for full results.

Figure 3 illustrates that scale-time equivalence can indeed be used to extrapolate performance on large scales and training times. Predictions of test and train performance under large model scales are particularly close to the true performance; notably, we can closely predict the scale at which generalization starts to occur. However, there is a small discrepancy between predicted and actual

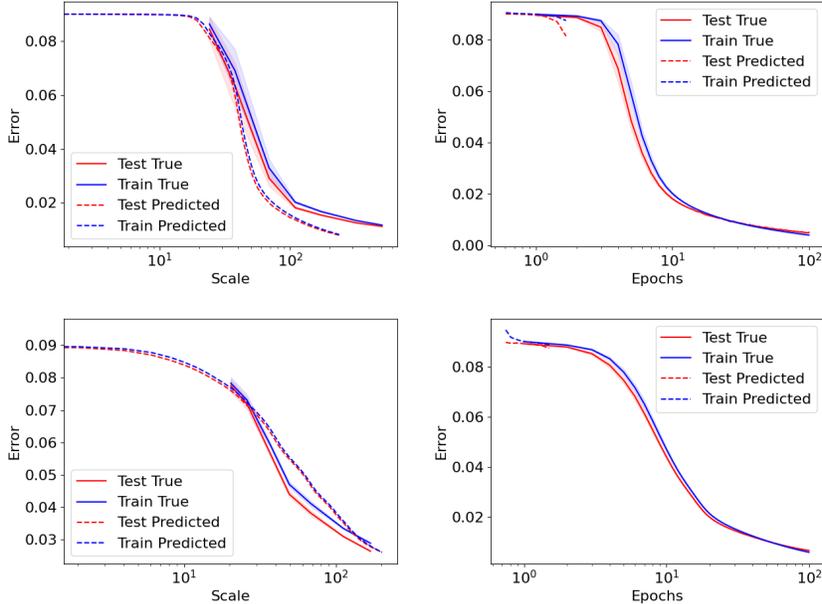


Figure 3: **By scale-time equivalence, small models trained for long times predict performance of large models trained for small times and vice versa: test of prediction.** Predicted and true test and train error of a CNN (top row) and MLP (bottom row) trained on MNIST. Column 1: predicting the performance of larger models over a few epochs by training smaller models for up to 100 epochs. Column 2: predicting performance of smaller models over many epochs by training larger models for 1 epoch. We use scale-time equivalence to predict the equivalent scale or number of epochs for each prediction. Margins indicate standard errors over 5 trials.

performance; we believe this can be corrected with dataset and model-specific tuning of the scale-time trade-off curve. Nevertheless, our findings reveal that scale-time equivalence can be used to predict optimal network scale and training time.

## 5 A UNIFIED VIEW OF DOUBLE DESCENT W.R.T. TRAINING TIME, MODEL SCALE AND TRAINING SET SIZE

We next leverage scale-time equivalence to obtain a more-unified understanding of the phenomenon of double descent, with respect to training time, parameter count and training set size.

### 5.1 ERROR SCALING OVER TIME IN A LINEAR MODEL

Following the approach of Pezeshki et al. (2022); Heckel & Yilmaz (2021); Stephenson & Lee (2021); Schaeffer et al. (2023), we first present a simple linear model that explains double descent with respect to time.

Consider a linear student-teacher setting in which training set outputs  $Y \in \mathbb{R}^n$  are constructed as:

$$Y = Xw + \varepsilon \tag{5}$$

where training data  $X \in \mathbb{R}^{n \times m}$ , noise  $\varepsilon \in \mathbb{R}^n$ ,  $w \in \mathbb{R}^m$  is the unknown true model,  $n$  is the number of training points and  $m$  is the model dimensionality. We assume  $w$  is drawn independently from  $\varepsilon$ . Then, the parameters  $\theta_t$  of a linear model learned after  $t$  time of gradient flow on mean squared error (with learning rate  $\eta$ ) can be expressed as:

$$\theta_t = X^\dagger (I - e^{-\eta X X^T t}) Y \tag{6}$$

The resulting prediction error  $x^T \theta_t - x^T w$  on a test point  $x$  can be expressed as:

$$x^T [X^\dagger (I - e^{-\eta X X^T t}) X - I] w + x^T X^\dagger (I - e^{-\eta X X^T t}) \varepsilon \tag{7}$$

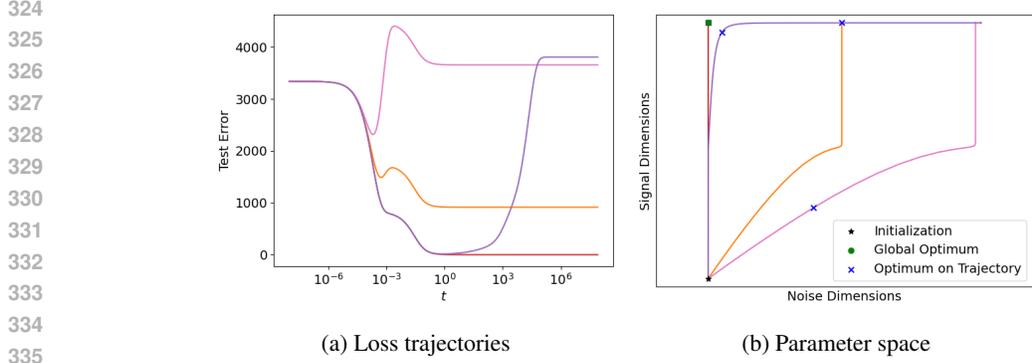


Figure 4: **Predicted variations in double-descent behavior, depending on training noise profile.** Schematic loss trajectories (a) and corresponding parameter space trajectories (b) of linear regression under various noise settings (different color curves). Depending on the noise profile, parameters may experience a temporary increase in error resembling an interpolation threshold.

We then use the singular value decomposition  $U\Sigma V^T$  of  $X$  to express the prediction error:

$$\begin{aligned}
 x^T V[\Sigma^\dagger(I - e^{-\eta\Sigma\Sigma^T t})\Sigma - I]V^T w + x^T V\Sigma^\dagger(I - e^{-\eta\Sigma\Sigma^T t})V^T \varepsilon \\
 = \sum_{i=1}^m -(x^T V)_i (V^T w)_i e^{-\eta\sigma_i^2 t} + (x^T V)_i (V^T \varepsilon)_i \frac{1 - e^{-\eta\sigma_i^2 t}}{\sigma_i} \quad (8)
 \end{aligned}$$

where  $\sigma_i$  are the singular values of  $X$ , and we denote  $\sigma_i = 0$  for  $i > n$  when  $n < m$  (in this case,  $\frac{1 - e^{-\eta\sigma_i^2 t}}{\sigma_i}$  denotes 0). Using the independence of  $w$  and  $\varepsilon$ , we finally may simply express the expected *squared* prediction error as:

$$\mathbb{E}[(x^T w - x^T \theta_t)^2] = \mathbb{E}[(\sum_{i=1}^m S_i e^{-\eta\sigma_i^2 t})^2] + \mathbb{E}[(\sum_{i=1}^m N_i \frac{1 - e^{-\eta\sigma_i^2 t}}{\sigma_i})^2] \quad (9)$$

where  $S_i = -(x^T V)_i (V^T w)_i$ ,  $N_i = (x^T V)_i (V^T \varepsilon)_i$ . The first, signal term captures how well  $w$  can be learned in the absence of noise. In the underparameterized regime ( $n > m$ ), this term approaches 0 as  $t \rightarrow \infty$ : without noise, the model can be learned perfectly. Observe that in general, the prediction error is *not* predicted to decay directly as a power law with  $t$ : instead, it decays or grows following a combination of exponential curves (though a combination of exponential decays at different rates can mimic a power law (Reed & Hughes, 2002)).

