Tensor Programs IV: Feature Learning in Infinite-Width Neural Networks

Greg Yang [1] **Edward J. Hu** [2 3]

Abstract

As its width tends to infinity, a deep neural network's behavior under gradient descent can become simplified and predictable (e.g. given by the Neural Tangent Kernel (NTK)), if it is parametrized appropriately (e.g. the NTK parametrization). However, we show that the standard and NTK parametrizations of a neural network do not admit infinite-width limits that can *learn* features, which is crucial for pretraining and transfer learning such as with BERT. We propose simple modifications to the standard parametrization to allow for feature learning in the limit. Using the *Tensor Programs* technique, we derive explicit formulas for such limits. On Word2Vec and few-shot learning on Omniglot via MAML, two canonical tasks that rely crucially on feature learning, we compute these limits exactly. We find that they outperform both NTK baselines and finite-width networks, with the latter approaching the infinite-width feature learning performance as width increases. See arXiv:2011.14522 for the full version of this paper.

1. Introduction

The study of infinite-width limits of neural networks, in particular the Neural Tangent Kernel (NTK), has recently solved many longstanding open problems on the optimization and generalization of overparametrized neural networks (Jacot et al., 2018). However, in the NTK limit, (last layer) features learned during pretraining are essentially the same as those from random initialization (Corollary 3.8 and Theorem N.12); this is verified empirically in Word2Vec in Fig. 1. As feature learning lies at the core of deep learning's far-ranging impact so far (Brown et al., 2020b; Devlin et al., 2019; He et al., 2016) (e.g. without feature learning, transfer learning via pretraining and finetuning, such as with BERT, is meaningless, as we will discuss in Section 3), this insight amounts to a fatal weakness of the NTK theory as a model of neural networks in practice.

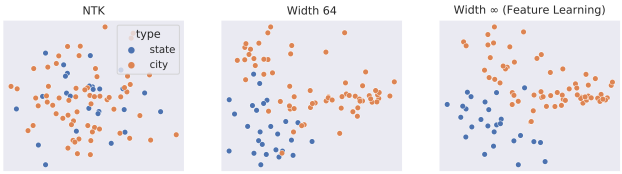

Figure 1. PCA of Word2Vec embeddings of top US cities and states, for NTK, width-64, and width-∞ feature learning networks (Definition 4.1). NTK embeddings are essentially random (thus trivializing any "transfer" learning), while cities and states get naturally separated in embedding space as width increases in the feature learning regime.

Our Contributions We seek to capture feature learning in overparametrized networks by considering other parametrizations and their infinite-width limits. By slightly modifying the standard parametrization (SP), in fact, we can enable feature learning that is *maximal* in a natural sense. We describe how to compute this limit exactly (and rigorously) via the *Tensor Programs* technique developed in (Yang, 2019a;b; 2020a;b). We explicitly calculate this limit for the tasks of Word2Vec (Mikolov et al., 2013a;b) and few-shot learning on Omniglot via MAML (Finn et al., 2017), two standard tasks relying crucially on feature learning. On both tasks, our proposed feature learning limit outperforms both NTK baselines and finite-width networks, with the latter approaching the infinite-width feature learning performance as width increases.

More generally, we classify a natural space of neural network parametrizations that generalizes standard, NTK, and Mean Field parametrizations. We show what we call the *Dynamical Dichotomy Theorem*, which roughly says any parametrization in this space either admits feature learning or has an infinite-width training dynamics given by kernel gradient descent, but not both. Furthermore, any such infinite-width limit can be computed using the Tensor Programs technique.

In the main text, we describe relevent related works as they arise, but see Appendix A for a more comprehensive discussion of the surrounding literature.

[1]Microsoft Research AI [2]Microsoft Dynamics 365 AI [3]Work done partly during the Microsoft AI Residency Program. Correspondence to: Greg Yang <gregyang@microsoft.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

2. abc-Parametrizations

This paper studies a natural class of parametrizations, which we call the *abc-Parametrization* and describe here. Consider an L-hidden-layer perceptron: For weight matrices $W^1 \in \mathbb{R}^{n\times d}$ and $W^2, \dots, W^L \in \mathbb{R}^{n\times n}$, and nonlinearity $\phi : \mathbb{R} \to \mathbb{R}$, such a neural network on input $\xi \in \mathbb{R}^d$ is given by $h^1(\xi) = W^1\xi \in \mathbb{R}^n$, and, for $l = 1, \dots, L-1$,

$$x^l(\xi) = \phi(h^l(\xi)),\ h^{l+1}(\xi) = W^{l+1}x^l(\xi) \in \mathbb{R}^n, \quad (1)$$

and the network output (also called the *logit(s)*) is $f(\xi) = W^{L+1}x^L(\xi)$ for $W^{L+1} \in \mathbb{R}^{1\times n}$. An *abc-parametrization* is specified by a set of numbers $\{a_l, b_l\}_l \cup \{c\}$ such that

(a) We parametrize each weight as $W^l = n^{-a_l}w^l$ for actual trainable parameter w^l

(b) We initialize each $w^l_{\alpha\beta} \sim \mathcal{N}(0, n^{-2b_l})$, and

(c) The SGD learning rate is ηn^{-c} for some width-independent η.[1] [2]

All major prior parametrization schemes, like NTK and standard parametrizations, are of this type, as summarized in Table 1. We can define abc-parametrization and generalize our results to arbitrary neural architectures (Appendix K), but we shall focus on MLPs in the main text.

3. Dynamical Dichotomy

Section Overview For any abc-parametrization, if c is too small (i.e. learning rate too large), SGD can lead to blowup of preactivation and/or logits; we say this parametrization is *unstable*. In practice this translates to numerical issues. If c is too large (i.e. learning rate too small), then the function computed by the network does not change in finite time; we say this parametrization is *trivial*. In this section, we prove what we call the *Dynamical Dichotomy theorem* (Corollary 3.8):

Any nontrivial stable *abc-parametrization yields a (discrete-time) infinite-width limit. This limit either*

1. *allows the embedding $x^L(\xi)$ to evolve nontrivially (Definition 3.4) or*
2. *is described by kernel gradient descent in function space (Definition 3.6), but* not both.

