Supporting Information for
Explaining Neural Scaling Laws

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, Utkarsh Sharma

Corresponding Author: Yasaman Bahri
E-mail: yasamanbahri@gmail.com

This PDF file includes:
- Supporting text
- Figs. S1 to S8
- Table S1
- SI References
Supporting Information Text

1. Experimental setup

A. Figure 1 (top-left). Experiments utilize relatively small models, with the number of trainable parameters $P \sim \mathcal{O}(1000)$, trained with full-batch gradient descent (GD) and small learning rate on datasets of size $D \gg P$. Each data point in the figure represents an average over subsets of size $D$ sampled from the full dataset. Experiments are done using the Neural Tangents (1) library based on JAX (2). All experiment except denoted as (CNN), use 3-layer, width-8 fully-connected networks. Convolutional neural network (CNN) architecture used is Myrtle-5 network (3) with 8 channels. ReLU activation function with critical initialization (4–6) was used. Unless specified cross-entropy loss was used. We performed full-batch gradient descent update for all dataset sizes without $L_2$ regularization. 20 different training data sampling seeds were averaged for each point. For fully-connected networks, input pooling of size 4 was performed for CIFAR-10/100 dataset and pooling of size 2 was performed for MNIST and Fashion-MNIST dataset. This was to reduce number of parameters in the input layer ( # of pixels \times width) which can be quite large even for small width networks.

B. Figure 1 (top-right). All experiments were performed using a Flax (7) implementation of Wide ResNet 28-10 (8). Models were trained for 78125 total steps with a cosine learning rate decay (9) and an augmentation policy consisting of random flips and crops. We report final loss, though we found no qualitative difference between using final loss, best loss, final accuracy or best accuracy (see Figure S1).

C. Figure 1 (bottom-left). The setup was identical to Figure 1 (top-right) except that the model considered was a depth 10 residual network with varying width.

D. Figure 1 (bottom-right). Experiments are done using Neural Tangents. All experiments use 100 training samples and two-hidden layer fully-connected networks of varying width (ranging from $w = 64$ to $w = 11, 585$) with ReLU nonlinearities unless specified as Erf. Full-batch gradient descent and cross-entropy loss were used unless specified as MSE, and the figure shows curves from a random assortment of training times ranging from 100 to 500 steps (equivalently, epochs). Training was done with learning rates small enough so as to avoid catapult dynamics (10) and no $L_2$ regularization; in such a setting, the infinite-width learning dynamics is known to be equivalent to that of linearized models (11). Consequently, for each random initialization of the parameters, the test loss of the finite-width linearized model was additionally computed in the identical training setting. This value approximates the limiting behavior $L(\infty)$ known theoretically and is subtracted off from the final test loss of the (nonlinear) neural network before averaging over 50 random initializations to yield each of the individual data points in the figure.

E. Deep teacher-student models. The teacher-student scaling with dataset size (figure S2) was performed with fully-connected teacher and student networks with two hidden layers and widths 96 and 192, respectively, using PyTorch (12). The inputs were random vectors sampled uniformly from a hypercube of dimension $d = 2, 3, \cdots, 9$. To mitigate noise, we ran the experiment on eight different random seeds, fixing the random seed for the teacher and student as we scanned over dataset sizes. We also used a fixed test dataset, and a fixed training set, which was sub-sampled for the experiments with smaller $D$. The student networks were trained using MSE loss and Adam optimizer with a maximum learning rate of $3 \times 10^{-3}$, a cosine learning rate decay, and a batch size of 64, and 40,000 steps of training. The test losses were measured with early stopping. We combine test losses from different random seeds by averaging the logarithm of the loss from each seed.

In our experiments, we always use inputs that are uniformly sampled from a $d$-dimensional hypercube, following the setup of (13). They also utilized several intrinsic dimension (ID) estimation methods and found the estimates were close to the input dimension, so we simply use the latter for comparisons. For the dataset size scans, we used randomly initialized teachers with width 96 and students with width 192. We found similar results with other network sizes.

The final scaling exponents and input dimensions are shown in the bottom of Figure 1b.

We used the same experiments for the top of that figure, interpolating the behavior of both teacher and a set of students between two fixed training points. The students only differed by the size of their training sets but had the same random seeds and were trained in the same way. In that figure, the input space dimension was four.

Finally, we also used a similar setup to study variance-limited exponents and scaling. In that case we used much smaller models, with 16-dimensional hidden layers, and a correspondingly larger learning rate. We then studied scaling with $D$ again, with results pictured in Figure 1a.

F. CNN architecture for resolution-limited scaling. Figure 1b includes data from CNN architectures trained on image datasets. The architectures are summarized in Table S1. We used Adam optimizer for training with cross-entropy loss. Each network was trained for long enough to achieve either a clear minimum or a plateau in test loss. Specifically, CIFAR-10, MNIST and Fashion MNIST were trained for 50 epochs, CIFAR-100 was trained for 100 epochs, and SVHN was trained for 10 epochs. The default Keras training parameters were used. In case of SVHN, we included the additional images as training data. We averaged (in log space) over 20 runs for CIFAR-100 and CIFAR-10, 16 runs for MNIST, 12 runs for Fashion MNIST, and 5 runs for SVHN. The results of these experiments are shown in Figure S3.

The measurement of input-space dimensionality for these experiments was done using the nearest-neighbour algorithm, described in detail in Appendix B and C in (13). We used 2, 3 and 4 nearest neighbors and averaged over the three.
G. Teacher-student experiment for scaling of loss with model size. We replicated the teacher-student setup in (13) to demonstrate the scaling of loss with model size. The resulting variation of $-4/\alpha_P$ with input-space dimensionality is shown in figure S4. In our implementation we averaged (in log space) over 15 iterations, with a fixed, randomly generated teacher.

H. Effect of aspect ratio on scaling exponents. We trained Wide ResNet architectures of various widths and depths on CIFAR-10 across dataset sizes. We found that the effect of depth on dataset scaling was mild for the range studied, while the effect of width impacted the scaling behavior up until a saturating width, after which the scaling behavior fixed. See Figure S5.

2. Proof of Theorem 1

We now prove Theorem 1, repeated below for convenience.

**Theorem 1** Let $\ell(f)$ be the test loss as a function of network output, $(L = \mathbb{E}[\ell(f)])$, and let $f_T$ be the network output after $T$ training steps, thought of as a random variable over weight initialization, draws of the training dataset, and optimization seed. Further let $f_T$ be concentrating with $\mathbb{E}[(f_T - \mathbb{E}[f_T])^k] = \mathcal{O}(\epsilon) \forall k \geq 2$. If $\ell$ is a finite degree polynomial, or has bounded second derivative, or is $2$-Hölder, then $\mathbb{E}[\ell(f_T)] - \ell(\mathbb{E}[f_T]) = \mathcal{O}(\epsilon)$.

**Case 1 – finite degree polynomial:** In this case, we can write,

$$\ell(f_T) - \ell(\mathbb{E}[f_T]) = \sum_{k=1}^{K} \frac{\ell^{(k)}(f_T)}{k!} (f_T - \mathbb{E}[f_T])^k, \quad [S1]$$

where $K$ is the polynomial degree and $\ell^{(k)}$ is the $k$-th derivative of $\ell$. Taking the expectation of Eq. (S1) and using the moment scaling proves the result.

**Case 2 – bounded second derivative:** The quadratic mean value theorem states that for any $f_T$, there exists a $c$ such that,

$$\ell(f_T) - \ell(\mathbb{E}[f_T]) = (f_T - \mathbb{E}[f_T]) \ell'(\mathbb{E}[f_T]) + \frac{1}{2} \ell''(c) (f_T - \mathbb{E}[f_T])^2. \quad [S2]$$

Taking the expectation of Eq. (S2) and using the fact that $f''(c)$ is bounded yields the desired result.

