A Spectral Condition for Feature Learning

Greg Yang* xAI

James B. Simon* UC Berkeley & Imbue

Jeremy Bernstein* MIT

Abstract

The push to train ever larger neural networks has motivated the study of initialization and training at large network width. A key challenge is to scale training so that a network's internal representations evolve nontrivially at all widths, a process known as *feature learning*. Here, we show that feature learning is achieved by scaling the *spectral norm* of weight matrices and their updates like $\sqrt{\texttt{fan-out}/\texttt{fan-in}}$, in contrast to widely used but heuristic scalings based on Frobenius norm and entry size. Our spectral scaling analysis also leads to an elementary derivation of *maximal update parametrization*. We develop this spectral perspective and structure the text with the intent of providing the reader with a solid conceptual understanding of the scaling behavior of feature learning in neural networks.

Schatten-2 norm (Frobenius) versus the Schetten-infinty norm (spectral)

max(spectral norm) < max(frobenius)

Hmm - wouldn't the spectral norm down be better since that means that the eigenspace is becoming more uniform* (and it thus uses more feature space)

Schatten-P norms: P(1) (nuclear) ≥ P(2) (Fro) ≥ P(inf) (spectral)

1 Introduction

Recent years have seen an unprecedented push to train deep learning systems with more and more parameters, leading to powerful models across domains and the unlocking of qualitatively new capabilities (Brown et al., 2020; Ramesh et al., 2022; Silver et al., 2016). This continuing trend, combined with the technical challenges of training large models, has motivated much recent study of the dynamics of neural networks at *large width*, and more generally the study of how their dynamics scale as network width grows. This program has yielded a cornucopia of theoretical insights (Arora et al., 2019; Canatar et al., 2021; Jacot et al., 2018; Lee et al., 2018) and practical scaling recommendations (Dey et al., 2023; Yang & Hu, 2021b; Yang et al., 2021).

A key challenge when training a network of large width is to ensure that *feature learning* occurs at hidden layers. By this, we mean that the hyperparameters of the network are scaled in a manner such that the hidden representations of the network (as obtained by partial evaluation of the network up to a certain layer) change substantially over the course of training. Naïve hyperparameter scaling rules, including the well-studied "neural tangent parametrization" (NTP), in fact *lose* feature learning at large width (Lee et al., 2019; Sohl-Dickstein et al., 2020). But ample evidence supports the conclusion that proper feature learning is necessary for achieving optimal performance on many tasks (Atanasov et al., 2022; Fort et al., 2020; Lee et al., 2020; Vyas et al., 2022). Furthermore, scaling training correctly can lead to new functionality such as *hyperparameter transfer*. For instance, the recently proposed *maximal update parametrization* (Yang & Hu, 2021b; Yang et al., 2021) allows for transferring hyperparameters from narrow models to wide models, avoiding the cost of tuning the wide model directly.

Maximal update parametrization (μP) is derived by fairly involved "tensor programs" arguments that track feature distributions analytically in the infinite width limit. Anecdotally, the principles underlying μP are not well understood by the community. In this paper, we provide a new perspective on μP, showing that its scaling relations can be obtained by elementary linear algebra arguments. In short, we show that μP is equivalent to scaling the *spectral norm* of any weight matrix or update like $\sqrt{\texttt{fan-out}/\texttt{fan-in}}$. This simple condition has various favorable numerical properties that contrast sharply with heuristic optimization strategies based on controlling the Frobenius norm (You et al., 2017) or entry size (Kingma & Ba, 2015) of updates. In the authors' experience, the spectral scaling condition both simplifies the implementation of μP in code, and is significantly easier to work with theoretically, leading to further conceptual advances in our research (Bernstein et al., 2023).

"update" here I think means linear projections (weight proj)

*Equal contribution.

On a more fundamental level, an important step to solving many problems in classical computer science is to write down a suitable distance function for the problem at hand (Dhillon & Tropp, 2008). This idea is of particular importance in the design of optimization algorithms, where a notion of parameter distance is needed (Amari, 1998; Nemirovsky & Yudin, 1983). While it can be tempting to use the Euclidean norm on parameter vectors to measure distance, this naïve choice risks discarding the structure of the problem. For example, neural networks involve compositions of linear operators, which we refer to as their *operator structure*. Past efforts to metrize the space of neural networks while accounting for their operator structure have included using the Frobenius norm to measure distance between matrices (Bernstein et al., 2020a), which motivates various optimization algorithms that make Frobenius-normalized updates (Liu et al., 2021; Shazeer & Stern, 2018; You et al., 2017; 2020). This paper shows that the spectral norm provides a better notion of distance between operators in the context of deep learning.

1.1 Road map for the paper

To begin, we state the precise conditions on hidden features that we wish to ensure. We will ask for two things: both the *features* and their *updates* upon a step of gradient descent must be the proper size. **Yep. Stability**

Kaiming He normal achieves this

Desideratum 1 (Feature learning)**.** Let $\boldsymbol{h}_\ell(\boldsymbol{x}) \in \mathbb{R}^{n_\ell}$ denote the features of input $\boldsymbol{x}$ at layer ℓ of a neural network, and let $\Delta\boldsymbol{h}_\ell(\boldsymbol{x}) \in \mathbb{R}^{n_\ell}$ denote their change after a gradient step. We desire that:

Since the variance of each param in each vector having an std. dev. of 1, the L2 of that vector should just be sqrt(width)

I think these are vectors

$$\|\boldsymbol{h}_\ell\|_2 = \Theta(\sqrt{n_\ell}) \quad \text{and} \quad \|\Delta\boldsymbol{h}_\ell\|_2 = \Theta(\sqrt{n_\ell}), \qquad \text{at layers } \ell = 1, ..., L-1.$$

L2 norm of the activated state **n=layer width**

the L2 norm of the activation being sqrt(width_n) means that the expectation of each dim in that feature is ~1

Let us unpack Desideratum 1. These conditions treat the ℓ^2*-norms* of the feature vectors $\boldsymbol{h}_\ell(\boldsymbol{x})$ and $\Delta\boldsymbol{h}_\ell(\boldsymbol{x})$, a framing which will prove convenient. Desideratum 1 amounts to asking that the "typical element size" of vectors $\boldsymbol{h}_\ell(\boldsymbol{x})$ and $\Delta\boldsymbol{h}_\ell(\boldsymbol{x})$ is $\Theta(1)$ with respect to width n_ℓ (we give a review of big-Θ notation in Section 2). Enforcing that hidden features have $\Theta(1)$ element size has long been a principle of deep learning parametrization, motivated by the fact that activation functions are designed to take order-one inputs and give order-one outputs (LeCun et al., 2002). Our second requirement stipulates that feature entries also undergo $\Theta(1)$ updates during training. Note that any larger updates would blow up at large width, and any smaller updates would vanish at large width. We take Desideratum 1 as our definition of feature learning.[1]

"order one" means that the magnitude is bounded

Our main message is that feature learning in the sense of Desideratum 1 may be ensured by the following *spectral scaling condition* on the weight matrices of a deep network and their gradient updates:

Condition 1 (Spectral scaling)**.** Consider applying a gradient update $\Delta\boldsymbol{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ to the ℓth weight matrix $\boldsymbol{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$. The spectral norms of these matrices should satisfy:

$$\|\boldsymbol{W}_\ell\|_* = \Theta\left(\sqrt{\frac{n_\ell}{n_{\ell-1}}}\right) \quad \text{and} \quad \|\Delta\boldsymbol{W}_\ell\|_* = \Theta\left(\sqrt{\frac{n_\ell}{n_{\ell-1}}}\right), \qquad \text{at layers } \ell = 1, ..., L.$$

Why do you want the spectral norm of the gradient to be this though?

```
>>>
np.linalg.norm(n
p.random.normal
(0, 1/
np.sqrt(width),
(width, width)),
ord=2)
1.9832788903797
645 # 2 is on the
order of 1
```

We review the spectral norm in Section 2. The spectral scaling condition has two components which will serve to enforce the respective components of Desideratum 1. The first component mandates that each *weight matrix* has a spectral norm of a certain size, which will serve to enforce that the layer passes forward features of the correct size. The second component mandates that each *gradient update* has a spectral norm of a certain size, which will ensure that subsequent features undergo a change of the correct size. We have implicitly assumed that the input has size $\|\boldsymbol{x}\|_2 = \Theta(\sqrt{n_0})$, which is standard for image data. Language models are an important counterexample, where embedding matrices take one-hot inputs and the $\sqrt{n_0}$ in Condition 1 should be replaced by 1. Appendix E provides a unifying treatment of these cases.

To get some quick intuition for the origin of Condition 1, observe that under the forward propagation $\boldsymbol{h}_\ell(\boldsymbol{x}) = \boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x})$, if the layer input $\boldsymbol{h}_{\ell-1}(\boldsymbol{x})$ aligns with the top singular vector of the weight matrix $\boldsymbol{W}_\ell$, then $\|\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \|\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2$. The requirement that $\|\boldsymbol{W}_\ell\|_* = \Theta(\sqrt{n_\ell/n_{\ell-1}})$ then follows from

Oh yeah, these `h` are vectors, so we use L2

[1] Our notion of feature learning might also be called "nontrivial feature evolution." While other authors may prefer different notions of "feature learning"—for example, the learning of interpretable, visualizable functions at hidden nodes (Olah et al., 2017; Zeiler & Fergus, 2014)—nontrivial feature evolution in our sense is necessary for any other reasonable definition of the term.

Example in code:
```
>>> width = 100
>>> batch = 1_000
>>> h = np.random.normal(0, 1.0, (batch, width))   # per-coordinate σ = 1
>>> np.linalg.norm(h, axis=1).mean()
~10 which is sqrt(width)
```

Example in code: (might be wrong, I thought u wanted var to be 1/sqrt(n))
```
X = np.random.normal(0, 1/np.sqrt(1_000), (1_000, 100))
# X is 1K features of dim 100 & variance = 1/n
(lambda x: (np.mean(x.sum(axis=1)), np.var(x.sum(axis=1))))(X)
(~0, ~1) # resulting mean & variance of the "mock activations" here is (0, 1)
```

Desideratum 1. The scaling of $\|\Delta \boldsymbol{W}_\ell\|_*$ can similarly be obtained by writing $\Delta \boldsymbol{h}_\ell(\boldsymbol{x}) = \Delta \boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x}) + \ldots$ and applying the same argument. The key missing step is to justify that layer inputs actually do line up with the top singular subspaces of both weight matrices and weight updates. As the paper will show, gradient descent training actually induces this form of alignment. Well this would really help with interpretability

The bulk of this paper is dedicated to thoroughly demonstrating that training in accordance with our spectral scaling condition satisfies Desideratum 1 in MLPs. As an accessible path to this conclusion, we begin in Section 3 with a simple model—a deep linear MLP trained for one step on one example—and then successively extend to multiple training steps, a nonlinear model, and multiple inputs. In the process, we give a scaling analysis of the dynamics of feature learning. We then explain how Condition 1 may be achieved in a standard deep learning setting and compare-and-contrast the resulting scaling prescription with others in the literature. In particular, we recover the recent "maximal-update parametrization" (μP) (Yang & Hu, 2021b).

1.2 Summary of contributions

Concretely, our contributions are as follows:

- We propose the *spectral scaling condition* (Condition 1) and show that it suffices to achieve feature learning in neural networks even at large width.
- We show how Condition 1 may be implemented: either via direct spectral normalization, or by layer-wise initialization scales $\{\sigma_\ell\}$ and learning rates $\{\eta_\ell\}$ that recover *maximal update parametrization*.[2]
Oh nice, I was thinking of just doing a spectral normalization, makes sense that initialization would be similar (assuming normal inputs)
- We show that other popular scaling rules, including so-called *standard parameterization* and *neural tangent parametrization*, fail to satisfy Condition 1.

In the main text, we focus on MLPs trained via ordinary gradient descent for clarity. Our results may actually be extended to cover any architecture and any adaptive optimizer (for a suitable definition of *any*, c.f. Appendix B). Therefore our spectral scaling condition provides a unifying hyperparameter scaling rule that remains the same whether the underlying optimizer is, say, SGD or Adam. We suggest that, when one wishes to determine how the hyperparameters of a new deep learning system should scale with width, one might turn to the satisfaction of Condition 1 as an overarching principle.

2 Preliminaries

Here we review standard notations which we use in our scaling analysis.

