A Experimental Details

A.1 MLPs

In Figure 8 we used a 3-layer MLP learning a Gegenbauer polynomial $Q_2(\boldsymbol{\beta}\cdot\boldsymbol{x})$ in $D = 5$ dimensions. Here $\boldsymbol{\beta}$ was a randomly chosen unit vector in $\mathbb{R}^D$. We implemented μP parameterization by hand. The output layer of the network was rescaled by $\alpha_0/\sqrt{N}$, consistent with μP. We chose $\alpha_0 = 1000$ to put us in the lazy regime. We set the the learning rate to be $5N/(1+\alpha_0^2)$. General arguments based on kernel scale indicate that the learning rate should be scaled as α_0^{-2} at large α.

In Figure 13 we used a 3-layer MLP learning a Gegenbauer polynomial $Q_2(\boldsymbol{\beta}\cdot\boldsymbol{x})$ in $D = 25$ dimensions. We set the learning rate to be nearly as high as possible before a loss explosion.

A.2 Vision

A.2.1 CIFAR-5m

All plots except Figure 5a and 5b: We trained with standard CIFAR data augmentation of random crop (RandomCrop(32, padding=4) in pytorch) and horizontal flip (RandomHorizontalFlip() in pytorch). As base network (for μP) we used ResNet18 where BatchNorm was replaced with LayerNorm (to maintain the consistency of the neural network between train and test). We used the SGD optimizer with learning rate of .05 with cosine decay over 20000 steps, .9 momentum and batch size of 250.

For Figure 5a, we used the above setup, but with a learning rate of 0.01 and a much higher batch size of 2000, so as to replicate the edge of stability phenomenon [45] which only occurs at high batch sizes. For Figure 5b, we used a learning rate of 0.3 and batch size of 32, so as to show the behavior of high learning rate and small batch size on train loss.

A.2.2 CIFAR-10 Multiple Passes

In Figure 10, we show the dynamics and representational consistency of ResNets trained on CIFAR-10 for several epochs. The architecture is a ResNet-18 with base-shape width set at $N = 64$ channels. The model is trained with SGD with learning rate 0.1 and cosine annealing schedule. The batch-size used is 128.

A.2.3 ImageNet

In all ImageNet experiments, we used a training subset of the ImageNet-1k dataset consisting of $2^{20} = 1048576$ labeled images and a test subset consisting of 1024 labeled images. Both subsets were randomly sampled from the full ImageNet-1k training and validation datasets, respectively. To extend the duration in training in which the network remains in the online regime beyond one epoch, we heavily augmented the images in the training dataset using PyTorch's `AutoAugment` transform with the default policy, `AutoAugmentPolicy.IMAGENET`.

We again used the ResNet-18 architecture with μP parameterization relative to the ResNet-18 network with base-shape width $N = 64$ channel [14]. All architectures and training procedures were implemented in Jax and used the auxiliary Flax and Optax packages, respectively.

Figures 2(b) and 6(b) were trained using the Adam optimizer with the following learning rate schedule: linear warm-up for 0.5 epochs from learning rate 8×10^{-5} to 8×10^{-3}, followed by cosine decay over 49.5 epochs to 8×10^{-5}.

A.3 Language

A.3.1 Wikitext-103 Language Modeling

For all Wikitext-103 tasks, we adopted the μP transformer as defined in the μP package [14]. In the plots shown in the main text, we used a depth-4 transformer, with $d_{model}, d_k, d_v = N$ and $d_{ffn} = 4N$. We performed a single pass through the train set in order to stay in the realistic online regime. We used a masked language modeling with sequence length S at varying input sequence lengths S. For Figure 1 d) we used the $S \times S$ attention matrix of an $S = 128$ transformer. In Figure 4 e) we used the attention matrix of an $S = 35$ transformer. We chose this different length simply to illustrate the consistent message across sequence lengths. We used a batch size of $B = 32$ for all experiments. The residual stream was thus a tensor of shape (S, B, d_{model}).

We used the Adam optimizer with a learning rate of 0.0001. We also ran the same configuration with SGD and a learning rate of 0.5 and observed the same behavior. See section B for further plots and details.