The second, noise term initially starts at 0 and grows over time. Notably, the size of the noise term is largest near the interpolation threshold (when  $m \approx n$ ) since the smallest singular values  $\sigma_i$  will take small, non-zero values. As  $n$  or  $m$  grows larger than the other (distance from the interpolation threshold increases), the size of the noise term decreases. However, the noise term's magnitude monotonically increases with  $t$ .

For any given singular component of  $X$ , if the noise components are large relative to signal components (i.e.  $|N_i| > |S_i|$ ), we expect that error will increase at the corresponding timescale, around  $t = \frac{1}{\eta\sigma_i^2}$  and vice versa. Thus, if the small singular value components are noisy, error will increase later during training, while if large singular value components are noisy, error will increase early during training. Double-descent occurs when noisy components are acquired first (increase in error) followed by signal components (decrease in error). Figure 4(a) illustrates how different acquisition rates of noise vs. signal components may yield different loss trajectories, with some corresponding to double descent. We may also view these trends geometrically in the parameter space; Figure 4(b) shows that if signal and noise correspond to orthogonal parameter dimensions, then double descent (orange curve) corresponds to a setting in which noise dimensions are learned rapidly before signal dimensions. The optimal training time depends on which point in the parameter trajectory is closest to the true optimum and may occur either at the end of training or at an intermediate point.

378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431

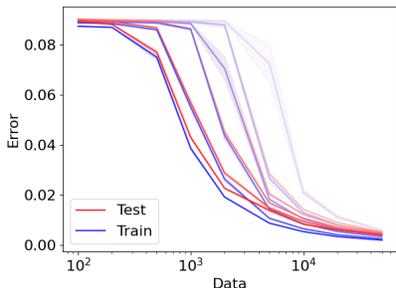


Figure 5: **Larger models require less data to interpolate: test of prediction.** Test and train mean squared error of CNN models trained on MNIST under varying levels of data. Different curves indicate different model scales; darker colors indicate larger models. Margins indicate standard errors over 5 trials.

## 5.2 EARLY NOISE ACQUISITION CAN EXPLAIN PARAMETER-WISE DOUBLE DESCENT

Next, we hypothesize that due to scale-time equivalence, parameter-wise double descent occurs *due to the same mechanism* as time-wise double descent. Namely, in settings where noise is acquired before signal, smaller scale models (which require effectively larger training time by scale-time equivalence) fail to acquire signal.

More precisely, by combining Equation 9 and scale-time equivalence, we propose the following scaling law in terms of the number of parameters  $p$  and time  $t$  by substituting  $t$  with  $pt$ :

$$error^2 = \mathbb{E}[(\sum_{i=1}^{\infty} S_i e^{-\eta \sigma_i^2 pt})^2] + \mathbb{E}[(\sum_{i=1}^{\infty} N_i \frac{1 - e^{-\eta \sigma_i^2 pt}}{\sigma_i})^2] \quad (10)$$

where  $\sigma_i$  depends on the number of training points  $n$  and  $S_i$  and  $N_i$  are random variables determining the strength of signal and noise respectively. This scaling law simultaneously explains double descent in terms of both  $p$  and  $t$ . Moreover, it can explain double descent in terms of data volume  $n$  as well: if  $n$  is set such that there are several small non-zero values of  $\sigma_i$ , the noise term becomes amplified. This explanation for double descent in  $n$  follows prior literature (Advani & Saxe, 2017).

The explanation for parameter-wise double descent differs from conventional wisdom in which double descent occurs due to the same reason as double descent in  $n$ : namely, when the number of model parameters is close to  $n$ , the model is highly sensitive to noisy directions in the training data (corresponding to small  $\sigma_i$  in a linear model), thus severely overfitting. How can we distinguish this hypothesis from ours? We propose three tests to separate the two hypotheses.

## 5.3 LESS DATA REQUIRED FOR GENERALIZATION WITH MODEL SCALE

The interpolation threshold can be defined as the point when the number of data points equals the effective complexity of a model (Nakkiran et al., 2019). Thus, for any model, the location of the interpolation threshold with respect to data volume is the model’s effective capacity: higher capacity models have a rightward-shifted interpolation threshold with respect to data volume. Conventional double descent theory argues that this point corresponds to a sharp decrease in the test set error as the amount of data grows. Starting from zero data, we would therefore expect that larger models (which have a larger effective capacity), would experience a sharp decrease in test set error at *higher data volumes*: larger models require *more* data to generalize. By contrast, under our explanation, as model size grows, the amount of data needed to generalize *decreases*: since model size corresponds to training time, generalization occurs more easily with larger models (equivalently, more training time).

To test this, we conduct experiments on benchmark datasets and architectures. Figure 5 reveals that larger models indeed require less data to generalize, thus supporting our hypothesis. (See Appendix C for experimental details and Figure 8 for full results.) Indeed, the *training set errors decrease* with data volume which is at odds with conventional double descent theory in which more

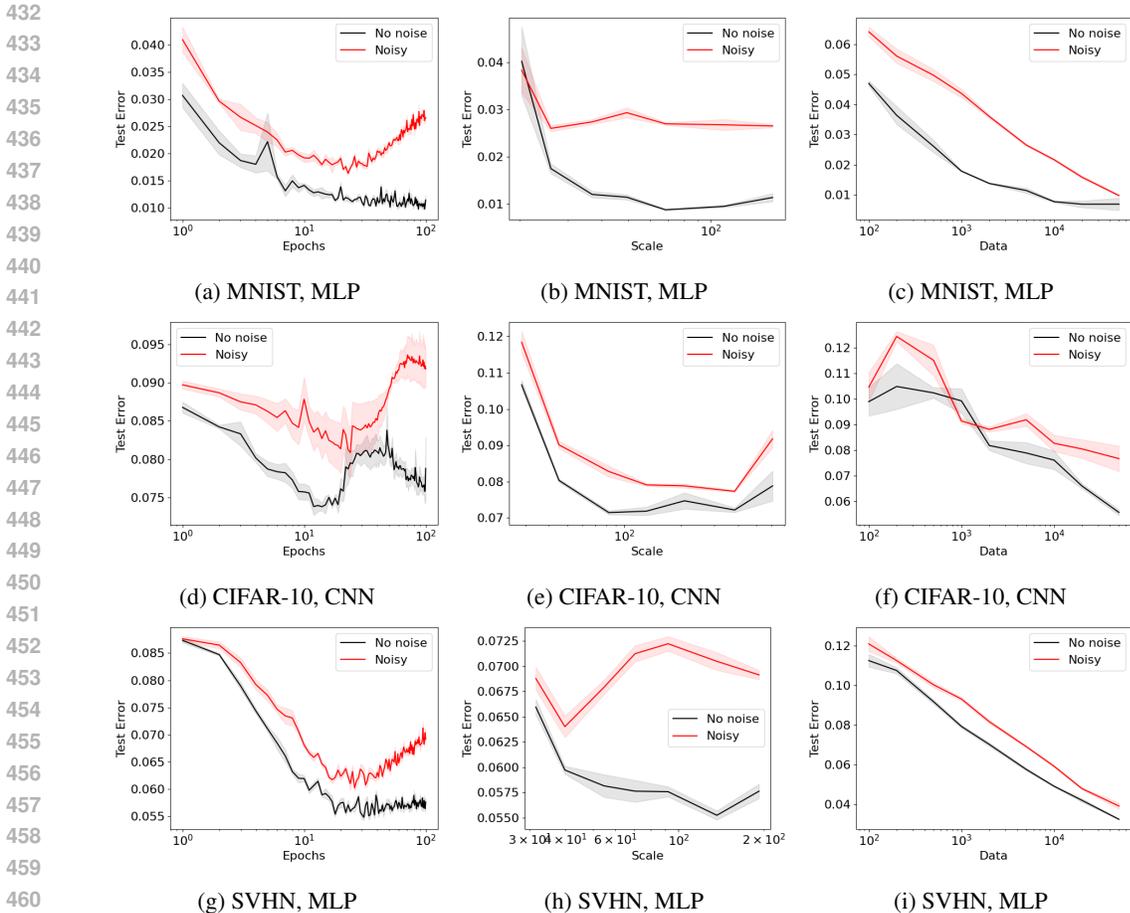


Figure 6: **Noise induces a persistent or growing error with training time and model scale, but not with dataset size.** Test mean squared error vs. number of epochs, model scale and training data under noisy and noise-free labels. Each row indicates a different combination of dataset and architecture. Margins indicate standard errors over 5 trials.

data is always harder to interpolate. Fundamentally, this is because conventional double descent theory typically assumes complete training convergence, which does not explain phenomena under the practically relevant setting of a fixed training budget. In summary, the increased ease of generalization with model scale does not neatly fit into standard theories of double descent, but readily fits in our explanation.