Scaling notation. We will use the usual big-O notation and variants to make statements about how various quantities scale with network width. Intuitively speaking:

- $f(n) = O(g(n))$ means that $f(n)$ "scales no faster than" $g(n)$,
- $f(n) = \Theta(g(n))$ means that $f(n)$ "scales like" or "is order" $g(n)$,
- $f(n) = \Omega(g(n))$ means that $f(n)$ "scales at least as fast as" $g(n)$.

Formally, $f(n) = \Theta(g(n))$ is equivalent to the statement that there exist constants $c, C > 0$ such that $c \cdot g(n) \leq f(n) \leq C \cdot g(n)$ for all sufficiently large d. The weaker statements $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ entail only the upper and lower bounds, respectively.

We will *only* be concerned with scaling with respect to layer widths in this paper. Big-O notation will hide any dependence on other factors — such as depth, dataset size, learning rate schedule, a global learning rate prefactor — and our statements purely concern how quantities will or should scale with model width.

Vector and matrix norms. We will use the standard ℓ^2-norm $\|\cdot\|_2$ to assess the size of a vector. For matrices, we will principally use the *spectral norm* $\|\cdot\|_*$ (a.k.a. *operator norm*) defined as follows:

Definition 1 (Spectral norm)**.** The spectral norm of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is given by

$$\|\boldsymbol{A}\|_* := \max_{\boldsymbol{v} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}} \frac{\|\boldsymbol{A}\boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2}. \tag{1}$$

[2] In fact, Condition 1 is the *unique* scaling on spectral norm that is equivalent to μP in its usual definition in terms of learning rate and initialization scaling (Theorem 1).

That is, the spectral norm is the largest factor by which a matrix can increase the norm of a vector on which it acts. The spectral norm of a matrix is equal to its largest singular value. We will sometimes contrast the spectral norm with the *Frobenius norm* $\|\cdot\|_F$ given by $\|\boldsymbol{A}\|_F^2 = \sum_{ij} A_{ij}^2$.

Properties of the spectral norm. Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m\times n}$ be arbitrary matrices and $\boldsymbol{v} \in \mathbb{R}^n$ be an arbitrary vector. As with all norms, the spectral norm is *subadditive*, meaning that it obeys the *triangle inequality* $\|\boldsymbol{A}+\boldsymbol{B}\|_* \leq \|\boldsymbol{A}\|_* + \|\boldsymbol{B}\|_*$. The spectral norm is also *submultiplicative* in the sense that $\|\boldsymbol{Av}\|_* \leq \|\boldsymbol{A}\|_* \cdot \|\boldsymbol{v}\|_2$ and $\|\boldsymbol{AB}\|_* \leq \|\boldsymbol{A}\|_* \cdot \|\boldsymbol{B}\|_*$. If we interpret a vector $\boldsymbol{v} \in \mathbb{R}^n$ as a $1 \times n$ matrix, then the ℓ^2, spectral and Frobenius norms are equivalent: $\|\boldsymbol{v}\|_2 = \|\boldsymbol{v}\|_* = \|\boldsymbol{v}\|_F$.

Special cases of the spectral norm. For a *rank-one* matrix $\boldsymbol{A}$, which can be written as an outer-product $\boldsymbol{A} = \boldsymbol{uv}^\top$, it holds that $\|\boldsymbol{A}\|_* = \|\boldsymbol{A}\|_F = \|\boldsymbol{u}\|_2 \cdot \|\boldsymbol{v}\|_2$. A matrix $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ is *semi-orthogonal* if either $\boldsymbol{B}^\top\boldsymbol{B} = \boldsymbol{I}_n$ or $\boldsymbol{BB}^\top = \boldsymbol{I}_m$. A semi-orthogonal matrix has unit spectral norm: $\|\boldsymbol{B}\|_* = 1$.

3 The spectral scaling condition induces feature learning

In this section, we show that our spectral scaling condition (Condition 1) achieves feature evolution of the correct scale in multilayer perceptrons (MLPs). We begin with a toy example which conveys key intuitions and then give a series of extensions which recover a much more general case.

3.1 Warmup: deep linear MLP, one step of SGD, on a single example

We begin with a simple model: a deep linear MLP trained for one step on a single input. While elementary, this example will be sufficient to capture the intuition for a much more general case.

Model definition. We will study an MLP composed of L successive linear transformations. We consider a single input $\boldsymbol{x} \in \mathbb{R}^{n_0}$ with norm $\|\boldsymbol{x}\|_2 = \Theta(\sqrt{n_0})$. The first hidden representation is $\boldsymbol{h}_1(\boldsymbol{x}) = \boldsymbol{W}_1\boldsymbol{x}$, with the rest of the network following recursively as:

$$\boldsymbol{h}_\ell(\boldsymbol{x}) = \boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x}) \qquad \text{for } \ell = 2, \ldots, L. \tag{2}$$

We let $\boldsymbol{h}_L(\boldsymbol{x}) \in \mathbb{R}^{n_L}$ be the network output. We will keep the input dimension n_0 and output dimension n_L fixed and consider scaling with respect to the hidden dimensions $n_1, \ldots, n_{L-1}$.

We let the global loss be $\mathcal{L} = g(\boldsymbol{h}_L(\boldsymbol{x}), \boldsymbol{y})$ where g and $\boldsymbol{y}$ are a loss function and target vector, respectively. During training, we will take gradient steps at each layer as $\Delta\boldsymbol{W}_\ell = -\eta_\ell \cdot \nabla_{\boldsymbol{W}_\ell}\mathcal{L}$, where η_ℓ is a layerwise learning rate. We will ultimately solve for the scale of η_ℓ, but for now we will be content to discuss the perturbation $\Delta\boldsymbol{W}_\ell$ directly. By Equation (2), hidden vector updates at subsequent layers are related by:

$$\boldsymbol{h}_\ell(\boldsymbol{x}) + \Delta\boldsymbol{h}_\ell(\boldsymbol{x}) = (\boldsymbol{W}_\ell + \Delta\boldsymbol{W}_\ell)(\boldsymbol{h}_{\ell-1}(\boldsymbol{x}) + \Delta\boldsymbol{h}_{\ell-1}(\boldsymbol{x})). \tag{3}$$

Hidden vector sizes. To reiterate Desideratum 1, we wish for features at the ℓth layer to have a norm which scales as $\|\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \Theta(\sqrt{n_\ell})$. Upon a gradient update, this feature vector should undergo an update of size $\|\Delta\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \Theta(\sqrt{n_\ell})$. We will show that Condition 1 is sufficient to achieve these aims at all layers.

Plan of attack. For simplicity, we will first focus on the *first step of gradient descent* after random initialization. We will argue recursively in depth, showing that if the features at layer $\ell-1$ and their updates satisfy Desideratum 1, then so will those at layer ℓ. In order to verify the desired scalings, we will show upper and lower scaling bounds separately: we will first show that the features and their updates are not larger than asked by Desideratum 1, and then show that they are in fact also not smaller than asked by Desideratum 1.

Hidden vector updates. By the subadditivity and submultiplicativity of the spectral norm, Equations (2) and (3) imply that:

$$\|\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 \leq \|\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 = \Theta(\sqrt{n_\ell}); \tag{4}$$

$$\|\Delta\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 \leq \|\Delta\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 + \|\boldsymbol{W}_\ell\|_* \cdot \|\Delta\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 + \|\Delta\boldsymbol{W}_\ell\|_* \cdot \|\Delta\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 = \Theta(\sqrt{n_\ell}), \tag{5}$$

where on the right hand sides of the inequality we have inserted Desideratum 1 and Condition 1. The spectral scaling condition thus gives features and feature updates obeying the correct *upper* bounds, and we need merely show comparable lower bounds.

Tightness of bounds via matrix-vector alignment. The upper bound in the submultiplicativity property $\|\boldsymbol{A}\boldsymbol{v}\|_2 \le \|\boldsymbol{A}\|_* \cdot \|\boldsymbol{v}\|_2$ can be very loose—in particular, this is the case when the vector only interacts with the small singular values in the matrix. We will now show that this is not the case in deep network training, and that these upper bounds provide a fairly accurate description of the way things scale. We make two observations regarding random weight matrices and gradient updates:

Claim 1 (Alignment of initial weight matrices)**.** *Fix a feature vector* $\boldsymbol{h}_{\ell-1}(\boldsymbol{x}) \in \mathbb{R}^{n_{\ell-1}}$. *Assume* $\boldsymbol{W}_\ell$ *in* $\mathbb{R}^{n_\ell \times n_{\ell-1}}$ *is sampled using a common weight initialization strategy (e.g., Gaussian or semi-orthogonal init). Provided that fan-out is no less than fan-in* ($n_\ell \ge n_{\ell-1}$)*, then with high probability:*

$$\|\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 = \Theta(\|\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2).$$

The principle dimensions line up

Claim 2 (Alignment of updates)**.** *For an update* $\Delta\boldsymbol{W}_\ell$ *given by gradient descent with batch size 1,*

$$\|\Delta\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 = \|\Delta\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2.$$

In words, Claim 1 states that random hidden weight matrices scale incoming vectors by factors commensurate to their spectral norms, so long as their fan-out is not smaller than their fan-in, which we will assume is the case for all but the final layer of the network. Claim 2 states the same of weight updates, but the proportionality constant is precisely one and requires no condition on dimensionality.[3]

We now justify these claims in turn. For Claim 1, first suppose that $\boldsymbol{W}_\ell$ is a random semi-orthogonal matrix as is a popular initialization strategy. Then all singular values of $\boldsymbol{W}_\ell$ are one and, since fan-out exceeds fan-in, the null-space of $\boldsymbol{W}_\ell$ is empty. Taken together, these observations imply the equality: $\|\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 = \|\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2$. Fortunately, if the elements of $\boldsymbol{W}_\ell$ are instead sampled i.i.d. from a centered Gaussian distribution with standard deviation σ_ℓ, then the situation is similar. It is easily shown by the law of large numbers that $\|\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 \approx \sigma_\ell \sqrt{n_\ell}\, \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2$, and it is a standard result in random matrix theory that $\|\boldsymbol{W}_\ell\|_* \approx \sigma_\ell(\sqrt{n_{\ell-1}} + \sqrt{n_\ell})$ (Rudelson & Vershynin, 2010; Vershynin, 2018). Claim 1 for Gaussian intialization follows by combining these results.

Claim 2 is easily verified as follows. Observe that we can write the update at layer ℓ as the outer-product:

$$\Delta\boldsymbol{W}_\ell = -\eta_\ell \cdot \underset{u}{\nabla_{\boldsymbol{h}_\ell(\boldsymbol{x})}\mathcal{L}} \cdot \underset{v}{\boldsymbol{h}_{\ell-1}(\boldsymbol{x})}^\top. \qquad \text{u @ v.T = rank 1 resulting matrix} \tag{6}$$

So the update $\Delta\boldsymbol{W}_\ell$ is rank-one with right singular vector $\boldsymbol{h}_{\ell-1}(\boldsymbol{x})$. To verify the claim, observe that:

$$\|\Delta\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 = \eta_\ell \cdot \left\|\nabla_{\boldsymbol{h}_\ell(\boldsymbol{x})}\mathcal{L}\right\|_2 \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2^2 = \|\Delta\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2. \tag{7}$$

With the claims established, we can now get lower bounds on hidden vector size which serve to verify Desideratum 1. The features at initialization scale correctly as:

$$\|\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \Theta\left(\|\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2\right) = \Theta(\sqrt{n_\ell}), \tag{8}$$

where we have first used Claim 1 and then inserted Condition 1. To bound the size of $\Delta\boldsymbol{h}_\ell(\boldsymbol{x})$, let us first observe from Equation (3) that

$$\Delta\boldsymbol{h}_\ell(\boldsymbol{x}) = \Delta\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x}) + \boldsymbol{W}_\ell \Delta\boldsymbol{h}_{\ell-1}(\boldsymbol{x}) + \Delta\boldsymbol{W}_\ell \Delta\boldsymbol{h}_{\ell-1}(\boldsymbol{x}). \tag{9}$$

So long as the first term $\Delta\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x})$ does not perfectly cancel with the latter two, we have that:

$$\|\Delta\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \Omega(\|\Delta\boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_{\ell-1}\|_2) = \Omega(\sqrt{n_\ell}), \tag{10}$$

where in the last step we have inserted Condition 1. Combining Equation (10) with our matching upper bound from Equation (5), we conclude that $\|\Delta\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \Theta(\sqrt{n_\ell})$ as desired.

We have achieved both clauses of Desideratum 1 at layer ℓ, completing a recursive step from layer $\ell - 1$. The norm of the input $\boldsymbol{x}$ is by assumption the correct size to serve as a base case, and thus we recursively have the correct feature scaling at all layers.

