INVITED PAPER

Mathematical Models of Overparameterized Neural Networks

The focus of this article is on theoretical developments concerning the analysis of overparameterized neural networks.

By Cong Fang, Hanze Dong, and Tong Zhang, *Fellow IEEE*

ABSTRACT | Deep learning has received considerable empirical success in recent years. However, while many *ad hoc* tricks have been discovered by practitioners, until recently, there has been a lack of theoretical understanding for tricks invented in the deep learning literature. Known by practitioners that overparameterized neural networks (NNs) are easy to learn, in the past few years, there have been important theoretical developments in the analysis of overparameterized NNs. In particular, it was shown that such systems behave like convex systems under various restricted settings, such as for two-layer NNs, and when learning is restricted locally in the so-called neural tangent kernel space around specialized initializations. This article discusses some of these recent signs of progress leading to a significantly better understanding of NNs. We will focus on the analysis of two-layer NNs and explain the key mathematical models, with their algorithmic implications. We will then discuss challenges in understanding deep NNs and some current research directions.

KEYWORDS | Mean-field (MF) analysis; neural networks (NNs); neural tangent kernel (NTK); overparameterization; random features.

I. INTRODUCTION

Neural networks (NNs) are computational models that are composed of (possibly multiple) feature representation layer(s) and a final linear learner. In recent years, deep NNs (DNNs) have largely improved the state-of-the-art performances in numerous real applications, such as image classification [1], [2], speech recognition [3], and natural language processing [4]. However, the theoretical understanding of these empirical successes for NNs is still limited. One main conceptual difficulty is the high nonconvexity of these models, which means that first-order algorithms, such as gradient descent (GD) or stochastic GD (SGD), may converge to bad local stationary points.

However, it is observed, in practice, that with the help of a number of tricks, such as dropout [5] and batch normalization [6], DNN can be reliably trained from random initialization with reproducible results. The solutions obtained by proper training procedures behave well and consistently. In other words, two different random initializations (using the same initialization and training strategy) generally lead to models that give similar predictions on test data. Thus, we may conclude that proper NN training leads to similar solutions. This behavior resembles that of convex optimization, instead of generic nonconvex optimization problems that tend to get stuck in suboptimal local stationary solutions. Because solutions from different random initializations are similar and reproducible, it can also be conjectured that, with proper training, DNNs can reach solutions that are near-global optimal. These empirical observations appear to be rather mysterious, and they require the development of new mathematical models for NNs that can bridge the gap between nonconvex and convex models to understand.

In addition to the above empirical observations, it is also known by practitioners that overparameterized NNs with many hidden units are easy to learn [7]. They achieve better and more consistent performance. Related to this empirical observation, it was noticed in the 1990s that NNs with infinitely many hidden units are easier to model and analyze theoretically [8], [9]. In the past few years, there have been many significant theoretical developments

Manuscript received August 1, 2020; revised October 27, 2020; accepted December 19, 2020. Date of publication January 18, 2021; date of current version April 30, 2021. This work was supported by GRF 16201320. *(Corresponding author: Tong Zhang.)*

Cong Fang is with the Wharton Statistics Department, University of Pennsylvania, Philadelphia, PA 16802 USA.

Hanze Dong is with the Mathematics Department, The Hong Kong University of Science and Technology, Hong Kong.

Tong Zhang is with the Computer Science and Mathematics Department, The Hong Kong University of Science and Technology, Hong Kong (e-mail: tongzhang@tongzhang-ml.org).

Digital Object Identifier 10.1109/JPROC.2020.3048020

in the analysis of overparameterized NNs with massive hidden units that approach infinity. Especially, it was shown that such systems behave like convex systems under various restricted settings. This provides theoretical justifications of the empirical observations of the reproducibility of NN training.

In this article, we review some recently developed mathematical models for overparameterized NNs, with the focus on the neural tangent kernel (NTK) view and the mean-field (MF) view. Section II introduces the basic formulation for two-layer NNs. Section III introduces a closely related learning model, i.e., random kitchen sinks [10]. In Section IV, we examine the NTK view for two-layer NNs, which shows that a two-layer NN can be written equivalently as a linear model in the tangent space under some specialized conditions. Section V considers the MF view for two-layer NNs. In this view, a continuous two-layer NN is regarded as a learned distribution over the weights, which leads to a more realistic mathematical model for analyzing practical behaviors of NNs. In Section VI, we compare the three models from the feature learning perspective. Section VII considers the possible extensions on DNNs for NTK and MFs. In Section VIII, we introduce some basic complexity results for NTK. Section IX reviews some other mathematical models. Finally, we conclude this article and outline active research directions in Section X.

II. TWO-LAYER NEURAL NETWORKS

Two-layer NNs have a history dating back to the 1940s [11]. A discrete two-layer NN can be viewed as a k-dimensional vector valued function of a d-dimensional input vector x, which has the following form:

$$f([u,\theta],x) = \frac{\alpha}{m}\sum_{j=1}^{m} u_j h(\theta_j, x) \tag{1}$$

where $x \in \mathbb{R}^d$, $\theta_j \in \mathbb{R}^d$, and $u_j \in \mathbb{R}^k$. The model parameters are $\{[u_j,\theta_j] : j = 1,\ldots,m\}$, which will be learned via training. Here, $\alpha > 0$ is a real-valued scaling parameter that is not learned. It is included here to differentiate two different regimes of overparameterized NNs. In this system, there are m hidden units (or neurons), and each hidden unit corresponds to a function $h(\theta_j, x)$ of the input x. It maps the original input feature x to a new feature $h(\theta_j, x)$, with a parameter θ_j that is learned. The function $h(\theta, x)$ is a real-valued function. In applications, it often takes the following form:

$$h(\theta, x) = h_0(\theta^\top x)$$

where $h_0(\cdot)$ is called an activation function, and the standard choices include rectified linear unit (ReLU) $h_0(z) = \max(0, z)$ and sigmoid $h_0(z) = e^z/(1+e^z)$.

In order to learn the NN parameters, we consider the minimization of the following optimization problem:

$$\min_{u,\theta} \phi(u,\theta)$$
$$\phi(u,\theta) = \frac{1}{n}\sum_{i=1}^{n} L(f([u,\theta],x_i),y_i) + R(u,\theta). \tag{2}$$

Here, $\{(x_i, y_i) : i = 1,\ldots,n\}$ are the training data, L is a loss function, such as the soft-max loss for k-class classification problem with $v \in \mathbb{R}^k$ and $y \in \{1,\ldots,k\}$

$$L(v,y) = -\log\frac{\exp(v_y)}{\sum_{j=1}^{k}\exp(v_j)}$$

and $R(u,\theta)$ is a regularizer (also called weight decay in the NN literature), such as the L_2 regularization

$$R(u,\theta) = \frac{1}{2m}\sum_{j=1}^{m}\left[\lambda_u\|u_j\|_2^2 + \lambda_\theta\|\theta_j\|_2^2\right].$$

In this article, we consider the situation that the regularizer $R(u,\theta)$ is convex in (u,θ), and the loss function $L(v,y)$ is convex in v.

In general, we consider random initialization, and specifically random Gaussians

$$u_j \sim N(0,\sigma^2), \quad \theta_j \sim N(0,\sigma^2)$$

with different scalings of α. The training is often performed via SGD, where we randomly pick a training datum i (or a minibatch of data points) and update parameters as

$$u \leftarrow u - \eta\nabla_u[L(f([u,\theta],x_i),y_i) + R(u,\theta)]$$
$$\theta \leftarrow \theta - \eta\nabla_\theta[L(f([u,\theta],x_i),y_i) + R(u,\theta)].$$

The parameter η is referred to as learning rate. It is known in optimization that, if we let $\eta \to 0$ properly, then the procedure converges to a point (u_∞,θ_∞) such that $\nabla\phi(u_\infty,\theta_\infty) = 0$. If the loss function $L(v,y)$ is convex in v, then $\phi(u,\theta)$ is convex in u but not convex in θ. Therefore, in general, $\nabla\phi(u_\infty,\theta_\infty) = 0$ does not imply that (u_∞,θ_∞) achieves a global optimal solution. In order to understand the empirical observation that (u_∞,θ_∞) behaves like a solution that is close to global optimal, especially when m is large, mathematical models have been developed in recent years to explain the empirical phenomenon.

III. RANDOM FEATURES

Before developing mathematical models for two-layer NNs, we will first consider a closely related machine learning model that employs random features. We note that, in a two-layer NN, the hidden units are feature functions $h(\theta_j, x)$ that contain parameters θ_j to be learned during

NN training. The random feature approach (denoted by RF in this article) is also referred to as random kitchen sinks [10], [12], [13]. In RF, we still consider the function (1), but assume that θ_j are fixed at $\tilde{\theta}_j$ that is generated from a random distribution, typically a Gaussian distribution

$$\tilde{\theta}_j \sim N(0, \sigma^2)$$

which is not learned during training. Only parameters $\{u_j : j = 1, \ldots, m\}$ are learned. In this case, we may take any fixed scaling α, such as $\alpha = 1$, because the scaling is not important for the random feature approach.

In RF, the model $f(\cdot, x)$ becomes linear with respect to the model parameters $\{u_j\}$. Therefore, if $L(v, y)$ is convex in v, then the objective function (2) is convex, and thus, the convergence of SGD is easy to analyze.

Since the parameters $\{\tilde{\theta}_j\}$ do not change during training, we apply SGD to learn $\{u_j\}$ only in the training process that optimizes (2)

$$u \leftarrow u - \eta \nabla_u [L(f([u, \tilde{\theta}], x_i), y_i) + R(u, \tilde{\theta})].$$

We are interested in the situation that the number of hidden units $m \to \infty$. It can be shown that in this case, the function learned by the random feature method converges to a well-defined limit [10]. To understand its behavior, it is useful to consider the kernel view for RF.

Note that if we let

$$u = \frac{\alpha}{\sqrt{m}}[u_1, \ldots, u_m]^\top \in \mathbb{R}^{d \times k}$$
$$h(x) = \frac{1}{\sqrt{m}}[h(\tilde{\theta}_1, x), \ldots, h(\tilde{\theta}_m, x)]^\top \in \mathbb{R}^d$$

then the random feature method corresponds to the linear model

$$f(u, x) = u^\top h(x). \tag{3}$$

We consider the L_2 regularization

$$R(u, \theta) = \frac{\lambda}{2m} \sum_{j=1}^{m} \|\alpha u_j\|_2^2 = \frac{\lambda}{2} \|u\|_F^2$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix. Then, with fixed θ_j, the objective function $\phi(u, \theta)$ of (2) becomes

$$\phi(u) = \frac{1}{n} \sum_{i=1}^{n} L(u^\top h(x_i), y_i) + \frac{\lambda}{2} \|u\|_F^2 \tag{4}$$

which is convex in u.

In order to obtain the kernel representation, we note that the first-order optimality condition of (4) at the optimal solution can be written as

$$u = \sum_{i=1}^{n} h(x_i) \beta_i^\top \tag{5}$$

where

$$\beta_i = -\frac{1}{\lambda n} L_1'(u^\top h(x_i), y_i)$$

and $L_1'(v, y) \in \mathbb{R}^k$ is the gradient of $L(v, y)$ with respect to v.

This leads to the kernel representation [14], where the kernel is defined as the inner product of the feature vectors $h(x)$ and $h(x')$ for two input variables $x \in \mathbb{R}^d$ and $x' \in \mathbb{R}^d$

$$k_m(x, x') = h(x)^\top h(x') = \frac{1}{m} \sum_{j=1}^{m} h(\tilde{\theta}_j, x) h(\tilde{\theta}_j, x').$$

Consider a function represented using this kernel with parameters $\beta_i \in \mathbb{R}^k$ for $i = 1, \ldots, n$

$$f_m(\beta, x) = \sum_{i=1}^{n} \beta_i k_m(x, x_i).$$

Then, we can see from (5) that $f_m(\beta, x) = u^\top h(x) = f(u, x)$. That is, the original linear function has a kernel representation. Moreover, the regularizer also has a kernel representation

$$\|u\|_F^2 = \sum_{j=1}^{n} \sum_{\ell=1}^{n} (\beta_j^\top \beta_\ell) k_m(x_i, x_j).$$

Using the kernel representation, the solution of RF, which minimizes the objective function (4) over u, is equivalent to the solution of the following kernel optimization problem:

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} L(f_m(\beta, x_i), y_i) + R_m(\beta) \tag{6}$$
$$R_m(\beta) = \frac{\lambda}{2} \sum_{j=1}^{n} \sum_{\ell=1}^{n} (\beta_j^\top \beta_\ell) k_m(x_i, x_j).$$

Let β be the solution of (6), and we may obtain the solution of (4) using the relationship between β and u in (5).

The kernel formulation (6) is particularly useful for analyzing the behavior of overparameterized RF in the limiting case of $m \to \infty$. This is because, when $m \to \infty$, we have

$$k_m(x, x') \to k_\infty(x, x') = \int h(\tilde{\theta}, x) h(\tilde{\theta}, x') d\rho_0(\tilde{\theta})$$

which is a well-defined kernel, where $\rho_0(\tilde{\theta})$ is the random distribution of $\tilde{\theta}$, such as the Gaussian distribution $N(0, \sigma^2)$

in our case. It follows that as $m \to \infty$, the kernel function:

$$f_m(\beta, x) = \sum_{i=1}^{n} \beta_i k_m(x, x_i) \to f_\infty(\beta, x)$$

where

$$f_\infty(\beta, x) = \sum_{i=1}^{n} \beta_i k_\infty(x, x_i).$$

Moreover, the corresponding optimization problem of (6) becomes

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} L(f_\infty(\beta, x_i), y_i) + R_\infty(\beta)$$
$$R_\infty(\beta) = \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{\ell=1}^{n} (\beta_i^\top \beta_\ell) k_\infty(x_i, x_\ell) \tag{7}$$

which is also well-defined.