#### 5.4 PERSISTENT VS LOCAL EFFECTS OF NOISE ON ERROR CURVES

Another key distinction is how the two explanations behave under varying levels of noise. Under the conventional explanation, as the noise ( $|N_i|$ ) increases, the error increases mostly *locally around*  $p \approx n$ , near the interpolation threshold. This is because in Equation 9, the noise coefficients  $N_i$  multiply terms that are largest near the interpolation threshold and decrease as either data volume or model scale grow larger than the other (Schaeffer et al., 2023). In contrast, under our hypothesis, as the noise increases, the error monotonically grows with both model scale and time (see Equation 10). In other words, when the noise level changes, our hypothesis predicts a global performance change with respect to model scale while the conventional explanation predicts a primarily local change.

We conduct experiments on benchmark datasets and architectures to validate this. We add label noise to the training set and evaluate performance as a function of epochs, model scale and data size. Appendix C includes experimental details. Noise leads to an increase in error in all cases, Figure 6 (See Appendix D for additional plots), with a persistent or growing error with training time (epochs)

486 and model scale. However, it leads to only localized increases in error in terms of the dataset size:  
 487 in the case of MNIST and SVHN, for instance, there is nearly no change in error at the highest data  
 488 volume.

### 490 5.5 SHAPE OF LOSS CURVE: UPTURN WITH MODEL SIZE

491  
 492 A final key distinction between the two explanations is their prediction of the shape of the loss curve.  
 493 Under the conventional explanation of double descent, past the interpolation threshold, increasing  
 494 model size always improves performance. However, under our hypothesis, increasing model size  
 495 may in certain cases *worsen performance*, if signal features are learned before noisy features. In  
 496 Figure 6, we find that CIFAR-10 and SVHN indeed exhibit a U-shaped error curve with respect to  
 497 model scale such that larger models do worse. This departure from a conventional double descent  
 498 curve is consistent with our model.

## 500 6 DISCUSSION

501  
 502 In this work, we demonstrate that model scale and training time can be traded off with each other.  
 503 This enables us to re-frame parameter-wise double descent as occurring due to the same mechanism  
 504 as epoch-wise double descent, which occurs due to the early acquisition of noise features during  
 505 training. This framing of parameter-wise double descent has a number of surprising implications  
 506 unexplained by standard explanations of double descent: 1) generalization requires less data with a  
 507 larger model, 2) label noise significantly increases test error even for highly overparameterized mod-  
 508 els, 3) increasing model scale for overparameterized models need not always improve performance.

509 How can we reconcile our findings with conventional theory and experimental findings on double de-  
 510 scent and scaling laws? Note that in past work, the presence of double descent with respect to model  
 511 scale is often dependent on the choice of dataset, model, and whether label noise is added (Nakki-  
 512 ran et al., 2021). While past explanations of parameter-wise double descent necessitate that it must  
 513 occur, our explanation is more flexible: double descent need not occur if noisy features are acquired  
 514 later in training than signal features. Indeed, our theory explains why local increases in test error in  
 515 terms of model scale are often relatively modest in contrast to the sharp spike predicted by conven-  
 516 tional theory (Belkin et al., 2019). Thus, our explanation is more consistent with the variability of  
 517 double descent observed in the literature.

518 Regarding scaling laws, our predicted scaling law in Equation 10 is more flexible than the power law  
 519 scalings predicted in prior literature; indeed, with the proper settings of signal and noise parameters  
 520  $S_i$  and  $N_i$ , we may recover power-law scalings of error with respect to time, model scale and data  
 521 volume. However, our approach retains the flexibility to explain error scalings in settings where a  
 522 power law *does not* explain empirical error trends, as we see in our experimental results.

523 Given the parametric flexibility of our scaling law, how can we use it to predict performance as model  
 524 scale or training time increases? We demonstrate that by scale-time equivalence, performance under  
 525 varying training times can be used to predict performance under varying model scales and vice versa.  
 526 Our approach eliminates the need for strong parametric assumptions in the form of scaling law to  
 527 make extrapolation predictions. This is particularly useful in cases where error *increases* with scale  
 528 for overparameterized models since our approach can be used to predict an optimal model size.  
 529 On the other hand, we require empirically evaluating the performance of models (under either a  
 530 smaller scale or lower training time). We believe our approach can be valuable to practitioners who  
 531 have the flexibility to run some limited empirical small-scale experiments before full-scale training.  
 532 Finally, our results suggest that smaller models trained for a longer time may behave as well as larger  
 533 models, which we empirically observe on vision benchmarks. This is particularly important in the  
 age of LLMs, where very large models are trained for a small number of epochs (often just one).

534 We also highlight some important limitations of our work. Our experiments are all conducted on  
 535 standard vision benchmarks; we believe testing our theory on the language domain is a critical future  
 536 direction. Another limitation is that we do not yet have a fundamental understanding of what sets  
 537 the appropriate model scale of a neural network. Experimentally, we found that the cube root of  
 538 the number of model parameters is appropriate, but without a strong theoretical basis; this deserves  
 539 further study. Overall, we believe our contributions not only shed light on neural scaling laws, but  
 also present exciting directions for future work.