[3]Note that $\boldsymbol{x}$ in Claim 2 is the same input that induced the gradient $\Delta\boldsymbol{W}_\ell$ in the previous step. Claim 2 is generally not an equality if this not true.

3.1.1 Key intuitions

We now pause to discuss key intuitions from the above argument which will carry through to the general case.

Weight updates are low-rank and aligned. An important observation is that weight updates are highly structured: they have low rank and align to incoming vectors. This motivates the spectral norm (which is the degree by which a matrix scales a "perfectly aligned" vector) as the correct measure of size.

Spectral variables enable simpler scaling analysis. Most prior work on hyperparameter scaling (Yaida, 2022; Yang & Hu, 2021b) discusses layerwise initialization scales $\{\sigma_\ell\}$ and learning rates $\{\eta_\ell\}$ as the primary variables—although there are exceptions (Bernstein et al., 2020a). By contrast, we work directly in terms of quantities that these hyperparameters regulate: the spectral norms of $\boldsymbol{W}_\ell$ and $\Delta\boldsymbol{W}_\ell$. This enables us to determine the sizes of hidden vectors and their updates quite easily, whereas the same calculation in terms of σ_ℓ and η_ℓ is more involved. Section 4 shows how to recover σ_ℓ and η_ℓ from our spectral scaling condition.

3.2 Extensions: additional gradient steps, nonlinearities, and multiple examples

We now extend our warmup example to successively more complex settings, ultimately recovering the general case. As we add back complexity, our spectral scaling condition will remain sufficient to achieve feature evolution of the proper size, and key intuitions from our warmup will continue to hold up to minor modifications. Each extension requires making a natural assumption. We empirically verify these assumptions for a deep MLP in Appendix C.

3.2.1 Additional gradient steps

Our warmup argument relied on two properties of $\boldsymbol{W}_\ell$ at initialization: its spectral norm being the correct size (Condition 1) and it passing forward features $\boldsymbol{h}_\ell(\boldsymbol{x})$ of the correct size (Claim 2). Fortunately, it is easily seen that these two properties remain true of $\boldsymbol{W}_\ell$ after a gradient step, and we can therefore treat the second (and later) steps exactly as we did the first. This follows quickly given the following assumption.

Assumption 1. Updates do not perfectly cancel initial quantities. That is: I feel like this is reasonable

$$\|\boldsymbol{W}_\ell + \Delta\boldsymbol{W}_\ell\|_* = \Theta\left(\|\boldsymbol{W}_\ell\|_* + \|\Delta\boldsymbol{W}_\ell\|_*\right) \tag{11}$$

$$\|\boldsymbol{h}_\ell(\boldsymbol{x}) + \Delta\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \Theta(\|\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 + \|\Delta\boldsymbol{h}_\ell(\boldsymbol{x})\|_2). \tag{12}$$

The sort of perfect cancellation required to violate this assumption will be rare in practice (and adding a small amount of randomness to the learning rate η_ℓ will fix any occurrence with high probability). It follows that $\|\boldsymbol{W}_\ell + \Delta\boldsymbol{W}_\ell\|_* = \Theta(\sqrt{n_\ell/n_{\ell-1}})$ and $\|\boldsymbol{h}_\ell(\boldsymbol{x}) + \Delta\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \Theta(\sqrt{n_\ell})$. With these facts in place, the same argument we used in Section 3.1 for the first step ensures that Desideratum 1 also holds at later steps.

3.2.2 Nonlinearities

We now add a nonlinearity ϕ to each layer of our MLP. The modified forward recursion relation is:

$$\boldsymbol{h}_\ell(\boldsymbol{x}) = \boldsymbol{W}_\ell \boldsymbol{h}'_{\ell-1}(\boldsymbol{x}), \qquad \boldsymbol{h}'_\ell(\boldsymbol{x}) = \phi(\boldsymbol{h}_\ell(\boldsymbol{x})); \qquad \text{for } \ell = 2, \ldots, L-1, \tag{13}$$

with base case $\boldsymbol{h}_1(\boldsymbol{x}) = \boldsymbol{W}_1\boldsymbol{x}$ and output $\boldsymbol{h}_L(\boldsymbol{x}) = \boldsymbol{W}_L\boldsymbol{h}'_{L-1}(\boldsymbol{x})$. We assume that the hidden features before and after the application of the nonlinearity are of the same scale:

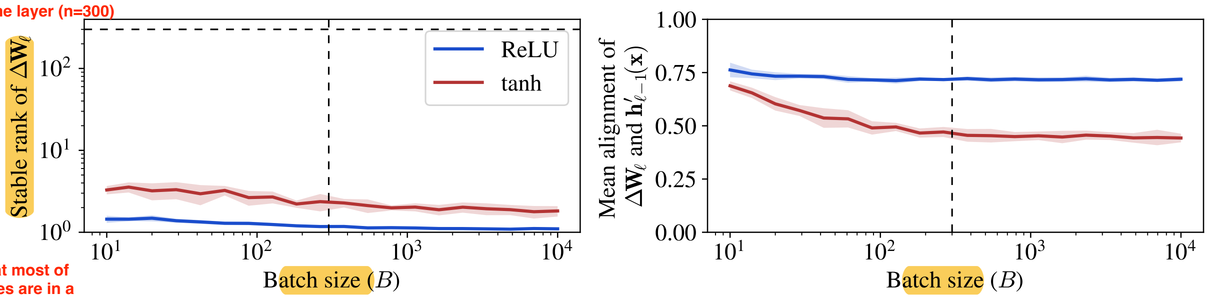
Assumption 2. $\|\boldsymbol{h}'_\ell(\boldsymbol{x})\|_2 = \Theta\left(\|\boldsymbol{h}_\ell(\boldsymbol{x})\|_2\right)$. Mmmm - I guess

This is the expected behavior for most activation functions (which are designed to take in order-one inputs and return outputs which neither explode nor uniformly vanish) and seems like a reasonable assumption. As in the linear case, $\Delta\boldsymbol{W}_\ell$ will be rank-one and align to incoming signal as:

$$\left\|\Delta\boldsymbol{W}_\ell \boldsymbol{h}'_{\ell-1}(\boldsymbol{x})\right\|_2 = \|\Delta\boldsymbol{W}_\ell\|_* \cdot \left\|\boldsymbol{h}'_{\ell-1}(\boldsymbol{x})\right\|_2 \tag{14}$$

at each step. All our scaling arguments from the linear case therefore carry through: the term $\Delta\boldsymbol{W}_\ell\boldsymbol{h}'_{\ell-1}(\boldsymbol{x})$ is sufficient to induce a change $\Delta\boldsymbol{h}_\ell(\boldsymbol{x})$ satisfying Desideratum 1. (The other terms which depend on $\Delta\boldsymbol{h}'_{\ell-1}(\boldsymbol{x})$ will be no larger.) Condition 1 therefore still achieves correctly-scaled feature evolution.

Max rank is the size of the layer (n=300)



This means that most of the grad updates are in a singular direction

Effective rank is a measure of the spread or dimensionality of a matrix using its singular values

Figure 1: **Gradient updates have low effective rank and high-alignment with incoming hidden vectors even at large batch size in MLPs on CIFAR-10.** We randomly initialize MLPs with depth $L = 3$, hidden widths $n_1 = n_2 = 300$, and ReLU and tanh activation functions. We then compute gradient updates $\Delta \boldsymbol{W}_\ell$ for layer $\ell = 2$ on randomly-sampled size-B subsets of CIFAR-10. **Left.** As a measure of effective rank, we report the *stable rank* $\operatorname{srank}(\Delta \boldsymbol{W}_\ell) := \|\Delta \boldsymbol{W}_\ell\|_F^2 \,/\, \|\Delta \boldsymbol{W}_\ell\|_*^2$ of gradient updates. The stable rank remains less than 10 even when B is large, which is much less than its maximal possible value of $\min(n_1, n_2) = 300$. **Right.** We report the average alignment $\|\Delta \boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x})\|_2 \,/\, \|\Delta \boldsymbol{W}_\ell\|_* \left\|\boldsymbol{h}'_{\ell-1}(\boldsymbol{x})\right\|_2$ of the weight update $\Delta \boldsymbol{W}_\ell$ to the incoming vector $\boldsymbol{h}'_{\ell-1}(\boldsymbol{x})$, averaged over $\boldsymbol{x}$ from the batch. Observe that alignment does not decay substantially with large batch size. Dashed lines on both axes in both subplots show the network width. Shaded regions denote one standard deviation of variation over random initializations and batches. Curves look similar after training. See Appendix A for experimental details.

3.2.3 Batch size greater than one

When training on a minibatch $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^B$ with size $B > 1$, each gradient step is simply an average of the steps on each example:

$$\Delta \boldsymbol{W}_\ell = \frac{1}{B} \sum_{i=1}^{B} \Delta \boldsymbol{W}_\ell^{(i)}, \tag{15}$$

matrix rank is additive under direct sum
rank(A + B) ≤ rank(A) + rank(B)
So, this generally means that the rank will grow

Intuitively this makes sense, as you expect the number of important update directions to be larger as the number of samples grows

where the sum runs over the minibatch index and $\Delta \boldsymbol{W}_\ell^{(i)}$ denotes the update that would result from a step on the single example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. While $\Delta \boldsymbol{W}_\ell$ is generally no longer rank one and cannot perfectly align to all B incoming vectors, Equation 15 makes it clear that at least one summand will align to each $\boldsymbol{h}_\ell(\boldsymbol{x}_i)$ from the batch. We first assume that this term is not perfectly cancelled by the others:

Assumption 3. $\|\Delta \boldsymbol{W}_\ell \boldsymbol{h}_\ell(\boldsymbol{x}_i)\|_2 = \Theta(\|\frac{1}{B} \Delta \boldsymbol{W}_\ell^{(i)} \boldsymbol{h}_\ell(\boldsymbol{x}_i)\|_2)$.

We additionally make the assumption that the batch size is fixed and independent of width:

Assumption 4. The batch size is width-independent: $B = \Theta(1)$.

Combining these assumptions, we find that $\|\Delta \boldsymbol{W}_\ell \boldsymbol{h}_\ell(\boldsymbol{x}_i)\|_2 = \Theta\left(\|\Delta \boldsymbol{W}_\ell\|_* \cdot \|\boldsymbol{h}_\ell(\boldsymbol{x}_i)\|_2\right)$: the batch update is aligned with incoming signal in a scaling sense. Our previous scaling arguments in fact needed only alignment in this "big-Θ" sense (as opposed to the perfect rank-one sense of Claim 2), and thus they still carry through: our spectral scaling condition continues to suffice to achieve proper feature learning as per Desideratum 1.

Empirical observation: low-rank structure remains at large batch size. Surprisingly, we observe numerically that MLP updates remain low (effective) rank and aligned with incoming vectors *even at large batch size B*. This is demonstrated in Figure 1.

They're highlighting this because the batch size mean that the grads are no longer rank 1 (and I think that the assumption is that is may increase with size)

3.3 Adam and other adaptive optimizers

Adam (like most adaptive optimizers in deep learning) processes gradients into updates via an entrywise function. For example, the momentum-less version of Adam is SignSGD (Bernstein et al., 2018) which just applies the sign function to each gradient entry. Via less elementary arguments detailed in Appendix B, all the above discussion holds for $\Delta \boldsymbol{W}_\ell$ calculated from these optimizers. In a gist, the main ingredients are 1) the nontrivial insight from *Tensor Programs* that gradients look like outer products of iid vectors, and 2) the fact that entrywise processing preserves the Frobenius norm of such a matrix up to multiplicative constants when width is large.

3.4 Uniqueness of spectral scaling condition

Technically, Desideratum 1 can be satisfied by scalings other than Condition 1. For example, if one implements Condition 1 at the first layer but sets $\|\Delta \boldsymbol{W}_\ell\|_* = 0$ for layers $\ell = 2, \ldots, L$, then Desideratum 1 still holds. However, Condition 1 is the unique *maximal* scaling: if any of $\|\Delta \boldsymbol{W}_\ell\|_*$ or $\|\boldsymbol{W}_\ell\|_*$ exceeds Condition 1, then training will blow up as width is increased. See Appendix B for a proof.