Note that, in the kernel formulation, the number of model parameters $\{\beta_i\}$ is n, which remains finite when $m \to \infty$. Therefore, the limit of $\{\beta_i\}$ is well-defined.

If we consider the original random feature function (3) in the case of $m \to \infty$, the number of parameters $\{u_j\}$ will also approach infinity. In this case, we may consider u as a function of $\tilde{\theta}$, write (3) as

$$f(u, x) = \int u(\tilde{\theta}) h(\tilde{\theta}, x) d\rho_0(\tilde{\theta})$$

in the limit of $m \to \infty$, and write the two norm regularizer as

$$R(u) = \frac{\lambda}{2} \int \|u(\tilde{\theta})\|_2^2 \, d\rho_0(\tilde{\theta}).$$

With this notation for $m = \infty$, we have the relationship

$$u(\tilde{\theta}) = \sum_{i=1}^{n} \beta_i h(\tilde{\theta}, x_i)$$

where β is the solution of the kernel formulation (7).

It is known that RF works well for certain problems [10], [12]. However, for many real-world applications, such as image classification, RF is inferior to NN that learns better feature representations than random ones. We will discuss the feature learning perspective in Section VI.

One interesting extension of the RF theory that can be used to analyze the behavior of NN is presented in [15] and [16]. A kernel called conjugate kernel was introduced, and it was shown that the GD process for the NNs under some special initializations belongs to this kernel space. Moreover, any function in this kernel space can be approximated by changing the weights of the last layer. Inspired by the above findings, the authors proved the global convergence of GD for NNs. However, in their analysis, only the GD updates on the last layer contribute to the global convergence, which ignores weight updates in the lower layers. In Section IV, we will introduce the NTK view that develops theoretical results showing that weight updates in the bottom layer of two-layer NNs can also contribute to the convergence of GD.

IV. NEURAL TANGENT KERNELS

In practice, the random feature approach is often inferior to two-layer NNs because the parameters $\{\theta_j\}$ are not trained. As we have seen, the overparameterized case with $m \to \infty$ corresponds to kernel learning with a well-defined kernel. One natural question is whether this point of view can be generalized to handle two-layer NNs, where the parameters $\{\theta_j\}$ are trained together with $\{u_j\}$. Such a generalization leads to NTK [17], which we shall describe in this section. We note that the connection of infinitely wide NNs and kernel methods (Gaussian processes) has already been known in the 1990s [8], [9]. However, the more rigorous theory of NTK has only appeared very recently, e.g., [17] and [18].

In NTK, we consider a specialized scaling and random initialization of parameters. The special scaling makes it possible to consider NN parameters in a small region around the initial value when $m \to \infty$. The resulting NN with parameters restricted in this region can be well approximated by a linear model fit with random features. Similar to RF of Section III, this linearization induces a kernel in the tangent space around the initialization, which becomes NTK [17], since the weights are near their initial values during the training of the NN. This phenomenon is also referred to as the "lazy training" regime in [19]. In this regime, the system becomes linear, and the dynamics of GD (or SGD) within this region can be tracked via properties of the associated NTK.

There have been a series of studies about NTK in recent years, in which a number of researchers proved polynomial convergence rates to the global optimal, e.g., [18] and [20]–[37], and sharper generalization errors, e.g., [20], [21], [23], and [38]–[40]. The scaling α also plays a significant role in the NTK view, where a relatively large scaling α of order $\sqrt{m}$ is needed. This means that, when $m \to \infty$, $\alpha \to \infty$.

To derive NTK under the assumption of $m \to \infty$, we consider the case that $h(\theta, x)$ is differentiable with respect to x, such as the sigmoid or tanh activation function. Note that the nondifferentiable ReLU activation function can also be handled similarly although many works consider the differentiable assumption for simplicity.

We now consider a random initialization at $[\tilde{u}, \tilde{\theta}]$, around which we can linearly approximate the NN as

$$f([u, \theta], x) \approx \frac{\alpha}{m} \sum_{j=1}^{m} \big[\tilde{u}_j h(\tilde{\theta}_j, x) + (u_j - \tilde{u}_j) h(\tilde{\theta}_j, x) + \tilde{u}_j (\theta_j - \tilde{\theta}_j)^\top \nabla_\theta h(\tilde{\theta}_j, x)\big] + \text{high order terms} \tag{8}$$

where we assume that both $u - \tilde{u}$ and $\theta - \tilde{\theta}$ are small. Note that the theory of NTK requires $\alpha = O(\sqrt{m})$ so that the term $(\alpha/m)\sum_{j=1}^{m} \tilde{u}_j h(\tilde{\theta}_j, x)$ has a bounded variance. A large scaling α ($\alpha \to \infty$ as $m \to \infty$) is important for this linearization because if we fix the coefficient $\alpha\tilde{u}_j(\theta_j - \tilde{\theta})$ for the random feature $\nabla_\theta h(\tilde{\theta}_j, x)$, then, when the scaling $\alpha \to \infty$ (as $m \to \infty$), one can show that a small change of θ leads to a big change of the output function values. This means that, in the continuous limit, we should let $(\theta_j - \tilde{\theta}) \to 0$. A similar claim holds for $u_j - \tilde{u}_j$. In this case, the higher order terms will be $o(\alpha\|u_j - \tilde{u}\| + \alpha\|\theta_j - \tilde{\theta}_j\|)$ that approach zero, and the linear approximation of (8) is accurate.

This linear approximation employs random features $h(\tilde{\theta}_j, x)$ and $\tilde{u}_j \nabla_\theta h(\tilde{\theta}_j, x)$ for $j = 1, \ldots, m$. Compared with the RF approach, which only uses the random features $h(\tilde{\theta}_j, x)$, two-layer NNs use additional random features $\tilde{u}_j \nabla_\theta h(\tilde{\theta}_j, x)$.

In order to motivate the NTK kernel, we consider the following representation, similar to (5) for RF:

$$\begin{cases} u_j = \tilde{u}_j + \alpha^{-1} \sum_{i=1}^{n} \beta_i^u h(\tilde{\theta}_j, x_i) \\ \theta_j = \tilde{\theta}_j + \alpha^{-1} \sum_{i=1}^{n} \tilde{u}_j^\top \beta_i^\theta \nabla h(\tilde{\theta}_j, x_i). \end{cases} \tag{9}$$

Here, we have both $\beta_i^u \in \mathbb{R}^k$ and $\beta_i^\theta \in \mathbb{R}^k$. Using this representation, the linear approximation of (8) becomes

$$\frac{1}{m}\sum_{j=1}^{m}\left[\alpha\tilde{u}_j h(\tilde{\theta}_j, x) + \sum_{i=1}^{n} \beta_i^u h(\tilde{\theta}, x_i) h(\tilde{\theta}_j, x) + \sum_{i=1}^{n} \tilde{u}_j \tilde{u}_j^\top \beta_i^\theta \nabla_\theta h(\tilde{\theta}_j, x_i)^\top \nabla_\theta h(\tilde{\theta}_j, x)\right].$$

Using the relationship of kernel as the inner product of features for linear models, this linear approximation of two-layer NN corresponds to the following kernel function representation:

$$f_m([\beta^u, \beta^\theta], x) = \sum_{i=1}^{n} \left[\beta_i^u k_m^u(x, x_i) + k_m^\theta(x, x_i)\beta_i^\theta\right]$$

where

$$k_m^u(x, x') = \frac{1}{m}\sum_{j=1}^{m} h(\tilde{\theta}_j, x) h(\tilde{\theta}_j, x')$$

which corresponds to the features of the linear coefficients u (also used by the RF model of Section III), and

$$k_m^\theta(x, x') = \frac{1}{m}\sum_{j=1}^{m} \tilde{u}_j \tilde{u}_j^\top \nabla_\theta h(\tilde{\theta}_j, x)^\top \nabla_\theta h(\tilde{\theta}_j, x')$$

which is a $k \times k$ matrix corresponding to the features of the linear coefficients θ (which was not used by the RF model of Section III). This kernel representation is referred to as NTK.

In the NTK representation, from the relationship of $[u, \theta]$ and $[\beta^u, \beta^\theta]$ in (9), we can see that, as $\alpha \to \infty$, $\theta \to \tilde{\theta}$ and $u \to \tilde{u}$, which means that we have a more accurate linear approximation when α is large.

In the infinity-width limit of $m \to \infty$, the kernel becomes the infinite-width NTK kernel, which is well-defined

$$k_m^u(x, x') \to k_\infty^u(x, x') = \int h(\tilde{\theta}, x) h(\tilde{\theta}, x') d\rho_0(\tilde{\theta})$$
$$k_m^\theta(x, x') \to k_\infty^\theta(x, x') = \int \tilde{u}\tilde{u}^\top \nabla_\theta h(\tilde{\theta}, x)^\top \nabla_\theta h(\tilde{\theta}, x') d\rho_0(\tilde{u}, \tilde{\theta})$$

where $\rho_0(\tilde{u}, \tilde{\theta})$ is the random initialization distribution for $[\tilde{u}, \tilde{\theta}]$, and in our case, it is chosen as $N(0, \sigma^2) \times N(0, \sigma^2)$. Moreover, $\rho_0(\tilde{\theta})$ is the marginal random initialization distribution for $\tilde{\theta}$, which, in our case, is $N(0, \sigma^2)$. With this choice, we note that $\int \tilde{u}\tilde{u}^\top d\rho_0(\tilde{u}|\tilde{\theta}) = \sigma^2\, I_{k\times k}$ is proportional to a diagonal matrix. Therefore, we may also replace the $k \times k$ matrix kernel $k_\infty^\theta(x, x')$ by the following scalar kernel:

$$\sigma^2 \int \nabla_\theta h(\tilde{\theta}, x)^\top \nabla_\theta h(\tilde{\theta}, x) d\rho_0(\tilde{\theta}).$$

We may view the infinite-width NN as a kernel method in a small neighborhood around the initialization as

$$f_\infty([\beta^u, \beta^\theta], x) = \sum_{i=1}^{n} \left[\beta_i^u k_\infty^u(x, x_i) + k_\infty^\theta(x, x_i)\beta_i^\theta\right].$$

This function gives an equivalent representation of the linear approximation of two-layer NN in (8) with $m \to \infty$ and $\alpha \to \infty$. Therefore, in this case, the optimization in $[u, \theta]$ can be equivalently represented using optimization in the kernel representation. The corresponding optimization problem using kernel representation can be written as

$$\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(f_\infty([\beta^u, \beta^\theta], x_i), y_i). \tag{10}$$

It is worth mentioning that, in the NTK view of NNs, we usually do not employ regularization $R(u, \theta)$. This is because the solution of NN with a nontrivial regularization will not lie in a small region around $[\tilde{u}, \tilde{\theta}]$.

In the NTK view, under appropriate conditions, it is possible to show that the optimization problem (10) in the kernel space is equivalent to the solution by SGD in the original representation (1) when α is large. In this case, the general SGD for two-layer NN in the original parameter space can be expressed as

$$u \leftarrow u - \eta_u \nabla_u L(f([u, \theta], x_i), y_i)$$
$$\theta \leftarrow \theta - \eta_\theta \nabla_\theta L(f([u, \theta], x_i), y_i).$$

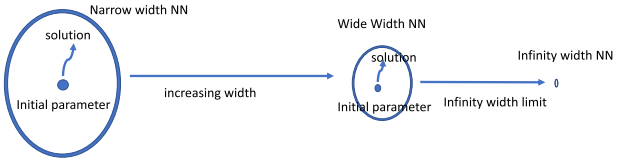


Fig. 1. ***Solution neighborhood size versus NN width m.***

If we consider using the same learning rate for the rescaled parameter $\alpha u/m$ and θ, as often done in practice, then we shall set

$$\frac{\alpha}{m}\eta_u = \eta, \quad \eta_\theta = \eta \tag{11}$$

where η is a small learning rate. We note that, in this case, the learning rate $\eta_u = O(\eta m/\alpha)$ will be large compared with η_θ if we set $\alpha = O(\sqrt{m})$ required by NTK. The large learning rate η_u will move u to be far away from the initialization $\tilde{u}$, violating the standard NTK requirement of $u \approx \tilde{u}$. Nevertheless, we note that the requirement of $\theta \approx \tilde{\theta}$ is more important in NTK for linearly approximating the nonlinear function $h(\theta, x)$. With additional complexity, it is, thus, possible to extend the NTK analysis to handle the situation that $\theta \approx \tilde{\theta}$, but u may not be close to $\tilde{u}$.

Because of the abovementioned complexity, for the theoretical analysis of NTK, one often assumes a smaller learning rate for u [41], such as

$$\alpha\eta_u = \eta, \quad \eta_\theta = \eta \tag{12}$$

or as

$$\eta_u = \eta, \quad \eta_\theta = \eta. \tag{13}$$

For the learning rates in (12), since η_u is much smaller than η_θ, we know that u moves very little compared with θ. Consequently, we can see from (9) that β^u moves very little compared with β^θ. Therefore, the kernel $k_\infty^\theta(\cdot,\cdot)$ is effective, while the kernel $k_\infty^u(\cdot,\cdot)$ is not effective. The learning rate (13) does not suffer from this problem. From (9), it can be seen that the corresponding modifications of both β^u and β^θ are at the same order. Therefore, in such case, both kernels are effective.

