A type of generalization error induced by initialization in deep neural networks

Yaoyu Zhang   YAOYU@IAS.EDU
School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540, USA

Zhi-Qin John Xu   XUZHIQIN@SJTU.EDU.CN[*]
School of Mathematical Sciences and Institute of Natural Sciences, Shanghai Jiao Tong University, Shanghai, China,

Tao Luo   LUO196@PURDUE.EDU
Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA

Zheng Ma   MA531@PURDUE.EDU
Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA

Abstract

How initialization and loss function affect the learning of a deep neural network (DNN), specifically its generalization error, is an important problem in practice. In this work, by exploiting the linearity of DNN training dynamics in the NTK regime (Jacot et al., 2018; Lee et al., 2019), we provide an explicit and quantitative answer to this problem. Focusing on regression problem, we prove that, in the NTK regime, for any loss in a general class of functions, the DNN finds the same *global* minima—the one that is nearest to the initial value in the parameter space, or equivalently, the one that is closest to the initial DNN output in the corresponding reproducing kernel Hilbert space. Using these optimization problems, we quantify the impact of initial output and prove that a random non-zero one increases the generalization error. We further propose an antisymmetrical initialization (ASI) trick that eliminates this type of error and accelerates the training. To understand whether the above results hold in general, we also perform experiments for DNNs in the non-NTK regime, which demonstrate the effectiveness of our theoretical results and the ASI trick in a qualitative sense. Overall, our work serves as a baseline for the further investigation of the impact of initialization and loss function on the generalization of DNNs, which can potentially guide and improve the training of DNNs in practice.

Keywords: Deep neural network; Generalization; Initialization; Kernel regime.

1. Introduction

The wide application of deep learning makes it increasingly urgent to establish quantitative theoretical understanding of the learning and generalization behaviors of deep neural networks (DNNs). In this work, we study theoretically the problem of how initialization and loss function quantitatively affect these behaviors of DNNs. Our study focuses on the regression problem, which plays a key role in many applications, e.g., simulation of physical systems (Zhang et al., 2018), prediction of time series (Qiu et al., 2014) and solving differential equations (E and Yu, 2018; Xu et al., 2019). For theoretical analysis, we consider training dynamics of sufficiently-wide DNNs in the NTK (Neural Tangent Kernel) regime, where they can be well approximated by linear gradient flows resembling kernel methods (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019b,a). Note that, theoretical

---

[*] Corresponding author

investigation of such a regime can provide insight into the understanding of general DNNs in practice by the following facts. Heavy overparameterization is one of the key empirical tricks to overcome the learning difficulty of DNNs (Zhang et al., 2016). DNNs in the extremely over-parameterized regime preserve substantive behavior as those in mildly over-parameterized regime. For example, stochastic gradient descent (SGD) can find global minima of the training objective of DNNs which generalizes well to the unseen data (Zhang et al., 2016).

In general, the error of DNN can be classified into three general types (Poggio et al., 2018): approximation error induced by the capacity of the hypothesis set, generalization error induced by the given training data, and training error induced by the given training algorithm. By the universal approximation theorem (Cybenko, 1989) and empirical experiments (Zhang et al., 2016), a large neural network often has the power to express functions of real datasets (small approximation error) and the gradient-based training often can find global minima (zero training error). Generalization error is the main source of error in applications. It can be affected by many factors, such as initialization and loss function as widely observed in experiments. Empirically, a large weight initialization often leads to a large generalization error (Xu et al., 2018, 2019). However, a too small weight initialization makes the training extremely slow. Note that zero initialization leads to a saddle point of DNN which makes the training impossible. Despite above empirically observations, it remains unclear how initialization is related to the generalization error. Regarding the loss function, it is also unclear how it affects the behavior of DNNs.

Technically, all our theoretical results are natural consequences of the linearity of NTK dynamics. Our major contribution lies in exploiting this linearity of DNNs in the NTK regime to answer explicitly and quantitatively the important question of how loss function and initialization affect generalization. As demonstrated by experiments, our quantitative results also qualitatively hold for DNNs in the non-NTK regime, thus shedding light into the interaction between loss, initialization and generalization for general DNNs.

