A. Related Works

Comparison with Mean Field Limits For 1-hidden-layer MLP, the mean field limit (Chizat & Bach, 2018; Mei et al., 2018; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2018) is equivalent to the μP limit modulo the symmetry of Eq. (13) (see Appendix B). Several works also proposed different versions of mean field frameworks for deeper MLPs (Araújo et al., 2019; Fang et al., 2020; Nguyen, 2019; Nguyen & Pham, 2020; Sirignano & Spiliopoulos, 2020). However, they did not consider the typical Gaussian $\mathcal{N}(0, 1/n)$ random initialization (or the appropriately rescaled version in their respective parametrizations)[17], which has a Central-Limit effect as opposed to a Law-of-Large-Numbers effect. For example, (Araújo et al., 2019; Nguyen & Pham, 2020) can cover the case of $\mathcal{N}(0, 1/n^2)$, instead of $\mathcal{N}(0, 1/n)$, initialization, which in fact causes the function to be stuck at initialization. Of these works, the mean field limit of (Fang et al., 2020) has the form most similar to what we derive here. There, as we do here, the coordinate distribution of each (pre)activation vector is tracked recursively. The main difference is, while (Fang et al., 2020) has an atypical initialization involving ℓ_2 regression, we consider the usual Gaussian $\mathcal{N}(0, 1/n)$ scheme. Such a (size $n \times n$) Gaussian matrix in the middle of the network has a distinctly different effect, more similar to that of a Gaussian matrix in the usual NNGP/NTK calculation,[18] than the "mean field" matrices considered in (Fang et al., 2020) and previous works (Araújo et al., 2019; Nguyen, 2019; Nguyen & Pham, 2020; Sirignano & Spiliopoulos, 2020), which has an "integral kernel" effect that is the straightforward generalization of matrices to function spaces. Nevertheless, discrete time versions of the 1-hidden-layer mean field limit and of many of the multi-layer limits (such as (Fang et al., 2020; Nguyen & Pham, 2020)) can be derived directly by writing the corresponding initialization and training inside a Tensor Program and applying the Master Theorem (Theorem G.4).

Discrete- vs Continuous-Time Gradient Descent At a high level, there are two natural limits of neural networks training dynamics: large-width and continuous-time. Most prior works on infinite-width limits of neural networks also took the continuous-time limit simultaneously, e.g. (Chizat & Bach, 2018; Jacot et al., 2018; Mei et al., 2018; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2018). In contrast, here we only take the large width limit, so that gradient descent stays discrete-time. Then the results of these prior works can be recovered by taking another continuous-time limit. From a practical perspective, the continuous-time limit is often unnatural, e.g. 1) because the step size is usually as large as possible to speed up training, 2) because of the task (such as reinforcement learning), or 3) because of the importance of hyperparameters like batch size that are hidden away in such limits. On the theory side, taking the continuous-time limit can create issues with 1) well-posedness and 2) existence and uniqueness of the resulting ODE/PDE. While they can sometimes be proved to hold, they are artifacts of the continuous-time limit, as the corresponding questions for the discrete time evolution are trivial, and thus not relevant to the behavior of real networks.

Technical Assumptions Earlier works on neural tangent or mean field limits (e.g. (Chizat & Bach, 2018; Fang et al., 2020; Jacot et al., 2018; Mei et al., 2018; Nguyen & Pham, 2020; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2018)) assume various forms of regularity conditions, such as 1) 0th, 1st, and/or 2nd order smoothness on the nonlinearity or other related functions, and 2) the support boundedness, subgaussianity, and/or PDF smoothness of initialization distributions. These are often either unnatural or difficult to check. In our work, the only assumption needed to rigorously obtain the infinite-width limit is that the nonlinearity ϕ has a polynomially bounded weak 2nd derivative and that the loss function has a continuous derivative w.r.t. the prediction (Assumption N.21). In particular, when we specialize to the 1-hidden-layer case and derive the discrete time version of the mean field limit, we cover the standard Gaussian initialization; in fact, we can allow any heavy-tailed initialization that can be written as the image of a Gaussian under a pseudo-Lipschitz function, which include nonsmooth PDFs and singular distributions.[19] This generosity of technical assumptions is due to that of the Tensor Programs Master Theorems proven in (Yang, 2019a; 2020a;b).

Training Time Many prior works (e.g. (Allen-Zhu et al., 2018; Huang & Yau, 2019; Mei et al., 2018)) derived explicit time dependence of the convergence to infinite-width limit, so that a larger width can allow the network to stay close to the limit for longer. In this paper, our results only concern training time independent of width, since our primary objective is to investigate the limit itself and its feature learning capabilities. Moreover, recent evidence suggests that, given a fixed computational budget, it's always better to train a larger model for a shorter amount of time (Li et al., 2020b), which validates the practical relevance of our limit mode. Nevertheless, it is possible to prove a quantitative version of the Tensor Programs Master Theorem, by which

[17] In fact, empirically we observe such Gaussian random initialization to be crucial to performance compared to the mean-field-style initialization in this literature.

[18] Actually, it is more similar to the Gaussian matrix in asymmetric message passing (Bayati & Montanari, 2011) in that care must be taken to keep track of correlation between W and $W^\top$.

[19] We won't expand further here, but it can be derived straightforwardly from the Master Theorem (Theorem G.4).

one can straightforwardly allow training time to increase with width.

Classification of Parametrizations (Chizat & Bach) pointed out that the weights move very little in the NTK limit, so that linearization approximately holds around the initial parameters, in contrast to the mean field limit (for 1-hidden-layer networks) where the weights move substantially. For this reason, they called the former "lazy training" and the latter "active training," which are classified nonrigorously by a multiplicative scaling factor of the logit (similar to $n^{-a_{L+1}}$ in this paper). While these terms are not formally defined, they intuitively correspond to the kernel and feature learning regimes in our paper. From a different perspective, (Mei et al., 2019) observed that the NTK and mean field limit can be thought of as short and long time-scale regimes of the mean field evolution equations. Neither of the above works attempted to formally classify natural parametrizations of neural networks. In contrast, (Woodworth et al., 2020) studied a toy class of neural networks in the context of implicit regularization due to the scale α of initialization (which is closely related to logit multiplier of (Chizat & Bach) noted above). They identified the $\alpha \to \infty$ limit (of the scale α, not of width) with the "kernel regime" and the $\alpha \to 0$ limit with what they call the "rich regime". They showed that the former is implicitly minimizing an ℓ_2 risk while the latter, an ℓ_1 risk. They claim width allows the toy model to enter the kernel regime more naturally, but as we see in this work, both kernel and feature learning regimes are admissible in the large width limit of a standard MLP. Closer to our approach, (Golikov, 2020) studied what amounts to a 2-dimensional subspace of the space of stable abc-parametrizations for $L = 1$. They proposed a notion of stability which is similar to the combination of stability and nontriviality in this paper. They characterized when the Neural Tangent Kernel, suitably generalized to any parametrization and playing a role similar to the feature kernel in this paper, evolves over time. However, to simplify the proofs, they assumed that the gradients for the different weight matrices are estimated using different inputs, a very unnatural condition. In contrast, here our results are for the usual SGD algorithm applied to MLPs of arbitrary depth. In all of the above works and most of existing literature, not much attention is paid to the feature learning capabilities of neural networks in the right parametrization, as opposed to our focus here. A notable exception is (Chizat & Bach, 2020), which showed that the mean field limit, but not the NTK limit, can learn low dimension linear structure of the input distribution resulting in ambient-dimension-independent generalization bounds.

Other Related Works (Lewkowycz et al., 2020) proposed a toy model to study how large learning rate can induce a neural network to move out of the kernel regime in $\Omega(\log(width))$ time. Since our dichotomy result only concerns training for $O(1)$ time (which, as we argue above, is more practically relevant), there is no contradiction. (Sohl-Dickstein et al., 2020) also noted that standard parametrization leads to unstable training dynamics. They then injected constants in the NTK parametrization, such as $\alpha/\sqrt{n}$ instead of $1/\sqrt{n}$ and tuned α in the resulting kernel. (Aitchison, 2020; Aitchison et al., 2020) also observed the lack of feature learning in NNGP and NTK limits but, in contrast to taking the exact limit of SGD training as we do here, they proposed a deep kernel process as a way of loosely mimicking feature learning in finite-width networks. (Gilboa & Gur-Ari, 2019) empirically observed that wider networks achieve better downstream performance with linear transfer learning, even though on the original pretraining task there can be little difference. (Li et al., 2020a) proved a complexity separation between NTK and finite-width networks by showing the latter approximates a sort of infinite-width feature learning network. In the literature surrounding NTK, often there are subtle differences in parametrization leading to subtle differences in conclusion (e.g. (Allen-Zhu et al., 2018; Du et al., 2018; Zou et al., 2018)). Our abc framework encapsulates all such parametrizations, and can easily tell when two ostensibly different parametrizations (e.g. (Du et al., 2018; Zou et al., 2018)) are actually equivalent or when they are really different (e.g. (Allen-Zhu et al., 2018; Du et al., 2018)) via Eq. (13).

B. Motivating Examples: Neural Tangent Kernel and Mean Field Limits

In this section, we motivate the discussion of feature learning vs kernel regime by reviewing the well-known tangent kernel and mean field limits of a shallow neural network.

For simplicity, define a shallow network $f(\xi)$ with input/output dimension 1 by

$$f(\xi) = Vx(\xi) \in \mathbb{R},\ x(\xi) = \phi(h(\xi)) \in \mathbb{R}^n,\ h(\xi) = U\xi \in \mathbb{R}^n. \tag{10}$$

As a specialization of Eq. (1), we parametrize weights $V = n^{-a_v}v \in \mathbb{R}^{1\times n}$ and $U = n^{-a_u}u \in \mathbb{R}^{n\times 1}$, where the width n should be thought of as tending to ∞, and v, u should be thought of as the actual trainable parameters. We will sample $v_\alpha \sim \mathcal{N}(0, n^{-2b_v}), u_\alpha \sim \mathcal{N}(0, n^{-2b_u})$ for $\alpha \in [n]$. The learning rate is ηn^{-c} for some η independent of n.

For example, in the *Neural Tangent Parametrization* (abbreviated *NTP*) (Jacot et al., 2018), $a_u = b_v = b_u = 0, a_v = 1/2, c = 0$. The *Mean Field Parametrization* (abbreviated *MFP*) corresponds to $a_v = 1, a_u = b_u = b_v = 0, c = -1$; however, as will be explained shortly, we will use the equivalent formulation $a_u = -1/2, a_v = b_u = b_v = 1/2, c = 0$ in this section so $c = 0$ for both NTP and MFP. We remark that the GP limit, i.e. training only the last layer of a infinitewide, randomly initialized network, is a special case of the NTK limit where the first layer is not trained. Everything we discuss below about the NTK limit specializes to the GP limit appropriately.

Given an input ξ, the gradient of f can be calculated as

$$dx(\xi) = V, \quad dh(\xi) = dx(\xi) \odot \phi'(h(\xi)),$$
$$dv(\xi) = n^{-a_v} x(\xi), \quad du(\xi) = n^{-a_u} dh(\xi)\xi$$

where $d\bullet(\xi)$ is shorthand for $\nabla_\bullet f(\xi)$ (however, note that later in Section 5, $d\bullet(\xi)$ will stand for $n\nabla_\bullet f(\xi)$). For loss function $\mathcal{L} : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$, the loss gradient on a pair (ξ, y) is then given by $\mathcal{L}'(f(\xi), y)[dv(\xi), du(\xi)]$ (where $\mathcal{L}'$ denotes derivative in first argument).

Note that one can keep the function f invariant while changing the magnitude of the gradient dv by changing a_v, b_v, holding $a_v + b_v$ constant; likewise for du. Thus, the trajectory of f stays fixed if, for any $\theta \in \mathbb{R}$, we set $a_u \leftarrow a_u + \theta, a_v \leftarrow a_v + \theta, b_u \leftarrow b_u - \theta, b_v \leftarrow b_v - \theta, c \leftarrow c - 2\theta$ (also see Eq. (13)). With $\theta = -1/2$, this explains why the two formulations of MFP above are equivalent. Then, for both NTP and MFP, we will consider the dynamics of f trained under stochastic gradient descent with learning rate $\eta = 1$ and batch size 1, where the network is fed the pair (ξ_t, y_t) at time t, starting with $t = 0$.

Notation and Setup Below, when we say a (random) vector $v \in \mathbb{R}^n$ has *coordinate size* $O(n^a)$ (written $v = O(n^a)$),[20] we mean $\sqrt{\|v\|^2/n} = O(n^a)$ with high probability for large n. Intuitively, this means that each coordinate has a typical fluctuation of $O(n^a)$. Likewise if $O(n^a)$ is replaced with $\Theta(n^a)$ or $\Omega(n^a)$. See Definition N.2 for a formal definition.

Let $f_t, h_t, x_t, U_t, V_t, dx_t, dh_t, dv_t, du_t$ denote the corresponding objects at time t, with $t = 0$ corresponding to random initialization. We also abuse notation and write $x_t = x_t(\xi_t)$, i.e. applying the function x_t specifically to tth input ξ_t; similarly for $f_t, h_t, dx_t, dh_t, dv_t, du_t$. These symbols will never appear by themselves to denote the corresponding function, so this should cause no confusion. Then SGD effectively updates U and V by

$$U_{t+1} = U_t - \chi_t n^{-a_u} du_t, \quad V_{t+1} = V_t - \chi_t n^{-a_v} dv_t.$$

where $\chi_t \stackrel{\text{def}}{=} \mathcal{L}'(f_t, y_t)$. Finally, let $\Delta\bullet_t \stackrel{\text{def}}{=} \bullet_t - \bullet_0$, for all $\bullet \in \{f, h, x, U, V, dx, dh, dv, du\}$. For example, after 1 SGD update, we have, for any $\xi \in \mathbb{R}$,

$$\begin{aligned}
\Delta h_1(\xi) &= h_1(\xi) - h_0(\xi) = -n^{-a_u}\chi_0\xi du_0 \\
&= -n^{-2a_u}\chi_0\xi_0\xi dh_0 \\
&= -n^{-2a_u}\chi_0\xi_0\xi dx_0 \odot \phi'(h_0) \qquad (11)\\
\Delta f_1(\xi) &= V_0\Delta x_1(\xi) + \Delta V_1 x_1(\xi) \\
&= V_0\Delta x_1(\xi) - n^{-a_v} dv_0^\top x_1(\xi) \\
&= V_0\Delta x_1(\xi) - n^{-2a_v} x_0^\top x_1(\xi) \qquad (12)
\end{aligned}$$

B.0.1. Key Observations

Let's list a few characteristics of the NTK and MF limits in the context of the shallow network in Eq. (10), and then discuss them in the general setting of deep MLP. We will keep our discussion intuitive to carry across the key ideas.

Feature Evolution For a generic $\xi \in \mathbb{R}$, its embedding vector $x_0(\xi)$ has coordinates of $\Theta(1)$ size in both NTP and MFP. However, for any $t \geq 1$ independent of n, $\Delta x_t(\xi)$ generically has coordinate size $\Theta(1/\sqrt{n})$ in NTP but $\Theta(1)$ in MFP.

Example for $t = 1$: By Eq. (11), we have

$$\Delta h_1(\xi) = n^{-2a_u}\chi_0\xi_0\xi dx_0 \odot \phi'(h_0).$$

Plug in $a_u = 0$ for NTP. Observe that $\xi_0, \xi, \chi_0 = \Theta(1)$,[21] so

$$\Delta h_1(\xi) = \Theta(1) \cdot dx_0 \odot \phi'(h_0). \qquad \text{(in NTP)}$$

In addition, $\phi'(h_0) = \Theta(1)$ because $h_0 = \Theta(1)$, so

$$\Delta h_1(\xi) = \Theta(1) \cdot dx_0 \odot \Theta(1). \qquad \text{(in NTP)}$$

Finally, $dx_0 = V_0 = \Theta(1/\sqrt{n})$ in NTP. Altogether, this implies

$$\begin{aligned}
&\Delta h_1(\xi) = \Theta(1/\sqrt{n}) \\
\implies &\Delta x_1(\xi) \approx \phi'(h_0(\xi)) \odot \Delta h_1(\xi) = \Theta(1/\sqrt{n}) \to 0. \qquad \text{(in NTP)}
\end{aligned}$$

On the other hand, in MFP, the only thing different is $a_u = -1/2$ and $dx_0 = \Theta(1/n)$, which implies

$$\begin{aligned}
\Delta h_1(\xi) &= \Theta(n) \cdot \Theta(1/n) \odot \Theta(1) = \Theta(1) \\
&\implies \Delta x_1(\xi) = \Theta(1). \qquad \text{(in MFP)}
\end{aligned}$$

Feature Kernel Evolution Therefore the *feature kernel* $F_t(\xi, \zeta) \stackrel{\text{def}}{=} x_t(\xi)^\top x_t(\zeta)/n$ does not change in the NTK limit but it does in the MF limit, i.e. for any fixed $t \geq 1$,[22]

$$\begin{aligned}
\lim_{n\to\infty} F_t(\xi, \zeta) &= \lim_{n\to\infty} F_0(\xi, \zeta), \quad \text{in NTP, but} \\
\lim_{n\to\infty} F_t(\xi, \zeta) &\neq \lim_{n\to\infty} F_0(\xi, \zeta), \quad \text{in MFP, in general.}
\end{aligned}$$

[20] Contrast this with a common semantics of $v = O(n^a)$ as $\|v\| = O(n^a)$.

[21] $\chi_0 = \mathcal{L}'(f_0, y_0) = \Theta(1)$ because f_0 has variance $\Theta(1)$.

[22] here the limit should be construed as almost sure limits; see Theorem G.4.

Indeed, regardless of parametrization, we have

$$F_t(\xi,\zeta) = \frac{1}{n}\Big[x_0(\xi)^\top x_0(\zeta) + \Delta x_t(\xi)^\top x_0(\zeta) + x_0(\xi)^\top \Delta x_t(\zeta) + \Delta x_t(\xi)^\top \Delta x_t(\zeta)\Big].$$

In NTP, because $\Delta x_t(\xi) = \Theta(1/\sqrt{n})$ as noted above,

$$\frac{1}{n}\Delta x_t(\xi)^\top x_0(\zeta) = \frac{1}{n}\sum_{\alpha=1}^{n} \Delta x_t(\xi)_\alpha x_0(\zeta)_\alpha = \frac{1}{n}\sum_{\alpha=1}^{n} O(n^{-1/2}) = O(n^{-1/2}),$$

and likewise the other terms involving Δx_t will vanish as $n \to \infty$. But in MFP, $\Delta x_t(\xi) = \Theta(1)$ will in general be correlated with $x_0(\zeta)$ such that $\frac{1}{n}\Delta x_t(\xi)^\top x_0(\zeta) = \frac{1}{n}\sum_{\alpha=1}^{n}\Theta(1) = \Theta(1)$.

It may seem somewhat puzzling how the NTK limit induces change in f without feature or feature kernel evolution. We give some intuition next.

How does the Function Change? If the NTK limit does not allow features to evolve, then how does learning occur? To answer this question, note

$$\Delta f_t(\xi) = V_0 \Delta x_t(\xi) + \Delta V_t x_0(\xi) + \Delta V_t \Delta x_t(\xi).$$

In short, then, the evolution of $f_t(\xi)$ in the NTK limit is predominantly due to $V_0\Delta x_t(\xi)$ and $\Delta V_t x_0(\xi)$ only, while in the MF limit, $\Delta V_t \Delta x_t(\xi)$ also contributes nontrivially.

Example: For $t = 1$, $\Delta f_1(\xi) = V_0\Delta x_1(\xi) + n^{-2a_v} x_0^\top x_0(\xi) + n^{-2a_v} x_0^\top \Delta x_1(\xi)$. In NTP, $a_v = 1/2$, so the term $n^{-2a_v} x_0^\top x_0(\xi) = \Theta(1)$ for generic ξ, ξ_0. On the other hand, $n^{-2a_v} x_0^\top \Delta x_1(\xi) = O(1/\sqrt{n})$ because $\Delta x_1(\xi) = O(1/\sqrt{n})$ as noted above. Likewise,

$$\begin{aligned} V_0\Delta x_1(\xi) &\approx V_0[\phi'(h_0(\xi)) \odot \Delta h_1(\xi)] \\ &= V_0[\phi'(h_0(\xi)) \odot \Delta h_1(\xi)] \\ &= C\sum_{\alpha=1}^{n} V_{0\alpha}\phi'(h_0(\xi)_\alpha) V_{0\alpha}\phi'(h_{0\alpha}) \\ &= C\sum_{\alpha=1}^{n} (V_{0\alpha})^2 \phi'(h_0(\xi)_\alpha)\phi'(h_{0\alpha}), \end{aligned}$$

where $C = \chi_0 \xi_0 \xi = \Theta(1)$. Now $(V_{0\alpha})^2 = \Theta(1/n)$ and is almost surely positive. On the other hand, $\phi'(h_0(\xi)_\alpha)\phi'(h_{0\alpha}) = \Theta(1)$ and should have a nonzero expectation over random initialization (for example, if ϕ is relu then this is obvious). Therefore, the sum above should amount to $V_0\Delta x_1(\xi) \approx \Theta(1)$. In summary, in the NTK limit, $\Delta f_1(\xi) = \Theta(1)$ due to the interactions between V_0 and $\Delta x_1(\xi)$ and between ΔV_1 and $x_0(\xi)$, but there is only vanishing interaction between ΔV_1 and $\Delta x_1(\xi)$.

The case for general t, again, can be derived easily using Tensor Programs.

Pretraining and Transfer Learning The simple fact above about the feature kernel K implies that the NTK limit is unable to perform linear transfer learning. By *linear transfer learning,* we mean the popular style of transfer learning where one discards the pretrained linear classifier layer and train a new one on top of the features (e.g. x in our example), which are fixed. Indeed, this is a linear problem and thus only depends on the kernel of the features. If this kernel is the same as the kernel at initialization, then the pretraining phase has had no effect on the outcome of this "transfer" learning.

In fact, a more sophisticated reasoning shows pretraining in the NTK limit is no better than random initialization for transfer learning even if finetuning is performed to the whole network, not just the classifier layer. This remains true if we replace the linear classifier layer by a new deep neural network. See Remark N.15 and Theorem N.16. The Word2Vec experiment we do in this paper is a linear transfer task.

In some other settings, such as some settings of metalearning, like the few-shot learning task in this paper, the last layer of the pretrained network is not discarded. This is called *adaptation*. Then the NTK limit does not automatically trivialize transfer learning. However, as will be seen in our experiments, the NTK limit still vastly underperforms the feature learning limit, which is exemplified by the MF limit here.

Kernel Gradient Descent in Function Space In NTP, as $n \to \infty$, $\langle \nabla_{U,V} f_0(\xi), \nabla_{U,V} f_0(\zeta)\rangle$ converges to some deterministic value $K(\xi,\zeta)$ such that K forms a kernel (the NTK). Then, in this limit, if the learning rate is η, the function f evolves according to kernel gradient descent $f_{t+1}(\xi) = f_t(\xi) - \eta K(\xi, \xi_t)\chi_t$. However, this shouldn't be the case for the MF limit. For example, if ϕ is identity, then intuitively $f_{t+1}(\xi) - f_t(\xi)$ should be quadratic in η, not linear, because two layers are updated at the same time.

C. abc-Parametrization

C.1. 1-Dimensional Redundancy in abc

We can scale the parameter gradients $\nabla_{w^l} f$ arbitrarily while keeping f fixed, if we vary a_l, b_l while fixing $a_l + b_l$: $\nabla_{w^l} f$ is scaled by $n^{-\theta}$ if $a_l \leftarrow a_l + \theta, b_l \leftarrow b_l - \theta$. In other words, changing a_l, b_l this way effectively gives w^l a per-layer learning rate. If we apply this gradient with learning rate ηn^{-c}, then the change in W^l is scaled by $\eta n^{-c-2\theta}$.

Consequently, if $c \leftarrow c - 2\theta$, then W^l is not affected by the change in a_l, b_l. In summary,

$$\text{For all } \theta \in \mathbb{R}, f_t(\xi) \text{ stays fixed for all } t \text{ and } \xi \text{ if we set}$$
$$a_l \leftarrow a_l + \theta,\ b_l \leftarrow b_l - \theta,\ c \leftarrow c - 2\theta. \tag{13}$$

This insight in particular implies MFP is a special case of MUP in the case of 1-hidden-layer MLPs.

D. Standard Parametrization: Pedagogical Examples

In this section, we give intuition for why gradient descent of neural network in standard parametrization (SP) will lead to logits blowup after 1 step, if the learning rate is $\omega(1/n)$, where n is the width. In addition, we will see why, with learning rate $O(1/n)$, SP is in kernel regime. We first consider the simplest example and then state the general result at the end of the section.

To demonstrate the general principle in deep networks, it is necessary to consider the behavior of an $n \times n$ matrix in the middle of the network. Thus, the simplest case is a 2-hidden-layer linear MLP, i.e. Eq. (1) with $L = 2$ and $\phi = id$. The standard parametrization is given by

$$a_l = 0\ \forall l, \quad b_1 = 0, \quad b_l = 1/2\ \forall l \geq 2. \tag{SP}$$

We consider 1 step of SGD with learning rate n^{-c} on a single data pair (ξ, y). Then we can without ambiguity suppress explicit dependence on ξ and write

$$f = V\bar{h}, \quad \bar{h} = Wh, \quad h = U\xi, \tag{14}$$

where $U_{\alpha\beta} \sim \mathcal{N}(0, 1)$ and $W_{\alpha\beta}, V_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$ are the trainable parameters. We use $\bullet_t$ to denote the quantity $\bullet$ after t step of SGD. Because we only focus on the 1st step of SGD, we lighten notation and write $\bullet = \bullet_0$.

Initialization Since U, W, V are independently sampled, a standard Central Limit argument would show that $h, \bar{h}, f$ all have roughly iid Gaussian coordinates of variance $\Theta(1)$.

First Gradient Now let's consider the gradients of f on the data pair (ξ, y), which are given by

$$\begin{aligned} d\bar{h} &= V^\top, \quad & dh &= W^\top d\bar{h}, \\ dV &= \bar{h}, \quad & dW &= d\bar{h}\, h^\top = V^\top h^\top, \quad dU = dh\, \xi^\top. \end{aligned} \tag{15}$$

For simplicity, suppose we only update W by learning rate n^{-c} (and leave U, V unchanged); our conclusion will not change in the general case where we train all layers. Then with χ denoting the loss derivative $\mathcal{L}'(f, y)$, we can write

$$W_1 = W - n^{-c}\chi\, dW.$$

We shall show now that $c \geq 1$ or else f_1 blows up with the width n after this SGD step.

After First SGD Step At $t = 1$, $h_1 = h$ since we did not update U, but

$$\bar{h}_1 = W_1 h = \bar{h} - n^{-c}\chi\, dWh = \bar{h} - n^{-c}\chi \cdot V^\top h^\top h \tag{16}$$

$$f_1 = V\bar{h}_1 = f - n^{-c}\chi\, VV^\top h^\top h. \tag{17}$$

Now, as noted above, h has iid $\Theta(1)$ coordinates, so $h^\top h = \Theta(n) \in \mathbb{R}$. Similarly, $V \in \mathbb{R}^{1\times n}$ has Gaussian coordinates of variance $\Theta(1/n)$, so $VV^\top = \Theta(1) \in \mathbb{R}$. Finally, for typical loss function $\mathcal{L}$ like MSE or cross entropy, $\chi = \mathcal{L}'(f, y)$ is of order $\Theta(1)$ because f fluctuates on the order $\Theta(1)$. Altogether,

$$f_1 = f - \Theta(n^{1-c}).$$

Therefore, for f_1 to remain $O(1)$, we must have $c \geq 1$, i.e. the learning rate is $O(1/n)$.

Kernel Regime and Lack of Feature Learning Consequently, the network cannot learn features in the large width limit if we would like the logits to not blow up. Indeed, this version of SGD where only W is updated can be seen to correspond to the limit where

$$a_1 = \theta, \quad b_1 = -\theta, \quad a_2 = 0, \quad b_2 = 1/2,$$
$$a_3 = \theta, \quad b_3 = -\theta + 1/2, \quad \theta \to \infty.$$

With $c = 1$ as derived above, the parametrization is stable and nontrivial, as can be checked from Theorems 3.2 and 3.3. Then we get $r = 1/2 > 0$, so by Corollary 3.8, this parametrization is in kernel regime and does not admit feature learning. We can also see this directly from Eq. (16): from our calculations above,

$$\bar{h}_1 - \bar{h} = O(n^{1-c})\, V^\top = O(1)\, V^\top$$

whose coordinates have size $O(n^{-1/2})$ since V's coordinates do, so there's no feature learning (at least in the first step). Finally, from Eq. (17), because $VV^\top \to 1$ and $n^{-c}h^\top h = n^{-1}h^\top h \to \|\xi\|^2$, we get[23]

$$f_1 - f \to -\chi K(\xi, \xi) \stackrel{\text{def}}{=} -\chi\|\xi\|^2,$$

i.e. f evolves by kernel gradient descent with the linear kernel. Our derivations here only illustrate the first SGD step, but we can get the same conclusion from all steps of SGD similarly.

We summarize the general case below, which follows trivially from Theorem 3.2 and Corollary 3.8.

[23]Formally, these are almost sure convergences, but we suppress these details to emphasize on intuition.

Theorem D.1. *An L-hidden-layer MLP in standard parametrization (see Eq. (SP) and Table 1) can only allow SGD learning rate of order $O(1/n)$ if we require $\lim_{n\to\infty} \mathbb{E} f_t(\xi)^2 < \infty$ for all training routine, time t, and input ξ. In this case, it is in kernel regime and does not admit feature learning.*

E. Maximal Update Parametrization: Pedagogical Examples

As shown in the last section, the standard parametrization does not admit a feature learning infinite-width limit without blowing up logits. Here we propose simple modifications of the standard parametrization to make this possible while maintaining stability: 1) To enable feature learning, it suffices to divide the logits by $\sqrt{n}$ and use $\Theta(1)$ learning rate, i.e. set $a_{L+1} = 1/2, c = 0$ on top of Eq. (SP); 2) to allow *every layer* to perform feature learning, we should furthermore set $a_1 = -1/2, b_1 = 1/2$. We will see that this essentially means we update each weight matrix as much as possible without blowing up the logits or activations, so we call this the Maximal Update Parametrization (abbreviated MUP or μP).

E.1. Dividing Logits by $\sqrt{n}$

For example, in the 2-hidden-layer linear MLP example above, the network would compute

$$f(\xi) = \frac{1}{\sqrt{n}} v\bar{h}(\xi), \quad \bar{h}(\xi) = Wh(\xi), \quad h(\xi) = U\xi, \tag{18}$$

where $U_{\alpha\beta} \sim \mathcal{N}(0, 1)$ and $W_{\alpha\beta}, v_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$ are the trainable parameters. Compared to SP (Eq. (14)), $h(\xi), \bar{h}(\xi)$ stays the same; only the logit $f(\xi)$ is scaled down. Again, to simplify notation, we abbreviate $\bullet = \bullet_0$ and suppress explicit dependence on ξ. This has two consequences

Logits at Initialization Converge to 0 since f has variance $\Theta(1/n)$ (compare to the GP limit of MLP in SP at initialization).