**Case 3 – 2-Hölder:** Lastly, the loss being 2-Hölder means we may write,

$$\ell(f_T) - \ell(\mathbb{E}[f_T]) \leq |\ell(f_T) - \ell(\mathbb{E}[f_T])| \leq K\ell (f_T - \mathbb{E}[f_T])^2. \quad [S3]$$

Again, taking the expectation of this inequality completes the proof.

**Remarks on loss variance.** Theorem 1 concerns the mean loss, however we would also like to understand if this scaling holds for typical instances. This can be understood by examining how the variance of the loss or alternatively how $\mathbb{E}[(\ell(f_T) - \ell(\mathbb{E}[f_T])^2$ scales.

For **Case 3 – 2-Hölder loss**, we can rerun the argument of Theorem 1, using Eq. (S3) to yield $\mathbb{E}[(\ell(f_T) - \ell(\mathbb{E}[f_T])^2] = \mathcal{O}(\epsilon)$. For **Cases 1 and 2**, we can attempt to apply the same argument as in the proof. This almost works. In particular, using Hölder’s inequality, $\mathbb{E}[(f_T - \mathbb{E}[f_T])^k] = \mathcal{O}(\epsilon) \forall k \geq 2$ implies $\mathbb{E}[(f_T - \mathbb{E}[f_T])^k] = \mathcal{O}(\epsilon) \forall k \geq 2$. Taking the absolute value and expectation of Eq. (S1) or Eq. (S2) then gives

$$\mathbb{E}[(\ell(f_T) - \ell(\mathbb{E}[f_T])^2] \leq |\ell'(\mathbb{E}[f_T])| |f_T - \mathbb{E}[f_T]| + \mathcal{O}(\epsilon). \quad [S4]$$

In general, the above assumptions on $\ell$ and $f_T$ imply only that $\mathbb{E}[|f_T - \mathbb{E}[f_T]|] = \mathcal{O}(\sqrt{\epsilon})$ and thus typical instances of the loss will exhibit a less dramatic scaling with $\epsilon$ than the mean. If we further assume, however, that $f_T$ on average has been trained such as to be sufficiently close to a local minimum of the loss, such that $|\ell'(\mathbb{E}[f_T])| = \mathcal{O}(\sqrt{\epsilon})$, then typical instances will also obey the $\mathcal{O}(\epsilon)$ scaling.

3. Variance-limited dataset scaling

In this section, we expand on our discussion of the variance-limited dataset scaling, $L(D) = \lim_{D \to \infty} L(D) = \mathcal{O}(D^{-1})$. We first explain some intuition for why this behavior might be expected for sufficiently smooth loss. We then derive it explicitly for losses that are polynomial in the weights. Finally, we present non-smooth examples where the scaling can be violated either by having unbounded loss, or first derivative.

**A. Intuition.** At a high level, the intuition is as follows. For any fixed value of weights, $\theta$, the training loss with $D$ training points (thought of as a random variable over draws of the dataset), $L_{\text{train}}[\theta]$ concentrates around the population loss $L_{\text{pop}}[\theta]$, with variance that scales as $\mathcal{O}(D^{-1})$.

Our optimization procedure can be thought of as a map from initial weights and training loss to final weights $\text{Op} : (\theta_0, L_{\text{train}}[\theta_0]) \rightarrow \theta_T$. If this map is sufficiently smooth – for instance satisfying the assumptions of Theorem 1 or well approximated by a Taylor series about all $\mathbb{E}[L_{\text{train}}[\theta_0]]$ – then the output, $\theta_T$, will also concentrate around its infinite $D$ limit with variance scaling as $\mathcal{O}(D^{-1})$. Finally, if the population loss is also sufficiently smooth, the test loss for a model trained on $D$ data points averaged over draws of the dataset, $L(D) = \mathbb{E}_D[L_{\text{pop}}[\theta_T]]$, satisfies $L(D) - \lim_{D \to \infty} L(D) = \mathcal{O}(D^{-1})$. We now walk through this in a little more detail.
Early time. We can follow this intuition a bit more explicitly for the first few steps of gradient descent. As the training loss at initialization, \( L_{\text{train}}[\theta_0] \), is a sample average over \( D \) i.i.d draws, it concentrates around the population loss \( L_{\text{pop}}[\theta_0] \) with variance \( O(D^{-1}) \). As a result, the initial gradient, \( g_0 = \frac{\partial L_{\text{train}}}{\partial \theta_{\text{pop}}} \) will also concentrate with \( O(D^{-1}) \) variance and so will the weights at time step 1, \( \theta_1 = \theta_0 - \eta g_0 \). The training loss at time step 1 is then given by (setting \( \eta = 1 \)),

\[
L_{\text{train}}[\theta_1] = L_{\text{train}}[\theta_0 - g_0].
\]

If \( L_{\text{train}} \) is sufficiently smooth around \( \theta_0 - E_D[g_0] \), then we get that \( L_{\text{train}}[\theta_1] \) concentrates around \( L_{\text{train}}[\theta_1] \) with \( O(D^{-1}) \) variance. We can keep bouncing back and forth between gradient (or equivalently weights) and training loss for any number of steps \( T \) which does not scale with \( D \). Plugging this final \( \theta_T \) into the population loss and taking the expectation over draws of the training set, \( L(D) = E_D[L_{\text{pop}}[\theta_T]] \). If \( L_{\text{pop}} \) is also sufficiently smooth, this yields \( L(D) = \lim_{D \to \infty} L(D) = O(D^{-1}) \).

Here we have used the term sufficiently smooth. A sufficient set of criteria are given in Theorem 1; however, this is likely too restrictive. Indeed, any set of train and population loss for which a Taylor series (or asymptomatic series with optimal truncation) give an \( O(D^{-1}) \) error around the training points \( E_D[\theta_{t=0...T}] \) will have this behavior.

Local minimum. The above intuition relied on training for a number of steps that was fixed as \( D \) is taken large. Here we present some alternative intuition for the variance-limited scaling at late times, as training approaches a local minimum in the loss. For simplicity we discuss a one-dimensional loss.

Consider a local minimum, \( \theta^* \), of the population loss. As \( D \) is taken large, with high probability, the training loss will have a local minimum, \( \theta^* \), such that \( |\theta^* - \theta| = O(D^{-1}) \). One way to see this is to note that for a generic local minimum the first derivative changes sign, i.e. we can find \( \theta_1, \theta_2 \) such that \( \theta_1 < \theta^* < \theta_2 \) and either \( L_{\text{pop}}[\theta_1] < 0, L_{\text{pop}}[\theta_2] > 0 \) or \( L_{\text{pop}}[\theta_1] < 0, L_{\text{pop}}[\theta_2] > 0 \). To be concrete let’s focus on the first case (the argument will be identical in either case). As \( D \) becomes large, the probability that the training loss at \( \theta_1 \) and \( \theta_2 \) differs significantly from the population loss approaches zero. This can be seen from Markov’s inequality, where, \( P \left( \left| L_{\text{train}}[\theta] - L_{\text{pop}}[\theta] \right| > a \right) \leq \frac{\text{Var}(L_{\text{train}}[\theta])}{a^2} \), or more dramatically from Hoeffding’s inequality (assuming bounded \( L_{\text{train}} - L_{\text{pop}} \) lying in an interval of size \( I \))

\[
P \left( \left| L_{\text{train}}[\theta] - L_{\text{pop}}[\theta] \right| > a \right) \leq 2e^{-\frac{2a^2}{I^2}}.
\]

Here to have non-vanishing probability as we take \( D \) large, \( L_{\text{train}}[\theta_1] \) and \( L_{\text{train}}[\theta_2] \) must be closer than \( O(D^{-1}) \). If \( \theta_1 \) and \( \theta_2 \) are taken to be \( O(D^0) \), then \( L_{\text{train}} \) must change sign, indicating an extremum of \( L_{\text{train}} \); however, we can do even better. If we assume \( L_{\text{train}} \) is Lipshitz about \( \theta^* \) then we can still ensure a sign change even if \( |\theta_1 - \theta^*|, |\theta_2 - \theta^*| = O(D^{-1}) \). Using concentration of \( L_{\text{train}}[\theta] \) ensures the extremum is a local minimum. For non-generic minimum (e.g. vanishing first derivatives) we can apply the same arguments to higher order derivatives (assuming they exist) of \( L_{\text{pop}} \). Thus for a local minimum of \( L_{\text{pop}} \), with high probability \( L_{\text{train}} \) will have a corresponding minimum within a distance \( O(D^{-1}) \).