Matrix	**Shape**	**Stable Rank**	**Spectral Norm**	**Frobenius Norm**	**RMS Entry Size**
$\boldsymbol{W}_\ell$	$n \times n$	n = (Fro^2)/(Spec^2)	1	$\sqrt{n}$	$\sqrt{1/n}$
$\Delta\boldsymbol{W}_\ell$	$n \times n$	1	1	1	$1/n$
$\boldsymbol{W}_\ell$	$1 \times n$	1	$\sqrt{1/n}$	$\sqrt{1/n}$	$1/n$
$\Delta\boldsymbol{W}_\ell$	$1 \times n$	1	$\sqrt{1/n}$	$\sqrt{1/n}$	$1/n$
$\boldsymbol{W}_\ell$	$n_\ell \times n_{\ell-1}$	$\min(n_\ell, n_{\ell-1})$	$\sqrt{\frac{n_\ell}{n_{\ell-1}}}$	$\sqrt{\min(n_\ell, n_{\ell-1}) \cdot \frac{n_\ell}{n_{\ell-1}}}$	$\sqrt{\frac{\min(n_\ell,n_{\ell-1})}{n_\ell \times n_{\ell-1}} \cdot \frac{n_\ell}{n_{\ell-1}}}$
$\Delta\boldsymbol{W}_\ell$	$n_\ell \times n_{\ell-1}$	1	$\sqrt{\frac{n_\ell}{n_{\ell-1}}}$	$\sqrt{\frac{n_\ell}{n_{\ell-1}}}$	$\sqrt{\frac{1}{n_\ell \times n_{\ell-1}} \cdot \frac{n_\ell}{n_{\ell-1}}}$

The wider (larger) the network, the smaller the spectral and Frobenius norm

Table 1: **Conversion between matrix norms for initial weights $\boldsymbol{W}_\ell$ and updates $\Delta\boldsymbol{W}_\ell$.** Results are first given for square matrices and vectors, then finally generalized to matrices of all shapes. Table entries denote the Θ-scaling of each quantity. To obtain the Frobenius norm, one multiplies the spectral norm by the square root of the stable rank. To obtain the root-mean-square entry size, one divides the Frobenius norm by the square root of the number of entries. Notice that a $n \times n$ initial matrix $\boldsymbol{W}_\ell$ with $\Theta(1)$ spectral norm has $\Theta(\sqrt{1/n})$ entry size, whereas a $n \times n$ update $\Delta\boldsymbol{W}_\ell$ with $\Theta(1)$ spectral norm has only $\Theta(1/n)$ entry size, which is a factor $\sqrt{n}$ smaller! The truth of the statement "the weights move only a negligible amount in wide neural networks" thus depends on whether this change is measured in entry size or spectral norm.

4 Efficient implementation of the spectral scaling condition

We have thus far proposed the *spectral scaling condition* (Condition 1) and argued in Section 3 that this condition induces feature learning. We now explain how this condition can be implemented in practice. Several implementation strategies are possible because relations exist between different notions of matrix norm (Section 3.4 provides a summary). In this section, we first discuss an implementation via direct spectral normalization, and ultimately recover a more standard setup in which each layer's weight matrix is controlled by an initialization scale σ_ℓ and a (SGD) learning rate η_ℓ as follows:

Parametrization 1 (Spectral parametrization)**.** *We claim that the spectral scaling condition (Condition 1) is satisfied and feature learning is achieved (as per Desideratum 1) if the initialization scale and learning rate of each layer ℓ are chosen according to:*

$$\sigma_\ell = \Theta\left(\frac{1}{\sqrt{n_{\ell-1}}}\min\left\{1, \sqrt{\frac{n_\ell}{n_{\ell-1}}}\right\}\right); \qquad \eta_\ell = \Theta\left(\frac{n_\ell}{n_{\ell-1}}\right).$$

Naïve method: direct spectral normalization. The most naïve way to impose Condition 1 is to directly normalize the relevant quantities by their spectral norm. For instance, to initialize the weight matrix $\boldsymbol{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ at the ℓth layer to have spectral norm $\sqrt{n_\ell/n_{\ell-1}}$, one could sample a temporary matrix $\boldsymbol{W}'_\ell$

using any standard initializer and then re-normalize according to:

$$\boldsymbol{W}_\ell = \sigma\sqrt{\frac{n_\ell}{n_{\ell-1}}} \times \frac{\boldsymbol{W}'_\ell}{\|\boldsymbol{W}'_\ell\|_*}, \tag{16}$$

where $\sigma = \Theta(1)$ is a width-independent prefactor. Similarly, to ensure that the gradient step $\Delta\boldsymbol{W}_\ell$ also has a spectral norm of the proper size, one might spectrally normalize the gradient according to:

$$\Delta\boldsymbol{W}_\ell = -\eta\sqrt{\frac{n_\ell}{n_{\ell-1}}} \times \frac{\nabla_{\boldsymbol{W}_\ell}\mathcal{L}}{\|\nabla_{\boldsymbol{W}_\ell}\mathcal{L}\|_*}, \tag{17}$$

where $\eta = \Theta(1)$ is a width-independent prefactor.

It is quick to check that normalizing as in Equations (16) and (17) will satisfy Condition 1, but computing spectral norms is expensive and (as we will now show) can be avoided entirely by working out how $\|\boldsymbol{W}'_\ell\|_*$ and $\|\nabla_{\boldsymbol{W}_\ell}\mathcal{L}\|_*$ will scale and dividing by the appropriate factor.

Random initialization. Let us suppose that, as is common practice, $\boldsymbol{W}_\ell$ is initialized as $\boldsymbol{W}_\ell = \sigma_\ell \cdot \boldsymbol{W}'_\ell$, where all elements of $\boldsymbol{W}'_\ell$ are initialized i.i.d. from a normal distribution with mean zero and unit variance. The spectral norm of a matrix thus constructed is roughly $\|\boldsymbol{W}_\ell\|_* \approx \sigma_\ell \cdot (\sqrt{n_\ell} + \sqrt{n_{\ell-1}})$ (Rudelson & Vershynin, 2010; Vershynin, 2018). To get the desired scaling $\|\boldsymbol{W}_\ell\|_* = \Theta(\sqrt{n_\ell/n_{\ell-1}})$, we need merely choose $\sigma_\ell = \Theta(\sqrt{n_\ell/n_{\ell-1}} \cdot (\sqrt{n_\ell} + \sqrt{n_{\ell-1}})^{-1})$. Simplifying within the $\Theta(\cdot)$, we arrive at σ_ℓ scaled as in the spectral parametrization (Parametrization 1). Initializing weights with a prefactor σ_ℓ scaling in this manner achieves the correct spectral norm of $\boldsymbol{W}_\ell$. We note that the constant factor suppressed by the $\Theta(\cdot)$ here will usually be small—for example, a prefactor of $\sqrt{2}$ agrees with typical practice for ReLU networks at most layers. If $\boldsymbol{W}'_\ell$ is instead a random semi-orthogonal matrix, then we can simply use a prefactor $\sigma_\ell = \Theta(\sqrt{n_\ell/n_{\ell-1}})$.

Gradient updates. Here we give two methods for obtaining weight updates $\Delta\boldsymbol{W}_\ell$ with the correct spectral norm. The first method is to note that, for a matrix with low stable rank, the Frobenius norm scales like the spectral norm (and is cheap to compute), so we may simply approximate $\|\nabla_{\boldsymbol{W}_\ell}\mathcal{L}\|_* \approx \|\nabla_{\boldsymbol{W}_\ell}\mathcal{L}\|_F$ and use Equation (17) directly. This approach is useful if one wants to avoid worrying about width-scaling pre-factors entirely. The second method—on which we will spend more time—is to make standard updates

$$\Delta\boldsymbol{W}_\ell = -\eta_\ell \nabla_{\boldsymbol{W}_\ell}\mathcal{L} \tag{18}$$

and find a scaling of η_ℓ such that $\|\Delta\boldsymbol{W}_\ell\|_* = \Theta(\sqrt{n_\ell/n_{\ell-1}})$.

The main challenge lies in finding the scaling of the gradient $\|\nabla_{\boldsymbol{W}_\ell}\mathcal{L}\|_*$. Note that we expect each gradient update $\Delta\boldsymbol{W}_\ell$ to induce a change $\|\Delta\boldsymbol{h}_L(\boldsymbol{x})\|_2 = \Theta(\sqrt{n_L})$ in the output which induces a change $\Delta\mathcal{L} = \Theta(1)$ for common loss functions $\mathcal{L}$. Taylor expanding the loss to first order, we also expect that

$$\Delta\mathcal{L} = \Theta(\langle\Delta\boldsymbol{W}_\ell, \nabla_{\boldsymbol{W}_\ell}\mathcal{L}\rangle) = \Theta(\|\Delta\boldsymbol{W}_\ell\|_F \cdot \|\nabla_{\boldsymbol{W}_\ell}\mathcal{L}\|_F) = \Theta\left(\|\Delta\boldsymbol{W}_\ell\|_* \cdot \|\nabla_{\boldsymbol{W}_\ell}\mathcal{L}\|_*\right), \tag{19}$$

where $\langle\cdot,\cdot\rangle$ denotes the trace inner product and we have used the facts that the two arguments of the inner product are (a) proportional to each other and (b) low-rank. Inserting $\Delta\mathcal{L} = \Theta(1)$ and the spectral scaling condition $\|\Delta\boldsymbol{W}_\ell\|_* = \Theta(\sqrt{n_\ell/n_{\ell-1}})$, we can conclude that

$$\|\nabla_{\boldsymbol{W}_\ell}\mathcal{L}\|_* = \Theta(\sqrt{n_{\ell-1}/n_\ell}). \tag{20}$$

This result may also be reached via direct layerwise-recursive analysis of the size of the gradient. Returning to Equation (18), we now see that we achieve a properly-scaled update $\|\Delta\boldsymbol{W}_\ell\|_* = \Theta(\sqrt{n_\ell/n_{\ell-1}})$ if we take $\eta_\ell = \Theta\left(n_\ell/n_{\ell-1}\right)$ as prescribed by Parametrization 1.

In summary: training with layerwise initialization σ_ℓ and learning rate η_ℓ scaled as in our Parametrization 1 will implement the spectral scaling condition and give features and feature evolution of the correct size.

5 Comparisons to existing parametrizations

We have shown that training in accordance with our *spectral scaling condition* (Condition 1) at every layer suffices to achieve correctly-scaled feature evolution, and we have given width scalings for layerwise hyperparameters σ_ℓ and η_ℓ that suffice to put this condition into action in a standard deep learning paradigm (Parametrization 1). Here we compare this "spectral parametrization" with popular parametrizations. We find that our parametrization recovers the "maximal update parametrization" (μP) at all layers and is different to other parametrizations.

5.1 Comparison with "maximal update parametrization"

Maximal update parametrization (μP) was recently proposed as a scaling rule that retains feature learning even at infinite width. μP as given in Table 3 of Yang et al. (2021) may be recovered from our Parametrization 1 by setting $n_0 = n_L = 1$ and $n_1 = n_2 = \ldots = n_{L-1}$. Our Parametrization 1 actually streamlines and generalizes μP: we provide a unifying treatment for any rectangular matrix, rather than treating input, hidden and output layers separately. In other words, from a spectral point of view, no layer is special. Our parametrization also includes scaling with respect to the input dimension n_0 and output dimension n_L (as opposed to neglecting them as agnostic $\Theta(1)$ quantities) and treats hidden widths of unequal dimension ($n_{\ell-1} \neq n_\ell$).

5.2 Contrast to "standard parametrization"

At present, the vast majority of deep learning systems use either "Kaiming," "Xavier," or "LeCun" initialization (Glorot & Bengio, 2010; He et al., 2015; LeCun et al., 2002) with layer-independent learning rates. Generically, we refer to this as "standard parametrization" (SP), where layerwise initialization and learning rates scale as:

$$\sigma_\ell = \Theta(1/\sqrt{n_{\ell-1}}) \qquad \text{and} \qquad \eta_\ell = \Theta(1). \tag{21}$$

Notice that SP initialization exceeds Parametrization 1 in any layer with fan-out smaller than fan-in. This includes the final layer in sufficiently wide networks. So, while Parametrization 1 implies that weight matrices have spectral norm $\Theta(\sqrt{\texttt{fan-out}/\texttt{fan-in}})$ at initialization, under SP the spectral norms of certain layers are initialized larger than this. This means that, under SP, network outputs can blow up if training aligns the layer inputs with the top singular subspaces (and in fact this alignment generally occurs).