$\Theta(1)$ **Learning Rate and Feature Learning** Even though $f \to 0$, the loss derivative $\chi = \mathcal{L}'(f, y)$ stays $\Theta(1)$ if $y \neq 0$. When we redo the calculation in Eq. (16), we see

$$\bar{h}_1 = \bar{h} - n^{-c-1/2}\chi\, v^\top h^\top h = \bar{h} - \Theta(n^{-c+1/2})v^\top \tag{19}$$

$$f_1 = f - n^{-c-1}\chi\, vv^\top h^\top h = f - \Theta(n^{-c}).$$

Because v has coordinates of size $\Theta(n^{-1/2})$, we see that $\bar{h}$ and f both change by $\Theta(1)$ coordinatewise if $c = 0$ (i.e. learning rate is $\Theta(1)$). This directly illustrates feature learning after just 1 step of SGD. For general MLPs, we can also check $a_{L+1} = 1/2, c = 0$ on top of Eq. (SP) implies $r = 0$ and thus admits feature learning by Theorem 3.5.

Kernel Behavior or Lack Thereof The example we have here, where we only train the middle layer in a linear MLP, actually *is* in kernel regime. This does not violate Corollary 3.8, however, which assumes Assumption I.1. If, for example, we have tanh nonlinearity, then it is easy to see the μP SGD dynamics does not have a kernel limit: If so, then $f_1 - f$ is linear in the learning rate η. But note $\bar{h}_1 - \bar{h}$ is $\Theta(1)$ as $n \to \infty$ and linear in η, as can be derived similarly to Eq. (19). Because tanh is bounded, this cannot happen. Contrast this with SP or NTP, where $\bar{h}_1 - \bar{h}$ is $\Theta(1/\sqrt{n})$ and thus "resides in the linear regime of tanh", allowing perfect scaling with η.

In addition, even in an linear MLP, if we train the middle layer *and* the last layer, then the dynamics intuitively will become quadratic in the weights, so will not have a kernel limit. Contrast this with SP or NTP, which suppress these higher order interactions because the learning rate is small, and a first order Taylor expansion heuristic holds.

How is this different from standard parametrization with learning rate $1/\sqrt{n}$? As shown above, the logit f blows up like $\Theta(\sqrt{n})$ after 1 step of SGD with learning rate $\Theta(1/\sqrt{n})$ in the standard parametrization, but remains $\Theta(1)$ in our parametrization here. The reason these two parametrizations seem similar is because in the 1st step, the weights receive the same updates modulo the loss derivative $\chi = \mathcal{L}'(f, y)$. Consequently, $x_1^L - x^L$ and $h_1^L - h^L$ are $\Theta(1)$ coordinatewise in both cases. However, this update makes x_1^L correlated with W_1^{L+1}, so that $W_1^{L+1}x_1^L$ (and f_1) scales like $\Theta(n^{1-a_{L+1}-b_{L+1}})$ due to Law of Large Numbers. Thus only in our parametrization here ($a_{L+1} = b_{L+1} = 1/2$) is it $\Theta(1)$, while in standard parametrization ($a_{L+1} = 0, b_{L+1} = 1/2$) it blows up like $\Theta(\sqrt{n})$. Contrast this with the behavior at initialization, where W^{L+1} and x^L are independent and zero-mean, so $W^{L+1}x^L$ scales like $\Theta(n^{1/2-a_{L+1}-b_{L+1}})$ by Central Limit Theorem.

E.2. First Layer Parametrization

While this now enables feature learning, the first layer preactivation h effectively stays fixed throughout training even if we were to train U. For example, if we update U in the linear MLP example Eq. (18), then by Eq. (15),

$$U_1 = U - n^{-c}\chi\, dU = U - n^{-c}\chi\, dh\xi^\top$$
$$h_1 = U_1\xi = h - n^{-c}\chi\, dh\xi^\top\xi = h - \Theta(n^{-c})dh$$

since $\xi^\top\xi, \chi = \Theta(1)$. Now $dh = W^\top d\bar{h} = W^\top \frac{1}{\sqrt{n}} v^\top$ has roughly iid Gaussian coordinates, each of size $\Theta(1/n)$, since $\frac{1}{\sqrt{n}} v^\top$ has coordinates of the same size. Therefore, even with $c = 0$, h changes by at most $O(1/n)$ coordinatewise, which is dominated by its value at initialization. This $O(1/n)$ change also induces a $O(1/n)$ change in f, which would be dominated by the $\Theta(1)$ change due to W's evolution, as seen in Eq. (19).

We therefore propose to set $a_1 = -1/2, b_1 = 1/2$ on top of Appendix E.1's parametrization. This implies the forward pass of f remains the same but U's gradient is scaled up by n, so that h now changes by $\Theta(1)$ coordinatewise. In summary, this yields Definition 4.1.

Notice that μP for a 1-hidden-layer perceptron is equivalent to the mean field parametrization by Eq. (13). We also describe μP for any architecture in Appendix K.1.

E.3. What is μP Maximal In?

For technical reasons, we adopt Assumption I.1 again for the formal results of this section.

In an abc-parametrization, the change in weight $W = W_t^l$ for any $l \geq 2$ due to learning rate n^{-c} is $\delta W \stackrel{\text{def}}{=} -n^{-c} \cdot n^{-2a} dh\, x^\top$ where we abbreviated $x = x_t^{l-1}, h = h_t^l, a = a_l$. (We will use δ to denote 1-step change, but Δ to denote lifetime change). In the next forward pass, δW contributes $\delta W \bar{x} = -n^{1-c-2a}(x^\top \bar{x}/n)dh$, where $\bar{x}$ is the new activation due to change in previous layers' weights. In general, x and $\bar{x}$ are strongly correlated. Then $x^\top \bar{x}/n \to R$ for some $R \neq 0$ by Law of Large Numbers (as they both have $\Theta(1)$ coordinates in a stable parametrization). One can heuristically see that dh has the same size as the last layer weights, which is $\Theta(n^{-(a_{L+1}+b_{L+1})} + n^{-(2a_{L+1}+c)})$ (where the first summand is from W_0^{L+1} and the other from ΔW_t^{L+1}). Thus, $\delta W \bar{x}$ is a vector with $\Theta(n^{-r_l}) \stackrel{\text{def}}{=} \Theta((n^{-(a_{L+1}+b_{L+1})} + n^{-(2a_{L+1}+c)})n^{1-c-2a})$ coordinates. If $r_l > 0$, then $\delta W \bar{x}$ contributes vanishingly; if $r_l < 0$, then $\delta W \bar{x}$ blows up. For $l = 1$, we get similar insights after accounting for the finite dimensionality of ξ.

Definition E.1. For $l \in [L]$, we say W^l *is updated maximally* if $\Delta W_t^l x_t^{l-1}(\xi)$ has $\Theta(1)$ coordinates for some training routine[24], time $t \geq 1$, and input ξ.

Proposition E.2. *In a stable abc-parametrization, for any $l \in [L]$, W^l is updated maximally iff*

$$r_l \stackrel{\text{def}}{=} \min(a_{L+1}+b_{L+1}, 2a_{L+1}+c)+c-1+2a_l+\mathbb{I}(l=1) = 0.$$

Note that r (Definition 3.1) is the minimum of r_l over all l. In μP, we can calculate that $r_l = 0$ for all $l \in [L]$, so *all $W^l, l \in [L]$, are updated maximally*. Put another way, the final embedding $x^L(\xi)$ will have nonvanishing (nonlinear) contributions from ΔW^l of all l. These contributions cause the logit $f(\xi)$ to change via interactions with W_0^{L+1} and ΔW_t^{L+1}. If both W_0^{L+1} and ΔW_t^{L+1} are too small, then the logit is fixed to its initial value, so all of the feature learning would have been useless.[25] It's also possible for one to contribute vanishingly but not the other.[26] But both contribute in μP.

Definition E.3. We say W^{L+1} *is updated maximally* (resp. *initialized maximally*) if $\Delta W_t^{L+1} x_t^L(\xi) = \Theta(1)$ (resp. $W_0^{L+1} \Delta x_t^L(\xi) = \Theta(1)$) for some training routine, time $t \geq 1$, and input ξ.

Note Definition E.3 is similar to Definition E.1 except $\Delta W_t^{L+1} x_t^L(\xi) \in \mathbb{R}$ but $\Delta W_t^l x_t^{l-1}(\xi) \in \mathbb{R}^n$.

Proposition E.4. *In a stable abc-parametrization, W^{L+1} is 1) updated maximally iff $2a_{L+1}+c = 1$, and 2) initialized maximally iff $a_{L+1} + b_{L+1} + r = 1$.*

We remark that, by Theorem 3.3, a parametrization is nontrivial iff W^{L+1} is maximally updated or initialized. Using Propositions E.2 and E.4 and Theorem 3.2, we can now easily conclude

Theorem E.5. *In μP, W^l is updated maximally for every $l \in [L+1]$, and W^{L+1} is also initialized maximally. μP is the unique stable abc-parametrization with this property.*

F. Deriving Feature Learning Infinite-Width Limit: Deep MLP Examples

F.1. 2-Hidden-Layer MLP: SGD with Partially Decoupled Backpropagation

A 2-hidden-layer MLP is given by

$$f(\xi) = V\bar{x}(\xi), \quad \bar{x}(\xi) = \phi(\bar{h}(\xi)), \quad \bar{h}(\xi) = Wx(\xi),$$
$$x(\xi) = \phi(h(\xi)), \quad h(\xi) = U\xi,$$

for $U \in \mathbb{R}^{n\times 1}, W \in \mathbb{R}^{n\times n}, V \in \mathbb{R}^{1\times n}$ parametrized like $U = \sqrt{n}u, V = \frac{1}{\sqrt{n}}v$ and with initialization $u_{\alpha\beta}, W_{\alpha\beta}, v_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$. The presence of the $n \times n$ Gaussian matrix W ("$\infty \times \infty$" as opposed to "$\infty\times$ finite" like U or "finite $\times\infty$" like V) is new and has two major effects on the infinite-width training dynamics: 1) A Central Limit effect from the random Gaussian nature of W and 2) a correlation effect between W and its transpose $W^\top$. We isolate the first effect here by analyzing a slightly different version of backpropagation (which has a different limit than normal backpropagation), and then discuss the second effect in the next section. We abuse notation and abbreviate $W = W_0$.

Partially Decoupled Backpropagation In this section, we analyze a version of SGD where the backpropagation

[24] Recall that *training routine* means a package of learning rate ηn^{-c}, training sequence $\{(\xi_t, y_t)\}_{t\geq 0}$, and a loss function $\mathcal{L}(f(\xi), y)$ that is continuously differentiable in the prediction of the model $f(\xi)$.

[25] It is indeed possible to perform feature learning in a trivial parametrization, e.g. $b_l = 1/2\ \forall l, a_1 = -1/2, a_2 = 100 + 1/2, c = -100$ in a 2-hidden-layer MLP.

[26] e.g. take $a_{L+1} = 100 + 1/2, b_{L+1} = -100 + 1/2$, then ΔW^{L+1} is negligible.

weights are partially decoupled from the forward propagation weights. Here, we think of ΔW_t as the trainable weights, initialized at 0, and think of the Gaussian W as untrainable "constants". The forward pass proceeds normally[27] with $W_t = W + \Delta W_t$. But we sample and fix an iid copy $\widetilde{W}$ of $W^\top$ before training, and in the backward pass compute

$$dx_t = (\widetilde{W} + \Delta W_t^\top) d\bar{h}_t \quad \text{instead of} \tag{20}$$

$$dx_t = (W^\top + \Delta W_t^\top) d\bar{h}_t = W_t^\top d\bar{h}_t. \tag{21}$$

In particular, at initialization, we would have $dx_0 = \widetilde{W} d\bar{h}_0$ instead of $dx_0 = W^\top d\bar{h}_0$. Everything else stays the same in the backward pass[28]. Finally, each weight is still updated by SGD via the usual outer products: with $\chi_t \stackrel{\text{def}}{=} \mathcal{L}'(f_t, y_t)$,

$$v_{t+1} = v_t - \chi_t \bar{x}_t^\top / \sqrt{n}, \quad \Delta w_{t+1} = \Delta w_t - \chi_t d\bar{h}_t x_t^\top / n, \\ u_{t+1} = u_t - \chi_t \xi_t dh_t^\top / \sqrt{n}. \tag{22}$$

Since $V = v/\sqrt{n}, W = w, U = \sqrt{n} u$ per μP, this causes the following changes in Ws:

$$V_{t+1} = V_t - \chi_t \bar{x}_t^\top / n, \quad \Delta W_{t+1} = \Delta W_t - \chi_t d\bar{h}_t x_t^\top / n, \\ U_{t+1} = U_t - \chi_t \xi_t dh_t^\top \tag{23}$$

Note here we update Δw and ΔW instead of w and W.

Why This Decoupled SGD? The reasons we talk about this version of SGD is that it isolates the effect of having a Gaussian $n \times n$ matrix $\widetilde{W}$ in the backward pass, and we can derive its infinite-width limit relatively easily using Central Limit heuristics. In the normal version of SGD, $\widetilde{W}$ would equal $W^\top$, and its correlation with W creates additional terms in the infinite-width dynamics, that are better explained on their own.

Again, we walk through the first few forward and backward passes to gain some intuition for the infinite-width limit, before stating the general case.

First Forward Pass is similar to that in Section 5.1 and follows the usual calculations involved in deriving the NNGP[29].

First Backward Pass is similar to that in Section 5.1 and to calculations involved in deriving Neural Tangent Kernel, except swapping $W^\top$ with $\widetilde{W}$ (which at this point has no visible effect, because of the Gradient Independence Phenomenon (Yang, 2020a); but the effect will become clear in the second forward pass)[30]. We end up with $\Delta W_1 = -\chi_0 d\bar{h}_0 x_0^\top$, as usual.

Second Forward Pass As usual, we have $Z^{h_1} = \xi_1 Z^{U_1} = \xi_1 Z^{U_0} - \mathring{\chi}_0 \xi_1 \xi_0 Z^{dh_0}$ and $Z^{x_1} = \phi(Z^{h_1})$, reflecting the coordinate distributions of h_1 and x_1[31]. Next,

$$\bar{h}_1 = W x_1 + \Delta W_1 x_1 = W x_1 - \chi_0 d\bar{h}_0 \frac{x_0^\top x_1}{n}. \tag{24}$$

On one hand, 1) $\frac{x_0^\top x_1}{n} \to \mathbb{E}\, Z^{x_1} Z^{x_0}$ by a Law of Large Numbers heuristic. On the other hand, 2) by a Central Limit heuristic, $W x_1$ should roughly have Gaussian coordinates $Z^{W x_1}$ correlated with $Z^{\bar{h}_0} = Z^{W x_0}$ with $\mathrm{Cov}(Z^{W x_1}, Z^{W x_0}) = \lim \frac{x_0^\top x_1}{n} = \mathbb{E}\, Z^{x_1} Z^{x_0}$. However, *very importantly*, this Central Limit heuristic is correct only because we used $\widetilde{W}$ in backprop instead of $W^\top$; otherwise, h_1 has a strong correlation with W through $dh_0 = \phi'(h_0) \odot (W^\top d\bar{h}_0)$, and thus so does x_1, so that $W x_1$ no longer has Gaussian coordinates. This is the "second major effect" referred to in the beginning of this section. See Appendix F.2 for how to handle this correlation.

In any case, in our scenario here,

$$Z^{\bar{h}_1} \stackrel{\text{def}}{=} Z^{W x_1} - c Z^{d\bar{h}_0}, \quad \text{where} \quad c = \mathring{\chi}_0 \, \mathbb{E}\, Z^{x_1} Z^{x_0},$$

is a linear combination of a Gaussian variable and the gradient $d\bar{h}_0$'s coordinate random variable. Finally, $Z^{\bar{x}_1} = \phi(Z^{\bar{h}_1})$ and the logit is $f_1 = \frac{1}{n} \sum_{\alpha=1}^n (nV_1)_\alpha \bar{x}_{1\alpha} \to \mathring{f}_1 \stackrel{\text{def}}{=} \mathbb{E}\, Z^{nV_1} Z^{\bar{x}_1} = \mathbb{E}\, Z^{nV_0} Z^{\bar{x}_1} - \mathring{\chi}_0 \, \mathbb{E}\, Z^{\bar{x}_0} Z^{\bar{x}_1}$.

Second Backward Pass Everything proceeds just like in the 1-hidden-layer case[32] except for the computation of

$$dx_1 = \widetilde{W} d\bar{h}_1 - \Delta W_1^\top d\bar{h}_1 = \widetilde{W} d\bar{h}_1 - \chi_0 x_0 \frac{d\bar{h}_0^\top d\bar{h}_1}{n}.$$

Like in the computation of $\bar{h}_1$ in Eq. (24), $\frac{d\bar{h}_0^\top d\bar{h}_1}{n} \to \mathbb{E}\, Z^{d\bar{h}_0} Z^{d\bar{h}_1}$ and $\widetilde{W} d\bar{h}_1$ is roughly Gaussian (and correlated with $\widetilde{W} d\bar{h}_0$ in the natural way). But again, for this

[27] i.e. $f_t = V_t \bar{x}_t, \bar{x}_t = \phi(\bar{h}_t), \bar{h}_t = (W + \Delta W_t) x_t, x_t = \phi(h_t), h_t = U \xi_t$.

[28] i.e. $d\bar{x}_t = nV_t^\top, d\bar{h}_t = \phi'(\bar{h}_t) \odot d\bar{x}_t, dh_t = \phi'(h_t) \odot dx_t$

[29] 1) h_0 is iid Gaussian with coordinates drawn from $Z^{h_0} = \xi_0 Z^{U_0}$; 2) x_0 has coordinates $Z^{x_0} = \phi(Z^{h_0})$; 3) $\bar{h}_0 = W x_0$ has roughly iid coordinates drawn from a zero-mean Gaussian $Z^{\bar{h}_0}$ by a Central Limit heuristic, where $Z^{\bar{h}_0}$ is correlated with $Z^{\bar{h}_0(\xi)}$ for any ξ (including $\xi = \xi_0$) with covariance $\mathrm{Cov}(Z^{\bar{h}_0}, Z^{\bar{h}_0(\xi)}) = \lim_{n\to\infty} \frac{1}{n} x_0^\top x_0(\xi) = \mathbb{E}\, Z^{x_0} Z^{x_0(\xi)}$; 4) $\bar{x}_0$ has coordinates $Z^{\bar{x}_0} = \phi(Z^{\bar{h}_0})$; 5) $f_0 = \frac{1}{n} \sum_{\alpha=1}^n (nV_0)_\alpha \bar{x}_{0\alpha} \to \mathring{f}_0 \stackrel{\text{def}}{=} \mathbb{E}\, Z^{nV_0} Z^{\bar{x}_0}$ by a Law of Large Number heuristic.

[30] 1) $d\bar{x}_0 = nV_0^\top$ so $Z^{d\bar{x}_0} = Z^{nV_0}$; 2) $Z^{d\bar{h}_0} = \phi'(Z^{\bar{h}_0}) \odot Z^{d\bar{x}_0}$; 3) $Z^{dx_0} = Z^{\widetilde{W} d\bar{h}_0}$ is Gaussian with covariance $\mathrm{Cov}(Z^{dx_0}, Z^{dx_0(\xi)}) = \lim_{n\to\infty} \frac{1}{n} dh_0^\top dh_0(\xi) = \mathbb{E}\, Z^{dh_0} Z^{dh_0(\xi)}$ for any input ξ; 4) $Z^{dh_0} = \phi'(Z^{h_0}) \odot Z^{dx_0}$. Since f converges to a deterministic number $\mathring{f}_0$, we also generically have $\mathcal{L}'(f, y_0) \to \mathring{\chi}_0 \stackrel{\text{def}}{=} \mathcal{L}'(\mathring{f}_0, y_0)$. Finally, the weights are updated like Eq. (23).

[31] Recall they abbreviate $h_1(\xi_1)$ and $x_1(\xi_1)$

[32] $d\bar{x}_1 = nV_1^\top, d\bar{h}_1 = d\bar{x}_1 \odot \phi'(\bar{h}_1), dh_1 = dx_1 \odot \phi'(h_1)$

Gaussian intuition to be correct, it is crucial that we use $\widetilde{W}$ here instead of $W^\top$, or else $d\bar{x}_1$ (and thus $d\bar{h}_1$) is strongly correlated with $W^\top$ (through $\bar{x}_0 = \phi(Wx_0)$ inside $n\Delta V_1 = -\chi_0 \bar{x}_0^\top$).

In any case, we have

$$Z^{dx_1} = Z^{\widetilde{W} d\bar{h}_1} - cZ^{x_0}, \quad \text{where} \quad c = \mathring{\chi}_0 \mathbb{E}\, Z^{d\bar{h}_0} Z^{d\bar{h}_1},$$

is a sum of Gaussian $Z^{\widetilde{W} d\bar{h}_1}$ and a multiple of Z^{x_0}. Then weights are updated according to Eq. (23).

tth Iteration For general t, we always have (true in normal SGD as well)

$$\Delta W_t = -\frac{1}{n} \sum_{s=0}^{t-1} \chi_s d\bar{h}_s x_s^\top$$

so that in the forward pass

$$\bar{h}_t = Wx_t + \Delta W_t x_t = Wx_t - \sum_{s=0}^{t-1} \chi_s d\bar{h}_s \frac{x_s^\top x_t}{n} \quad (25)$$

$$Z^{\bar{h}_t} \stackrel{\text{def}}{=} Z^{Wx_t} - \sum_{s=0}^{t-1} \mathring{\chi}_s Z^{d\bar{h}_s} \mathbb{E}\, Z^{x_s} Z^{x_t}.$$

Here Z^{Wx_t} is Gaussian with covariance $\mathrm{Cov}(Z^{Wx_t}, Z^{Wx_s}) = \mathbb{E}\, Z^{x_t} Z^{x_s}$ for any s. This means that $Z^{\bar{h}_t}$ and $Z^{\bar{h}_s}$ are correlated through Z^{Wx_t}, Z^{Wx_s} (but also through $Z^{d\bar{h}_r}, r \leq \min(t, s)$). Likewise, in the backward pass,

$$dx_t = \widetilde{W} d\bar{h}_t - \Delta W^\top d\bar{h}_t = \widetilde{W} d\bar{h}_t - \sum_{s=0}^{t-1} \chi_s x_s \frac{d\bar{h}_s^\top d\bar{h}_t}{n}$$

$$Z^{dx_t} \stackrel{\text{def}}{=} Z^{\widetilde{W} d\bar{h}_t} - \sum_{s=0}^{t-1} \mathring{\chi}_s Z^{x_s} \mathbb{E}\, Z^{d\bar{h}_s} Z^{d\bar{h}_t}$$

Here, $Z^{\widetilde{W} d\bar{h}_t}$ is Gaussian with covariance $\mathrm{Cov}(Z^{\widetilde{W} d\bar{h}_t}, Z^{\widetilde{W} d\bar{h}_s}) = \mathbb{E}\, Z^{d\bar{h}_t} Z^{d\bar{h}_s}$ for any s. Thus, Z^{dx_t} and Z^{dx_s} are correlated through $Z^{\widetilde{W} d\bar{h}_t}, Z^{\widetilde{W} d\bar{h}_s}$ (but also through $Z^{x_r}, r \leq \min(t, s)$). Again, the Gaussianity of Z^{Wx_t} and $Z^{\widetilde{W} d\bar{h}_t}$ depend crucially on the fact that we use $\widetilde{W}$ instead of $W^\top$ in backpropagation.

Other parts of the forward and backward propagations are similar to before. Our reasoning can be formalized via Tensor Programs to prove the following

Theorem F.1. *Consider a 2-hidden-layer MLP in μP with partially decoupled backpropagation as in Eq. (21) and any training routine with learning rate 1. Suppose ϕ' is pseudo-Lipschitz.*[33] *As $n \to \infty$, for every input ξ,*

$$f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi), \quad \textit{where } \mathring{f}_t(\xi) \textit{ is defined as follows:}$$

(forward pass)

$$\mathring{f}_t(\xi) \stackrel{\text{def}}{=} \mathbb{E}\, Z^{nV_t} Z^{\bar{x}_t(\xi)}, \quad Z^{\bar{x}_t(\xi)} \stackrel{\text{def}}{=} \phi(Z^{\bar{h}_t(\xi)}),$$
$$Z^{x_t(\xi)} \stackrel{\text{def}}{=} \phi(Z^{h_t(\xi)}), \quad Z^{h_t(\xi)} \stackrel{\text{def}}{=} \xi Z^{U_t}$$
$$Z^{\bar{h}_t(\xi)} \stackrel{\text{def}}{=} Z^{Wx_t(\xi)} - \sum_{s=0}^{t-1} \mathring{\chi}_s Z^{d\bar{h}_s} \mathbb{E}\, Z^{x_s} Z^{x_t(\xi)} \quad (26)$$
$$\{Z^{Wx_t(\xi)}\}_{\xi,t} \textit{ centered, jointly Gaussian with}$$
$$\mathrm{Cov}(Z^{Wx_t(\xi)}, Z^{Wx_s(\zeta)}) = \mathbb{E}\, Z^{x_t(\xi)} Z^{x_s(\zeta)}$$

(backward pass)

$$\chi_t \stackrel{\text{def}}{=} \mathcal{L}'(\mathring{f}_t, y_t), \quad Z^{d\bar{x}_t} \stackrel{\text{def}}{=} Z^{nV_t}, \quad Z^{d\bar{h}_t} \stackrel{\text{def}}{=} \phi'(Z^{\bar{h}_t}) Z^{d\bar{x}_t}$$
$$Z^{dh_t} \stackrel{\text{def}}{=} \phi'(Z^{h_t}) Z^{dx_t}$$
$$Z^{dx_t} \stackrel{\text{def}}{=} Z^{\widetilde{W} d\bar{h}_t} - \sum_{s=0}^{t-1} \mathring{\chi}_s Z^{x_s} \mathbb{E}\, Z^{d\bar{h}_s} Z^{d\bar{h}_t} \quad (27)$$
$$\{Z^{\widetilde{W} d\bar{h}_t}\}_t \textit{ centered, jointly Gaussian with}$$
$$\mathrm{Cov}(Z^{\widetilde{W} d\bar{h}_t}, Z^{\widetilde{W} d\bar{h}_s}) = \mathbb{E}\, Z^{d\bar{h}_t} Z^{d\bar{h}_s}$$

(U, V updates)

$$Z^{nV_{t+1}} \stackrel{\text{def}}{=} Z^{nV_t} - \mathring{\chi}_t Z^{\bar{x}_t} \quad Z^{U_{t+1}} \stackrel{\text{def}}{=} Z^{U_t} - \mathring{\chi}_t \xi_t Z^{dh_t}$$

with Z^{U_0} and Z^{nV_0} being independent standard Gaussians as initial conditions, and by definition, $\{Z^{Wx_t(\xi)}\}_{\xi,t}$, $\{Z^{\widetilde{W} d\bar{h}_t}\}_t$, Z^{U_0}, and Z^{nV_0} are mutually independent sets of random variables. Here, if h_t appears without argument, it means $h_t(\xi_t)$; likewise for $\bar{h}_t, x_t, \bar{x}_t, dh_t, d\bar{h}_t, dx_t, d\bar{x}_t, \mathring{f}_t$.

F.2. 2-Hidden-Layer MLP: Normal SGD

Finally, we dicuss normal SGD for 2-hidden-layer MLP, i.e. in backprop we compute

$$dx_t = W_t^\top d\bar{h}_t = (W^\top + \Delta W^\top) d\bar{h}_t.$$

The first forward and backward passes are essentially the same as in the last section. However, as mentioned there, in the second forward pass, Wx_1 (a part of $\bar{h}_1 = Wx_1 + \Delta W_1 x_1$) will no longer be approximately Gaussian because of the correlation between x_1 and W. Let's first get some intuition for why this is before stating the infinite-width limit formally.

Warmup: $\phi = \mathrm{id}$ First, as warmup, suppose $\phi = \mathrm{id}$. In this case, Wx_1 will actually still be Gaussian, but its variance will be different than what's predicted in the previous section. To lighten notation, we write $x = x_1$ in this section. Then unwinding the definition of x, we have

$$x = h + aW^\top z$$

[33] This roughly means that ϕ' has a polynomially bounded weak derivative; see Definition L.3.

where we abbreviated $h = \xi_1 U_0, z = d\bar{h}_0, a = -\chi_0\xi_0\xi_1$. Then Wx has coordinates

$$(Wx)_\alpha = (Wh)_\alpha + a(WW^\top z)_\alpha.$$

As derived in the first forward pass in Appendix F.1, $(Wh)_\alpha$ is approximately Gaussian (particularly because W, U_0 are independent). This is true for $(WW^\top z)_\alpha$ as well here because we assumed $\phi = \mathrm{id}$, but not true generally. Indeed,

$$\begin{aligned}(WW^\top z)_\alpha &= \sum_{\beta,\gamma} W_{\alpha\beta} W_{\gamma\beta} z_\gamma \\ &= z_\alpha \sum_\beta (W_{\alpha\beta})^2 + \sum_\beta \sum_{\gamma\neq\alpha} W_{\alpha\beta} W_{\gamma\beta} z_\gamma.\end{aligned}$$

We will soon see the derivations of Appendix F.1 correspond to ignoring the first term: In the second term, there are n summands of the form $\sum_{\gamma\neq\alpha} W_{\alpha\beta} W_{\gamma\beta} z_\gamma$ that are approximately iid with variance $\approx \|z\|^2/n^2$. Thus, the second term itself, by a Central Limit heuristic, should converge to $\mathcal{N}(0, \lim_{n\to\infty} \|z\|^2/n)$. On the other hand, the first term $z_\alpha \sum_\beta (W_{\alpha\beta})^2 \to z_\alpha$ by Law of Large Numbers. Tying it all together, $(Wx)_\alpha$ is a linear combination of two Gaussian terms $(Wh)_\alpha$ and $\sum_\beta \sum_{\gamma\neq\alpha} W_{\alpha\beta} W_{\gamma\beta} z_\gamma$, as well as as z_α (which is Gaussian in the case of $\phi = \mathrm{id}$, but not generally).

Note that, if we did $(W\widetilde{W} z)_\alpha$ instead of $(WW^\top z)_\alpha$, as in the last section, then the same analysis would show the first term is $z_\alpha \sum_\beta W_{\alpha\beta} \widetilde{W}_{\beta\alpha} \to 0$, while the second term converge in distribution to the same Gaussian. Thus, the effect of decoupling in Appendix F.1 is killing the copy of z in $(Wx)_\alpha$.

We can summarize our derivation here in terms of Z:

$$\begin{aligned}\text{For } \phi = \mathrm{id}: \quad Z^{Wx} &\stackrel{\text{def}}{=} Z^{Wh} + aZ^{WW^\top z} &(28)\\ &= Z^{Wh} + a(\hat{Z}^{WW^\top z} + Z^z), &(29)\\ \text{where} \quad \hat{Z}^{WW^\top z} &\stackrel{\text{def}}{=} \mathcal{N}(0, \mathbb{E}(Z^z)^2).\end{aligned}$$

Note the Central Limit heuristic in the derivation of $\hat{Z}^{WW^\top z}$ also shows $\hat{Z}^{WW^\top z}$ is jointly Gaussian with Z^{Wh} with $\mathrm{Cov}(\hat{Z}^{WW^\top z}, Z^{Wh}) = \mathbb{E}\, Z^{W^\top z} Z^h$. So, to put Eq. (29) in a form more suggestive of the general case, we will write

$$\begin{aligned} Z^{Wx} &= \hat{Z}^{Wx} + aZ^z, \quad \text{where} \\ \hat{Z}^{Wx} &= Z^{Wh} + a\hat{Z}^{WW^\top z} \stackrel{\text{d}}{=} \mathcal{N}(0, \mathbb{E}(Z^x)^2). \end{aligned} \tag{30}$$

General ϕ Unwinding the definition of x, we have

$$x = \phi(h + aW^\top z \odot \phi'(h_0)). \tag{31}$$

By Taylor-expanding ϕ, we can apply a similar (though more tedious) argument as above to derive

$$Z^{Wx} = \hat{Z}^{Wx} + cZ^z \tag{32}$$

where $c = a\,\mathbb{E}\,\phi'(Z^{h_1})\phi'(Z^{h_0})$ and $\hat{Z}^{Wx} \stackrel{\text{d}}{=} \mathcal{N}(0, \mathbb{E}(Z^x)^2)$. In the case of $\phi = \mathrm{id}$, c reduces to a as above, recovering Eq. (30). For general ϕ, we can immediately see that Z^{Wx} is not Gaussian because $Z^z = Z^{d\bar{x}_0}\phi'(Z^{\bar{h}_0})$ is not. In the Tensor Programs framework formalized in Appendix G, cZ^z is denoted $\dot{Z}^{Wx}$.