## REFERENCES

- 540  
541  
542 Ben Adlam and Jeffrey Pennington. Understanding double descent requires a fine-grained bias-  
543 variance decomposition. In *NeurIPS*, 2020. <https://arxiv.org/abs/2011.03321>.
- 544 Ben Adlam, Jake Levinson, and Jeffrey Pennington. A random matrix perspective on mixtures of  
545 nonlinearities for deep learning. *AISTATS*, 2022.
- 546  
547 Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural  
548 networks. *arXiv preprint*, 2017.
- 549 Ibrahim Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in  
550 language and vision. In *NeurIPS*, 2022.
- 551 Alnur Ali, J Zico Kolter, and Ryan J Tibshirani. A continuous-time view of early stopping for least  
552 squares regression. In *AISTATS*, 2019.
- 553  
554 Alexander Atanasov, Blake Bordelon, Sabarish Sainathan, and Cengiz Pehlevan. The onset of  
555 variance-limited behavior for networks in the lazy and rich regimes. *ICLR*, 2024.
- 556  
557 Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural  
558 scaling laws. *arXiv preprint*, 2021.
- 559 Yamini Bansal, Behrooz Ghorbani, Ankush Garg, Biao Zhang, Colin Cherry, Behnam Neyshabur,  
560 and Orhan Firat. Data scaling laws in nmt: The effect of noise and architecture. In *ICML*, pp.  
561 1466–1482. PMLR, 2022.
- 562 Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-  
563 learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy*  
564 *of Sciences*, 116(32):15849–15854, 2019.
- 565  
566 Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling  
567 laws. *ICML*, 2024.
- 568 Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoff-  
569 mann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling  
570 laws for routed language models. In *International Conference on Machine Learning*, pp. 4057–  
571 4086. PMLR, 2022.
- 572  
573 Stéphane D’Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double trouble in double  
574 descent: Bias and variance(s) in the lazy regime. In *ICML*, 2020.
- 575 Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE*  
576 *Signal Processing Magazine*, 29(6):141–142, 2012.
- 577  
578 Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number  
579 recognition from street view imagery using deep convolutional neural networks. *arXiv preprint*,  
580 2013.
- 581 Peter Hall and Ker-Chau Li. On almost Linearity of Low Dimensional Projections from High Di-  
582 mensional Data. *The Annals of Statistics*, 21(2):867 – 889, 1993. doi: 10.1214/aos/1176349155.  
583 URL <https://doi.org/10.1214/aos/1176349155>.
- 584 Reinhard Heckel and Fatih Furkan Yilmaz. Early stopping in deep networks: Double descent  
585 and how to eliminate it. In *ICLR*, 2021. URL <https://openreview.net/forum?id=t1V90jvZbw>.
- 586  
587  
588 Marcus Hutter. Learning curve theory. *arXiv preprint*, 2021.
- 589 Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and gener-  
590 alization in neural networks. *NeurIPS*, 31, 2018.
- 591  
592 Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child,  
593 Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language  
models. *arXiv preprint*, 2020.

- 594 Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.  
595
- 596 Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.  
597
- 598 Licong Lin, Jingfeng Wu, Sham M Kakade, Peter L Bartlett, and Jason D Lee. Scaling laws in linear  
599 regression: Compute, parameters, and data. *NeurIPS*, 2024.
- 600 Rafid Mahmood, James Lucas, David Acuna, Daiqing Li, Jonah Philion, Jose M Alvarez, Zhiding  
601 Yu, Sanja Fidler, and Marc T Law. How much more data do i need? estimating requirements for  
602 downstream tasks. In *CVPR*, pp. 275–284, 2022.
- 603 Alexander Maloney, Daniel A Roberts, and James Sully. A solvable model of neural scaling laws.  
604 *arXiv preprint*, 2022.  
605
- 606 Song Mei and Andrea Montanari. The generalization error of random features regression: Precise  
607 asymptotics and double descent curve. *arXiv preprint*, 2019.
- 608 Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred  
609 Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity.  
610 *NeurIPS*, 2019.  
611
- 612 Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep  
613 double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics:  
614 Theory and Experiment*, 2021(12):124003, 2021.
- 615 Elliot Paquette, Courtney Paquette, Lechao Xiao, and Jeffrey Pennington. 4+ 3 phases of compute-  
616 optimal neural scaling laws. *NeurIPS*, 2024.  
617
- 618 Mohammad Pezeshki, Amartya Mitra, Yoshua Bengio, and Guillaume Lajoie. Multi-scale fea-  
619 ture learning dynamics: Insights for double descent. In Kamalika Chaudhuri, Stefanie Jegelka,  
620 Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th Inter-  
621 national Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning  
622 Research*, pp. 17669–17690. PMLR, 17–23 Jul 2022. URL [https://proceedings.mlr.  
623 press/v162/pezeshki22a.html](https://proceedings.mlr.press/v162/pezeshki22a.html).
- 624 William J. Reed and Barry D. Hughes. Power-law distributions from exponential processes : An  
625 explanation for the occurrence of long-tailed distributions in biology and elsewhere. *Physical  
626 Review E*, 66:4, 2002.
- 627 Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction  
628 of the generalization error across scales. In *ICLR*, 2020.  
629
- 630 Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathy, Kateryna Pistunova, Ja-  
631 son W Rocks, Ila Rani Fiete, and Oluwasanmi Koyejo. Double descent demystified: Identifying,  
632 interpreting & ablating the sources of a deep learning puzzle. *arXiv preprint*, 2023.
- 633 Utkarsh Sharma and Jared Kaplan. Scaling laws from the data manifold dimension. *JMLR*, 23:9–1,  
634 2022.  
635
- 636 James B Simon, Dhruva Karkada, Nikhil Ghosh, and Mikhail Belkin. More is better in modern  
637 machine learning: when infinite overparameterization is optimal and overfitting is obligatory.  
638 *ICLR*, 2024.
- 639 Cory Stephenson and Tyler Lee. When and how epochwise double descent happens. *arXiv preprint*,  
640 2021.  
641
- 642 Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan  
643 Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from  
644 pre-training and fine-tuning transformers. In *ICLR*, 2022.  
645  
646  
647

## 648 A PROOF OF THEOREM 1

649 First, observe that gradient descent applied to  $\theta$  corresponds to:

$$650 \dot{\theta} = -\eta R^T K^T \nabla L(\alpha) \quad (11)$$

651 This yields a change in  $\alpha$  of:

$$652 \dot{\alpha} = -\eta K R R^T K^T \nabla L(\alpha) \quad (12)$$

653 **Properties of  $K R R^T K^T$**  First, we investigate the properties of the  $r \times r$  random matrix  
654  $K R R^T K^T$ . Since the elements of  $R$  are drawn iid from a unit Gaussian,  $\mathbb{E}[R R^T] = pI$ . Thus,

$$655 \mathbb{E}[K R R^T K^T] = p K K^T \quad (13)$$

656 Next, consider the element at the  $i$ th row and  $j$ th column of  $K R R^T K^T$ . This may be expressed as:

$$657 K_{i,:} R R^T K_{:,j}^T = \sum_l K_{i,:} R_{:,l} R_{l,:}^T K_{:,j}^T = \sum_l (K_{i,:} R_{:,l}) (K_{j,:} R_{:,l}) \quad (14)$$

658 where we express different columns of  $R$  as  $R_{:,l}$ . Note that each term in the summand is independent  
659 from one another since  $R$  has iid elements. Observe that  $K_{i,:} R_{:,l} = \sum_k K_{i,k} R_{k,l}$  and each term  
660  $K_{i,k} R_{k,l}$  is an independent mean zero Gaussian with variance  $K_{i,k}^2$ . Thus,  $K_{i,:} R_{:,l}$  is a mean zero  
661 Gaussian with variance  $K_{i,:} K_{:,i}^T$ . Moreover, since the expectation of  $K_{i,:} R R^T K_{:,j}^T$  is  $p K_{i,:} K_{:,j}^T$ ,  
662  $K_{i,:} R_{:,l}$  and  $K_{j,:} R_{:,l}$  are jointly Gaussian with covariance  $K_{i,:} K_{:,j}^T$ .

663 Now, consider the expectation of  $(K_{i,:} R R^T K_{:,j}^T)^2$ :

$$664 \mathbb{E}[(K_{i,:} R R^T K_{:,j}^T)^2] = \mathbb{E}\left[\sum_l (K_{i,:} R_{:,l}) (K_{j,:} R_{:,l}) \sum_{l'} (K_{i,:} R_{:,l'}) (K_{j,:} R_{:,l'})\right] \quad (15)$$

665 since  $(K_{i,:} R_{:,l}) (K_{j,:} R_{:,l})$  and  $(K_{i,:} R_{:,l'}) (K_{j,:} R_{:,l'})$  are independent for  $l \neq l'$ :

$$666 \mathbb{E}[(K_{i,:} R R^T K_{:,j}^T)^2] = \mathbb{E}\left[\sum_l (K_{i,:} R_{:,l})^2 (K_{j,:} R_{:,l})^2\right] + p(p-1) \mathbb{E}[(K_{i,:} R_{:,l}) (K_{j,:} R_{:,l})]^2 \quad (16)$$

667  $\mathbb{E}[(K_{i,:} R_{:,l}) (K_{j,:} R_{:,l})]$  is simply  $K_{i,:} K_{:,j}^T$ . Since  $K_{i,:} R_{:,l}$  and  $K_{j,:} R_{:,l}$  are jointly Gaussian, we may  
668 reparameterize them as:

$$669 K_{i,:} R_{:,l} = a z_i \quad (17)$$

$$670 K_{j,:} R_{:,l} = b z_i + c z_j \quad (18)$$

671 where  $z_i$  and  $z_j$  are independent unit Gaussians,  $a = \sqrt{K_{i,:} K_{:,i}^T}$ ,  $b = \frac{K_{i,:} K_{:,j}^T}{a}$ ,  $c = \sqrt{K_{j,:} K_{:,j}^T - b^2}$ .