We call the former kind a *feature learning limit* and the latter a *kernel limit*. For 1-hidden-layer MLPs, the former is exemplified by MFP, and the latter, NTP. We demonstrate this via intuitive calculations in Appendix B. This dichotomy implies that certain functional dynamics, such as higher order generalizations of the NTK dynamics, are not valid infinite-width limits (see Remark 3.11). In addition, the neural network function f (defined in Eq. (1)) in any feature learning limit must be identically 0 at initialization (see Corollary 3.9).[3] Finally, we discuss why pretraining and finetuning are trivialized by any kernel limit such as NTK.

In the rest of this section, we formalize the claims above rigorously, under Assumption I.1 (i.e. considering only tanh and smooth relu nonlinearities).

Stable abc-Parametrizations We will only consider abc-parametrizations such that, as $n \to \infty$, 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 $O(1)$ throughout the course of SGD.[4] 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. We call any such parametrization *stable* (see Definition N.4 for a formal definition). Thus unstable parametrizations are of no practical interest.

It turns out stable abc-parametrizations can be characterized by a set of inequalities on $\{a_l, b_l\}_l \cup \{c\}$ (so that the stable ones form a polyhedron). To present these inequalities succinctly, it's useful to have

Definition 3.1. For any abc-parametrization, we define

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

For example, in NTP, $r = 1/2$, while in MFP (when $L = 1$), $r = 0$. Intuitively, r is the exponent such that $\Delta x^L_t(\xi) = \Theta(n^{-r})$ (see Table 1). Thus, to avoid activation blowup, we want $r \geq 0$; to perform feature learning, we want $r = 0$.

Theorem 3.2 (Stability Characterization, c.f. Theorem N.6)**.** *An abc-parametrization is stable iff all of the following are true (with intuitions in parentheses):*

1. *((pre)activations x^l_0, h^l_0 at initialization are $\Theta(1)$ and logits f_0 are $O(1)$)*

$$a_1 + b_1 = 0; \quad a_l + b_l = 1/2,\ \forall l \in [2, L]; \quad a_{L+1} + b_{L+1} \geq 1/2. \quad (2)$$

[1] Observe that by changing a_l, b_l while holding $a_l + b_l$ fixed, we effectively give layer l its own learning rate. As such, a_l and b_l cannot be absorbed into a single number.

[2] One can further include a set of constants in front of n^{-a_l} and n^{-b_l}, for example powers of input dimension d, but we shall keep it simple here as we are only concerned with scaling behavior with n.

[3] We stress this is in the $n \to \infty$ limit, so does not contradict the feature learning seen in finite-width SP NN.

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

Table 1. SP (standard), NTP (Neural Tangent), MFP (Mean Field, for 1-hidden-layer nets), μP (Maximal Update, ours) as abc-parametrizations. We show the minimal value of c such that the parametrization is stable (Definition N.4). We also list the quantities $r, 2a_{L+1}+c, a_{L+1}+b_{L+1}+r$ involved in stability, feature learning, and kernel regime properties of the parametrizations. Here we only focus on scaling with n and ignore dependence on input dimension. Recall the MLP definition:

$$h^1 = W^1\xi \in \mathbb{R}^n, x^l = \phi(h^l) \in \mathbb{R}^n, h^{l+1} = W^{l+1}x^l \in \mathbb{R}^n, f(\xi) = W^{L+1}x^L$$

	Definition	SP	SP (w/ LR $\frac{1}{n}$)	NTP	MFP ($L=1$)	μP (ours)
a_l	$W^l = n^{-a_l}w^l$	0	0	$\begin{cases}0 & l=1\\ 1/2 & l\geq 2\end{cases}$	$\begin{cases}0 & l=1\\ 1 & l=2\end{cases}$	$\begin{cases}-1/2 & l=1\\ 0 & 2\leq l\leq L\\ 1/2 & l=L+1\end{cases}$
b_l	$w^l_{\alpha\beta}\sim\mathcal{N}(0,n^{-2b_l})$	$\begin{cases}0 & l=1\\ 1/2 & l\geq 2\end{cases}$	$\begin{cases}0 & l=1\\ 1/2 & l\geq 2\end{cases}$	0	0	$1/2$
c	$LR=\eta n^{-c}$	0	1	0	-1	0
r	Definition 3.1	0	$1/2$	$1/2$	0	0
$2a_{L+1}+c$		0	1	1	1	1
$a_{L+1}+b_{L+1}+r$		$1/2$	1	1	1	1
Nontrivial?		✓	✓	✓	✓	✓
Stable?		✗	✓	✓	✓	✓
Feature Learning?		-	✗	✗	✓	✓
Kernel Regime?		-	✓	✓	✗	✗

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

$$r \geq 0. \tag{3}$$

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

$$2a_{L+1}+c \geq 1; \quad a_{L+1}+b_{L+1}+r \geq 1. \tag{4}$$

Nontrivial abc-Parametrizations Among stable abc-parametrizations, there are also those where f does not change throughout training in the infinite-width limit. We say such parametrizations are *trivial*. Our dichotomy result will only apply to nontrivial stable abc-parametrizations.[5]

Nontrivial abc-parametrizations can also be described by a disjunction of equations on $\{a_l, b_l\}_l \cup \{c\}$ (geometrically, they correspond to the union of two faces on the polyhedron of stable abc-parametrizations).

Theorem 3.3. *A stable abc-parametrization is nontrivial iff $a_{L+1}+b_{L+1}+r = 1$ or $2a_{L+1}+c = 1$.*

Feature Learning Below, for brevity, we say *training routine* to mean the package of learning rate ηn^{-c}, training sequence $\{(\xi_t, y_t)\}_{t\geq 0}$,[6] and a loss function $\mathcal{L}(f(\xi), y)$ that is continuously differentiable in the prediction of the model $f(\xi)$. As above, we use $\bullet_t$ to denote the object $\bullet$ after t steps of SGD.