Similarly, coordinates distribution of $dx_1 = W_1^\top d\bar{h}_1$ will also change in the backward pass.

General t For general t, we obtain dynamical equations in Z identical to those in Theorem F.1 except that Eq. (26) and Eq. (27) need to be modified. We state the general result below.

Theorem F.2. *Consider a 2-hidden-layer MLP in μP and any training routine with learning rate 1. Suppose ϕ' is pseudo-Lipschitz.*[34] *As $n \to \infty$, for every input ξ, $f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi)$ where $\mathring{f}_t(\xi)$ is defined the same way as in Theorem F.1 except that Eq. (26) should be replaced with*

$$Z^{\bar{h}_t(\xi)} \stackrel{\text{def}}{=} \hat{Z}^{Wx_t(\xi)} + \dot{Z}^{Wx_t(\xi)} - \sum_{s=0}^{t-1} \mathring{\chi}_s Z^{d\bar{h}_s}\, \mathbb{E}\, Z^{x_s} Z^{x_t(\xi)}$$

$$\{\hat{Z}^{Wx_t(\xi)}\}_{\xi,t} \text{ centered, jointly Gaussian with}$$

$$\mathrm{Cov}(\hat{Z}^{Wx_t(\xi)}, \hat{Z}^{Wx_s(\zeta)}) = \mathbb{E}\, Z^{x_t(\xi)} Z^{x_s(\zeta)}$$

and Eq. (27) should be replaced with

$$Z^{dx_t} \stackrel{\text{def}}{=} \hat{Z}^{W^\top d\bar{h}_t} + \dot{Z}^{W^\top d\bar{h}_t} - \sum_{s=0}^{t-1} \mathring{\chi}_s Z^{x_s}\, \mathbb{E}\, Z^{d\bar{h}_s} Z^{d\bar{h}_t}$$

$$\{\hat{Z}^{W^\top d\bar{h}_t}\}_t \text{ centered, jointly Gaussian with}$$

$$\mathrm{Cov}(\hat{Z}^{W^\top d\bar{h}_t}, \hat{Z}^{W^\top d\bar{h}_s}) = \mathbb{E}\, Z^{d\bar{h}_t} Z^{d\bar{h}_s}.$$

Like in Theorem F.1, by definition, $\{\hat{Z}^{Wx_t(\xi)}\}_{\xi,t}$, $\{\hat{Z}^{W^\top d\bar{h}_t}\}_t$, Z^{U_0}, and Z^{nV_0} are mutually independent sets of random variables.

Here, $\dot{Z}^{Wx_t(\xi)} \stackrel{\text{def}}{=} \sum_{r=0}^{t-1} \theta_r Z^{d\bar{h}_r}$ where θ_r is calculated like so: $Z^{x_t(\xi)}$ by definition is constructed as

$$Z^{x_t(\xi)} = \Phi(\hat{Z}^{W^\top d\bar{h}_0}, \ldots, \hat{Z}^{W^\top d\bar{h}_{t-1}}, Z^{U_0})$$

for some function[35] *$\Phi : \mathbb{R}^{t+1} \to \mathbb{R}$. Then*

$$\theta_r \stackrel{\text{def}}{=} \mathbb{E}\, \partial\Phi(\hat{Z}^{W^\top d\bar{h}_0}, \ldots, \hat{Z}^{W^\top d\bar{h}_{t-1}}, Z^{U_0})/\partial\hat{Z}^{W^\top d\bar{h}_r}.$$

Likewise, $\dot{Z}^{W^\top d\bar{h}_t} \stackrel{\text{def}}{=} \sum_{r=0}^{t-1} \theta_r Z^{x_r}$ where θ_r is calculated as follows: $Z^{d\bar{h}_t}$ by definition is constructed as

$$Z^{d\bar{h}_t} = \Psi(\hat{Z}^{Wx_0}, \ldots, \hat{Z}^{Wx_{t-1}}, Z^{V_0})$$

[34] This roughly means that ϕ' has a polynomially bounded weak derivative; see Definition L.3.

[35] that may depend on various scalars such as $\mathring{\chi}_s$, $\mathbb{E}\, Z^{x_s} Z^{x_{s'}(\xi)}$, and $\mathbb{E}\, Z^{d\bar{h}_s} Z^{d\bar{h}_{s'}}$

for some function[35] $\Psi : \mathbb{R}^{t+1} \to \mathbb{R}$. *Then*

$$\theta_r \stackrel{def}{=} \mathbb{E}\,\partial\Psi(\hat{Z}^{Wx_0}, \ldots, \hat{Z}^{Wx_{t-1}}, Z^{V_0})/\partial\hat{Z}^{Wx_r}.$$

For example, generalizing Eq. (31), for any input ξ, we have

$$Z^{x_1(\xi)} = \Phi(Z^{W^\top d\bar{h}_0}, Z^{U_0}), \quad \text{where}$$
$$\Phi(z, u) \stackrel{\text{def}}{=} \phi(\xi u - \mathring{\chi}_0\xi_0\xi\phi'(\xi_0 u)z).$$

Then $\theta_0 = \mathbb{E}\,\partial_z\Phi(Z^{W^\top d\bar{h}_0}, Z^{U_0}) = -\mathring{\chi}_0\xi_0\xi\,\mathbb{E}\,\phi'(Z^{h_1(\xi)})\phi'(Z^{h_0})$, which specializes to c in Eq. (32). Altogether, $\dot{Z}^{Wx_1(\xi)} = -\mathring{\chi}_0\xi_0\xi Z^{d\bar{h}_0}\,\mathbb{E}\,\phi'(Z^{h_1(\xi)})\phi'(Z^{h_0})$.

Note that $\hat{Z}^{Wx_t}$ here does not equal Z^{Wx_t} in Eq. (26) in general, because the covariance $\mathrm{Cov}(\hat{Z}^{Wx_t}, \hat{Z}^{Wx_s}) = \mathbb{E}\,Z^{x_t}Z^{x_s}$ is affected by the presence of $\dot{Z}^{Wx_r}$ for all $r \le \max(s, t)$.

F.3. MLP of Arbitrary Depth

The μP limit of deeper MLPs can be derived along similar logic; see Appendices N.3 to N.5 for a rigorous treatment within the Tensor Programs framework, which also covers all stable abc-parametrizations.

What happens in other feature learning parametrizations If we are in the feature learning regime, then any W^l that is not maximally updated (Definition E.1) will be effectively fixed (to its initialized value) in the infinite-width limit (i.e. no learning occurs).

F.4. Summary of Main Intuitions for Deriving the μP Limit

Law of Large Numbers Any vector z has roughly iid coordinates given by Z^z. For any two vectors $z, z' \in \mathbb{R}^n$, $\frac{1}{n}\sum_{\alpha=1}^n z_\alpha z'_\alpha \to \mathbb{E}\,Z^z Z^{z'}$.

1. This is all we needed to derive the 1-hidden-layer dynamics of Section 5.1, since all the matrices there are size-n vectors.
2. In Appendices F.1 and F.2, this is also used in calculating the limit of $\Delta W_t x_t$.

Central Limit If the underlying computation graph never involves the transpose $W^\top$ of a $n \times n$ Gaussian matrix W in a matrix multiplication, then Wz is roughly iid Gaussian with coordinate $Z^{Wz} \stackrel{\text{d}}{=} \mathcal{N}(0, \mathbb{E}(Z^z)^2)$ (if $W_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$)

1. This along with the last intuition are all we used to derive the 2-hidden-layer decoupled dynamics of Appendix F.1, where W is the middle layer weight matrix.

(W, $W^\top$) Correlation If $W^\top$ is involved, then Wz has coordinates distributed like random variable $\hat{Z}^{Wz} + \dot{Z}^{Wz}$ where $\hat{Z}^{Wz}$ is the Gaussian obtained by pretending W is independent from $W^\top$, and $\dot{Z}^{Wz}$ results from the correlation between W and $W^\top$. $\dot{Z}^{Wz}$ is purely a linear combination of $Z^{z'}$ for previously defined vectors z' such that z depends on $W^\top z'$.

1. All three intuitions above are needed to derive the 2-hidden-layer dynamics of normal SGD (Appendix F.2), where $W^\top$ is used in backpropagation.
2. The calculation of $\dot{Z}^{Wx}$ is quite intricate, which is why we first discussed decoupled SGD in Appendix F.1, which doesn't need $\dot{Z}^{Wx}$ calculation, before discussing normal SGD in Appendix F.2.

G. Tensor Programs Framework

While the previous section demonstrates the intuition of how to derive the μP limit, it also lays bare 1) the increasing complexity of a manual derivation as the training goes on, as well as 2) the mounting uncertainty for whether the intuition still holds after many steps of SGD. This is a perfect call for the Tensor Programs framework, which automates (and makes rigorous) the limit derivation for any "computation graph" — including the computation graph underlying SGD. Here we review this framework (developed in Yang (2019a;b; 2020a;b)) in the context of μP limit. Fig. 5 graphically overviews the content of this section.

As seen abundantly in Section 5, the computation underlying SGD can be expressed purely via three instructions: matrix multiplication (by a Gaussian matrix, e.g. W_0x_0), coordinatewise nonlinearities (e.g. ϕ), and taking coordinatewise average (e.g. $\frac{1}{n}\sum_{\alpha=1}^n (nV_1)_\alpha x_{1\alpha}$). In deriving the μP SGD limit, we focused mostly on keeping track of $\mathbb{R}^n$ vectors (e.g. $\bar{x}_t$ or dh_t), but importantly we also computed scalars f_t and χ_t by (what amounts to) taking coordinatewise average (e.g. $f_1 = \frac{1}{n}\sum_{\alpha=1}^n (nV_1)_\alpha x_{1\alpha}$). We implicitly compute scalars as well inside $\Delta W_t x_t$. This motivates the following notion of a *program*, which can be thought of as a low-level symbolic representation of a computation graph common in deep learning (e.g. underlying `Tensorflow` and `Pytorch`).

Definition G.1. A *Tensor Program*[36] is a sequence of $\mathbb{R}^n$-vectors and $\mathbb{R}$-scalars inductively generated via one of the following ways from an initial set $\mathcal{C}$ of random scalars, $\mathcal{V}$ of random $\mathbb{R}^n$ vectors, and a set $\mathcal{W}$ of random $\mathbb{R}^{n\times n}$ matrices (which will be sampled with iid Gaussian entries in Setup G.2)

[36]What we refer to as Tensor Program is the same as NETSOR$\top^+$ in Yang (2020b); we will not talk about other languages (like NETSOR$\top$) so this should not cause any confusion

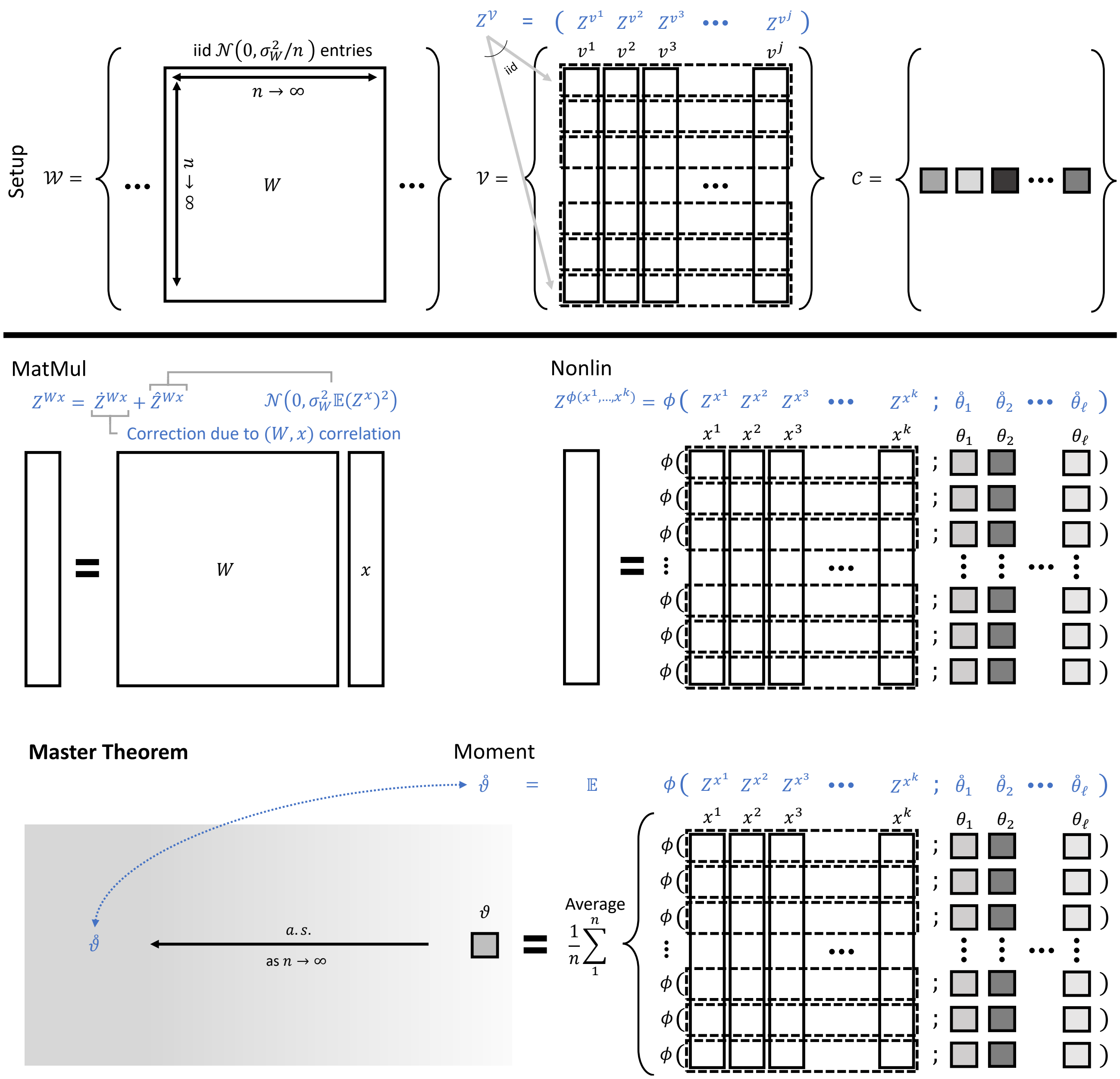

Figure 5. **Graphical overview of the Tensor Programs framework.** For the Master Theorem, we illustrate Theorem G.4(2) since Theorem G.4(1) is a corollary of Theorem G.4(2) for a larger program.

MatMul Given $W \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$, we can generate $Wx \in \mathbb{R}^n$ or $W^\top x \in \mathbb{R}^n$

Nonlin Given $\phi : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}$, previous scalars $\theta_1, \ldots, \theta_l \in \mathbb{R}$ and vectors $x^1, \ldots, x^k \in \mathbb{R}^n$, we can generate a new vector

$$\phi(x^1, \ldots, x^k; \theta_1, \ldots, \theta_l) \in \mathbb{R}^n$$

where $\phi(-; \theta_1, \ldots, \theta_l)$ applies coordinatewise to each "α-slice" $(x^1_\alpha, \ldots, x^k_\alpha)$.

Moment Given same setup as above, we can also generate a new scalar

$$\frac{1}{n}\sum_{\alpha=1}^{n} \phi(x^1_\alpha, \ldots, x^k_\alpha; \theta_1, \ldots, \theta_l) \in \mathbb{R}.$$

Explanation of Definition G.1 The *vectors* mentioned in Definition G.1 are exemplified by h_t, x_t, dh_t, dx_t in Section 5. The *scalars* mentioned are exemplified by f_t, χ_t as well as e.g. $x_s^\top x_t/n$ inside the calculating of h_t (Eq. (25)). The θ_is in **Nonlin** and **Moment** rules may appear cryptic at first. These scalars are not needed in the first forward and backward passes. But in the second forward pass, for example for the 1-hidden-layer MLP (Section 5.1), $x_1 = \phi(h_1) = \phi(\xi_1 U_0 - \chi_0 \xi_1 \xi_0 n V_0 \phi'(h_0))$ depends on the scalar χ_0, ξ_0, ξ_1, and can be written in the form of **Nonlin** as $\bar\phi(U_0, nV_0, h_0; \chi_0)$ for some $\bar\phi$ appropriately defined.

The *initial set of scalars* $\mathcal{C}$ is the training sequence $\{\xi_t, y_t\}_t$ for all three examples of Section 5. In our 2-hidden-layer MLP examples, the *initial set of matrices* $\mathcal{W}$ is $\{W\}$ (Appendix F.2) or $\{W, \widetilde{W}\}$ (Appendix F.1), i.e. the random $\mathbb{R}^{n\times n}$ Gaussian matrices. On the other hand, in the 1-hidden-layer MLP example (Section 5.1), $\mathcal{W}$ is empty. The *initial set of vectors* $\mathcal{V}$ in all three examples are $\mathcal{V} = \{U_0, nV_0\}$.[37][38] Notice how the vectors of these $\mathcal{V}$ are sampled with iid standard Gaussian coordinates. We formalize a more general setup for arbitrary Tensor Programs:

Setup G.2. *1) For each initial $W \in \mathcal{W}$, we sample iid $W_{\alpha\beta} \sim \mathcal{N}(0, \sigma_W^2/n)$ for some variance σ_W^2 associated to W, independent of other $W' \in \mathcal{W}$; 2) for some multivariate Gaussian $Z^{\mathcal{V}} = \left\{Z^h : h \in \mathcal{V}\right\} \in \mathbb{R}^{\mathcal{V}}$, we sample the initial set of vectors $\mathcal{V}$ like $\{h_\alpha : h \in \mathcal{V}\} \sim Z^{\mathcal{V}}$ iid for each $\alpha \in [n]$. 3) For each initial scalar $\theta \in \mathcal{C}$, we require $\theta \xrightarrow{\text{a.s.}} \mathring{\theta}$ for some deterministic $\mathring{\theta} \in \mathbb{R}$.*

[37] Here we write nV_0 instead of V_0 because we want all vectors to have $\Theta(1)$ coordinates; see Setup G.2.

[38] In Section 5 we assumed input dimension is 1. In general, each column of U_0 would be a separate initial vector. Likewise, if the output dimension is greater than 1, then each row of V_0 would be a separate initial vector.

In all of our examples, we took $\sigma_W^2 = 1$ for simplicity, but Setup G.2 allows for other initializations (e.g. a typical initialization for relu networks is $\sigma_W^2 = 2$); additionally, $Z^h, h \in \mathcal{V}$, are all standard Gaussians, independent from one another, since U_0, nV_0 are sampled this way; and our initial scalars $\{\xi_t, y_t\}_t$ are fixed with n, so they are their own limits.[39]

What Does a Tensor Program Vector Look Like? Recall that we represented the coordinate distribution of each vector h with a random variable Z^h in Section 5 and kept track of how different Zs are correlated with each other. We also calculated scalar limits like $f_t \to \mathring{f}_t, \chi_t \to \mathring{\chi}_t$. These calculations led to a set of formulas for the μP limit (e.g. Theorems 5.1, F.1 and F.2). We can also construct such Z^h and $\mathring{\theta}$ for vectors h and scalars θ in any Tensor Program. They intuitively capture the coordinate distribution of vector h and the deterministic limit of θ. The following definition formally defines Z^h and $\mathring{\theta}$, but the connection between Z^h (resp. $\mathring{\theta}$) and the coordinates of h (resp. θ) is not made rigorously until Theorem G.4 later. The **ZMatMul** rule below perhaps asks for some discussion, and we shall do so after the definition.

Definition G.3 (Z^h and $\mathring{\theta}$)**.** Given a Tensor Program, we recursively define Z^h for each vector h and $\mathring{\theta}$ for each scalar θ as follows.

ZInit If $h \in \mathcal{V}$, then Z^h is defined as in Setup G.2. We also set $\hat{Z}^h \stackrel{\text{def}}{=} Z^h$ and $\dot{Z}^h \stackrel{\text{def}}{=} 0$.

ZNonlin$^+$ Given $\phi : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}$, previous scalars $\theta_1, \ldots, \theta_l \in \mathbb{R}$ and vectors $x^1, \ldots, x^k \in \mathbb{R}^n$, we have

$$Z^{\phi(x^1, \ldots, x^k; \theta_1, \ldots, \theta_l)} \stackrel{\text{def}}{=} \phi(Z^{x^1}, \ldots, Z^{x^k}; \mathring{\theta}_1, \ldots, \mathring{\theta}_l).$$

ZMoment Given same setup as above and scalar $\theta = \frac{1}{n}\sum_{\alpha=1}^{n} \phi(x^1_\alpha, \ldots, x^k_\alpha; \theta_1, \ldots, \theta_l)$, then

$$\mathring{\theta} \stackrel{\text{def}}{=} \mathbb{E}\, \phi(Z^{x^1}, \ldots, Z^{x^k}; \mathring{\theta}_1, \ldots, \mathring{\theta}_l).$$

Here $\mathring{\theta}_1, \ldots, \mathring{\theta}_l$ are deterministic, so the expectation is taken over $Z^{x^1}, \ldots, Z^{x^k}$.

ZMatMul $Z^{Wx} \stackrel{\text{def}}{=} \hat{Z}^{Wx} + \dot{Z}^{Wx}$ for every matrix W (with $\mathcal{N}(0, \sigma_W^2/n)$ entries) and vector x, where

ZHat $\hat{Z}^{Wx}$ is a Gaussian variable with zero mean. Let $\mathcal{V}_W$ denote the set of all vectors in the program of the form Wy for some y. Then $\{\hat{Z}^{Wy} :$

[39] Since $\{\xi_t, y_t\}_t$ are fixed with n, we can WLOG absorb them into any nonlinearities in **Nonlin** that they are involved in, and set $\mathcal{C} = \emptyset$. But, in kernel regime or nonmaximal feature learning parametrization, we usually have initial scalars, such as $n^{-2a_{L+1}-c}$, that tend to 0 with n; see Appendix N.4.

$Wy \in \mathcal{V}_W\}$ is defined to be jointly Gaussian with zero mean and covariance

$$\mathrm{Cov}\left(\hat{Z}^{Wx}, \hat{Z}^{Wy}\right) \stackrel{\text{def}}{=} \sigma_W^2 \,\mathbb{E}\, Z^x Z^y,$$

for any $Wx, Wy \in \mathcal{V}_W$. Furthermore, $\{\hat{Z}^{Wy} : Wy \in \mathcal{V}_W\}$ is mutually independent from $\{\hat{Z}^v : v \in \mathcal{V} \cup \bigcup_{\bar{W} \neq W} \mathcal{V}_{\bar{W}}\}$, where $\bar{W}$ ranges over $\mathcal{W} \cup \{A^\top : A \in \mathcal{W}\}$.

ZDot We can always unwind $Z^x = \Phi(\cdots)$, for some arguments $(\cdots) = (\{\hat{Z}^{W^\top y^i}\}_{i=1}^k, \{\hat{Z}^{z^i}\}_{i=1}^j; \{\mathring{\theta}_i\}_{i=1}^l)$, $z^i \notin \mathcal{V}_{W^\top}$ (where $\mathcal{V}_{W^\top}$ is defined in **ZHat**), and deterministic function $\Phi : \mathbb{R}^{k+j+l} \to \mathbb{R}$. Define $\partial Z^x / \partial \hat{Z}^{W^\top y^i} \stackrel{\text{def}}{=} \partial_i \Phi(\cdots)$. Then we set

$$\dot{Z}^{Wx} \stackrel{\text{def}}{=} \sigma_W^2 \sum_{i=1}^k Z^{y^i}\, \mathbb{E}\, \frac{\partial Z^x}{\partial \hat{Z}^{W^\top y^i}}, \tag{33}$$

There is some nuance in this definition, so see Remark L.1 and L.2.

Explanation of Definition G.3 **Nonlin** and **Moment** should appear only natural. However, we pause to digest the meaning of **ZMatMul** by relating back to our examples in Section 5. First notice that $\dot{Z}^{Wx} = 0$ if $W^\top$ is not used in the program, so that $Z^{Wx} = \hat{Z}^{Wx}$. This is the case in Appendix F.1, where $\widetilde{W}$ is used in backprop instead of $W^\top$. There (in Eq. (26)), Z^{Wx_t} is Gaussian with covariance $\mathrm{Cov}(Z^{Wx_t}, Z^{Wx_s}) = \mathbb{E}\, Z^{x_t} Z^{x_s}$ for any s, consistent with **ZHat**. In Appendix F.2, however, $\dot{Z}^{Wx} \neq 0$ in general. The **ZDot** rule is a direct generalization of the calculation of $\dot{Z}$ in Theorem F.2.

$\dot{Z}^{Wx_t}$ and $\dot{Z}^{W^\top d\bar{h}_t}$ of Appendix F.2 for general t will all be nonzero but have no easy expression. Here we seek to convey the complexity of computing them; this is optional reading for the first time reader. To calculate $\dot{Z}^{Wx_t}$ ($\dot{Z}^{W^\top d\bar{h}_t}$ is similar), we need to express Z^{x_t} as a function of purely $\hat{Z}^{W^\top d\bar{h}_s}$, $s < t$, and $Z^{U_0} = \hat{Z}^{U_0}$. Then we symbolically differentiate Z^{x_t} by $\hat{Z}^{W^\top d\bar{h}_s}$ and take expectation to obtain the coefficient of $Z^{d\bar{h}_s}$ in $\dot{Z}^{Wx_t}$. For $t = 1$ as in the examples in Appendix F.2, this task is easy because $\hat{Z}^{W^\top d\bar{h}_0} = \hat{Z}^{dx_0} = Z^{dx_0}$. But in general, the calculation can balloon quickly. Indeed, note $Z^{x_t} = \phi(Z^{h_t})$ and

$$\begin{aligned} Z^{h_t} &= \xi_t Z^{U_t} = \xi_t Z^{U_0} - \xi_t \sum_{s=0}^{t-1} \mathring{\chi}_s \xi_s Z^{dh_s} \\ &= \xi_t Z^{U_0} - \xi_t \sum_{s=0}^{t-1} \mathring{\chi}_s \xi_s \phi'(Z^{h_s}) Z^{dx_s}. \end{aligned}$$

However, each Z^{dx_s} is a linear combination of $Z^{W^\top d\bar{h}_s} = \hat{Z}^{W^\top d\bar{h}_s} + \dot{Z}^{W^\top d\bar{h}_s}$ and $Z^{x_r}, r < s$ (coming from $\Delta W_t^\top d\bar{h}_s$). Each of $\dot{Z}^{W^\top d\bar{h}_s}$ and Z^{x_r} then needs to be recursively expanded in terms of $\hat{Z}$ before we can calculate the symbolic partial derivative $\partial Z^{x_t} / \partial \hat{Z}^{W^\top d\bar{h}_s}$.

Master Theorem Finally, we relate the *symbolic* nature of a Tensor Program given in Definition G.3 to the *analytic* limit of its computation, in the following *Master Theorem*. Pseudo-Lipschitz functions are, roughly speaking, functions whose (weak) derivatives are polynomially bounded. We state the theorem assuming mild regularity conditions (Assumption L.4) that roughly says most nonlinearities in the program should be pseudo-Lipschitz.

Theorem G.4 (Tensor Program Master Theorem, c.f. Theorem E.15 of (Yang, 2020b))**.** *Fix a Tensor Program initialized accordingly to Setup G.2. Adopt Assumption L.4. Then*

1. *For any fixed k and any pseudo-Lipschitz $\psi : \mathbb{R}^k \to \mathbb{R}$, as $n \to \infty$,*

$$\frac{1}{n}\sum_{\alpha=1}^n \psi(h_\alpha^1, \ldots, h_\alpha^k) \xrightarrow{\text{a.s.}} \mathbb{E}\, \psi(Z^{h^1}, \ldots, Z^{h^k}), \tag{34}$$

for any vectors $h^1, \ldots, h^k$ in the program, where Z^{h^i} are as defined in Definition G.3.

2. *Any scalar θ in the program tends to $\mathring{\theta}$ almost surely, where $\mathring{\theta}$ is as defined in Definition G.3.*

Intuitively, Theorem G.4(1) says that each "coordinate slice" $(h_\alpha^1, \ldots, h_\alpha^k)$ can be thought of as an iid copy of $(Z^{h^1}, \ldots, Z^{h^k})$.[40] This intuition is consistent with our heuristic derivation in Section 5, and Theorem G.4 underlies the proof of Theorems 5.1, F.1 and F.2. Theorem G.4(2) allows us to directly obtain the function learned at the end of training: For example, for a 1-hidden-layer MLP, it shows that the network's output on any input ξ at time t converges to $\mathring{f}_t(\xi)$ given in Theorem 5.1.

Algorithm 1 summarizes how to compute the infinite-width limit of any network in any abc-parametrization and for any task, using the Tensor Programs framework laid out in this section. It generalizes the manual derivations of Section 5. We carry out Algorithm 1 for MLPs in all of our experiments.

Architectural and algorithmic universality Given that Tensor Programs can express the first forward and backward computation of practically any architecture (Yang, 2019a; 2020a), it should perhaps come as no surprise that they can

[40] This implies an explicit convergence in distribution (see (Yang, 2020b)), but this convergence in distribution is strictly weaker than the formulation in Theorem G.4, which is in general much more useful.

Algorithm 1 Compute the infinite-width limit of an NN in any abc-parametrization and any task

1: Write the computation graph underlying training and inference in a Tensor Program (akin to writing low level PyTorch or Tensorflow code).
2: Calculate Z^h for each vector h and $\mathring{\theta}$ for each scalar θ in the program, according to Definition G.3.
3: The logits $f_t(\xi)$ of the neural network at any time t should be written as a collection of scalars, so $\mathring{f}_t(\xi)$ is calculated in the previous step. For t being inference time, $\mathring{f}_t(\xi)$ is the output of the infinite-width network after training.

also express practically any training and inference procedure — or just any computation — involving any such architecture. This includes both feature learning and kernel limits. We leverage this flexibility to derive and compute the μP and kernel limits for metalearning and Word2Vec; see Section 6.

Extensions We focused on programs whose vectors all have the same dimension n here. But it's easy to generalize to the case where vectors have different dimensions, which corresponds to e.g. when a network's widths are non-uniform. See (Yang, 2020b).

H. Computational Considerations

While the TP framework is very general, computing the feature learning limits analytically is inherently computationally intensive aside from special cases like the linear 1-hidden-layer MLP (Corollary 5.2). Here we explain why, so as to motivate our experimental choices below.