672 Then,  $(K_{i,:} R_{:,l})^2 (K_{j,:} R_{:,l})^2$  may be expressed as:

$$673 (K_{i,:} R_{:,l})^2 (K_{j,:} R_{:,l})^2 = (a z_i)^2 (b z_i + c z_j)^2 = a^2 z_i^2 (b z_i + c z_j)^2 = a^2 b^2 z_i^4 + 2 a^2 b c z_i^3 z_j + a^2 c^2 z_i^2 z_j^2 \quad (19)$$

674 Taking the expectation:

$$675 \mathbb{E}[a^2 b^2 z_i^4 + 2 a^2 b c z_i^3 z_j + a^2 c^2 z_i^2 z_j^2] = a^2 b^2 \mathbb{E}[z_i^4] + 2 a^2 b c \mathbb{E}[z_i^3 z_j] + a^2 c^2 \mathbb{E}[z_i^2 z_j^2] \quad (20)$$

676 Using the moments of unit Gaussians, we have  $\mathbb{E}[z_i^4] = 3$ ,  $\mathbb{E}[z_i^3 z_j] = 0$ ,  $\mathbb{E}[z_i^2 z_j^2] = 1$ . Thus, we  
677 have

$$678 \mathbb{E}[(K_{i,:} R_{:,l})^2 (K_{j,:} R_{:,l})^2] = 3 a^2 b^2 + a^2 c^2 = a^2 (3 b^2 + c^2) \\ 679 = a^2 (K_{j,:} K_{:,j}^T + 2 \frac{(K_{i,:} K_{:,j}^T)^2}{a^2}) = K_{i,:} K_{:,i}^T K_{j,:} K_{:,j}^T + 2 (K_{i,:} K_{:,j}^T)^2 \quad (21)$$

680 Finally, we may express the expectation of  $(K_{i,:} R R^T K_{:,j}^T)^2$  as:

$$681 \mathbb{E}[(K_{i,:} R R^T K_{:,j}^T)^2] = p (K_{i,:} K_{:,i}^T K_{j,:} K_{:,j}^T + 2 (K_{i,:} K_{:,j}^T)^2) + p(p-1) (K_{i,:} K_{:,j}^T)^2 \quad (22)$$

**Decomposition into expectation and noise** We now decompose  $KRR^TK^T$  into an expectation term and a mean-zero noise term:

$$KRR^TK^T = pKK^T + N \quad (23)$$

where  $N \in \mathbb{R}^{r \times r}$  is a mean-zero matrix. Note that each element of  $N$  has variance:

$$\mathbb{E}[N_{i,j}^2] = \mathbb{E}[(K_{i,:}RR^TK_{:,j}^T)^2] - p^2(K_{i,:}K_{:,j}^T)^2 = p(K_{i,:}K_{:,i}^TK_{j,:}K_{:,j}^T + (K_{i,:}K_{:,j}^T)^2) \quad (24)$$

Thus, the squared Frobenius norm of  $N$  has expectation:

$$\mathbb{E}[\|N\|_F^2] = \sum_{i,j} \mathbb{E}[N_{i,j}^2] = p \sum_{i,j} K_{i,:}K_{:,i}^TK_{j,:}K_{:,j}^T + (K_{i,:}K_{:,j}^T)^2 = p(\|K\|_F^4 + \|KK^T\|_F^2) \quad (25)$$

We express the dynamics of  $\alpha$  as:

$$\dot{\alpha} = -\eta pKK^T\nabla L(\alpha) - \eta N\nabla L(\alpha) \quad (26)$$

Now suppose we have two copies of  $\alpha$ : one copy ( $\alpha^{(1)}$ ) with noise-free dynamics, and a second copy with noise:

$$\dot{\alpha}^{(1)} = -\eta pKK^T\nabla L(\alpha^{(1)}) \quad (27)$$

$$\dot{\alpha}^{(2)} = -\eta pKK^T\nabla L(\alpha^{(2)}) - \eta N\nabla L(\alpha^{(2)}) \quad (28)$$

We define the discrepancy between them as  $\delta = \alpha^{(2)} - \alpha^{(1)}$ , which has dynamics:

$$\dot{\delta} = -\eta pKK^T[\nabla L(\alpha^{(2)}) - \nabla L(\alpha^{(1)})] - \eta N\nabla L(\alpha^{(2)}) \quad (29)$$

Consider the rate of change of the  $\ell_2$  norm of  $\delta$ :

$$\frac{d}{dt}\|\delta\| \leq \|\dot{\delta}\| \leq \eta p\|KK^T[\nabla L(\alpha^{(2)}) - \nabla L(\alpha^{(1)})]\| + \eta\|N\nabla L(\alpha^{(2)})\| \quad (30)$$

Using the Lipschitz bounds on  $\nabla L$  and  $L$ :

$$\frac{d}{dt}\|\delta\| \leq \eta p h\|KK^T\|\|\delta\| + \eta\|N\|l \quad (31)$$

where the matrix norms in the expression denote  $\ell_2$  operator norm.

Now, returning to  $\|N\|_F^2$ , note that by Markov's inequality, with probability  $1 - \epsilon$ :

$$\|N\|_F^2 \leq \frac{p}{\epsilon}(\|K\|_F^4 + \|KK^T\|_F^2) \quad (32)$$

Using the fact that  $\|N\| \leq \|N\|_F$ , with probability  $1 - \epsilon$ :

$$\frac{d}{dt}\|\delta\| \leq \eta p h\|KK^T\|\|\delta\| + \eta \frac{\sqrt{p}}{\sqrt{\epsilon}} \sqrt{\|K\|_F^4 + \|KK^T\|_F^2} l \quad (33)$$

This is a differential inequality in  $\|\delta\|$ . Observe that  $\|\delta\|$  takes its maximum possible trajectory at equality. Assuming  $\|\delta\| = 0$  at time  $t = 0$ , this differential inequality may be solved by simply solving the equality case and setting the solution as the upper bound on  $\delta_t$  (with subscript denoting time  $t$ ):

$$\|\delta_t\| \leq \frac{\eta \frac{\sqrt{p}}{\sqrt{\epsilon}} \sqrt{\|K\|_F^4 + \|KK^T\|_F^2} l}{\eta p h\|KK^T\|} (e^{\eta p h\|KK^T\|t} - 1) = \frac{\sqrt{\|K\|_F^4 + \|KK^T\|_F^2} l}{\sqrt{p\epsilon} h\|KK^T\|} (e^{\eta p h\|KK^T\|t} - 1) \quad (34)$$

**Solving the noise-free dynamics** Now, we consider the noise-free dynamics:

$$\dot{\alpha}^{(1)} = -\eta pKK^T\nabla L(\alpha^{(1)}) \quad (35)$$

Note that  $A_{pt}$  solves the dynamical equation for  $\alpha^{(1)}$  by the chain rule:

$$\frac{d}{dt}A_{pt} = p(-\eta KK^T\nabla L(A_{pt})) = -\eta pKK^T\nabla L(A_{pt}) \quad (36)$$

Thus, we may simply express  $\alpha_t^{(1)}$  as  $A_{pt}$ .