If we now consider an initialization \( \theta_0 \) and training procedure such that training converges to the local minimum of the training loss, \( \theta^* \), and that the population loss is sufficiently smooth about \( \theta^* \) (e.g. Lipshitz), then \( E_D[L_{\text{train}}[\theta^*] - L_{\text{pop}}[\theta^*]] = E_D[L_{\text{train}}[\theta^*] - L_{\text{pop}}[\theta^*]] + E_D[L_{\text{pop}}[\theta^*] - L_{\text{pop}}[\theta^*]] \). The first term vanishes, while the second is \( O(D^{-1}) \). If we further assume that this happens on average over choices of \( \theta_0 \) then we expect \( L(D) = \lim_{D \to \infty} L(D) = O(D^{-1}) \).

Stochastic gradient descent (SGD). At first blush it may be surprising that the variance-limited scaling holds even for mini-batch training. Indeed in this case, there is batch noise that comes in at a much higher scale than any variance due to the finite training set size. Indeed, the effect of mini-batching changes the final test loss, however if we fix the SGD procedure or average over SGD seeds, as we take \( D \) large, we can still ask how the training loss for a model trained under SGD on a training set of size \( D \) differs from that for a model trained under SGD on an infinite training set.

To see this, we first consider averaging over minibatches of size \( B \), but where points are drawn i.i.d. with replacement. If we denote the batch at step \( t \) by \( B_t \) and the average over independent draws of this batch by \( E_B[\bullet] \), then note we can translate moments with respect to batch draws with empirical averages over the entire training set. Explicitly, consider \( c_a \) and \( d_a \) potentially correlated, but each drawn i.i.d. within a batch. We have that,

\[
E_B \left[ \frac{1}{B} \sum_{a \in B_t} c_a \right] = \frac{1}{B} \sum_{a=1}^{D} c_a \\
E_B \left[ \left( \frac{1}{B} \sum_{a \in B_t} c_a \right) \left( \frac{1}{B} \sum_{a' \in B_t} d_{a'} \right) \right] = \left( 1 - \frac{1}{B} \right) \left( \frac{1}{B} \sum_{a=1}^{D} c_a \right) \left( \frac{1}{B} \sum_{a'=1}^{D} d_{a'} \right) + \frac{1}{B} \sum_{a=1}^{D} c_a d_a.
\]

This procedure means, after taking an average over draws of SGD batch, rather than thinking about a function of mini-batch averages, we can equivalently consider a modified function, with explicit dependence on the batch size, but that is only a function of empirical means over the training set. We can thus recycle the above intuition for the scaling of smooth functions of empirical means.

The above relied on independently drawing every sample from every batch. At the other extreme, we can consider drawing batches without shuffling and increasing training set size by \( B \) datapoints at a time, so as to keep the initial set of batches in
an epoch fixed. In this case, the first deviation in training between a dataset of size \( D \) and one of size \( D + B \) happens at the last batch in the first epoch after processing \( D \) datapoints.

As an extreme example, consider the case where \( D > BT \). In this case, as we only take \( T \) steps, the loss is constant for all \( D > BT \) and so \( \lim_{D \to \infty} \mathbb{L}(D; T; B) = \mathbb{L}(BT; T; B) \) and thus \( \mathbb{L}(D > BT) - \lim_{D \to \infty} \mathbb{L}(D) = 0 \) (and in particular is trivially \( \mathcal{O}\left(D^{-1}\right)\)).

### B. Polynomial loss

Before discussing neural network training, we review the concentration behavior of polynomials of sample means.

**Lemma 1** Let \( \bar{c}^{(i)} = \frac{1}{D} \sum_{a=1}^{D} c^{(i)}_a \) for \( i = 0, \ldots, J \) be empirical means, over \( D \) i.i.d. draws of \( c^{(i)}_a \) and let \( \bar{e}^{(i)} \) denote the distributional mean. Further, let \( X = (\bar{e}^{(0)}k_0 \bar{e}^{(1)}k_1 \cdots \bar{e}^{(J)}k_J) \) be a monomial in the sample means. Then \( X \) concentrates with moments \( \mathcal{O}\left(D^{-1}\right) \).

\[
\mathbb{E}_D \left[ (X - (\bar{e}^{(0)}k_0 \bar{e}^{(1)}k_1 \cdots \bar{e}^{(J)}k_J))^n \right] = \mathcal{O}\left(D^{-1}\right). \tag{S8}
\]

Here, \( \mathbb{E}_D [\bullet] \) denotes the average over independent draws of \( D \) samples.

To establish this we can proceed by direct computation.

\[
\mathbb{E}_D \left[ \left( X - (\bar{e}^{(0)}k_0 \bar{e}^{(1)}k_1 \cdots \bar{e}^{(J)}k_J) \right)^n \right] = \sum_{p=0}^{n} (-1)^{n-p} \binom{n}{p} \mathbb{E}_D \left[ X^p \right] \left( (\bar{e}^{(0)}k_0 \bar{e}^{(1)}k_1 \cdots \bar{e}^{(J)}k_J) \right)^{n-p}. \tag{S9}
\]

Each term in the sum can be computed using

\[
\mathbb{E}_D \left[ X^p \right] = \mathbb{E}_D \left[ (\bar{e}^{(0)}k_0 \bar{e}^{(1)}k_1 \cdots \bar{e}^{(J)}k_J)^p \right] = \frac{1}{D^p \sum_{i=0}^{k_1} \sum_{a_{(i)}^J} \mathbb{E}_D \left[ \left( \sum_{i=0}^{k_1} c^{(0)}a_{(i)}^J \right)^p \left( \sum_{i=0}^{k_1} c^{(1)}a_{(i)}^J \right)^p \cdots \left( \sum_{i=0}^{k_1} c^{(J)}a_{(i)}^J \right)^p \right] \right]
\]

\[
\mathbb{E}_D \left[ \left( \sum_{i=0}^{k_1} c^{(0)}a_{(i)}^J \right)^p \left( \sum_{i=0}^{k_1} c^{(1)}a_{(i)}^J \right)^p \cdots \left( \sum_{i=0}^{k_1} c^{(J)}a_{(i)}^J \right)^p \right] = \frac{D(D-1)\cdots(D-(p\sum_{i=0}^{k_1} k_i - 1))}{D^p \sum_{i=0}^{k_1} k_i} (\bar{e}^{(0)}k_0 \bar{e}^{(1)}k_1 \cdots \bar{e}^{(J)}k_J)^p + \mathcal{O}\left(D^{-1}\right).
\]

Plugging this into Eq. (S9) establishes the lemma. In the above, we use the multi-index notation \( \{a_{(i)}^J\} \) for the collection of indices on the \( e \) and the notation \( \{a_{(i)}^J \neq a_{(i)}^J\} \) for the subset of terms in the sum where all indices take different values.

Lemma 1 immediately implies that the mean of polynomials of \( \bar{e}^{(i)} \) concentrate around their infinite data limit.

\[
\mathbb{E}_D \left[ \left( g(\bar{e}^{(0)}, \bar{e}^{(1)}, \ldots, \bar{e}^{(K)}) - g(\bar{e}^{(0)}, \bar{e}^{(1)}, \ldots, \bar{e}^{(K)}) \right)^n \right] = \mathcal{O}\left(D^{-1}\right). \tag{S10}
\]

for \( g \in \mathbb{P}_K \left[ \bar{e}^{(0)}, \bar{e}^{(1)}, \ldots, \bar{e}^{(K)} \right] \).