No closed-form formula for evaluating the expectations (e.g. in Eq. (34)) involving general nonlinearities except in special cases For example, for a 1-hidden-layer MLP (Section 5.1), after 1 step of SGD, the logit is of the form $\mathbb{E}(Z_1 + b\phi(Z_2))\phi(Z_3 + cZ_1\phi'(Z_2))$ where Z_is denote different (correlated) Gaussians (Eq. (7)). While one can still evaluate this via Monte-Carlo, the error will compound quickly with training time. On the other hand, because of the nesting of ϕ' inside ϕ, there is no closed-form formula for this expectation in general.

Notable Exception: If the nonlinearity ϕ is polynomial, then the expectation is a polynomial moment of a multivariate Gaussian and can be evaluated analytically, e.g. using Isserlis' theorem from the covariance matrix.

Even with nonlinear polynomial ϕ, there is exponential computational bottleneck As training time t increases, due to the nesting of ϕ and ϕ' in the preactivations, the integrand of the expectation, e.g. $\mathbb{E}\, Z^{\bar{x}_t} Z^{nV_t}$, will turn out to be a polynomial in $\Omega(1)$ Gaussian variables with degree $\Omega(2^t)$. The covariance matrix of the Gaussian variables will in general be nontrivial, so evaluating the expectation, e.g. using Isserlis' theorem, requires super-exponential time. This is because we would need to expand the polynomial integrand into monomials, and there would be $\Omega(2^t)$ monomials, each of which require $\Omega(2^t)$ time to evaluate using Isserlis' theorem.

$n \times n$ Gaussian matrices Both points above apply to 1-hidden-layer MLPs. Additional difficulties with deeper networks is caused by the $n \times n$ initial Gaussian matrix $W_0^l, 2 \leq l \leq L$, in the middle of the network. 1) In general, due to the nonlinearities, x_t^{l-1} would be linearly independent from x_s^{l-1} for all $s < t$. Therefore, in calculating $W_t^l x_t^{l-1} = W_0^l x_t^{l-1} + \Delta W_t^l x_t^{l-1}$, we create a new Gaussian variable $\hat{Z}^{W_0^l x_t^{l-1}}$ linearly independent from all previous $\hat{Z}^{W_0^l x_s^{l-1}}, s < t$. This then requires us to compute and store the covariance between them. Thus, t steps of SGD costs $\Omega(t^2)$ space and time (not mentioning that the computation of each covariance entry can require exponential time, as discussed above). 2) In addition, due to the interaction between W_t^l in the forward pass and $W_t^{l\top}$ in the backward pass, there is nonzero $\dot{Z}$, as demonstrated in Eq. (32). This $\dot{Z}$ is generally a linear combination of $\Omega(t)$ terms, and the coefficients of this combination require evaluation of some expectations that typically run into the exponential bottleneck discussed above.

Summary From easiest to hardest in terms of μP limit's computational cost, we have 1) 1-hidden-layer linear networks; 2) L-hidden-layer linear MLP, $L \geq 2$; 3) nonlinear MLP with polynomial activations; 4) nonlinear MLP with nonpolynomial activations. Nevertheless, 1-hidden-layer linear networks are more than sufficient to demonstrate feature learning in Word2Vec and few-shot learning with MAML, as we show below.

I. Assumptions

I.1. Assumptions of Section 3

Assumption I.1. *Our main results in Section 3 (and this section only) will assume ϕ is either tanh or a smooth version of relu called σ-gelu (see Definition N.1), for sufficiently small $\sigma > 0$ (which means σ-gelu approximates relu arbitrarily well).*

Note this assumption is only needed for the classification of abc-parametrizations. For deriving the infinite-width limits, the much weaker Assumption N.21 suffices. We believe our results here will hold for generic nonlinearities, but making this precise is outside our scope. (See Remark N.14 for some discussion).

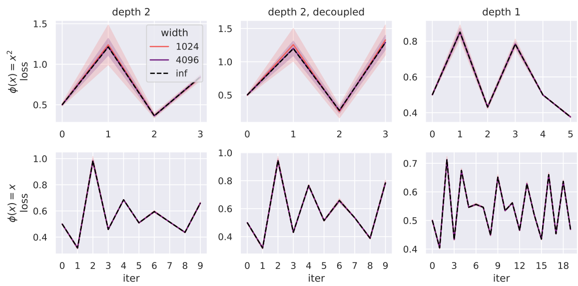

Figure 6. **Empirical Simulation Agrees with Theory.** We analytically compute the infinite-width μP limit for the three kinds of networks (depth 1, depth 2 decoupled, depth 2) described in Section 5, with either quadratic $\phi(x) = x^2$ or linear $\phi(x) = x$ activation. The training set is random $\xi_t \in \{\pm 1\}, y_t \in \{\pm 1\}$, so that the deviation of finite width from infinite width losses are accentuated. We compare against finite width μP networks with width 1024 or 4096. For each width, we randomly initialize with 100 different seeds and aggregate the loss curves. The mean across these seeds is plotted as solid curves, and the standard deviation represented by the shade. As discussed in Appendix H, nonlinear activation functions and higher depth face computational difficulties exponential with training time. Thus here we only train for a few steps. We observe that the quadratic network converges slower to the limit with width. This is expected since the tail of Z^{x_t} is fatter for a quadratic activation than a linear activation.

J. Experiments

J.1. Verifying the Theory

In Fig. 6, we analytically computed the μP limits derived in Section 5 for quadratic and linear activations, and verified them against finite width networks.

J.2. Few-Shot Learning on Omniglot via First Order MAML

J.2.1. Overview

MAML In Model Agnostic Meta-Learning (MAML), the model performs few-shot learning by one or more SGD steps on the given training data; this is called *adaptation*. In a pretraining (also called *meta-training*) phase, MAML learns a *good initialization* of the model parameters for this adaptation. The training objective is to minimize the loss on a random task's test set after the model has adapted to its training set. More precisely, the basic *First Order* MAML at training time goes as follows: With f_θ denoting the model with parameters θ, and with step sizes ϵ, η, we do

1. At each time point, sample a few-shot task $\mathcal{T}$
2. From $\mathcal{T}$, sample a training set $\mathcal{D}$
3. Adapt $\theta' \leftarrow \theta - \epsilon \nabla_\theta \mathcal{L}_\mathcal{D}(f_\theta)$, where $\mathcal{L}_\mathcal{D}(f_\theta)$ is the loss of f_θ over $\mathcal{D}$
4. Sample a test set $\mathcal{D}'$ from $\mathcal{T}$
5. Update $\theta \leftarrow \theta - \eta \nabla_{\theta'} \mathcal{L}_{\mathcal{D}'}(f_{\theta'})$, where $\mathcal{L}_{\mathcal{D}'}(f_{\theta'})$ is the loss of $f_{\theta'}$ over $\mathcal{D}'$
6. Repeat

In practice, we batch the tasks, just like batches in SGD, so that we accumulate all the gradients from Step 5 and update θ only at the end of the batch.

During *meta-test* time, we are tested on random unseen few-shot tasks, where each task $\mathcal{T}$ provides a training set $\mathcal{D}$ and a test set $\mathcal{D}'$ as during meta-training. We adapt to $\mathcal{D}$ as in Step 3 above (or more generally we can take multiple gradient steps to adapt better) to obtain adapted parameters θ'. Finally, we calculate the accuracy of θ' on the test set $\mathcal{D}$. We average this accuracy over many tasks $\mathcal{T}$, which we report as the *meta-test accuracy*.

First Order vs Second Order MAML Notice in Step 5, we take the gradient of $\mathcal{L}_{\mathcal{D}'}(f_{\theta'})$ with respect to the adapted parameters θ'. In *Second Order* MAML, we would instead take the gradient against the unadapted parameters θ, which would involve the Hessian $\nabla_\theta \nabla_\theta \mathcal{L}_\mathcal{D}(f_\theta)$. Second Order MAML generally achieves performance slightly better than First Order MAML, but at the cost of significantly slower updates (Nichol et al., 2018). In order to scale up, we will focus on First Order MAML, hereafter referred to as just MAML.

Few-Shot Learning Terminologies An N-way classification task asks the model to predict a class from N possibilities. A K-shot classification task provides K input/output pairs per class, for a total of NK training points for N-way classification.

Omniglot Omniglot is a standard few-shot learning benchmark. It consists of 20 instances of 1623 characters from 50 different alphabets, each handwritten by a different person. We test our models on 1-shot 5-way classification: We draw 5 random characters, along with 1 training instance and 1 test instance for each character. After the model adapts to the training instances, it's asked to predict the character of the test instances (choosing among the 5 characters).

Hyperparameters We use (task) batch size 32 and adaptation step size 0.4 (ϵ in Step 3). We also clip the gradient in Step 5 if the gradient has norm ≥ 0.5.[41] For each model, we tune its weight initializaton variances and the meta learning rate (η in Step 5). During meta-test time, we take 20

[41] One can write down gradient clipping easily in a Tensor Program, so the its infinite-width limit can be computed straightforwardly via Theorem G.4; see Algorithms 2 and 3.

gradient steps during adaptation (i.e. we loop Step 3 above 20 times to obtain θ').

J.2.2. LINEAR 1-HIDDEN-LAYER μP NETWORK

We discuss the implementation details for our μP network. We consider a linear 1-hidden-layer MLP with bias, input dimension d, output dimension d_o, given by

$$f(\xi) = Vh(\xi) \in \mathbb{R}^{d_o}, \quad h(\xi) = U\xi + B \in \mathbb{R}^n,$$

where $\xi \in \mathbb{R}^d$. Following μP, we factor $U = \sqrt{n}u \in \mathbb{R}^{n\times d}, V = \frac{1}{\sqrt{n}}v \in \mathbb{R}^{d_o\times n}, B = \alpha\sqrt{n}\beta \in \mathbb{R}^n$, where u, v, β are the trainable parameters. We initialize $u_{\alpha\beta} \sim \mathcal{N}(0, \sigma_u^2/n), v_{\alpha\beta} \sim \mathcal{N}(0, \sigma_v^2/n)$, $\beta = 0 \in \mathbb{R}^n$. We can cancel the factors of $\sqrt{n}$ and rewrite

$$f(\xi) = vh(\xi) \in \mathbb{R}^{d_o}, \quad h(\xi) = u\xi + b \in \mathbb{R}^n,$$

where $b = \alpha\beta$. We will also consider gradient clipping with threshold g and weight decay with coefficient γ. So in summary, the hyperparameters are

$$\sigma_u, \sigma_v \text{ (init. std.)}, \quad \alpha \text{ (bias multiplier)}, \quad \eta \text{ (LR)},$$
$$g \text{ (grad. clip)}, \quad \gamma \text{ (weight decay)}.$$

As in Corollary 5.2, it's easy to see that each column of u_t at any time t is always a linear combination of the columns of u_0 and the rows of v_0 such that the coefficients of these linear combinations converge deterministically in the $n \to \infty$ limit; likewise for b_t and the rows of v_t. To track the evolution of f, it suffices to track these coefficients. Therefore, for implementation, we reparametrize as follows:

Coefficient matrix and vector Let $\mu_1, \ldots, \mu_d, \nu_1, \ldots, \nu_{d_o} \in \mathbb{R}^n$ be standard Gaussian vectors such that the columns of u_0 will be initialized as $\sigma_u\mu_1/\sqrt{n}, \ldots, \sigma_u\mu_d/\sqrt{n}$ and the rows of V_0 will be initialized as $\sigma_v\nu_1/\sqrt{n}, \ldots, \sigma_v\nu_{d_o}/\sqrt{n}$. Write $\mu = (\mu_1, \ldots, \mu_d) \in \mathbb{R}^{n\times d}, \nu = (\nu_1, \ldots, \nu_{d_o}) \in \mathbb{R}^{n\times d_o}$. Define coefficient matrices

$$\boldsymbol{u}^\top \in \mathbb{R}^{d\times(d+d_o)}, \boldsymbol{v} \in \mathbb{R}^{d_o\times(d+d_o)},$$

such that at any time, $(u, v^\top) \in \mathbb{R}^{n\times(d+d_o)}$ is $\frac{1}{\sqrt{n}}(\mu, \nu)(\boldsymbol{u}, \boldsymbol{v}^\top)$ in the infinite-width limit. We initialize

$$\begin{pmatrix} \boldsymbol{u}^\top \\ \boldsymbol{v} \end{pmatrix} \leftarrow \begin{pmatrix} \sigma_u I & 0 \\ 0 & \sigma_v I \end{pmatrix},$$

i.e. a "diagonal" initialization. Likewise, define coefficient vector $\boldsymbol{b} \in \mathbb{R}^{d+d_o}$, initialized at 0, such that, at any time, b is approximately distributed as $\frac{1}{\sqrt{n}}(\mu, \nu)\boldsymbol{b}$. To track the evolution of the infinite-width network, we will track the evolution of $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{b}$.

Algorithm 2 SGD Training of Finite-Width Linear μP 1-Hidden-Layer Network

Require: Hyperparameters $n, \sigma_u, \sigma_v, \alpha, \eta, g, \gamma$.
Initialize $u_{\alpha\beta} \sim \mathcal{N}(0, \sigma_u^2/n)$
Initialize $v_{\alpha\beta} \sim \mathcal{N}(0, \sigma_v^2/n)$
Initialize $b \leftarrow 0$
for each batch of inputs $\Xi \in \mathbb{R}^{B\times d}$ and labels $Y \in \mathbb{R}^{B\times d_o}$ **do**
  // Forward Pass
  $H \leftarrow \Xi u^\top + b \in \mathbb{R}^{B\times n}$
  $f(\Xi) \leftarrow Hv^\top \in \mathbb{R}^{B\times d_o}$
  // Backward Pass
  $\chi \leftarrow \mathcal{L}'(f(\Xi), Y) \in \mathbb{R}^{B\times d_o}$
  $du \leftarrow -v^\top \chi^\top \Xi \in \mathbb{R}^{n\times d}$
  $dv \leftarrow -\chi^\top H \in \mathbb{R}^{d_o\times n}$
  $db \leftarrow -\alpha^2 \mathbf{1}^\top \chi v \in \mathbb{R}^n$
  // Gradient Clipping
  $G \leftarrow \sqrt{\|du\|_F^2 + \|dv\|_F^2 + \|\frac{db}{\alpha}\|^2}$
  $\rho \leftarrow \min(1, g/G)$
  $du \leftarrow \rho du$
  $dv \leftarrow \rho dv$
  $db \leftarrow \rho db$
  // Gradient Step w/ Weight Decay
  $u \mathrel{+}= \eta du - \eta\gamma u \in \mathbb{R}^{d\times n}$
  $v \mathrel{+}= \eta dv - \eta\gamma v \in \mathbb{R}^{d_o\times n}$
  $b \mathrel{+}= \eta db - \eta\gamma b \in \mathbb{R}^n$
end for

In general, we use **bold** to denote the coefficients (in μ, ν) of a tensor (e.g. $\boldsymbol{b}$ for coefficients of b). We also use capital letters to denote the batched version (e.g. H for batched version of h). Algorithms 2 and 3 below summarize the SGD training of the finite- and the infinite-width networks. Note that aside from initialization and the hidden size (n vs $d + d_o$), the algorithms are essentially identical.

During inference, we just run the *Forward Pass* section with Ξ substituted with test data.

The algorithms for MAML can then be obtained by a straightforward modification of these algorithms. (Note that in MAML, we do not clip gradients during adaptation, but rather clip the gradient against the validation loss of task; we also disable weight decay by setting the coefficient γ to 0).

Hyperparameter Sweep We sweep σ_u, σ_v, η and α with the following grid for finite width and μP networks.

- $\sigma_u : [0.5, 1, 2, 4, 8]$,
- $\sigma_v : [2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}]$,
- $\eta : [0.025, 0.05, 0.1, 0.2, 0.4]$,

Algorithm 3 SGD Training of Infinite-Width Linear μP 1-Hidden-Layer Network

Require: Hyperparameters $\sigma_u, \sigma_v, \alpha, \eta, g, \gamma$.

Initialize $\boldsymbol{u}^\top \leftarrow (\sigma_u I, 0)$
Initialize $\boldsymbol{v} \leftarrow (0, \sigma_v I)$
Initialize $\boldsymbol{b} \leftarrow 0$
for each batch of inputs $\Xi \in \mathbb{R}^{B \times d}$ and labels $Y \in \mathbb{R}^{B \times d_o}$ **do**
  // Forward Pass
  $\boldsymbol{H} \leftarrow \Xi \boldsymbol{u}^\top + \boldsymbol{b} \in \mathbb{R}^{B \times (d+d_o)}$
  $f(\Xi) \leftarrow \boldsymbol{H} \boldsymbol{v}^\top \in \mathbb{R}^{B \times d_o}$
  // Backward Pass
  $\chi \leftarrow \mathcal{L}'(f(\Xi), Y) \in \mathbb{R}^{B \times d_o}$
  $d\boldsymbol{u} \leftarrow -\boldsymbol{v}^\top \chi^\top \Xi \in \mathbb{R}^{(d+d_o) \times d}$
  $d\boldsymbol{v} \leftarrow -\chi^\top \boldsymbol{H} \in \mathbb{R}^{d_o \times (d+d_o)}$
  $d\boldsymbol{b} \leftarrow -\alpha^2 \mathbf{1}^\top \chi \boldsymbol{v} \in \mathbb{R}^{d+d_o}$
  // Gradient Clipping
  $G \leftarrow \sqrt{\|d\boldsymbol{u}\|_F^2 + \|d\boldsymbol{v}\|_F^2 + \|\frac{d\boldsymbol{b}}{\alpha}\|^2}$
  $\rho \leftarrow \min(1, g/G)$
  $d\boldsymbol{u} \leftarrow \rho d\boldsymbol{u}$
  $d\boldsymbol{v} \leftarrow \rho d\boldsymbol{v}$
  $d\boldsymbol{b} \leftarrow \rho d\boldsymbol{b}$
  // Gradient Step w/ Weight Decay
  $\boldsymbol{u} \mathrel{+}= \eta d\boldsymbol{u} - \eta\gamma\boldsymbol{u} \in \mathbb{R}^{(d+d_o) \times d}$
  $\boldsymbol{v} \mathrel{+}= \eta d\boldsymbol{v} - \eta\gamma\boldsymbol{v} \in \mathbb{R}^{d_o \times (d+d_o)}$
  $\boldsymbol{b} \mathrel{+}= \eta d\boldsymbol{b} - \eta\gamma\boldsymbol{b} \in \mathbb{R}^{d+d_o}$
end for

- α : $[0.25, 0.5, 1, 2, 4]$

We are interested in 1-shot, 5-way learning with Omniglot. This means that each task provides 5 training samples, each corresponding to one of the 5 labels of the task. Each hyperparameter combination above is used to train for 100 epochs over 3 random seeds, where each epoch consists of 100 batches of 32 tasks. We average the validation accuracy across the last 10 epochs and document the best hyperparameters in Table 4, along with the test accuracy from a 15-seed rerun[42] for better benchmarking. For NTK and GP, we additionally tune the initialization σ_b for biases, which is set to 0 for both finite and μP networks for simplicity.

J.2.3. NNGP and NTK for Relu Networks

We discuss the implementation details for our relu NTK and GP baselines.

Consider a kernel K, which in our case will be the NNGP or NTK of a 1-hidden-layer relu network. WLOG, it is induced by an embedding Φ such that $K(\xi, \zeta) = \langle \Phi(\xi), \Phi(\zeta) \rangle$ where $\langle , \rangle$ is the inner product in the embedding space; we do not care about the details of Φ or $\langle , \rangle$ as eventually our algorithm only depends on K.

In our setting, we will train a linear layer W on top of Φ via MAML, $f(\xi) \stackrel{\text{def}}{=} \langle W, \Phi(\xi) \rangle$. One can see easily that W is always a linear combination of $\Phi(\zeta)$ for various ζ from the training set we've seen so far. Thus, to track W, it suffices to keep an array Q of pairs (ζ, q) such that $W = \sum_{(\zeta,q) \in Q} q\Phi(\zeta)$ at all times. Let f_Q be the function with W given by Q. Then

$$f_Q(\xi) = \sum_{(\zeta, q_\zeta) \in Q} q_\zeta K(\zeta, \xi).$$

In our case, the number of possible inputs is too large to instantiate a value q for every ζ, so we gradually grow a dynamic array Q, which we model as a stack. Then MAML can be implemented as in Algorithm 4.

Algorithm 4 MAML Training of Kernel Model with Kernel K

Require: Kernel K, adaptation step size ϵ, meta learning rate η, batch size B, gradient clip g

Initialize $Q = \{\}$
while True **do**
  Draw a batch of tasks
  for each task in batch **do**
    // Adaptation
    Sample training set $\mathcal{D}$
    for each input/label pair $(\xi_i, y_i) \in \mathcal{D}$ **do**
      $\chi_i \leftarrow \mathcal{L}'(f_Q(\xi_i), y_i)$
    end for
    for each input/label pair $(\xi_i, y_i) \in \mathcal{D}$ **do**
      $Q.\texttt{push}((\xi_i, -\epsilon\chi_i))$
    end for
    // Calculate Test Set Gradient
    Sample test set $\hat{\mathcal{D}}$
    for each input/label pair $(\hat{\xi}_i, \hat{y}_i) \in \hat{\mathcal{D}}$ **do**
      $\hat{\chi}_i \leftarrow \mathcal{L}'(f_Q(\hat{\xi}_i), \hat{y}_i)$
    end for
    for each input/label pair $(\xi_i, y_i) \in \mathcal{D}$ **do**
      $Q.\texttt{pop}((\xi_i, -\epsilon\chi_i))$
    end for
    // Gradient Clip
    $G \leftarrow \sqrt{\sum_{(\hat{\xi}_i, \hat{y}_i) \in \hat{\mathcal{D}}} \sum_{(\hat{\xi}_j, \hat{y}_j) \in \hat{\mathcal{D}}} \hat{\chi}_i \hat{\chi}_j K(\hat{\xi}_i, \hat{\xi}_j)}$
    $\rho \leftarrow \min(1, g/G)$
    // Gradient Update
    for each input/label pair $(\hat{\xi}_i, \hat{y}_i) \in \hat{\mathcal{D}}$ **do**
      $Q.\texttt{push}((\hat{\xi}_i, -\rho\eta\hat{\chi}_i))$
    end for
  end for
end while

[42] After excluding outliers at least one standard deviation away from the mean.

Table 4. **Best hyperparameters for the MAML experiment.**

$\log_2$Width/Limit	σ_u	σ_v	σ_b	η	α	Val. Acc. (%)	Test Acc. (%)
1	0.5	0.5	-	0.05	2	46.72 ± 4.30	55.34 ± 1.24
3	0.5	0.25	-	0.1	1	$65.30 \pm .27$	$64.54 \pm .70$
5	1	0.125	-	0.4	0.5	$68.74 \pm .18$	$66.21 \pm .15$
7	1	0.125	-	0.1	1	$69.03 \pm .04$	$66.31 \pm .16$
9	1	0.03125	-	0.1	1	$69.32 \pm .07$	$66.43 \pm .23$
11	1	0.03125	-	0.1	1	$69.27 \pm .11$	$66.36 \pm .22$
13	1	0.03125	-	0.1	1	$69.27 \pm .14$	$66.41 \pm .18$
μP	1	0.03125	-	0.1	1	$69.26 \pm .13$	$66.42 \pm .19$
NTK	0.25	1	1	0.05	1	$47.47 \pm .13$	$47.82 \pm .04$
GP	1	0.25	1	0.05	1	$38.92 \pm .15$	$47.60 \pm .02$

Hyperparameter Sweep We sweep σ_u, σ_v, σ_b and η with the following grid for GP and NTK.

- $\sigma_u : [0.25, 0.5, 1, 2, 4]$,
- $\sigma_v : [0.25, 0.5, 1, 2, 4]$,
- $\sigma_b : [0.25, 0.5, 1, 2, 4]$,
- $\eta : [0.05, 0.1, 0.2, 0.4, 0.8]$

Each hyperparameter combination above is used to train for 5 epochs (the first epoch is almost always the best) over 3 random seeds, where each epoch consists of 100 batches of 32 tasks. We take the validation accuracy among all epochs and document the best hyperparameters in Table 4, along with the test accuracy from a 15-seed rerun.

J.3. Word2Vec

Word2Vec Pretraining Consider training on a corpus with vocabulary $\mathcal{V}$. At each time step, we sample a sentence for the corpus and choose a word $i \in \mathcal{V}$. This word's context $J \subseteq \mathcal{V}$ is a window of words around it in the sentence, thought of as a bag of words. Let $\xi^i \in \mathbb{R}^{|\mathcal{V}|}$ be the one-hot vector corresponding to word i. We pass the averaged context $\xi^J \stackrel{\text{def}}{=} \frac{1}{|J|} \sum_{j \in J}^{n} \xi^j$ through a 1-hidden-layer MLP with hidden size n and identity activation:

$$f(\xi^J) = V h(\xi^J) \in \mathbb{R}^{|\mathcal{V}|}, \quad h(\xi^J) = U \xi^J \in \mathbb{R}^n, \quad (35)$$

where $V \in \mathbb{R}^{|\mathcal{V}| \times n}, U \in \mathbb{R}^{n \times |\mathcal{V}|}$ factor as $V = n^{-a_v} v, U = n^{-a_u} u$ with initialization $v_\alpha \sim \mathcal{N}(0, n^{-2b_v}), u_\alpha \sim \mathcal{N}(0, n^{-2b_u})$, where $\{a_v, b_v, a_u, b_u\}$ specify the parametrization of the network. After each forward pass, we sample a target word τ from $\mathcal{V}$: with probability p, we take $\tau = i$; with probability $1-p$, we sample τ uniformly from $\mathcal{V} \setminus \{i\}$. Following (Mikolov et al., 2013a;b), we take $p = 1/21 \approx 4.76\%$. The loss is then calculated with the Sigmoid function $\sigma(\cdot)$:

$$\mathcal{L}(f(\xi^J), \xi^\tau) = \begin{cases} \log(1 - \sigma(f(\xi^J)^\top \xi^\tau)) & \tau = i \\ \log \sigma(f(\xi^J)^\top \xi^\tau) & \tau \neq i \end{cases} \quad (36)$$

Then v and u are updated via SGD as usual (causing V and U to update). Conventionally, $h(\xi) \in \mathbb{R}^n$ is taken as the Word2Vec embedding for a word ξ after many iterations of forward-backward updates.

Word Analogy Evaluation We evaluate the word embeddings $h(\xi)$ with the word analogy task. This task asks the question of the kind: *What to a 'queen' is as a 'man' to a 'woman'?* (answer is 'king'). The Word2Vec model answers this question by computing

$$\operatorname*{argmax}_i h(\xi^i)^\top (h(\xi^{\text{'man'}}) - h(\xi^{\text{'woman'}}) + h(\xi^{\text{'queen'}})) \quad (37)$$

where i ranges over $\mathcal{V} \setminus \{$'man', 'woman', 'queen'$\}$. If the argmax here is $i =$ 'king', then the model answers correctly; otherwise, it's incorrect. The accuracy score is the percentage of such questions answered correctly.

Dataset We train the models on `text8`,[43] a clean dataset consisting of the first 100 million characters of a 2006 Wikipedia dump. The dataset has been featured in the original Word2Vec codebase and the Hutter Prize. `text8` contains the first 100 million characters of `fil9`, a larger dataset obtained by filtering the first 1 billion characters in the aforementioned Wikipedia dump. We space-separate the datasets into tokens and keep ones that appear no less than 5 times in the entire dataset for `text8` and 10 times for `fil9`. The resulting datasets have 71,291 and 142,276 unique vocabulary items.

[43] `http://mattmahoney.net/dc/textdata.html`

J.3.1. IMPLEMENTATION OF μP LIMIT

We shall derive the training algorithm for μP Word2Vec. First, we introduce the notation for word embeddings. We denote $\Phi^i \stackrel{\text{def}}{=} h(\xi^i)$. If ξ^i is a one-hot vector with the i^{th} element set to 1, Φ^i is essentially the i^{th} column of the weight matrix U. We also define the following shorthands for the context embedding: $\Phi^J \stackrel{\text{def}}{=} \mathbb{E}_{j\in J}\,\Phi^j = h(\xi^J)$. Similarly, $V^\top\xi^\tau$ describes a row in V; we can define $\Phi^{\hat{\tau}} \stackrel{\text{def}}{=} \hat{h}(\xi^\tau) \stackrel{\text{def}}{=} V^\top\xi^\tau$ and rewrite the loss function.

$$\mathcal{L}(f(\xi^J), \xi^\tau) = \begin{cases} \log(1-\sigma(\Phi^{J\top}\Phi^{\hat{\tau}})) & \tau = i \\ \log\sigma(\Phi^{J\top}\Phi^{\hat{\tau}}) & \tau \neq i. \end{cases} \tag{38}$$

Consequently, the backward pass becomes:

$$\begin{aligned} \Delta\Phi^j &= \frac{1}{|J|}\Delta\Phi^J = \frac{\eta}{|J|}\frac{\partial\mathcal{L}}{\partial\Phi^J} \\ &= \begin{cases} \frac{\eta}{|J|}\Phi^{\hat{\tau}}(1-\sigma(\Phi^{J\top}\Phi^{\hat{\tau}})) & \tau = i \\ -\frac{\eta}{|J|}\Phi^{\hat{\tau}}\sigma(\Phi^{J\top}\Phi^{\hat{\tau}}) & \tau \neq i. \end{cases} \end{aligned} \tag{39}$$

Following μP, we initialize $U_{\alpha\beta} \sim \mathcal{N}(0, \sigma_u n^{-1})$ and $V_{\alpha\beta} \sim \mathcal{N}(0, \sigma_v n^{-1})$, where n is the width of the finite network. (Here the explicit multipliers of $\sqrt{n}$ in U and $1/\sqrt{n}$ in V cancel out because the network is linear). The tunable hyperparameters are the initialization std $\sigma_u$and σ_v, learning rate η and weight decay ratio γ. Rather than tuning the hyperparameters extensively for each width, we pick some reasonable values and use them for all of our experiments. Specifically, we have $\sigma_u = \sigma_v = 1$, $\eta = 0.05$ and $\gamma = 0.001$.

Again, using Corollary 5.2, we can train the μP limit in the coefficient space of $\boldsymbol{u}^\top \in \mathbb{R}^{|\mathcal{V}|\times 2|\mathcal{V}|}$, $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{V}|\times 2|\mathcal{V}|}$, with the same "diagonal" initialization:

$$\begin{pmatrix}\boldsymbol{u}^\top \\ \boldsymbol{v}\end{pmatrix} \leftarrow \begin{pmatrix}\sigma_u I & 0 \\ 0 & \sigma_v I\end{pmatrix},$$

We can adopt the embedding notation and represent a row of $\boldsymbol{u}$ with the embedding coefficient vector $\boldsymbol{\Phi}^\bullet$ and a column of $\boldsymbol{v}$ with $\boldsymbol{\Phi}^{\hat{\bullet}}$. This is computationally equivalent to training with a hidden size of $2|\mathcal{V}|$ and with embeddings initialized as rows (or columns) of one-hot vectors. The full algorithm is described in Algorithm 2 and Algorithm 3; in this case, we remove biases and use weight decay with coefficient $\gamma = 0.001$. After training, rows of the weight matrix u (resp. coefficient matrix $\boldsymbol{u}$), i.e. $\Phi^\bullet$ (resp. $\boldsymbol{\Phi}^\bullet$), are taken as the word vectors.