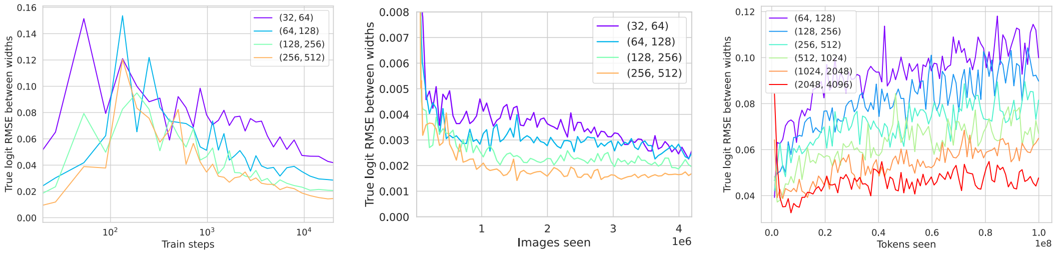
For figure 3, we used the Wikitext-103 validation set in order to measure the evolution of the predictions on masked logits. In 3 f), we averaged the mean squared error from the widest transformer by using 100 test points.

A.3.2 C4 Language Modelling

Figure 1 (b) we trained with base network being a 125m parameter transformer model on 2.5 billion tokens using the Mosaic ML's LLM codebase (`https://web.archive.org/web/20230519184343/https://github.com/mosaicml/examples/tree/main/examples/llm`). See `https://web.archive.org/web/20230519183813/https://github.com/mosaicml/examples/blob/main/examples/llm/yamls/mosaic_gpt/125m.yaml` for the full hyperparameter details. We were limited by time and computational resources in our ability to explore further details of the C4 transformer model.

B Further Plots of Convergence

In this section, we show additional figures illustrating convergence of network quantities across widths that we did not have space for in the main text.



(a) CIFAR-5m (b) Imagenet (c) Wikitext-103

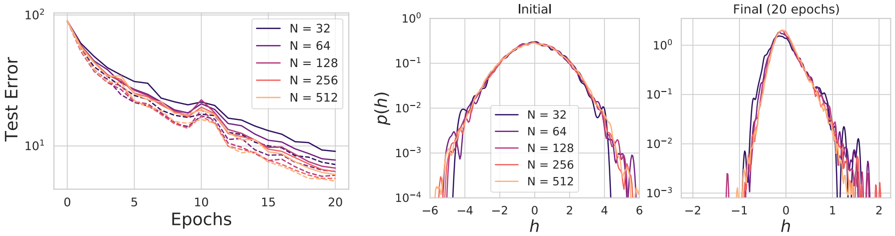
Figure 9: Analog of Figure 3 but comparing networks of successive widths rather than comparing all networks to the widest. Again, we see that as the network width grows, the difference between successive networks shrinks.

B.1 Vision

In Figure 9 a), b) we plot the analogue of the first two columns of Figure 3, but now instead of computing logit RMSE to the widest network, we compute it between networks of successive widths for vision tasks.

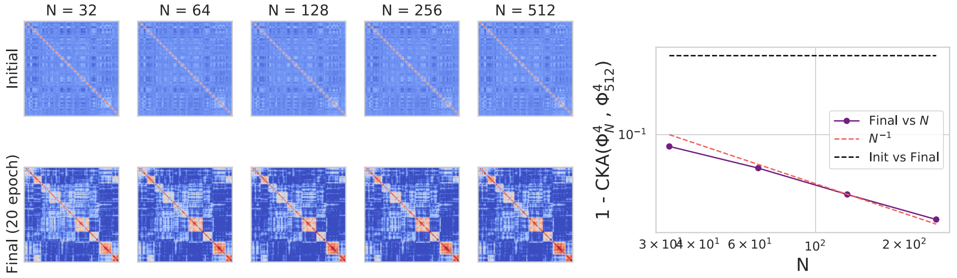
A simple setting in which convergence properties are particularly clear and simple to study is for a ResNet learning CIFAR-10 and going over multiple passes of the dataset. In Figure 10 we plot a 20-epoch pass over CIFAR 10, and study the generalization error, initial and final preactivations in the last layer, and final layer kernels across widths. The training error begins to exhibit pathologies after sufficiently many epochs, related to the discussion in section 3.2.