With this out of the way, we can proceed to analyzing the scaling of trained neural networks. Here we consider the simplified setting where the network map, \( f \), and loss \( \ell \) evaluated on each training example, \( x_a = (x_a, y_a) \), are polynomial of degree \( J \) and \( K \) in the weights, \( \theta_j \).

\[
f(x) = \sum_{j=1}^{J} \beta_{j \mu_1 \mu_2}^{(i)}(x) \theta_{\mu_1} \theta_{\mu_2} \cdots \theta_{\mu_j}, \quad \ell(x_a) = \sum_{i=1}^{K} c^{(i)}_{\mu_1 \mu_2 \cdots \mu_i}(x_a) \theta_{\mu_1} \theta_{\mu_2} \cdots \theta_{\mu_i}. \tag{S11}
\]

The training loss can then be written as,

\[
L_{\text{train}} = \sum_{i=1}^{K} c^{(i)}_{\mu_1 \mu_2 \cdots \mu_i} \theta_{\mu_1} \theta_{\mu_2} \cdots \theta_{\mu_i}, \quad \bar{c}^{(i)} = \frac{1}{D} \sum_{a=1}^{D} c^{(i)}(x_a). \tag{S12}
\]

Here we have used the convention that the repeated weight indices \( \mu_j \) are summed over.
Gradient descent. As a result of the gradient descent weight update, \( \theta_{t+1} = \theta_t - \eta \frac{\partial L_{\text{train}}}{\partial \theta} \), the weights at time \( T \) are a polynomial of degree \((K - 1)^T\) in the \( \epsilon^{(i)} \).

\[
\theta_T \in P_{(K-1)^T} \left[ \epsilon^{(0)}, \epsilon^{(1)}, \ldots, \epsilon^{(K)} \right].
\]

[S13]

The coefficients of this polynomial depend on the initial weights, \( \theta_0 \). Plugging these weights back into the network output, we have that the network function at time \( T \) is again a polynomial in \( \epsilon^{(i)} \), now with degree \( J (K - 1)^T \).

\[
f_T(x) \in P_{J(K-1)^T} \left[ \epsilon^{(0)}, \epsilon^{(1)}, \ldots, \epsilon^{(K)} \right].
\]

[S14]

Thus, again using Lemma 1, \( f_T \) concentrates with variance \( \mathcal{O}(D^{-1}) \).

[S15]

and by Theorem 1 the loss will obey they variance-limited scaling.

Stochastic gradient descent. We now consider the same setup of polynomial loss, but now trained via stochastic gradient descent (SGD). We consider SGD batches drawn i.i.d. with replacement and are interested in the test loss averaged over SGD draws, with fixed batch size, \( B \).

We proceed by proving the following lemma, which allows us to reuse a similar argument to the GD case.

Lemma 2 Let \( \tilde{c}^{(i)} = \frac{1}{B} \sum_{a \in B_t} c_a^{(i)} \) for \( i = 0 \ldots J \) be mini-batch averages, over \( B \) i.i.d. draws of \( c_a^{(i)} \). Further, let \( X = \langle \tilde{c}^{(0)}, \tilde{c}^{(1)}, \ldots, \tilde{c}^{(J+1)} \rangle \) be a monomial in the mini-batch means. Then \( E_B[X] \in P_{\sum_j \tilde{a}^{(j)}} \left[ \tilde{d}^{(0)}, \tilde{d}^{(1)}, \ldots, \tilde{d}^{(J+1)} \right] \), where \( \tilde{d}^{(i)} \) are empirical means over the full training set of i.i.d. random variables as in Lemma 1 and \( E_B [\cdot] \) denotes the expectation over draws of SGD batches of size \( B \).

Expectations over draws of batches at different time steps are independent. Thus, w.l.o.g., we can consider \( t := t_0 = t_1 = \ldots = t_J \). We can again proceed by direct computation, expanding the mini-batch sums,

\[
E_B[X] = \frac{1}{B^{\sum_j \tilde{a}^{(j)}}} \sum_{\tilde{a}^{(i)} \in B_t} \left[ (c_a^{(0)})_{a_1} \cdots (c_a^{(0)})_{a_{k_0}} (c_a^{(1)})_{a_1} \cdots (c_a^{(1)})_{a_{k_1}} \cdots (c_a^{(J)})_{a_1} \cdots (c_a^{(J)})_{a_{k_J}} \right].
\]

[S16]

To proceed, we must keep track of terms in the sum where the \( a_i \) take the same or different values. If all \( a_i \) are different, the expectation over batch draws fully factorizes. More generally Eq. (S16) can be decomposed as a sum over products.

One way of keeping track of the index combinatorics is to introduce a set of graphs, \( \Gamma \), where each graph \( \gamma \in \Gamma \) has \( k_0 \) vertices of type 0, \( k_1 \) vertices of type 1, \ldots., and \( k_J \) vertices of type \( J \) (one vertex for each \( a_i \) index). Any pair of vertices may have zero or one edge between them. For any set of three vertices, \( v_1, v_2, \) and \( v_3 \) with edges (\( v_1, v_2 \)) and (\( v_2, v_3 \)) there must also be an edge (\( v_1, v_3 \)). The set \( \Gamma \) consists of all possible ways of connecting these vertices consistent with these rules.

For each graph, \( \gamma \), we denote connected components by \( \sigma \) and denote the number of vertices of type \( i \) within the connected component \( \sigma \) by \( m^{(i)}_{\sigma} \). With this we can write the sum, Eq. (S16) as

\[
E_B[X] = \sum_{\gamma \in \Gamma} S_{\gamma}(B) \prod_{\sigma \in \gamma} \left[ \frac{1}{B} \sum_{a \in B_t} \left( c_a^{(0)} \right)_{m^{(0)}_{\sigma}} \left( c_a^{(1)} \right)_{m^{(1)}_{\sigma}} \cdots \left( c_a^{(J)} \right)_{m^{(J)}_{\sigma}} \right].
\]

[S17]

Here \( S_{\gamma}(B) \) is a combinatoric factor associated to each graph, not relevant for the argument. The \( m^{(i)}_{\sigma} \) take on values 0 to \( k_i \), so the multi-index can take on \( \prod_{i=1}^{J} (k_i + 1) \) different values, which we re-index to \( \tilde{d}^{(0)}, \tilde{d}^{(1)}, \ldots, \tilde{d}^{(J+1)} \). Meanwhile, the degree of Eq. (S17) in \( \tilde{d}^{(i)} \) is bounded by the number of total vertices in each graph, i.e. \( \sum_{i=0}^{J} k_i \). This establishes the lemma. For a polynomial loss of degree \( K \), the mini-batch training loss at each time step takes the form

\[
L^{(t)}_{\text{train}} = \sum_{i=1}^{K} \epsilon^{(i)}_{\mu_1 \mu_2 \ldots \mu_i} \theta_{\mu_1} \theta_{\mu_2} \ldots \theta_{\mu_i}, \quad \epsilon^{(i)} \sim \frac{1}{B} \sum_{a \in B_t} \epsilon^{(i)}(x_a).
\]

[S18]

The update rule, \( \theta_{t+1} = \theta_t - \eta \frac{\partial L^{(t+1)}_{\text{train}}}{\partial \theta} \), ensures that \( \theta_T \) is a polynomial of degree \((K - 1)^T\) in the \( \epsilon^{(0)}, \epsilon^{(1)}, \ldots, \epsilon^{(K)} \)

\[
\theta_T \in P_{(K-1)^T} \left[ \epsilon^{(0)}, \epsilon^{(1)}, \ldots, \epsilon^{(0)}, \epsilon^{(1)}, \ldots, \epsilon^{(1)}, \ldots, \epsilon^{(1)}, \ldots, \epsilon^{(K)}, \ldots, \epsilon^{(K)} \right],
\]