**Final result** Combining and reexpressing our previous results, we may write that with probability  $1 - \epsilon$ :

$$\|\alpha_t^{(2)} - A_{pt}\| = \|\alpha_t^{(2)} - \alpha_t^{(1)}\| = \|\delta_t\| \leq \frac{l\sqrt{\|K\|_F^4 + \|KK^T\|_F^2}}{h\sqrt{p\epsilon}\|KK^T\|} (e^{\eta p t h \|KK^T\|} - 1) \quad (37)$$

Thus, the true dynamics deviate from  $A_{pt}$  by a bounded amount.

## B MEASURING EFFECTIVE PARAMETER COUNT

In this section, we aim to provide a justification for why the *cube root* of the absolute number of parameters of a network is a good proxy for the *effective* number of parameters. Finding the effective number of parameters requires defining the parameter count at which the interpolation threshold occurs; this number is the effective parameter count. Unfortunately, in most models, training data is not *perfectly* interpolated by the model; thus defining the interpolation threshold location exactly is nontrivial. Instead, we identify a property of the interpolation threshold in the training error of linear models, then identify at which parameter count this property also holds in nonlinear models.

**Training error in linear models** Suppose we are provided a training set  $X \in \mathbb{R}^{n \times p}$  and corresponding training labels  $Y \in \mathbb{R}^n$  where  $n$  is the number of data points. Assume that  $X$  has elements drawn uniformly from a unit Gaussian. We aim to find a parameter  $\theta \in \mathbb{R}^p$  such that:

$$Y \approx X\theta \quad (38)$$

The minimum norm solution minimizing the mean squared error is:

$$\theta = X^\dagger Y \quad (39)$$

where  $\dagger$  denotes pseudoinverse. Denoting the predicted training labels as  $\hat{Y}$ , the mean squared error is then:

$$\frac{1}{n} \|Y - \hat{Y}\|_2^2 = \frac{1}{n} \|Y - XX^\dagger Y\|_2^2 = \frac{1}{n} Y^T (I - XX^\dagger) Y \quad (40)$$

Finally, assuming that  $Y_i^2 = 1$  for all  $i$ , we may write the mean training error as:

$$\frac{1}{n} \|Y - \hat{Y}\|_2^2 = \max(1 - \frac{p}{n}, 0) \quad (41)$$

Thus, for a linear model, the training error decreases linearly at rate  $-\frac{1}{n}$  with respect to  $p$  before the interpolation threshold ( $p = n$ ), and then is zero after the interpolation threshold. We extract a key property around the interpolation threshold from the linear model: the training error is  $\frac{1}{n}$  at  $p = n - 1 = O(n)$ . Note that for any  $\alpha < 1$ , the training error is  $\frac{1}{n^\alpha}$  before  $O(n)$ .

**Power law training rate decay** Next, we consider power-law decays of training rate error and characterize which power laws are consistent with the interpolation threshold properties outlined above. Consider a power law decay of the mean training error of  $\frac{n^\alpha}{p^\beta}$ . Note that in order to satisfy condition (2), we must have:

$$\frac{1}{n} = O\left(\frac{n^\alpha}{n^\beta}\right) \quad (42)$$

where we set  $p = O(n)$ . Thus,  $\beta - \alpha = 1$ . Power-law decays of the form  $\frac{n^\alpha}{p^{\alpha+1}}$  are consistent with the interpolation threshold property.

**Error scaling in neural networks** Next, we turn to model training error scaling in neural networks. We make the following heuristic argument: to fit  $n$  training points, a network of width  $m$  needs to encode  $O(mn)$  numbers corresponding to  $m + 1$  numbers for each training point (to represent the features and labels of each training point). On the other hand, the network has  $O(m^2)$  parameters since its intermediate layer weights have  $m^2$  parameters. We expect that scaling both the network's capacity of  $O(m^2)$  and the required capacity of  $O(mn)$  at the same rate will not change the training error. Thus, we expect the training error to be a function of  $\frac{mn}{m^2} = \frac{n}{m}$ .

810 To find *which* function of  $\frac{n}{m}$  models the training error, we introduce another argument. We assume  
 811 that the network can be approximated as an ensemble of  $O(m^2)$  submodels, each with  $O(1)$  param-  
 812 eters. Suppose the model output is the average of the outputs of the submodels. Then, assuming that  
 813 the distribution of the submodel outputs on any given training point has variance  $O(1)$  over different  
 814 model initializations, the variance of the ensemble output is  $O(\frac{1}{m^2})$ . Thus, over model initialization,  
 815 the model output at any given training point will center around a mean value with deviations on the  
 816 order of  $O(\frac{1}{m})$ . Assuming that as the network capacity goes to  $\infty$ , the training error goes to 0, the  
 817 mean value of the predicted output on a training point must be the true value; thus, the predicted  
 818 output differs from the true output by  $O(\frac{1}{m})$ . This corresponds to a mean squared error on the train-  
 819 ing points scaling as  $O(\frac{1}{m^2})$ . Finally, to get the dependence on  $n$ , we use the observation that the  
 820 error must depend on only the fraction  $\frac{n}{m}$ . Thus, the mean squared training error scales as  $O(\frac{n^2}{m^2})$ .  
 821

822 Finally, we know by the argument above that power law error rate scaling of the form  $\frac{n^\alpha}{p^{\alpha+1}}$  are  
 823 consistent with the interpolation threshold property, where  $p$  is the effective number of parameters.  
 824 Equating this with the neural network scaling result of  $O(\frac{n^2}{m^3})$ , we must have that  $\alpha = 2$ , yielding:  
 825

$$826 \quad O\left(\frac{n^2}{p^3}\right) = O\left(\frac{n^2}{m^2}\right) \quad (43)$$

827  
 828 Equating the denominators, we have  $p^3 = m^2$ , or  $p = m^{2/3}$ . In other words, the effective number  
 829 of parameters is  $m^{2/3}$ . Since the absolute number of parameters in the network is  $m^2$ , the effective  
 830 number of parameters is the cube root of the absolute parameter count.  
 831  
 832  
 833  
 834  
 835  
 836  
 837  
 838  
 839  
 840  
 841  
 842  
 843  
 844  
 845  
 846  
 847  
 848  
 849  
 850  
 851  
 852  
 853  
 854  
 855  
 856  
 857  
 858  
 859  
 860  
 861  
 862  
 863

## C EXPERIMENTAL DETAILS

### C.1 MODEL TRAINING

We used three benchmark datasets: CIFAR-10, MNIST, and SVHN. Each dataset was subject to preprocessing involving standard normalization. For CIFAR-10 and SVHN, the normalization was performed using means and standard deviations of (0.5, 0.5, 0.5). For MNIST, the normalization used a mean and standard deviation of 0.5.

Two types of neural network architectures were evaluated:

- **Convolutional Neural Network (CNN):** Our CNN architecture consisted of six ReLU activated convolutional layers, with kernel sizes (3, 3, 3, 3, 3, 2 for CIFAR-10/SVHN or 1 for MNIST), strides (2, 1, 2, 1, 1, 1), and number of filters (5s, 10s, 20s, 40s, 80s) where s is a width parameter. This was followed by a fully connected layer.
- **Multilayer Perceptron (MLP):** The MLP architecture comprised six fully connected layers with hidden layer width 10s where s is a scale parameter.

