WHEN DOES SCALE TRADE FOR TIME?

Anonymous authors
Paper under double-blind review

ABSTRACT

Can a larger model trained briefly behave like a smaller model trained for longer? We study this question, which we call *scale-time equivalence*, in a deliberately small and controlled setting where it can be derived, measured, and falsified. Theoretically, we analyze a *random subspace model*, in which the trained network occupies a random p-dimensional subspace of a larger nonlinear model. This model contains Gaussian linear random-feature models, a workhorse of scaling theory, as the special case of a quadratic loss, but, unlike them, allows the network output to depend nonlinearly on the trainable parameters. We prove that under fixed-learning-rate gradient flow, the function learned by a model of scale p after time t is approximated by a p-independent trajectory evaluated at pt, with an error that decays as $p^{-1/2}$ for any fixed value of pt. The bound grows exponentially only in the effective training progress pt, not along the scale-time tradeoff itself, so it guarantees equivalence between models of very different scales as long as both are large and their shared value of pt is moderate. Empirically, in MLPs and CNNs trained with fixed-learning-rate SGD on MNIST, SVHN, and CIFAR-10, the number of epochs needed to generalize follows an approximate power law in model scale, with a common exponent across datasets, architectures, and training-set sizes, and small models trained for long predict large models trained briefly. With Adam, which lies outside the theorem's assumptions, the equivalence visibly breaks. Finally, we conjecture, and give preliminary evidence, that where scale-time equivalence holds, parameter-wise and time-wise double descent share a mechanism: the relative rates at which signal and noise are acquired.

1 INTRODUCTION

Neural network performance improves predictably with model size and data (Kaplan et al., 2020; Rosenfeld et al., 2020; Bahri et al., 2021; Sharma & Kaplan, 2022), but much less is understood about how model size interacts with training time on a fixed dataset. A basic version of this question is: *when can a larger model trained briefly behave like a smaller model trained for longer, and when can it not?* The answer matters for allocating compute and, as we argue in Section 5, for interpreting phenomena such as double descent that are usually analyzed at convergence.

We call the phenomenon in which scale substitutes for time *scale-time equivalence*, and study it in a small, controlled setting chosen so that it can be derived, measured, and falsified: a tractable theoretical model, and MLPs and CNNs on standard vision benchmarks, where we can train complete grids spanning two orders of magnitude in width, nine training-set sizes, two optimizers, and five seeds. **Our goal is not to establish scale-time equivalence for modern large-scale training.** It is to characterize, in a tractable model, a regime in which the equivalence provably holds, check whether the same regime is observable in real networks, and identify where it breaks.

Theory. We analyze a *random subspace model*, in which the trained network is a random p-dimensional subspace of the parameter space of a larger nonlinear model, and its function depends on a fixed low-dimensional projection of the parameters (Section 3.1). This strictly generalizes the Gaussian linear random-feature models that underlie much of current scaling theory (Maloney et al., 2022; Bordelon et al., 2024; Paquette et al., 2024; Lin et al., 2024): they are recovered exactly when the loss is quadratic, whereas our analysis allows the output to be nonlinear in the trainable parameters. We prove that under fixed-learning-rate gradient flow, the learned function depends on p and t primarily through pt (Theorem 3.1). The error bound grows exponentially in pt, the amount of

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: **Evidentiary status of the paper's claims.** "Proved" claims concern the random subspace model only; "observed" claims concern the specific networks and training setups we test.

Claim	Status	Where
Random subspace model, fixed-learning-rate gradient flow: the representation at scale p, time t is within $O\big(p^{-1/2}(e^{c\,pt}-1)\big)$ of a p-independent trajectory evaluated at pt.	Proved	Thm. 3.1, App. A
Same model: models of different scale with equal pt agree up to an error that decreases with scale and does not grow with the scale ratio.	Proved	Cor. 3.2
Tested MLPs and CNNs (MSE loss, fixed-learning-rate SGD, $\leq$100 epochs): time to generalize follows an approximate power law in scale with a common exponent; small models trained longer predict large models trained briefly.	Observed	Sec. 4, Figs. 1–2
Under Adam, the scale-time tradeoff visibly breaks.	Observed	Sec. 4, Fig. 6
Effective scale in these networks $\approx$ (parameter count)$^{1/3}$.	Empirical; heuristic support	Sec. 4, App. B
Where scale-time equivalence holds, parameter-wise double descent can arise from early noise acquisition.	Conjecture; preliminary evidence	Sec. 5
Theorem 3.1 applies directly to real networks; the equivalence is universal or holds empirically for cross-entropy, adaptive optimizers, or large-scale architectures.	*Not claimed*	—

training progress, but *not* along the tradeoff: at fixed pt it *decreases* with scale, so a model of scale p trained for time t approximately matches one of scale kp trained for time t/k, for arbitrarily large k under gradient flow, as long as p is large and pt is moderate (Corollary 3.2).

Experiments. We then test the theory's qualitative predictions in MLPs and CNNs trained with fixed-learning-rate SGD on MNIST, SVHN, and CIFAR-10 (Section 4). The number of epochs needed to generalize decreases as an approximate power law in model scale, with a common exponent across datasets, architectures, and training-set sizes, and small models trained for many epochs predict large models trained for few. Adam, which violates the theorem's fixed-learning-rate assumption, serves as a negative control: there, the equivalence visibly breaks. The theorem is not applied to these networks directly; the experiments test whether the behavior it predicts is present.

A conjecture on double descent. Finally, combining scale-time equivalence with an established analysis of time-wise double descent (Pezeshki et al., 2022; Heckel & Yilmaz, 2021; Stephenson & Lee, 2021), we conjecture that parameter-wise double descent can arise from the same mechanism as time-wise double descent, namely noise being acquired faster than signal, and not only from the interpolation threshold (Section 5). Three experiments that distinguish the two accounts are consistent with the conjecture; we present this part explicitly as a hypothesis with preliminary evidence. Table 1 summarizes what is proved, observed, and conjectured, and what we do not claim.

2 Related work

Scaling laws. Power-law scaling of error with data and model size has been widely observed (Kaplan et al., 2020; Rosenfeld et al., 2020; Clark et al., 2022) and explained theoretically (Bahri et al., 2021; Sharma & Kaplan, 2022; Hutter, 2021; Paquette et al., 2024), often with linear random-feature models; other work questions whether scale alone predicts performance or whether power laws are the right description (Tay et al., 2022; Alabdulmohsin et al., 2022; Bansal et al., 2022; Mahmood et al., 2022). Compute-optimal scaling trades model size against training tokens (Hoffmann et al., 2022), which in the single-epoch regime conflates data and training time; Muennighoff et al. (2023) study repeated epochs over limited data. We instead hold the dataset fixed and ask when scale and training time are interchangeable. Nakkiran et al. (2019) observed qualitatively that loss curves look alike across epochs and network sizes; we characterize this tradeoff quantitatively and relate it to a mechanism.

Table 2: **Random subspace model vs. linear random-feature models.** Gaussian random-feature regression is the special case of our model in which the loss is quadratic in α (Section 3.1).

	Linear random-feature models	**Random subspace model (ours)**
Trainable $\theta \in \mathbb{R}^p$	weights on p fixed random features	coordinates in a random p-dim. subspace of a larger model
Output vs. θ	linear	nonlinear (through α)
Loss vs. θ	convex quadratic	general; typically non-convex
Fixed in training	the random features (KR)	the subspace R and projection K

Double descent. Error can decrease, briefly rise, and decrease again as a function of data volume, model size, or training time (Belkin et al., 2019; Nakkiran et al., 2021). In model and data size, this is usually explained by overfitting at the *interpolation threshold*, where the model is just large enough to fit the training data (Adlam & Pennington, 2020; D'Ascoli et al., 2020; Advani & Saxe, 2017), typically via random matrix theory for random-feature models (Simon et al., 2024; Atanasov et al., 2024; Adlam et al., 2022; Mei & Montanari, 2019; Ali et al., 2019). Nakkiran et al. (2019; 2021) hypothesized that training longer effectively increases model size, so that time-wise double descent inherits this explanation. More recent work instead explains time-wise double descent by features being learned at different rates (Pezeshki et al., 2022; Heckel & Yilmaz, 2021; Stephenson & Lee, 2021): error rises as models fit quickly-learned noisy features and falls as they slowly learn signal features.

3 Scale-time equivalence in a random subspace model

3.1 The random subspace model

Setup. Let $\beta \in \mathbb{R}^P$ be the parameters of a large nonlinear model, which is never trained directly. The network we actually train has parameters $\theta \in \mathbb{R}^p$ ($p \leq P$) and occupies a p-dimensional affine subspace of this larger parameter space:

$$\beta = R\theta + \beta_0, \tag{1}$$

where $\beta_0 \in \mathbb{R}^P$ is fixed and $R \in \mathbb{R}^{P\times p}$ has i.i.d. standard Gaussian entries. Any p-dimensional affine subspace of $\mathbb{R}^P$ can be written in this form; drawing R at random models a *generic* subspace. We further assume that the network's function depends on β only through an r-dimensional linear projection,

$$\alpha = K\beta \in \mathbb{R}^r, \tag{2}$$

where $K \in \mathbb{R}^{r\times P}$ is fixed during training. The network's outputs on all inputs, training and test alike, are (possibly highly nonlinear) functions of α, and we write the training loss as $L(\alpha)$. We therefore call α the network's *representation*: it summarizes everything about the parameters that affects the function. Model scale is the subspace dimension p.