It can be shown that, with learning rates set as either (12) or (13), when $m \to \infty$ and $\alpha \to \infty$, the final solution of (2) without regularization can reach zero-error within a very small neighborhood of the initialization, with radius approaching zero as $\alpha \to \infty$ (e.g., [18]). This phenomenon is illustrated in Fig. 1. This regime is called the NTK regime, where the two-layer NN can be linearized as a kernel method, and the optimization process lies inside a small neighborhood around the initialization.

This property can also be validated empirically in the actual NN optimization process. In this article, we use the MNIST handwritten digits data set available from http://yann.lecun.com/exdb/mnist/ [42] to demonstrate various aspects of the different frameworks. This data set has a training set of 60 000 examples and a test set of 10 000 examples and is one of the standard data set for image classification. The theory of NTK implies that, when the scaling parameter α increases, the NN training process belongs to a smaller and smaller neighborhood of the initialization, and NTK approximation becomes more and more accurate. This phenomenon is shown in Fig. 2, where the average distances between θ (and u) and the initialization are plotted for different values of α.

Another factor that determines the accuracy of NTK approximation is the number of hidden units m, which measures the degree of overparameterization. Fig. 3 shows that, when m increases, the solution of the NNs becomes closer to the initialization, and this phenomenon happens both with the practical learning rate $\eta_u/\eta_\theta = m/\alpha$ and with the NTK theoretical learning rate $\eta_u/\eta_\theta = 1$. From this experiment, it is reasonable to speculate that, as m approaches infinity, the first-order approximation of NN (NTK) will become more reliable.

It is also useful to point out that NTK approximation works better on simple data sets and fails more easily on complex data sets. Fig. 4 compares the objective functions of the NN (both on training and test data) to its linearized NTK approximations during the training process. We compare the linearization error both on the MNIST data set and on a simpler 100-D synthetic data set made by make_classification in sklearn. For an NN with $m = 1000$ and $\eta = 0.1$, we can notice that the linearization of NN on the simpler data set is almost perfect, which means that the entire training process is well-approximated by NTK. However, there is a noticeable discrepancy between NTK and the actual NN on the MNIST data, which means that NTK does not fully explain the actual NN training process well in this case. This experiment implies that, for relatively complex data sets, the NTK approximation requires a much wider NN.

In summary, when $\alpha \to \infty$ and $m \to \infty$, and the formulation does not contain regularization, then we have the so-called NTK regime, with the following properties.

1) Initialize NN with a certain scaling.
2) The network is sufficiently large.
3) The formulation does not contain regularization.
4) The learning rate is sufficiently small.
5) The NN solution path remains close to the initialization and can be linearized around the initialization.
6) The linearization induces a kernel (NTK) that is convex.

The implications of the NTK view are given as follows.

1) The solution is in a tiny neighborhood of the initial point.
2) The problem becomes convex, which can be solved efficiently.
3) The solution path is reproducible in kernel representation.

The theory of NTK applies to a specialized regime with special initialization. It also assumes a small learning rate so that the learned parameter does not escape from the NTK region. Although this is a nice mathematical model,

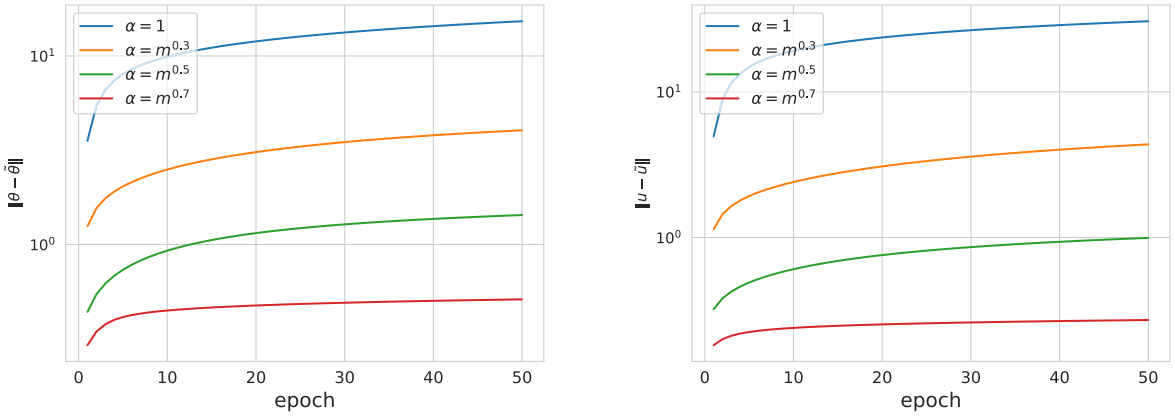


Fig. 2. ***Impact of α for NN optimization ($\eta_u/\eta_\theta = 1, m = 10^4$).***

there are several problems, making it unsuitable for a general theory of NNs.

Note that RF can be considered as a two-layer NN where the bottom layer is fixed. The model is linear with respect to the top layer, and one can incorporate regularization to improve generalization. In contrast, in the NTK regime, we perform GD to optimize NN weights in all layers, with random features generated in the tangent space around the initialization. In order to ensure that the solution of (2) can be approximated by (10), the resulting optimization problem (10) of NTK cannot include nontrivial regularizers that will pull the training process out of the tangent space. It is natural to extend (10) by adding regularization, where we may consider the following more general formulation of NTK in (10) to the following regularized NTK method, which may be regarded as an NTK motivated new learning algorithm, although it may not be a good approximation for two-layer NNs anymore:

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} L(f_\infty([\beta^u, \beta^\theta], x_i), y_i) + R([\beta^u, \beta^\theta]) \tag{14}$$

where

$$R([\beta^u, \beta^\theta]) = \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{\ell=1}^{n} \left[\left(\left(\beta_i^u\right)^\top \beta_\ell^u \right) k_\infty^u(x_i, x_\ell) + \left(\beta_i^\theta\right)^\top k_\infty^\theta(x_i, x_\ell) \beta_\ell^\theta \right].$$

Although this regularized formulation is natural in the kernel learning setting, it is not equivalent to two-layer NN due to the addition of the regularization term.

A major problem of the NTK view is that the practical performance of the NTK solution from (14) is often inferior to that of the fully trained NNs, despite the equivalence that can be proved under certain theoretical assumptions. Even an infinitely wide NTK cannot achieve state-of-the-art performance. In the following, we explain why this happens in practice and what can break the NTK view for NN learning. We show that the NTK regime is broken by standard tricks in NN learning.

Although the random parameter initialization with large scaling α is used by practitioners, and the choice is consistent with that required by the NTK viewpoint,

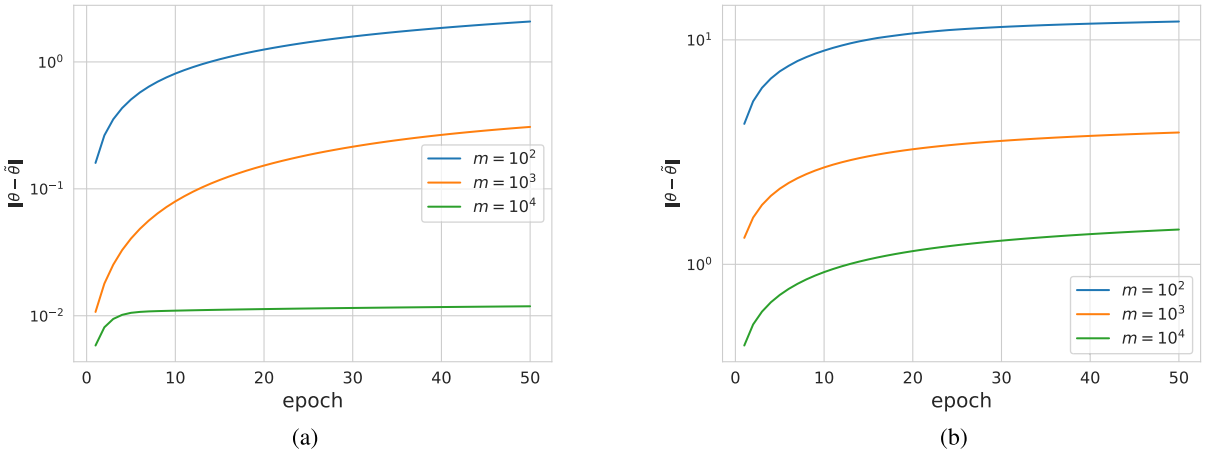


Fig. 3. ***Impact of m for NN optimization ($\alpha = \sqrt{m}$). (a) $\eta_u/\eta_\theta = m/\alpha$. (b) $\eta_u/\eta_\theta = 1$.***

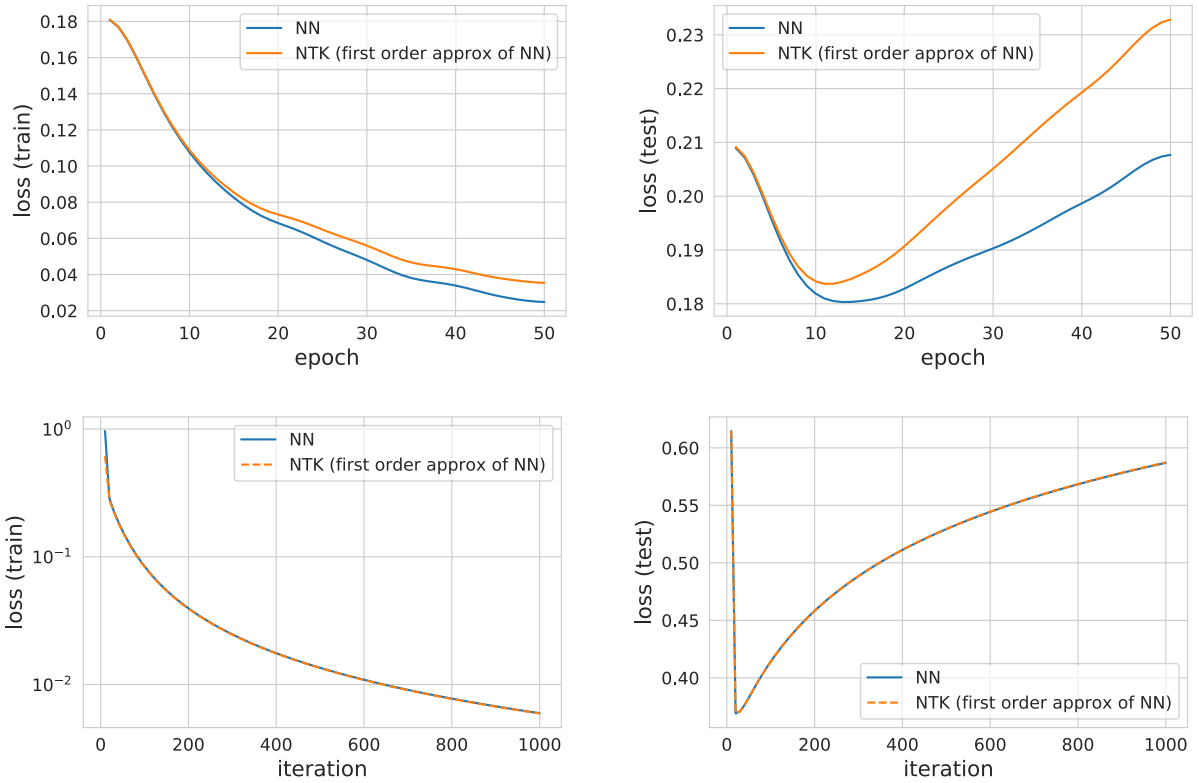


Fig. 4. ***Linearization of NN (first row: MNIST; second row: synthetic data).***

practitioners use a large initial learning rate, while the small learning rates are required by the theoretical analysis. Consequently, the optimized NN by using practical SGD procedures goes out of the NTK regime. Because of this, the NTK linear approximation fails.

Moreover, in practice, NNs are not infinite-width, and thus, the large m theory does not exactly match the practical behavior of NN learning. Another theoretical condition for the NTK view is to not impose nontrivial regularization[1] because regularization automatically pushes the solution away from the initialization, which violates the NTK regime. However, regularization (or weight decay) is frequently used by practitioners. Besides, there is currently no analysis of NNs that can incorporate batch normalization in the NTK regime.

One of the key reasons that NTK does not perform well in practice is that the NTK method is very similar to RF, in which it also employs random features. Compared with RF of Section III, NTK contains an additional kernel corresponding to the random features $\tilde{u}\nabla_\theta h(\tilde{\theta}, x)$. In fact, the mathematical theory of NTK relies mostly on the modification of θ associated with random features $\tilde{u}\nabla_\theta h(\tilde{\theta}, x)$ to reduce training loss, while the mathematical theory of RF relies on the modification of u associated with random features $h(\tilde{\theta}, x)$ to reduce training loss. This is why the theoretical analysis of NTK relies on a small learning rate η_u. Nevertheless, the additional random features used by NTK provide extra information over RF.

Since NTK is still a random feature-based method, it does not learn feature representations. In contrast, it is well-known by practitioners that a key benefit of NN learning is the ability to learn useful feature representations. The theory of NTK completely fails to explain the benefit of feature learning by NNs. We will investigate this issue further in Section VI.

The NTK view is also inconsistent with many technical tricks used in practical NN training, which benefits learning performance, such as large initial learning rate and momentum, which may push the model parameters out of the initialization neighborhood. Here, we show some cases in real practice to demonstrate these phenomena.

As shown in Fig. 5, when we use a large learning rate or employ momentum, the difference between the initial parameters and the final solution increases. In such a situation, the first-order approximation (19) used by NTK fails to capture the dynamics of NN. The gap between NN and NTK is significant, and the performance of NN is better with these tricks. A few other works have also mentioned the phenomenon that a large initial learning rate leads to better NN solution [43], [44], which cannot be explained by the NTK view very well.