5.3 Contrast to "neural tangent parametrization"

The neural tangent parametrization (NTP) of Jacot et al. (2018) parameterizes a weight matrix as $\boldsymbol{W}_\ell/\sqrt{n_{\ell-1}}$ where the entries of $\boldsymbol{W}_\ell$ are sampled iid standard normal, and applies gradient descent with a layer-independent step-size. Since dividing a weight matrix by $\sqrt{n_{\ell-1}}$ also divides the gradient of that layer by the same factor, NTP is equivalent to training in our setup with a layer-wise standard deviation and learning rate:

$$\sigma_\ell = \Theta(1/\sqrt{n_{\ell-1}}) \qquad \text{and} \qquad \eta_\ell = \Theta(1/n_{\ell-1}). \tag{22}$$

Comparing Equations (21) and (22), we see that NTP shares the same initialization scaling as SP but uses a smaller step-size. To see the deficiency of NTP, notice the output layer σ_L is $\sqrt{n_{L-1}}$ larger than Parametrization 1, so that by the linearity of backpropagation, the gradient to any middle layer $\boldsymbol{W}_\ell$ is also $\sqrt{n_{L-1}}$ larger. Then the $1/n_{\ell-1}$ learning rate in NTP induces a change $\Delta\boldsymbol{W}_\ell$ that is $\sqrt{n_{L-1}}/n_{\ell-1}$ (smaller) compared to Parametrization 1, which prescribes a learning rate of $\eta_\ell = 1$. Because Parametrization 1 guarantees $\|\Delta\boldsymbol{W}_\ell\|_* = \Theta(1)$, NTP causes $\Delta\boldsymbol{W}_\ell$ to vanish in spectral norm as hidden widths $n_{\ell-1}, n_L \to \infty$.

It bears noting that an MLP parameterized with the NTP can be made to undergo feature evolution by simply rescaling the network output (and appropriately scaling down the global learning rate) (Bordelon & Pehlevan, 2022; Chizat et al., 2019). This operation transforms the NTP into μP.

5.4 Contrast to "Frobenius-normalized updates"

Condition 1 mandates that the spectral norm of the update at layer ℓ be proportional to the spectral norm of the weight matrix to which it is applied: $\|\Delta\boldsymbol{W}_\ell\|_* \propto \|\boldsymbol{W}_\ell\|_*$. This contrasts with a body of optimization work (Bernstein et al., 2020a;b; Liu et al., 2021; Shazeer & Stern, 2018; You et al., 2017) that has suggested instead that the *Frobenius norm* of the update at layer ℓ be proportional to the *Frobenius norm* of the weight matrix to which it is applied: $\|\Delta\boldsymbol{W}_\ell\|_F \propto \|\boldsymbol{W}_\ell\|_F$. For instance, Bernstein et al. (2020a) analysed the operator structure of deep neural networks and wrote down perturbation bounds on network activations in terms of perturbations to the weight matrices at each layer, and used these perturbation bounds to motivate making updates that are small in Frobenius norm. The flaw in that analysis is the assumption that weight matrices and their perturbations have identical conditioning structure (Bernstein et al., 2020a, Condition 2 of Theorem

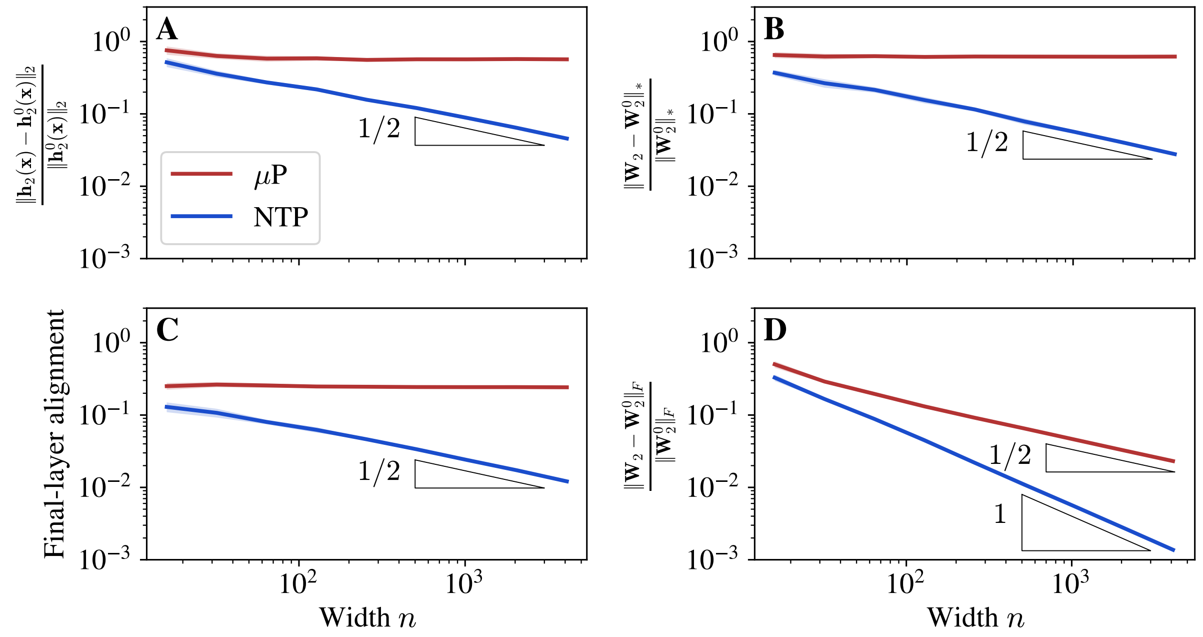


Figure 2: **Spectral quantities are $\Theta(1)$ under μP but decay with width under NTP.** We train multilayer perceptrons of varying width, and plot the following quantities computed between the initial and final network: **(A)** Average relative change in features. **(B)** Relative change in weights in spectral norm. **(C)** Final layer alignment with incoming vectors (Equation (24)). **(D)** Relative change in weights in Frobenius norm. Labeled triangles show predicted powerlaw slopes. Shaded regions show one standard deviation over random data and initialization.

1), when in reality weight matrices have high stable rank and updates have $\Theta(1)$ stable rank. In light of this fact, the Frobenius-normalized proportionality rule should be modified to $\|\Delta \boldsymbol{W}_\ell\|_F \propto \|\boldsymbol{W}_\ell\|_F / \sqrt{\min(n_\ell, n_{\ell-1})}$ in order to see proper feature evolution.

6 Demonstration: μP versus NTP

Here we discuss a simple, illustrative experiment in which we directly verify that μP obeys our spectral scaling condition (Condition 1) and achieves leading-order feature evolution (Desideratum 1) and the NTP does not. We do so via direct measurement of spectral quantities in MLPs of varying width trained on the same task. Appendix A provides full experimental details, and results are plotted Figure 2.

Model and data. We train MLPs with $L = 3$ linear layers, no biases, and ReLU activations. For demonstration purposes, we train on a small subset of $B = 200$ examples from a two-class subset of CIFAR-10. The model has input dimension $n_0 = 3072$, output dimension $n_3 = 1$, and uniform hidden dimension $n_1 = n_2 = n$, with n varied between training runs. We initialize and train each MLP twice with hyperparameters obtained using μP and NTP scalings, respectively. We train networks of widths $n \in [16, 4096]$ to near-zero training loss. After training, we compute various spectral quantities which we now discuss.

Norms of feature updates. We measure the average relative change in features over training:

$$\mathbb{E}_{\boldsymbol{x}}\left[\frac{\|\boldsymbol{h}_2(\boldsymbol{x}) - \boldsymbol{h}_2^0(\boldsymbol{x})\|_2}{\|\boldsymbol{h}_2^0(\boldsymbol{x})\|_2}\right], \tag{23}$$

where $\boldsymbol{h}_2^0(\boldsymbol{x})$ and $\boldsymbol{h}_2(\boldsymbol{x})$ are the second (preactivation) hidden vector at initialization and after training respectively, and the expectation is over samples $\boldsymbol{x}$ from the batch. As shown in Figure 2A, this feature evolution ratio remains roughly fixed as width d grows when using μP, in satisfaction of Desideratum 1. By contrast, it decays as $1/\sqrt{n}$ for the NTP as predicted by Lee et al. (2019).

Spectral norms of weight updates. We measure the relative change in weights in spectral norm: $\|\boldsymbol{W}_2 - \boldsymbol{W}_2^0\|_* / \|\boldsymbol{W}_2^0\|_*$, where $\boldsymbol{W}_2^0$ and $\boldsymbol{W}_2$ are the second weight matrix before and after training, respectively. As shown in Figure 2B, this ratio remains roughly fixed with width in the case of μP, in accordance with Condition 1. By contrast, it decays as $1/\sqrt{n}$ for the NTP.

Final-layer alignment. We measure the alignment of the final layer to incoming vectors as follows:

$$\text{Final-layer alignment} := \mathbb{E}_x \left[\frac{\|\boldsymbol{W}_3 \boldsymbol{h}_2'(\boldsymbol{x})\|_2}{\|\boldsymbol{W}_3\|_* \cdot \|\boldsymbol{h}_2'(\boldsymbol{x})\|_2} \right]. \tag{24}$$

This quantity is $\Theta(1/\sqrt{n})$ at initialization. As shown in Figure 2C, it grows to $\Theta(1)$ when using μP but remains $\Theta(1/\sqrt{n})$ when using the NTP.

Frobenius norms of weight updates. One often hears the claim that "the weights don't move" when training very wide neural networks. Here we show that the validity of this claim crucially depends on the choice of metric. In Figure 2D, we show that the Frobenius norm is deceptive: the net relative change $\|\boldsymbol{W}_2 - \boldsymbol{W}_2^0\|_F / \|\boldsymbol{W}_2^0\|_F$ can decay with width even when the relative change in spectral norm is constant. This provides crucial context for interpreting existing results in the literature (Lee et al., 2019, Figure 1).

7 Related work

μP was derived heuristically from spectral norm considerations in talks given by the first author in 2021 (Yang & Hu, 2021a). Earlier work (Bernstein et al., 2020a) derived a spectral analysis of feature learning based on perturbation bounds, but that work obtained the wrong scaling relation with network width due to a flawed conditioning assumption on gradients. Below, we review various strands of related work on feature learning and training strategies.

Parametrizations for wide neural networks. Much work has examined the scaling behavior of training dynamics of networks at large width. Together, works on the "neural tangent kernel" (NTK) limit (Jacot et al., 2018; Lee et al., 2019), the "mean-field" limit (Mei et al., 2019; Rotskoff & Vanden-Eijnden, 2022; Sirignano & Spiliopoulos, 2022), and the related "feature learning" limit (Geiger et al., 2020; Yaida, 2022; Yang & Hu, 2021b) (i.e. the μP limit), paint a rich picture of a family of possible infinite-width scalings. After healthy debate regarding the relative empirical performance of the (more analytically tractable) NTK limit and the feature learning limit, a recent consensus holds that learning features is usually beneficial in practical large-scale deep learning settings (Chizat et al., 2019; Fort et al., 2020; Vyas et al., 2022). Our spectral scaling analysis recovers the feature learning limit in a simpler manner than previous analyses.

Spectral normalization. Spectral normalization emerged as a form of weight normalization in the generative adversarial network literature (Miyato et al., 2018). This form of normalization acts on the weight matrices, and is used analogously to other normalization schemes such as batchnorm (Ioffe & Szegedy, 2015) and layernorm (Ba et al., 2016). In contrast to our spectral scaling condition (Condition 1), this method treats only the weights $\boldsymbol{W}_\ell$, not the updates $\Delta \boldsymbol{W}_\ell$. Furthermore, spectral normalization implementations typically set the spectral norm of weight matrices either to one or to a tunable hyperparameter (Farnia et al., 2019), and do not include the key factor of $\sqrt{n_\ell / n_{\ell-1}}$ in our Condition 1.

Operator theory of neural networks. Neural networks are constructed by composing linear operators with elementwise nonlinearities. One line of work studies this operator structure and how it behaves under perturbation to understand how step sizes should be set in gradient descent. For instance, Bernstein et al. (2020a) derive perturbation bounds on the maximum amount of feature change that can be induced by a gradient step in terms of the operator properties of the weight matrices. Meanwhile, Yang & Hu (2021b) study the operator structure of neural networks in the limit that width is taken to infinity, proposing a parametrization that obtains feature learning in this limit.

Optimization theory. A body of literature studies optimization algorithms for deep networks that take steps whose size is set relative to the weights to which they are applied (Bernstein et al., 2020a;b; Carbonnelle & Vleeschouwer, 2019; Liu et al., 2021; Shazeer & Stern, 2018; You et al., 2017). A particular focus has been placed on setting the Frobenius norm of update steps to be small relative to the Frobenius norm of the weight matrices (Bernstein et al., 2020a;b; Liu et al., 2021; You et al., 2017). A main practical takeaway of this paper is that the Frobenius norm should be replaced by the spectral norm to get proper width scaling. The source of the difference between Frobenius and spectral norm is that gradient updates tend to have low stable rank, as shown in Figure 1, while the weights themselves tend to have high stable rank.