Our key results are summarized as follows.

i) We prove that, for a general class of loss functions, the NTK gradient flow, despite trajectory difference, finds the same *global* minimum.

ii) We prove that the bias induced by a nonzero h_{ini} (initial DNN output) equals to the residual of h_{ini} trained by the same DNN on the same training inputs initialized with $h_{\mathrm{ini}} = 0$. We also show by experiments that this equality approximately holds in non-NTK regime.

iii) We prove that a random h_{ini} leads to a specific type of generalization error that worsens generalization.

iv) We propose an AntiSymmetrical Initialization (ASI) trick for general DNNs that can offset any h_{ini} to zero while keeping the NTK at initialization unchanged. Demonstrated by experiments using general DNNs, this trick accelerates the training and improves generalization.

2. Related works

The impact of loss and initialization for classical linear regression problem has been discussed in Gunasekar et al. (2018). However, such results for deep neural networks are absent. Based on the recent discovery of the NTK regime of DNNs, our work provides quantitative results about loss and initialization, which also sheds light into general DNNs in the non-NTK regime.

There are a series of research on the NTK regime of DNNs (Jacot et al., 2018; Arora et al., 2019a; Du et al., 2018; Zou et al., 2018; Allen-Zhu et al., 2018; E et al., 2019a,b; Sankararaman et al., 2019;

Arora et al., 2019b; Cao and Gu, 2019). Previous works found that the learning of DNNs in the NTK regime is equivalent to kernel ridge regression, such as Mei et al. (2019); Banburski et al. (2019), and the convergence point in the parameter space of a DNN remains close to the initialization, such as Chizat and Bach (2018); Oymak and Soltanolkotabi (2018); Jacot et al. (2018). For the completeness of the paper, we present proofs of the equivalence among problems of NTK gradient flow, minimum Euclidean norm to initialization in parameter space and minimum kernel norm to initial DNN output h_{ini} in the corresponding reproducing kernel Hilbert space (RKHS), which are the foundation of our results.

Previous works (Xu et al., 2018; Xu, 2018b,a; Xu et al., 2019; Rahaman et al., 2018) discover a Frequency-Principle (F-Principle) that DNNs prefer to learn the training data by a low-frequency function. Based on F-principle, Xu et al. (2018, 2019) postulate that the final output of a DNN tends to inherit high frequencies of its initial output that can not be well constrained by the training data (Xu et al., 2018, 2019). Our theoretical and empirical results on initialization provide justification of this postulate.

3. Preliminary

3.1. Notations

Ω : a compact domain of $\mathbb{R}^d$; d: dimension of input of DNN; f: target function, $f \in L^\infty(\Omega)$; M: number of parameter of DNN; n: number of training samples; $\boldsymbol{X}$: inputs of training set $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^\intercal \in \mathbb{R}^{n \times d}$; $\boldsymbol{Y}$: outputs of training set $(y_1, \cdots, y_n)^\intercal \in \mathbb{R}^n$; $\boldsymbol{g}(\boldsymbol{X})$: $(g(\boldsymbol{x}_1), \cdots, g(\boldsymbol{x}_n))^\intercal$ for any function g on Ω; $h(\boldsymbol{x}, \boldsymbol{\theta})$: output of DNN of parameters $\boldsymbol{\theta}$ at $\boldsymbol{x}$; $\nabla_{\boldsymbol{\theta}}(\cdot)$: $(\partial_{\theta_1}(\cdot), \cdots, \partial_{\theta_m}(\cdot))^\intercal$; dist: a general differentiable loss function satisfying conditions (i)–(iii) in Section 3.2; $k(\cdot, \cdot)$: kernel function defined as $\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^\intercal \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)$ if there is no ambiguity; $k_{\boldsymbol{\theta}'}(\cdot, \cdot)$: kernel function defined as $\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}')^\intercal \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}')$; $H_k(\Omega)$: reproducing kernel Hilbert space (RKHS) with respect to kernel k at domain Ω. $\langle \cdot, \cdot \rangle_k$: inner product of space $H_k(\Omega)$; $\|\cdot\|_k$: norm of space $H_k(\Omega)$; $h_k(\boldsymbol{x}; h_{\text{ini}}, \boldsymbol{X}, \boldsymbol{Y})$: the solution of problem (5) depending on kernel k, initial function h_{ini}, inputs $\boldsymbol{X}$ and outputs $\boldsymbol{Y}$ of training set.