J.3.2. IMPLEMENTATION OF NTK LIMIT

In the NTK parametrization, V and U in Eq. (35) factor as $V = \frac{1}{\sqrt{n}}v$ and $U = u$, and the learning rate is $\Theta(1)$. Each column $U_{\bullet i}$ of U is equal to $h(\xi^i)$. At any fixed time t, it is easy to see via Tensor Programs that

$$h_t(\xi^i) = h_0(\xi^i) + \sum_{j\in\mathcal{V}} O(1/\sqrt{n})v_j + O_{coord}(1/n)$$

where v_j denotes the jth row of v at initialization, and where $O_{coord}(1/n)$ means a vector that is $O(1/n)$ coordinatewise. Recall that $U = u$ and v are initialized with iid standard Gaussian entries. Because ξ^i is one-hot, this in particular implies $h_0(\xi^i)$ has standard Gaussian entries, and $h_0(\xi^i)$ is independent from $h_0(\xi^j)$ for $i \neq j$. Then for any $i \neq j$,

$$\frac{1}{\sqrt{n}}h_t(\xi^i)^\top h_t(\xi^j) - \frac{1}{\sqrt{n}}h_0(\xi^i)^\top h_0(\xi^j) \xrightarrow{\text{a.s.}} 0,$$
$$\frac{1}{\sqrt{n}}h_0(\xi^i)^\top h_0(\xi^j) \xrightarrow{\text{d}} \mathcal{N}(0,1)$$

by Law of Large Numbers (or more formally, Theorem G.4) and Central Limit Theorem. In other words, $\frac{1}{\sqrt{n}}h_0(\xi^i)^\top h_0(\xi^j)$ is distributed completely randomly, with no regard to the semantic similarities of i and j. Likewise, the inner product in Eq. (37) is random, and the argmax is a uniform sample.[44] Therefore, in the NTK limit, Word2Vec gives random answers and achieves an accuracy of $\frac{1}{|\mathcal{V}|-3}$.

K. abc-Parametrization for General Neural Architectures

We can straightforwardly generalize abc-parametrizations to an arbitrary neural architecture. Each parameter tensor W would get its own a_W and b_W, such that $W = n^{-a_W}w$ and w is the actual trainable parameter with initialization $w_{\alpha\beta} \sim \mathcal{N}(0, n^{-2b_W})$. The learning rate is still ηn^{-c} for some fixed η.

K.1. Maximal Update Parametrization

MLP with Biases Suppose in Eq. (1), for each $l \in [L]$, we have $h^l(\xi) = W^l x^{l-1}(\xi) + b^l$ instead, for bias $b^l \in \mathbb{R}^n$. Then in μP, the bias b^l should have $a_{b^l} = -1/2$ and $b_{b^l} = 1/2$. We can also have bias b^{L+1} in the logits $f(\xi) = W^{L+1}x^L(\xi) + b^{L+1}$. Then we set $a_{b^{L+1}} = b_{b^{L+1}} = 0$.

General Neural Architectures More generally, μP can be defined easily for any neural architecture whose forward pass can be written down as a Tensor Program (e.g. ResNet or Transformer; see (Yang, 2019a) for explicit programs). The learning rate is always independent of width, i.e. $c = 0$. For any parameter tensor W, b_W is always $1/2$, and a_W can be defined as follows: If W is not an output weight

[44] Here the randomness comes from initialization: the argmax is different for different random initializations, but it is fixed throughout training in the large width limit.

matrix, then a_W should be set to $-1 + \frac{1}{2}p_W$, where $p_W = \lim_{n\to\infty} \log_n \#(W)$ is a) 0 if both sides of W are fixed w.r.t. n; b) 1 if W is a vector (e.g. bias) or with one side being fixed dimensional (e.g. W^1); and c) 2 if W is a matrix with both sides scaling like n (e.g. weights in the middle of an MLP). If W is an output weight matrix (and thus the output dimension is fixed w.r.t. n), then a_W should be $\frac{1}{2}$. If W is an output bias, then a_W should be 0.

Optimality Properties One can formalize, in this general context, the notion of *stability* and the notions of a parameter tensor being *updated maximally* and (a set of readout weights) being initialized maximally. Then one can show that μP is the unique stable abc-parametrization such that all of its parameter tensors are updated maximally and all of its readout weights are initialized maximally.

L. Nuances of the Master Theorem

Remark L.1 (Partial derivative)**.** The partial derivative in **ZDot** should be interpreted as follows. By a simple inductive argument, Z^x for every vector x in the program is defined *uniquely* as a deterministic function $\varphi(\hat{Z}^{x^1}, \ldots, \hat{Z}^{x^k})$ of some $x^1, \ldots, x^k$ in $\mathcal{V}$ or introduced by **MatMul** (notationally, we are suppressing the possible dependence on limit scalars $\mathring{\theta}_1, \ldots, \mathring{\theta}_l$). For instance, if in a program we have $A \in \mathcal{W}, v \in \mathcal{V}$, $y = Av, x = A^\top y$, then $Z^x = \hat{Z}^x + \hat{Z}^v$, so φ is given by $\varphi(a, b) = a + b$. Then

$$\partial Z^x / \partial \hat{Z}^{x^i} \stackrel{\text{def}}{=} \partial_i \varphi(\hat{Z}^{x^1}, \ldots, \hat{Z}^{x^k})$$
$$\partial Z^x / \partial \hat{Z}^z \stackrel{\text{def}}{=} 0 \text{ for any } z \notin \{x^1, \ldots, x^k\}.$$

Note this definition depends on the precise way the program is written, not just on the underlying mathematics. For example, if $y, z \in \mathcal{V}$ and $x = \phi(W(y + z))$, then $Z^x = \phi(\hat{Z}^{W(y+z)})$ so that $\partial Z^x / \partial \hat{Z}^{Wy} = \partial Z^x / \partial \hat{Z}^{Wz} = 0$. If instead, we have $x = \phi(Wy + Wz)$, then $Z^x = \phi(\hat{Z}^{Wy} + \hat{Z}^{Wz})$ so that $\partial Z^x / \partial \hat{Z}^{W(x+y)} = 0$. However, in both cases, $\dot{Z}^{W^\top x} = (Z^y + Z^z) \mathbb{E}\, \phi'(\hat{Z}^{W(y+z)})$.

Remark L.2 (Partial derivative expectation)**.** The quantity $\mathbb{E} \frac{\partial Z^x}{\partial \hat{Z}^{W^\top y}}$ is well defined if Z^x is differentiable in $\hat{Z}^{W^\top y}$. However, even if this is not the case, e.g. if $x = \theta(W^\top y)$ where θ is the Heavyside step function, we can still define this expectation by leveraging Stein's lemma:

In **ZDot**, suppose $\{W^\top y^i\}_{i=1}^k$ are all elements of $\mathcal{V}_{W^\top}$ introduced before x. Define the matrix $C \in \mathbb{R}^{k\times k}$ by $C_{ij} \stackrel{\text{def}}{=} \mathbb{E}\, Z^{y^i} Z^{y^j}$ and define the vector $b \in \mathbb{R}^k$ by $b_i \stackrel{\text{def}}{=} \mathbb{E}\, \hat{Z}^{W^\top y^i} Z^x$. If $a = C^+ b$ (where C^+ denotes the pseudoinverse of C), then in **ZDot** we may set

$$\sigma_W^2 \,\mathbb{E} \frac{\partial Z^x}{\partial \hat{Z}^{W^\top y^i}} = a_i. \tag{40}$$

This definition agrees with the partial derivative expectation by Stein's lemma when the latter is well defined. Theorem G.4 holds with this broader definition of partial derivative expectation.

Pseudo-Lipschitz functions are, roughly speaking, functions whose weak derivatives are polynomially bounded.

Definition L.3. A function $f : \mathbb{R}^k \to \mathbb{R}$ is called *pseudo-Lipschitz* of degree d if $|f(x) - f(y)| \le C\|x - y\|(1 + \sum_{i=1}^k |x_i|^d + |y_i|^d)$ for some C. We say f is pseudo-Lipschitz if it is so for any degree.

Here are some basic properties of pseudo-Lipschitz functions:

- The norm $\|\cdot\|$ in Definition L.3 can be any norm equivalent to the ℓ_2 norm, e.g. $\ell_p, p \ge 1$, norms. Similarly, $\sum_{i=1}^k |x_i|^d + |y_i|^d$ can be replaced by $\|x\|_p^d + \|y\|_p^d$, for any $p \ge 1$.
- A pseudo-Lipschitz function is polynomially bounded.
- A composition of pseudo-Lipschitz functions of degrees d_1 and d_2 is pseudo-Lipschitz of degree $d_1 + d_2$.
- A pseudo-Lipschitz function is Lipschitz on any compact set.

We adopt the following assumption for the Master Theorem Theorem G.4.

Assumption L.4. *Suppose*

1. *If a function $\phi(;-) : \mathbb{R}^{0+l} \to \mathbb{R}$ with only parameter arguments is used in **Moment**, then ϕ is continuous in those arguments.*
2. *Any other function $\phi(-;-) : \mathbb{R}^{k+l} \to \mathbb{R}$ with parameters (where $k > 0$) used in **Nonlin** or **Moment** is pseudo-Lipschitz in all of its arguments (both inputs and parameters).*

Statement 1 in Assumption L.4 essentially says that if we have scalars $\theta_1, \ldots, \theta_l$ in the program, then we can produce a new scalar by applying a continuous function (a weaker restriction than a pseudo-Lipschitz function) to them. Indeed, if $\theta_1, \ldots, \theta_l$ converge almost surely, then this new scalar does too. In our setting, statement 1 is used to allow any loss function whose derivative is continuous.

Other versions of the Master Theorem can be found in (Yang, 2020b), for example, versions where the we do not assume any smoothness condition at all on the nonlinearities beyond that they be polynomially bounded, in exchange for assuming what's called a *rank stability* condition. This rank stability should be generically true, but checking it rigorously is subtle, so we are content with the pseudo-Lipschitz condition in this paper.

M. A Rough Sketch of the Geometry of abc-Parametrizations

By the results of Section 3, the stable abc-parametrizations form a polyhedron defined by the inequalities of Theorem 3.2. We call the polyhedron obtained by quotienting Eq. (13) the *stable polyhedron*. In this section, we remark on some geometric properties of this polyhedron.

First, observe that the stable polyhedron is unbounded (thus, we say *polyhedron* instead of *polytope*). Indeed, given any stable parametrization, for any l, we can set $a_l \leftarrow a_l + \theta, b_l \leftarrow b_l - \theta$ for any $\theta \geq 0$ to obtain another stable parametrization. This corresponds decreasing the layer l learning rate, so that as $\theta \to \infty$, W^l is not trained.

Second, by Theorem 3.3, the nontrivial parametrizations reside in two facets of the stable polyhedron. These facets are unbounded for the same reason as above.

Next, we show that NTP (as well as μP) is a vertex on the intersection of these two facets, and NTP and μP are connected by an edge.

Definition M.1. Consider a stable abc-parametrization of the MLP in Eq. (1). We say the body of the MLP is *uniformly updated* if, for some training routine, time $t \geq 1$, and input ξ, $\Delta W_t^l x_t^l(\xi) = \Theta(n^{-r})$ for all l simultaneously, where r is as defined in Definition 3.1.

In the results of this section below, we assume Assumption N.21.

Proposition M.2. *In a stable abc-parametrization, the MLP body is uniformly updated iff $r_l = r$ for all $l \in [L]$, where r_l is as defined in Proposition E.2.*

Theorem M.3. *In NTP, the MLP body is updated uniformly and W^{L+1} is both initialized and updated maximally. Furthermore, at initialization, f_0 converges in distribution*[45] *to a Gaussian Process with nonzero kernel. NTP is the unique (modulo Eq. (13)) stable abc-parametrization with both of these properties.*

Theorem M.4. *For any $r \in [0, 1/2]$, there is a unique (modulo Eq. (13)) stable abc-parametrization with 1) that value of r and the property that 2) the MLP body is updated uniformly and W^{L+1} is both initialized and updated maximally. We call this parametrization the* Uniform Parametrization *with r-value r, denoted UP$_r$. Its abc values are*

$$a_l = -\frac{1}{2}\mathbb{I}(l = 1) + r \;\forall l \in [L],\; a_{L+1} = 1/2;$$
$$b_l = 1/2 - r; \quad c = 0.$$

In particular, UP$_0$ is μP and UP$_{1/2}$ is NTP. For $r > 1/2$, such a uniform parametrization is not stable because W_0 would need to be $\Theta(n^{r-1})$, which would cause the initial GP to blow up. Thus, geometrically, UP$_r, r \in [0, 1/2]$, form an edge of the stable polyhedron.

We can define the *uniform stable polyhedron* to be the subset of the stable polyhedron corresponding to parametrizations which update the MLP body uniformly. This is isomorphic to the stable polyhedron when $L = 1$. Since stable abc-parametrizations with $L = 1$ has only 3 degrees of freedom, say a_1, a_2, b_2 while we fix $c = 0$ (via Eq. (13)) and $b_1 = -a_1$, we can visualize the corresponding stable polyhedron in 3D. However, the nontrivial parametrizations only reside in the boundary of this polyhedron. Because of its unbounded nature, we can project its boundary in 2D and visualize it. This is done in Fig. 7.

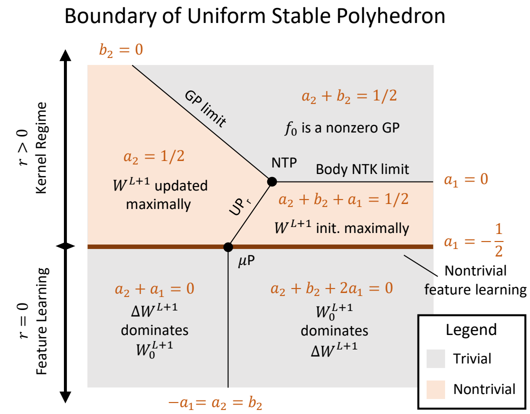

Figure 7. **2D Projection of the Boundary of the Uniform Stable Polyhedron (Equivalently, the Boundary of the Stable Polyhedron for $L = 1$).** Here, we label each facet and edge of the graph with orange text to indicate the corresponding defining algebraic condition in the $L = 1$ case (as part of the stable polyhedron, assuming $c = 0$ and $b_1 = -a_1$), and with **black text** to indicate the verbal interpretation valid for all L (as part of the uniform stable polyhedron). We obtain the caricature in Fig. 2 by taking the *nontrivial* subspace of the graph here and quotienting the two facets by their respective points at infinity. *Explanation of some captions: GP limit* means the training dynamics amounts to training only the last layer in the infinite-width limit, starting from a nonzero initial GP. *Body NTK limit* means NTK dynamics except the last layer does not contribute to the NT kernel.

[45] as is conventional in the machine learning literature, the convergence in distribution we mean here is really over *finite dimensional marginals*, i.e. $(f_0(\xi_1), \ldots, f_0(\xi_k)) \xrightarrow{\mathrm{d}} (\mathring{f}_0(\xi_1), \ldots, \mathring{f}_0(\xi_k))$ where $\mathring{f}_0$ is the limit GP.

N. Proofs of Main Results

N.1. Rigorous Statements of Main Results

Applicable Nonlinearities For technical reasons, in our main results we restrict our attention to the canonical examples of nonlinearities: tanh and relu — or rather, a smooth version of relu called gelu (Hendrycks & Gimpel, 2020) common in transformer models (Brown et al., 2020a). More precisely,

Definition N.1. Define σ-gelu to be the function $x \mapsto \frac{1}{2}x\mathrm{erf}(\sigma^{-1}x) + \sigma\frac{e^{-\sigma^{-2}x^2}}{2\sqrt{\pi}} + \frac{x}{2}$.

σ-gelu is a smooth approximation of relu and is the integral of $\frac{1}{2}(\mathrm{erf}(\sigma^{-1}x) + 1)$ that is 0 at $-\infty$. The large σ is, the smoother σ-gelu is. As $\sigma \to 0$, σ-gelu converges to relu. We believe our results will hold for generic nonlinearities, but making this precise is outside our scope here. (See Remark N.14 for some discussion).

Notations and Terminologies

Definition N.2 (Big-O Notation)**.** Given a sequence of scalar random variables $c = \{c^n \in \mathbb{R}\}_{n=1}^{\infty}$, we write $c = \Theta(n^{-a})$ if there exist constants A, B such that $An^{-a} \le |c| \le Bn^{-a}$ for sufficiently large n, almost surely[46]. Given a sequence of random vectors $x = \{x^n \in \mathbb{R}^n\}_{n=1}^{\infty}$, we say *$x$ has coordinates of size* $\Theta(n^{-a})$ and write $x = \Theta(n^{-a})$ to mean the scalar random variable sequence $\{\sqrt{\|x^n\|^2/n}\}_n$ is $\Theta(n^{-a})$. Similarly for the notations $O(n^{-a}), \Omega(n^{-a})$. We use the notations $\Theta_\xi(n^{-a}), O_\xi(n^{-a}), \Omega_\xi(n^{-a})$ if the hidden constants A, B are allowed to depend on some object ξ. For brevity, we will often abuse notation and say c itself is a random variable or x itself is a random vector.

Most often, the vector x will have "approximately iid" coordinates, so the notation $x = \Theta(n^{-a})$ can be interpreted intuitively to say x has coordinates of "standard deviation" $\Theta(n^{-a})$, which justifies the name.

Definition N.3. An *abc-parametrization* is a joint parametrization of an MLP and the learning rate specified by the numbers $\{a_l, b_l\}_l \cup \{c\}$ as in Eq. (1). Below we will often say *abc-parametrization of an MLP* for short, even though the parametrization affects the learning rate as well. A *training routine* is a combination of learning rate ηn^{-c}, training sequence $\{(\xi_t, y_t)\}_{t\ge 0}$, and a loss function $\mathcal{L}(f(\xi), y)$ that is continuously differentiable in the prediction of the model $f(\xi)$.

[46]Here *almost surely* means *for almost every instantiation of* $c^1, c^2, \ldots$, i.e. it is with regard to the product probability space generated by all of $\{c^n\}_{n=1}^{\infty}$. In this paper, this probability space will be generated by random initializations of a neural network at every width n. Very importantly, note the order of the qualifiers: we are saying *for almost every instantiation of* $c^1, c^2, \ldots$, *for large enough* n, $An^{-a} \le |c| \le Bn^{-a}$.

Main Results We will mainly focus on *stable* parametrizations, defined below, which intuitively means 1) the preactivations $\{h^l\}_l$ and activations $\{x^l\}_l$ have $\Theta(1)$ coordinates at initialization, and 2) their coordinates and the logit $f(\xi)$ all stay $O(1)$ (i.e. bounded independent of n) throughout the course of SGD.[47] Otherwise, they tend to ∞ with n, eventually going out of floating point range. Indeed, this is an acute and real problem common in modern deep learning, where float16 is necessary to train large models.

Definition N.4 (Stability)**.** We say an abc-parametrization of an L-hidden layer MLP is *stable* if

1. For every nonzero input $\xi \in \mathcal{X}$,

$$h_0^l(\xi), x_0^l(\xi) = \Theta_\xi(1), \forall l \in [L], \quad \text{and} \qquad \mathbb{E}\, f_0(\xi)^2 = O_\xi(1), \tag{41}$$

   where the expectation is taken over the random initialization.

2. For any training routine, any time $t \ge 0$, $l \in [L]$, $\xi \in \mathcal{X}$, we have

$$\Delta h_t^l(\xi), \Delta x_t^l(\xi) = O_*(1), \forall l \in [L], \quad \text{and} \qquad f_t(\xi) = O_*(1),$$

   where the hidden constant inside O can depend on the training routine, t, ξ, and the initial function values $f_0(\mathcal{X})$.[48]

Recall from the main text,

Definition N.5. For any abc-parametrization, we write r for the quantity

$$r \stackrel{\text{def}}{=} \min(a_{L+1}+b_{L+1}, 2a_{L+1}+c)+c-1+\min_{l=1}^{L}\left[2a_l + \mathbb{I}(l=1)\right].$$

For example, in NTP, $r = 1/2$, while in μP, $r = 0$. Intuitively, r is the exponent such that $\Delta x_t^L(\xi) = \Theta_\xi(n^{-r})$. Thus, to avoid activation blowup, we want $r \ge 0$; to perform feature learning, we want $r = 0$.

Theorem N.6 (Stability Characterization)**.** *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. An abc-parametrization is stable iff all of the following are true (with intuitions in parentheses):*

1. *((pre)activations at initialization are $\Theta(1)$ and logits are $O(1)$)*

$$a_1 + b_1 = 0; \quad a_l + b_l = 1/2,\ \forall l \in [2, L]; \qquad a_{L+1} + b_{L+1} \ge 1/2. \tag{42}$$

[47]but they may depend on training time and η; in particular, it's possible that they diverge with time

[48]For e.g. the NTK limit, f_0 is a GP, so that we should expect the bounds on $\Delta h_t^l(\xi), \Delta x_t^l(\xi)$ to depend on f_0.

2. *(features don't blowup, i.e. $\Delta x_t^l = O(1)$ for all l)*

$$r \geq 0. \tag{43}$$

3. *(logits don't blow up during training, i.e. $\Delta W_t^{L+1} x_t^L, W_0^{L+1} \Delta x_t^L = O(1)$)*

$$2a_{L+1} + c \geq 1; \quad a_{L+1} + b_{L+1} + r \geq 1. \tag{44}$$

Here, r is as defined in Definition N.5.

In Eq. (44), ΔW_t^{L+1} turns out to be $\Theta(n^{-(2a_{L+1}+c)})$ and is correlated with $x_t^L = \Theta(1)$ such that their product behaves according to Law of Large Numbers; the first inequality says this should not blow up. Similarly, $W_0^{L+1} = \Theta(n^{-(a_{L+1}+b_{L+1})})$ and it turns out $\Delta x_t^L = \Theta(n^{-r})$ and they will interact via Law of Large Numbers, so the second inequality says their product shouldn't blow up.

Our main results concern *nontrivial* parametrizations:

Definition N.7 (Nontriviality)**.** We say an abc-parametrization of an L-hidden layer MLP is *trivial* if for every training routine, $f_t(\xi) - f_0(\xi) \xrightarrow{\text{a.s.}} 0$ for any time $t \geq 1$ and input $\xi \in \mathcal{X}$ (i.e. the function does not evolve in the infinite-width limit). We say the parametrization is *nontrivial* otherwise.

Theorem N.8 (Nontriviality Characterization)**.** *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. A stable abc-parametrization is nontrivial iff $a_{L+1} + b_{L+1} + r = 1$ or $2a_{L+1} + c = 1$.*

Definition N.9 (Feature Learning)**.** We say an abc-parametrization of an L-hidden layer MLP *admits feature learning in the lth layer* if there exists some training routine such that

$$\Delta x_t^l(\xi) = \Omega_*(1) \tag{45}$$

for some $t \geq 0, \xi \in \mathcal{X}$, where the hidden constant inside Ω can depend on the training routine, t, ξ, and the initial function values $f_0(\mathcal{X})$. We say the parametrization *admits feature learning* if it does so in any layer.

We say the parametrization *fixes the lth layer features* if for all training routine,

$$\|\Delta x_t^l(\xi)\|^2/n \xrightarrow{\text{a.s.}} 0$$

for all $t \geq 0, \xi \in \mathcal{X}$. We say the parametrization *fixes all features* if it does so in every layer.

We make similar definitions as above replacing *feature* with *prefeature* and x^l with h^l.

Note that the probabilistic nature of $\Omega_*(1)$ means that *no feature learning* does not imply *fixing all features* (because $\Delta x_t^l(\xi)$ can just fluctuate wildly between 0 and infinity), but we will see that in the context of nontrivial stable abc-parametrizations, this is true.

A somewhat stronger notion of feature learning is that the feature kernel evolves. This is, for example, essential for linear transfer learning such as in self-supervised learning of image data.

Definition N.10 (Feature Kernel Evolution)**.** We say an abc-parametrization of an L-hidden layer MLP *evolves the lth layer feature kernel* if there exists some training routine such that

$$x_t^l(\xi)^\top x_t^l(\zeta)/n - x_0^l(\xi)^\top x_0^l(\zeta)/n = \Omega_*(1)$$

for some $t \geq 0, \xi, \zeta \in \mathcal{X}$, where the hidden constant inside Ω can depend on the training routine, t, ξ, ζ, and the initial function values $f_0(\mathcal{X})$. We say the parametrization *evolves feature kernels* if it does so in any layer.

We say the parametrization *fixes the lth layer feature kernel* if for all training routine,

$$x_t^l(\xi)^\top x_t^l(\zeta)/n - x_0^l(\xi)^\top x_0^l(\zeta)/n \xrightarrow{\text{a.s.}} 0, \quad \text{as} \quad n \to \infty,$$

for all $t \geq 0, \xi, \zeta \in \mathcal{X}$. We say the parametrization *fixes all feature kernels* if it does so in every layer.

We make similar definitions as above replacing *feature* with *prefeature* and x^l with h^l.

Intuitively, for a stable parametrization, feature kernel evolution should imply feature learning (one can see the contrapositive easily). In fact, we shall see below they are equivalent notions.

On the other hand, from the NTK example, we know certain limits can be described entirely through kernel gradient descent with some kernel. Appropriately, we make the following definition.

Definition N.11 (Kernel Regime)**.** We say an abc-parametrization of an L-hidden layer MLP *is in kernel regime* if there exists a positive semidefinite kernel $K : \mathcal{X}^2 \to \mathbb{R}$ such that for every training routine, the MLP function evolves under kernel gradient descent, i.e. there exist random variables $\mathring{f}_t(\xi)$ for each time $t \geq 0$ and input $\xi \in \mathcal{X}$ such that, as $n \to \infty$,[49]

$$\{f_t(\xi)\}_{t \leq T, \xi \in \mathcal{X}} \xrightarrow{\text{d}} \{\mathring{f}_t(\xi)\}_{t \leq T, \xi \in \mathcal{X}}, \quad \forall T \geq 1,$$

where $\xrightarrow{\text{d}}$ denotes convergence in distribution, and

$$\mathring{f}_{t+1}(\xi) = \mathring{f}_t(\xi) - \eta K(\xi, \xi_t) \mathcal{L}'(\mathring{f}_t(\xi_t), y_t), \quad \forall t \geq 0. \tag{46}$$

[49] Here because we want to avoid topological issues arising for convergence in distribution of infinite sequences, we only require convergence in distribution jointly in all $\xi \in \mathcal{X}$ and time t below some cutoff T for every finite T.

Observe that, in kernel regime, $\mathring{f}_t(\xi)$ is deterministic conditioned on $\mathring{f}_0(\xi)$, as evident inductively from Eq. (46). For example, in the NTK limit, $\{\mathring{f}_0(\xi) : \xi \in \mathcal{X}\}$ is a nontrivial Gaussian Process (GP), but the function evolution conditioned on this GP is deterministic.

All of the concepts defined above are related to each other by the following theorem.

Theorem N.12 (Classification of abc-Parametrizations)**.** *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. Consider a nontrivial stable abc-parametrization of an L-hidden layer MLP. Then*

1. *The following are equivalent to $r = 0$*
   (a) *feature learning*
   (b) *feature learning in the Lth layer*
   (c) *feature kernels evolution*
   (d) *feature kernel evolution in the Lth layer*
   (e) *prefeature learning*
   (f) *prefeature learning in the Lth layer*
   (g) *prefeature kernels evolution*
   (h) *prefeature kernel evolution in the Lth layer*
2. *The following are equivalent to $r > 0$*
   (a) *kernel regime*
   (b) *fixes all features*
   (c) *fixes features in the Lth layer*
   (d) *fixes all feature kernels*
   (e) *fixes feature kernel in the Lth layer*
   (f) *fixes all prefeatures*
   (g) *fixes prefeatures in the Lth layer*
   (h) *fixes all prefeature kernels*
   (i) *fixes prefeature kernel in the Lth layer*
3. *If there is feature learning* or *feature kernel evolution* or *prefeature learning* or *prefeature kernel evolution in layer l, then there is feature learning* and *feature kernel evolution* and *prefeature learning* and *prefeature kernel evolution in layers $l, \ldots, L$.*
4. *If $r = 0$, then for all $\xi \in \mathcal{X}$, $f_0(\xi) \xrightarrow{\text{a.s.}} 0$ and $f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi)$ for some deterministic $\mathring{f}_t(\xi)$. However, the converse is not true.*
5. *If $r > 0$, $a_{L+1} + b_{L+1} + r > 1$ and $2a_{L+1} + c = 1$, then we have the Neural Network-Gaussian Process limit.*

In particular, Statement 4 implies that feature learning, at least in our context, is incompatible with Bayesian, distributional perspectives of neural network limits, such as the NNGP limit.

The characterization above then trivially implies the following dichotomy.

Corollary N.13 (Dynamical Dichotomy)**.** *For ϕ being tanh or σ-gelu for sufficiently small σ, a nontrivial stable parametrization of an L-hidden layer MLP either admits feature learning or is in kernel regime, but not both.*

Remark N.14 (The Role of the ϕ Assumption)**.** The dependence on ϕ being tanh or σ-gelu for sufficiently small σ is only needed to explicitly construct a training routine that leads to feature learning for $r = 0$. We expect this should be true for generic ϕ, but we leave this for future work. We expand more on the role of the ϕ assumption below.

To calculate the infinite width limit of any abc-parametrization rigorously, we only need the nonlinearity to have a polynomially bounded 2nd derivative (or more generally pseudo-Lipschitz, so as to apply the Master Theorem). The specific choice of tanh or gelu is needed to prove the part of the Dynamical Dichotomy that says a limit cannot be simultaneously in kernel regime and in feature learning regime (which, e.g. is not true for linear activation). To do so, we use Properties N.43 and N.46 of tanh and gelu, expanded below. This is really for a more convenient proof, but we believe a more general approach should work for general nonlinearities. Our argument is as follows (this is also overviewed in the start of Appendix N.7): If $r = 0$, we show that a sufficiently small nonzero learning rate (scaled with width in the corresponding parametrization) in 1 SGD step 1) induces a change in the features but 2) the resulting change in the NN output is not linear in the loss derivative χ. 1) means it's feature learning, and 2) means it's not in kernel regime. This argument involves showing certain derivatives of certain expectations with respect to learning rate is positive. In the case of tanh and gelu, this is checked explicitly using Properties N.43 and N.46.

Remark N.15. The equivalence between kernel regime and fixed feature kernel implies that linear transfer learning is trivialized in any kernel regime limit. This is where the classifier layer of the pretrained network is discarded and a new one (potentially outputting to a new output space) is trained on top of the body of the pretrained network. But we can in fact say more: any *nonlinear* transfer learning, where we replace the classifier layer with a neural network instead of a linear layer, is trivialized as well. In addition, linear or nonlinear transfer learning has no effect even if we finetune the entire network, instead of just the new classification network. The intuitive reason for this is that, as discussed in Appendix B, the effect of $\Delta x^L(\xi)$ on the output of the MLP is solely through the interaction with W_0^{L+1}. If $W^{L+1}, W^{L+2}, \ldots,$ are sampled anew, then this effect vanishes. We formalize this below.

Theorem N.16 (Kernel Regime Limit Trivializes Transfer

Learning). *Suppose f is an L-hidden-layer MLP*[50] *in a stable kernel regime parametrization. Let A and B be two training routines.*[51]

For any $T, t \geq 0$,[52] *we define a network*[53] *$g_{T;t}$ as follows. Train f on A for T steps to obtain f_T. Then discard W^{L+1} in f_T and extend the body of f_T into an M-hidden-layer MLP g, where $M \geq L$.*[54] *Parametrize and initialize the new weights of g according to any stable abc-parametrization that extends the parametrization of f. Train g on B for t steps to obtain $g_{T;t}$.*

Then

1. *(Finetuning the whole network) As $n \to \infty$, for any $\xi \in \mathcal{X}$ and $T, t \geq 0$,*
$$g_{T;t}(\xi) - g_{0;t}(\xi) \xrightarrow{\text{a.s.}} 0.$$
2. *(Training only the classifier) The above is true even if we define $g_{T;t}$ by only training the new weights $W^{L+1}, \ldots, W^M$ in g.*

The Organization for the Proof of Our Main Results Above

Definition N.17. Below, we will abbreviate *abc-parametrization of an L-layer MLP* to just *parametrization*. We will call parametrizations satisfying the conditions of Theorem N.6 *pseudostable* while we try to prove Theorem N.6 (which, in this terminology, says stability and pseudostability are equivalent).