8 Conclusion

We have presented an analysis of the dynamics of feature learning in deep neural networks, beginning with desired conditions on feature evolution (Desideratum 1) and culminating in the demonstration that these conditions may be achieved by simple scaling rules (Condition 1 and Parametrization 1) applied uniformly to each layer. Our analysis recovers and generalizes practically-important "feature-learning parametrizations" and provides a simple, unifying perspective on the question of parametrization in wide neural networks. For comparison, formal results derived under the *tensor programs* framework are given in Appendix B.

Our discussion has focused principally on MLPs for clarity, but our feature learning desideratum and spectral scaling condition can be directly applied to structured architectures. The spectral scaling condition may be applied to multi-index tensors as appear in convolutional architectures by applying the condition to appropriate "slices" of the full tensor. Simple application of our spectral scaling condition recovers μP scalings reported for these model classes (see e.g. Yang et al. (2021), Table 8 and Section J.2). We also give the hyperparameter scalings for biases (which are easily derived but omitted in the main text for clarity) in Appendix D. This architectural universality is also proven rigorously in Appendix B.

Finally, we note that under a natural redefinition of the ℓ^2-norm of a vector $\boldsymbol{v} \in \mathbb{R}^n$ incorporating a normalization prefactor, *all vector and matrix norms used in our spectral analysis* become $\Theta(1)$, permitting an elegant summary of our conclusions. We state and discuss this nondimensionalization procedure in Appendix E. This generalization permits the extension of our results to e.g. the input layer in language modeling, in which one-hot embeddings violate our assumption that $\|\boldsymbol{x}\|_2 = \Theta(\sqrt{n_0})$.

Acknowledgements

The authors thank Josh Albrecht, Blake Bordelon, Alex Wei, Nikhil Ghosh, and Dhruva Karkada for useful discussions and comments on the manuscript. JS gratefully acknowledges support from the National Science Foundation Graduate Fellow Research Program (NSF-GRFP) under grant DGE 1752814.

Author Contributions

GY developed our core insight regarding the utility of the spectral norm, produced our tensor programs theory (Appendix B), and aided in refinement of the paper. JS spearheaded the writing of the paper, led iteration towards simple analysis which communicates our spectral picture, and ran experiments. JB developed an early incarnation of the spectral picture (Bernstein et al., 2020a), contributed key insights simplifying our exposition including unifying all layers under single formulae, aided in writing the paper, and ran experiments.

References

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 1998. Cited on page 2.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Neural Information Processing Systems*, 2019. Cited on page 1.

Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: the silent alignment effect. In *International Conference on Learning Representations*, 2022. Cited on page 1.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016. Cited on page 12.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: Compressed Optimisation for Non-Convex Problems, February 2018. URL `https://arxiv.org/abs/1802.04434v3`. Cited on page 7.

Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming-Yu Liu. On the distance between two neural networks and the stability of learning. In *Neural Information Processing Systems*, 2020a. Cited on pages 2, 6, 10, 12, and 13.

Jeremy Bernstein, Jiawei Zhao, Markus Meister, Ming-Yu Liu, Anima Anandkumar, and Yisong Yue. Learning compositional functions via multiplicative weight updates. In *Neural Information Processing Systems*, 2020b. Cited on pages 10, 12, and 13.

Jeremy Bernstein, Chris Mingard, Kevin Huang, Navid Azizan, and Yisong Yue. Automatic Gradient Descent: Deep Learning without Hyperparameters. *arXiv:2304.05187*, 2023. Cited on page 1.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. In *Neural Information Processing Systems*, 2022. Cited on page 10.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Neural Information Processing Systems*, 2020. Cited on page 1.

Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 2021. Cited on page 1.

Simon Carbonnelle and Christophe De Vleeschouwer. Layer rotation: A surprisingly simple indicator of generalization in deep networks? In *ICML Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019. Cited on page 12.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Neural Information Processing Systems*, 2019. Cited on pages 10 and 12.

Nolan Dey, Gurpreet Gosal, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, Joel Hestness, et al. Cerebras-GPT: Open compute-optimal language models trained on the Cerebras wafer-scale cluster. *arXiv:2304.03208*, 2023. Cited on page 1.

Inderjit S. Dhillon and Joel A. Tropp. Matrix nearness problems with Bregman divergences. *SIAM Journal on Matrix Analysis and Applications*, 2008. Cited on page 2.

Farzan Farnia, Jesse Zhang, and David Tse. Generalizable adversarial training via spectral normalization. In *International Conference on Learning Representations*, 2019. Cited on page 12.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Neural Information Processing Systems*, 2020. Cited on pages 1 and 12.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020. Cited on page 12.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010. Cited on page 10.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *International Conference on Computer Vision*, 2015. Cited on page 10.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015. Cited on page 12.

Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *Neural Information Processing Systems*, 2018. Cited on pages 1, 10, 12, and 17.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. Cited on page 1.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Cited on page 17.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*. Springer, 2002. Cited on pages 2 and 10.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*, 2018. Cited on page 1.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Neural Information Processing Systems*, 2019. Cited on pages 1, 12, and 17.

Jaehoon Lee, Samuel S. Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. In *Neural Information Processing Systems*, 2020. Cited on page 1.

Yang Liu, Jeremy Bernstein, Markus Meister, and Yisong Yue. Learning by turning: Neural architecture aware optimisation. In *International Conference on Machine Learning*, 2021. Cited on pages 2, 10, 12, and 13.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: Dimension-free bounds and kernel limit. In *Conference on Learning Theory*, 2019. Cited on page 12.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for Generative Adversarial Networks. In *International Conference on Learning Representations*, 2018. Cited on page 12.

Arkady S. Nemirovsky and David B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley, 1983. Cited on page 2.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. Cited on page 2.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Neural Information Processing Systems*, 2016. Cited on page 21.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv:2204.06125*, 2022. Cited on page 1.

Grant M. Rotskoff and Eric Vanden-Eijnden. Trainability and accuracy of artificial neural networks: An interacting particle system approach. *Communications on Pure and Applied Mathematics*, 2022. Cited on page 12.

Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: Extreme singular values. In *International Congress of Mathematicians*, 2010. Cited on pages 5 and 9.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, 2018. Cited on pages 2, 10, and 12.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016. Cited on page 1.

Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of deep neural networks. *Mathematics of Operations Research*, 2022. Cited on page 12.

Jascha Sohl-Dickstein, Roman Novak, Samuel S. Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv:2001.07301*, 2020. Cited on page 1.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018. Cited on pages 5 and 9.

Nikhil Vyas, Yamini Bansal, and Preetum Nakkiran. Limitations of the NTK for understanding generalization in deep learning. *arXiv:2206.10012*, 2022. Cited on pages 1 and 12.

Sho Yaida. Meta-principled family of hyperparameter scaling strategies. *arXiv:2210.04909*, 2022. Cited on pages 6 and 12.

Greg Yang and Edward J. Hu. Feature learning in infinite-width neural networks. In *ICML 2021 Workshop on Over-parameterization: Pitfalls and Opportunities*, 2021a. `https://slideslive.com/38963058/feature-learning-in-infinitewidth-neural-networks`. Cited on page 12.

Greg Yang and Edward J. Hu. Tensor Programs IV: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, 2021b. Cited on pages 1, 3, 6, 12, and 18.

Greg Yang and Etai Littwin. Tensor programs IVb: Adaptive optimization in the ∞-width limit. *arXiv:2308.01814*, 2023. Cited on page 18.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor Programs V: Tuning large neural networks via zero-shot hyperparameter transfer. In *Neural Information Processing Systems*, 2021. Cited on pages 1, 10, and 13.

Yang You, Igor Gitman, and Boris Ginsburg. Scaling SGD batch size to 32K for ImageNet training. Technical Report UCB/EECS-2017-156, University of California, Berkeley, 2017. Cited on pages 1, 2, 10, 12, and 13.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations*, 2020. Cited on page 2.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, 2014. Cited on page 2.

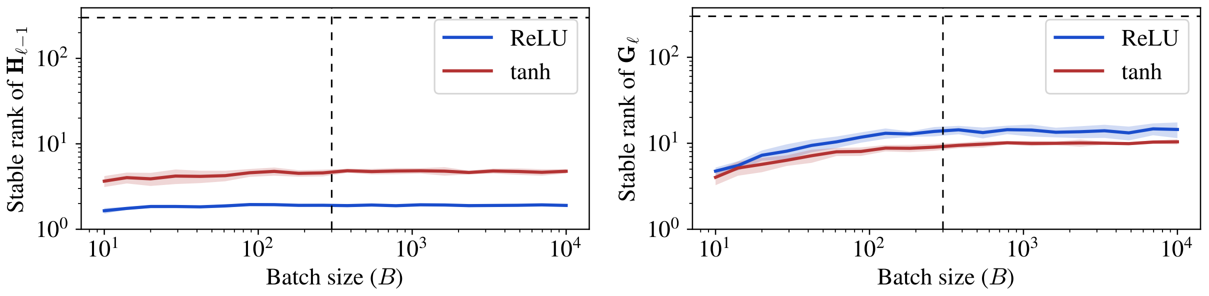


Figure 3: Stable ranks for the set of forward hidden vectors $\boldsymbol{H}'_{\ell-1}$ and the set of backward hidden vectors $\boldsymbol{G}_\ell$ for $\ell = 2$. Even at large batch size B, both are consistently much lower than their maximal possible values of $d = 300$. Dashed lines on both axes in both subplots show the network width d. Note that the stable rank of a matrix $\boldsymbol{M}$ is given by $\|\boldsymbol{M}\|_F^2 / \|\boldsymbol{M}\|_*^2$.

A Experimental details

Experimental details for Figure 1. We examine randomly-initialized MLPs with depth $L = 3$, widths $n_0 = 3072, n_1 = n_2 = d = 300, n_3 = 10$, and ReLU and tanh activation functions. We then pass single batches of CIFAR-10 data of varying size B through the model and compute a single gradient step $\Delta\boldsymbol{W}_\ell$ at layer $\ell = 2$ with arbitrary learning rate. We then compute the stable rank of $\Delta\boldsymbol{W}_\ell$ and the alignment metric

$$\mathbb{E}_{\boldsymbol{x}}\left[\frac{\|\Delta\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{x}_i)\|_2}{\|\Delta\boldsymbol{W}_\ell\|_* \, \|\boldsymbol{h}'_{\ell-1}(\boldsymbol{x}_i)\|_2}\right], \tag{25}$$

where the expectation is taken over $\boldsymbol{x}$ from the batch. Shaded envelopes in Figure 1 denote one standard deviation with respect to both random network initialization and random batch selection over 10 trials.

We conducted an additional experiment to try to ascertain the source of the low effective rank of $\Delta\boldsymbol{W}_\ell$. Let $\boldsymbol{H}'_{\ell-1} = [\boldsymbol{h}'_1(\boldsymbol{x}_1) \dots \boldsymbol{h}'_1(\boldsymbol{x}_B)] \in \mathbb{R}^{d\times B}$ be the matrix formed from stacking the $\ell = 1$ post-nonlinearity hidden vectors from the full batch, and let $\boldsymbol{G}_\ell = [\boldsymbol{g}_2(\boldsymbol{x}_1) \dots \boldsymbol{g}_2(\boldsymbol{x}_B)] \in \mathbb{R}^{d\times B}$ with $\boldsymbol{g}_\ell(\boldsymbol{x}) = \nabla_{\boldsymbol{h}_\ell(\boldsymbol{x})}\mathcal{L}$ be the matrix formed from stacking the loss gradients at layer $\ell = 2$ from the full batch. It is the case that $\Delta\boldsymbol{W}_\ell \propto \boldsymbol{G}_\ell \boldsymbol{H}'^{\top}_{\ell-1}$, and if $\Delta\boldsymbol{W}_\ell$ has low stable rank, it is a reasonable guess that either $\boldsymbol{H}'_{\ell-1}$ and $\boldsymbol{G}_\ell$ also has low stable rank. In fact, we find that *both* $\boldsymbol{H}'_{\ell-1}$ and $\boldsymbol{G}_\ell$ have low stable rank, as shown in Figure 3.

Experimental details for Figure 2. We train MLPs with depth $L = 3$, widths $n_0 = 3072, n_1 = n_2 = d, n_3 = 1$, and ReLU activation functions. The data consists of 200 samples from CIFAR-10 (Krizhevsky, 2009) from only the classes `airplane` and `automobile` and uses ± 1 targets.