The width parameter was set to values of 1, 2, 5, 10, 20, 50, and 100. When we refer to "model scale", we quantify this as the cube root of the number of network parameters rather than the value of the width parameter. We tested two learning rates (0.001 and 0.01) in combination with two optimizers (Adam and SGD respectively). Label noise was introduced at levels of 0.0 (no noise) and 0.2 (20% noise) to evaluate the robustness of the models. The number of training samples was varied among 100, 200, 500, 1000, 2000, 5000, 10000, 20000, and 50000. The samples were randomly selected from their respective base training sets. A constant batch size of 32 and 100 training epochs were used across all experiments. Mean Squared Error (MSE) was used as the loss function. Model performance was assessed using Mean Squared Error (MSE) on both training and test sets. For reproducibility, we set a manual seed for the PyTorch random number generator. Five different seeds (101, 102, 103, 104, 105) were used to assess the variance in results due to random initialization.

### C.2 SCALE-TIME TRADEOFF VALIDATION

**Linear model** We construct a loss function as:

$$L(\theta) = \|K(R\theta + \beta_0) - \alpha^*\|^2 \quad (44)$$

where  $\alpha^*$  is a target value of  $\alpha$ .  $K$ ,  $R$ ,  $\beta_0$  and  $\alpha^*$  are all independently sampled from unit Gaussians, and  $\theta$  is initialized from a unit Gaussian. We set  $P = 1000$  and  $r = 3$ . We train  $\theta$  via gradient descent with learning rate  $10^{-6}$  and evaluate the number of iterations required to reach various values of the loss for varying values of  $p$ . This setup corresponds to a linear classifier trained with gradient descent on mean squared error using a training batch of  $k = 3$  points where only  $p$  of the  $P = 1000$  parameters are controllable.

**Neural networks** To evaluate the scale time tradeoff, for each model, we compute the minimum number of epochs necessary to achieve a test set MSE below 0.09.

### C.3 SHIFTING INTERPOLATION THRESHOLD

Experimental results are shown for networks trained with SGD.

### C.4 IMPACT OF NOISE AND SHAPE OF LOSS CURVE

Experimental results are shown for networks trained with Adam; since Adam trains faster, this allows us to examine training trends at later points in training.

918 C.5 OPTIMAL SCALE PREDICTIONS  
919

920 To predict error over scale using error over time or vice versa, we treat the quantity  $pt$  as a predictor  
921 of performance, where  $p$  is the effective number of network parameters. Thus, if we wish to know  
922 the error of a network at  $(p_1, t_1)$  and we know the error of the network for all  $p_0$ , then we may  
923 predict the error at  $(p_1, t_1)$  as the error at  $(p_0, (\frac{p_1}{p_0})t_1)$ .  
924

925 C.6 COMPUTING INFRASTRUCTURE  
926

927 Experiments were run on a computing cluster with GPUs ranging in memory size from 11 GB to 80  
928 GB.  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971

## D ADDITIONAL EXPERIMENTS

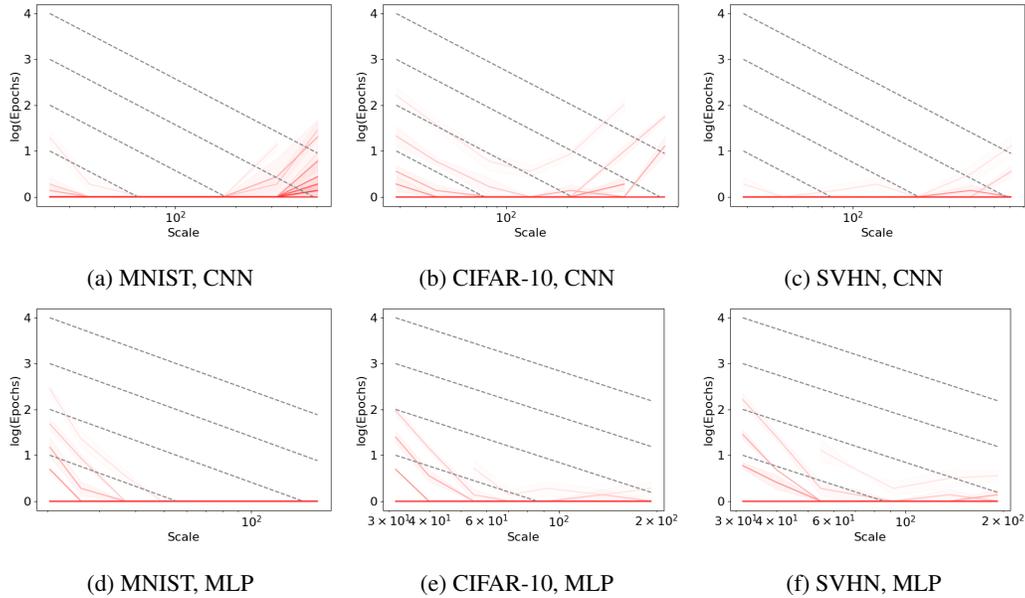


Figure 7: Red lines indicate tradeoff curves between number of training epochs and network scale for different datasets and architectures trained with Adam. Different curves indicate different amounts of training data darker lines indicate more data. Curves are computed by, for each network scale, measuring the minimum amount of training time necessary to achieve non-zero generalization. Margins indicate standard errors over 5 trials. Grey curves are lines of 1:1 proportionality between scale and training epochs.

1026  
 1027  
 1028  
 1029  
 1030  
 1031  
 1032  
 1033  
 1034  
 1035  
 1036  
 1037  
 1038  
 1039  
 1040  
 1041  
 1042  
 1043  
 1044  
 1045  
 1046  
 1047  
 1048  
 1049  
 1050  
 1051  
 1052  
 1053  
 1054  
 1055  
 1056  
 1057  
 1058  
 1059  
 1060  
 1061  
 1062  
 1063  
 1064  
 1065  
 1066  
 1067  
 1068  
 1069  
 1070  
 1071  
 1072  
 1073  
 1074  
 1075  
 1076  
 1077  
 1078  
 1079

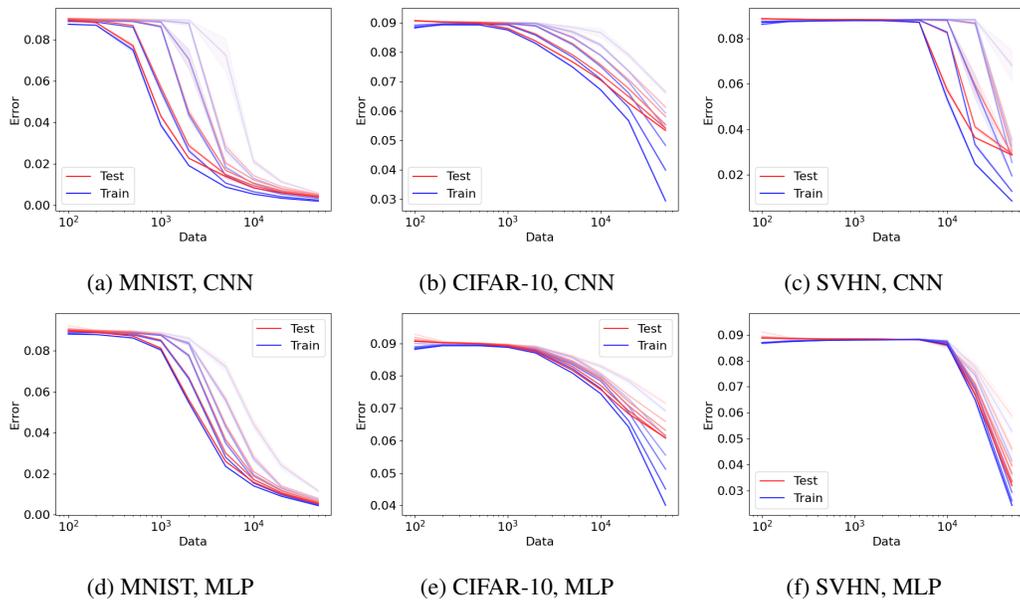


Figure 8: Test and train mean squared error of MLP and CNN models trained on benchmark datasets under varying levels of data. Different curves indicate different model scales; darker colors indicate larger models. Margins indicate standard errors over 5 trials.