Next in figure 11, we show a higher-resolution plot of the kernel Gram matrices across widths and across layers for the CIFAR-5M ResNet after a pass through the data. The larger resolution allows one to see that even the fine-grained details in the structure of the Gram matrix are consistent across widths.



(a) Error Consistency

(b) Last Layer Preactivation Consistency



(c) Last Layer Kernel Consistency

(d) Kernel Alignment vs N

Figure 10: Training on CIFAR-10 in a ResNet-18 for multiple epochs generates dynamic preactivation densities and feature kernels which converge at realistic widths. (a) The test classification error curves for single models (solid) and ensembled (dashed) converge for realistic widths. (b) At initialization preactivation distributions in the last hidden layer of the CNN for a randomly chosen data point are Gaussian (as expected) and are very consistent across model widths N. To obtain histograms we train an ensemble of $E = 8$ independently initialized networks concatenate activation patterns across members of the ensemble. After 20 epochs of training (models are around $\sim 95\%$ accuracy), the preactivation distributions for the same data point have become non-Gaussian (consistent with infinite width theory) but are still remarkably consistent for large widths. (c) The final layer's feature kernel at initialization shows very little structure, but (d) after training networks of all widths converge to similar kernels. The plot in (d) compares ensemble averaged kernels with the $N = 512$ ensembled kernel.

B.2 Language

In Figure 9c) We plot the RMSE difference between the values placed on the correct logit by networks of successive width (rather than comparing to the widest as in Figure 3). We again see that as the widths grow the differences shrink.

Next, in Figure 12, we create an analog of the language column of Figure 3, this time for μP transformers of the same architecture and dataset but now optimized with vanilla SGD. The fact that wider transformers perform better still holds, and one can clearly see narrower networks approaching wider ones in their output logit values.

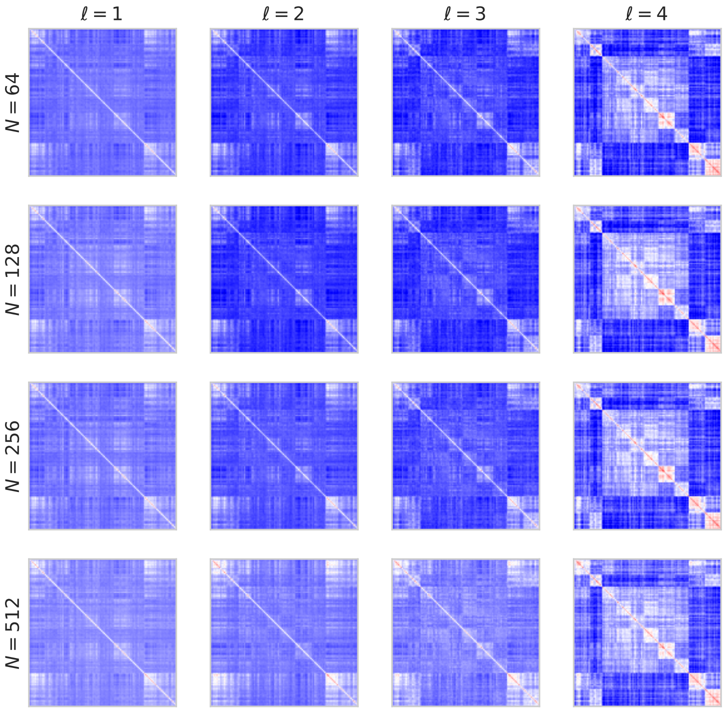
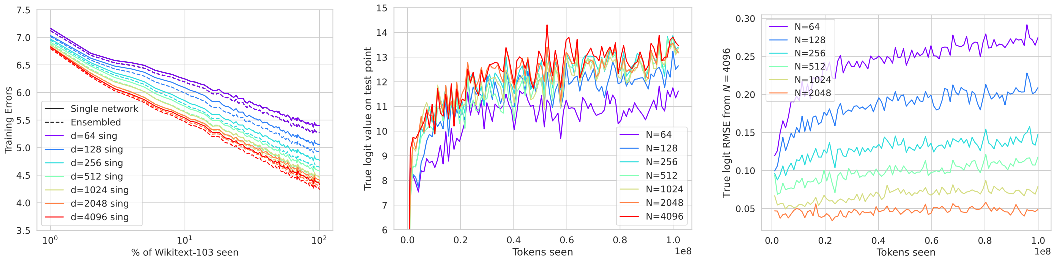


Figure 11: Convergence of layerwise representations in each layer (block) ℓ of the ResNet-18 at large width N after training on CIFAR-5M.