[S19]
and consequently, denoting the test loss evaluated at $\theta_T$ by $L[\theta_T]$,

$$L[\theta_T] \in P_{K(K-1)^T} \left[ c^{(0,0)}, c^{(0,1)}, \ldots, c^{(0,T)}, c^{(1,0)}, c^{(1,1)}, \ldots, c^{(1,T)}, \ldots, c^{(K,0)}, c^{(K,1)}, \ldots, c^{(K,T)} \right].$$  \[S20\]

Using Lemma 2, the expectation of $L[\theta_T]$ over draws of SGD batches is given by

$$\mathbb{E}_B [L[\theta_T]] \in P_{K(K-1)^T} \left[ d^{(0)}, \ldots, d^{(K(K-1)^T K)} \right].$$  \[S21\]

Finally, denoting $\mathbb{E}_D [\mathbb{E}_B [L[\theta_T]]]$ by $L(D; B)$ and applying Lemma 1 gives

$$L(D; B) - \lim_{D \to \infty} L(D; B) = \mathcal{O} \left( D^{-1} \right).$$  \[S22\]

### C. Non-smooth examples

Here we present two worked examples where non-bounded or non-smooth loss leads to violations of the variance dominated scaling. In example one, the system obeys the variance dominated scaling at early times, but exhibits different behavior for times larger than the dataset size. In the second example, the system violates the variance dominated scaling even for two gradient descent steps, as a result of an unbounded derivative in the loss.

**Example 1 – unbounded loss at late times.** Consider a dataset with two varieties of data points, drawn with probabilities $\alpha$ and $1 - \alpha$, and one-dimensional quadratic losses, $\ell_1$ (concave up) and $\ell_2$ (concave down), on these two varieties.

$$\ell_1(\theta) = \frac{1}{2} \theta^2, \quad \ell_2(\theta) = -\frac{1}{2} \theta^2.$$  \[S23\]

If, in a slight abuse of notation, we further denote the training loss on a sample with $n_1$ points of type 1 and $D - n_1$ points of type two by $\ell_{n_1}$ and the population loss at a given value of the weight by $L_{\text{pop}}$, we have

$$\ell_{n_1} = \left( \frac{n_1}{D} - \frac{1}{2} \right) \theta^2, \quad L_{\text{pop}} = \left( \alpha - \frac{1}{2} \right) \theta^2.$$  \[S24\]

For this example we take $\alpha > 1/2$. In this case, the minimum of the population loss is at zero, while the minimum of the training loss can be at zero or at $\pm \infty$ depending on whether the training sample has $n_1$ greater than or less than $D/2$. We can thus create a situation where at late training times, $\theta_T$ does not concentrate around the minimum of the population loss.

As we work through this example explicitly, we will see the following. (i) A mismatch larger than $\mathcal{O}(D^{-1})$ between the population minimum and the minimum found by training on a sample set of size $D$ requires times $T$ larger than a constant multiple of $D$. (ii) The quantity we study throughout this work is the difference between the infinite data limit of the test loss and the finite data value, $L(D) - \lim_{D \to \infty} L(D)$. The minimum of the infinite data limit of the test loss is not the same as the minimum of the population loss, $\min_{D \to \infty} L(D) \neq \min_{D \to \infty} L_{\text{pop}}$. In this example one diverges, while the other is finite. In particular this example evades the scaling result by $L(D)$ for times larger than $D$ having a diverging limit.

Explicitly, we study the evolution of the model under gradient flow.

$$\dot{\theta} = -2 \left( \frac{n_1}{D} - \frac{1}{2} \right) \theta, \quad \theta_T = e^{-2(\frac{n_1}{D} - \frac{1}{2})T} \theta_0.$$  \[S25\]

The test loss averaged over draws of the dataset is given by

$$L(D; T) = \mathbb{E}_{n_1} \left[ \left( \alpha - \frac{1}{2} \right) \theta_T^2 \right] = e^{2T} \left( \alpha - \frac{1}{2} \right) \left( 1 - \alpha \left( 1 - e^{-\frac{4T}{D}} \right) \right)^D \theta_0^2$$  \[S26\]

If we consider this loss at large $D$ and fixed $T$ we get

$$L(D; T) = e^{-4(\alpha - \frac{1}{2})T} \left( \alpha - \frac{1}{2} \right) \theta_0^2 \left( 1 + \frac{8T^2 \alpha (1 - \alpha) D}{\alpha} + \mathcal{O} \left( D^{-2} \right) \right),$$  \[S27\]

and thus $L(D; T) - \lim_{D \to \infty} L(D; T) = \mathcal{O} \left( D^{-1} \right)$ as expected.

If on the other hand we consider taking $T \gg D$ we have

$$L(D; T \gg D) = e^{2T} \left( \alpha - \frac{1}{2} \right) \left( 1 - \alpha \right)^D \theta_0^2,$$  \[S28\]

the limit $\lim_{D \to \infty} L(D; T \gg D)$ diverges.

Lastly, we note that if we take $T = \beta D$ with $\beta < \left| \log(1 - \alpha) \right|/2$ we can approach the large $D$ limit with non-generic, tuneable exponential convergence.
Example 2 – unbounded derivative. Again, consider a two variety setup, this case with equal probabilities and per sample losses,

\[ \ell_1(\theta) = \frac{1}{2} \theta^2 + \frac{1}{2} |\theta|^\alpha, \quad \ell_2(\theta) = \frac{1}{2} \theta^2 - \frac{1}{2} |\theta|^\alpha. \]  

We will consider different values of \( \alpha > 0 \). The train loss and population loss are then,

\[ \ell_{n_1} = \frac{1}{2} \theta_1^2 + \frac{1}{\alpha} \left( \frac{n_1}{D} - \frac{1}{2} \right) |\theta|^{\alpha}, \quad L_{\text{pop}} = \frac{1}{2} \theta^2. \]  

We consider a model initialized to \( \theta_0 = 1 \) and trained for two steps of gradient descent with learning rate 1.

\[ g_t = \theta_t + \left( \frac{n_1}{D} - \frac{1}{2} \right) |\theta_0|^{\alpha-2}, \quad \theta_{t+1} = \theta_t - g_t. \]  

Two update steps gives

\[ \theta_2 = \left| \frac{n_1}{D} - \frac{1}{2} \right|^{\alpha}. \]  

The test loss is given by the population loss evaluated at \( \theta_2 \) averaged over test set draws.

\[ L(D) = E_{n_1} \left[ \frac{1}{2} \theta^2 \right] = \frac{1}{2^{D-1}} \sum_{n_1=0}^{D} \left( \frac{D}{n_1} \right) \left| \frac{n_1}{D} - \frac{1}{2} \right|^{2\alpha} \]

\[ = \sqrt{\frac{D}{2\pi}} \int_{-\infty}^{\infty} e^{-2D(x - \frac{1}{2})^2} x - \frac{1}{2} \frac{\alpha}{\alpha + \frac{1}{2}} + O \left( D^{-1} \right) \]

\[ = \frac{\Gamma \left( \alpha + \frac{1}{2} \right)}{2^{1+\alpha} \sqrt{\pi}} D^{-\alpha} + O \left( D^{-1} \right). \]

4. Proof of Theorems 2 and 3

In this section we detail the proof of Theorems 2 and 3. The key observation is to make use of the fact that nearest neighbor distances for \( D \) points sampled i.i.d. from a \( d \)-dimensional manifold have mean \( E_{D,x} \| x - \hat{x} \| = O \left( D^{-1/d} \right) \), where \( \hat{x} \) is the nearest neighbor of \( x \) and the expectation is the mean over data-points and draws of the dataset see e.g. (14).

The theorem statements are copied for convenience. In the main text, in an abuse of notation, we used \( L(f) \) to indicate the value of the test loss as a function of the network \( f \), and \( L(D) \) to indicate the test loss averaged over the population, draws of the dataset, model initializations and training. To be more explicit below, we will use the notation \( \ell(f(x)) \) to indicate the test loss for a single network evaluated at single test point.