[1]We note that the NTK regime can still incorporate very small regularizers. For example, one can add a regularizer as $\mu(\|\theta\|^2+\|u\|^2)$, where μ is much smaller than $(1/m)$. In this case, there is still a solution in the neighborhood of the initialization. Moreover, we may add regularizers centered at the initialization $\tilde{\theta}$, such as $\|\theta-\tilde{\theta}\|_2^2$, although they are not used by practitioners.

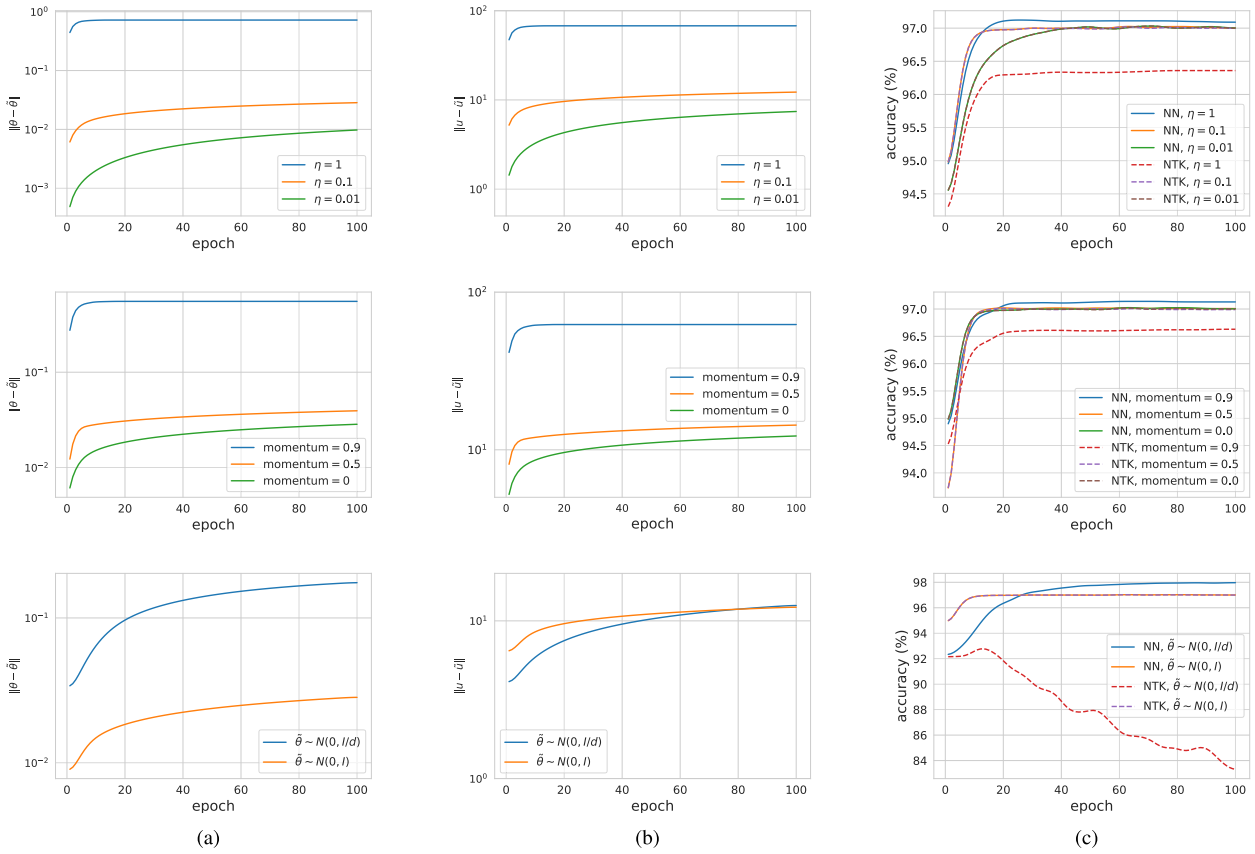


Fig. 5. ***Factors to break NTK regime (m = 5000). (a) Change of θ. (b) Change of u. (c) NN and NTK performance.***

Note that most of our NTK experiments employ the random initialization $\tilde{\theta}_j \sim N(0,1)$. In the analysis, the variance σ^2 is fixed as a constant, which does not influence asymptotic behavior significantly. However, the actual value of σ^2 plays a noticeable role when m is not large enough compared with the input dimension d. Fig. 5 shows that, when we use $\tilde{\theta}_j \sim N(0, 1/d)$ (which is a standard initialization technique with good practical performance, referred to as He initialization [45]), the NTK regime can be broken more easily. Therefore, the gap between the first-order approximation of NTK and the actual NN training cannot be ignored in practice.

V. MEAN FIELD VIEW

In order to overcome the limitations of the NTK view that we explained above, other theoretical models have been developed to investigate overparameterized two-layer NNs. In this section, we introduce another line of research, which applies the mathematical tools of the MF analysis from statistical physics to study two-layer NNs [46]–[51], [51]–[54].

In order to motivate the MF analysis for overparameterized NNs, it is instructive to first investigate the continuous dynamics of infinitely wide NNs, known as the MF limit, and then consider the finite-width NNs as its approximation. In this article, we call the corresponding analysis as the MF view. The idea of studying the mean filed limit comes from statistical physics [55], which suggests that the mathematical model of a large number of interacting neurons can be simplified using the probability distribution that represents the average effect.

Unlike the NTK view, which requires $\alpha \to \infty$ as $m \to \infty$, in the MF view [46], we may set the scaling fixed at a constant, such as $\alpha = 1$, while letting $m \to \infty$. In this case, the solution is allowed to go far from the initialization, which remedies the main limitations of NTK. Therefore, one may argue that this approach gives a more realistic mathematical model for practical behaviors of NNs.

In the limit of $m \to \infty$ with fixed α, we may consider the continuous limit of two-layer NNs in (1) as

$$f(\rho, x) = \int \alpha u h(\theta, x) d\rho(u, \theta) \tag{15}$$

where $\rho(u, \theta)$ is a probability distribution over $[u, \theta]$. In this continuous formulation, we can regard the probability measure ρ as the model parameter. Therefore, the original model parameters $\{[u_j, \theta_j]\}$ in the discrete NN formulation can be viewed as a discrete probability distribution on the model parameter space $[u, \theta] \in \mathbb{R}^k \times \mathbb{R}^d$, and this discrete probability distribution puts a mass of $1/m$ at each point $[u_j, \theta_j]$ $(j = 1, \ldots, m)$. In the continuous limit of

$m \to \infty$, this discrete probability distribution naturally converges to the distribution parameter ρ in the continuous NN formulation (15). It is easy to see that the function represented by two-layer NN becomes linear in ρ.

In the continuous limit, the training objective (2) for the discrete NN becomes

$$\phi(\rho) = \frac{1}{n}\sum_{i=1}^{n} L(f(\rho, x_i), y_i) + R(\rho)$$
$$R(\rho) = \int r(u,\theta)d\rho(u,\theta) \tag{16}$$

for the continuous NN, where $r(u,\theta)$ is a regularizer of $[u,\theta]$, such as the L_2 regularization

$$r(u,\theta) = \frac{\lambda_u}{2}\|u\|_2^2 + \frac{\lambda_\theta}{2}\|\theta\|_2^2.$$

It follows that the training objective (16) is convex with respect to ρ if both the loss function $L(\cdot)$ and the regularizer $R(\cdot)$ are convex. In fact, the global optimal solution of ρ satisfies the first-order optimality condition: for all probability measures $\rho'(u,\theta)$

$$\int g(\rho,u,\theta)d\rho'(u,\theta) \geq \int g(\rho,u,\theta)d\rho(u,\theta) \tag{17}$$

where

$$g(\rho,u,\theta) = \frac{1}{n}\sum_{i=1}^{n} \alpha L_1'(f(\rho,x_i),y_i)uh(\theta,x_i) + r(u,\theta) \tag{18}$$

is the derivative of $\phi(\rho)$ with respect to the component $\rho(u,\theta)$ by regarding the distribution ρ as an infinite-dimensional vector $\rho = \{\rho(u,\theta)\}$. Here, $L_1'(v,y) = \nabla_v L(v,y)$. Note that, if we can find ρ such that

$$g(\rho,u,\theta) = c \tag{19}$$

for a constant c, then (17) is satisfied. This is because, in this case, we have, for all ρ':

$$\int g(\rho,u,\theta)d\rho'(u,\theta) = c = \int g(\rho,u,\theta)d\rho(u,\theta).$$

In the MF view, we may take the following connection of the discrete NN versus continuous NN when m is large. The hidden units $[u_j,\theta_j]$ of the discrete NN (1) can be viewed as m particles sampled from the distribution $\rho(u,\theta)$. In the training process, we move each particle $[u_j,\theta_j]$ using SGD, which is the derivative of the objective function with respect to each particle. In the continuous limit, we have infinitely many particles, and each particle $[u,\theta]$ also moves according to the gradient of the objective function with respect to the parameter. In the literature, such a gradient is often referred to as gradient flow [56], which characterizes the learning dynamics of the continuous formulation. In the following, we will present a more mathematical description.

In the continuous formulation, a hidden unit can be regarded as a particle indexed by a parameter z_0 sampled from a distribution $\nu_0(z_0)$. Here, z_0 only plays the role of discrete index j in the discrete formulation, and its own value is of no significance. The initial distribution $\nu_0(z_0)$ is introduced for convenience so that we can sample over the index z_0. In the discrete setting, it is simply the uniform distribution over $j=1$ to $j=m$.

Each particle indexed by z_0 also has a parameter $[u,\theta]$, which will be trained. We assume that, at time t, we move each particle during the training process so that the model parameter becomes $[u(t,z_0),\theta(t,z_0)]$. If we take $z_0 \in \mathbb{R}^{k+d}$, with $\nu_0(z_0)$ as a Gaussian distribution, then we may simply initialize $[u,\theta]$ as $[u(0,z_0),\theta(0,z_0)] = z_0$. Since z_0 is sampled from $\nu_0(z_0)$, the particles $[u(t,z_0),\theta(t,z_0)]$ induce a probability measure $\rho_t(u,\theta)$ on $[u,\theta] \in \mathbb{R}^k \times \mathbb{R}^d$. Here, the time-dependent parameters $u(t,z_0)$ and $\theta(t,z_0)$ are obtained via training over the time.

Using the above terminology, the optimization of $\rho(u,\theta)$ in (15) leads to a distribution $\rho_t(u,\theta)$ at training time t, which is by moving hidden unit parameters $[u,\theta]$ via gradient flow with respect to the objective function $\phi(\rho_t)$. More precisely, the corresponding particle movement in the continuous limit obeys the gradient flow equation [47]

$$\begin{cases} \dfrac{\partial u(t,z_0)}{\partial t} = -\eta(t)\nabla_u g(\rho_t, u(t,z_0),\theta(t,z_0)) \\ \dfrac{\partial \theta(t,z_0)}{\partial t} = -\eta(t)\nabla_\theta g(\rho_t, u(t,z_0),\theta(t,z_0)) \end{cases} \tag{20}$$

where the particle gradient $g(\rho,u,\theta)$ for a particle $[u,\theta]$ is defined in (18). The gradient flow direction of a particle $[u,\theta]$ is the gradient of $g(\rho,u,\theta)$, which is equivalent to the gradient of the objective with respect to each particle parameter $[u_j,\theta_j]$ in the discrete NN formulation. Therefore, (20) is the continuous version of GD method with respect to the model parameter $[u,\theta]$ associated with the hidden units, and this continuous version of GD method tries to minimize the objective function (16).

The gradient flow equation (20) implies a partial differential equation (PDE) for the probability measure ρ_t as

$$\frac{\partial \rho_t(u,\theta)}{\partial t} = \eta(t)\nabla \cdot [\rho_t(u,\theta)\nabla g(\rho_t,u,\theta)] \tag{21}$$

with its solution interpreted in the weak sense. This equation describes the dynamics of the objective function parameter ρ of (16) under the continuous GD method of (20). Here, we use the simplified notation

$$\nabla g(\rho,u,\theta) = [\nabla_u g(\rho,u,\theta), \nabla_\theta g(\rho,u,\theta)].$$

The differential equation of ρ_t in (21) characterizes the dynamics of ρ_t in the NN training process, and it can be shown that the objective value reduces according to the following ordinary differential equation:

$$\begin{aligned}\frac{d\phi(\rho_t)}{dt} &= \int \frac{\delta\phi(\rho_t)}{\delta\rho_t} \cdot \frac{\partial\rho_t}{\partial t} d\theta du \\ &= \int g(\rho_t, u, \theta)\ \eta(t)\ \nabla \cdot [\rho_t(u,\theta)\nabla g(\rho_t, u, \theta)] d\theta du \\ &= -\eta(t) \int \|\nabla g(\rho_t, u, \theta)\|_2^2\ d\rho_t(u,\theta). \end{aligned} \tag{22}$$

The derivation of the first equation has used the calculus of variations, which may be considered as the functional gradient of ϕ with respect to ρ_t by treating ρ_t as an infinite-dimensional vector indexed by (θ, u). The functional gradient is given by $g(\rho_t, u, \theta)$, which leads to the second equation. In the third equation, we have used the integration by parts, and with a slight abuse of notation, we have used the notation $d\rho_t(u,\theta) = \rho_t(u,\theta)d\theta du$, which does not differentiate measure ρ_t and its corresponding density representation. This equation is the key to prove global convergence in the MF approach. It shows that the GD method of (20) reduces the objective function of (16), and the result is stated using the probability measure ρ_t, which is what we want to learn in the continuous NN formulation.