Definition 3.4 (c.f. Definitions N.9 and N.10)**.** We say an abc-parametrization *admits feature learning* (resp. *evolves the feature kernel*) if, as $n\to\infty$, $\Delta x^L_t(\xi)$ has $\Omega(1)$ coordinates (resp. $\frac{1}{n}(x^L_t(\xi)^\top x^L_t(\zeta) - x^L_0(\xi)^\top x^L_0(\zeta)) = \Omega(1)$) for some training routine, time $t\geq 1$, and input ξ (resp. ξ,ζ).[7]

MFP, in the 1-hidden-layer case, is an example of feature learning parametrization.

Intuitively, feature kernel evolution implies feature learning, but *a priori* it seems possible that the latter can occur without the former (akin to some kind of rotation of features). If so, then, e.g. in terms of linear transfer learning (c.f. discussion below Remark 3.11), the pretraining ultimately had no benefit. But, in fact,

Theorem 3.5. *A nontrivial stable abc-parametrization admits feature learning iff it evolves the feature kernel iff $r = 0$.*

Kernel Regime While feature learning here is defined by looking at the embedding of an input ξ, we can also look at the dynamics of the *function* represented by the neural network.

Definition 3.6 (c.f. Definition N.11)**.** We say an abc-

[5] In particular, it's possible for the function f to stay fixed with time, but for the features to change.

[6] For simplicity, we only consider batch size 1; it's straightforward to generalize to larger batch sizes.

[7] For the sake of streamlining the main text presentation, we defined feature learning and feature kernel evolution slightly differently than in Definition N.9, but ultimately they are equivalent as a result of our theorems.

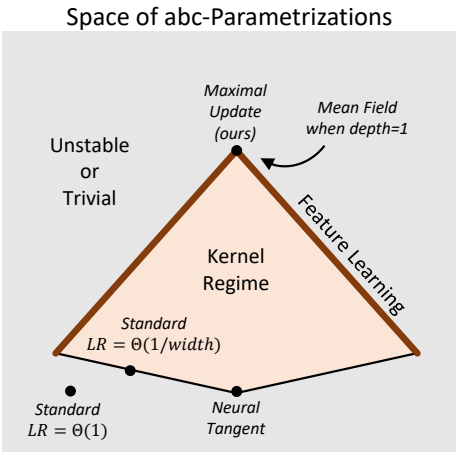

Figure 2. **A Caricature of abc-Parametrizations.** The nontrivial stable parametrizations form a high dimensional polyhedron. Those on a part of its boundary admit feature learning, while all others are in kernel regime. μP is a vertex in the former, while NTP, latter. See Fig. 7 for a more geometrically accurate depiction.

parametrization *is in kernel regime* if there exists a positive semidefinite kernel K such that, for any training routine, time $t \geq 0$, and input ξ, in the $n \to \infty$ limit,

$$f_{t+1}(\xi) = f_t(\xi) - \eta K(\xi, \xi_t)\mathcal{L}'(f_t(\xi_t), y_t), \quad \forall t \geq 0. \quad (5)$$

In other words, SGD reduces to kernel gradient descent in the large n limit.

Theorem 3.7. *A nontrivial stable abc-parametrization is in kernel regime iff $r > 0$.*

NTP is a typical example of this, where $r = 1/2$ and K is given by the NTK.

Dynamical Dichotomy Since a stable abc-parametrization has either $r = 0$ or $r > 0$ by Eq. (3):

Corollary 3.8. *A nontrivial stable abc-parametrization either admits feature learning or is in kernel regime, but not both.*

We note that we are under Assumption I.1. For example, if ϕ is linear, then this dichotomy doesn't hold, as a 1-hidden-layer linear network where only the first layer is trained would both admit feature learning and is in kernel regime.

An interesting consequence of Dynamical Dichotomy is

Corollary 3.9. *Any nontrivial stable* feature learning *abc-parametrization must have $\lim_{n\to\infty} f_0(\xi) = 0$ for all ξ, where the limit is almost sure.*

Theorems 3.5 and 3.7 and Corollary 3.9 are consequences of the more general classification theorem Theorem N.12, which in addition shows: 1) feature learning in layer l would imply the same for layers $l, \ldots, L$; 2) in any feature learning parametrization, f_t in the large n limit becomes deterministic, and thus is incompatible with any Bayesian perspective (in contrast to the NNGP limit).

Dynamical Dichotomy in the shallow perceptron case is illustrated by the NTK and MF limits, as presented in Appendix B, which shows the NTK limit exemplifies Theorem 3.7 while the MF limit exemplifies Theorem 3.5. We present a simplified picture of abc-parametrizations in Fig. 2, but see Fig. 7 for a more geometrically accurate depiction.

Remark 3.10 (Function Space Picture)**.** A kernel regime limit resides solely in the *function space picture*, i.e. the evolution of f at any time being solely determined by the function values $\{\lim f_t(\zeta)\}_\zeta$ themselves (as opposed to the internal activations of f as well) along with η, $\mathcal{L}$, and (ξ_t, y_t). Intuitively, this cannot be true for the feature learning limit, and therefore, at least informally, Dynamical Dichotomy is also a dichotomy over the sufficiency of the function space picture for determining the training evolution: We can construct two settings where $\{\lim f_t(\zeta)\}_\zeta$, η, $\mathcal{L}$, and (ξ_t, y_t) are the same but f_{t+1} are different. 1) The first setting is at $t = 0$, where $\lim f_t(\zeta) = 0$ for all input ζ by Corollary 3.9. Here a typical SGD will change f. 2) In the second setting, suppose ϕ is relu. Design a sequence of inputs such that training the MLP on them with very large learning rate will make all relu neurons saturated in the 0 region. Then f is everywhere 0, and an SGD step will not change that.

Remark 3.11 (Not All Dynamics are Infinite-Width Limits)**.** Accordingly, a nonlinear function space dynamics cannot be a valid infinite-width limit of some abc-parametrization. By *nonlinear*, we mean $f_{t+1}(\xi) - f_t(\xi)$ is nonlinear in $\mathcal{L}'(f_t(\xi_t), y_t)$. For example, any natural higher-order generalization of Eq. (5) (perhaps derived from a Taylor expansion at initialization) is not a valid limit.[8]

Pretraining and Transfer Learning By Corollary 3.8, any kernel regime parametrization trivializes pretraining and transfer learning in the infinite-width limit. For example, consider *linear transfer learning,* the popular style of transfer learning where one discards the pretrained linear classifier layer and train a new one on top of the features, 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, as is the case for

[8] It may seem that Neural Tangent Hierarchy (Huang & Yau, 2019), which allow some kind of higher order dynamics in the function space, violates our observation. But their infinite-width limit is identical to NTK in the constant time $t = O(1)$ regime, which is what Remark 3.11 (and this paper) concerns. Moreover, here we are talking about functional dynamics that doesn't depend on n (because we are already at the $n \to \infty$ limit) whereas their functional dynamics does.

kernel regime limits (by Corollary 3.8 and Theorem 3.5), 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.