Why this model? The model encodes three properties of neural networks. (i) *Smaller networks are linear subspaces of larger ones.* Fixing the extra weights of a wider network to zero is one embedding (Hall & Li, 1993), but not the only one. For instance, a two-layer network $f(x) = a^\top\sigma(Wx)$ of width m equals the width-km network with stacked weights $W' = [W; \ldots; W]$ and output weights $a' = [c_1 a; \ldots; c_k a]$ for any $c_1 + \cdots + c_k = 1$. For fixed c, the map $(W, a) \mapsto (W', a')$ is linear and its image generically has no zero entries; permuting hidden units gives still more embeddings. A small architecture thus sits inside a larger one in many linear ways, and the random R in equation 1 models a generic one. (ii) *The function depends on few directions in parameter space.* Networks are highly redundant, and gradient descent concentrates in a small, slowly changing subspace (Gur-Ari et al., 2018); the fixed projection K captures this. (iii) *Random subspaces of real networks are expressive.* Training confined to a random low-dimensional subspace of a network's parameters, i.e., the parameterization in equation 1 with the full network as the larger model, recovers much of the performance of full training (Li et al., 2018; Aghajanyan et al., 2021).

Relation to random-feature models. Many theoretical analyses of neural scaling use linear random-feature models, whose output is a linear combination of p fixed random features, hence

linear in the trainable parameters, with a convex quadratic loss (Maloney et al., 2022; Bordelon et al., 2024; Paquette et al., 2024; Lin et al., 2024; Atanasov et al., 2024; Mei & Montanari, 2019); linearized analyses of wide networks share this property (Jacot et al., 2018; Chizat et al., 2019). The random subspace model contains the Gaussian version of these models as a special case. Let α collect the network's outputs on a finite set of inputs, and let L be the squared error on the training inputs. Then $\alpha = KR\theta + K\beta_0$ is the output of p random features whose values on these inputs, the columns of KR, are i.i.d. $\mathcal{N}(0, KK^\top)$: Gaussian random features with kernel matrix $KK^\top$. Equivalently, β weights a fixed P-dimensional feature map, whose values on the inputs form the rows of K, and $R^\top$ is a random Gaussian projection of it, as in the sketched linear models of Bordelon et al. (2024); Paquette et al. (2024); Lin et al. (2024) (up to the normalization of R; Section 3.3). Our model removes exactly one assumption of this setting, that L is quadratic in α, and thereby allows outputs that are nonlinear in the trainable parameters and losses that are non-convex in them, as in real networks (Table 2). The remaining assumptions, a fixed Gaussian R and a fixed K, are exactly those that define Gaussian random features. Note that α itself is linear in θ: nonlinearity enters through the map from α to the outputs and through L. These assumptions are still idealizations; in particular, feature learning can rotate the function-relevant directions that K holds fixed. Our experiments (Section 4) therefore test the model's predictions rather than presuppose its assumptions.

3.2 Main result

We train θ by continuous-time gradient flow, $\dot\theta = -\eta \nabla_\theta L(K(R\theta + \beta_0))$, from $\theta_0 = 0$, with a learning rate η that is the same for every scale p. Let $\alpha_t = K(R\theta_t + \beta_0)$. Throughout, $\|\cdot\|$ denotes the Euclidean norm for vectors and the operator norm for matrices; let $\lambda_K = \|KK^\top\|$ and $C_K = \sqrt{\|K\|_F^4 + \|KK^\top\|_F^2} \,/\, \lambda_K$.

Theorem 3.1 (Scale-time equivalence in the random subspace model)**.** *Suppose $L : \mathbb{R}^r \to \mathbb{R}$ has Lipschitz constant l and its gradient has Lipschitz constant h. Let $A_t \in \mathbb{R}^r$ solve*

$$\dot A_t = -\eta KK^\top \nabla L(A_t), \qquad A_0 = K\beta_0, \tag{3}$$

which does not depend on p or R. Then for any $\epsilon \in (0,1)$, with probability at least $1-\epsilon$ over R, simultaneously for all $t \geq 0$,

$$\|\alpha_t - A_{pt}\| \leq \frac{l\, C_K}{h\sqrt{p\epsilon}} \left(e^{\eta h \lambda_K\, pt} - 1\right). \tag{4}$$

The theorem (proof in Appendix A) bounds the distance between the scale-p model's representation at time t and a reference trajectory that depends on p and t only through pt. Because α determines the network's outputs on every input, the same bound controls any quantity that is a Lipschitz function of α, such as the *test* loss when the map from α to test outputs is Lipschitz.

Why scale acts like time. Under gradient flow on θ, the representation evolves as $\dot\alpha = -\eta\, KRR^\top K^\top \nabla L(\alpha)$. Each of the p columns of R contributes an independent random direction, and $\mathbb{E}[KRR^\top K^\top] = pKK^\top$: adding trainable directions speeds up the motion of α linearly in p, while the fluctuations around this mean grow only as $\sqrt{p}$. Rescaling time by p therefore yields the p-independent dynamics in equation 3, up to a relative error of order $p^{-1/2}$. Scale-time equivalence is thus about *models of different scale trained with the same learning rate*: larger models move faster in function space. The content of the theorem is that they also move along the *same* trajectory, up to fluctuations that vanish relative to the mean.

Corollary 3.2 (Equivalence along curves of constant pt; proof in Appendix A)**.** *Fix $s > 0$ and scales p_1, p_2, and let $t_i = s/p_i$. Let $\alpha^{(i)}$ denote the representation of the model of scale p_i (with its own random subspace). Then, with probability at least $1 - 2\epsilon$,*

$$\left\|\alpha^{(1)}_{t_1} - \alpha^{(2)}_{t_2}\right\| \leq \frac{l\, C_K}{h\sqrt{\epsilon}} \left(e^{\eta h \lambda_K s} - 1\right) \left(\frac{1}{\sqrt{p_1}} + \frac{1}{\sqrt{p_2}}\right). \tag{5}$$

Remark 3.3 (On the assumptions)**.** (a) The Lipschitz conditions are needed only where training goes: $L(\alpha_t)$ and $L(A_{pt})$ are non-increasing in t, so l may be taken as a bound on $\|\nabla L\|$, and h as a Lipschitz constant of ∇L, on the sublevel set $\{\alpha : L(\alpha) \leq L(K\beta_0)\}$. This covers the squared loss:

in the random-feature case with $L(\alpha) = \|\alpha_{\text{train}} - y\|^2$, $l = 2\sqrt{L(K\beta_0)}$ and $h = 2$. (b) Gaussianity of R is not essential: for any i.i.d. entries with mean zero, unit variance, and finite fourth moment, the bound holds with a modified constant (Appendix A, Remark A.1).

3.3 When is the bound informative?

The exponential factor depends on training progress, not on the scale ratio. The bound in equation 4 depends on p and t through two factors: $p^{-1/2}$, and $e^{\eta h \lambda_K pt} - 1$, which depends only on the product $s = pt$. It is useful to read it along two directions of the (p, t) plane.

- *Along the tradeoff ($pt = s$ fixed).* Trading scale for time, which is what scale-time equivalence is about, leaves the exponential factor unchanged, and the bound *decreases* as $p^{-1/2}$. By Corollary 3.2, the discrepancy between two models on the same curve is controlled by the smaller scale and does not grow with the ratio p_2/p_1: relating scale p at time t to scale $1000p$ at time $t/1000$ incurs the same exponential factor as relating p to $2p$.
- *Across the tradeoff (s increasing).* Training further increases the bound exponentially in s, through a worst-case Grönwall-type estimate over all losses with the given smoothness constants. This limits *how much training progress* the guarantee covers, not how far one can move along the tradeoff.

Regime of validity. Take $P \to \infty$ with C_K and λ_K bounded, so that p may be arbitrarily large. For any $s_{\max}$ and accuracy $\rho > 0$, the right-hand side of equation 4 is at most ρ for all $pt \le s_{\max}$ once $p \ge C(s_{\max})/\rho^2$. Scale-time equivalence thus holds to arbitrary accuracy for large enough models over a bounded amount of effective progress, i.e., for $t = O(1/p)$, and becomes exact as $p \to \infty$ with pt fixed. This parallels the infinite-width limit of neural tangent kernel theory (Jacot et al., 2018): parameters move vanishingly little ($t \to 0$) while the function moves a finite amount ($pt > 0$). Unlike that limit, however, the limiting dynamics in equation 3 do not linearize the network: outputs remain nonlinear functions of A_t. We call this *local* scale-time equivalence: it holds in a region of the (p, t) plane (large p, bounded pt), not globally.