Since the objective function is bounded from below, from (22), we can obtain that, as $t \to \infty$, we must have $d\phi(\rho_t)/dt \to 0$. It follows that:

$$\lim_{t\to\infty} \int \|\nabla g(\rho_t, u, \theta)\|_2^2 d\rho_t(u,\theta) = 0. \tag{23}$$

However, this does not ensure that the objective function reaches the global minimum, unless additional conditions are imposed. Next, we shall present an intuitive explanation first and then describe more rigorous results.

From (23), if we can show $d\rho_t(u,\theta) \neq 0$ for all $[u,\theta]$, then we have $\|\nabla g(\rho_t, u, \theta)\|_2^2 = 0$ for all $[u,\theta]$. In this case, from $\nabla g(\rho_t, u, \theta) \to 0$, we obtain $g(\rho_t, u, \theta) \to c$ for a constant c, which implies the first-order condition (19). This result implies that GD training converges to the global optimal solution of (16) in the continuous setting.

A more rigorous treatment of the above reasoning was presented in [46], which considered a formulation with an additional entropy regularization term in $R(\rho)$. This entropy regularizer ensures that $d\rho_t(u,\theta) \neq 0$ for all $[u,\theta]$. In fact, with this regularization, the measure $\rho(u,\theta)$ always has a density, $d\rho(u,\theta) = p(u,\theta)dud\theta$, and we can write the regularizer as

$$R(\rho) = \lambda_p \int p(u,\theta)\log p(u,\theta) dud\theta + \int r(u,\theta)p(u,\theta)dud\theta$$

which modifies the regularizer in (16) by adding an extra entropy term. Using this regularizer, it can be shown that there is a unique global solution that satisfies (19). Moreover, under mild conditions, we have $\rho_t(u,\theta)$ converges (weakly) to a distribution $\rho_\infty(u,\theta)$ that can be lower bounded by a normal distribution [46]. Then, by using the Poincaré inequality for the Gaussian random variables (which states that, if X is a standard normal random variable, and $f(X)$ is a real-valued function, then $\mathrm{Var}(f(X)) \leq \mathbb{E}\|\nabla f(X)\|_2^2$), we may obtain from (23) that $g(\rho_\infty, u, \theta) = c$ almost everywhere for some constant c. This implies that the first-order condition (19) holds. It follows that, as $t \to \infty$, the solution converges to the unique global optimal solution.

In a practical implementation of the GD rule in (18) with entropy regularization, we need to compute the gradient $\nabla \log p(u,\theta)$ in $\nabla g(\rho, u, \theta)$. It can be shown that an equivalent implementation is to add a random noise, and the corresponding gradient flow equation of (20) becomes a stochastic PDEs (SDE) with $t \geq 0$

$$\begin{aligned} d\left[u(t,z_0), \theta(t,z_0)\right] = &-\eta(t)\nabla g(\rho_t, u(t,z_0), \theta(t,z_0))dt \\ &+ \sqrt{2\lambda_p \eta(t)}\, d\, B(t) \end{aligned}$$

where $\{dB(t)\}_{t\geq 0}$ is the standard Brownian motion in $\mathbb{R}^{k+d}$, and $g(\cdot)$ is defined in (18).

In the GD (or SGD) implementation of the Brownian motion component of this SDE, we simply add a Gaussian noise of $N(0, 2\lambda_p\eta(t))$ to each GD update step with learning rate $\eta(t)$. This method is referred to as noisy GD in the literature. With the help of entropy regularization, it can be shown that noisy GD for continuous NN converges to the unique global optimal solution, and the overparameterized discrete NN with a sufficiently large m approximately reaches this solution. This result can be used to explain why training of overparameterized NN is easier in practice, and why the (idealized) two-layer NN training process can reach a good solution with consistent performance.

The benefits of entropy regularization are threefold.

1) When we supplement the loss function with entropy regularization, the overall learning problem becomes strictly convex. Thus, a unique global minimum can be guaranteed.
2) The implementation of the entropy regularization is a very simple addition of noise. In practice, one can often observe that adding noise helps us to find a better solution.
3) After injecting noise, $d\rho_t \neq 0$ for all $[\mu,\theta]$. This fact, combined with (23), implies the pointwise vanishing of $\nabla g(\rho_t, u, \theta)$, which implies that the global minimum is achievable.

On the other hand, without the extra entropy regularization, the objective function of (16) may not have a unique global optimal solution. That is, there can be more than one solutions that satisfy the first-order condition (17). However, we can still achieve the global convergence to the optimal objective function value of (16) although the

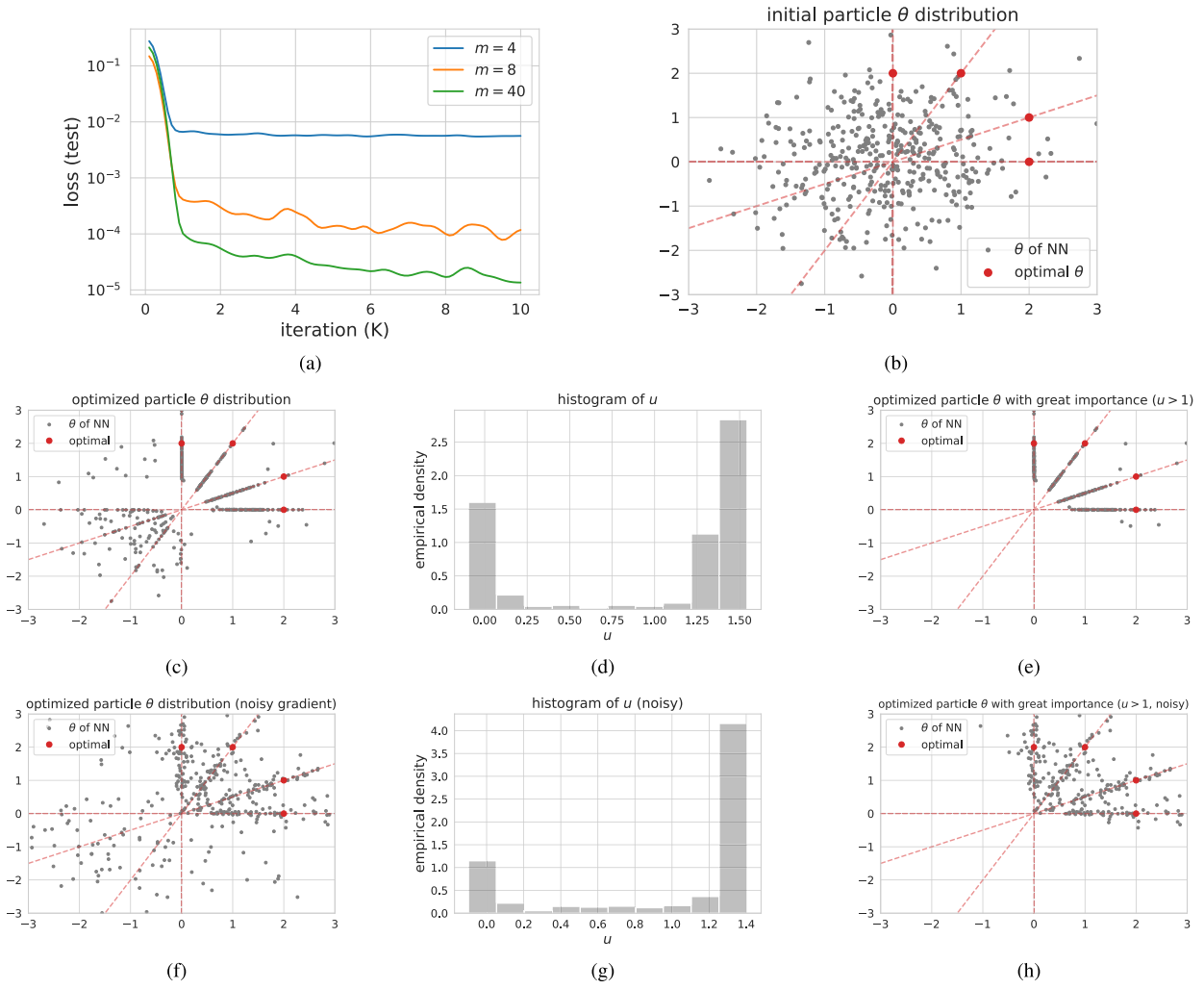


Fig. 6. ***NN optimization from an MF perspective. (a) Change of loss. (b) Initial $\tilde{\theta}$. (c) Optimized θ. (d) Normalized histogram of u. (e) Optimized θ with large u. (f) Optimized with noisy gradient. (g) Normalized histogram of u (noisy). (h) Optimized with large u (noisy).***

final solution the training process converge to may not be unique. To analyze this situation (without the extra entropy regularization), Chizat and Bach [47] considered a different assumption with homogeneous activation functions (such as ReLU) and homogeneous regularizers. Under such assumptions, it is possible to show that the solution $\rho_t(u, \theta)$ converges to a global optimal solution (which may not be unique) that satisfies (17) as $t \to \infty$.

In the MF approach, learning the distribution ρ can be viewed as learning effective feature representations. The ability of NN to learn feature representations is consistent with empirical observations. This perspective also explains why fully trained NN is better than RF and NTK, both of which employ random feature representations that are not learned. We will further discuss this aspect in Section VI.

To visualize the process of learning ρ, we conduct an experiment to reproduce an $m = 4$ sigmoid activated NN ($u_i = 1$, $i = 1, 2, 3$, and 4, denoted as F_4) by NNs with different width m. Note that the process of learning the target function can be recognized as the process of learning the target optimal $\rho_* = \sum_{i=1}^{4} \delta_{\theta_i}/4$, where δ_x is Dirac delta function at point x and θ_i is the weight of the NN to be reproduced. Although we know that the target function can be represented by four neurons, Fig. 6 shows that using a larger m leads to better learning. This is consistent with the theory of overparameterization. In Fig. 6(a), we use the mean-squared-error loss to measure the difference between the target and optimized NN. The training values are $F_4(x), x \sim N(0, 100^2 I)$ blurred with noise $\epsilon \sim N(0, 0.1^2)$ (the reason to use large scale input is to improve the reconstruction difficulty). It can be seen that, with $m = 4$, we will get stuck at a local minimum and cannot learn the correct target function. When we increase m, we achieve more and more accurate learning of the target function. Fig. 6(b) and (c) shows the distributions of θ at initialization and at the optimal solution when training convergence. We can see that they differ significantly, and thus, in this case, NN training goes out of the NTK regime.

In the end, the distributions of the neurons are scattered, with a large number of neurons become aligned with the target $\{\theta_i\}$ represented by the four red dots. Since the target function does not have a unique representation, therefore, we cannot recover the parameters $\{\theta_i\}$ but only recover the function value represented by each $\{\theta_i\}$ using multiple θ parameters distributed over the lines of the targets. Therefore, we can learn the target function reliably when m is large although we do not necessarily learn the four target parameters $\{\theta_i\}$. This is consistent with the analysis in [47], where the NN training reaches the minimal training error but not necessarily unique. Fig. 6(d) shows that many particles have a very small u, which means that they are "wasted neurons" that do not affect the function value. If we remove these wasted neurons with small u, then we can display the effective neurons in Fig. 6(e), which are well-aligned with the target neuron directions, and they can approximately recover the functions represented by the four target neurons. The phenomenon of wasted neurons in Fig. 6(c) is because the target function is not strongly convex in ρ. Therefore, there can be many solutions that achieve global optimal. The analysis in [47] demonstrates, that under suitable conditions, the training process will converge to the global optimal although the solution may not be unique. However, if we add the entropy regularization as in [29], then the global solution becomes unique. Since the entropy regularization can be implemented using a noisy gradient, we show the effect of using a noisy gradient on this problem in Fig. 6(f). It shows that, with this regularization, there is a significant reduction of wasted neurons, and the final solution is nearly aligned with the directions of the four target neurons. Because the function is not uniquely represented by the four neurons, we still do not recover the four target neurons. Instead, the final solution converges to a unique optimal distribution ρ_*, which is a smooth distribution around the directions of $\{\theta_i\}$ in the continuous limit. In real finite-width NNs, some neurons may still get stuck in the low-density regions. We can, thus, observe a small portion of wasted neurons, which will decrease with wider NNs or larger noises.

We may summarize some key points of the MF approach as follows.

1) The method learns a distribution ρ, which behaves like learning effective feature representations.
2) GD or noisy GD (SGD) over model parameters define gradient flows with dynamics characterized by PDEs.
3) Under appropriate assumptions, the solution of the underlying PDE converges to the optimal solutions in ρ that satisfies the first-order condition (19).
4) The optimal solution can be far from the initial parameter, leading to a more realistic model for NN learning than NTK.

It can be shown that, as $\alpha \to \infty$, the dynamics of MF becomes similar to that of NTK under suitable conditions, and the solution becomes closer and closer to the initialization [29], [57]. When we reduce α, the final solution becomes farther apart from the initialization. This migrates from the NTK regime to the MF regime. This phenomenon is illustrated in Fig. 7. We also summarize the relationship between NTK and MF in Table 1.

Table 1 Comparison of the NTK View and the MF View

NTK regime	MF regime
Large initial parameter	Small initial parameter
Solution close to initialization	Solution far from initialization
Learning model parameters	Learning feature distributions
Without weight-decay regularization	Can incorporate regularization

While MF for two-layer NN is well-understood, compared with NTK, it is significantly more difficult to generalize MF to handle DNN structures. It is also more difficult to obtain concrete complexity results using MF, which requires study both the discretized differential equations, and the convergence rate in terms of letting $m \to \infty$.