4. Maximal Update Parametrization

By calculating r for the standard parametrization (SP) (c.f. Table 1), we easily see that it cannot admit feature learning in the sense here without becoming unstable. Intuitively, this is because of an imbalance of gradient and parameter sizes in SP: The last layer weight matrix is too large at init and gets too much gradient, so that the learning rate needs to be small ($O(1/n)$) in order for SGD not to blow up – but then the features of the network don't change enough. We can fix this by dividing the logits by $\sqrt{n}$ and using $\Theta(1)$ learning rate, so that $f(\xi) = n^{-1/2} w^{L+1} x^L(\xi)$ (where w^{L+1} has $\mathcal{N}(0, 1/n)$ initialization, as usual). This suffices to enable feature learning.

Another minor problem of SP is that the input layer gets too little gradient compared to other layers. We can fix this by multiplying the first layer preactivation by $\sqrt{n}$ and use fan-out initialization for w^1, i.e. $h^1(\xi) = \sqrt{n} w^1 \xi$ where w^1 has initialization $\mathcal{N}(0, 1/n)$.

We can summarize these modifications as follows:

Definition 4.1. The *Maximal Update Parametrization (abbreviated MUP, or μP),* in the context of an L-hidden-layer MLP (Eq. (1)), is given by

$$c = 0, \quad b_l = 1/2 \; \forall l, \quad a_l = \begin{cases} -1/2 & l = 1 \\ 0 & 2 \le l \le L \\ 1/2 & l = L + 1. \end{cases}$$

Readers who find the abc values too abstract are recommended to see Appendices D and E for pedagogical examples working out why SP cannot do feature learning and how μP fixes this problem.

How is μP Maximal? Recall $\Delta \bullet_t$ means $\bullet_t - \bullet_0$ for any object $\bullet$. Simple arithmetic shows the NN function changes like $\Delta f_t(\xi) = \Delta W_t^{L+1} x_t^L(\xi) + W_0^{L+1} \Delta x_t^L(\xi)$ over the course of SGD. μP turns out to be the *unique* stable abc-parametrization where both contributions to $\Delta f_t(\xi)$ are $\Theta(1)$ and where *every layer* learns features (in the sense that $\Delta W_t^l x_t^{l-1}$ has $\Theta(1)$ coordinates for every $l \in [L+1]$). See Appendix E.3 for a formalization of this statement.

MFP is a Special Case of μP Different abc values can actually correspond to equivalent parametrizations, intuitively because adjusting a_l, b_l values effectively gives W^l its own learning rate, so that the global learning rate c is algebraically redundant; this is made rigorous in Appendix C.1. It turns out, in 1-hidden-layer networks, MFP is equivalent to μP for this reason, which is reflected in Fig. 2.

5. Deriving Feature Learning Infinite-Width Limit: Intuition and Examples

We propose *the Tensor Programs technique* for deriving the infinite-width limit of any abc-parametrization. This ultimately just requires the researcher to mechanically apply a set of rules to the computation graph underlying SGD. However, while operationally simple, this procedure would seem "too magical" at first. In the main text, we seek to build some intuition for what is being automated by this procedure. Then, in Appendix G, we formally describe the Tensor Programs framework.

Setup and Notation For pedagogical simplicity, we only consider input dimension $d = 1$ and learning rate $\eta = 1$ here, but generalization to $d > 1, \eta \neq 1$ is straightforward. We consider SGD with a singleton minibatch $\{(\xi_t, y_t)\}$ at time $t = 0, 1, 2, \ldots$, where ξ_t is the network input and y_t is the label. We write W_t^l for the matrix W^l after t steps of such training. For any network input $\xi \in \mathbb{R}$, we write $x_t^l(\xi)$ (resp. $h_t^l(\xi)$, $f_t(\xi)$) for the activation x^l (resp. preactivation h^l, logits f) of the network after t steps of SGD. We denote the scaled gradient $n \nabla_{x_t^l} f_t(\xi)$ (resp. $n \nabla_{h_t^l} f_t(\xi)$) by $dx_t^l(\xi)$ (resp. $dh_t^l(\xi)$). For brevity, we abuse notation and use x_t^l (without being applied to ξ) to also denote the vector $x_t^l(\xi_t)$ (applied specifically to ξ_t); likewise for $h_t^l, dh_t^l, dx_t^l, f_t$. We will not use x_t^l on its own to denote the function $\xi \mapsto x_t^l(\xi)$ so this should not cause confusion. The loss function is denoted $\mathcal{L}$ and the loss derivative $\mathcal{L}'(logit, target)$ is in the first argument. We write $\chi_t \stackrel{\text{def}}{=} \mathcal{L}'(f_t, y_t)$.

5.1. 1-Hidden-Layer MLP

As mentioned above, for 1 hidden layer, the infinite-width μP limit is the same as the mean field limit of (Chizat & Bach, 2018; Mei et al., 2018; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2018). Nevertheless, we

present a slightly different derivation of this that is more consistent with the philosophy of Tensor Programs. Such a network on input $\xi \in \mathbb{R}$ is given by

$$f(\xi) = Vx(\xi), \quad x(\xi) = \phi(h(\xi)), \quad h(\xi) = U\xi, \quad (6)$$

for $U \in \mathbb{R}^{n\times 1}, V \in \mathbb{R}^{1\times n}$ parametrized like $U = \sqrt{n}u, V = \frac{1}{\sqrt{n}}v$ and with initialization $u_{\alpha\beta}, v_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$.[9] Then U_0 (the initial value of U) has iid $\mathcal{N}(0,1)$ coordinates. It will turn out to be convenient to represent each such coordinate distribution as a random variable $Z^{U_0} \overset{\text{def}}{=} \mathcal{N}(0,1)$. Likewise, let $Z^{nV_0} \overset{\text{def}}{=} \mathcal{N}(0,1)$, independent from Z^{U_0}, represent the coordinate distribution of nV_0 (we do nV_0 instead of V_0 so that the Z random variable is always independent of n). We derive the μP limits of the first forward and backward passes manually before stating the general case (Theorem 5.1). To lighten notation, we suppress the $t = 0$ subscript (e.g. $U = U_0, h = h_0, f = f_0$, etc), as we will spend some time on the first SGD step.