(a) Training Loss (b) Correct logit value (c) RMSE from widest

Figure 12: An analog of Figure 3 for μP transformers trained with SGD. a) Training loss. It is interesting that in μ parameterization the SGD optimized network is competitive with the Adam-optimized network. b) Value placed on the correct logit for a specific masked token. c) RMSE of correct logit value from the widest network.

C Defining μP and SP

There are several detailed discussions about μP vs SP scaling [10, 12, 7, 8, 11]. The aim of this section is to simply give an accessible and conceptual overview of their distinction, as well as a motivation for μP from the perspective of keeping features moving in time even at infinite width. .

There are several equivalent ways of parameterizing neural networks that give rise to the same dynamical effects, whether in μ-parameterization or standard parameterization. We give the definitions in the case of a single-output feed-forward network and demonstrate that SP and μP give rise to $O_N(N^{-1/2}), O_N(1)$ feature movement at initialization, respectively.

Generalizations to other architectures (ResNets, Transformers) are straightforward. For a detailed discussion see [10] and also [11].

C.1 SP

We assume all hidden layers have equal width N. Let the input space have dimension D. Let μ be the index of the training point in the dataset. At each layer ℓ, the pre-activation $\boldsymbol{h}_\mu^{\ell+1}$ in layer $\ell+1$ is given by

$$\boldsymbol{h}_\mu^{\ell+1} = \frac{1}{\sqrt{N}} \boldsymbol{W}^\ell \cdot \phi(\boldsymbol{h}_\mu^\ell), \tag{2}$$

where ϕ is an element-wise non-linearity, often taken to be the ReLU function. Here the $N^{-1/2}$ out front allows $\boldsymbol{h}_\mu^{\ell+1}$ to be $O_N(1)$ at initialization as $N \to \infty$ by the law of large numbers. The output of the network f_μ is then given by:

$$f_\mu = \frac{\alpha}{\sqrt{N}} \boldsymbol{w}^L \cdot \phi(\boldsymbol{h}_\mu^\ell). \tag{3}$$

Here again the $N^{-1/2}$ scaling again yields that f_μ will be $O_N(1)$ as $N \to \infty$. In SP, α is taken to be 1, but we will keep it explicit as it plays an important role in distinguishing the parameterizations. It is the laziness parameter identified in [28]. The change in the function is given by

$$\frac{df_\mu}{dt} = -\eta \sum_\nu K_{\mu\nu} \ell'(f_\nu, y_\nu). \tag{4}$$

Here $K_{\mu\nu} = \nabla_\theta f_\mu \cdot \nabla_\theta f_\nu$ is the NTK gram matrix. y^ν is the true label. ℓ is the loss function (e.g. MSE or crossentropy) and ℓ' is its derivative with respect to the first argument. The NTK is easily seen to be order α^2 and ℓ' is order 1 at small α. In order to have the change in the function be $O(1)$ we set $\eta = \eta_0/\alpha^2$.

Using the chain rule, one can directly see that the pre-activations evolve as [8, 3, 29]

$$\frac{d\boldsymbol{h}^\ell}{dt} \sim \eta \frac{\alpha}{\sqrt{N}} = \frac{\eta_0}{\alpha\sqrt{N}}. \tag{5}$$

Thus, at large N and $\alpha = 1$ the pre-activations of this network evolve as $O(N^{-1/2})$. Consequently, at infinite width the feature do not evolve and infinitely wide networks in standard parameterization become kernel machines with the static and initialization-independent infinite-width NTK.

In many machine learning libraries, the factors of $1/\sqrt{N}$ are not explicitly placed in front of each multiplication with the weight matrices. Rather, the weight matrices themselves are drawn from a distribution $\boldsymbol{W}^\ell \sim \mathcal{N}(0, \frac{1}{N}\mathbf{1})$. Although this gives identical forward pass, this changes the N scaling of the gradients in the backward pass. As long as the learning rate is appropriately rescaled to account for this, the dynamics are equivalent to the SP parameterization discussed above.