VI. IMPORTANCE OF FEATURE LEARNING

In Sections III–V, we presented three mathematical models closely related to two-layer NNs: RF, NTK, and MF. A summary of the pros and cons of the three models is shown in Table 2. The first two approaches, RF and NTK, employ simplified mathematical models by treating two-layer NNs as linear models with random features. The MF view, on the other hand, directly models the feature learning dynamics of NN. It was argued by Fang *et al.* [58] that a theoretical understanding of feature learning is the key to explain the success of NN. Following the argument of Fang *et al.* [58], this section compares the three models empirically from the feature learning perspective.

It was pointed out in [58] that, when m is large, the hidden units of a discrete NN in (1) can be regarded as m (nearly) independent samples from a distribution ρ, which is the distribution of the corresponding continuous NN in (15). If we treat the function value $f(\rho, x)$ of the continuous NN as the target, then it follows that the error of discretize NN is caused by the variance of sampling m hidden units from ρ, which converges to

$$\|f([u,\theta],x) - f(\rho,x)\|_2^2 \approx \frac{1}{m}\int \|\alpha u h(\theta,x) - f(\rho,x)\|_2^2 d\rho(u,\theta) \quad (24)$$

Fig. 7. ***Scaling factor α controls NN behavior: NTK versus MF.***

Table 2 Pros and Cons of RF, NTK, and MF. Note That Though RF and NTK Can Be Applied on DNN, They Are Still a Linear (One-Layer) Model and Do Not Fully Explore the Hierarchical Architecture (See Discussion on Further Directions in Section X)

Properties	RF	NTK	MF
Special initialization	yes	yes	no
Incorporate regularization	yes	no	yes
Random/discriminative feature	random	random	discriminative
Quantitative computational complexity result	yes	yes	open
Applicable on deep neural network	yes	yes	open

where $f([u,\theta],x)$ represents the discrete NN of (1), with each hidden unit j sampled (independently) from ρ, and $f(\rho,x)$ is the corresponding continuous NN of (15).

Since, under suitable conditions, the continuous representation $f(\rho,x)$ can reach a globally optimal solution via training, it can be regarded as the target function that we try to learn with discrete NN. A good feature representation of the target is, thus, a feature distribution ρ so that its continuous NN can be well approximated by the corresponding discrete NN via (24). This means that the variance on the right-hand side of (24) should be small. If we consider using the L_2 regularization for u, and assume that $h(\theta,x)$ is batch-normalized as

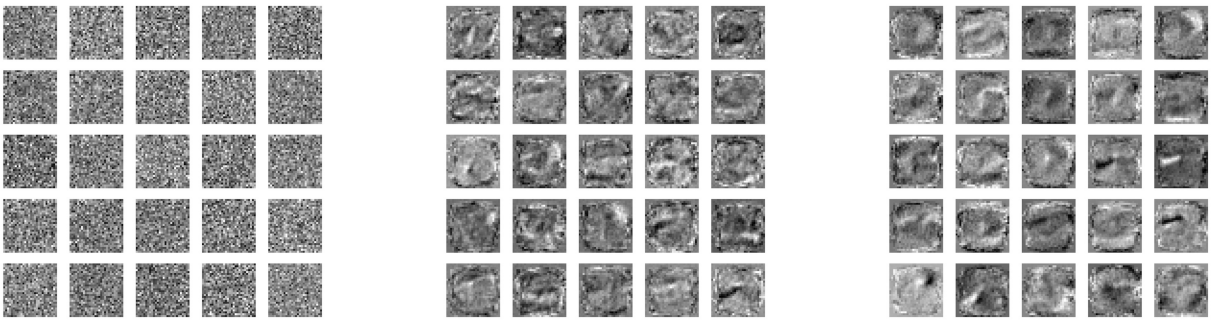
$$\frac{1}{n}\sum_{i=1}^{n} h(\theta,x_i)^2 = 1$$

for all θ, then it is shown in [58] that, when fully optimized, $\|u\|_2$ is nearly a constant with respect to the distribution $\rho(u,\theta)$. It implies that the variance of (24) achieved by NN training is nearly minimized among all values of ρ' such that $f(\rho',x) = f(\rho,x)$. Therefore, for a fixed m, we will achieve the smallest error with discrete NN and the learned probability measure $\rho(u,\theta)$. We, thus, conclude from this result that, after NN training, the discrete NN can efficiently represent the target function by learning an effective feature representation characterized by the feature distribution $\rho(u,\theta)$.

If we compare this learned feature representation to the random feature approaches (RF or NTK), the feature representation learned by NN leads to more efficient discrete representation by sampling from the distribution. This efficiency explains the superiority of NN over the random feature approach. A consequence of the optimal feature representation point of view in [58] is the possibility to use a generative model to learn such a distribution ρ and then use this generative model to replace the initial random features (i.e., random Gaussian distributions) in RF and NTK to generate hidden units of the NN. If we consider the random features sampled from this learned distribution, instead of random features at the initialization, more effective RF and NTK can be obtained. This was illustrated in [58] and [59], which we present here as well.

For convolutional NNs (CNN), the phenomenon of learning features is a consensus among practitioners [59], [60]. A visualization of this phenomenon is shown in Fig. 8. When we use a variational autoencoder (VAE) [61] to learn the optimized ρ distribution from samples of pretrained models, we observe meaningful patterns not found at the initialization. In particular, to obtain samples from ρ_*, we prepare 1000 pretrained two-layer NNs ($m = 100$) with different initializations. Note that the weight of the pretrained NNs can be regarded as samples $\{\theta_i\}$ from ρ_*. We can then use the generative model (VAE) to learn the transform from the standard normal distribution to the target distribution ρ_* with these samples $\{\theta_i\}$.

Fig. 9 illustrates the repopulation phenomenon in NNs. In the repopulation process, we use VAE to learn the feature distribution ρ and then sample weights from the learned generative model. We then fix the generated features and learn parameter u only as in (3), just like the



(a) (b) (c)

Fig. 8. ***Visualization of weights on the MNIST data set (reshaped to 28 × 28). (a) Random. (b) Optimized. (c) Generated.***

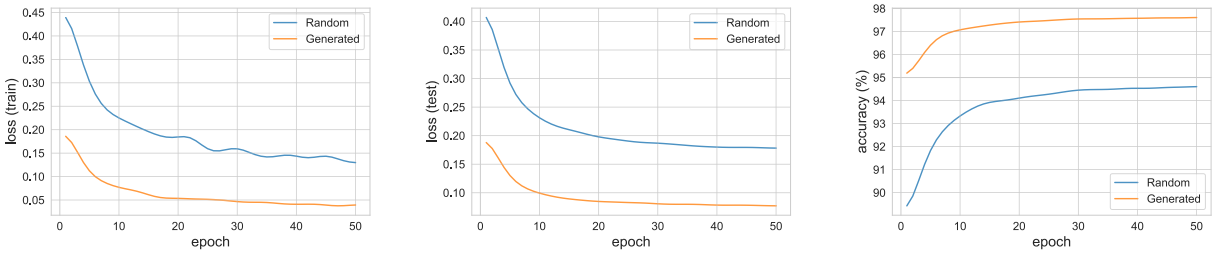


Fig. 9. ***Random feature versus repopulated feature ($m = 10^3$).***

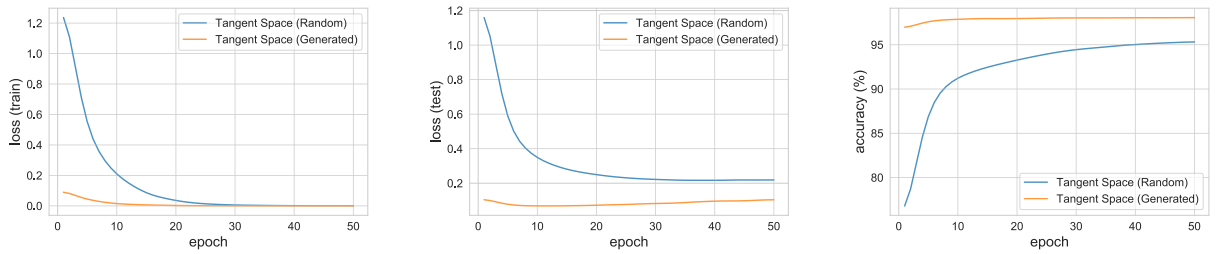


Fig. 10. ***Tangent space comparison ($m = 10^3$).***

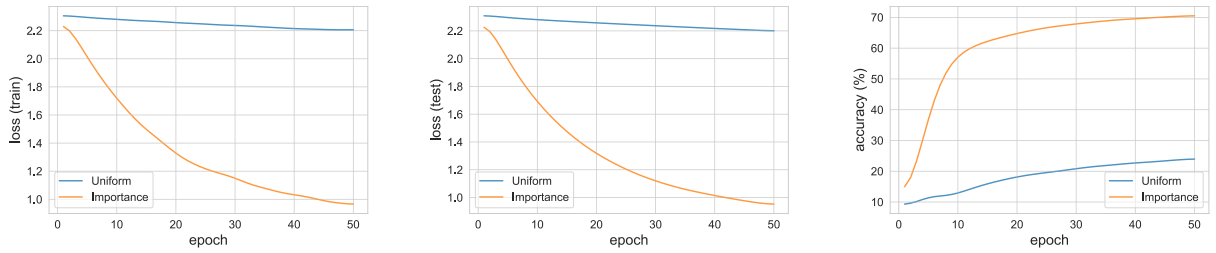


Fig. 11. ***Importance sampling from large NN to remove "wasted" neurons ($m = 10$).***

RF method. From this experiment, we can see that the performance of the repopulated features outperforms that of the initial random features. This means the random features learned by NN are superior to the Gaussian random features at initialization. This is consistent with the theory of [58].

Another approach to examine the effectiveness of ρ is to compare the tangent spaces at the initial and the final solutions using the linear approximation (8). This scenario has also been investigated in [62]. If the representation power of NTK matches that of NN in practice, then the performance using the tangent space at the initialization should be similar to that of the learned distribution ρ. We compare random weights and generated ones in Fig. 10 on the MNIST data set. Note that the generated weights are learned by VAE at the final solution. In training, both approaches achieve very small errors. However, the generalization ability differs significantly: the learned ρ provides a more robust model in the testing stage. Many analyses of NTK investigated the training loss, which can become almost zero due to the effectiveness of tangent space. However, the restricted space cannot perform, as well as the full NN in terms of the generalization ability.

As indicated in Fig. 6(d), many neurons of NN can be "wasted," and it can be identified by u with proper regularization. Therefore, it is possible to perform importance sampling to select effective weights from a very wide NN, which can also be regarded as an approach of pruning. We train a large NN ($m = 10\,000$) with regularization 10^{-3}, which leads to many "wasted" neurons. We want to prune the NN by choosing only ten effective neurons and fine-tune the weight u. There are two strategies to select the neurons: 1) uniform sampling that does not distinguish the importance of neurons and 2) importance sampling that takes the corresponding u as the importance of neurons. After selecting neurons, we fix the first layer θ and train u. Note that the optimization of u is a convex problem. Fig. 11 shows that the performance of importance sampling outperforms uniform sampling significantly. This also confirms that, in a wide network, some neurons may get stuck due to the nonstrong convexity of the formulation.

Since random features of NTK are not learned during training, there have been several works that tried

to investigate the difference between the lazy training condition in NTK and the actual training process of NNs [52], [58], [62]–[65]. Notably, Yehudai and Shamir [63] showed that random features cannot be used to learn even a single ReLU neuron unless the number of the hidden units is exponentially large in d. Ghorbani *et al.* [62] considered the quadratic activation function and showed that GD achieves a lower prediction risk in the actual training process when the number of neurons is small. As discussed earlier, Fang *et al.* [58] showed that, with appropriate regularization, NN can learn optimal feature representations that are superior to random features.

Because MF outperforms NTK in the feature learning perspective, in many cases, better generalization bounds can be obtained for MF than those of NTK. In particular, as shown by Wei *et al.* [52] and Fang *et al.* [58], learning a two-layer NN with an ℓ-2 norm regularizer on the weights is equivalent to solving an ℓ-1 norm regularized problem in the feature space. This is consistent with the empirical observation that MF learns meaningful features because ℓ-1 regularization has a strong capability for feature selection and sparse representation learning. In contrast, the kernel methods typically consider an ℓ-2 norm regularizer. Moreover, in [52], a simple d-dimensional distribution was constructed, for which MF needs $O(d)$ samples to learn. However, kernel methods (including NTK) require at least $\Omega(d^2)$ samples, which demonstrated the superiority of MF in terms of generalization. Recently, Chizat and Bach [66] obtained an interesting result, which shows that, even without a regularizer, GD can implicitly converge to the ℓ-1 norm regularized solution in the MF limit.

VII. OVERPARAMETERIZED DEEP NEURAL NETWORKS

We have explained the concepts of NTK and MF using two-layer NNs. A number of articles have considered extensions of these models to DNNs.

A. NTK

In general, NTK can be generalized to DNNs without much difficulty, e.g., [18], [22], and [24], and the technique can also be generalized to handle more complex topological structures, such as recurrent NNs [21] and residual NNs [18].

In these approaches, with proper initialization, we can linearize the nonlinear NN models at the initialization, similar to what we have done for two-layer NNs. By showing that the training process with small learning rates leads to a zero-training error within a small neighbor of the initialization, the entire NN train lies in the so-called NTK (or lazy-learning) regime, and the linear approximation is effective throughout the training. Similar to the situation of two-layer NN, this requires specialized initialization and specialized learning rates that are often different from what is used by practitioners.