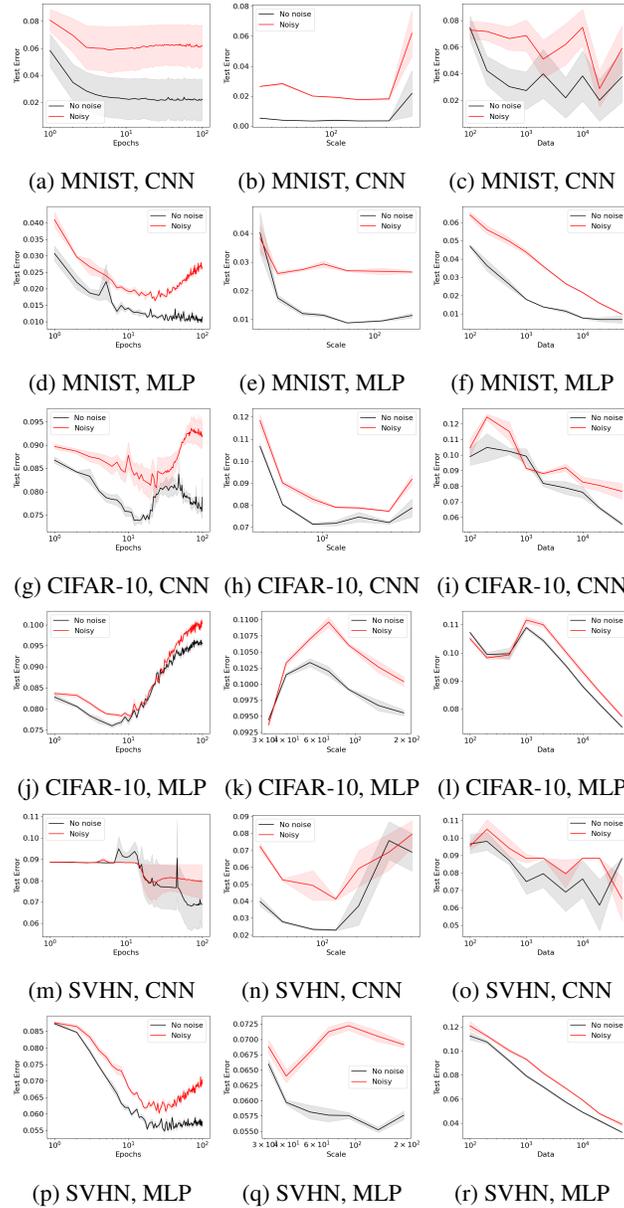


Figure 9: Test mean squared error vs. number of epochs, model scale and training data under noisy and noise-free labels. Each row indicates a different combination of dataset and architecture. Margins indicate standard errors over 5 trials.

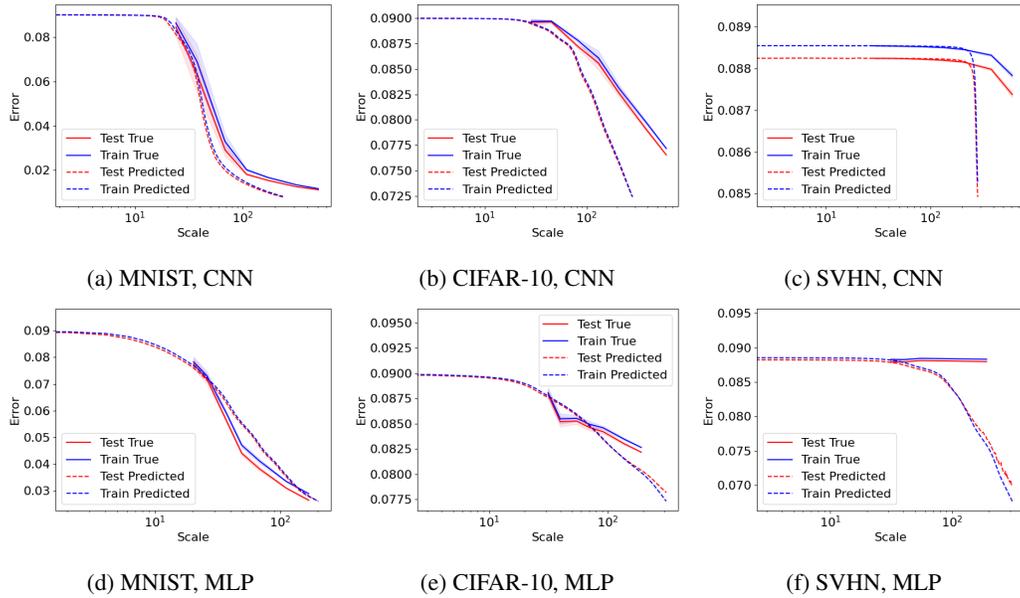


Figure 10: Predicted and true test and train mean squared error of MLP and CNN models trained on benchmark datasets under varying model widths for 10 epochs. Margins indicate standard errors over 5 trials. Predictions are generated by training a small model for 100 epochs and using scale-time equivalence to predict the equivalent scale.

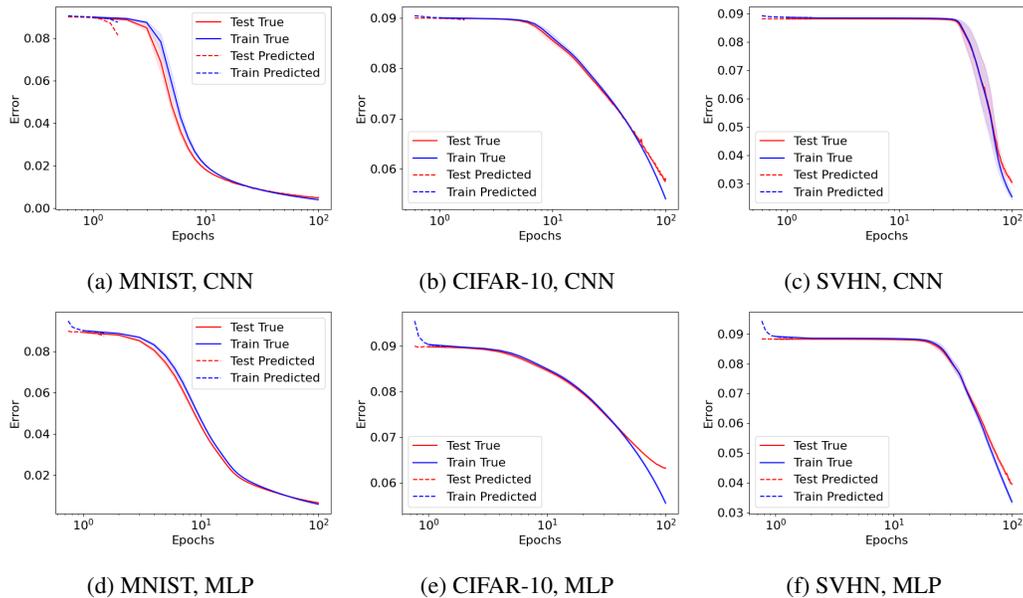


Figure 11: Predicted and true test and train mean squared error of MLP and CNN models trained on benchmark datasets under over training time for a medium-sized model. Margins indicate standard errors over 5 trials. Predictions are generated by training models of varying sizes for 1 epoch and using scale-time equivalence to predict the equivalent number of training epochs.