When the guarantee does not apply. The theorem gives no guarantee for small p, for large pt (much effective progress, however it is split between p and t), or when its assumptions fail, most importantly when the effective learning rate differs across scales. Normalization matters even within the model: rescaling R by $p^{-1/2}$, as in NTK-style parameterizations, gives $\mathbb{E}[KRR^\top K^\top] = KK^\top$ and removes the speedup. Equivalently, the unnormalized scale-p model is a normalized one trained with learning rate ηp, so in discrete time, stability (roughly $\eta p h \lambda_K < 2$) caps the scales over which the equivalence can hold at a fixed step size. In our experiments, wider networks trained with a fixed learning rate do generalize sooner (Section 4), as in the unnormalized case. Adaptive optimizers such as Adam (Kingma & Ba, 2015) rescale each parameter's update by gradient statistics that themselves change with width, so the effective learning rate is no longer fixed across scales; the theorem is silent there, and we use Adam as a negative control.

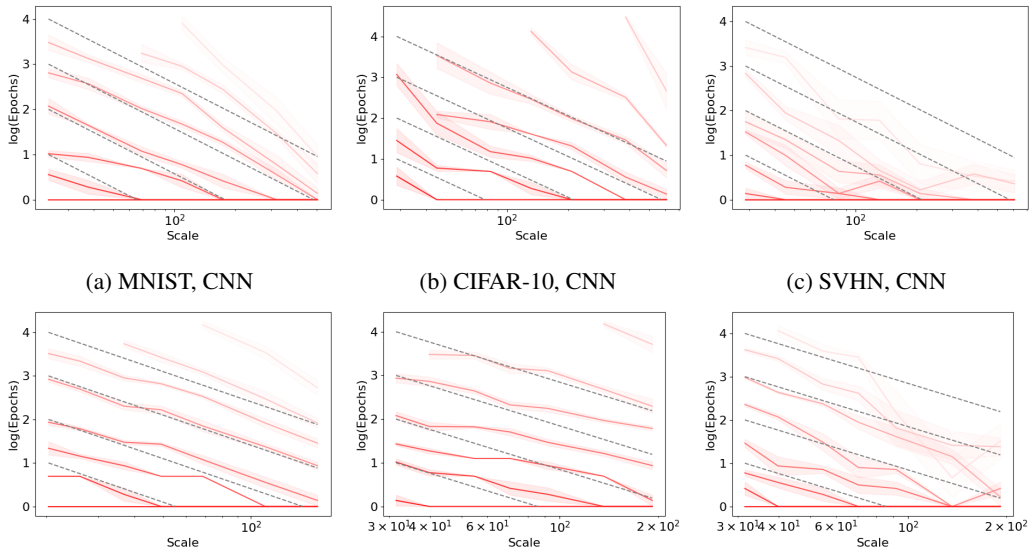
3.4 From the theorem to testable predictions

Theorem 3.1 is proved for the random subspace model, not for neural networks. We use it as a source of qualitative predictions, which we test in real networks in Section 4:

P1 (iso-performance curves). If test performance depends on (p, t) only through pt, then the first time $t^*(p)$ at which the test loss crosses a fixed threshold satisfies $p\, t^*(p) = \text{const}$, i.e., $t^* \propto p^{-1}$: a straight line of slope -1 in log-log coordinates. Changing the training set changes L, and hence the constant, but not the slope, so curves for different dataset sizes should be roughly parallel. The approximation should be most accurate for large p and early training. Simulations of the model itself confirm this (Appendix E, Figure 5).

P2 (extrapolation). The performance of a model of scale p_1 at time t_1 can be predicted from a model of scale p_0 at time $(p_1/p_0)\, t_1$.

P3 (optimizer dependence). The guarantee requires a learning rate that is fixed across scales; with adaptive optimizers the relationship need not hold.



(a) MNIST, CNN (b) CIFAR-10, CNN (c) SVHN, CNN

(d) MNIST, MLP (e) CIFAR-10, MLP (f) SVHN, MLP

Figure 1: **Test of P1: time to generalize trades off against model scale (fixed-learning-rate SGD).** Red curves show, for each model scale $p_{\text{eff}} = (\text{parameter count})^{1/3}$, the minimum number of training epochs needed for the test MSE to fall below 0.09 (log-log axes). Darker curves use more training data. Grey lines have slope -1 (1:1 proportionality between scale and epochs); the exponent $1/3$ is fixed once for all panels (Section 4). Margins indicate standard errors over 5 seeds. The Adam counterpart is in Figure 6.

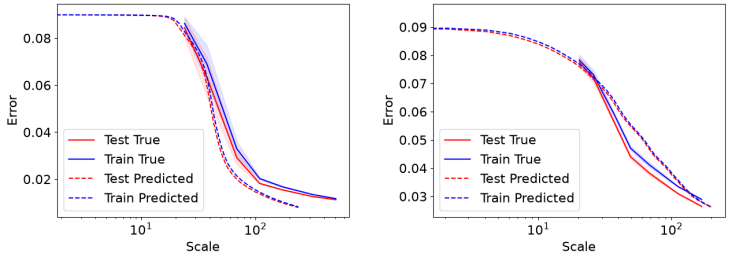


Figure 2: **Test of P2: small models trained for long predict large models trained briefly.** Predicted and measured test and train MSE of a CNN (left) and MLP (right) on MNIST, as a function of model scale, after 10 epochs of training. Predictions use a small model trained for up to 100 epochs and the correspondence $(p_1, t_1) \leftrightarrow (p_0, (p_1/p_0)\, t_1)$ with $p = p_{\text{eff}}$. Margins indicate standard errors over 5 seeds. Full results in Figures 9 and 10.

Effective scale. The p of the theorem counts independent random directions in the trainable subspace. A real network's parameters are neither independent nor random directions, so there is no reason for p to equal the raw parameter count. We therefore test P1 in the form $t^* \propto p_{\text{eff}}^{-1}$, with $p_{\text{eff}} = (\text{parameter count})^{\gamma}$ and a *single* exponent γ for all datasets, architectures, and training-set sizes. The testable content of P1 is then that (a) tradeoff curves are straight in log-log coordinates, (b) one exponent serves all settings, and (c) curves for different data sizes are roughly parallel. The value of γ is estimated, not predicted.

4 Testing the predictions in neural networks

Setup. We train 6-layer MLPs and 7-layer CNNs on MNIST (Deng, 2012), SVHN (Goodfellow et al., 2013), and CIFAR-10 (Krizhevsky, 2009) with squared (MSE) loss on one-hot labels, using

SGD with a constant learning rate that is the same at every width, batch size 32, and at most 100 epochs. We vary a width multiplier from 1 to 100 and the training-set size from 100 to 50,000, with 5 seeds per configuration (Appendix D). These benchmarks are small enough to train the complete grids that a controlled test of P1–P3 requires, and the widest models are large relative to the tasks; they are not meant to represent modern large-scale training. *Why MSE?* Squared loss matches the signal/noise decomposition used in Section 5 and is competitive with cross-entropy for classification (Hui & Belkin, 2021). Theorem 3.1 does not require squared loss, but our empirical conclusions are limited to MSE-trained classifiers; cross-entropy is a natural next test.

Measuring time to generalize. In these networks, the test loss stays nearly flat early in training and then begins to fall. For each run, we record t^*, the first epoch at which the test MSE falls below 0.09. With the MSE averaged over the 10 output dimensions, an input-independent predictor that outputs the class frequencies attains a test MSE of approximately 0.09, so t^* marks the onset of input-dependent generalization. Our qualitative conclusions are insensitive to this particular value: for each threshold we tried in $[0.08, 0.092]$, the tradeoff curves remain approximately straight, with the same qualitative pattern. Outside this range, t^* stops being informative for practical reasons: below it, some models never reach the threshold within 100 epochs, so t^* is censored; above it, nearly all models cross within the first epoch, so t^* reflects epoch-level resolution and seed-to-seed noise. Finally, Theorem 3.1 is most accurate for large models early in training, and the experiments stay as close to this regime as our setup allows: t^* is by construction the onset of generalization, all runs last at most 100 epochs, and widths reach $100\times$ the base model. We cannot, however, verify that the bound is quantitatively informative for these networks. Because the theorem controls α, and hence predictions on test inputs, comparing test error across models at matched pt, as t^* does, is the kind of comparison it addresses.

P1: a predictable tradeoff between scale and time. Figure 1 shows t^* against model scale. In all six dataset-architecture pairs, the curves are close to straight lines in log-log coordinates: multiplying scale by a fixed factor divides the time needed to generalize by a corresponding fixed factor. Curves for different training-set sizes are shifted relative to one another but have similar slopes, as P1 predicts.

Effective scale. The tradeoff is approximately 1:1 not in the raw parameter count but in its cube root: doubling training time is roughly equivalent to multiplying the parameter count by 2^3. We therefore fix $p_{\text{eff}} = (\text{parameter count})^{1/3}$, i.e., $\gamma = 1/3$, once for all datasets, architectures, and data sizes; with this single choice, the curves in all settings approximately follow slope -1. This choice does not affect whether curves are straight or how they are ordered: using the raw parameter count instead only rescales the logarithmic horizontal axis by a factor of 3, and using the width multiplier as the scale variable leads to the same qualitative conclusions. Appendix B gives a heuristic argument for the exponent $1/3$; we treat it as an empirical regularity of these architecture families, not a universal constant.