Theorem 2 Let \( \ell(f) \), \( f \) and \( \mathcal{F} \) be Lipschitz with constants \( K_L, K_f, \) and \( K_{\mathcal{F}} \) and \( \ell(\mathcal{F}) = 0 \). Further let \( \mathcal{D} \) be a training dataset of size \( D \) sampled i.i.d from \( \mathcal{M}_d \) and let \( f(x) = \mathcal{F}(x) \forall x \in \mathcal{D} \), then \( L(D) = O \left( K_L \max(K_f, K_{\mathcal{F}}) D^{-1/d} \right) \).

Consider a network trained on a particular draw of the training data. For each training point, \( x \), let \( \hat{x} \) denote the neighboring training data point. Then by the above Lipschitz assumptions and the vanishing of the loss on the true target, we have

\[ \ell(f(x)) \leq K_L \| f(x) - \mathcal{F}(x) \| \leq K_L (K_f + K_{\mathcal{F}}) \| x - \hat{x} \|. \]  

With this, the average test loss is bounded as

\[ L(D) \leq K_L (K_f + K_{\mathcal{F}}) E_{D,x} \| x - \hat{x} \| = O \left( K_L \max(K_f, K_{\mathcal{F}}) D^{-1/d} \right). \]  

In the last equality, we used the above mentioned scaling of nearest neighbor distances.

Theorem 3 Let \( \ell(f) \), \( f \) and \( \mathcal{F} \) be Lipschitz with constants \( K_L, K_f, \) and \( K_{\mathcal{F}}. \) Further let \( f(x) = \mathcal{F}(x) \) for \( P \) points sampled i.i.d from \( \mathcal{M}_d \) then \( L(P) = O \left( K_L \max(K_f, K_{\mathcal{F}}) P^{-1/d} \right). \)

Denote by \( \mathcal{P} \) the \( P \) points, \( z \), for which \( f(z) = \mathcal{F}(z) \). For each test point \( x \) let \( \hat{x} \) denote the closest point in \( \mathcal{P} \), \( \hat{x} = \arg\min_{z} (\| x - z \|) \). Adopting this notation, the result follows by the same argument as Theorem 2.
5. Random feature models

Here we present random feature models in more detail. We begin by reviewing exact expressions for the loss. We then go onto derive its asymptotic properties. We again consider training a model \( f(x) = \sum_{\mu=1}^{P} \theta_{\mu} f_{\mu}(x) \), where \( f_{\mu} \) are drawn from some larger pool of features, \( \{F_{M}\} \), \( f_{\mu}(x) = \sum_{M=1}^{S} P_{M} F_{M}(x). \)

Note, if \( \{F_{M}(x)\} \) form a complete set of functions over the data distribution, then any target function, \( y(x) \), can be expressed as \( y = \sum_{M=1}^{S} \omega_{M} F_{M}(x) \). The extra constraint in a teacher-student model is specifying the distribution of the \( \omega_{M} \). The variance-limited scaling goes through with or without the teacher-student assumption; however it is crucial for analyzing the resolution-limited behavior.

As in section 2.3 of the main text, we consider models with weights initialized to zero and trained to convergence with mean squared error loss.

\[
L_{\text{train}} = \frac{1}{2D} \sum_{a=1}^{D} (f(x_a) - y_a)^2 .
\]  

[S35]

The data and feature second moments play a central role in our analysis. We introduce the notation,

\[
\bar{C} = \mathbb{E}_x \left[ F(x) F^T (x) \right] , \quad \bar{C} = \frac{1}{D} \sum_{a=1}^{D} F(x_a) F^T (x_a) , \quad C = \mathcal{P} \bar{C} \mathcal{P}^T , \quad \bar{C} = \mathcal{P} \bar{C} \mathcal{P}^T .
\]

[S36]

\[
K(x,x') = \frac{1}{S} \sum_{a=1}^{D} F(x_a) F(x'_a), \quad \bar{K} = \bar{K} \big|_{D_{\text{train}}} , \quad K(x,x') = \frac{1}{P} \sum_{a} f(x_a) f(x'_a), \quad \bar{K} = \bar{K} \big|_{D_{\text{train}}} .
\]

Here the script notation indicates the full feature space while the block letters are restricted to the student features. The bar represents restriction to the training dataset. We will also indicate kernels with one index in the training set as \( \bar{K}(x) := K(x,x_{a=1,..D}) \) and \( \bar{K}(x) := K(x,x_{a=1,..D}) \). After this notation spree, the test loss can be written for under-parameterized models \( P \leq D \) as

\[
L(D,P) = \frac{1}{2S} \mathbb{E}_D \left[ \text{Tr} \left( \mathcal{C} + \bar{C} \mathcal{P} \mathcal{P}^T \bar{C} \mathcal{P} \mathcal{P}^T \mathcal{C} - 2 \bar{C} \mathcal{P} \mathcal{P}^T \mathcal{C} \right) \right] ,
\]

[S37]

and for over-parameterized models (at the unique minimum found by GD, SGD, or projected Newton’s method),

\[
L(D,P) = \frac{1}{2} \mathbb{E}_{x,D} \left[ \mathcal{K}(x,x) + \bar{K}(x)^T \bar{K}^{-1} \bar{K} \right] .
\]

[S38]

Here the expectation \( \mathbb{E}_D [\bar{K}] \) is an expectation with respect to i.i.d. draws of a dataset of size \( D \) from the input distribution, while \( \mathbb{E}_x [\mathcal{K}] \) is an ordinary expectation over the input distribution. Note, expression Eq. (S37) is also valid for over-parameterized models and Eq. (S38) is valid for under-parameterized models if the inverses are replaces with the Moore-Penrose pseudo-inverse. Also note, the two expressions can be related by exchanging the projections onto finite features with the projection onto the training dataset and the sums of teacher features with the expectation over the data manifold. This realizes the duality between dataset and features discussed above.

A. Asymptotic expressions. We are interested in Eq. (S37) and Eq. (S38) in the limits of large \( P \) and \( D \).

Variance-limited scaling. We begin with the under-parameterized case. In the limit of lots of data, the sample estimate of the feature-feature second moment matrix, \( \mathcal{C} \), approaches the true second moment matrix, \( \bar{C} \). Explicitly, if we define the difference \( \delta \mathcal{C} \) by \( \bar{C} = \mathcal{C} + \delta \mathcal{C} \), we have

\[
\mathbb{E}_D [\delta \mathcal{C}] = 0
\]

\[
\mathbb{E}_D [\delta \mathcal{C}_{M_1 N_1} \delta \mathcal{C}_{M_2 N_2}] = \frac{1}{D} \mathbb{E}_x \left[ F_{M_1}(x) F_{N_1}(x) F_{M_2}(x) F_{N_2}(x) \right] - \mathcal{C}_{M_1 N_1} \mathcal{C}_{M_2 N_2} \]

[S39]

\[
\mathbb{E}_{D} [\delta \mathcal{C}_{M_1 N_1} \cdots \delta \mathcal{C}_{M_n N_n}] = \mathcal{O} \left( D^{-2} \right) \forall n > 2 .
\]

The key takeaway from Eq. (S39) is that the dependence on \( D \) is manifest.

Using these expressions in Eq. (S37) yields

\[
L(D,P) = \frac{1}{2S} \text{Tr} \left( \mathcal{C} - \mathcal{C} \mathcal{P} \mathcal{P}^T \mathcal{C} \mathcal{P} \mathcal{P}^T \mathcal{C} \right) + \frac{1}{2DS} \sum_{M_1 \geq N_1, M_2 > N_2} T_{M_1 N_1 M_2 N_2} \left[ \delta \mathcal{C}_{M_1 M_2} (\mathcal{P} \mathcal{C} \mathcal{P}^T \mathcal{C})_{N_1 N_2}^{-1} + (\mathcal{C}^{-1} \mathcal{C}^{-1} \mathcal{P} \mathcal{P}^T \mathcal{C} \mathcal{C}^{-1})_{M_1 M_2} C_{N_1 N_2}^{-1} - 2 (\mathcal{C} \mathcal{P} \mathcal{P}^T \mathcal{C})_{M_1 M_2} (\mathcal{P} \mathcal{C} \mathcal{P}^T \mathcal{C})_{N_1 N_2}^{-1} \right] + \mathcal{O} \left( D^{-2} \right) .
\]

[S40]
Here we have introduced the notation, $T_{M_1N_1M_2N_2} = \mathbb{E}_x [F_{M_1}(x)F_{N_1}(x)F_{M_2}(x)F_{N_2}(x)]$.