First Forward Pass After randomly initialization, the preactivation $h = h(\xi)$ (where $\xi = \xi_0 \in \mathbb{R}$ is the first input) has iid coordinates, each a sample from $Z^h \overset{\text{def}}{=} \xi Z^U \in \mathbb{R}$. Naturally, $x = x(\xi)$ has iid coordinates as well, each a sample from $Z^x \overset{\text{def}}{=} \phi(Z^h)$. Finally, $f = Vx = \frac{1}{n}\sum_{\alpha=1}^n (nV)_\alpha x_\alpha \to \mathring{f} \overset{\text{def}}{=} \mathbb{E}\, Z^{nV} Z^x$ by Law of Large Numbers as $n \to \infty$.[10] In particular, f becomes deterministically 0 in this limit because V and U are independent. For a typical loss function $\mathcal{L}$, the loss derivative $\chi \overset{\text{def}}{=} \mathcal{L}'(f, y)$ then also become deterministic, $\chi \to \mathring{\chi} \overset{\text{def}}{=} \mathcal{L}'(\mathring{f}, y)$.

First Backward Pass Similarly, $dx = nV^\top$ (recall $dx_t \overset{\text{def}}{=} n\nabla_{x_t} f_t$) has coordinates distributed like $Z^{dx} \overset{\text{def}}{=} Z^{nV}$ and $dh = dx \odot \phi'(h)$ has coordinates distributed like $Z^{dh} \overset{\text{def}}{=} Z^{dx}\phi'(Z^h) = Z^{nV}\phi'(Z^h)$. Then SGD with learning rate 1 makes the following updates:

$$v_1 = v - \chi x/\sqrt{n} \implies V_1 = V - \chi x/n$$
$$u_1 = u - \chi\xi\, dh/\sqrt{n} \implies U_1 = U - \chi\xi\, dh.$$

Since χ converges to a deterministic limit $\mathring{\chi}$, the coordinates of these updates are roughly iid, corresponding to an update of Z random variables:

$$Z^{nV_1} = Z^{nV} - \mathring{\chi} Z^x, \quad Z^{U_1} = Z^U - \mathring{\chi}\xi Z^{dh}.$$

Second Forward Pass Thus V_1 and U_1 still have roughly iid coordinates after 1 SGD step. Then, in the second forward pass, h_1 has coordinates

$$Z^{h_1} \overset{\text{def}}{=} \xi_1 Z^{U_1} = \xi_1 Z^U - \xi_1\mathring{\chi}\xi Z^{dh} = \xi_1 Z^U - \xi_1\mathring{\chi}\xi Z^{nV}\phi'(Z^h),$$

x_1 has coordinates $Z^{x_1} \overset{\text{def}}{=} \phi(Z^{h_1})$, and the output is

$$f_1 = \frac{1}{n}\sum_{\alpha=1}^n (nV_1)_\alpha x_\alpha \quad (7)$$
$$\to \mathring{f}_1 \overset{\text{def}}{=} \mathbb{E}\, Z^{nV_1} Z^{x_1} = \mathbb{E}(Z^{nV} - \mathring{\chi} Z^x) Z^{x_1}$$

as $n \to \infty$. Then $\chi_1 \overset{\text{def}}{=} \mathcal{L}'(f_1, y_1) \to \mathring{\chi}_1 \overset{\text{def}}{=} \mathcal{L}'(\mathring{f}_1, y_1)$ becomes deterministic. The gradient vectors have roughly iid coordinates by a similar logic.

***t*th Iteration** Repeating the above reasoning shows that at any time t (independent of n), we obtain

Theorem 5.1. *Consider a 1-hidden-layer MLP in μP (Eq. (6)) and any training routine with learning rate 1. Suppose ϕ' is pseudo-Lipschitz.[11] As $n \to \infty$, for every input ξ, $f_t(\xi)$ converges almost surely to $\mathring{f}_t(\xi)$ defined as follows:*

$$f_t(\xi) \xrightarrow{\text{a.s.}} \mathring{f}_t(\xi) \overset{def}{=} \mathbb{E}\, Z^{nV_t} Z^{x_t(\xi)}, \quad (8)$$
$$Z^{x_t(\xi)} \overset{def}{=} \phi(Z^{h_t(\xi)}), \quad Z^{h_t(\xi)} \overset{def}{=} \xi Z^{U_t},$$
$$\mathring{\chi}_t \overset{def}{=} \mathcal{L}'(\mathring{f}_t, y_t), \quad Z^{nV_{t+1}} \overset{def}{=} Z^{nV_t} - \mathring{\chi}_t Z^{x_t}, \quad (9)$$
$$Z^{U_{t+1}} \overset{def}{=} Z^{U_t} - \mathring{\chi}_t \xi_t Z^{nV_t}\phi'(Z^{h_t}),$$

with, as initial conditions, Z^{U_0} and Z^{nV_0} being independent standard Gaussians, where in Eq. (9) we abbreviated $\mathring{f}_t = \mathring{f}_t(\xi_t), x_t = x_t(\xi_t), h_t = h_t(\xi_t)$.

As aforementioned, this is a discrete time, minibatched version of the mean field limit of (Chizat & Bach, 2018; Mei et al., 2018; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2018).[12] When ϕ is identity, it's easy to see that Z^{nV_t} and Z^{U_t} are always (deterministic) linear combinations of Z^{nV_0} and Z^{U_0}, say $Z^{nV_t} = A_t Z^{nV_0} + B_t Z^{U_0}$ and $Z^{U_t} = C_t Z^{nV_0} + D_t Z^{U_0}$. Then the limit $\mathring{f}_t$ depends solely on A_t, B_t, C_t, D_t. By tracking their evolution, we get the following greatly simplified formula for an infinite-width μP linear network.