P3: Adam as a negative control. Under Adam, the tradeoff curves depart clearly from scale-time equivalence (Appendix E, Figure 6). Thus our measurement procedure does not produce straight tradeoff curves with a common slope automatically, and the equivalence is not a generic property of wide networks trained briefly: it depends on the optimizer, as the theorem's fixed-learning-rate assumption suggests.

P2: extrapolating across scale and time. If performance depends on (p_{eff}, t) only through $p_{\text{eff}}\, t$, a small model trained for long should predict a large model trained briefly. We train a small model for up to 100 epochs and use this correspondence to predict the error of wider models trained for 10 epochs (Figure 2; full results in Figures 9–10). The predictions capture the measured trends and closely locate the scale at which generalization begins. Each prediction pairs a smaller model trained longer with a larger model trained briefly at the same $s = p_{\text{eff}}\, t$; in the random subspace model, this is the direction along which the approximation error does not grow with the scale ratio (Corollary 3.2).

5 A CONJECTURED CONSEQUENCE: EARLY NOISE ACQUISITION AND DOUBLE DESCENT

This section is more speculative than Sections 3–4: we combine scale-time equivalence with an existing account of time-wise double descent into a conjecture about parameter-wise double descent, and report three experiments that are consistent with it but do not establish it.

5.1 BACKGROUND: TIME-WISE DOUBLE DESCENT IN A LINEAR MODEL

We summarize existing results (Pezeshki et al., 2022; Heckel & Yilmaz, 2021; Stephenson & Lee, 2021; Schaeffer et al., 2023) with a linear student-teacher model; these works report that such analyses predict behavior in nonlinear networks. Training labels are $Y = Xw + \varepsilon \in \mathbb{R}^n$, with inputs $X \in \mathbb{R}^{n\times m}$ (n points, model dimension m), true weights $w \in \mathbb{R}^m$, and zero-mean label noise ε independent of (X, x, w). For a linear model trained from zero by gradient flow on the squared error with learning rate η, the expected squared error on a test point x after time t splits into a signal term and a noise term (derivation in Appendix C):

$$\mathbb{E}[(x^\top w - x^\top \theta_t)^2] = \mathbb{E}\Big[\Big(\sum_{i=1}^{m} S_i e^{-\eta\sigma_i^2 t}\Big)^2\Big] + \mathbb{E}\Big[\Big(\sum_{i=1}^{m} N_i \frac{1-e^{-\eta\sigma_i^2 t}}{\sigma_i}\Big)^2\Big]. \tag{6}$$

Here $X = U\Sigma V^\top$ is a singular value decomposition with singular values σ_i (set to 0 for $i > n$, in which case the noise term is 0), and $S_i = -(x^\top V)_i (V^\top w)_i$ and $N_i = (x^\top V)_i (U^\top \varepsilon)_i$ measure how strongly the test point aligns with the true model and with the noise along each singular mode.

Scaling with time. The signal terms of equation 6 decrease with t, most sharply (on a logarithmic time axis) around $t = 1/(\eta\sigma_i^2)$ for component i; the noise terms *increase*, most sharply at the same time. Where noise dominates ($|N_i|/\sigma_i > |S_i|$), error therefore rises around that component's timescale. Time-wise double descent occurs when noise-dominated components are acquired before signal-dominated ones (see the schematic in Appendix C, Figure 4).

Scaling with data and model size. Data size n enters mainly through the σ_i. Near the interpolation threshold ($n \approx m$), the smallest nonzero singular values of the near-square random matrix X become small, inflating the corresponding noise terms; moving n away from m reduces them, yielding double descent in n. Varying model size m has the same effect.

5.2 THE CONJECTURE

Conjecture 5.1 (informal)**.** *Where scale-time equivalence holds, the test error as a function of scale p and time t is approximately obtained from equation 6 by replacing t with pt:*

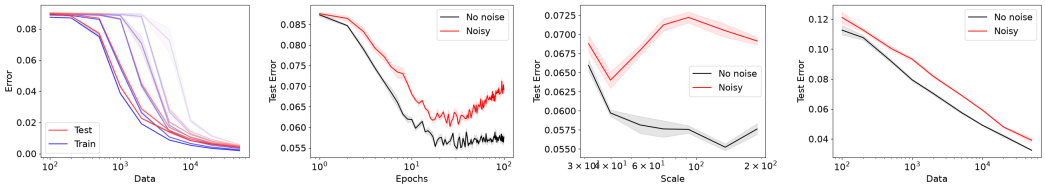
$$\textit{error}^2 \approx \mathbb{E}\Big[\Big(\sum_i S_i e^{-\eta\sigma_i^2 pt}\Big)^2\Big] + \mathbb{E}\Big[\Big(\sum_i N_i \frac{1-e^{-\eta\sigma_i^2 pt}}{\sigma_i}\Big)^2\Big], \tag{7}$$

where the σ_i depend on the training data and S_i, N_i set the strengths of signal and noise. In particular, parameter-wise double descent can arise from the same mechanism as time-wise double descent: when noise components are acquired faster than signal components, models of intermediate scale, which by scale-time equivalence have made intermediate effective progress, have fit noise but not yet signal.

The interpolation-threshold account locates the error peak in small singular values in the *denominators* of the noise term when model size is close to n; the conjecture locates it in the *exponents*, i.e., in the order in which components are acquired. The two are not mutually exclusive, but they make different predictions.

5.3 THREE TESTS

Test 1: less data is needed to generalize at larger scale (SGD). The interpolation threshold in data volume is where the number of training points equals the model's effective complexity (Nakkiran et al., 2019), so it lies at larger data volumes for larger models. The conventional account places



(a) MNIST, CNN (SGD) (b) SVHN, MLP (Adam) (c) SVHN, MLP (Adam) (d) SVHN, MLP (Adam)

Figure 3: **Tests of the double-descent conjecture (Section 5.3).** (a) Test 1: test and train MSE of CNNs on MNIST trained with SGD, versus training-set size; darker curves are larger models. (b–d) Tests 2–3: test MSE of MLPs on SVHN trained with Adam, versus epochs (b), model scale (c), and training-set size (d), with and without 20% label noise. Full results in Figures 7 and 8. Margins indicate standard errors over 5 seeds.

an error peak at this threshold, after which error drops as data grows, and so predicts that larger models need *more* data before their error drops. Under the conjecture, larger models behave like smaller models trained longer, so they should need *less* data, in line with the observation that larger models are more sample-efficient (Kaplan et al., 2020). Figure 3(a) (full results in Figure 7) shows that larger models generalize with less data. (*Training* error also falls with more data, since a fixed number of epochs then means more gradient steps.)

Test 2: label noise has a global effect along scale and time, but a local one along data size. In the interpolation-threshold account, increasing the noise ($|N_i|$) raises the error mainly *near* $m \approx n$, where the terms it multiplies peak (Schaeffer et al., 2023). In the conjecture, noise raises the error increasingly with both model scale and time (equation 7), i.e., globally. With 20% label noise on the training set (Figure 3(b–d); all settings in Figure 8), error increases in all cases, persistently or increasingly along training time and model scale, but only locally along data size: for MNIST and SVHN, the error at the largest data volume barely changes.

Test 3: error can turn upward with model size. In standard analyses of the interpolation-threshold account, error typically decreases with model size beyond the threshold. Under the conjecture, it can *hurt* if signal is acquired before noise, just as training longer can. CIFAR-10 and SVHN indeed show a U-shaped error curve in model scale (Figure 3(c) and Figure 8).

Which optimizer each test uses. Test 1 uses fixed-learning-rate SGD. Tests 2 and 3 use Adam, which reaches the late phase of training, where noise effects appear, within our 100-epoch budget. Since Adam lies outside the regime of Theorem 3.1 and does not show clean scale-time equivalence (Section 4), Tests 2 and 3 therefore fall outside the conjecture's stated scope. They show that the qualitative signatures it predicts also occur there, but they are not evidence for scale-time equivalence itself and support the conjecture more weakly than Test 1.

6 Discussion

Limitations. Beyond the scope summarized in Table 1: (1) The theory abstracts real training: K is fixed, R is random rather than structured, and training is continuous-time gradient flow, whereas discrete steps limit how far the equivalence can extend at a fixed step size. Extending the analysis to a time-varying K, i.e., to feature learning, is the most important theoretical direction. (2) The exponent $1/3$ relating effective to raw parameter count is an empirical regularity with a heuristic justification (Appendix B), not a derived quantity. (3) Our experiments use MSE-trained MLPs and CNNs on small vision benchmarks; we make no claim about cross-entropy training, other optimizers, or large-scale architectures such as transformers. (4) Two of the three double-descent tests use Adam (Section 5.3).

Outlook. Beyond cross-entropy and larger architectures, parameter-efficient fine-tuning is a natural test bed: LoRA (Hu et al., 2022) and random-subspace training (Li et al., 2018; Aghajanyan et al., 2021) train a low-dimensional parameterization of a much larger model, whose dimension may trade off against training time.