We use two different hyperparameter schemes as follows. To implement μP, we take

$$\sigma_1 = \sqrt{\frac{2}{n_0}}; \quad \sigma_2 = \sqrt{\frac{2}{d}}; \quad \sigma_3 = \frac{\sqrt{2}}{d}; \quad \eta_1 = \eta \cdot \frac{d}{n_0}; \quad \eta_2 = \eta; \quad \eta_3 = \eta \cdot \frac{1}{d}, \tag{μP}$$

with global learning rate $\eta = 0.1$. To implement NTP, we follow Jacot et al. (2018) and Lee et al. (2019):

$$\sigma_\ell = \sqrt{\frac{2}{n_{\ell-1}}}; \qquad \eta_\ell = \eta \cdot \frac{1}{n_{\ell-1}} \tag{NTP}$$

at all layers, again with $\eta = 0.1$. These parameterizations are equivalent at $d = 1$, which lets us view each parameterization as a particular scaling prescription applied to a narrow base network.

We train full-batch for 10^4 steps, which is sufficient for all widths to drop below 0.01 training loss on average by the end of training. We do not expect that training for many more steps would saliently change the resulting plots. Shaded envelopes in Figure 2 denote one standard deviation with respect to both random network initialization and random batch selection over 10 experiment trials.

It is perhaps worth emphasizing that these experiments worked much better than they had to. Our theory strictly applies only to the case of a small number of gradient steps relative to network width, but the net updates shown in each subplot of Figure 2 reflect the accumulation of thousands of gradient steps, a number which is larger than network width in all cases. We were thus surprised by the very clear agreement of this experiment with predicted power laws.

B Tensor programs theory

For simplicity, assume all hidden widths $n_1 = \cdots = n_{L-1}$ are the same. Borrowing from Yang & Hu (2021b), an *abc-parametrization* is just a recipe for scaling (as powers of width) the multiplier, initializer scale, and learning rate of all parameter tensors of a neural network. SP, NTP, and μP are all examples of abc-parametrizations. A *stable* abc-parametrization is one whose (pre)activations and output do not blow up with width at any step of training. Then (under the generous Assumption 5) the following theorem is our main result:

Theorem 1. *In μP, for almost every learning rate (in the measure-theoretic sense), Condition 1 is satisfied at any time during training for sufficiently large width. μP is the unique stable abc-parametrization with this property.*

The main insight leading to Theorem 1 is that: $\Delta \boldsymbol{W}_\ell / \|\Delta \boldsymbol{W}_\ell\|_F$ converges to a Hilbert-Schmidt operator (in an appropriate space) as width goes to infinity. This in turn implies:

Lemma 1. *Unless $\Delta \boldsymbol{W}_\ell = 0$, $\|\Delta \boldsymbol{W}_\ell\|_F / \|\Delta \boldsymbol{W}_\ell\|_* = \Theta(1)$ as width goes to infinity.*[4]

These statements are universal: they hold for any architecture and any adaptive optimizers representable by tensor programs (including convolutional neural networks, residual networks, and transformers, etc., as well as RMSProp, Adam, etc.), not just MLP and SGD.

B.1 Proofs

As we have explained the core intuitions of our spectral perspective of feature learning in the main text, here we focus on proving the most general result in the most concise way.

Our proofs will rely on the following notions defined in prior work:

- representable architecture (Yang & Littwin, 2023, Defn 2.9.1)
- matrix/vector/scalar parameters (Yang & Littwin, 2023, Defn 2.9.1)
- abcd-parametrization for representable architectures (Yang & Littwin, 2023, Defn 2.9.7)
- entrywise optimizer (Yang & Littwin, 2023, Sec 2.1)
- ket and iid-copy notation (Yang & Littwin, 2023, Sec 1.2)

Everything here follows under the following:

Assumption 5. Assume our architecture is representable by a Ne⊗orT program with pseudo-Lipschitz nonlinearities, trained by an entrywise optimizer with pseudo-Lipschitz update functions.

Random Initialization

The following is our main proposition for initialization:

Proposition 1. *In any abcd-parametrization, any matrix parameter $\boldsymbol{W}$ at random initialization has $\|\boldsymbol{W}\|_F / \|\boldsymbol{W}\|_* = \Theta(\sqrt{n})$, but any vector or scalar parameter $\boldsymbol{W}$ has $\|\boldsymbol{W}\|_F / \|\boldsymbol{W}\|_* = 1$.*

[4]It's possible to make precise quantitative statements here using a quantitative version of the Master Theorem but we won't be concerned with it here.

For a rectangular matrix initialized with man 0 and std. dev. S, the Frobenius norm is ~ S* sqrt(in_dim * out_dim)

Proof. This is a claim about random matrices and follows from classical random matrix theory. Here's a quick sketch: If σ is the standard deviation of a matrix entry, then $\|\boldsymbol{W}\|_F \approx \sigma \cdot n$ from the central limit theorem, and $\|\boldsymbol{W}\|_* \approx 2\sigma \cdot \sqrt{n}$ as stated in the main text, from which the first part of the proposition follows. For vectors and scalars, the stated ratio is always 1. □

Matrix Updates

In a tensor program, each vector x of the program converges to a random variable $\llbracket x \rangle$ (called a *ket*) as width goes to infinity in the sense that the scaled inner product $\langle x, y\rangle / n$ converges almost surely to $\langle x \llbracket y\rangle = \mathbb{E}\llbracket x\rangle\llbracket y\rangle$. The kets form a Hilbert space $\mathcal{Z}$. Then any weight parameter converges to a linear operator from $\mathcal{Z}$ to $\mathcal{Z}$; any vector parameter converges to a linear operator from $\mathcal{Z}$ to $\mathbb{R}$ or $\mathbb{R}$ to $\mathcal{Z}$. (Any scalar parameter converges to a random real, but that is not too important here).

Proposition 2. *Consider any abcd-parametrization. For any matrix or vector parameter $\boldsymbol{W}$, at any step of training, $\Delta\boldsymbol{W} / \|\Delta\boldsymbol{W}\|_F$ converges to a Hilbert-Schmidt integral operator.*

Proof. This is trivial for vector parameters. For matrix parameters, observe $\widetilde{\Delta\boldsymbol{W}} = \Delta\boldsymbol{W} / \|\Delta\boldsymbol{W}\|_F$ is always a nonlinear outer product

$$\widetilde{\Delta\boldsymbol{W}} = Q(\boldsymbol{x}; \boldsymbol{y}; \boldsymbol{c})$$

for multi-vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ and multi-scalars $\boldsymbol{c}$ and some nonlinearity Q. Then the limit $\llbracket\widetilde{\Delta\boldsymbol{W}}\rrbracket$ acts on a ket $\llbracket z\rangle$ by

$$\llbracket\widetilde{\Delta\boldsymbol{W}}\rrbracket z\rangle = \mathbb{E}_{\boxed{1}}\, Q(\llbracket\boldsymbol{x}\rangle; \llbracket\boldsymbol{y}\rangle^{\boxed{1}}; \mathring{\boldsymbol{c}})\llbracket z\rangle^{\boxed{1}}$$

integrating over $\llbracket\boldsymbol{y}\rangle$ and $\llbracket z\rangle$. The Hilbert-Schmidt norm of $\llbracket\widetilde{\Delta\boldsymbol{W}}\rrbracket$ is

$$\left\|\llbracket\widetilde{\Delta\boldsymbol{W}}\rrbracket\right\|_{HS} = \sqrt{\mathbb{E}_{\boxed{0},\boxed{1}}\, Q(\llbracket\boldsymbol{x}\rangle^{\boxed{0}}; \llbracket\boldsymbol{y}\rangle^{\boxed{1}}; \mathring{\boldsymbol{c}})^2}$$

which is finite by the Master Theorem. Therefore $\llbracket\widetilde{\Delta\boldsymbol{W}}\rrbracket$ is Hilbert-Schmidt. □

Proposition 3. *In Proposition 2, for large enough width, unless $\Delta\boldsymbol{W} = 0$, both spectral norm and Frobenius norm of $\Delta\boldsymbol{W} / \|\Delta\boldsymbol{W}\|_F$ are $\Theta(1)$.*

This proposition implies Lemma 1.

Proof. This is obvious for Frobenius norm since it converges to the Hilbert-Schmidt norm of the operator limit.

However, the spectral norm cannot be expressed directly in such form. But, by definition of spectral norm and of the the ket space $\mathcal{Z}$, one can construct a nonzero vector z in an extension of the program that defined $\Delta\boldsymbol{W}$, such that

$$\left\|\llbracket\widetilde{\Delta\boldsymbol{W}}\rrbracket z\rangle\right\| = \theta \left\|\llbracket\widetilde{\Delta\boldsymbol{W}}\rrbracket\right\|_* \|\llbracket z\rangle\|,$$

for some $\theta \in (1/2, 1]$.[5] This implies, by the Master Theorem, that

$$\frac{\theta}{2} \frac{\left\|\widetilde{\Delta\boldsymbol{W}} z\right\|}{\|z\|} \leq \left\|\widetilde{\Delta\boldsymbol{W}}\right\|_*$$

for sufficiently large width. The LHS is $\Theta(1)$, so we are done.

□

Proposition 4. *In μP, at any fixed time during training, as width $\to \infty$, any matrix parameter $\boldsymbol{W}$ has spectral norm $\Theta(1)$ and Frobenius norm $\Theta(\sqrt{n})$.*

[5]In fact, we can pick $\theta \in [1 - \epsilon, 1]$ for any $\epsilon > 0$.

Proof. In μP, it's trivial to see that $\|\Delta \boldsymbol{W}\|_F = \Theta(1)$. So any nonzero $\Delta \boldsymbol{W}$ (at any point of training) has both spectral norm and Frobenius norm $\Theta(1)$ by the above.

Simple calculation then shows that $\boldsymbol{W}$ at any fixed time has $\Theta(\sqrt{n})$ since this is the case at initialization by Proposition 1. This means the quadratic mean of the singular values of $\boldsymbol{W}$ is $\Theta(1)$, so its max singular value must be $\Omega(1)$. But it furthermore must be $\Theta(1)$ because $\boldsymbol{W}$ at initialization and all of its updates have $O(1)$ spectral norm. □

Theorem 2. *In μP, for all but a measure-zero set of learning rates, Condition 1 is satisfied at any time during training for sufficiently large width. μP is the unique stable and faithful abcd-parametrization with this property.*

Proof. In μP, by Proposition 4, all matrix parameters satisfy Condition 1 no matter what the learning rate is. However, for vector parameters, it's possible for some specific learning rate to cause the weights to vanish after an update, but at most a (Lebesgue) measure-zero set of learning rates will cause this to happen. Assuming this vanishing does not happen, $\boldsymbol{W}$ has $\Theta(1)$ Frobenius norm at initialization and at all times during training. By Proposition 3, $\boldsymbol{W}$ also has $\Theta(1)$ spectral norm, so Condition 1 is satisfied. A similar but easier argument applies to all scalar parameters. This shows μP satisfies Condition 1.

Since any other stable and faithful parametrization essentially just rescales the initialization and the update, we see that no other parametrization can satisfy Condition 1. □

For SGD, all abc-parametrizations are equivalent to a faithful abcd-parametrization because Q is identity, so Theorem 2 recovers Theorem 1.

C Empirical checks of Assumptions 1 to 3

In Section 3, we first illustrated how Condition 1 satisfies Desideratum 1 in a minimal model—a deep linear network trained for one step on one sample—and then iteratively extended our argument to multiple steps, nonlinearities, and multiple samples. Each extension came with a mild assumption. These assumptions are intended to be natural conditions one expects from generic network dynamics.

In this section, we restate each assumption, explain further why it is expected to hold, and then present a validating experiment. All experiments use the same setup as that of Figure 2. Plotted quantities depending on the sample $\boldsymbol{x}$ are averaged over all $\boldsymbol{x}$ in the batch, with shaded envelopes showing one standard deviation of this mean over five experiment trials.