C.2 μP

One of the simplest ways to define the μ-parameterization is to take $\alpha = 1/\sqrt{N}$. This implies that we simply replace the final layer of the network by:

$$f_\mu = \frac{1}{N} \boldsymbol{w}^L \phi(\boldsymbol{h}_\mu^\ell). \tag{6}$$

As the prior analysis shows, in order to have df_μ/dt be $O(1)$ at initialization, we take $\eta = N\eta_0$, so the learning rate in this definition scales extensively with N. In this setting, we now have that

$$\frac{d\boldsymbol{h}^\ell}{dt} \sim O_N(1). \tag{7}$$

In [10, 12], gives an equivalent definition of μP that gives rise to the same dynamics but keeps the learning rate to be $O_N(1)$. We use this version of μP in the experiments that we run, simply because that is what is used in the package [14]. Consequently, our learning rate does not need to be changed as width grows.

D μP versus Standard Parameterization

D.1 MLPs

In figure 13, we show a 3-layer MLP learning a quadratic polynomial. In subfigure a) use a batch size of 10 and a learning rate going as $\eta = 5N/(1+\alpha_0^2)$ with $\alpha_0 = 1$. The output layer is scaled as $\alpha_0/\sqrt{N}$, putting us in the rich regime. The learning rate has been picked to be nearly as large as possible at this batch size in order to maximize the large loss curve fluctuations yielded by large learning rate effects. In subfigure c) we do not rescale the output layer, and have a width-indepdent learning rate going as $\eta = 50/(1+\alpha_0^2)$ with $\alpha_0 = 1$. This puts is in the large-learning rate regime for a standard parameterized network. See Appendix A for more details.

We plot the learning curves across widths and find striking agreement, even at the fine-grained level of fluctuations due to small batch size and large learning rate effects. Although this is not exactly the full-batch edge-of-stability effect reported in [45], the large oscillations may be similar to a small batch size analog. We plot the absolute difference from the widest network in subfigure b) to highlight the strong agreement across widths.

In subfigure c), we have the same network but in standard parameterization. The narrower networks now learn features more quickly, leading to inconsistent dynamics across widths.

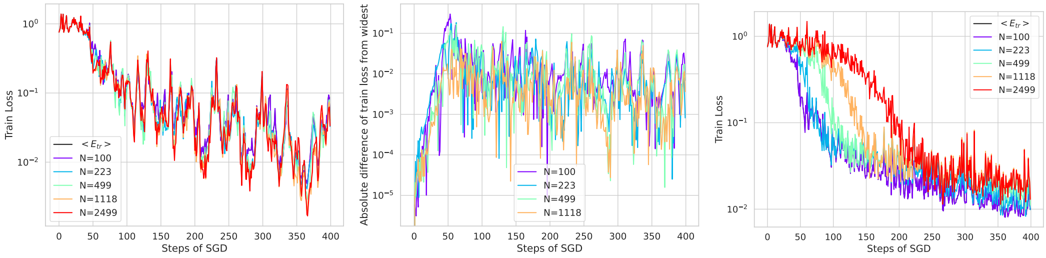


Figure 13: 3-Layer MLP learning a quadratic polynomial $y = Q_2(\boldsymbol{\beta} \cdot \boldsymbol{x})$ on the sphere. Data is provided in an online setting in the same order across widths, as in the realistic experiments in the main text. Large learning rate small batch size effects in MLPs are consistent across large widths. a) For μP networks, the loss curves match across widths, even accounting for fluctuations due to batch size or large learning rate edge-of-stability-like effects. b) Plot of the difference in training error from the widest network. c) The same network but in standard parameterization. Dynamics are no longer consistent across widths, and wider networks approach a lazy limit.