As above, defining

$$L(P) := \lim_{D \to \infty} L(D, P) = \frac{1}{2S} \text{Tr} \left( C - CP^TC^{-1}PC \right),$$

we see that though $L(D, P) - L(P)$ is a somewhat cumbersome quantity to compute, involving the average of a quartic tensor over the data distribution, its dependence on $D$ is simple.

For the over-parameterized case, we can similarly expand Eq. (S38) using $K = \check{K} + \delta \check{K}$. With fluctuations satisfying,

$$\mathbb{E}_P [\delta \check{K}] = 0$$

$$\mathbb{E}_P [\delta \check{K}_{ab} \delta \check{K}_{a'b'_{b''}}] = \frac{1}{P} \left( \mathbb{E}_P [f_\mu(x_a)] f_\mu(x_{b_1}) f_\mu(x_{a_2}) f_\mu(x_{b_2}) - \check{K}_{ab} \check{K}_{a'b'b''} \right)$$

$$\mathbb{E}_P [\delta \check{K}_{a_1} \cdots \delta \check{K}_{a_n}] = \mathcal{O}(P^{-2}) \quad \forall n > 2.$$

This gives the expansion

$$L(D, P) = \frac{1}{2} \mathbb{E}_{x,C} [\check{K}(x,x) - \check{C}(x)^T \check{K}^{-1} \check{C}(x)] + \mathcal{O}(P^{-1})$$

and

$$L(D) = \frac{1}{2} \mathbb{E}_{x,C} [\check{K}(x,x) - \check{C}(x)^T \check{K}^{-1} \check{C}(x)].$$

Resolution-limited scaling. We now move onto studying the parameter scaling of $L(P)$ and dataset scaling of $L(D)$. We explicitly analyze the dataset scaling of $L(D)$, with the parameter scaling following via the dataset parameter duality.

Much work has been devoted to evaluating the expression Eq. (S44) (15–17). One approach is to use the replica trick—a tool originating in the study of disordered systems which computes the expectation of a logarithm of a random variable via simpler moment contributions and analyticity assumptions (18). The replica trick has a long history as a technique to study the generalization properties of kernel methods (19–25). We will most closely follow the work of (26), who use the replica method to derive an expression for the test loss of linear feature models in terms of the eigenvalues of the kernel $\check{C}$ and $\check{w}$, the coefficient vector of the target labels in terms of the model features.

$$L(D) = \kappa^2 \sum_i \frac{\lambda_i \omega_i^2}{(\kappa + D \lambda_i)^2}$$

$$\kappa = \frac{\sum_i \kappa \lambda_i}{\kappa + D \lambda_i},$$

$$\gamma = \frac{\sum_i \frac{D \lambda_i^2}{(\kappa + D \lambda_i)^2}}.$$  

This is the ridge-less, noise-free limit of equation (4) of (26). Here we analyze the asymptotic behavior of these expressions for eigenvalues satisfying a power-law decay, $\lambda_i = i^{-(1+\alpha \kappa)}$ and for targets coming from a teacher-student setup, $w \sim \mathcal{N}(0, 1/S)$.

To begin, we note that for teacher-student models in the limit of many features, the overlap coefficients $\check{w}$ are equal to the teacher weights, up to a rotation $\check{w} = O_{MM} w$. As we are choosing an isotropic Gaussian initialization, we are insensitive to this rotation and, in particular, $\mathbb{E}_w [\check{w}^2] = 1/S$. See Figure S8 for empirical support of the average constancy of $\check{w}$ for the teacher-student setting and contrast with realistic labels.

With this simplification, we now compute the asymptotic scaling of Eq. (S45) by approximating the sums with integrals and expanding the resulting expressions in large $D$. We use the identities:

$$\int_1^\infty dx \frac{x^{-n(1+\alpha)}}{(x + D x^{-1})^m} = \kappa^{-m} \Gamma \left( n - \frac{1}{1+\alpha} \right) \frac{\Gamma \left( n + \frac{\alpha}{1+\alpha} \right)}{\Gamma \left( n + \frac{\alpha}{1+\alpha} \right)} 2F_1 \left( m, n - \frac{1}{1+\alpha}, n + \frac{\alpha}{1+\alpha}, \frac{-D}{\kappa} \right)$$

$$2F_1(a, b, c, -y) \asymp y^{-a} + By^{-b} + \ldots.$$  

Here $2F_1$ is the hypergeometric function and the second line gives its asymptotic form at large $y$. $B$ is a constant which does not affect the asymptotic scaling.

Using these relations yields

$$\kappa \propto D^{-\alpha \kappa}, \quad \gamma \propto D^0, \quad \text{and} \quad L(D) \propto D^{-\alpha \kappa},$$

as promised. Here we have dropped sub-leading terms at large $D$. Scaling behavior for parameter scaling $L(P)$ follows via the data-parameter duality.

Duality beyond asymptotics. Expressions Eq. (S37) and Eq. (S38) are related by changing projections onto finite feature set and finite dataset even without taking any asymptotic limits. We thus expect the dependence of test loss on parameter count and dataset size to be related quite generally in linear feature models. See Section 6 for further details.
6. Learned features

In this section, we consider linear models with features coming from pretrained neural networks. Such features are useful for transfer learning applications (e.g. (27, 28)). In Figures S6 and S7, we take pretrained embedding features from an EfficientNet-B5 model (29) using TF hub*. The EfficientNet model is pretrained using the ImageNet dataset with input image size of (456, 456). To extract features for the (32, 32) CIFAR-10 images, we use bilinear resizing. We then train a linear classifier on top of the penultimate pretrained features. To explore the effect of feature size $P$ and dataset size $D$, we randomly subset the feature dimension and training dataset size and average over 5 random seeds. Prediction on test points are obtained as a kernel ridge regression problem with linear kernel. We note that the regularization ridge parameter can be mapped to an inverse early-stopping time (30, 31) of a corresponding ridgeless model trained via gradient descent. Inference with low regularization parameter denotes training for long time while tuned regularization parameter is equivalent to optimal early stopping.

In Figure S7 we see evidence of all four scaling regimes for low regularization (left four) and optimal regularization (right four). We speculate that the deviation from the predicted variance-limited exponent $\alpha_P = \alpha_D = 1$ for the case of fixed low regularization (late time) is possibly due to the double descent resonance at $D = P$, which interferes with the power law fit.

In Figure S6, we observe the duality between dataset size $D$ (solid) and feature size $P$ (dashed) – the loss as a function of the number of features is identical to the loss as a function of dataset size for both the optimal loss (tuned regularization) or late-time loss (low regularization).

In Figure S8, we also compare properties of random features (using the infinite-width limit) and learned features from trained Wide Resnet (WRN) 28-10 models. We note that teacher-student models, where the feature class matches the target function and ordinary, fully trained models on real data (Figure 1a) have significantly larger exponents than models with fixed features and realistic targets.

The measured $\bar{\omega}_i$ – the coefficient of the task labels under the $i$-th feature Eq. (S45) – are approximately constant as function of index $i$ for all teacher-student settings. However, for real targets, $\bar{\omega}_i$ are only constant for the well-performing Myrtle-10 and WRN trained features (last two columns).