We first characterize stability at initialization and prove Eq. (41) holds iff Eq. (42) (Appendix N.2). Then, we describe the Tensor Program encoding the SGD of an MLP, assuming its parametrization is pseudostable. The Master Theorem then naturally lets us calculate its infinite-width limit. We then divide into the case of $r > 0$ and $r = 0$. In the former case, we show the infinite-width limit is described by kernel gradient descent as in Eq. (46). In the latter case, we construct a training routine where feature learning occurs and where the limit is *not* given by kernel gradient descent for any kernel. Finally, in Appendix N.8, we combine all of our analyses to prove the main results in this section.

N.2. Stability at Initialization

In this section, we characterize stability at initialization, which will form a foundation for our later results.

[50] the "pretrained network"

[51] the "pretraining dataset" and the "finetuning dataset"

[52] the "pretraining time" and "finetuning time"

[53] the "finetuned network"

[54] If $M = L$, then this is linear transfer learning where we replace just the last layer of f; otherwise, it's nonlinear transfer learning.

Theorem N.18. *Assume ϕ is not zero almost everywhere. For any parametrization, Eq. (41) holds iff Eq. (42) holds, i.e. the following are equivalent*

1. For every nonzero input $\xi \in \mathcal{X}$,
$$h_0^l(\xi), x_0^l(\xi) = \Theta_\xi(1), \forall l \in [L], \quad \text{and}$$
$$\mathbb{E}\, f_0(\xi)^2 = O_\xi(1),$$
where the expectation is taken over the random initialization.
2. $a_1 + b_1 = 0$; $\quad a_l + b_l = 1/2$, $\forall l \in [2, L]$; $\quad a_{L+1} + b_{L+1} \geq 1/2$.

Proof. Fix an input $\xi \neq 0$. Here, because we focus on initialization, we will suppress the time 0 subscript and ξ dependence of h^l, x^l to mean $t = 0$, applied to ξ.

Obviously, $h^1 = W^1\xi$ is a Gaussian vector with $\mathcal{N}(0, n^{-(a_1+b_1)}\|\xi\|^2)$ coordinates, so $h^1 = \Theta_\xi(1)$ iff $a_1 + b_1 = 0$. Assume $a_1 + b_1 = 0$. By Law of Large Numbers, $\frac{1}{n}\|x^1\|^2 \xrightarrow{\text{a.s.}} \mathbb{E}\,\phi(Z^{h^1})^2$ where $Z^{h^1} = \mathcal{N}(0, \|\xi\|^2)$. Since ϕ is not almost everywhere zero and $\xi \neq 0$, this expectation is nonzero so that $x^1 = \Theta_\xi(1)$.

We construct the following Tensor Program: the lone initial vector is h^1, the initial matrices are $\widehat{W}^l, 2 \leq l \leq L$, and initial scalars $\theta_l \stackrel{\text{def}}{=} n^{1/2-(a_l+b_l)}$. We sample $h^1_\alpha \sim \mathcal{N}(0, \|\xi\|^2)$ and $\widehat{W}^l_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$. Mathematically, we will represent $W^l = \theta_l \widehat{W}^l$. The program is then given by
$$x^l = \phi(h^l), \forall l \in [L], \quad \hat{h}^l = \widehat{W}^l x^{l-1}, h^l = \theta_l \hat{h}^l, \forall l \in [2, L],$$
where we used Nonlin, MatMul, and Nonlin (with parameter θ_l).

Suppose $a_l + b_l = 1/2$ (i.e. $\theta_l = 1$) for all $2 \leq l \leq L$. Then, $Z^{h^l} = Z^{\hat{h}^l} = \mathcal{N}(0, \mathbb{E}\,\phi(Z^{h^{l-1}})^2)$ for each $l \leq L$. Because ϕ is not everywhere zero, this inductively implies $\mathbb{E}(Z^{h^l})^2 > 0$ (and so also $\mathbb{E}(Z^{x^l})^2 > 0$) for all $l \leq L$. By the Master Theorem, $\frac{1}{n}\|h^l\|^2 \xrightarrow{\text{a.s.}} \mathbb{E}(Z^{h^l})^2$ and $\frac{1}{n}\|x^l\|^2 \xrightarrow{\text{a.s.}} \mathbb{E}(Z^{x^l})^2$ so this implies $h^l, x^l = \Theta_\xi(1)$ for all $l \leq L$ as desired.

Conversely, suppose m is the smallest $l \geq 2$ such that $a_l + b_l \neq 1/2$. Then by the above reasoning, $\hat{h}^m = \Theta_\xi(1)$ so $h^m = \Theta_\xi(n^{1/2-(a_l+b_l)})$ is either blowing up to ∞ or shrinking to 0 with n. This shows that $h^l, x^l = \Theta_\xi(1)$ for all $l \leq L$ iff $a_1 + b_1 = 0$ and $a_l + b_l = 1/2$ for all $2 \leq l \leq L$.

Finally, if $a_1 + b_1 = 0$ and $a_l + b_l = 1/2$ for all $2 \leq l \leq L$, then we see $\mathbb{E}\, f_0(\xi)^2 = (n^{1/2-(a_{L+1}+b_{L+1})})^2\, \mathbb{E}\,\|Z^{x^L}\|^2/n$. For large n, this is $\Theta_\xi((n^{1/2-(a_{L+1}+b_{L+1})})^2)$ and is $O_\xi(1)$ iff $a_{L+1} + b_{L+1} \geq 1/2$. $\square$

Definition N.19. We say a parametrization is *initialization-stable* if it satisfies Eq. (41) (or equivalently, Eq. (42)).

N.3. Program Setup

In the next section, we construct the Tensor Program that encodes the training of an L-hidden layer MLP under an abc-parametrization. Here we first describe the initial matrices, vectors, and scalars of the program, along with necessary notations.

We first remark on a simplification we will make to streamline the proof.

The Size of W_0^{L+1} vs ΔW_t^{L+1} By construction, $W_0^{L+1} = \Theta(n^{-(a_{L+1}+b_{L+1})})$. If $x_t^L(\xi) = \Theta(1)$ as in a stable parametrization, then $\Delta W_t^{L+1} = \Theta(n^{-(2a_{L+1}+c)})$. Therefore, if $a_{L+1} + b_{L+1} \leq 2a_{L+1} + c$, then W_0^{L+1} is at least as large as ΔW_t^{L+1}, so that W_t^{L+1} will stay the same order (in terms of n) for all t. If the reverse inequality is true, then W_0^{L+1} is smaller than $W_t^{L+1}$for $t \geq 1$. This in particular implies that the gradients at time 0 is smaller than gradients at subsequent times. For example, we can take $a_{L+1} + b_{L+1} \to \infty$ while fixing $2a_{L+1} + c$, in which case $W_0^{L+1} = 0$ and the weight gradients at initialization are all 0 except for that of W^{L+1}. One can thus think of this as a "lag" in the training dynamics for 1 step.

Assumption N.20. *For clarity of the proof, we will assume $a_{L+1} + b_{L+1} \leq 2a_{L+1} + c$, i.e. W_t^{L+1} stays the same order for all t. The case of $a_{L+1} + b_{L+1} > 2a_{L+1} + c$, corresponding to a 1-step "lag" as explained above, can be dealt with similarly. We will remark whenever this requires some subtlety.*

For the construction of the program and the application of the Master Theorem, we will also assume the following for the rest of this paper.

Assumption N.21. *ϕ' is pseudo-Lipschitz and not almost everywhere zero.*

Initial Matrices, Vectors, Scalars We will assume the parametrization is initialization-stable. For ease of presentation, we also assume the input dimension $d = 1$.

1. Initial matrices: $W_0^2, \ldots, W_0^L$, sampled like $(W_0^l)_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$.

2. Initial vectors: input layer matrix $W_0^1 \in \mathbb{R}^{n\times 1}$ and *normalized* output layer matrix $\widehat{W}_0^{L+1} \stackrel{\text{def}}{=} W_0^{L+1} n^{a_{L+1}+b_{L+1}} \in \mathbb{R}^{1\times n}$, sampled like $(W_0^1)_\alpha, (\widehat{W}_0^{L+1})_\alpha \sim \mathcal{N}(0, 1)$.

3. Initial scalars: We define the following scalars (where we explain the intuition in parenthesis). The reader can skip this part on a first read but come back when referred to.

   (a) (n times the scale of coordinates of ΔW_t^l) For $l \geq 2$, define
   $$\theta_{W^l} \stackrel{\text{def}}{=} n^{-(a_{L+1}+b_{L+1}+c-1+2a_l)}$$

   (b) (scale of coordinates of ΔW_t^1 and Δh_t^1) Define
   $$\theta_1 = \theta_{W^1} \stackrel{\text{def}}{=} n^{-(a_{L+1}+b_{L+1}+c+2a_1)}$$

   (c) (scale of coordinates of ΔW_t^{L+1})
   $$\theta_{L+1} = \theta_{W^{L+1}} \stackrel{\text{def}}{=} n^{-2a_{L+1}-c}$$

   (d) (scale of Δh_t^l and Δx_t^l) For $l \in [L]$, define
   $$\theta_{h^l} = \theta_{x^l} = \theta_l \stackrel{\text{def}}{=} \max_{m\leq l} \theta_{W^m} = \max(\theta_{W^l}, \theta_{l-1}) \tag{47}$$
   $$= n^{-(a_{L+1}+b_{L+1}+c-1+\min_{m=1}^{l}(2a_m+\mathbb{I}(m=1)))}$$

   Note that $\theta_L = n^{-r}$ with r defined in Definition N.5.

   (e) (scale of W_t^{L+1})
   $$\theta_f \stackrel{\text{def}}{=} n^{-(a_{L+1}+b_{L+1})}$$

   (f) (convenience scalars)
   $$\begin{aligned}
   \theta_{x^{l-1}/h^l} &= \theta_{x^{l-1}}/\theta_{h^l} \\
   \theta_{W^l/h^l} &= \theta_{W^l}/\theta_{h^l} \\
   \theta_{W^l x^{l-1}/h^l} &= \theta_{W^l}\theta_{x^{l-1}}/\theta_{h^l} \\
   \theta_{L+1/f} &= \theta_{L+1}/\theta_f \\
   \theta'_{L+1} &= n\theta_{L+1} = n^{1-2a_{L+1}-c} \\
   \theta'_{Lf} &= n\theta_L\theta_f = n^{1-(r+a_{L+1}+b_{L+1})}
   \end{aligned}$$

   (g) Depending on the value of $a_{L+1} + b_{L+1}$, we will also construct the values of f at initialization as initial scalars. See Appendix N.4.1 for an explanation.

By our assumption that $a_{L+1} + b_{L+1} \leq 2a_{L+1} + c$, the pseudostability inequalities of Theorem N.6 imply all of these θs either converge to 0 or stay constant at 1. This means that, assuming appropriate regularity conditions on the nonlinearities and rank stability, we can apply the Master Theorem (if θ blows up to ∞ then we can't do that).

Notations We use $:=$ to more clearly denote assignment happening in the program, as opposed to mathematical equality. To clearly demonstrate the application of Nonlin, we will also freely introduce function symbols Ψ to put things into Nonlin form.

Preview of Names for Vectors In the program, for each $z \in \{x^l, h^l\}_l$, we will construct vectors $\delta z_t(\xi)$ to mathematically represent $\theta_z^{-1}(z_t(\xi) - z_{t-1}(\xi))$ (intuition: change in z scaled to have $\Theta(1)$ coordinates). Similarly, for $w \in \{W^{L+1}, W^1\}$, we will construct δw_t to mathematically represent $\theta_w^{-1}(w_t - w_{t-1})$ (intuition: change in w scaled to have $\Theta(1)$ coordinates). Then, mathematically, $z_t(\xi) = z_{t-1}(\xi) + \theta_z \delta z_t(\xi), w_t = w_{t-1} + \theta_w \delta w_t$.

We will also construct dz to mathematically represent $\theta_f^{-1}\nabla_z f$ (intuition: gradient $\nabla_z f$ scaled to have $\Theta(1)$ coordinates). For weight changes, we have the following identity

$$\begin{aligned} W_t^l - W_{t-1}^l &= -\eta n^{-c}\chi_{t-1} n^{-2a_l}\theta_f dh_{t-1}^l x_{t-1}^{l-1\top} \\ &= -\eta\chi_{t-1}\theta_{W^l}\frac{1}{n}h_{t-1}^l x_{t-1}^{l-1\top}, \quad \forall l \in [2, L], \end{aligned} \tag{48}$$

and for $l = 1$,

$$\begin{aligned} W_t^l - W_{t-1}^l &= -\eta n^{-c}\chi_{t-1} n^{-2a_l}\theta_f dh_{t-1}^l \xi_{t-1}^\top \\ &= -\eta\chi_{t-1}\theta_{W^l} h_{t-1}^l \xi_{t-1}^\top. \end{aligned} \tag{49}$$

N.4. Program Construction

Here we construct the Tensor Program encoding the SGD of an MLP. We separately describe the first forward and backward passes followed by the later forward and backward passes.

N.4.1. First Forward Pass

For every $\xi \in \mathcal{X}$, we compute $h_0^1(\xi) := W_0^1\xi \in \mathbb{R}^n$ via Nonlin (as $\Psi(W_0^1;\xi)$, where Ψ is multiplication by ξ), and we construct the following vectors via Nonlin and MatMul

$$x_0^l(\xi) := \phi(h_0^l(\xi)) \in \mathbb{R}^n, \quad h_0^{l+1}(\xi) := W_0^{l+1}x_0^l(\xi) \in \mathbb{R}^n, \\ \text{for } l = 1, \dots, L-1, \tag{50}$$

Function Output The first output is $f_0(\xi) = W_0^{L+1}x_0^L(\xi)$, but we will define $f_0(\xi)$ in the program slightly differently.

Case when $a_{L+1} + b_{L+1} > 1/2$ Then $f_0(\xi) \xrightarrow{\text{a.s.}} 0$ for all $\xi \in \mathcal{X}$. In the program, we will construct $f_0(\xi)$ as an *initial scalar* mathematically defined by $W_0^{L+1}x_0^L(\xi)$.[55][56]

[55] It is completely OK to define an initial scalar using randomness from other parts of the program, as long as this scalar converges almost surely to a deterministic limit

[56] We cannot define it using a `Moment` instruction because, intuitively, the mechanism of this convergence is through CLT, not Law of Large Numbers.

Case when $a_{L+1} + b_{L+1} = 1/2$ If $a_{L+1} + b_{L+1} = 1/2$, then $f_0(\xi)$ converges to a nontrival Gaussian via CLT (Yang, 2019a), so we will condition on $f_0(\xi)$ for all $\xi \in \mathcal{X}$. Given values $g(\xi) \in \mathbb{R}$ for all $\xi \in \mathcal{X}$, let $\mathcal{E}$ be the event that $f_0(\xi) = \frac{1}{\sqrt{n}}\widehat{W}_0^{L+1}x_0^L(\xi)$ equals $g(\xi)$ for all $\xi \in \mathcal{X}$. The distribution of $\widehat{W}_0^{L+1}$ conditioned on $\mathcal{E}$ is given by

$$\widehat{W}_0^{L+1} \overset{\text{d}}{=}_{\mathcal{E}} \sqrt{n}X^+g + \Pi\widetilde{W}_0^{L+1}$$

where $\widetilde{W}_0^{L+1}$ is an iid copy of $\widehat{W}_0^{L+1}$, $g \in \mathbb{R}^{\mathcal{X}}$ is the vector of $\{g(\xi) : \xi \in \mathcal{X}\}$, $X \in \mathbb{R}^{\mathcal{X}\times n}$ has $x_0^L(\xi)$ as rows, and Π is the orthogonal projection into the orthogonal complement of the space spanned by $\{x_0^L(\xi) : \xi \in \mathcal{X}\}$. Here X^+ denotes the pseudo-inverse of X.

By standard formulas for pseudo-inverse and orthogonal projection, we can write $X^+ = \frac{1}{n}X^\top(XX^\top/n)^+, \Pi = I - \frac{1}{n}X^\top(XX^\top/n)^+X$.

Let $\Sigma \overset{\text{def}}{=} XX^\top/n$ and $\gamma \overset{\text{def}}{=} (X\widetilde{W}_0^{L+1}/n)$. Then $\Pi\widetilde{W}_0^{L+1} = \widetilde{W}_0^{L+1} - X^\top\Sigma^+\gamma$, and $\sqrt{n}X^+g = \frac{1}{\sqrt{n}}X^\top\Sigma^+g$.

By the Master Theorem, $\gamma \xrightarrow{\text{a.s.}} 0$ because $\widetilde{W}_0^{L+1}$ is independent from X, and $\Sigma \xrightarrow{\text{a.s.}} \mathring{\Sigma}$ for some PSD matrix $\mathring{\Sigma}$. At this point in the program, all scalars we used (like ξ) are constant with n and can be absorbed into nonlinearities. By the rank stability property of any program without scalars (Yang, 2020b), the rank of Σ is fixed for large enough n, almost surely, so $\Sigma^+ \xrightarrow{\text{a.s.}} \mathring{\Sigma}^+$ by the continuity of pseudo-inverse on fixed rank matrices.

We will now replace $\widehat{W}_0^{L+1}$ in the program with

$$\widehat{W}_{\mathcal{E}}^{L+1} \overset{\text{def}}{=} X^\top\left(\Sigma^+\frac{g}{\sqrt{n}}\right) + \widetilde{W}_0^{L+1} - X^\top\left(\Sigma^+\gamma\right)$$

constructed using Nonlin, where $\left(\Sigma^+\frac{g}{\sqrt{n}}\right)$ and $(\Sigma^+\gamma)$ are finite dimensional and formally considered (collections of) scalars involved as coefficients for linear combination of rows of X. Since $\Sigma^+\frac{g}{\sqrt{n}}, \Sigma^+\gamma \xrightarrow{\text{a.s.}} 0$, we have $Z^{\widehat{W}_{\mathcal{E}}^{L+1}} = Z^{\widetilde{W}_0^{L+1}}$. Intuitively, this means that, even after conditioning on $f_0 = g$, the conditional distribution of $\widehat{W}_0^{L+1}$ is practically the same as the original distribution. We can then proceed exactly as in the case when $a_{L+1} + b_{L+1} > 1/2$, with $\widehat{W}_{\mathcal{E}}^{L+1}$ taking the role of $\widehat{W}_0^{L+1}$. The program then encodes the evolution of f *conditioned on* $f_0(\xi) = g(\xi), \forall\xi \in \mathcal{X}$.[57]

Assumption N.22. *For the above reason, we will assume* $a_{L+1} + b_{L+1} > 1/2$*, and remark whenever the case* $a_{L+1} + b_{L+1} = 1/2$ *involves subtleties.*

[57] Formally, we can also have $\{g(\xi) : \xi \in \mathcal{X}\}$ as initial scalars, but since they are fixed with n, they can be absorbed into the Nonlin that defines $\widehat{W}_{\mathcal{E}}^{L+1}$.

N.4.2. First Backward Pass

Next, we write the backward pass

$$
\begin{aligned}
dx_0^L(\xi) &:= \widehat{W}_0^{L+1} \\
dh_0^l(\xi) &:= dx_0^l(\xi) \odot \phi'(h_0^l(\xi)) \\
dx_0^{l-1}(\xi) &:= W_0^{l\top} dh_0^l(\xi)
\end{aligned}
$$

where, recall, dz mathematically equals $\theta_f^{-1}\nabla_z f$.

For $\xi = \xi_0$ and its label y_0, we define the first loss derivative as

$$\chi_0 := \mathcal{L}'(f_0(\xi_0), y_0) \xrightarrow{\text{a.s.}} \mathring{\chi}_0(\xi) = \mathcal{L}'(0, y_0)$$

where the convergence is because $\mathcal{L}'$ is continuous by assumption.

We also define

$$\delta W_1^{L+1} := -\eta\chi_0 x_0^L(\xi_0)$$

to represent the (normalized) change in W^{L+1} due to the first gradient step.

N.4.3. tth Forward Pass, $t \geq 1$

Overview We iteratively define $\delta z_t(\xi)$ to mathematically represent $\theta_z^{-1}(z_t(\xi) - z_{t-1}(\xi))$, for $z \in \{x^l, h^l\}_l$. Then we eventually set

$$z_t(\xi) := z_0(\xi) + \theta_z \delta z_1(\xi) + \cdots + \theta_z \delta z_t(\xi).$$

Likewise, we will define δW_t^{L+1} so that $W_t^{L+1} = \theta_f \widehat{W}_0^{L+1} + \theta_{L+1}(\delta W_1^{L+1} + \cdots + \delta W_t^{L+1})$. In the program, we will not directly use W_t^{L+1} but instead use

$$\widehat{W}_t^{L+1} := \widehat{W}_0^{L+1} + \theta_{L+1/f}(\delta W_1^{L+1} + \cdots + \delta W_t^{L+1}) \tag{51}$$

where $\theta_{L+1/f} = \theta_{L+1}/\theta_f$. Mathematically, $\widehat{W}_t^{L+1} = \theta_f^{-1} W_t^{L+1}$.

Recall we shorthand $z_t = z_t(\xi_t)$ for all $z \in \{x^l, h^l, dx^l, dh^l\}_l \cup \{f, \chi\}$.

The Construction of (Pre)Activations We start with $h = h^1$: By Eq. (49), we have

$$\delta h_t(\xi) := -\eta\chi_{t-1}\xi_{t-1}^\top \xi dh_{t-1} = \Psi(dh_{t-1}; \xi_{t-1}^\top \xi, \eta\chi_{t-1}).$$

(Notationally, recall we freely introduce function symbols Ψ to clarify the way we apply Nonlin). For higher layers, if $h = h^l$, $x = x^{l-1}$, and $W = W^l$, then $h = Wx$. By Eq. (48), we have, mathematically,

$$
\begin{aligned}
&\theta_h \delta h_t(\xi) \\
&= \theta_x W_{t-1}\delta x_t(\xi) + (W_t - W_{t-1})x_t(\xi) \\
&= \theta_x \left(W_0 \delta x_t(\xi) + \sum_{s=1}^{t-1} (W_s - W_{s-1})\delta x_t(\xi) \right) \\
&\quad + (W_t - W_{t-1})x_t(\xi) \\
&= \theta_x \left(W_0 \delta x_t(\xi) - \eta\theta_W \sum_{s=1}^{t-1} \chi_{s-1} \frac{x_{s-1}^\top \delta x_t(\xi)}{n} dh_{s-1} \right) \\
&\quad - \eta\chi_{t-1}\theta_W \frac{x_{t-1}^\top x_t(\xi)}{n} dh_{t-1}
\end{aligned}
$$

Recall $\theta_{x/h} = \theta_h^{-1}\theta_x, \theta_{W/h} = \theta_h^{-1}\theta_W, \theta_{Wx/h} = \theta_h^{-1}\theta_W\theta_x$. With c_s denoting $\frac{x_s^\top \delta x_t(\xi)}{n}$, we construct

$$
\begin{aligned}
\delta h_t(\xi) &:= \theta_{x/h} W_0 \delta x_t(\xi) - \eta\theta_{Wx/h} \sum_{s=1}^{t-1} \chi_{s-1} c_{s-1} dh_{s-1} \\
&\quad - \eta\chi_{t-1}\theta_{W/h} c_{t-1} dh_{t-1} \\
&= \Psi(W_0\delta x_t(\xi), dh_0, \ldots, dh_{t-1}; \\
&\qquad \eta, \theta_{x/h}, \theta_{Wx/h}, \theta_{W/h}, \{c_s, \chi_s\}_{s=0}^{t-1})
\end{aligned}
$$

If $x = x^l$, $h = h^l$, then $x = \phi(h)$, and (using $\theta_x = \theta_h$ (Eq. (47))),

$$
\begin{aligned}
\delta x_t(\xi) &:= \theta_h^{-1}(\phi(h_{t-1}(\xi) + \theta_h \delta h_t(\xi)) - \phi(h_{t-1}(\xi))) \\
&= \Psi(h_{t-1}(\xi), \delta h_t(\xi); \theta_h)
\end{aligned} \tag{52}
$$

where Ψ is precisely the difference quotient for the function ϕ.[58]

The Function Outputs We do not construct $f_t(\xi)$ directly, but rather through scalars $\delta f_t(\xi) = f_t(\xi) - f_{t-1}(\xi)$, so that

$$f_t(\xi) := f_0(\xi) + \delta f_1(\xi) + \cdots + \delta f_t(\xi).$$

Mathematically, $\delta f_t(\xi) = \theta_{L+1}\delta W_t^{L+1} x_t^L(\xi) + W_{t-1}^{L+1}\theta_L \delta x_t^L(\xi)$, but we shall write it slightly differently in the program:

$$\delta f_t(\xi) := \theta'_{L+1} \frac{\delta W_t^{L+1} x_t^L(\xi)}{n} + \theta'_{Lf} \frac{\widehat{W}_{t-1}^{L+1} \delta x_t^L(\xi)}{n}$$

where $\theta'_{L+1} = n\theta_{L+1}, \theta'_{Lf} = n\theta_L\theta_f$ and $\widehat{W}_{t-1}^{L+1}$ is constructed in Eq. (51).

[58] The pseudo-Lipschitzness of ϕ' assumed in Assumption N.21 implies that Ψ here is pseudo-Lipschitz, so that we can ultimately apply our Master Theorem.

N.4.4. tTH BACKWARD PASS, $t \geq 1$

In the last layer, we construct

$$dx_t^L(\xi) := \widehat{W}_t^{L+1}.$$

For each $l = L, \ldots, 1$ for dh^l and $l = L, \ldots, 2$ for dx^{l-1}, we also calculate

$$dh_t^l(\xi) := dx_t^l(\xi) \odot \phi'(h_t^l(\xi))$$
$$dx_t^{l-1}(\xi) := W_0^{l\top} dh_t^l(\xi) - \eta\theta_{W^l} \sum_{s=0}^{t-1} \chi_s c_s x_s^{l-1}$$
$$= \Psi(W_0^{l\top} dh_t^l(\xi), x_0^{l-1}, \ldots, x_{t-1}^{l-1}; \eta\theta_{W^l}, \{\chi_s, c_s\}_{s=0}^{t-1})$$

where $c_s = \frac{dh_s^{l\top} dh_t^l(\xi)}{n}$. For $\xi = \xi_t$ and its label y_t, we define[59]

$$\chi_t := \mathcal{L}'(f_t(\xi_t), y_t).$$

Finally, we compute the (normalized) change in W^{L+1} after this SGD update.

$$\delta W_{t+1}^{L+1} := -\eta\chi_t x_t^L(\xi_t).$$

N.5. The Infinite-Width Limit

In this section, we describe the Z random variables (Definition G.3) corresponding to the vectors of the program constructed above. According to the Master Theorem, each such vector z will have roughly iid coordinates distributed like Z^z in the large n limit.

Let $\mathring{\theta}_\bullet$ denote the limit of any $\theta_\bullet$ in Appendix N.3. If pseudostability holds, then $\mathring{\theta}_\bullet$ is either 0 or 1, as one can easily verify. We can construct the Z random variables for each vector in the program, as follows.

1. For the first forward and backward passes, we have,

$$Z^{h_0^1(\xi)} = \xi Z^{W_0^1}, \qquad Z^{h_0^{l+1}(\xi)} = Z^{W_0^{l+1} x_0^l(\xi)},$$
$$Z^{x_0^l(\xi)} = \phi(Z^{h_0^l(\xi)}), \qquad Z^{dh_0^l(\xi)} = Z^{dx_0^l(\xi)}\phi'(Z^{h_0^l(\xi)}),$$
$$Z^{dx_0^L(\xi)} = Z^{\widehat{W}_0^{L+1}}, \qquad Z^{dx_0^{l-1}(\xi)} = Z^{W_0^{l\top} dh_0^l(\xi)}$$

2. For $z \in \{x^l, h^l\}_l$, we have

$$Z^{z_t(\xi)} = Z^{z_0(\xi)} + \mathring{\theta}_z Z^{\delta z_1(\xi)} + \cdots + \mathring{\theta}_z Z^{\delta z_t(\xi)} \tag{53}$$

3. For $l \in [L], x = x^l, h = h^l$, we have $Z^{\delta x_t(\xi)} = \Psi(Z^{h_{t-1}(\xi)}, Z^{\delta h_t(\xi)}; \mathring{\theta}_h)$ where Ψ is as in Eq. (52). If $\mathring{\theta}_h = 0$ (e.g. if $r > 0$), then

$$Z^{\delta x_t(\xi)} = \phi'(Z^{h_{t-1}(\xi)}) Z^{\delta h_t(\xi)}. \tag{54}$$

Otherwise, $\mathring{\theta}_h = 1$, and

$$Z^{\delta x_t(\xi)} = \phi(Z^{h_t(\xi)}) - \phi(Z^{h_{t-1}(\xi)}). \tag{55}$$

4. For $h = h^1$, we have

$$Z^{\delta h_t(\xi)} = -\eta \mathring{\chi}_{t-1} \xi_{t-1}^\top \xi Z^{dh_{t-1}}.$$

5. For $l \geq 2, h = h^l, x = x^{l-1}, W = W^l$, we have

$$\begin{aligned} Z^{\delta h_t(\xi)} = \mathring{\theta}_{x/h} Z^{W_0 \delta x_t(\xi)} \\ - \eta \mathring{\theta}_{Wx/h} \sum_{s=0}^{t-2} \mathring{\chi}_s Z^{dh_s} \mathbb{E}\, Z^{x_s} Z^{x_t(\xi)} \\ - \eta \mathring{\chi}_{t-1} \mathring{\theta}_{W/h} Z^{dh_{t-1}} \mathbb{E}\, Z^{x_{t-1}} Z^{x_t(\xi)} \end{aligned} \tag{56}$$

where at least one of $\mathring{\theta}_{x/h}$ and $\mathring{\theta}_{W/h}$ equals 1. As usual, here we have the **ZHat-ZDot** decomposition of $Z^{W_0 \delta x_t(\xi)}$.

$$\begin{aligned} Z^{W_0 \delta x_t(\xi)} &= \hat{Z}^{W_0 \delta x_t(\xi)} + \dot{Z}^{W_0 \delta x_t(\xi)} \\ &= \hat{Z}^{W_0 \delta x_t(\xi)} + \sum_{s=0}^{t-1} Z^{dh_s} \mathbb{E} \frac{\partial Z^{\delta x_t(\xi)}}{\partial \hat{Z}^{W_0^\top dh_s}}. \end{aligned}$$

6. For last layer weight

$$Z^{\delta W_t^{L+1}} = -\eta \mathring{\chi}_{t-1} Z^{x_{t-1}^L} \tag{57}$$

and

$$Z^{\widehat{W}_t^{L+1}} = Z^{\widehat{W}_0^{L+1}} + \mathring{\theta}_{L+1/f}(Z^{\delta W_1^{L+1}} + \cdots + Z^{\delta W_t^{L+1}}) \tag{58}$$

7. The output deltas have limits

$$\begin{aligned} \delta \mathring{f}_t(\xi) = \mathring{\theta}'_{L+1} \mathbb{E}\, Z^{\delta W_t^{L+1}} Z^{x_t^L(\xi)} \\ + \mathring{\theta}'_{Lf} \mathbb{E}\, Z^{\widehat{W}_{t-1}^{L+1}} Z^{\delta x_t^L(\xi)} \end{aligned} \tag{59}$$

and

$$\mathring{f}_t(\xi) = \delta \mathring{f}_1(\xi) + \cdots + \delta \mathring{f}_t(\xi).$$

8. For gradients:

$$\begin{aligned} Z^{dx_t^L(\xi)} &= Z^{\widehat{W}_t^{L+1}} \\ Z^{dh_t^l(\xi)} &= Z^{dx_t^l(\xi)} \phi'(Z^{h_t^l(\xi)}) \\ Z^{dx_t^{l-1}(\xi)} &= Z^{W_0^{l\top} dh_t^l(\xi)} \\ &\quad - \eta \mathring{\theta}_{W^l} \sum_{s=0}^{t-1} \mathring{\chi}_s Z^{x_s^{l-1}} \mathbb{E}\, Z^{dh_s^l} Z^{dh_t^l(\xi)} \end{aligned}$$

9. Loss derivative

$$\mathring{\chi}_t = \mathcal{L}'(\mathring{f}_t, y_0).$$

---

[59] Here we use **Moment** with the function $\phi(; f_t(\xi_t)) = \mathcal{L}'(f_t(\xi_t), y_t)$ with no input and one parameter (we absorb y_t into ϕ since it does not change with n). The continuity of $\mathcal{L}'$ in its first argument satisfies Assumption L.4(1), so the Master Theorem can apply.