D.2 Vision

Next, we focus on a vision task and compare the large learning rate small batch size effects in SP to the μP parameterized network in Figure 5a. By contrast to that figure, we see significantly different dynamics and batch variation across widths. In Figure 14 a) we plot the early time behavior of a CIFAR-5m task at large learning rate. The large learning rate effects cause the loss to substantially oscillate, but the oscillations across widths are inconsistent by contrast to 5a b) . Further, at late times in Figure 14, the sharp spikes in the loss function due to large learning rate effects become substantially different across widths. Indeed, in SP some widths converge for a given learning rate while others do not. This trend has already been well-studied in [12]. We again stress that our observation is that not only are the final losses similar across widths in μP (as observed in [12]), but that the individual batch and large learning rate fluctuations agree across widths at early times in μP as well.

D.3 Language

Finally, we present a complementary set of figures to those in the right columns of Figures 3 and 12 for transformers of the same architecture on Wikitext-103 but in standard parameterization.

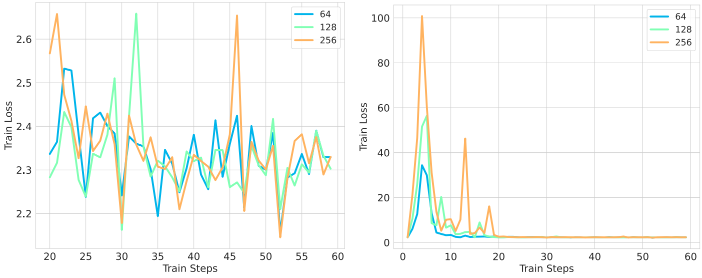
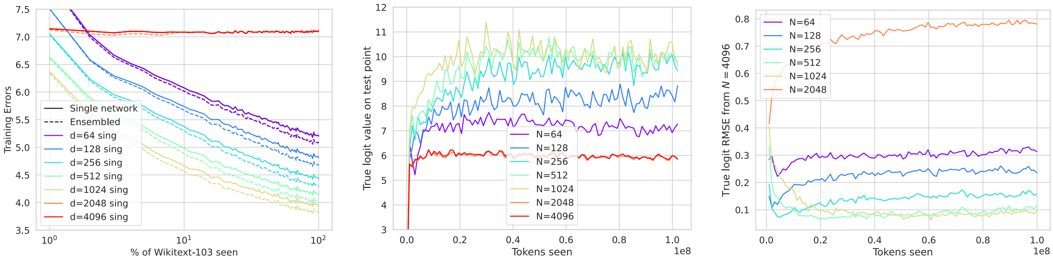


Figure 14: Different widths have different loss curves. a) Early time dynamics of the loss across widths is not consistent. b) Dynamics of the loss across widths at later times also does not appear consistent. There are explosions that happen at different times and scales across widths.



(a) Training Loss

(b) Correct logit value

(c) RMSE from widest

Figure 15: An analog of Figure 3: SP transformers trained with Adam. a) Training loss. At large widths, the learning rate chosen is too big for the network to properly learn, and the loss is flat. This is consistent with what is observed in [12] — the optimal learning rate in SP changes with width. b) Value placed on the correct logit for a specific masked token. c) RMSE of correct logit value from the widest network. In both of these plots, the monotonic behavior across width evident under the μP parameterization is violated. Even after discarding the networks that do not converge under SP, the behavior remains non-monotonic across width.

E Overview of Finite Width Corrections to Feature Learning Networks

In this section we review some basic ideas from the mean field theory of feature learning neural networks. We first describe the predictions that mean field theory makes about infinite width networks before describing finite size corrections to the dynamics of learning. To eliminate unnecessary complexity, we will focus on MLP layers, but these arguments can be easily extended to CNN and self-attention layers as well. We start by defining a MLP in a parameterization equivalent to μP

$$f_\mu = \frac{1}{N} \boldsymbol{w}^L \cdot \phi(\boldsymbol{h}^\ell_\mu) \ , \ \boldsymbol{h}^{\ell+1}_\mu = \frac{1}{\sqrt{N}} \boldsymbol{W}^\ell \phi(\boldsymbol{h}^\ell_\mu) \ , \ \boldsymbol{h}^1_\mu = \frac{1}{\sqrt{D}} \boldsymbol{W}^0 \boldsymbol{x}_\mu . \tag{8}$$

We will consider these networks trained from a random Gaussian initialization of the weights so that $\boldsymbol{\theta} = \text{Vec}\{\boldsymbol{w}^L, ..., \boldsymbol{W}^0\}$ follows $\boldsymbol{\theta} \sim \mathcal{N}(0, \boldsymbol{I})$ at initialization. This network is then trained with some gradient based optimizer, leading to dynamical predictions $f_\mu(t)$ and dynamical preactivations $\boldsymbol{h}^\ell_\mu(t)$. Because of the random initialization of weights, the outputs of the network and the precise preactivations are random variables. However, at infinite width $N \to \infty$, a dramatic simplification of the dynamics occurs.