Corollary 5.2. *Consider a 1-hidden-layer* linear *MLP in μP (Eq. (6)) and any training routine with learning rate 1. As $n \to \infty$, for every input ξ, $f_t(\xi)$ converges almost surely*

[9] Again, more generally, we can insert constants in this parametrization, like $U = \frac{\sqrt{n}}{\sqrt{d}}u$, but we omit them here for simplicity.

[10] All convergence in this section will be almost sure, but to focus on the intuition here and less on the formalities, we do not explicitly write this down.

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

[12] (Chizat & Bach, 2018; Mei et al., 2018; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2018) present the equations in terms of the PDF of Z random variables. Formally, the PDF limit can be obtained by taking the continous-time limit of Eqs. (8) and (9) and then applying Focker-Planck. Note our derivation, when formalized using the Tensor Programs framework below, does not require smoothness and support assumptions on the initialization of U, V in those works: The initialization distribution here can be replaced with any image of Gaussians under pseudo-Lipschitz functions, which includes nonsmooth and singular distributions.

to $\mathring{f}_t(\xi)$ defined as follows:

$$\mathring{f}_t(\xi) = (A_tC_t + B_tD_t)\xi, \quad \mathring{\chi}_t = \mathcal{L}'(\mathring{f}_t, y_t),$$
$$(A_{t+1}, B_{t+1}) = (A_t, B_t) - \mathring{\chi}_t\xi_t(C_t, D_t),$$
$$(C_{t+1}, D_{t+1}) = (C_t, D_t) - \mathring{\chi}_t\xi_t(A_t, B_t),$$

with initial condition $A_0 = D_0 = 1, B_0 = C_0 = 0$.

This can be easily generalized to larger input and output dimenions (see Appendix J.2). In a gist, such an infinite-width μP linear network with input dimension d and output dimension d_o is equivalent to a width-$(d+d_o)$ linear network with the same input/output dimensions but a "diagonal", instead of random, initialization. Our Word2Vec and MAML experiments will crucially rely on this simplifying observation. We remark that, in contrast to our approach, such an observation would be obscured by the PDE perspective of prior works (Chizat & Bach, 2018; Mei et al., 2018; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2018).

5.2. Deep MLP: A Summary

For more than 1 hidden layer, the mathematics of the μP limit becomes significantly more complicated. For the lack of space and because our main experiments only use the 1-hidden-layer limit, we will just summarize the key points here for the deep case, but see Appendix F for pedagogical examples that are worked out completely.

Compared to the shallow case, a deep MLP involves $n \times n$ random Gaussian matrices corresponding to the initialization of middle layer weights. Such a matrix W has a unique set of behaviors not seen in the weights of the shallow MLP, where only one dimension tends to infinity. 1) (*Gaussian behavior*) If x is roughly independent from W, then Wx will be a Gaussian vector with roughly iid coordinates. This intuition should be familiar to readers having seen NTK or NNGP calculations. 2) (*Correlation with $W^\top$*) In a typical NTK calculation, $W^\top$ can be safely assumed to be independent from W (i.e. Gradient Independence Assumption (Yang, 2020a)). This turns out to be wrong if we unroll SGD for more than 1 step, so one must take care to keep track of the contributions from W and $W^\top$ separately so as to calculate their interactions that arise over the course of SGD.

The *Tensor Programs* framework neatly and rigorously packages such complex calculations into a mechanical algorithm generalizing the calculus of Z random variables in Section 5.1; see Appendix F. This framework can calculate the limit of any abc-parametrization; see Appendix N.

6. Experiments

We present our main experiments (Omniglot and Word2Vec) in the main text, while we also empirically verified the validity of our infinite-width theory in various toy settings in Appendix J.1.

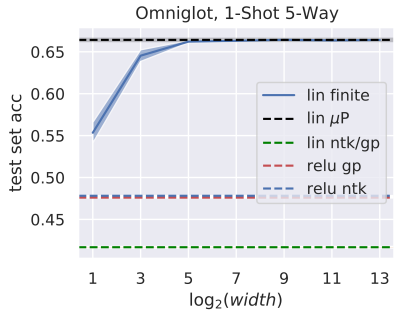

Figure 3. Our Omniglot Results

6.1. Few-Shot Learning on Omniglot via First Order MAML

In few-shot learning, the model is given only a small number of labeled examples before asking to make predictions on unseen data. Therefore, this tests whether a model contains a good *prior* that can adapt quickly to the small amount of data at hand. We compare finite- and infinite-width models on Omniglot (Lake et al., 2015), a standard few-shot learning benchmark, via First Order MAML (Finn et al., 2017), a standard few-shot learning algorithm using metalearning. We will be concerned with the 1-shot, 5-way classification task (i.e. there are 5 labels, and the training set consists of 1 example for each label). Readers needing a refresher on few-shot learning and MAML can see Appendix J.2.

Models Our main model is the μP limit of a 1-hidden-layer linear MLP. We compare against: 1) finite width versions of the same;[13] 2) the NNGP and NTK limits of the same; 3) the NNGP and NTK limits of a 1-hidden-layer relu MLP. Note 2) is equivalent to a 0-hidden-layer perceptron, because the NNGP and NTK there are both linear kernels. In addition, the infinite-width SP limit of a 1-hidden-layer network is the same as the NNGP limit. Both 2) and 3) are equivalent to linear models with fixed (not learned) features, so MAML's adaptation only applies to the linear weights. On the other hand, the μP limit and the finite μP networks will learn new representations of the data over time that can quickly adapt to new tasks.[14]

The hyperparameters for our experiments can be found in Appendix J.2.

[13] Because we will tune initialization variances, our results also represent finite-width SP networks.

[14] Note that the transfer learning comment in Appendix B does not apply directly to the few-shot setting here, because the readout weights of the network carry over from the pretraining phase. Nevertheless, we will see a large performance gap between the kernel limits (2,3) and the μP limit.