Assumption 1. *Updates do not perfectly cancel initial quantities. That is:*

$$\|\boldsymbol{W}_\ell + \Delta \boldsymbol{W}_\ell\|_* = \Theta\left(\|\boldsymbol{W}_\ell\|_* + \|\Delta \boldsymbol{W}_\ell\|_*\right) \tag{26}$$

$$\|\boldsymbol{h}_\ell(\boldsymbol{x}) + \Delta \boldsymbol{h}_\ell(\boldsymbol{x})\|_2 = \Theta(\|\boldsymbol{h}_\ell(\boldsymbol{x})\|_2 + \|\Delta \boldsymbol{h}_\ell(\boldsymbol{x})\|_2). \tag{27}$$

Recall that the cancellation of two high-dimensional tensors (i.e. matrices or vectors) tends to be unlikely as dimension grows (unless there is a good reason to expect cancellation, which there is not here). For the small learning rate used in our experiment, Assumption 1 is in fact true simply because small updates are not big enough to cancel the initial quantities even if aligned. In Figure 4, we verify that Assumption 1 also holds when $\Delta \boldsymbol{W}_\ell$ and $\Delta \boldsymbol{h}_\ell(\boldsymbol{x})$ are respectively replaced by $\boldsymbol{W}_\ell - \boldsymbol{W}_\ell^0$ and $\boldsymbol{h}_\ell(\boldsymbol{x}) - \boldsymbol{h}_\ell^0(\boldsymbol{x})$, the total updates across all of training.

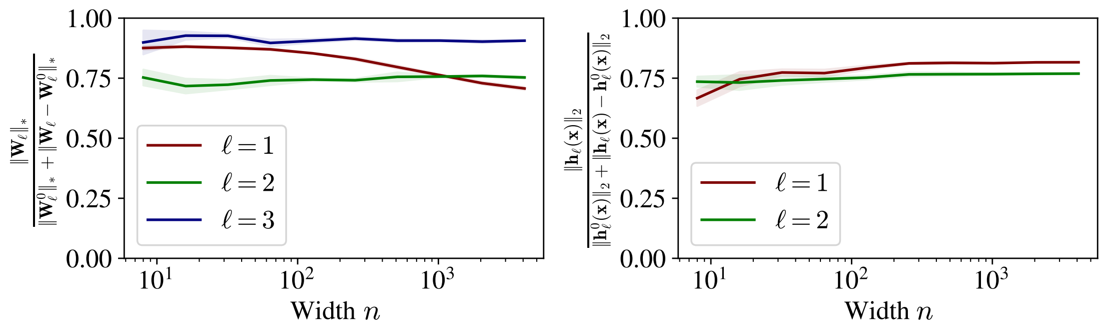


Figure 4: **Verification of Assumption 1.** Subplots show the ratios between left- and right-hand sides of conditions of Assumption 1 at various layers. Envelopes show variation over trials. The ratios are $\Theta(1)$ (showing no obvious decay with network width), verifying Assumption 1.

Assumption 2. $\|\boldsymbol{h}'_\ell(\boldsymbol{x})\|_2 = \Theta\left(\|\boldsymbol{h}_\ell(\boldsymbol{x})\|_2\right)$.

Recall that $\boldsymbol{h}'_\ell(\boldsymbol{x}) = \phi(\boldsymbol{h}_\ell(\boldsymbol{x}))$, and that $\boldsymbol{h}_\ell(\boldsymbol{x})$ contains elements of size $\Theta(1)$. Satisfaction of Assumption 2 merely requires that ϕ maps a nonvanishing fraction of preactivations to nonzero quantities. Its violation would require preactivations to concentrate in regions of $\mathbb{R}$ which ϕ maps to zero. This might occur, for example, in the unlikely scenario in which $\phi = \text{ReLU}$ and almost all preactivations are negative.[6] For activations that do not map nonzero inputs to zero (for example, $\phi = \tanh$), Assumption 2 may be dropped altogether. Figure 5 verifies Assumption 2 in a deep ReLU MLP. See Poole et al. (2016) for a theoretical framework sufficient to check Assumption 2 at initialization.

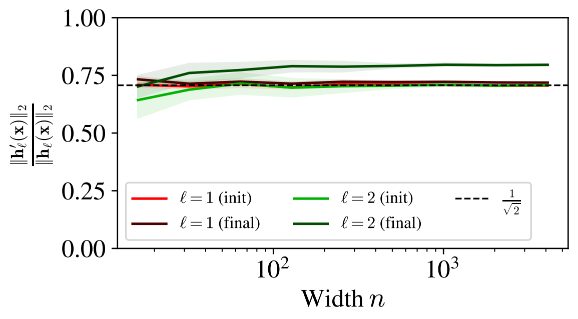


Figure 5: **Verification of Assumption 2.** At all layers, the norm of the activation vector scales as the norm of the corresponding preactivation vector, both before and after training. (Note that the ratio in this case is close to $1/\sqrt{2}$, which is what one expects from an approximately-mean-zero random variable passed through a ReLU nonlinearity.)

Assumption 3. $\|\Delta \boldsymbol{W}_\ell \boldsymbol{h}_\ell(\boldsymbol{x}_i)\|_2 = \Theta(\|\frac{1}{B}\Delta \boldsymbol{W}_\ell^{(i)} \boldsymbol{h}_\ell(\boldsymbol{x}_i)\|_2)$.

Like Assumption 1, a violation of Assumption 3 would require a perfect cancellation of high-dimensional matrices, which is unlikely. We verify Assumption 3 in Figure 6. Assumption 3 is also verified implicitly by the right-hand subplot of Figure 1.

[6]It will also occur if the activation function is $\phi(z) = 0$, but we can neglect this edge case.

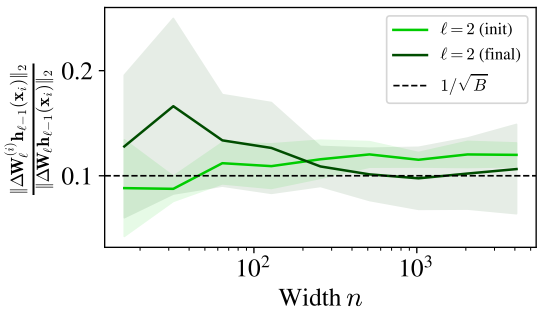


Figure 6: **Verification of Assumption 3.** We compute $\|\Delta \boldsymbol{W}_\ell^{(i)} \boldsymbol{h}_\ell(\boldsymbol{x}_i)|_2 / \|\Delta \boldsymbol{W}_\ell \boldsymbol{h}_\ell(\boldsymbol{x}_i)\|_2$ at layer $\ell = 2$ for a single step both at initialization and after training. As expected, this ratio remains $\Theta(1)$ as width grows — note the lack of decay with width. We note as a curiosity that this quantity hovers around $1/\sqrt{B}$ (dashed line), which one expects from the central limit theorem if, for example, $\Delta \boldsymbol{W}_\ell$ resembles a sum of B terms similar to $\Delta \boldsymbol{W}_\ell^{(i)}$ but with random signs.

D Scalings for biases

Here we extend our spectral analysis to biases. Let $\boldsymbol{b}_\ell \in \mathbb{R}^{n_\ell}$ be a bias vector which enters during forward propagation as $\boldsymbol{h}_\ell(\boldsymbol{x}) = \boldsymbol{W}_\ell \boldsymbol{h}'_{\ell-1}(\boldsymbol{x}) + \boldsymbol{b}_\ell$. We may choose to view the bias vector as a weight matrix $\boldsymbol{b}_\ell \in \mathbb{R}^{n_\ell \times 1}$ connecting an auxiliary layer with width 1 and output 1 to the ℓth hidden layer, after which we may simply apply our scaling analysis for weight matrices. The spectral scaling condition (Condition 1) prescribes that $\|\boldsymbol{b}_\ell\|_2 = \Theta(\sqrt{n_\ell})$ and $\|\Delta \boldsymbol{b}_\ell\|_2 = \Theta(\sqrt{n_\ell})$, and Parametrization 1 prescribes that the initialization scale and learning rate should be $\sigma_\ell^b = \Theta(1)$ and $\eta_\ell^b = \Theta(n_\ell)$. In practice, one may usually just take $\sigma_\ell^b = 0$.

E Nondimensionalization and natural norms

While there are conventional notions of vector norms and matrix spectral norms, one can equip any norm on the input and output spaces of a matrix, thereby inducing a corresponding spectral norm on it. For example, the conventional spectral norm of a matrix is defined assuming the typical ℓ^2 norm on the input and output spaces; if the input space is instead given the ℓ^1 norm, then the spectral norm is something very different. As we explain below, the vectors encountered in deep neural networks naturally come with a specific notion of norm, which we refer to as the "natural norm." When we equip vectors at various layers of the neural network with their natural norms, we induce what we call a "natural spectral norm" on the weight matrices. Our primary criterion Condition 1 is then simplified to stipulating that these *natural spectral norms* should be $\Theta(1)$ in the limit of large network width (Condition 2).

E.1 Dense and sparse vectors

Vectors in the context of deep learning typically fall into one of two categories: dense and sparse. Dense vectors are characterized by having every entry contribute a constant amount to the squared Euclidean norm of the vector, while sparse vectors have only a constant number of entries that do so. Here, the term "constant" means $\Theta(1)$ with respect to the size of the vector.[7]

For example: in the input layer, image data typically takes the form of dense vectors, while the one-hot encoding in language models takes the form of sparse vectors. The output vector of a network is typically a dense vector. All hidden vectors (pre- or post-activation) are dense.

[7]Actually, a vector of d iid samples from a unbounded subgaussian or subexponential distribution will have its extreme elements scale like polylog(d), so the really correct thing to say here is $\tilde{\Theta}(1)$, but in the main text we convey the key intuition in the main text without being too pedantic. This fine detail does not affect the conclusion of this section.

E.2 Defining natural norms

Definition 2 (Natural ℓ^2-norm)**.** The *natural ℓ^2-norm* for a *dense* vector $\boldsymbol{v} \in \mathbb{R}^m$ is the RMS norm

$$\|\boldsymbol{v}\|_{\tilde{2}} := \frac{1}{\sqrt{m}} \|\boldsymbol{v}\|_2 . \tag{28}$$

The *natural ℓ^2-norm* for a *sparse* vector $\boldsymbol{v}$ is simply the usual ℓ^2-norm:

$$\|\boldsymbol{v}\|_{\tilde{2}} := \|\boldsymbol{v}\|_2 . \tag{29}$$

While we say "dense" or "sparse" vector, these adjectives really apply to the space that contains such a vector. For example, the set of one-hot encodings form a "sparse" input space, while the pre-activations at layer ℓ forms a "dense" hidden space.

Definition 3 (Natural spectral norm)**.** Given a parameter matrix $\boldsymbol{A}$, equip its input and output spaces with their natural norms. Then the *natural spectral norm* $\|\boldsymbol{A}\|_{\tilde{*}}$ of $\boldsymbol{A}$ is defined as the induced spectral norm.

For example, if both the input and output spaces are dense, then the natural spectral norm of $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is

$$\|\boldsymbol{A}\|_{\tilde{*}} := \frac{\sqrt{m}}{\sqrt{n}} \|\boldsymbol{A}\|_* . \tag{30}$$

E.3 Spectral scaling in natural norms

By equipping vectors and matrices with their natural norms, we simplify the conditions Desideratum 1 and Condition 1 from the main text:

Desideratum 2 (Feature learning, natural norms)**.** Let $\boldsymbol{h}_\ell(\boldsymbol{x}) \in \mathbb{R}^{n_\ell}$ denote the features of input $\boldsymbol{x}$ at layer ℓ of a deep neural network, and let $\Delta \boldsymbol{h}_\ell(\boldsymbol{x}) \in \mathbb{R}^{n_\ell}$ denote their change after a gradient update. We desire that:

$$\|\boldsymbol{h}_\ell\|_{\tilde{2}} = \Theta(1) \quad \text{and} \quad \|\Delta \boldsymbol{h}_\ell\|_{\tilde{2}} = \Theta(1), \qquad \text{at layers } \ell = 1, ..., L-1.$$

Note the norm here is just RMS norm because we only talk about hidden vectors (which are always dense).

Condition 2 (Spectral scaling, natural norms)**.** Consider applying a gradient update $\Delta \boldsymbol{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ to the ℓth weight matrix $\boldsymbol{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$. The spectral norms of these matrices should satisfy:

$$\|\boldsymbol{W}_\ell\|_{\tilde{*}} = \Theta(1) \quad \text{and} \quad \|\Delta \boldsymbol{W}_\ell\|_{\tilde{*}} = \Theta(1), \qquad \text{at layers } \ell = 1, ..., L.$$

In summary, using these rescaled norms (that we call "natural norms"), our problem is nondimensionalized: feature vectors, feature vector updates, weight matrices, and weight matrix updates are $\Theta(1)$ in norm. These natural norms provide a universal framework that covers specific cases, such as one-hot embeddings in language models, that are not handled directly by Condition 1.