*https://www.tensorflow.org/hub
Fig. S1. Alternate metrics and stopping conditions We find similar scaling behavior for both the loss and error, as well as for final and best (early stopped) metrics.
Fig. S2. This figure shows scaling trends of MSE loss with dataset size for teacher/student models. The exponents extracted from these fits and their associated input-space dimensionalities are shown in Figure 1b.
Table S1. CNN architectures for CIFAR-10, MNIST, Fashion MNIST (left), CIFAR-100 (center) and SVHN (right).

| Layer                        | Width |
|------------------------------|-------|
| CNN window (3, 3)            | 50    |
| 2D Max Pooling (2, 2)        |       |
| CNN window (3, 3)            | 100   |
| 2D Max Pooling (2, 2)        |       |
| CNN window (3, 3)            | 100   |
| Dense                        | 64    |
| Dense                        | 10    |

| Layer                        | Width |
|------------------------------|-------|
| CNN window (3, 3)            | 50    |
| 2D Max Pooling (2, 2)        |       |
| CNN window (3, 3)            | 100   |
| 2D Max Pooling (2, 2)        |       |
| CNN window (3, 3)            | 200   |
| Dense                        | 128   |
| Dense                        | 100   |

| Layer                        | Width |
|------------------------------|-------|
| CNN window (3, 3)            | 64    |
| 2D Max Pooling (2, 2)        |       |
| CNN window (3, 3)            | 64    |
| 2D Max Pooling (2, 2)        |       |
| CNN window (3, 3)            | 256   |
| Dense                        | 128   |
| Dense                        | 10    |
Fig. S3. This figure shows scaling trends of cross-entropy loss with dataset size for various image datasets. The exponents extracted from these fits and their associated input-space dimensionalities are shown in Figure 1b.
Fig. S4. This figure shows the variation of $\alpha_p$ with the input-space dimension. The exponent $\alpha_p$ is the scaling exponent of loss with model size for teacher-student setup.
Fig. S5. Effect of aspect ratio on dataset scaling. We find that for WRN-d-k trained on CIFAR-10, varying depth from 10 to 40 has a relatively mild effect on scaling behavior, while varying the width multiplier, $k$, from 1 to 12 has a more noticeable effect, up until a saturating width.
Fig. S6. **Duality between dataset size vs feature number in pretrained features** Using pretrained embedding features of EfficientNet-B5 (29) for different levels of regularization, we see that loss as function of dataset size or loss as a function of the feature dimension track each other both for small regularization (left) and for tuned regularization (right). Note that regularization strength with trained-feature kernels can be mapped to inverse training time (30, 31). Thus (left) corresponds to long training time and exhibits double descent behavior, while (right) corresponds to optimal early stopping.
Fig. S7. Four scaling regimes exhibited by pretrained embedding features. Using pretrained embedding features of EfficientNet-B5 (29) for fixed low regularization (left) and tuned regularization (right), we can identify four regimes of scaling using real CIFAR-10 labels.
Fig. S8. Loss on the teacher targets scale better than real targets for both untrained and trained features. The first three columns are infinite width kernels while the last column is a kernel built out of features from the penultimate layer of pretrained WRN 28-10 models on CIFAR-10. The first row is the loss as a function of dataset size D for teacher-student targets vs real targets. The observed dataset scaling exponent is denoted in the legend. The second row is the normalized partial sum of kernel eigenvalues. The partial sum's scaling exponent is measured to capture the effect of the finite dataset size when empirical $\alpha_K$ is close to zero. The third row shows $\omega_i$ for teacher-student and real target compared against the kernel eigenvalue decay. We see the teacher-student $\omega_i$ are approximately constant.
References

[S1] R Novak, et al., Neural Tangents: Fast and easy infinite neural networks in python in International Conference on Learning Representations. (2020).

[S2] J Bradbury, et al., JAX: composable transformations of Python+NumPy programs (2018).

[S3] V Shankar, et al., Neural kernels without tangents in International Conference on Machine Learning. (2020).

[S4] SS Schoenholz, J Gilmer, S Ganguli, J Sohl-Dickstein, Deep information propagation. International Conference on Learning Representations (2017).

[S5] J Lee, et al., Deep neural networks as Gaussian processes in International Conference on Learning Representations. (2018).

[S6] L Xiao, Y Bahri, J Sohl-Dickstein, S Schoenholz, J Pennington, Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks in International Conference on Machine Learning. (2018).

[S7] J Heek, et al., Flax: A neural network library and ecosystem for JAX (2020).

[S8] S Zagoruyko, N Komodakis, Wide residual networks in British Machine Vision Conference. (2016).

[S9] I Loshchilov, F Hutter, Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016).

[S10] A Lewkowycz, Y Bahri, E Dyer, J Sohl-Dickstein, G Gur-Ari, The large learning rate phase of deep learning: the catapult mechanism. arXiv preprint arXiv:2003.02218 (2020).

[S11] J Lee, et al., Wide neural networks of any depth evolve as linear models under gradient descent in Advances in Neural Information Processing Systems. (2019).

[S12] A Paszke, et al., Pytorch: An imperative style, high-performance deep learning library. Advances Neural Information Processing Systems 32, 8026–8037 (2019).

[S13] U Sharma, J Kaplan, Scaling laws from the data manifold dimension. Journal Machine Learning Research 23, 1–34 (2022).

[S14] E Levina, PJ Bickel, Maximum likelihood estimation of intrinsic dimension in Advances in neural information processing systems. pp. 777–784 (2005).

[S15] CK Williams, F Vivarelli, Upper and lower bounds on the learning curve for gaussian processes. Machine Learning 40, 77–102 (2000).

[S16] D Malzahn, M Opper, A variational approach to learning curves in Advances in Neural Information Processing Systems, eds. T Dietterich, S Becker, Z Ghahramani. (MIT Press), Vol. 14, pp. 463–469 (2002).

[S17] P Sollich, A Halees, Learning curves for gaussian process regression: Approximations and bounds. Neural computation 14, 1393–1428 (2002).

[S18] G Parisi, A sequence of approximated solutions to the SK model for spin glasses. Journal Physics A: Mathematical General 13, L115 (1980).

[S19] P Sollich, Learning curves for gaussian processes in Proceedings of the 11th International Conference on Neural Information Processing Systems. pp. 344–350 (1998).

[S20] D Malzahn, M Opper, Learning curves for gaussian processes regression: A framework for good approximations. Advances neural information processing systems pp. 273–279 (2001).

[S21] D Malzahn, M Opper, Learning curves and bootstrap estimates for inference with gaussian processes: A statistical mechanics study. Complexity 8, 57–63 (2003).

[S22] MJ Urry, P Sollich, Replica theory for learning curves for gaussian processes on random graphs. Journal Physics A: Mathematical Theoretical 45, 425005 (2012).

[S23] O Cohen, O Malka, Z Ringel, Learning curves for overparametrized deep neural networks: A field theory perspective. Phys. Rev. Res. 3, 023034 (2021).

[S24] F Gerace, B Loureiro, F Krzakala, M Mézard, L Zdeborová, Generalisation error in learning with random features and the hidden manifold model in International Conference on Machine Learning. (PMLR), pp. 3452–3462 (2020).

[S25] B Bordelon, A Canatar, C Pehlevan, Spectrum dependent learning curves in kernel regression and wide neural networks in International Conference on Machine Learning. (PMLR), pp. 1024–1034 (2020).

[S26] A Canatar, B Bordelon, C Pehlevan, Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. Nature communications 12, 1–12 (2021).

[S27] S Kornblith, J Shlens, QV Le, Do better imagenet models transfer better? in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2661–2671 (2019).

[S28] A Kolesnikov, et al., Big transfer (bit): General visual representation learning. arXiv preprint arXiv:1912.11370 6, 8 (2019).

[S29] M Tan, Q Le, Efficientnet: Rethinking model scaling for convolutional neural networks in International Conference on Machine Learning. (PMLR), pp. 6105–6114 (2019).

[S30] A Ali, JZ Kolter, RJ Tibshirani, A continuous-time view of early stopping for least squares regression in The 22nd International Conference on Artificial Intelligence and Statistics. pp. 1370–1378 (2019).

[S31] J Lee, et al., Finite versus infinite neural networks: an empirical study. Advances Neural Information Processing Systems 33 (2020).