The following fact follows from the results of (Yang, 2020a) (or can be verified by straightforward calculation) and will be useful for us.

Proposition N.23. *$\dot{Z}^{dx_0^l(\xi)} = 0$ and $Z^{dx_0^l(\xi)} = \hat{Z}^{dx_0^l(\xi)}$ for any $\xi \in \mathcal{X}$.*

If the parametrization is pseudostable, then all the $\theta_\bullet$ converge to 0 or 1 so Setup G.2 is satisfied. Therefore, the Master Theorem applies and says that, for any collection of vectors $v^1, \ldots, v^k$ such that Z^{v^i} is defined above, we have

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi(v_\alpha^1, \ldots, v_\alpha^k) \xrightarrow{\text{a.s.}} \mathbb{E}\,\psi(Z^{v^1}, \ldots, Z^{v^k})$$

for any pseudo-Lipschitz ψ. In addition,[60]

$$\delta f_t(\xi) \xrightarrow{\text{a.s.}} \delta \mathring{f}_t(\xi), \quad f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi),$$
$$\chi_t \xrightarrow{\text{a.s.}} \mathring{\chi}_t, \quad \forall \xi \in \mathcal{X}, t \geq 1.$$

We now describe some immediate consequences of this.

N.5.1. Some Immediate Results

Proposition N.24. *A pseudostable parametrization is trivial if*

$$2a_{L+1} + c > 1 \quad \text{and} \quad a_{L+1} + b_{L+1} + r > 1.$$

Proof. In this case, $\theta'_{L+1}, \theta'_{Lf}, \theta'_{L,L+1} \to 0$, and $\delta\mathring{f}_t(\xi) = 0$ for all t and $\xi \in \mathcal{X}$ by Eq. (59). □

Proposition N.25. *A pseudostable parametrization is stable.*

Proof. For a pseudostable parametrization, all of θs converge to 1 or 0, and all of the $Z^{\delta h_t^l(\xi)}, Z^{\delta x_t^l(\xi)}$ have well defined (finite) limits, which implies $\Delta h_t^l(\xi), \Delta x_t^l(\xi) = O_*(1), \forall l \in [L],$ and $f_t(\xi) = O_*(1)$. □

Proposition N.26. *Consider a pseudostable parametrization. If $r > 0$, then it fixes all (pre)features and all (pre)feature kernels. In addition, $\Delta W_t^{L+1} \Delta x_t^L(\xi) \xrightarrow{\text{a.s.}} 0$.*

Proof. If $r > 0$, then $\theta_l \to 0$ for all $l \in [L]$, so that for all $z \in \{x^l, h^l\}_l$, $\Delta z_t(\xi) = z_t(\xi) - z_0(\xi) = \theta_z \delta z_1(\xi) + \cdots + \theta_z \delta z_t(\xi)$ has $\|\Delta z_t(\xi)\|^2/n \xrightarrow{\text{a.s.}} 0$ by Eq. (53) and the Master Theorem, i.e. all features are fixed. Similarly, for any pair $\xi, \bar{\xi} \in \mathcal{X}$, $z_t(\xi)^\top z_t(\bar{\xi})/n - z_0(\xi)^\top z_0(\bar{\xi})/n \xrightarrow{\text{a.s.}} 0$, so all feature kernels are fixed. Finally, $r > 0$ implies $\theta'_{L,L+1} \to 0$, which means $\Delta W_t^{L+1} \Delta x_t^L(\xi) \xrightarrow{\text{a.s.}} 0$ by the Master Theorem. □

[60] Again, if $a_{L+1} + b_{L+1} = 1/2$, remember we are conditioning on $f_0(\xi), \xi \in \mathcal{X}$.

Proposition N.27. *An initialization-stable parametrization with $r < 0$ is not stable.*

Proof. If $r < 0$, then there is some $\ell \in [L]$ such that $\theta_L \geq \cdots \geq \theta_\ell > 1 \geq \theta_{\ell-1} \geq \cdots \geq \theta_1$. For $h = h^\ell, x = x^{\ell-1}, W = W^\ell$, we would have $\theta_{x/h} = \theta_{\ell-1}/\theta_\ell \to 0$, $\theta_{W/h} = 1$, and $\theta_{Wx/h} = \theta_{W/h}\theta_{\ell-1} \to 0$. The Tensor Program up to the definition of $\delta h_1(\xi_0)$ satisfies the conditions of the Master Theorem. Therefore, $\|\delta h_1(\xi_0)\|^2/2 \xrightarrow{\text{a.s.}} \mathbb{E}(Z^{\delta h_1(\xi_0)})^2 = \mathbb{E}(\eta\mathring{\chi}_{t-1} Z^{dh_0}\,\mathbb{E}\,Z^{x_0} Z^{x_1(\xi_0)})^2$. If $\xi_0 \neq 0$, then $\mathbb{E}(Z^{dh_0})^2 > 0$. If η is in addition sufficiently small but nonzero, then $\mathbb{E}\,Z^{x_0} Z^{x_1(\xi_0)} \approx \mathbb{E}(Z^{x_0})^2 > 0$. Therefore, under these conditions, and with a training sequence that has $\mathring{\chi}_0 \neq 0$, we have $\mathbb{E}(\eta\mathring{\chi}_{t-1} Z^{dh_0}\,\mathbb{E}\,Z^{x_0} Z^{x_1(\xi_0)})^2 > 0$, so that $\delta h_1(\xi_0) = \Theta_{\xi_0}(1)$. However, $\Delta h_1(\xi_0) = \theta_h \delta h_1(\xi_0)$ and $\theta_h = \theta_\ell \to \infty$. Hence $\Delta h_1(\xi_0) \neq O_{\xi_0}(1)$, as desired. □

N.6. $r > 0$ Implies Kernel Regime

In this section, we analyze the case when $r > 0$. Our main result is deriving the corresponding infinite-width kernel gradient descent dynamics (Theorem N.31). Nothing here depends on ϕ being tanh or σ-gelu.

Preliminary Derivations If $r > 0$, then $\mathring{\theta}_l = \mathring{\theta}_{W^l} = 0$ for all $l \in [L]$, so that we have

$$Z^{h_t^l(\xi)} = Z^{h_0^l(\xi)}, Z^{x_t^l(\xi)} = Z^{x_0^l(\xi)}, Z^{dh_t^l(\xi)} = Z^{dh_0^l(\xi)},$$
$$Z^{dx_t^l(\xi)} = Z^{dx_0^l(\xi)}, Z^{\widehat{W}_t^{L+1}} = Z^{\widehat{W}_0^{L+1}}$$

for all t and $\xi \in \mathcal{X}$. Let $\ell \in [L]$ be the unique ℓ such that $1 = \theta_L/\theta_L = \cdots = \theta_\ell/\theta_L > \theta_{\ell-1}/\theta_L \geq \cdots \geq \theta_1/\theta_L$. Then for $l \geq \ell + 1$ and shorthand $h = h^l, x = x^{l-1}, W = W^l$, we have $\mathring{\theta}_{x/h} = 1$, $\mathring{\theta}_{Wx/h} = 0$ and, by Eq. (56),

$$\begin{aligned} Z^{\delta h_t(\xi)} &= Z^{W_0 \delta x_t(\xi)} - \eta\mathring{\chi}_{t-1}\mathring{\theta}_{W/h} Z^{dh_{t-1}}\,\mathbb{E}\,Z^{x_{t-1}} Z^{x_t(\xi)}, \\ &= Z^{W_0 \delta x_t(\xi)} - \eta\mathring{\chi}_{t-1}\mathring{\theta}_{W/h} Z^{dh_0(\xi_{t-1})}\,\mathbb{E}\,Z^{x_0(\xi_{t-1})} Z^{x_0(\xi)} \end{aligned} \tag{60}$$

where $\mathring{\theta}_{W/h}$ can be either 0 or 1. For $l = \ell$, because $\theta_h = \theta_l = \max_{m \leq l} \theta_{W^m} = \max(\theta_{W^l}, \theta_{l-1}) = \max(\theta_{W^l}, \theta_x)$ so $\mathring{\theta}_{x/h} = \mathring{\theta}_{Wx/h} = 0$ and $\mathring{\theta}_{W/h} = 1$, we also have

$$\begin{aligned} Z^{\delta h_t(\xi)} &= -\eta\mathring{\chi}_{t-1} Z^{dh_{t-1}}\,\mathbb{E}\,Z^{x_{t-1}} Z^{x_t(\xi)} \\ &= -\eta\mathring{\chi}_{t-1} Z^{dh_0(\xi_{t-1})}\,\mathbb{E}\,Z^{x_0(\xi_{t-1})} Z^{x_0(\xi)}. \end{aligned} \tag{61}$$

Finally, for all $l \in [L]$, we have, by Eq. (54),

$$Z^{\delta x_t(\xi)} = \phi'(Z^{h_{t-1}(\xi)}) Z^{\delta h_t(\xi)} = \phi'(Z^{h_0(\xi)}) Z^{\delta h_t(\xi)}.$$

Definition N.28. For $1 \leq m \leq l$ and $\xi, \zeta \in \mathcal{X}$, define

$$\begin{aligned} \Sigma^{ml}(\xi, \zeta) \stackrel{\text{def}}{=}\; &\mathbb{E}\,Z^{x_0^m(\xi)} Z^{x_0^m(\zeta)} \\ &\times \mathbb{E}\,\phi'(Z^{h_0^{m+1}(\xi)})\phi'(Z^{h_0^{m+1}(\zeta)}) \times \cdots \\ &\times \mathbb{E}\,\phi'(Z^{h_0^l(\xi)})\phi'(Z^{h_0^l(\zeta)}). \end{aligned}$$

We also define

$$\Sigma^{0l}(\xi,\zeta) \overset{\text{def}}{=} \xi^\top \zeta \times \mathbb{E}\,\phi'(Z^{h_0^{m+1}(\xi)})\phi'(Z^{h_0^{m+1}(\zeta)}) \times \cdots \times \mathbb{E}\,\phi'(Z^{h_0^{l}(\xi)})\phi'(Z^{h_0^{l}(\zeta)})$$

For example,

$$\Sigma^{ll}(\xi,\zeta) = \mathbb{E}\, Z^{x_0^l(\xi)} Z^{x_0^l(\zeta)}$$
$$\Sigma^{l,l+1}(\xi,\zeta) = \mathbb{E}\, Z^{x_0^l(\xi)} Z^{x_0^l(\zeta)}\,\mathbb{E}\,\phi'(Z^{h_0^{l+1}(\xi)})\phi'(Z^{h_0^{l+1}(\zeta)}),$$

and so on.

Notation For brevity, below we will shorthand $\vartheta_m = \theta_{W^m/h^m}$. We write $Z^x \equiv Z^y \mod \hat{Z}^{W\bullet}$ if $Z^x - Z^y$ is a linear combination of $\hat{Z}^{Wu}$ for various vectors u.

Lemma N.29. *For any input ξ, any $l \geq \ell$, at any time t,*

$$Z^{\delta h_t^l(\xi)} \equiv -\eta\mathring{\chi}_{t-1} Z^{dh_0^l(\xi_{t-1}a)} \times \sum_{m=\ell-1}^{l-1} \mathring{\vartheta}_{m+1}\Sigma^{m,l-1}(\xi_{t-1},\xi) \mod \hat{Z}^{W_0^l\bullet}. \tag{62}$$

Proof. We proceed by induction.

Base Case $l=\ell$: this is given by Eq. (61).

Induction: Assume Eq. (62) holds for $l-1$, and we shall prove it for l.

To alleviate notation, we write $x = x_t^{l-1}, \bar{x} = x_{t-1}^{l-1}, x_0 = x_0^{l-1}, h = h_t^{l-1}, \bar{h} = h_{t-1}^{l-1}, h_0 = h_0^{l-1}, \bar{\xi} = \xi_{t-1}, W = W_0^l$, i.e. we use $\bar{\bullet}$ to denote time $t-1$ in contrast to $\bullet$ for time t, and we suppress layer index. In contrast, we will write h_0^l, h_t^l, and ξ for their usual meanings.

First, note that $Z^{\delta x(\xi)} = \phi'(Z^{\bar{h}(\xi)})Z^{\delta h(\xi)}$ by Eq. (54). Because $Z^{\bar{h}(\xi)} = Z^{h_0(\xi)}$, and, by induction hypothesis, $Z^{\delta h(\xi)}$ is a scalar multiple of $Z^{dh_0(\bar{\xi})} = Z^{dx_0(\bar{\xi})}\phi'(Z^{h_0(\bar{\xi})})$, $Z^{\delta x(\xi)}$ is symbolically solely a function of $Z^{h_0(\xi)}, Z^{h_0(\bar{\xi})}, Z^{dx_0(\bar{\xi})}$,all of which are equal to their $\hat{Z}$ versions (with the last due to Proposition N.23). Among these, only $Z^{dx_0(\bar{\xi})} = Z^{W^\top dh_0^l(\bar{\xi})}$ is constructed from matrix multiplication with $W_0^\top$. Thus,

$$\dot{Z}^{W_0\delta x(\xi)} = Z^{dh_0^l(\bar{\xi})}\,\mathbb{E}\,\frac{\partial Z^{\delta x(\xi)}}{\partial Z^{dx_0(\bar{\xi})}} = Z^{dh_0^l(\bar{\xi})}\,\mathbb{E}\,\phi'(Z^{h_0(\xi)})\frac{\partial Z^{\delta h(\xi)}}{\partial Z^{dx_0(\bar{\xi})}}. \tag{63}$$

By induction hypothesis,

$$\frac{\partial Z^{\delta h(\xi)}}{\partial Z^{dx_0(\bar{\xi})}} = -\eta\mathring{\chi}_{t-1}\phi'(Z^{h_0(\bar{\xi})})\sum_{m=\ell-1}^{l-2}\mathring{\vartheta}_{m+1}\Sigma^{m,l-2}(\bar{\xi},\xi).$$

Therefore,

$$\mathbb{E}\,\phi'(Z^{h_0(\xi)})\frac{\partial Z^{\delta h(\xi)}}{\partial Z^{dx_0(\bar{\xi})}} = -\eta\mathring{\chi}_{t-1}\,\mathbb{E}\left[\phi'(Z^{h_0(\xi)})\phi'(Z^{h_0(\bar{\xi})})\right] \times \sum_{m=\ell-1}^{l-2}\mathring{\vartheta}_{m+1}\Sigma^{m,l-2}(\bar{\xi},\xi).$$

By definition of Σ^{ml}, this equals

$$\mathbb{E}\,\phi'(Z^{h_0(\xi)})\frac{\partial Z^{\delta h(\xi)}}{\partial Z^{dx_0(\bar{\xi})}} = -\eta\mathring{\chi}_{t-1}\sum_{m=\ell-1}^{l-2}\mathring{\vartheta}_{m+1}\Sigma^{m,l-1}(\bar{\xi},\xi).$$

Plugging this back into Eq. (63), we get

$$\dot{Z}^{W_0\delta x(\xi)} = -\eta\mathring{\chi}_{t-1}Z^{dh_0^l(\bar{\xi})}\sum_{m=\ell-1}^{l-2}\mathring{\vartheta}_{m+1}\Sigma^{m,l-1}(\bar{\xi},\xi). \tag{64}$$

Finally, by Eq. (60),

$$Z^{\delta h_t^l(\xi)} = \dot{Z}^{W_0\delta x(\xi)} - \eta\mathring{\chi}_{t-1}\mathring{\vartheta}_l Z^{dh_0^l(\bar{\xi})}\,\mathbb{E}\,Z^{x_0(\bar{\xi})}Z^{x_0(\xi)} = \dot{Z}^{W_0\delta x(\xi)} - \eta\mathring{\chi}_{t-1}\mathring{\vartheta}_l Z^{dh_0^l(\bar{\xi})}\Sigma^{l-1,l-1}(\bar{\xi},\xi).$$

Together with Eq. (64), this completes the induction. ☐

Lemma N.30. *Assume pseudostability, $r>0$, and $a_{L+1}+b_{L+1} \leq 2a_{L+1}+c$. If $\mathring{\theta}_{L+1/f}=1$ then $\mathring{\theta}'_{Lf}=0$.*

Proof. $a_{L+1}+b_{L+1} \leq 2a_{L+1}+c$ iff $\theta_{L+1} \leq \theta_f$. So $\mathring{\theta}_{L+1/f}=1$ implies $\theta_{L+1}=\theta_f$. By pseudostability, $n\theta_{L+1} \leq 1$. Since $\theta_L = n^{-r}$, we have $\theta'_{Lf} = n\cdot n^{-r}\cdot\theta_f = n^{-r}\cdot n\theta_{L+1} < 0$ since $r>0$. Therefore $\mathring{\theta}'_{Lf}=0$. ☐

Theorem N.31. *Consider a pseudostable parametrization. At any time t, for any input $\xi \in \mathcal{X}$, we have*

$$\delta\mathring{f}_t(\xi) = -\eta\mathring{\chi}_{t-1}\Sigma(\xi_{t-1},\xi),$$

where the kernel Σ is defined for any $\xi,\zeta \in \mathcal{X}$ by

$$\Sigma(\zeta,\xi) \overset{\text{def}}{=} \mathring{\theta}'_{L+1}\Sigma^{LL}(\zeta,\xi) + \mathring{\theta}'_{Lf}\sum_{m=\ell-1}^{L-1}\mathring{\vartheta}_{m+1}\Sigma^{mL}(\zeta,\xi).$$

Observe that in the NTK parametrization, $\ell=1$, and $\mathring{\theta}'_{L+1} = \mathring{\theta}'_{Lf} = \mathring{\vartheta}_{m+1} = 1$ for all m, so $\Sigma = \sum_{m=0}^{L}\Sigma^{mL}$ is precisely the NTK (for MLP without biases).

Proof. By Eqs. (58) and (59),

$$\delta\mathring{f}_t(\xi) = \mathring{\theta}'_{L+1}\,\mathbb{E}\,Z^{\delta W_t^{L+1}}Z^{x_t^L(\xi)} + \mathring{\theta}'_{Lf}\,\mathbb{E}\,Z^{\widehat{W}_{t-1}^{L+1}}Z^{\delta x_t^L(\xi)}$$
$$Z^{\widehat{W}_t^{L+1}} = Z^{\widehat{W}_0^{L+1}} + \mathring{\theta}_{L+1/f}(Z^{\delta W_1^{L+1}} + \cdots + Z^{\delta W_t^{L+1}}).$$

Now by Lemma N.30, either $\mathring{\theta}_{L+1/f} = 0$ or $\mathring{\theta}'_{Lf} = 0$. In both cases, $(Z^{\delta W_1^{L+1}} + \cdots + Z^{\delta W_t^{L+1}})$ contributes 0 to $\delta \mathring{f}_t(\xi)$. So we can replace $Z^{\widehat{W}_{t-1}^{L+1}}$ with $Z^{\widehat{W}_0^{L+1}}$ above, and write

$$\delta \mathring{f}_t(\xi) = \mathring{\theta}'_{L+1} \mathbb{E}\, Z^{\delta W_t^{L+1}} Z^{x_t^L(\xi)} + \mathring{\theta}'_{Lf} \mathbb{E}\, Z^{\widehat{W}_0^{L+1}} Z^{\delta x_t^L(\xi)}.$$

If Eq. (62) is true for $l = L$, then

$$\begin{aligned}&\mathbb{E}\, Z^{\widehat{W}_0^{L+1}} Z^{\delta x_t^L(\xi)}\\ &= -\eta \mathring{\chi}_{t-1} \mathbb{E}\, Z^{\widehat{W}_0^{L+1}} Z^{dh_0^L(\xi_{t-1})} \phi'(Z^{h_0^L(\xi)})\\ &\quad \times \sum_{m=\ell-1}^{L-1} \mathring{\vartheta}_{m+1} \Sigma^{m,L-1}(\xi_{t-1}, \xi)\end{aligned}$$

where the contributions from $\hat{Z}^{W_0^L \bullet}$ in $Z^{\delta x_t^L(\xi)}$ vanish as they are independent from $Z^{\widehat{W}_0^{L+1}}$. Since $Z^{dh_0^L(\xi)} = Z^{\widehat{W}_0^{L+1}} \phi'(Z^{h_0^L(\xi)})$, we continue

$$\begin{aligned}&\mathbb{E}\, Z^{\widehat{W}_0^{L+1}} Z^{\delta x_t^L(\xi)}\\ &= -\eta \mathring{\chi}_{t-1}\\ &\quad \times \mathbb{E} \left(Z^{\widehat{W}_0^{L+1}}\right)^2 \phi'(Z^{h_0^L(\xi_{t-1})}) \phi'(Z^{h_0^L(\xi)})\\ &\quad \times \sum_{m=\ell-1}^{L-1} \mathring{\vartheta}_{m+1} \Sigma^{m,L-1}(\xi_{t-1}, \xi)\\ &= -\eta \mathring{\chi}_{t-1} \sum_{m=\ell-1}^{L-1} \mathring{\vartheta}_{m+1} \Sigma^{mL}(\xi_{t-1}, \xi).\end{aligned}$$

Similarly, by Eq. (57),

$$\begin{aligned}\mathbb{E}\, Z^{\delta W_t^{L+1}} Z^{x_t^L(\xi)} &= -\eta \mathring{\chi}_{t-1} \mathbb{E}\, Z^{x_{t-1}^L(\xi_{t-1})} Z^{x_t^L(\xi)}\\ &= -\eta \mathring{\chi}_{t-1} \mathbb{E}\, Z^{x_0^L(\xi_{t-1})} Z^{x_0^L(\xi)}\\ &= -\eta \mathring{\chi}_{t-1} \Sigma^{LL}(\xi_{t-1}, \xi).\end{aligned}$$

Altogether, these prove the desired claim. □

Corollary N.32. *A pseudostable parametrization with $r > 0$ is nontrivial iff $a_{L+1} + b_{L+1} + r = 1$ or $2a_{L+1} + c = 1$.*

Proof. The kernel Σ in Theorem N.31 is nonzero iff $\mathring{\theta}'_{L+1}$ or $\mathring{\theta}'_{Lf}$ is 1, which is equivalent to saying $a_{L+1} + b_{L+1} + r = 1$ or $2a_{L+1} + c = 1$. □

Corollary N.33. *An initialization-stable parametrization with $r > 0$ but $a_{L+1} + b_{L+1} + r < 1$ or $2a_{L+1} + c < 1$ is not stable.*

Proof. If $a_{L+1} + b_{L+1} + r < 1$ or $2a_{L+1} + c < 1$, then $\theta'_{L+1} \to \infty$ or $\theta'_{Lf} \to \infty$. Clearly, from the definition, $\Sigma^{mL}(\xi, \xi) > 0$ for any $\xi \neq 0$ and $m \in [0, L]$. All of our reasoning leading up to Theorem N.31 applied at $t = 1$ holds, so Theorem N.31 (along with the Master Theorem) implies $|\delta f_t(\xi)| \xrightarrow{\text{a.s.}} \infty$. □

Corollary N.34. *If $a_{L+1} + b_{L+1} + r > 1$ and $2a_{L+1} + c = 1$, then for all $\xi \in \mathcal{X}$, $\mathring{f}_t(\xi) \xrightarrow{\text{a.s.}} 0$ and $\delta \mathring{f}_t(\xi) = -\eta \mathring{\chi}_{t-1} \Sigma^{LL}(\xi_{t-1}, \xi)$, i.e. we have the Neural Network-Gaussian Process (NNGP) limit.*

Conventionally, the NNGP limit is associated with only training the last layer and nothing else. This result says that the same limit can be achieved if we train the body of the network slightly, so that Δx_t^L does not interact with W_0^{L+1} enough (embodied in the inequality $a_{L+1} + b_{L+1} + r > 1$) to cause changes in f_t.

Proof. The premise implies $\mathring{\theta}'_{L+1} = 1$ and $\mathring{\theta}'_{Lf} = 0$, and the rest follows from Theorem N.31. □

Remark N.35. We have assumed for simplicity of the proof that $a_{L+1} + b_{L+1} \leq 2a_{L+1} + c$. If this is not the case, then we can easily see Corollary N.34 applies anyway.

N.7. $r = 0$ Implies Feature Learning

In this section, we assume $r = 0$ and show any such pseudostable parametrization 1) admits (pre)feature learning and (pre)feature kernel evolution, and 2) is *not* in kernel regime (Theorem N.50). The overarching logic goes like this.

1. The Master Theorem shows that the specific entry $\frac{1}{n}\|x_1^L(\xi_0)\|^2$ of the feature kernel converges to $\mathbb{E}(Z^{x_1^L(\xi_0)})^2$. If the learning rate $\eta = 0$, then $x_1^L(\xi_0) = x_0^L$ and $\mathbb{E}(Z^{x_1^L(\xi_0)})^2 = \mathbb{E}(Z^{x_0^L})^2$. We hope to say that as η increases, $\mathbb{E}(Z^{x_1^L(\xi_0)})^2$ moves away from $\mathbb{E}(Z^{x_0^L})^2$, which would imply feature kernel evolution in layer L. To do so, we compute $\partial_\eta^2 \mathbb{E}(Z^{x_1^L(\xi_0)})^2$ evaluated at $\eta = 0$ and show it is nonzero (it turns out ∂_η vanishes, so the next best thing is ∂_η^2). This then also implies feature learning in layer L. Analogous results for prefeatures and for other layers can be derived similarly.

2. If the parametrization is in the kernel regime with kernel K, the first step of SGD in the large width limit would look like $\mathring{f}_1(\xi) - \mathring{f}_0(\xi) = -\eta \mathring{\chi}_0 K(\xi, \xi_0)$; in particular, $\mathring{f}_1(\xi) - \mathring{f}_0(\xi)$ is linear in η. To show that a pseudostable parametrization with $r = 0$ is not in the kernel regime, we will show $\partial_\eta^3(\mathring{f}_1(\xi) - \mathring{f}_0(\xi)) = \partial_\eta^3 \mathring{f}_1(\xi)$ is nonzero. (It turns out ∂_η^2 vanishes, so the next best thing is ∂_η^3).

To calculate these η derivatives, we will derive recurrence relations involving quantities defined below (see Lemma N.37 and Theorem N.40).

Setup and Notation First, write

$$Z_t^l \stackrel{\text{def}}{=} Z^{h_t^l(\xi_0)},\ \hat{Z}_t^l \stackrel{\text{def}}{=} \hat{Z}^{W^l x_t^{l-1}(\xi_0)},\ \dot{Z}_0^l \stackrel{\text{def}}{=} Z^{dh_0^l}.$$

Note that $\dot{Z}_0^l$ is a centered Gaussian independent from $\hat{Z}_t^l, Z_t^l$. Then we define

$$\gamma^l(\eta) \stackrel{\text{def}}{=} \mathbb{E}\,\phi(Z_0^l)\phi(Z_1^l), \quad \gamma_{11}^l(\eta) \stackrel{\text{def}}{=} \mathbb{E}\,\phi'(Z_0^l)\phi'(Z_1^l),$$
$$\gamma_{02}^l(\eta) \stackrel{\text{def}}{=} \mathbb{E}\,\phi(Z_0^l)\phi''(Z_1^l),$$
$$\gamma_{20}^l(\eta) \stackrel{\text{def}}{=} \mathbb{E}\,\phi''(Z_0^l)\phi(Z_1^l), \quad \lambda^l(\eta) \stackrel{\text{def}}{=} \mathbb{E}\,\phi(Z_1^l)^2$$

where the dependence on η is from Z_1^l. Naturally, since ϕ and ϕ' are not almost everywhere zero, we have $\gamma^l(0), \lambda^l(0), \gamma_{11}^l(0) > 0$. Note at $\eta = 0$, we have $Z_1^l = Z_0^l$, so $\gamma^l(0) = \lambda^l(0) = \mathbb{E}\,\phi(Z_0^l)^2$. Observe that $(\hat{Z}_1^l, \hat{Z}_0^l)$ is jointly Gaussian with mean zero and covariance

$$\Gamma^l(\eta) \stackrel{\text{def}}{=} \begin{pmatrix} \lambda^l(\eta) & \gamma^l(\eta) \\ \gamma^l(\eta) & \lambda^l(0) \end{pmatrix}. \tag{65}$$

WLOG, for simplicity of notation, we assume we choose a training routine such that $\mathring{\chi}_0 = 1$. We assume $\xi_0 \neq 0$.

Since $r = 0$, WLOG we can suppose for some $\ell \in [L]$, we have $\theta_L = \cdots = \theta_\ell = 1 > \theta_{\ell-1} \geq \cdots \geq \theta_1$.

Lemma N.36. *With the setup above, we have*

$$Z_0^{\ell-1} = Z_1^{\ell-1}, \ldots, Z_0^1 = Z_1^1,$$

and

$$Z_1^l = \hat{Z}_1^l + \eta\beta^l \dot{Z}_0^l \phi'(Z_0^l), \quad \forall l \in [\ell, L],$$

where β^l is defined recursively by

$$\beta^l = \beta^l(\eta) \stackrel{\text{def}}{=} -\gamma^{l-1}(\eta) + \beta^{l-1}(\eta)\gamma_{11}^{l-1}(\eta)$$
$$\beta^{\ell-1}(\eta) \stackrel{\text{def}}{=} 0.$$

Additionally, $\beta^l(0) < 0$ for all $l \geq \ell$.