AI USE STATEMENT

In this work, we used generative AI tools for writing and finding related work. We have not used generative AI tools for developing the core ideas of this work, experiments or proofs. We have reviewed all AI-assisted work. All LLM generated text was reviewed carefully by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ben Adlam and Jeffrey Pennington. Understanding double descent requires a fine-grained bias-variance decomposition. In *NeurIPS*, 2020. https://arxiv.org/abs/2011.03321.

Ben Adlam, Jake Levinson, and Jeffrey Pennington. A random matrix perspective on mixtures of nonlinearities for deep learning. *AISTATS*, 2022.

Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint*, 2017.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL*, pp. 7319–7328, 2021.

Ibrahim Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. In *NeurIPS*, 2022.

Alnur Ali, J Zico Kolter, and Ryan J Tibshirani. A continuous-time view of early stopping for least squares regression. In *AISTATS*, 2019.

Alexander Atanasov, Blake Bordelon, Sabarish Sainathan, and Cengiz Pehlevan. The onset of variance-limited behavior for networks in the lazy and rich regimes. *ICLR*, 2024.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint*, 2021.

Yamini Bansal, Behrooz Ghorbani, Ankush Garg, Biao Zhang, Colin Cherry, Behnam Neyshabur, and Orhan Firat. Data scaling laws in nmt: The effect of noise and architecture. In *ICML*, pp. 1466–1482. PMLR, 2022.

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, 2019.

Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. *ICML*, 2024.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *NeurIPS*, 32, 2019.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, pp. 4057–4086. PMLR, 2022.

Stéphane D'Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double trouble in double descent: Bias and variance(s) in the lazy regime. In *ICML*, 2020.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint*, 2013.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint*, 2018.

Peter Hall and Ker-Chau Li. On almost Linearity of Low Dimensional Projections from High Dimensional Data. *The Annals of Statistics*, 21(2):867 – 889, 1993. doi: 10.1214/aos/1176349155. URL https://doi.org/10.1214/aos/1176349155.

Reinhard Heckel and Fatih Furkan Yilmaz. Early stopping in deep networks: Double descent and how to eliminate it. In *ICLR*, 2021. URL https://openreview.net/forum?id=tlV90jvZbw.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *NeurIPS*, 35:30016–30030, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ICLR*, 2022.

Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *ICLR*, 2021.

Marcus Hutter. Learning curve theory. *arXiv preprint*, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *NeurIPS*, 31, 2018.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *ICLR*, 2018.

Licong Lin, Jingfeng Wu, Sham M Kakade, Peter L Bartlett, and Jason D Lee. Scaling laws in linear regression: Compute, parameters, and data. *NeurIPS*, 2024.

Rafid Mahmood, James Lucas, David Acuna, Daiqing Li, Jonah Philion, Jose M Alvarez, Zhiding Yu, Sanja Fidler, and Marc T Law. How much more data do i need? estimating requirements for downstream tasks. In *CVPR*, pp. 275–284, 2022.

Alexander Maloney, Daniel A Roberts, and James Sully. A solvable model of neural scaling laws. *arXiv preprint*, 2022.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint*, 2019.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language models. *NeurIPS*, 36:50358–50376, 2023.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. *NeurIPS*, 2019.

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, 2021.

Elliot Paquette, Courtney Paquette, Lechao Xiao, and Jeffrey Pennington. 4+ 3 phases of compute-optimal neural scaling laws. *NeurIPS*, 2024.

Mohammad Pezeshki, Amartya Mitra, Yoshua Bengio, and Guillaume Lajoie. Multi-scale feature learning dynamics: Insights for double descent. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17669–17690. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/pezeshki22a.html.

Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. In *ICLR*, 2020.

Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathy, Kateryna Pistunova, Jason W Rocks, Ila Rani Fiete, and Oluwasanmi Koyejo. Double descent demystified: Identifying, interpreting & ablating the sources of a deep learning puzzle. *arXiv preprint*, 2023.

Utkarsh Sharma and Jared Kaplan. Scaling laws from the data manifold dimension. *JMLR*, 23:9–1, 2022.

James B Simon, Dhruva Karkada, Nikhil Ghosh, and Mikhail Belkin. More is better in modern machine learning: when infinite overparameterization is optimal and overfitting is obligatory. *ICLR*, 2024.

Cory Stephenson and Tyler Lee. When and how epochwise double descent happens. *arXiv preprint*, 2021.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. In *ICLR*, 2022.

A PROOFS

Throughout, $\|\cdot\|$ denotes the Euclidean norm for vectors and the operator norm for matrices, and K_i denotes the ith row of K.

Dynamics of the representation. Gradient flow on θ is

$$\dot{\theta} = -\eta R^\top K^\top \nabla L(\alpha), \tag{8}$$

so the representation evolves as

$$\dot{\alpha} = -\eta K R R^\top K^\top \nabla L(\alpha). \tag{9}$$

Properties of $KRR^\top K^\top$. Since the entries of R are i.i.d. standard Gaussian, $\mathbb{E}[RR^\top] = pI$, and hence

$$\mathbb{E}[KRR^\top K^\top] = pKK^\top. \tag{10}$$

Writing $R_{:,q}$ for the qth column of R, the (i,j) entry of $KRR^\top K^\top$ is

$$K_i R R^\top K_j^\top = \sum_{q=1}^{p} (K_i R_{:,q})(K_j R_{:,q}). \tag{11}$$

The p summands are independent because the columns of R are independent. For each q, $K_i R_{:,q}$ and $K_j R_{:,q}$ are linear functions of the same standard Gaussian vector $R_{:,l}$, hence jointly Gaussian with mean zero, variances $\|K_i\|^2$ and $\|K_j\|^2$, and covariance $K_i K_j^\top$.

Expanding the square and using independence across $q \neq q'$,

$$\mathbb{E}[(K_i R R^\top K_j^\top)^2] = \sum_q \mathbb{E}[(K_i R_{:,q})^2 (K_j R_{:,q})^2] + p(p-1)(K_i K_j^\top)^2. \tag{12}$$

To compute the fourth moment, write $K_i R_{:,q} = au$ and $K_j R_{:,q} = bu + cv$, where u, v are independent standard Gaussians, $a = \|K_i\|$, $b = K_i K_j^\top / a$, and $c = \sqrt{\|K_j\|^2 - b^2}$ (if $K_i = 0$, both sides of the identity below vanish). Then

$$\begin{aligned}\mathbb{E}[(K_i R_{:,q})^2 (K_j R_{:,q})^2] &= a^2 b^2 \mathbb{E}[u^4] + 2a^2 bc\, \mathbb{E}[u^3 v] + a^2 c^2 \mathbb{E}[u^2 v^2] \\ &= 3a^2 b^2 + a^2 c^2 = \|K_i\|^2 \|K_j\|^2 + 2(K_i K_j^\top)^2, \end{aligned} \tag{13}$$

using $\mathbb{E}[u^4] = 3$, $\mathbb{E}[u^3 v] = 0$, and $\mathbb{E}[u^2 v^2] = 1$. Therefore

$$\mathbb{E}[(K_i R R^\top K_j^\top)^2] = p\big(\|K_i\|^2 \|K_j\|^2 + 2(K_i K_j^\top)^2\big) + p(p-1)(K_i K_j^\top)^2. \tag{14}$$

Decomposition into mean and fluctuation. Write

$$K R R^\top K^\top = pKK^\top + N, \tag{15}$$

where $N \in \mathbb{R}^{r \times r}$ has mean zero. Each entry of N has variance

$$\mathbb{E}[N_{ij}^2] = \mathbb{E}[(K_i R R^\top K_j^\top)^2] - p^2 (K_i K_j^\top)^2 = p\big(\|K_i\|^2 \|K_j\|^2 + (K_i K_j^\top)^2\big), \tag{16}$$

so

$$\mathbb{E}[\|N\|_F^2] = \sum_{i,j} \mathbb{E}[N_{ij}^2] = p\big(\|K\|_F^4 + \|KK^\top\|_F^2\big). \tag{17}$$

The dynamics of α become

$$\dot\alpha = -\eta p K K^\top \nabla L(\alpha) - \eta N \nabla L(\alpha). \tag{18}$$

Comparing noisy and noise-free dynamics. Consider the noise-free dynamics $\alpha^{(1)}$ and the true (noisy) dynamics $\alpha^{(2)} = \alpha$, both started at $K\beta_0$:

$$\dot\alpha^{(1)} = -\eta p K K^\top \nabla L(\alpha^{(1)}), \qquad \dot\alpha^{(2)} = -\eta p K K^\top \nabla L(\alpha^{(2)}) - \eta N \nabla L(\alpha^{(2)}). \tag{19}$$

The discrepancy $\delta = \alpha^{(2)} - \alpha^{(1)}$ satisfies

$$\dot\delta = -\eta p K K^\top [\nabla L(\alpha^{(2)}) - \nabla L(\alpha^{(1)})] - \eta N \nabla L(\alpha^{(2)}), \tag{20}$$

so

$$\|\dot\delta\| \le \eta p \lambda_K \|\nabla L(\alpha^{(2)}) - \nabla L(\alpha^{(1)})\| + \eta \|N\| \|\nabla L(\alpha^{(2)})\| \le \eta p h \lambda_K \|\delta\| + \eta \|N\| l, \tag{21}$$

where the last step uses $\|\nabla L(\alpha^{(2)}) - \nabla L(\alpha^{(1)})\| \le h\|\delta\|$ and $\|\nabla L(\alpha^{(2)})\| \le l$.