3.2. NTK regime of DNN

In the following, we consider the regression problem of fitting the target function $f \in C(\Omega)$, where Ω is a compact domain in $\mathbb{R}^d$. Clearly, $f \in L^p(\Omega)$ for $1 \le p \le \infty$. Specifically, we use a DNN, $h(\boldsymbol{x}, \boldsymbol{\theta}(t)) : \Omega \times \mathbb{R}^m \to \mathbb{R}$, to fit the training dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ of n sampling points, where $\boldsymbol{x}_i \in \Omega$, $y_i = f(\boldsymbol{x}_i)$ for each i. For the convenience of notation, we denote $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^\intercal$, $\boldsymbol{Y} = (y_1, \cdots, y_n)^\intercal$, and $\boldsymbol{g}(\boldsymbol{X}) := (g(\boldsymbol{x}_1), \cdots, g(\boldsymbol{x}_n))^\intercal$ for any function g defined on Ω.

The NTK regime refers to a state of DNN that its NTK defined as

$$k(\cdot, \cdot) = \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}(t))^\intercal \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}(t))$$

almost does not change throughout the training (Jacot et al., 2018; Chizat and Bach, 2018; Arora et al., 2019a). Note that, we have the following requirements for h which are easily satisfied for common DNNs: For any $\boldsymbol{\theta} \in \mathbb{R}^M$, there exists a derivative of $h(\cdot, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ and $\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}) \in C(\Omega; \mathbb{R}^M)$. In the NTK regime, a DNN can be accurately approximated throughout the training as

$$h(\boldsymbol{x}, \boldsymbol{\theta}) \approx h(\boldsymbol{x}, \boldsymbol{\theta}_0) + \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}_0) \cdot (\boldsymbol{\theta} - \boldsymbol{\theta}_0),$$

which is the Taylor expansion of the DNN output function at initialization $\boldsymbol{\theta}_0$ up to first order. Note that, for convenience, we also use $\boldsymbol{\theta}_0$ for $\boldsymbol{\theta}(0)$, i.e., the initial parameter set.

We restrict our work in the the NTK regime of DNNs. It has been shown in Jacot et al. (2018); Chizat and Bach (2018); Lee et al. (2019) that, for any $t \geq 0$, by scaling the initial parameters of layer l of width m_l by $1/\sqrt{m_l}$ as $m_l \to \infty$ for all l, $|\boldsymbol{\theta}(t) - \boldsymbol{\theta}(0)| \to 0$ indicating a NTK regime. Moreover, in Arora et al. (2019a), a non-asymptotic proof is provided that relaxes the infinite width condition to certain sufficiently large width. For simplicity of our analysis, we consider the linearized DNN, i.e.,

$$h(\boldsymbol{x}, \boldsymbol{\theta}) = h(\boldsymbol{x}, \boldsymbol{\theta}_0) + \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}_0) \cdot (\boldsymbol{\theta} - \boldsymbol{\theta}_0). \tag{1}$$