Proof. Straightforward calculation using Moment and Zdot. Here, $-\gamma^{l-1}(\eta)$ comes from $\Delta W_1^l x_1^1(\xi_0)$ and $\beta^{l-1}(\eta)\gamma_{11}^{l-1}(\eta)$ comes from $\dot{Z}^{h_1^l(\xi_0)}$. Since $\gamma^l(0), \gamma_{11}^{l-1}(0) > 0$ for all l, the recurrence on β^l implies that $\beta^l(0) < 0$ for all $l \geq \ell$. ☐

N.7.1. Deriving Recurrence Relations on $\partial_\eta\lambda^l, \partial_\eta\gamma^l, \partial_\eta^2\lambda^l, \partial_\eta^2\gamma^l$

Below, we derive the recurrence relations required for our main result. They depend on the following constants.

$$\kappa_1^l \stackrel{\text{def}}{=} \mathbb{E}\left[(\phi^2)''(Z_0^l)\right], \quad \kappa_2^l \stackrel{\text{def}}{=} \mathbb{E}\left[(\phi^2)''(Z_0^l)\phi'(Z_0^l)^2\right],$$
$$\kappa_3^l \stackrel{\text{def}}{=} \mathbb{E}\left[\phi(Z_0^l)\phi''(Z_0^l)\phi'(Z_0^l)^2\right].$$

Lemma N.37. *With the setup above, we have, for all $l \in [L]$,*

$$\partial_\eta\lambda^l = \frac{1}{2}\kappa_1^l\partial_\eta\lambda^{l-1} \tag{66}$$
$$\partial_\eta\gamma^l = \frac{1}{2}\gamma_{02}^l\partial_\eta\lambda^{l-1} + \gamma_{11}^l\partial_\eta\gamma^{l-1}.$$

Proof. We first derive the recurrence on $\partial_\eta\lambda^l$. By Lemma N.38 below, we have

$$\partial_\eta\lambda^l = 2\,\mathbb{E}\,\phi(Z_1^l)\partial_\eta\phi(Z_1^l) + \frac{1}{2}\,\mathbb{E}(\phi^2)''(Z_1^l)\partial_\eta\lambda^{l-1}.$$

Since

$$\partial_\eta\phi(Z_1^l) = \phi'(Z_1^l)(\beta^l\dot{Z}_0^l\phi'(Z_0^l) + \eta\dot{Z}_0^l\phi'(Z_0^l)\partial_\eta\beta^l), \tag{67}$$

we compute

$$\mathbb{E}\,\phi(Z_1^l)\partial_\eta\phi(Z_1^l) = \mathbb{E}\,\phi(Z_1^l)\phi'(Z_1^l)(\beta^l\dot{Z}_0^l\phi'(Z_0^l) + \eta\dot{Z}_0^l\phi'(Z_0^l)\partial_\eta\beta^l) = 0$$

because $\dot{Z}_0^l$ is independent from everything else in the first expectation. This directly implies the result for $\partial_\eta\lambda^l$.

For $\partial_\eta\gamma^l$, let $\Sigma = \Sigma(\eta) \stackrel{\text{def}}{=} \begin{pmatrix} \gamma_{02}^l & \gamma_{11}^l \\ \gamma_{11}^l & \gamma_{20}^l \end{pmatrix}$. With Γ^{l-1} as in Eq. (65), we have

$$\partial_\eta\gamma^l = \mathbb{E}\,\phi(Z_0^l)\partial_\eta\phi(Z_1^l) + \frac{1}{2}\langle\Sigma, \partial_\eta\Gamma^{l-1}\rangle$$

By same reasoning as in Eq. (66), the first term of this sum is zero. Since $\partial_\eta\Gamma^{l-1}(\eta) \stackrel{\text{def}}{=} \begin{pmatrix} \partial_\eta\lambda^{l-1}(\eta) & \partial_\eta\gamma^{l-1}(\eta) \\ \partial_\eta\gamma^l(\eta) & 0 \end{pmatrix}$, we have

$$\partial_\eta\gamma^l = \frac{1}{2}\langle\Sigma, \partial_\eta\Gamma^{l-1}\rangle = \frac{1}{2}\gamma_{02}^l\partial_\eta\lambda^{l-1} + \gamma_{11}^l\partial_\eta\gamma^{l-1}.$$

☐

Lemma N.38. *Consider a twice continuously differentiable f and Gaussian vector $Z \sim \mathcal{N}(0, \Sigma)$ such that f and Σ both depend on a parameter η. Then*

$$\partial_\eta\,\mathbb{E}\,f(Z) = \mathbb{E}\,\partial_\eta f(Z) + \frac{1}{2}\langle\mathbb{E}\,\nabla^2 f(z), \partial_\eta\Sigma\rangle,$$

where ∇^2 denotes Hessian wrt z, and $\langle,\rangle$ denotes trace inner product of matrices.

Proof. Let $p(z)$ denote the PDF of Z. We have

$$\partial_\eta\,\mathbb{E}\,f(Z) = \partial_\eta\int f(z)p(z)\,\mathrm{d}z$$
$$= \int\partial_\eta f(z)p(z)\,\mathrm{d}z + \int f(z)\partial_\eta p(z)\,\mathrm{d}z$$

The first integral is $\mathbb{E}\,\partial_\eta f(Z)$. The second integral can be rewritten using integration-by-parts as $\langle\mathbb{E}\,\nabla^2 f(z), \partial_\eta\Sigma\rangle$. (e.g. see Lemma F.18 of (Yang et al., 2019)) ☐

We then easily have

Theorem N.39. *For all $l \in [L]$,*

$$\partial_\eta \gamma^l(0) = \partial_\eta \lambda^l(0) = 0.$$

Proof. For $l < \ell$, we obviously have $\partial_\eta \gamma^l(\eta) = \partial_\eta \lambda^l(0) = 0$ for all η. Then this follows from Lemma N.37 and a simple induction. ☐

Unfortunately, this means that the first η derivative doesn't give us what we need. So we try the second derivative, which will turn out to work.

Theorem N.40. *For all $l < \ell$,$\partial_\eta^2 \lambda^l(0) = \partial_\eta^2 \gamma^l(0) = 0$, and for all $l \geq \ell$,*

$$\partial_\eta^2 \lambda^l(0) = C\kappa_2^l + \frac{1}{2}\kappa_1^l \partial_\eta^2 \lambda^{l-1}(0)$$
$$\partial_\eta^2 \gamma^l(0) = C\kappa_3^l + \frac{1}{2}\gamma_{02}^l(0)\partial_\eta^2 \lambda^{l-1}(0) + \gamma_{11}^l(0)\partial_\eta^2 \gamma^{l-1}(0),$$

where $C = 2(\beta^l(0))^2\, \mathbb{E}(\dot{Z}_0^l)^2 > 0$.

Proof. We start with the $\partial_\eta^2 \lambda^l(0)$ recurrence. For $l \geq \ell$, $\partial_\eta^2 \lambda^l$ is a sum of 3 terms, representing 1) 2 derivatives in the integrand, 2) 2 derivatives in the Gaussian variance, and 3) 1 derivative each. When evaluated at $\eta = 0$, only the first two terms survive because $\partial_\eta \lambda^{l-1}(0) = 0$ by Theorem N.39:

$$\partial_\eta^2 \lambda^l(0) = \mathbb{E}\, \partial_\eta^2 \phi^2(Z_1^l)|_{\eta=0} + \frac{1}{2}\, \mathbb{E}(\phi^2)''(Z_0^l)\partial_\eta^2 \lambda^{l-1}(0).$$

Now

$$\begin{aligned}
&\mathbb{E}\, \partial_\eta^2 \phi^2(Z_1^l) \\
&= 2\partial_\eta(\mathbb{E}\, \phi(Z_1^l)\phi'(Z_1^l)(\beta^l \dot{Z}_0^l \phi'(Z_0^l) + \eta \dot{Z}_0^l \phi'(Z_0^l)\partial_\eta \beta^l)) \\
&= 2\, \mathbb{E}(\phi^2)''(Z_1^l)(\beta^l \dot{Z}_0^l \phi'(Z_0^l) + \eta \dot{Z}_0^l \phi'(Z_0^l)\partial_\eta \beta^l)^2 + \cdots
\end{aligned}$$

where other terms appear in this sum but they vanish because $\dot{Z}_0^l$ appears unpaired in the expectation. Thus,

$$\mathbb{E}\, \partial_\eta^2 \phi^2(Z_1^l)|_{\eta=0} = 2(\beta^l(0))^2\, \mathbb{E}(\dot{Z}_0^l)^2\, \mathbb{E}(\phi^2)''(Z_0^l)\phi'(Z_0^l)^2.$$

Plugging this back in, we get the recurrence on $\partial_\eta^2 \lambda^l(0)$.

The $\partial_\eta^2 \gamma^l(0)$ recurrence is derived similarly. ☐

The following result will be useful for showing $\partial_\eta^3 \mathring{f}_1(\xi_0) \neq 0$.

Theorem N.41. *Define*

$$\dot{\kappa}_3^l \overset{def}{=} \mathbb{E}\left[\phi'''(Z_0^l)\phi'(Z_0^l)^3\right], \quad \gamma_{13}^l \overset{def}{=} \mathbb{E}\, \phi'(Z_0^l)\phi'''(Z_0^l),$$
$$\gamma_{22}^l \overset{def}{=} \mathbb{E}\, \phi''(Z_0^l)^2.$$

Then for all $l \geq \ell$,

$$\partial_\eta^2 \gamma_{11}^l(0) = C\dot{\kappa}_3^l + \frac{1}{2}\gamma_{13}^l \partial_\eta^2 \lambda^{l-1}(0) + \gamma_{22}^l \partial_\eta^2 \gamma^{l-1}(0),$$

where $C = 2(\beta^l(0))^2\, \mathbb{E}(\dot{Z}_0^l)^2 > 0$.

Proof. Similar to the proof of Theorem N.40. ☐

The following result will be useful for showing prefeature kernel evolution.

Theorem N.42. *For all $l \geq \ell$,*

$$\partial_\eta^2\, \mathbb{E}(Z_1^l)^2|_{\eta=0} = 2C + \gamma_{11}^l(0)\partial_\eta^2 \lambda^{l-1}(0),$$

where $C = 2(\beta^l(0))^2\, \mathbb{E}(\dot{Z}_0^l)^2 > 0$.

Proof. Similar to the proof of Theorem N.40. ☐

N.7.2. Applications to σ-Gelu

The following proposition regarding σ-gelu is easy to verify.

Proposition N.43. *Let ϕ be σ-gelu. For any centered Gaussian $Z \in \mathbb{R}$ with nonzero variance,*

$$\begin{aligned}
\mathbb{E}(\phi^2)''(Z) &> 0 \\
\mathbb{E}(\phi^2)''(Z)\phi'(Z)^2 &> 0 \\
\mathbb{E}\, \phi(Z)\phi''(Z)\phi'(Z)^2 &> 0 \\
\mathbb{E}\, \phi(Z)\phi''(Z) &> 0 \\
\mathbb{E}\, \phi''(Z)^2 &> 0,
\end{aligned}$$

and they converge to 0 as $\sigma \to 0$. Also,

$$\mathbb{E}\, \phi'''(Z)\phi'(Z)^3, \mathbb{E}\, \phi'(Z)\phi'''(Z) < 0,$$

and they converge to $-\infty$ as $\sigma \to 0$.

This particularly implies that $\kappa_1^l, \kappa_2^l, \kappa_3^l, \gamma_{02}^l(0), \gamma_{22}^l > 0$ and converges to 0 with small σ, but $\dot{\kappa}_3^l, \gamma_{13}^l < 0$ and diverges to $-\infty$ with small σ.

Theorem N.44. *Consider a pseudostable parametrization with $r = 0$. If ϕ is σ-gelu, then for all $l \geq \ell$,*

$$\partial_\eta^2 \gamma^l(0), \partial_\eta^2 \lambda^l(0) > 0$$

and they converge to 0 as $\sigma \to 0$.

Proof. We always have $(\beta^l(0))^2, \mathbb{E}(\dot{Z}_0^l)^2 > 0$. By Proposition N.43, $\kappa_1^l, \kappa_2^l > 0$ as well. Thus, by Theorem N.40, $\partial_\eta^2 \lambda^l(0) > 0$ for all $l \geq \ell$. By Proposition N.43, $\kappa_3^l, \gamma_{02}^l(0) > 0$, so by Theorem N.40, $\partial_\eta^2 \gamma^l(0) > 0$ for all $l \geq \ell$ as well. As $\sigma \to 0$, $\kappa_1^l, \kappa_2^l, \kappa_3^l, \gamma_{02}^l(0) \to 0$, so $\partial_\eta^2 \lambda^l(0), \partial_\eta^2 \gamma^L(0) \to 0$. ☐

Theorem N.45. *Consider a pseudostable parametrization with $r = 0$. Suppose $a_{L+1} + b_{L+1} + r = 1$ or $2a_{L+1} + c = 1$. If ϕ is σ-gelu for sufficiently small σ, then*

$$\partial_\eta^3 \mathring{f}_1(\xi_0) \neq 0.$$

Proof. We have $\mathring{f}_1(\xi_0) = \mathring{\theta}'_{L+1} \mathbb{E}\, Z^{\delta W_1^{L+1}} Z^{x_1^L(\xi_0)} + \mathring{\theta}'_{Lf} \mathbb{E}\, Z^{\widehat{W}_0^{L+1}} Z^{\delta x_1^L(\xi_0)}$, where at least one of $\mathring{\theta}'_{Lf}$ and $\mathring{\theta}'_{L+1}$ is 1 because $a_{L+1}+b_{L+1}+r=1$ or $2a_{L+1}+c=1$. We have

$$\mathbb{E}\, Z^{\delta W_1^{L+1}} Z^{x_1^L(\xi_0)} = -\eta\, \mathbb{E}\, Z^{x_0^L} Z^{x_1^L(\xi_0)}$$

and

$$\begin{aligned}&\mathbb{E}\, Z^{\widehat{W}_0^{L+1}} Z^{x_1^L(\xi_0)}\\ &= \mathbb{E}\, Z^{\widehat{W}_0^{L+1}} \phi(Z^{h_0^L} - \eta Z^{\widehat{W}_0^{L+1}} \phi'(Z^{h_0^L})\, \mathbb{E}\, Z^{x_0^{L-1}} Z^{x_1^{L-1}(\xi_0)})\\ &= -\eta\, \mathbb{E}\, \phi'(Z^{h_1^L(\xi_0)})\phi'(Z^{h_0^L})\, \mathbb{E}\, Z^{x_0^{L-1}} Z^{x_1^{L-1}(\xi_0)}\end{aligned}$$

where we used Stein's Lemma for the last equality. Thus

$$\partial_\eta^3 \mathring{f}_1(\xi_0) = -\left(\mathring{\theta}'_{L+1}\partial_\eta^2\gamma^L(0) + \mathring{\theta}'_{Lf}\partial_\eta^2(\gamma_{11}^L\gamma^{L-1})(0)\right).$$

Below we will show that for small σ, $\partial_\eta^2\gamma^L(0)$ is small and positive and $\partial_\eta^2(\gamma_{11}^L\gamma^{L-1})(0)$ is large and negative, so $\partial_\eta^3 \mathring{f}_1(\xi_0)$ cannot be 0 no matter the values of $\mathring{\theta}'_{L+1}$ and $\mathring{\theta}'_{Lf}$.

Claim: For sufficiently small σ, $\partial_\eta^2\gamma_{11}^L(0) < 0$. It converges to $-\infty$ as $\sigma \to 0$.

Proof: By Theorem N.41, $\partial_\eta^2\gamma_{11}^l(0) = C\dot\kappa_3^l + \frac{1}{2}\gamma_{13}^l\partial_\eta^2\lambda^{l-1}(0) + \gamma_{22}^l\partial_\eta^2\gamma^{l-1}(0)$. Note $\partial_\eta^2\lambda^{l-1}(0) \geq 0$ by Theorem N.44. Also, by Proposition N.43, $\dot\kappa_3^l, \gamma_{13}^l < 0, \gamma_{22}^l > 0$, and as $\sigma \to 0$, $\dot\kappa_3^l, \gamma_{13}^l \to -\infty, \gamma_{22}^l \to 0$ (as well as $\partial_\eta^2\gamma^{L-1}(0), \partial_\eta^2\lambda^l(0) \to 0$ by Theorem N.44). One can see that C converges to a positive constant as $\sigma \to 0$ as well. Therefore, for small enough σ, $\partial_\eta^2\gamma_{11}^l(0) < 0$, and as $\sigma \to 0$, $\partial_\eta^2\gamma_{11}^L(0) \to -\infty$.

Claim: For sufficiently small σ, $\partial_\eta^2(\gamma_{11}^L\gamma^{L-1})(0) < 0$. It converges to $-\infty$ as $\sigma \to 0$.

Proof: Observe $\partial_\eta^2(\gamma_{11}^L\gamma^{L-1})(0) = \partial_\eta^2\gamma_{11}^L(0)\gamma^{L-1}(0) + \gamma_{11}^L(0)\partial_\eta^2\gamma^{L-1}(0)$ because $\partial_\eta\gamma^{L-1}(0) = 0$ by Theorem N.39. So the above claim and Theorem N.44 yield the desired results.

Finishing the main proof: Therefore, if $\mathring{\theta}'_{L+1} = 1$ but $\mathring{\theta}'_{Lf} = 0$, then $-\partial_\eta^3 \mathring{f}_1(\xi_0) > 0$ because $\partial_\eta^2\gamma^L(0) > 0$; if $\mathring{\theta}'_{L+1} = 0$ but $\mathring{\theta}'_{Lf} = 1$, then $-\partial_\eta^3 \mathring{f}_1(\xi_0) < 0$ for small σ because $\partial_\eta^2(\gamma_{11}^L\gamma^{L-1})(0) < 0$; if $\mathring{\theta}'_{L+1} = \mathring{\theta}'_{Lf} = 1$, then $-\partial_\eta^3 \mathring{f}_1(\xi_0) < 0$ for small σ because $\partial_\eta^2(\gamma_{11}^L\gamma^{L-1})(0) \to -\infty$ while $\partial_\eta^2\gamma^L(0) \to 0$ as $\sigma \to 0$. □

N.7.3. Applications to Tanh

The following property of $\tanh$ is easy to verify.

Proposition N.46. *Let $\phi = \tanh$. For any centered Gaussian $Z \in \mathbb{R}$ with nonzero variance,*

$$\mathbb{E}(\phi^2)''(Z), \mathbb{E}(\phi^2)''(Z)\phi'(Z)^2, \mathbb{E}\,\phi''(Z)^2 > 0,$$

and

$$\mathbb{E}\,\phi(Z)\phi''(Z)\phi'(Z)^2, \mathbb{E}\,\phi(Z)\phi''(Z), \mathbb{E}\,\phi'''(Z)\phi'(Z)^3, \mathbb{E}\,\phi'(Z)\phi'''(Z) < 0.$$

In particular, this means

$$\kappa_1^l, \kappa_2^l, \gamma_{22}^l > 0, \quad \kappa_3^l, \gamma_{02}^l(0), \dot\kappa_3^l, \gamma_{13}^l < 0.$$

Theorem N.47. *Consider a pseudostable parametrization with $r = 0$. If ϕ is $\tanh$, then for all $l \geq \ell$,*

$$\partial_\eta^2\gamma^l(0) < 0, \quad \partial_\eta^2\lambda^l(0) > 0.$$

Proof. Similar to the proof of Theorem N.44, except that here $\kappa_3^l, \gamma_{02}^l(0) < 0$, making $\partial_\eta^2\gamma^l(0) < 0$. □

Theorem N.48. *Consider a pseudostable parametrization with $r = 0$. Suppose $a_{L+1}+b_{L+1}+r = 1$ or $2a_{L+1}+c = 1$. If ϕ is $\tanh$, then*

$$\partial_\eta^3 \mathring{f}_1(\xi_0) \neq 0.$$

Proof. Similar to the proof of Theorem N.45, except in the expression

$$\partial_\eta^3 \mathring{f}_1(\xi_0) = -\left(\mathring{\theta}'_{L+1}\partial_\eta^2\gamma^L(0) + \mathring{\theta}'_{Lf}\partial_\eta^2(\gamma_{11}^L\gamma^{L-1})(0)\right),$$

$\partial_\eta^2\gamma^L(0)$ and $\partial_\eta^2(\gamma_{11}^L\gamma^{L-1})(0)$ are both negative. The former is because of Theorem N.47. The latter is because $\partial_\eta^2\gamma^{L-1}(0) \leq 0$ for the same reason, and $\partial_\eta^2\gamma_{11}^L(0) < 0$ since $\dot\kappa_3^l, \gamma_{13}^l < 0, \gamma_{22}^l > 0$ by Proposition N.46. □

N.7.4. Main Results

Proposition N.49. *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. A pseudostable parametrization with $r = 0$ is nontrivial iff $a_{L+1} + b_{L+1} = 1$ or $2a_{L+1} + c = 1$.*

Proof. If $a_{L+1} + b_{L+1} + r = 1$ or $2a_{L+1} + c = 1$, then Theorem N.45 and Theorem N.48 show that the parametrization is nontrivial. Otherwise, it is trivial by Proposition N.24. □

Theorem N.50. *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. For any nontrivial pseudostable parametrization with $r = 0$, the following are true of the parametrization:*

1. *not in kernel regime*
2. *feature learning*
3. *feature learning in the Lth layer*
4. *feature kernels evolution*
5. *feature kernel evolution in the Lth layer*
6. *prefeature learning*

7. *prefeature learning in the Lth layer*
8. *prefeature kernels evolution*
9. *prefeature kernel evolution in the Lth layer*
10. *if there is feature learning* or *feature kernel evolution* or *prefeature learning* or *prefeature kernel evolution in layer l, then there is feature learning* and *feature kernel evolution* and *prefeature learning* and *prefeature kernel evolution in layers $l, \dots, L$.*

Proof. The parametrization cannot be in kernel regime since $\partial_\eta^3 \mathring{f}_1(\xi_0) \neq 0$ by Theorem N.48 or Theorem N.45. By Theorem N.44 or Theorem N.47, $\partial_\eta^2 \lambda^l(0) > 0$ for all $l \geq \ell$, so the feature kernel evolves in layer $\ell, \dots, L$, for some normalized learning rate $\eta > 0$. This implies feature learning in layer $\ell, \dots, L$, since $Z^{x_1^L(\xi_0)} - Z^{x_0^L} \neq 0$ in this case. This then implies $Z^{h_1^L(\xi_0)} - Z^{h_0^L} \neq 0$, so we have prefeature learning in layer $\ell, \dots, L$. Prefeature kernel evolution in layer $\ell, \dots, L$ is implied by Theorem N.42. Finally, the last statement follows clearly from our logic above. □

Corollary N.51. *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. Consider any initialization-stable parametrization with $r = 0$. If $a_{L+1} + b_{L+1} < 1$ or $2a_{L+1} + c < 1$, then the parametrization is not stable.*

Proof. First suppose $a_{L+1} + b_{L+1} < 1$ and $2a_{L+1} + c \geq 1$. Then $\theta'_{Lf} = n^{1-(a_{L+1}+b_{L+1})} \to \infty$ but $\mathring{\theta}'_{L+1} \leq 1$. As in the proof of Theorem N.45, there is some $\eta \neq 0$ such that $\mathbb{E}\, Z^{\widehat{W}_0^{L+1}} Z^{\delta x_1^L(\xi_0)} = R$ for some $R \neq 0$. Therefore, by the Master Theorem, $\frac{1}{n}\widehat{W}_0^{L+1} \delta x_1^L(\xi_0) \xrightarrow{\text{a.s.}} R \implies |W_0^{L+1} \Delta x_1^L(\xi_0)| = \Theta(n^{1-(a_{L+1}+b_{L+1})}) \to \infty$. This dominates $\Delta W_1^{L+1} x_1^L(\xi_0)$, which by similar reasoning is $O(1)$. So $f_1(\xi_0)$ diverges and the parametrization is not stable.

Now suppose $a_{L+1} + b_{L+1} \geq 1$ and $2a_{L+1} + c < 1$. This violates our simplifying assumption that $a_{L+1} + b_{L+1} \leq 2a_{L+1} + c$, but it's easy to see that $\frac{1}{n}\delta W_1^{L+1} x_1^L(\xi_0) \xrightarrow{\text{a.s.}} -\eta \mathring{\chi}_0 \,\mathbb{E}\, Z^{x_0^L} Z^{x_1^L(\xi_0)}$. For η small enough, this is close to $-\eta \mathring{\chi}_0 \,\mathbb{E}(Z^{x_0^L})^2$ and thus is nonzero. Then $|\Delta W_1^{L+1} x_1^L(\xi_0)| = \Theta(n^{1-(2a_{L+1}+c)}) \to \infty$. This dominates $W_0^{L+1} \Delta x_1^L(\xi_0) = O(1)$, so $f_1(\xi_0)$ diverges. Therefore, the parametrization is not stable.

Finally, suppose both $a_{L+1} + b_{L+1}, 2a_{L+1} + c < 1$. If $a_{L+1} + b_{L+1} \neq 2a_{L+1} + c$, then we have one of $\Delta W_1^{L+1} x_1^L(\xi_0)$ and $W_0^{L+1} \Delta x_1^L(\xi_0)$ dominate the other like the above, leading to divergence. If $a_{L+1} + b_{L+1} = 2a_{L+1} + c$, then in the case of σ-gelu with small σ, $W_0^{L+1} \Delta x_1^L(\xi_0)$ will dominate $\Delta W_1^{L+1} x_1^L(\xi_0)$, as in Theorem N.45; and in the case of tanh, both have the same sign, as in Theorem N.48. In either case, $f_1(\xi_0)$ diverges, so the parametrization is not stable. □

N.8. Putting Everything Together

Finally, in this section we tie all of our insights above to prove our main theorems.

Theorem N.52. *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. A parametrization is stable iff it is pseudostable.*

Proof. The "if" direction is given by Proposition N.25. We now show that when any (in)equality of pseudostability is violated, the parametrization is not stable.

First, if Eq. (42) is not satisfied, then Theorem N.18 shows lack of stability.

Second, if Eq. (42) is satisfied but $r < 0$, then Proposition N.27 shows lack of stability.

Finally, if Eq. (42) is satisfied and $r \geq 0$ but $a_{L+1} + b_{L+1} < 1$ or $2a_{L+1} + c < 1$, then Corollary N.51 or Corollary N.33 shows lack of stability. □

Given this result, we will now just say "stable" instead of "pseudostable" from here on.

Theorem N.8 (Nontriviality Characterization)**.** *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. A stable abc-parametrization is nontrivial iff $a_{L+1} + b_{L+1} + r = 1$ or $2a_{L+1} + c = 1$.*

Proof. The case of $r = 0$ and the case of $r > 0$ are resp. given by Proposition N.49 and Corollary N.32. □

Theorem N.12 (Classification of abc-Parametrizations)**.** *Suppose ϕ is tanh or σ-gelu for sufficiently small σ. Consider a nontrivial stable abc-parametrization of an L-hidden layer MLP. Then*

1. *The following are equivalent to $r = 0$*
   (a) *feature learning*
   (b) *feature learning in the Lth layer*
   (c) *feature kernels evolution*
   (d) *feature kernel evolution in the Lth layer*
   (e) *prefeature learning*
   (f) *prefeature learning in the Lth layer*
   (g) *prefeature kernels evolution*
   (h) *prefeature kernel evolution in the Lth layer*
2. *The following are equivalent to $r > 0$*
   (a) *kernel regime*
   (b) *fixes all features*
   (c) *fixes features in the Lth layer*
   (d) *fixes all feature kernels*
   (e) *fixes feature kernel in the Lth layer*
   (f) *fixes all prefeatures*

(g) *fixes prefeatures in the Lth layer*

(h) *fixes all prefeature kernels*

(i) *fixes prefeature kernel in the Lth layer*

3. *If there is feature learning* or *feature kernel evolution* or *prefeature learning* or *prefeature kernel evolution in layer l, then there is feature learning* and *feature kernel evolution* and *prefeature learning* and *prefeature kernel evolution in layers $l, \ldots, L$.*

4. *If $r = 0$, then for all $\xi \in \mathcal{X}$, $f_0(\xi) \xrightarrow{\text{a.s.}} 0$ and $f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi)$ for some deterministic $\mathring{f}_t(\xi)$. However, the converse is not true.*

5. *If $r > 0$, $a_{L+1} + b_{L+1} + r > 1$ and $2a_{L+1} + c = 1$, then we have the Neural Network-Gaussian Process limit.*

Proof. A nontrivial stable parametrization has either $r = 0$ or $r > 0$. By Theorem N.50, Proposition N.26, and Theorem N.31, $r = 0$ implies all of the statements in (1) and $r > 0$ implies all of the statements in (2). Consequently, if feature learning happens, then clearly r cannot be positive, so r must be 0. Likewise, all of the statements in (1) imply $r = 0$. Symmetrically, all of the statements in (2) about *fixing features* imply $r > 0$. Finally, if the parametrization is in kernel regime, then by Theorem N.50(1), r cannot be 0, so $r > 0$. This proves (1) and (2).

If the premise of (3) holds, then by the above, $r = 0$, so the conclusion follows from Theorem N.50. This proves (3).

If $r = 0$, then nontriviality means $a_{L+1} + b_{L+1} \geq 1$. This implies $f_0(\xi) \xrightarrow{\text{a.s.}} 0$ for all $\xi \in \mathcal{X}$ (more precisely, $f_0(\xi)$ has standard deviation $\Theta(n^{1/2-(a_{L+1}+b_{L+1})}) \to 0$ by Central Limit Theorem). The program describes the unconditional SGD trajectory of f (as opposed to the case when $a_{L+1} + b_{L+1} = 1/2$), so $f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi)$ does not depend on f_0. The converse is not true, for example because of Corollary N.34. This prove (4).

(5) follows from Corollary N.34 (which actually allows much more general ϕ). □

Proofs of Theorems 5.1, F.1 and F.2 For any finite subset $\mathcal{X}$ of the input space $\mathbb{R}^d$ (where $d = 1$ here), we can write out the SGD computation as a Tensor Program like in Appendix N.4. Then the Master Theorem implies the convergence of $f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi)$ for every $\xi \in \mathcal{X}$. Let $\mathcal{X}_1 \subseteq \cdots \subseteq \mathcal{X}_k \subseteq \cdots$ be an infinite chain of finite subsets of $\mathbb{R}^d$ such that $\bigcup_k \mathcal{X}_k$ is a dense subset of $\mathbb{R}^d$. Then the convergence of $f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi)$ holds for every $\xi \in \bigcup_k \mathcal{X}_k$ (because we have almost sure convergence). Finally, we apply a continuity argument to get this convergence for all of $\mathbb{R}^d$: Because ϕ' and thus ϕ are pseudo-Lipschitz, they are locally Lipschitz (i.e. Lipschitz on any compact set). In addition, the operator norms of W^L are almost surely bounded from standard matrix operator bounds. Thus one can see that the Tensor Program is locally Lipschitz in ξ. Consequently, $\mathring{f}_t(\xi)$ is continuous in ξ. This allows to pass from $\bigcup_k \mathcal{X}_k$ to $\mathbb{R}^d$.

Proofs of Propositions E.2, M.2 and E.4 and Theorems M.3 and M.4 follow by dividing into cases of $r > 0$ and $r = 0$ and easy modification of the reasoning in Appendices N.6 and N.7.

Proof of Theorem N.16 follows from straightforward calculations. The basic outline of the calculations is: 1) During pretraining, f's change is purely due to a) the interaction betwen $\Delta W^l, l \leq L$, and W_0^{L+1}, and b) the interaction between x^L and ΔW^{L+1}. 2) When W^{L+1} is re-initialized in g, these interactions are killed. The pretrained $\Delta W^l, l \leq L$, will cause x^M to differ by $\Theta(1/\sqrt{n})$ coordinatewise compared to if $\Delta W^l, l \leq L$, are all reset to 0, but this difference is uncorrelated with the last layer weights W^{M+1} of g, so their interaction is subleading in n, i.e. in the infinite-width limit,

$$g_{T;t}(\xi) - g_{0;t}(\xi) \xrightarrow{\text{a.s.}} 0,$$

whether all of g or just the new weights are trained during fintetuning.