E.1 The Infinite Width/Mean Field Limit

The predictions $f_\mu(t)$ and internal representations of infinite width limit of neural networks admit a description in terms of non-random initialization-independent dynamical feature kernels $\Phi^\ell_{\mu\nu}(t,s)$ and gradient kernels $G^\ell_{\mu\nu}(t,s)$ defined as

$$\Phi^\ell_{\mu\nu}(t,s) = \frac{1}{N} \phi(\boldsymbol{h}^\ell_\mu(t)) \cdot \phi(\boldsymbol{h}^\ell_\nu(s)) \ , \ G^\ell_{\mu\nu}(t,s) = \frac{1}{N} \boldsymbol{g}^\ell_\mu(t) \cdot \boldsymbol{g}^\ell_\nu(s), \tag{9}$$

where μ, ν index data points and t, s index training time and $\boldsymbol{g}^\ell_\mu(t) = N \frac{\partial f_\mu}{\partial \boldsymbol{h}^\ell}$ are back-propagated gradient signals [10, 11]. Further, all preactivation vectors $\boldsymbol{h}^\ell_\mu(t) \in \mathbb{R}^N$ have entries that become iid draws from a (potentially non-Gaussian) single site density $p(h)$, which converges as

$$\frac{1}{N}\sum_{i=1}^{N} \delta(h - h_i) \to p(h), \tag{10}$$

which should be understood in terms of integration of these densities against test functions. At infinite width, the sums over neurons in a layer can be replaced by deterministic integrals over this single site density $\Phi^\ell_{\mu\nu}(t,s) = \int p(h^\ell_\mu(t), h^\ell_\nu(s))\phi(h^\ell_\mu(t))\phi(h^\ell_\nu(s)) dh^\ell_\mu(t) dh^\ell_\nu(s)$.

E.2 Finite Width Effects

At finite width, the internal kernels $\{\Phi^\ell_{\mu\nu}(t,s), G^\ell_{\mu\nu}(t,s)\}$ and predictions $f_\mu(t)$ of the model become initialization and width-dependent and deviate from their mean field dynamics. For Gaussian random initialization of the weights of the network, the predictions and kernels fluctuate (from init to init) with variance that scales asymptotically like $\mathcal{O}(1/N)$ for width N (or $1/d_{model}$ for transformer) [36]. Further, the *ensemble averaged* values for the predictions $\langle f_\mu(t)\rangle$ and kernels $\left\langle \Phi^\ell_{\mu\nu}(t,s)\right\rangle$ differ asymptotically from their infinite width values by $\mathcal{O}(N^{-1})$. Both of these two leading order effects can influence the expected (train or test) loss of the model. At fixed width and late training time, finite size effects beyond leading order can accumulate and become relevant, however theory predicts that any observable average at width N admits an asymptotic series in powers of N^{-1} [36].

E.2.1 Trainability at Finite Size

The $\mathcal{O}(N^{-1})$ correction to feature and gradient kernels can lead to non-trivial corrections to the loss dynamics. Working in continuous time, we can define the neural tangent kernel (NTK) as $K_{\mu\nu}(t) = \sum_\ell G^{\ell+1}_{\mu\nu}(t,t)\Phi^\ell_{\mu\nu}(t,t)$, where base cases are $\Phi^0_{\mu\nu}(t,s) = \frac{1}{D}\boldsymbol{x}_\mu \cdot \boldsymbol{x}_\nu$ and $G^{L+1}_{\mu\nu}(t,s) = 1$. Following the approximation to online dynamics with MSE loss in Section 4, we consider a gradient flow on the average dynamical NTK