Table 2. **Omniglot Meta-Test Accuracies after Pretraining with First Order MAML.**

$\phi = $ relu		$\phi = $ identity ; number = $\log_2 width$								
GP	NTK	1	3	5	7	9	11	13	μP	GP/NTK
47.60 ±.02	47.82 ±.04	55.34 ±1.24	64.54 ±0.70	66.21 ±.15	66.31 ±.16	66.43 ±.23	66.36 ±.22	66.41 ±.18	66.42 ±.19	41.68 ±.09

Findings Our results are summarized in the Fig. 3 and Table 2, where curves indicate means and shades indicate standard deviations. There are three key takeaways: 1) The feature learning μP limit significantly outperforms the kernel limits. 2) The benefit of feature learning dominates the benefit of having nonlinearities. 3) As width increases, the finite μP networks approach the performance of the μP limit from below.

6.2. Word2Vec

Word2Vec (Mikolov et al., 2013a;b) is an early example of large-scale pretraining and transfer learning in natural language processing, where one learns an feature vector (i.e. embedding) for every word based on the principle of distributional semantics. Here we will focus on the Context as a Bag-of-Words (CBOW) method of Word2Vec. During pretraining, we seek to learn word embeddings such that for a typical sentence (in the training corpus), a word's embedding should be close to the average of the embeddings of those in its surrounding context. We evaluate the learned embeddings on word analogy, which asks questions of the kind "what to a queen is as a man to a woman?" To answer such a question, we would calculate the linear combination of embeddings $man - woman + queen$ and return the word whose embedding is closest to it. We pretrain our models on two standard datasets, `text8` and `fil9`. For a more thorough review of Word2Vec and a description of the datasets, see Appendix J.3.

Models Our main model is the μP limit of Word2Vec.[15] We compare against the baselines of 1) finite-width versions of the same, and 2) the NTK and GP limits. As shown in Corollary 3.8, the features of the NTK limit are fixed at initialization as $n \to \infty$ (and so are those of the GP limit, by definition), so its answer to Eq. (37) is uniformly selected from the whole vocabulary.[16] Its accuracy is thus $\frac{1}{|\mathcal{V}|-3}$, where $|\mathcal{V}|$ is the vocabulary size. Since $|\mathcal{V}|$ is 71,291 for `text8` and 142,276 for `fil9`, this number is practically 0. We compute the μP limit according to the generalized version of Corollary 5.2, but we relate more implementation details in Appendix J.3.

[15] More precisely, the μP limit of Eq. (35) in Appendix J.3.

[16] There is some nuance here because $h(\xi)^\top h(\bar{\xi})$ is actually $\Theta(\sqrt{n})$ instead of $\Theta(n)$ because $\xi, \bar{\xi}$ are one-hot, but the conclusion is the same; see Appendix J.3.

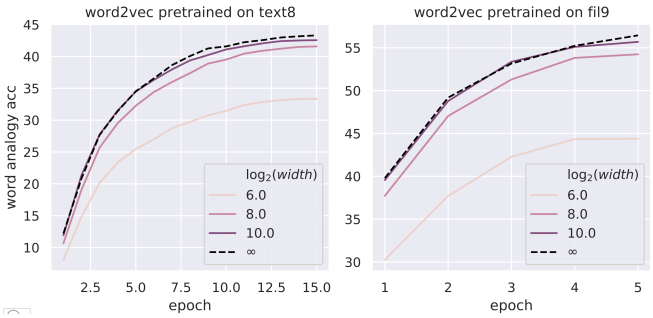

Figure 4. Our Word2Vec Results. The accuracies of the GP/NTK models are practically 0, so we omit them in these plots.

Table 3. **Test Accuracies on Word Analogy after Pretraining with CBOW Word2Vec.**

	number = $\log_2 width$				
Dataset	6	8	10	μP	GP/NTK
`text8`	33.35	41.58	42.56	**43.31**	0.0
`fil9`	44.39	54.24	55.69	**56.45**	0.0

Findings We show our results in Table 3 and Fig. 4. As expected, the infinite-width and finite-width μP networks significantly outperform the NTK limit. In addition, we observe the finite width μP networks converge to the performance of the μP limit from below, as width increases.

7. Conclusion

In this paper, we presented a framework, based on the notion of *abc-parametrizations* and *Tensor Programs* technique, that unifies the Neural Tangent Kernel (NTK) and Mean Field limits of large width neural networks (NNs). In the Dynamical Dichotomy theorem, we classified the abc-parametrizations into feature learning and kernel regimes. We identified the lack of feature learning as a fatal weakness of NTK as a model for real NN. In fact, we showed the standard parametrization suffers from the same problem. As a solution, we proposed the Maximal Update Parametrization (μP) and derived its infinite-width limit, which admits feature learning. Through experiments on Word2Vec and few-shot learning, we demonstrated that μP is a good model for feature learning behavior in neural networks.

References

Aitchison, L. Why bigger is not always better: on finite and infinite neural networks. *arXiv:1910.08013 [cs, stat]*, June 2020. URL http://arxiv.org/abs/1910.08013.

Aitchison, L., Yang, A. X., and Ober, S. W. Deep kernel processes. *arXiv:2010.01590 [cs, stat]*, October 2020. URL http://arxiv.org/abs/2010.01590.

Allen-Zhu, Z., Li, Y., and Song, Z. A Convergence Theory for Deep Learning via Over-Parameterization. *arXiv:1811.03962 [cs, math, stat]*, November 2018. URL http://arxiv.org/abs/1811.03962.

Araújo, D., Oliveira, R. I., and Yukimura, D. A mean-field limit for certain deep neural networks. *arXiv:1906.00193 [cond-mat, stat]*, June 2019. URL http://arxiv.org/abs/1906.00193.

Bayati, M. and Montanari, A. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Transactions on Information Theory*, 57 (2):764–785, February 2011. ISSN 0018-9448, 1557-9654. doi: 10.1109/TIT.2010.2094817. URL http://arxiv.org/abs/1001.3448.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020a. URL http://arxiv.org/abs/2005.14165.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020b. URL http://arxiv.org/abs/2005.14165.

Chizat, L. and Bach, F. A Note on Lazy Training in Supervised Differentiable Programming. pp. 19.

Chizat, L. and Bach, F. On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport. *arXiv:1805.09545 [cs, math, stat]*, May 2018. URL http://arxiv.org/abs/1805.09545.