One difficulty with the NTK approach for DNNs is that it cannot satisfactorily explain the benefit of using deeper structures. This because the NTK view essentially corresponds to a linear model using an infinite-dimensional random feature representation that defines the underlying NTK. Although, with deeper structures, we add more and more random features, similar to the situation of the two-layer NNs, these features are not learned.

If we want to apply NTK to real problems, efficient computation of the NTK kernel is necessary, which may require a special design. For example, an efficient exact algorithm to compute convolutional NTK was proposed in [30]. In practice, kernel methods have a quadratic complexity with respect to the number of training data, and the computational cost can be prohibitive for big data applications. Various algorithms have been investigated to alleviate this problem in the traditional kernel learning literature. We refer the readers to [14] and references therein.

B. MF

Unlike NTK, it is nontrivial to generalize MF to DNNs. There were a number of recent works that attempted to generalize MF [57], [66]–[71]. This is still an active research area that has not matured. We will, thus, describe some of the challenges and the latest results.

First, it is not easy to formulate the continuous limit of DNNs. Consider a three-layer NN as an example. The hidden units of the upper layer are functions of hidden units of the lower layer. However, if we allow the number of hidden units of the lower layer to go to infinity (as we do in the two-layer NN), then there are infinitely many features for every hidden unit of the upper hidden layer. If we let the number of hidden units of the upper layer to go to infinity, then there are infinitely many such functions, each with infinitely many features (each feature corresponds to a hidden unit of the lower layer). It is nontrivial to model these functions mathematically. One of the attempted approaches is to model DNNs with nested measures (also known as multilayer measures [72], [73]). However, as mentioned in [67], the mathematical limit may not be well-defined. Another approach considered the continuous limit of DNNs under special conditions. For example, Araújo *et al.* [69] and Nguyen and Pham [70] investigated the continuous limit of DNNs under the initialization that all weights were i.i.d. realizations of a fixed distribution (with finite variance) independent of the number of hidden units. Unfortunately, in such a setting, all neurons in a middle layer will have the same output value at initialization, and this property holds during the entire training process. It is clearly not an appropriate mathematical model for general DNNs. In real applications, initialization strategies, e.g., [45] and [74], sample the NN weights from $\mathcal{N}(0, O(m))$, with variance approaching infinity as the number of hidden nodes m goes to ∞. More recently, Fang *et al.* [71] designed a new MF framework

for DNNs, in which a DNN is represented by probability measures and functions over outputs of the hidden units instead of the NN parameters. This new representation overcomes the degenerate situation exited in some earlier attempts, where all the hidden units essentially have only one meaningful hidden unit in each middle layer.

The second difficulty is that a DNN cannot be regarded as a linear model with respect to the distribution of the parameters. Unlike the case of two-layer NN, which is convex with respect to a reparameterization of the model using the corresponding feature distribution, it is much harder to derive a convex formulation of DNN with appropriate reparameterization. Therefore, the global minimum is hard to be identified, and GD potentially leads to suboptimal solutions. Recently, Nguyen and Pham [70] and Fang *et al.* [71] showed that GD can find a global minimal solution for three-layer and multilayer DNNs, respectively. Notably, they assumed that no regularization is imposed, and the activation function can achieve universal approximation. Under such conditions, the global minimum can be identified as 0. Another remarkable work is [68] and the closely related study [75], in which the authors introduced a new technique, called neural feature repopulation (NFR), to reparameterize the DNNs. Using the NFR technique, one can decouple the distributions of the features from the loss function, and their impact can be integrated into the regularizer. Surprisingly, with suitable regularizers, it can be shown that the overall objective function under the special reparameterization is convex, which is analogous to the case of two-layer NNs. Moreover, they proposed a new optimization process to find the global minimal solution under such regularizers. It remains an open theoretical question to show that GD type of algorithms can find a global optimal solution for the associated convex formulation.

VIII. COMPLEXITY ANALYSIS FOR OVERPARAMETERIZED NNs

The theoretical properties of the linearized system in the NTK view are much easier to analyze. Therefore, it is possible to prove rigorous convergence and statistical complexity bounds under the NTK regime, and polynomial convergence rates can be obtained under various conditions.

For two-layer NNs, for example, by assuming that the minimum eigenvalue of the kernel matrix for the training data is positive, denoted as λ_0, it was shown in [41] that, when the number of hidden units is greater than $n^6\lambda_0^{-4}\delta^{-2}$, with a learning rate of $\eta = O(\lambda_0 n^{-2})$, then, with probability at least $1-\delta$, the GD method finds an ϵ-global minimum in $O(\eta^{-1}\lambda_0^{-1}\log(\epsilon^{-1}))$ steps. Before Du *et al.* [41], Li and Liang [20] studied a different data assumption. They showed that a polynomial convergence rate can be achieved under appropriate separability conditions of the data.

The above results can be generalized to DNNs. For example, Du *et al.* [18] showed that, as long as the number of hidden units is larger than $\tilde{\Omega}(\text{poly}(\lambda_0, n)2^L)$, GD finds a global minimal solution in $\tilde{O}(\text{poly}(\lambda_0, n)2^L)$ steps for standard L-layer DNNs, where $\tilde{\Omega}$ and $\tilde{O}$ hide polylogarithmic terms. Moreover, for the residual NNs, the exponential dependencies on L can be reduced to polynomial dependence. Similarly, Allen-Zhu *et al.* [24] and Zou *et al.* [22] adopted the data assumption in [20] and achieved polynomial complexities.

Some other researchers, e.g., Allen-Zhu and Li [21], Hanin and Nica [76], Bai and Lee [77], and Huang and Yau [78], have tried to model NNs beyond a linear approximation of NTK, typically second-order approximation. For example, Allen-Zhu and Li [21], Bai and Lee [77], and Chen *et al.* [79] proposed a training procedure with randomization techniques to extract the second-order approximation, sharpening complexity bounds. In general, the second-order approximation satisfies the so-called strict saddle property [80] and, thus, is solvable efficiently by saddle-escaping algorithms, e.g., [81]–[83]. Specifically, Bai and Lee [77] showed that complexity bounds for learning polynomials on uniform distributions are lower than those of NTK by a factor of $O(d)$.

IX. OTHER MATHEMATICAL MODELS OF NNs

A number of recent works have considered approximation properties of NNs, leading to a better understanding of why DNNs are superior to shallow networks in terms of function approximation. It is well known that two-layer NNs are universal approximators [84], [85]. However, for certain functions that can be represented by DNNs with a small number of nodes, the exponential number of nodes are needed to represent them with shallow NNs [86], [87]. Related results show that DNNs can represent any function with a constant number of nodes per layer [88], [89], which suggest a tradeoff between the depth and the width in terms of universal approximation. More generally, in order to represent a complex function, we can either increase a network's width or its depth. It was observed, in practice, that it is beneficial to increase both depth and width simultaneously to balance the tradeoff [90].

Before the development of recent mathematical models of overparameterized NNs, such as NTK and MF, which tried to formulate the NN optimization procedure as convex optimization, there were developments in the machine learning research community that focused on the nonconvex optimization aspect of NNs.

In order to understand the NN training process, a number of earlier works studied the loss landscape of NNs. For example, several researchers observed that NN's generalization ability is related to the sharpness/flatness of the local minimal solution resulted from training and discussed different methods to characterize flatness [91]–[93]. There are also works, e.g., [94] and [95], that attempted to understand the Hessian matrices of NNs. With the help of restrictive assumptions, or for specialized models, a number of earlier works, e.g., [96]–[103],

studied the theoretical characterizations of NN landscapes, for example, under the assumption that the input follows Gaussian distribution or the activation function is linear or quadratic. These results, in general, showed that, for any NN that satisfies strict saddle property, standard saddle-escaping algorithms can converge to a global minimal solution. We also refer the readers to the review [104] and the references therein for the global landscape of NNs.

Related studies of NN training were investigated from the generic nonconvex optimization point of view, where the main issue was the complexity of stochastic optimization algorithms, such as SGD to escape saddle points and converge to local minimal solutions [80], [81]. This question was resolved satisfactorily for general nonconvex problems, where both the convergence rate of SGD and that of the optimal stochastic algorithm were known [82], [83], [105].

The kernel representation in the RF/NTK view has a natural connection to Gaussian processes, which has a Bayesian statistics interpretation. The earliest study of overparameterized infinite-width NN was motivated by this Bayesian interpretation [8], [9], [106], where the relationship of infinite-width NN and Gaussian processes was investigated, which is only based on the random feature. More recently, some articles also investigate the kernel form of the NTK regime to perform the Bayesian inference [107], which has a larger function class than NF. Moreover, the Bayesian interpretation can be used to derive uncertainty estimation for NNs. For example, it was argued in [108] using the Gaussian process point of view that dropout [5] can be used to obtain uncertainty estimation for NNs.

X. CONCLUSION

NN has become an essential tool in machine learning and artificial intelligence, with a wide range of applications. Although there have been significant empirical signs of progress, theoretical understanding is rather limited, due to the complexity of the nonconvexity in NN modeling. It has been noted by practitioners that overparameterized NNs are easier to optimize, and the solutions are often reproducible with good performances that are difficult to explain from a nonconvex optimization point of view. To explain this mystery, there have been numerous works to develop mathematical models for overparameterized NNs in recent years. Due to these efforts, we begin to understand how NN works, especially in the continuous limit of overparameterized NNs. Surprisingly, under these models, overparameterized NNs behave more like convex systems, which can explain why they lead to reproducible results observed in practice. There are many research activities in developing better mathematical theories of DNNs. We outline some of the current directions that we feel are particularly promising.

1) For two-layer NNs, NTK can achieve a polynomial computational cost although a relatively weak generalization result. In comparison, MF achieves better generalization but lacks quantitative computational results under general conditions. Therefore, we need a more sophisticated analysis of two-layer NNs showing better generalization behavior than NTK (especially in terms of feature learning), with polynomial computational complexity.
2) The understanding of DNNs is still quite limited. Although it can be shown that GD converges globally in the NTK regime, in the existing analysis, the weight updates can be ignored except for the second to the last layer. This is clearly inconsistent with practice. Moreover, because NTK is effectively a linear (single-layer) model with respect to random features, existing results on DNN approximation imply immediately that representations requiring deep structures cannot be learned efficiently by NTK. As an example, the random features cannot even represent a single ReLU neuron unless there are an exponential number of hidden units [63]. Although this gap can be potentially addressed by the MF view, its analysis is even further behind in which we still do not have a satisfactory theory of MF for DNNs. Even if a satisfactory theory of MF is developed, quantitative results on generalization and optimization remain open. Therefore, for DNNs, both optimization and generalization analysis require further study. As one interesting work in this direction, Allen-Zhu and Li [109] established a principle called "backward feature correction" and showed that GD can learn hierarchical features when the activation function is quadratic. We expect to see more results of this type that can truly illustrate the benefits of deep structures beyond the inherently shallow structure of NTK.
3) The success of NNs has been largely attributed to their abilities to learn discriminative features. Related topics of transfer learning, pretraining, semisupervised learning, and so on have been actively studied by practitioners with great successes. We expect more and more theoretical investigations of these topics in the future, which will inevitably lead to a better understanding of NNs ability to learn feature representations. This may inspire the development of more robust algorithms for representation learning.
4) In recent years, many specialized NN architectures and components are designed by practitioners that are effective in various different tasks. For example, architectures, such as ResNet, CNNs, transformers, and components, such as attention and batch normalization, have become widely used. We start to see the theoretical analysis for such architectures and components. For example, Huang *et al.* [110] provided theoretic justification for Res-Net and showed that Res-Nets generalize better than DNNs by comparing NTKs. We expect that a more sophisticated theoretical analysis of special NN structures can lead to better

understanding and eventually more effective neural architecture design.

5) Practitioners have made many interesting empirical observations of NNs that remain to be explained theoretically. For example, Nakkiran *et al.* [111] showed that NN learning exhibits a double-descent phenomenon, where, when we increase the model size or the number of training epochs, the test performance deteriorates first and then becomes better. In another work, He and Su [112] observed the so-called local elasticity phenomenon where the prediction of a datum x' will not be significantly affected after an SGD update at a datum x, which is not close to x'. As another example, Papyan *et al.* [113] observed a phenomenon called neural collapse, which states that predicted class means collapse to the vertices of a simplex equiangular tight frame at the final training stage. Developing theoretical explanations of such practical phenomena can lead to a better understanding of how NN works.

Finally, we conclude that the study of overparameterized NNs is still in its infancy. We expect that deep mathematical insights obtained from these theoretical investigations will help us to develop solid theoretical foundations for NNs and motivate effective algorithms and architectures in the coming years. ■

Code Availability

Codes for illustration are available at Github.[2]

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[3] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[7] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[8] R. Neal, "Bayesian learning for neural networks," Ph.D. dissertation, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 1995.

[9] C. K. Williams, "Computing with infinite networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 295–301.

[10] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1313–1320.

[11] S. Kleene, "Representation of events in nerve nets and finite automata," *Automata Stud.*, vol. 3, p. 41, 1951.

[12] A. Rahimi and B. Recht, "Uniform approximation of functions with random bases," in *Proc. 46th Annu. Allerton Conf. Commun., Control, Comput.*, 2008, pp. 555–561.

[13] F. Bach, "On the equivalence between kernel quadrature rules and random feature expansions," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 714–751, 2017.

[14] B. Schlkopf, A. J. Smola, and F. Bach, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2018.

[15] A. Daniely, "SGD learns the conjugate kernel class of the network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2422–2430.

[16] A. Daniely, R. Frostig, and Y. Singer, "Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2253–2261.