$$\frac{d}{dt}\boldsymbol{\Delta}(t) = -\langle \boldsymbol{K}(t)\rangle \boldsymbol{\Delta}(t) \implies \boldsymbol{\Delta}(t) = \mathcal{T}\exp\left(-\int_0^t ds \, \langle \boldsymbol{K}(s)\rangle\right)\boldsymbol{y}, \tag{11}$$

where $\mathcal{T}$ is the time-ordering operator. We now consider the leading correction to the average NTK around infinite width $\langle \boldsymbol{K}(t)\rangle = \boldsymbol{K}_\infty(t) + \frac{1}{N}\boldsymbol{K}^1(t) + \mathcal{O}(N^{-2})$. With this correction, we see that the dynamics of errors $\boldsymbol{\Delta}$

$$\boldsymbol{\Delta}(t) = \mathcal{T}\exp\left(-\int_0^t ds \boldsymbol{K}_\infty(s) - \frac{1}{N}\int_0^t ds \boldsymbol{K}^1(s) + \mathcal{O}(N^{-2})\right)\boldsymbol{y}. \tag{12}$$

The fact that the $\frac{1}{N}\boldsymbol{K}^1$ correction is integrated over time and placed in the matrix exponential indicates that small corrections to NTK dynamics can lead to large dynamical amplification of logit corrections. This fact was pointed out in another work [36] which tried to motivate a study of perturbation theory in logarithms of the transition matrix $\log \boldsymbol{T}(t)$ defined as

$$\frac{d}{dt}\boldsymbol{T}(t) = -\langle \boldsymbol{K}(t)\rangle \boldsymbol{T}(t) \; , \; \boldsymbol{T}(0) = \boldsymbol{I} \; , \; \boldsymbol{R}(t) = \log \boldsymbol{T}(t). \tag{13}$$

The solution to this can be used to construct the errors at a later time $\boldsymbol{\Delta}(t) = \exp\left(\boldsymbol{R}(t)\right)\boldsymbol{y}$.

F Offline Training

Figure 16 depicts the loss curve for a ConvNeXt-T (tiny) model trained on ImageNet in the typical, offline setting — where data is encountered repeatedly across many epochs. As the networks overfit the training data — in Figure 16, beyond 40,000 training steps or five epochs — the loss curves diverge dramatically for different-width networks. Width consistency subsequently erodes.

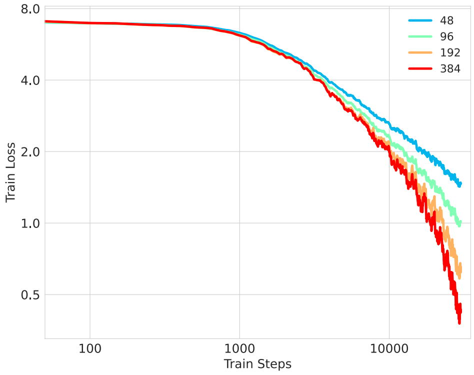


Figure 16: Consistency of loss curves across widths in the beginning and separation as loss becomes sufficiently small in offline learning.

G Use of Compute

For most experiments, we used Nvidia A100 SXM4 40GB and 80 GB GPUs on an academic cluster.

For the Wikitext-103 tasks, each width included 4 ensembles loaded onto an A100 GPU that ran for a range between 1 to 3 days. For each sweep over widths this corresponds to about 8 A100-days. Accounting for sweeps over different sequence lengths, optimizers, and parameterizations, this corresponds to about 50 A100-days.

All MLP tasks, including the calculation of empirical NTKs and their spectral properties were done in 15-30 minute Colab sessions using the basic GPUs provided.

The CIFAR-10 ResNet experiments in Figure 10 were done using a total of less than 1 A100-day of compute across all widths and ensembles.

The ImageNet ResNet experiments vectorize training over between one to four same-width neural networks on one A100 GPU. Each experiment training a collection of networks for 30 epochs takes between one to three A100-days. Overall, these experiments expended roughly 30 A100-days.

For the CIFAR-5m experiments in Figure 2 and 3, across all widths, it required a few hours of A100 GPU. For Figure 6 and 7, as these were ensembled across multiple runs, these required close to 1-2 days of A100-GPUs. Figure 5a was just run for a few 100 steps of the training, so didn't use much compute power.