These two conditions are needed only on the sublevel set $S_0 = \{\alpha : L(\alpha) \le L(K\beta_0)\}$ (Remark 3.3a). Indeed, both trajectories stay in S_0: $\frac{d}{dt} L(\alpha_t^{(2)}) = -\eta \|R^\top K^\top \nabla L(\alpha_t^{(2)})\|^2 \le 0$, since $\alpha^{(2)}$ is the image of gradient flow on θ, and $\frac{d}{dt} L(\alpha_t^{(1)}) = -\eta p \|K^\top \nabla L(\alpha_t^{(1)})\|^2 \le 0$. For the squared loss $L(\alpha) = \|\alpha_{\text{train}} - y\|^2$ we have $\|\nabla L(\alpha)\| = 2\sqrt{L(\alpha)}$ and a constant Hessian, giving $l = 2\sqrt{L(K\beta_0)}$ and $h = 2$ on S_0.

By Markov's inequality applied to equation 17, with probability at least $1 - \epsilon$,

$$\|N\| \le \|N\|_F \le \sqrt{\frac{p}{\epsilon}\big(\|K\|_F^4 + \|KK^\top\|_F^2\big)}. \tag{22}$$

On this event, which does not depend on t, writing $c_1 = \eta p h \lambda_K$ and $c_2 = \eta\, l \sqrt{p/\epsilon} \sqrt{\|K\|_F^4 + \|KK^\top\|_F^2}$, we have $\|\dot\delta_t\| \le c_1 \|\delta_t\| + c_2$, and hence, since $\delta_0 = 0$,

$$\|\delta_t\| \le \int_0^t \|\dot\delta_\tau\|\, d\tau \le \int_0^t \big(c_1 \|\delta_\tau\| + c_2\big)\, d\tau. \tag{23}$$

Grönwall's inequality (integral form) then gives, for all $t \ge 0$,

$$\|\delta_t\| \le \frac{c_2}{c_1}\big(e^{c_1 t} - 1\big) = \frac{l\, C_K}{h\sqrt{p\epsilon}}\big(e^{\eta h \lambda_K p t} - 1\big). \tag{24}$$

Solving the noise-free dynamics. By the chain rule, $t \mapsto A_{pt}$ satisfies

$$\frac{d}{dt}A_{pt} = p\,\dot{A}_{pt} = -\eta p K K^\top \nabla L(A_{pt}), \tag{25}$$

with $A_0 = K\beta_0$, which is the noise-free dynamics. Hence $\alpha_t^{(1)} = A_{pt}$.

Conclusion. Combining, with probability at least $1-\epsilon$, for all $t \geq 0$,

$$\|\alpha_t - A_{pt}\| = \|\alpha_t^{(2)} - \alpha_t^{(1)}\| = \|\delta_t\| \leq \frac{l\,C_K}{h\sqrt{p\epsilon}}\big(e^{\eta h \lambda_K p t} - 1\big). \qquad \square \tag{26}$$

Proof of Corollary 3.2. Apply Theorem 3.1 to each model at time $t_i = s/p_i$, so that $p_i t_i = s$: with probability at least $1-\epsilon$ each, $\|\alpha_{t_i}^{(i)} - A_s\| \leq lC_K(e^{\eta h \lambda_K s} - 1)/(h\sqrt{p_i\epsilon})$. By a union bound, both hold with probability at least $1-2\epsilon$ (regardless of any dependence between the two subspaces), and the triangle inequality gives equation 5. $\square$

Remark A.1 (Non-Gaussian subspaces)**.** Gaussianity is used only to compute $\mathbb{E}[(K_i R_{:,q})^2 (K_j R_{:,q})^2]$. If the entries of R are i.i.d. with mean 0, variance 1, and fourth moment μ_4, then $\mathbb{E}[RR^\top] = pI$ still holds, and direct expansion gives $\mathbb{E}[(K_i R_{:,q})^2 (K_j R_{:,q})^2] = \|K_i\|^2\|K_j\|^2 + 2(K_i K_j^\top)^2 + (\mu_4 - 3)\sum_k K_{ik}^2 K_{jk}^2$. Summing over i, j and using $\sum_k (\sum_i K_{ik}^2)^2 \leq \|K\|_F^4$,

$$\mathbb{E}[\|N\|_F^2] \leq p\big((1 + \max(\mu_4 - 3, 0))\|K\|_F^4 + \|KK^\top\|_F^2\big), \tag{27}$$

so Theorem 3.1 and Corollary 3.2 hold with $\|K\|_F^4$ replaced by $(1 + \max(\mu_4 - 3, 0))\|K\|_F^4$ in C_K. For example, Rademacher entries ($\mu_4 = 1$) give exactly the Gaussian bound.

B A HEURISTIC FOR THE EFFECTIVE PARAMETER COUNT

This appendix gives a heuristic argument, not a derivation, for why the cube root of the parameter count behaves as an effective parameter count in our architectures. Defining the effective parameter count requires locating the interpolation threshold, but networks rarely interpolate their training data exactly. Instead, we identify a property of the interpolation threshold in linear models and ask at which parameter count a network exhibits the same property.

Training error in linear models. Let $X \in \mathbb{R}^{n \times p}$ have i.i.d. standard Gaussian entries and let $Y \in \mathbb{R}^n$ satisfy $Y_i^2 = 1$. The minimum-norm least-squares solution is $\theta = X^\dagger Y$, with training error

$$\frac{1}{n}\|Y - XX^\dagger Y\|_2^2 = \frac{1}{n}Y^\top(I - XX^\dagger)Y. \tag{28}$$

For $p < n$, $XX^\dagger$ projects onto a uniformly random p-dimensional subspace, so the expected training error is

$$\mathbb{E}\Big[\frac{1}{n}\|Y - XX^\dagger Y\|_2^2\Big] = \max\Big(1 - \frac{p}{n}, 0\Big). \tag{29}$$

The training error decreases linearly in p with slope $-1/n$ and vanishes at the interpolation threshold $p = n$. We extract the following property: *the training error reaches order* $1/n$ *when the parameter count is of order* n (it equals $1/n$ at $p = n - 1$).

Power-law training error. Suppose the training error decays as n^a/p^b. The property above requires $n^a/n^b = \Theta(1/n)$ at $p = \Theta(n)$, i.e., $b = a + 1$. Power laws of the form n^a/p^{a+1} are thus consistent with the interpolation-threshold property.

Training error in neural networks. Consider a network of width m. To fit n training points, it must encode $O(mn)$ numbers ($O(m)$ features per training point), while it has $O(m^2)$ parameters in its hidden-to-hidden weights. Scaling capacity and requirement at the same rate should leave the training error unchanged, so we expect it to be a function of $mn/m^2 = n/m$. To find which function, suppose the network behaves like an average of $O(m^2)$ submodels with $O(1)$ parameters each, whose outputs on a training point have $O(1)$ variance over initializations. The averaged output then deviates from its mean by $O(1/m)$, and if the training error vanishes as capacity grows, this mean must equal the target, giving a mean squared training error of $O(1/m^2)$ at fixed n. Since the error depends only on n/m, it scales as $O(n^2/m^2)$.

Effective parameter count. Matching $O(n^2/m^2)$ to the form n^a/p^{a+1} requires $a = 2$ and $p^3 = m^2$, i.e., $p = m^{2/3} = (m^2)^{1/3}$. Since the raw parameter count scales as m^2, the effective parameter count scales as its cube root. This argument locates the effective count through the interpolation threshold, whereas Section 3.4 uses it to set the speed of training; in the linear model the same p governs both, and we assume the same holds for the effective count. Each step above is heuristic; for instance, in narrow networks the input layer, rather than the $O(m^2)$ hidden-to-hidden weights, can dominate the parameter count. We therefore use the exponent $1/3$ as an empirically supported choice (Section 4) rather than a derived one.

C Derivation of the linear model of time-wise double descent

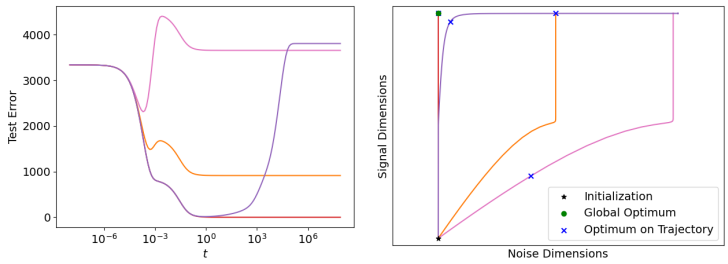
With $Y = Xw + \varepsilon$, gradient flow on $\frac{1}{2}\|X\theta - Y\|^2$ with learning rate η from $\theta_0 = 0$ gives

$$\theta_t = X^\dagger (I - e^{-\eta X X^\top t}) Y. \tag{30}$$

Writing $X = U\Sigma V^\top$, so that $X^\dagger = V\Sigma^\dagger U^\top$ and $e^{-\eta X X^\top t} = U e^{-\eta \Sigma\Sigma^\top t} U^\top$, the prediction error on a test point x is

$$\begin{aligned} x^\top \theta_t - x^\top w &= x^\top V[\Sigma^\dagger (I - e^{-\eta\Sigma\Sigma^\top t})\Sigma - I]V^\top w + x^\top V \Sigma^\dagger (I - e^{-\eta\Sigma\Sigma^\top t}) U^\top \varepsilon \\ &= \sum_{i=1}^{m} -(x^\top V)_i (V^\top w)_i e^{-\eta\sigma_i^2 t} + (x^\top V)_i (U^\top \varepsilon)_i \frac{1 - e^{-\eta\sigma_i^2 t}}{\sigma_i}, \end{aligned} \tag{31}$$

where $\sigma_i = 0$ for $i > n$ when $n < m$, and the corresponding noise term is 0. Since ε has mean zero and is independent of (X, x, w), the cross term vanishes in expectation, and squaring gives equation 6.