Chizat, L. and Bach, F. Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. *arXiv:2002.04486 [cs, math, stat]*, June 2020. URL http://arxiv.org/abs/2002.04486.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv: 1810.04805 version: 2.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient Descent Provably Optimizes Over-parameterized Neural Networks. *arXiv:1810.02054 [cs, math, stat]*, October 2018. URL http://arxiv.org/abs/1810.02054.

Fang, C., Lee, J. D., Yang, P., and Zhang, T. Modeling from Features: a Mean-field Framework for Over-parameterized Deep Neural Networks. *arXiv:2007.01452 [cs, math, stat]*, July 2020. URL http://arxiv.org/abs/2007.01452. arXiv: 2007.01452.

Finn, C., Abbeel, P., and Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400 [cs]*, July 2017. URL http://arxiv.org/abs/1703.03400.

Gilboa, D. and Gur-Ari, G. Wider Networks Learn Better Features. September 2019. URL https://arxiv.org/abs/1909.11572v1.

Golikov, E. A. Dynamically Stable Infinite-Width Limits of Neural Classifiers. *arXiv:2006.06574 [cs, stat]*, October 2020. URL http://arxiv.org/abs/2006.06574.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. pp. 770–778, 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Hendrycks, D. and Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv:1606.08415 [cs]*, July 2020. URL http://arxiv.org/abs/1606.08415.

Huang, J. and Yau, H.-T. Dynamics of deep neural networks and neural tangent hierarchy, 2019.

Jacot, A., Gabriel, F., and Hongler, C. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. *arXiv:1806.07572 [cs, math, stat]*, June 2018. URL http://arxiv.org/abs/1806.07572.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic

program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL https://science.sciencemag.org/content/350/6266/1332.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. The large learning rate phase of deep learning: the catapult mechanism. *arXiv:2003.02218 [cs, stat]*, March 2020. URL http://arxiv.org/abs/2003.02218.

Li, Y., Ma, T., and Zhang, H. R. Learning over-parametrized two-layer neural networks beyond ntk. In Abernethy, J. and Agarwal, S. (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 2613–2682. PMLR, 09–12 Jul 2020a. URL http://proceedings.mlr.press/v125/li20a.html.

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., and Gonzalez, J. E. Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers. *arXiv:2002.11794 [cs]*, June 2020b. URL http://arxiv.org/abs/2002.11794.

Mei, S., Montanari, A., and Nguyen, P.-M. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33): E7665–E7671, August 2018. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1806579115. URL https://www.pnas.org/content/115/33/E7665.

Mei, S., Misiakiewicz, T., and Montanari, A. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. *arXiv:1902.06015 [cond-mat, stat]*, February 2019. URL http://arxiv.org/abs/1902.06015.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, September 2013a. URL http://arxiv.org/abs/1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *arXiv:1310.4546 [cs, stat]*, October 2013b. URL http://arxiv.org/abs/1310.4546.

Nguyen, P.-M. Mean Field Limit of the Learning Dynamics of Multilayer Neural Networks. *arXiv:1902.02880 [cond-mat, stat]*, February 2019. URL http://arxiv.org/abs/1902.02880.

Nguyen, P.-M. and Pham, H. T. A Rigorous Framework for the Mean Field Limit of Multilayer Neural Networks. *arXiv:2001.11443 [cond-mat, stat]*, January 2020. URL http://arxiv.org/abs/2001.11443.

Nichol, A., Achiam, J., and Schulman, J. On First-Order Meta-Learning Algorithms. March 2018. URL https://arxiv.org/abs/1803.02999v3.

Rotskoff, G. M. and Vanden-Eijnden, E. Neural Networks as Interacting Particle Systems: Asymptotic Convexity of the Loss Landscape and Universal Scaling of the Approximation Error. *arXiv:1805.00915 [cond-mat, stat]*, May 2018. URL http://arxiv.org/abs/1805.00915.

Sirignano, J. and Spiliopoulos, K. Mean Field Analysis of Neural Networks. *arXiv:1805.01053 [math]*, May 2018. URL http://arxiv.org/abs/1805.01053.

Sirignano, J. and Spiliopoulos, K. Mean Field Analysis of Deep Neural Networks. *arXiv:1903.04440 [math, stat]*, February 2020. URL http://arxiv.org/abs/1903.04440.

Sohl-Dickstein, J., Novak, R., Schoenholz, S. S., and Lee, J. On the infinite width limit of neural networks with a standard parameterization. *arXiv:2001.07301 [cs, stat]*, January 2020. URL http://arxiv.org/abs/2001.07301.

Woodworth, B., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., Soudry, D., and Srebro, N. Kernel and Rich Regimes in Overparametrized Models. *arXiv:2002.09277 [cs, stat]*, July 2020. URL http://arxiv.org/abs/2002.09277.

Yang, G. Tensor Programs I: Wide Feedforward or Recurrent Neural Networks of Any Architecture are Gaussian Processes. *arXiv:1910.12478 [cond-mat, physics:math-ph]*, December 2019a. URL http://arxiv.org/abs/1910.12478.

Yang, G. Scaling Limits of Wide Neural Networks with Weight Sharing: Gaussian Process Behavior, Gradient Independence, and Neural Tangent Kernel Derivation. *arXiv:1902.04760 [cond-mat, physics:math-ph, stat]*, February 2019b. URL http://arxiv.org/abs/1902.04760.

Yang, G. Tensor Programs II: Neural Tangent Kernel for Any Architecture. *arXiv:2006.14548 [cond-mat, stat]*, August 2020a. URL http://arxiv.org/abs/2006.14548.

Yang, G. Tensor Programs III: Neural Matrix Laws. *arXiv:2009.10685 [cs, math]*, September 2020b. URL http://arxiv.org/abs/2009.10685.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. A Mean Field Theory of Batch Normalization. *arXiv:1902.08129 [cond-mat]*, February 2019. URL http://arxiv.org/abs/1902.08129.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. Stochastic Gradient Descent Optimizes Over-parameterized Deep ReLU Networks. *arXiv:1811.08888 [cs, math, stat]*, November 2018. URL http://arxiv.org/abs/1811.08888.