[17] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8571–8580.

[18] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1675–1685.

[19] L. Chizat, E. Oyallon, and F. Bach, "On lazy training in differentiable programming," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2937–2947.

[20] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8157–8166.

[21] Z. Allen-Zhu and Y. Li, "What can resnet learn efficiently, going beyond kernels?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 9017–9028.

[22] D. Zou, Y. Cao, D. Zhou, and Q. Gu, "Stochastic gradient descent optimizes over-parameterized deep relu networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 467–492.

[23] S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 322–332.

[24] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 242–252.

[25] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6158–6169.

[26] G. Yang, "Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation," 2019, *arXiv:1902.04760*. [Online]. Available: http://arxiv.org/abs/1902.04760

[27] S. Oymak and M. Soltanolkotabi, "Overparameterized nonlinear learning: Gradient descent takes the shortest path?" in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4951–4960.

[28] D. Zou and Q. Gu, "An improved analysis of training over-parameterized deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2055–2064.

[29] S. Mei, T. Misiakiewicz, and A. Montanari, "Mean-field theory of two-layers neural networks: Dimension-free bounds and kernel limit," in *Proc. Annu. Conf. Learn. Theory*, 2019, pp. 2388–2464.

[30] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang, "On exact computation with an infinitely wide neural net," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8141–8150.

[31] B. Tzen and M. Raginsky, "A mean-field theory of lazy training in two-layer neural nets: Entropic regularization and controlled McKean-vlasov dynamics," 2020, *arXiv:2002.01987*. [Online]. Available: http://arxiv.org/abs/2002.01987

[32] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," 2019, *arXiv:1901.00137*. [Online]. Available: http://arxiv.org/abs/1901.00137

[33] Q. Cai, Z. Yang, J. D. Lee, and Z. Wang, "Neural temporal-difference learning converges to global optima," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11315–11326.

[34] B. Liu, Q. Cai, Z. Yang, and Z. Wang, "Neural proximal/trust region policy optimization attains globally optimal policy," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 10565–10576.

[35] L. Wang, Q. Cai, Z. Yang, and Z. Wang, "Neural policy gradient methods: Global optimality and rates of convergence," 2019, *arXiv:1909.01150*. [Online]. Available: http://arxiv.org/abs/1909.01150

[36] Y. Zhang, Q. Cai, Z. Yang, and Z. Wang, "Generative adversarial imitation learning with neural networks: Global optimality and convergence rate," 2020, *arXiv:2003.03709*. [Online]. Available: http://arxiv.org/abs/2003.03709

[37] L. Liao, Y.-L. Chen, Z. Yang, B. Dai, Z. Wang, and M. Kolar, "Provably efficient neural estimation of structural equation model: An adversarial approach," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2020.

[38] L. Su and P. Yang, "On learning over-parameterized neural networks: A functional approximation perspective," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2641–2650.

[39] Y. Cao and Q. Gu, "Generalization bounds of stochastic gradient descent for wide and deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 10836–10846.

[40] Z. Ji and M. Telgarsky, "Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020.

[41] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," in *Proc. Int. Conf. Learn. Represent.* (ICLR), 2019.

[42] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11,

[2] https://github.com/hendrydong/NTK-and-MF-examples

pp. 2278–2324, Nov. 1998.
[43] Y. Li, C. Wei, and T. Ma, "Towards explaining the regularization effect of initial large learning rate in training neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11674–11685.
[44] S. Jastrzkebski *et al.*, "Three factors influencing minima in SGD," in *Proc. Int. Conf. Artif. Neural Netw.*, 2018, pp. 392–402.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.
[46] S. Mei, A. Montanari, and P.-M. Nguyen, "A mean field view of the landscape of two-layer neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 33, pp. E7665–E7671, Aug. 2018.
[47] L. Chizat and F. Bach, "On the global convergence of gradient descent for over-parameterized models using optimal transport," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3036–3046.
[48] J. Sirignano and K. Spiliopoulos, "Mean field analysis of neural networks: A central limit theorem," *Stochastic Processes Their Appl.*, vol. 130, no. 3, pp. 1820–1852, Mar. 2020.
[49] J. Sirignano and K. Spiliopoulos, "Mean field analysis of neural networks: A law of large numbers," *SIAM J. Appl. Math.*, vol. 80, no. 2, pp. 725–752, Jan. 2020.
[50] G. M. Rotskoff and E. Vanden-Eijnden, "Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, p. 22.
[51] X. Dou and T. Liang, "Training neural networks as learning data-adaptive kernels: Provable representation and approximation benefits," *J. Amer. Stat. Assoc.*, pp. 1–14, Apr. 2020.
[52] C. Wei, J. D. Lee, Q. Liu, and T. Ma, "Regularization matters: Generalization and optimization of neural nets v.s. their induced kernel," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 9712–9724.
[53] K. Hu, Z. Ren, D. Siska, and L. Szpruch, "Mean-field Langevin dynamics and energy landscape of neural networks," 2019, *arXiv:1905.07769*. [Online]. Available: http://arxiv.org/abs/1905.07769
[54] Y. Zhang, Q. Cai, Z. Yang, Y. Chen, and Z. Wang, "Can temporal-difference and Q-learning learn representation? A mean-field theory," 2020, *arXiv:2006.04761*. [Online]. Available: http://arxiv.org/abs/2006.04761
[55] A. Engel and C. Van den Broeck, *Statistical Mechanics of Learning*. Cambridge, U.K.: Cambridge Univ. Press, 2001.
[56] L. Ambrosio, N. Gigli, and G. Savaré, *Gradient flows. In Metric Spaces and in the Space of Probability Measures*. Basel, Switzerland: Springer, 2008.
[57] Z. Chen, Y. Cao, Q. Gu, and T. Zhang, "A generalized neural tangent kernel analysis for two-layer neural networks," in *Proc. 34th Conf. Neural Inf. Process. Syst.*, 2020, p. 33.
[58] C. Fang, H. Dong, and T. Zhang, "Over parameterized two-level neural networks can learn near optimal feature representations," 2019, *arXiv:1910.11508*. [Online]. Available: http://arxiv.org/abs/1910.11508
[59] A. Atanov, A. Ashukha, K. Struminsky, D. Vetrov, and M. Welling, "The deep weight prior," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.
[60] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *Univ. Montreal*, vol. 1341, no. 3, p. 1, 2009.
[61] D. P Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114
[62] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, "Limitations of lazy training of two-layers neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 9111–9121.
[63] G. Yehudai and O. Shamir, "On the power and limitations of random features for understanding neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6598–6608.
[64] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, "Linearized two-layers neural networks in high dimension," 2019, *arXiv:1904.12191*. [Online]. Available: http://arxiv.org/abs/1904.12191
[65] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, "When do neural networks outperform kernel methods?" in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2020.
[66] L. Chizat and F. Bach, "Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss," in *Proc. Annu. Conf. Learn. Theory*, 2020, pp. 1305–1338.
[67] J. Sirignano and K. Spiliopoulos, "Mean field analysis of deep neural networks," 2019, *arXiv:1903.04440*. [Online]. Available: http://arxiv.org/abs/1903.04440
[68] C. Fang, Y. Gu, W. Zhang, and T. Zhang, "Convex formulation of overparameterized deep neural networks," 2019, *arXiv:1911.07626*. [Online]. Available: http://arxiv.org/abs/1911.07626
[69] D. Araújo, R. I. Oliveira, and D. Yukimura, "A mean-field limit for certain deep neural networks," 2019, *arXiv:1906.00193*. [Online]. Available: http://arxiv.org/abs/1906.00193
[70] P.-M. Nguyen and H. T. Pham, "A rigorous framework for the mean field limit of multilayer neural networks," 2020, *arXiv:2001.11443*. [Online]. Available: http://arxiv.org/abs/2001.11443
[71] C. Fang, J. D. Lee, P. Yang, and T. Zhang, "Modeling from features: A mean-field framework for over-parameterized deep neural networks," 2020, *arXiv:2007.01452*. [Online]. Available: http://arxiv.org/abs/2007.01452
[72] D. A. Dawson and K. J. Hochberg, "Wandering random measures in the fleming-viot model," *Ann. Probab.*, vol. 10, no. 3, pp. 554–580, Aug. 1982.
[73] D. A. Dawson, "Multilevel mutation-selection systems and set-valued duals," *J. Math. Biol.*, vol. 76, nos. 1–2, pp. 295–378, Jan. 2018.
[74] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[75] Y. Gu, W. Zhang, C. Fang, J. Lee, and T. Zhang, "How to characterize the landscape of overparameterized convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, p. 33.
[76] B. Hanin and M. Nica, "Finite depth and width corrections to the neural tangent kernel," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020.
[77] Y. Bai and J. D. Lee, "Beyond linearization: On quadratic and higher-order approximation of wide neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.
[78] J. Huang and H.-T. Yau, "Dynamics of deep neural networks and neural tangent hierarchy," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4542–4551.
[79] M. Chen *et al.*, "Towards understanding hierarchical learning: Benefits of neural representations," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2020.
[80] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points—Online stochastic gradient for tensor decomposition," in *Proc. Annu. Conf. Learn. Theory*, 2015, pp. 797–842.
[81] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1724–1732.
[82] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 689–699.
[83] C. Fang, Z. Lin, and T. Zhang, "Sharp analysis for nonconvex SGD escaping from saddle points," in *Proc. Annu. Conf. Learn. Theory*, 2019, pp. 1192–1234.
[84] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Netw.*, vol. 6, no. 6, pp. 861–867, Jan. 1993.
[85] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 930–945, May 1993.
[86] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.
[87] M. Telgarsky, "Benefits of depth in neural networks," in *Proc. Annu. Conf. Learn. Theory*, 2016, pp. 1517–1539.
[88] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6231–6239.
[89] B. Hanin, "Universal function approximation by deep neural nets with bounded width and ReLU activations," *Mathematics*, vol. 7, no. 10, p. 992, Oct. 2019.
[90] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[91] S. Hochreiter and J. Schmidhuber, "Flat minima," *Neural Comput.*, vol. 9, no. 1, pp. 1–42, Jan. 1997.
[92] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.
[93] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6389–6399.
[94] V. Papyan, "The full spectrum of deepnet hessians at scale: Dynamics with SGD training and sample size," 2018, *arXiv:1811.07062*. [Online]. Available: http://arxiv.org/abs/1811.07062
[95] B. Ghorbani, S. Krishnan, and Y. Xiao, "An investigation into neural net optimization via hessian eigenvalue density," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2232–2241.
[96] K. Kawaguchi, "Deep learning without poor local minima," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 586–594.
[97] D. Soudry and Y. Carmon, "No bad local minima: Data independent training error guarantees for multilayer neural networks," 2016, *arXiv:1605.08361*. [Online]. Available: http://arxiv.org/abs/1605.08361
[98] M. Hardt and T. Ma, "Identity matters in deep learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.
[99] C. D. Freeman and J. Bruna, "Topology and geometry of half-rectified network optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.
[100] I. Safran and O. Shamir, "Spurious local minima are common in two-layer relu neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4433–4441.
[101] Q. Nguyen and M. Hein, "The loss surface of deep and wide neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2603–2612.
[102] Y. Zhou and Y. Liang, "Critical points of neural networks: Analytical forms and landscape properties," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.
[103] S. S. Du and J. D. Lee, "On the power of over-parametrization in neural networks with quadratic activation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2132–2141.
[104] R. Sun, D. Li, S. Liang, T. Ding, and R. Srikant, "The global landscape of neural networks: An overview," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 95–108, Sep. 2020.
[105] Y. Arjevani, Y. Carmon, J. C. Duchi, D. J. Foster, N. Srebro, and B. Woodworth, "Lower bounds for non-convex stochastic optimization," 2019, *arXiv:1912.02365*. [Online]. Available: http://arxiv.org/abs/1912.02365

[106] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as Gaussian processes," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.
[107] J. Lee *et al.*, "Wide neural networks of any depth evolve as linear models under gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8572–8583.
[108] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.
[109] Z. Allen-Zhu and Y. Li, "Backward feature correction: How deep learning performs deep learning," 2020, *arXiv:2001.04413*. [Online]. Available: http://arxiv.org/abs/2001.04413
[110] K. Huang, Y. Wang, M. Tao, and T. Zhao, "Why do deep residual networks generalize better than deep feedforward networks?—A neural tangent kernel perspective," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2020.
[111] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020.
[112] H. He and W. Su, "The local elasticity of neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.
[113] V. Papyan, X. Y. Han, and D. L. Donoho, "Prevalence of neural collapse during the terminal phase of deep learning training," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 40, pp. 24652–24663, Oct. 2020.

ABOUT THE AUTHORS

Cong Fang received the Ph.D. degree from Peking University, Beijing, China, in 2019.

He is currently a Postdoctoral Researcher with the University of Pennsylvania, Philadelphia, PA, USA. His research interests include machine learning and optimization.

Hanze Dong received the B.Sc. degree in mathematics from Fudan University, Shanghai, China, in 2019. He is currently working toward the Ph.D. degree at The Hong Kong University of Science and Technology, Hong Kong.

His research interests include machine learning algorithms and theory.

Tong Zhang (Fellow, IEEE) received the B.A. degree in mathematics and computer science from Cornell University in 1994, and the Ph.D. degree in computer science from Stanford University in 1999.

He is currently a Chair Professor of Computer Science and Mathematics at The Hong Kong University of Science and Technology. His research interests are machine learning, big data, and their applications.

Dr. Zhang is a Fellow of the American Statistical Association and the Institute of Mathematical Statistics.