(a) Loss trajectories (b) Parameter space

Figure 4: **Double-descent behavior depends on the order in which signal and noise are acquired (schematic).** Loss trajectories (a) and corresponding parameter-space trajectories (b) of linear regression under different noise settings (colors). Deviations along noise dimensions increase test error. Depending on the noise profile, error may temporarily increase as at an interpolation threshold (pink: large; orange: small), increase persistently (purple), or not increase noticeably (red).

D Experimental details

D.1 Model training

We used three benchmark datasets: CIFAR-10, MNIST (Creative Commons license), and SVHN (non-commercial use only). Each dataset was normalized in the standard way: for CIFAR-10 and SVHN, with per-channel means and standard deviations of (0.5, 0.5, 0.5); for MNIST, with mean and standard deviation 0.5.

We evaluated two architectures:

- Convolutional neural network (CNN): 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 numbers of filters (5s, 5s, 10s, 20s, 40s, 80s), where s is the width multiplier, followed by a fully connected layer.

- Multilayer perceptron (MLP): six fully connected layers with hidden width 10s, where s is the width multiplier.

The width multiplier took values 1, 2, 5, 10, 20, 50, and 100. "Model scale" refers to the cube root of the number of network parameters, not to the width multiplier. We used SGD with learning rate 0.01 and Adam with learning rate 0.001; each optimizer uses the same learning rate at every width. Label noise was introduced at levels 0.0 (no noise) and 0.2 (20% of training labels corrupted). The number of training samples was 100, 200, 500, 1000, 2000, 5000, 10000, 20000, or 50000, randomly selected from the respective training set. All experiments used batch size 32 and 100 training epochs, and the loss was the mean squared error (MSE). Performance was measured as MSE on the training and test sets. Five seeds (101–105) for the PyTorch random number generator were used to assess variability due to random initialization.

D.2 Scale-time tradeoff

Random subspace simulation. We simulate the random subspace model with a quadratic loss,

$$L(\theta) = \|K(R\theta + \beta_0) - \alpha^*\|^2, \tag{32}$$

where α^* is a target value of α. K, R, β_0, and α^* are sampled independently from standard Gaussians, and θ is initialized from a standard Gaussian. We set $P = 1000$ and $r = 3$, train θ by gradient descent with learning rate 10^{-6}, and record the number of iterations needed to reach various loss values for varying p. This corresponds to a linear model with $P = 1000$ parameters fit by squared loss to $r = 3$ training points, trained only within a random p-dimensional subspace of its parameter space.

Neural networks. For each model, we compute the minimum number of epochs needed to reach a test MSE below 0.09. As discussed in Section 4, the qualitative conclusions are unchanged for any threshold in $[0.08, 0.092]$.

D.3 Shifting interpolation threshold

Results are for networks trained with SGD.

D.4 Impact of noise and shape of the loss curve

Results are for networks trained with Adam, which trains faster and thus lets us observe later phases of training within 100 epochs. As discussed in Section 5.3, these experiments lie outside the regime of Theorem 3.1.

D.5 Extrapolation across scale and time

To predict error across scale from error across time, or vice versa, we treat pt as the predictor of performance, where p is the effective parameter count. To predict the error of a network at (p_1, t_1) from a network of scale p_0 whose error is known over time, we use its error at $(p_0, (p_1/p_0)\, t_1)$.

D.6 Computing infrastructure

Experiments were run on a computing cluster with GPUs ranging in memory size from 11 GB to 80 GB.

E Additional experiments

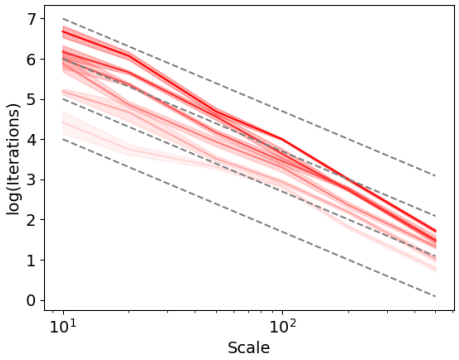
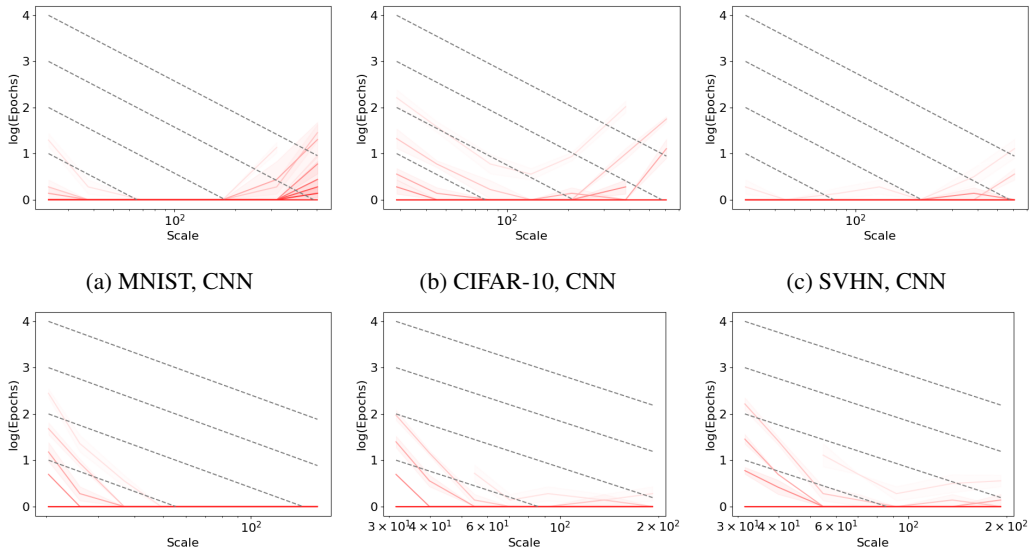


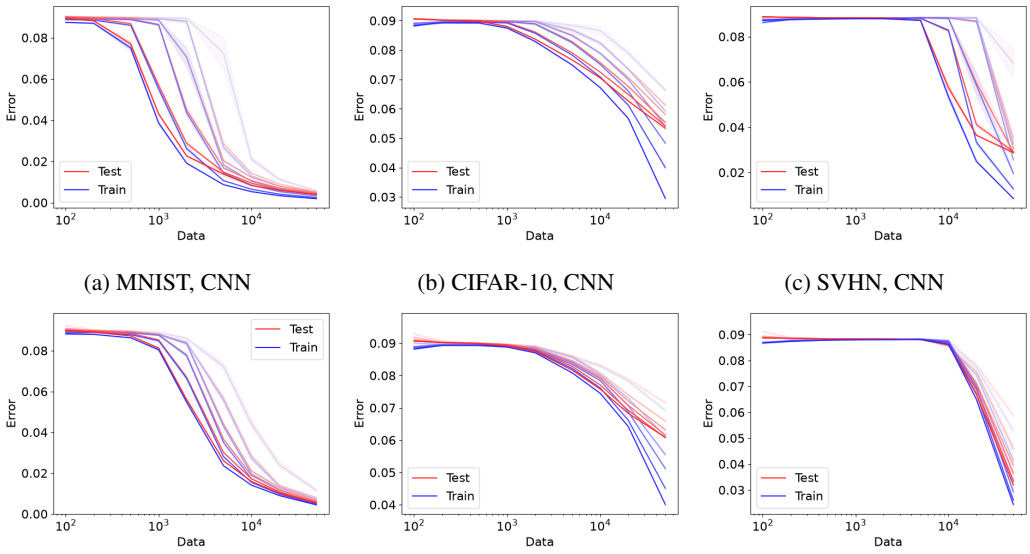
Figure 5: **Proportional tradeoff between model scale and training time in a direct simulation of the random subspace model.** Red curves show, for each subspace dimension p, the minimum number of gradient descent iterations needed to reach a given loss; darker curves correspond to smaller loss thresholds. Margins indicate standard errors over 5 trials. Grey dashed lines indicate 1:1 proportionality between scale and iterations.



(a) MNIST, CNN (b) CIFAR-10, CNN (c) SVHN, CNN

(d) MNIST, MLP (e) CIFAR-10, MLP (f) SVHN, MLP

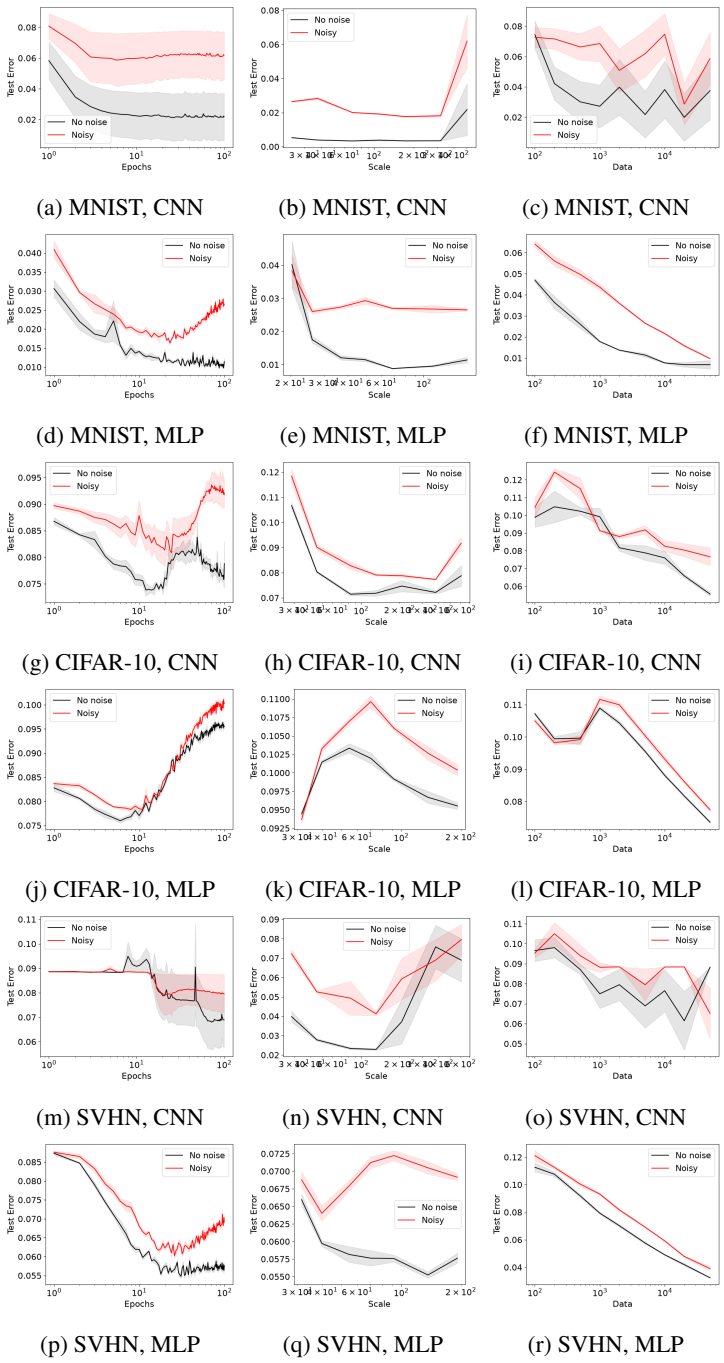
Figure 6: **Negative control (P3): with Adam, the scale-time tradeoff departs from equivalence.** Same measurement as Figure 1, for networks trained with Adam. Red curves show the minimum number of epochs needed for the test MSE to fall below 0.09, for each model scale; darker curves use more training data. Margins indicate standard errors over 5 seeds. Grey lines indicate 1:1 proportionality between scale and training epochs.



(a) MNIST, CNN (b) CIFAR-10, CNN (c) SVHN, CNN

(d) MNIST, MLP (e) CIFAR-10, MLP (f) SVHN, MLP

Figure 7: Test and train MSE of MLPs and CNNs trained with SGD on benchmark datasets, as a function of training-set size. Darker curves are larger models. Margins indicate standard errors over 5 seeds.



(a) MNIST, CNN (b) MNIST, CNN (c) MNIST, CNN

(d) MNIST, MLP (e) MNIST, MLP (f) MNIST, MLP

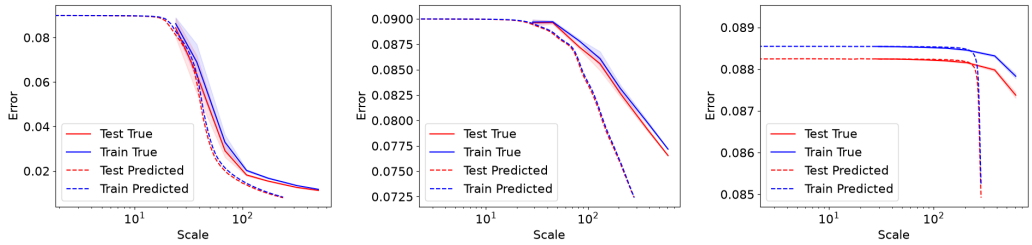
(g) CIFAR-10, CNN (h) CIFAR-10, CNN (i) CIFAR-10, CNN

(j) CIFAR-10, MLP (k) CIFAR-10, MLP (l) CIFAR-10, MLP

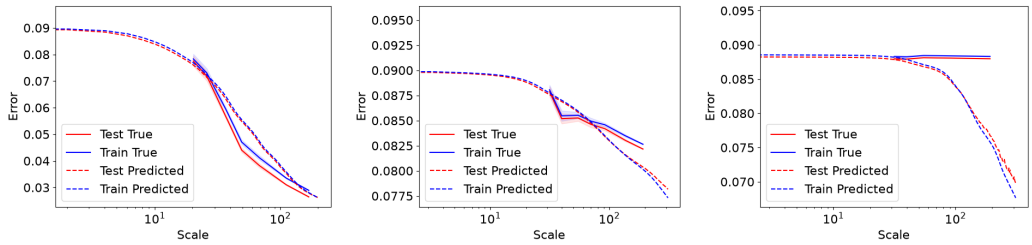
(m) SVHN, CNN (n) SVHN, CNN (o) SVHN, CNN

(p) SVHN, MLP (q) SVHN, MLP (r) SVHN, MLP

Figure 8: Test MSE vs. number of epochs, model scale, and training-set size, with and without 20% label noise, for networks trained with Adam. Each row is a dataset-architecture pair. Margins indicate standard errors over 5 seeds.

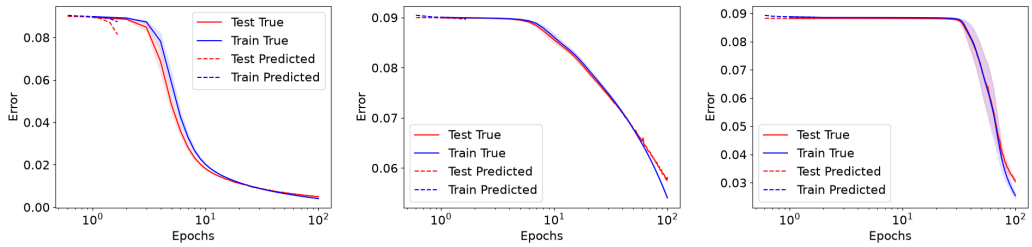


(a) MNIST, CNN (b) CIFAR-10, CNN (c) SVHN, CNN

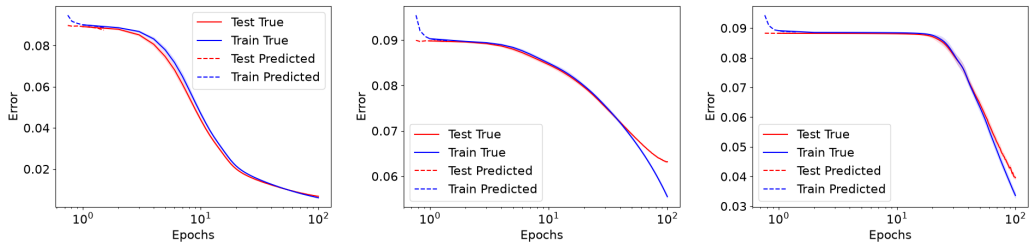


(d) MNIST, MLP (e) CIFAR-10, MLP (f) SVHN, MLP

Figure 9: **Extrapolating across scale (P2).** Predicted and measured test and train MSE of MLPs and CNNs trained for 10 epochs, as a function of model scale. Predictions are generated by training a small model for up to 100 epochs and using scale-time equivalence to map its training time to an equivalent scale. Margins indicate standard errors over 5 seeds.



(a) MNIST, CNN (b) CIFAR-10, CNN (c) SVHN, CNN



(d) MNIST, MLP (e) CIFAR-10, MLP (f) SVHN, MLP

Figure 10: **Extrapolating across time (P2).** Predicted and measured test and train MSE of a medium-sized MLP or CNN as a function of training time. Predictions are generated by training models of varying sizes for 1 epoch and using scale-time equivalence to map their scale to an equivalent number of epochs. Predicted curves are truncated because training the corresponding larger models is computationally infeasible. Margins indicate standard errors over 5 seeds.