In the following, we refer to DNNs in the NTK regime to the function satisfying the above linearized condition. For the closeness of the linearied model and the DNN trained by gradient descent, We refer readers to Arora et al. (2019a). In the NTK regime, for the loss function (also known as the emprical risk)

$$R_S(\boldsymbol{\theta}) = \text{dist}(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{Y}),$$

where dist is the distance function to be explained in Section 4, the NTK gradient flow, i.e., gradient flow of the linearized model $h(\boldsymbol{x}, \boldsymbol{\theta}(t))$, in parameter space follows

$$\frac{\mathrm{d}\boldsymbol{\theta}(t)}{\mathrm{d}t} = -\nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \nabla_{\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t))} \text{dist}(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t)), \boldsymbol{Y}), \tag{2}$$

with initial value $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$, where $\nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \in \mathbb{R}^{m \times n}$, and $(\nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0))_{ki} = \nabla_{\theta_k} h(\boldsymbol{x}_i, \boldsymbol{\theta}_0)$, and $\nabla_{\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t))} \text{dist}(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t)), \boldsymbol{Y}) \in \mathbb{R}^n$. In function space, the NTK gradient flow yields,

$$\partial_t h(\boldsymbol{x}, t) = -\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}) \nabla_{\boldsymbol{h}(\boldsymbol{X}, t)} \text{dist}(\boldsymbol{h}(\boldsymbol{X}, t), \boldsymbol{Y}), \tag{3}$$

with $h(\boldsymbol{x}, t) = h(\boldsymbol{x}, \boldsymbol{\theta}(t))$, initial value $h(\cdot, 0) = h(\cdot, \boldsymbol{\theta}_0)$, and kernel $k \in C(\Omega \times \Omega)$ defined as

$$k(\cdot, \cdot) = \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0),$$

where $\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0) = (\partial_{\theta_1} h(\cdot, \boldsymbol{\theta}_0), \cdots, \partial_{\theta_m} h(\cdot, \boldsymbol{\theta}_0))^{\mathsf{T}} \in \mathbb{R}^{m \times 1}$, $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}) \in \mathbb{R}^{1 \times n}$ for any $\boldsymbol{x} \in \Omega$. Note that Eq. (3) of h is a closed system. By Jacot et al. (2018), k is symmetric and positive semi-definite. In the following, we may denote $k_{\boldsymbol{\theta}_0}(\cdot, \cdot) = \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)$ when we need to differentiate kernels corresponding to different architectures or different initializations of DNNs.

3.3. Reproducing kernel Hilbert space (RKHS)

The kernel k can induce a RKHS as follows. First, we cite the Mercer's theorem (Mercer (1909)).

Theorem 1 *(Mercer's theorem (Mercer (1909))) Suppose k is a continuous symmetric positive semi-definite kernel. Then there is an orthonormal basis $\{\phi_j\}$ of $L^2(\Omega)$ consisting of eigenfunctions of T_k defined as $[T_k g](\cdot) = \int_\Omega k(\cdot, \boldsymbol{x}) g(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$ such that the corresponding sequence of eigenvalues σ_j is nonnegative. The eigenfunctions corresponding to non-zero eigenvalues are continuous on Ω and k has the representation*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{\infty} \sigma_j \phi_j(\boldsymbol{x}) \phi_j(\boldsymbol{x}'),$$

where the convergence is absolute and uniform.

Then, we can define the RKHS as $H_k(\Omega) := \{g \in L^2(\Omega) \mid \sum_{i=1}^{\infty} \sigma_i^{-1} \langle g, \phi_i \rangle_{L^2(\Omega)}^2 < \infty\}$, and the inner product in $H_k(\Omega)$ is given by

$$\langle f, g \rangle_k = \sum_{i=1}^{\infty} \sigma_i^{-1} \langle f, \phi_i \rangle_{L^2(\Omega)} \langle g, \phi_i \rangle_{L^2(\Omega)},$$

where $\langle g, \phi_i \rangle_{L^2(\Omega)} = \int_{\Omega} g(\boldsymbol{x}) \phi_i(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$. Define $k^{-1}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{\infty} \sigma_i^{-1} \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{x}')$, then the kernel norm of any $g \in H_k(\Omega)$ can be expressed as

$$\|g\|_k = \langle g, g \rangle_k^{1/2} = \left(\int_{\Omega \times \Omega} g(\boldsymbol{x}) g(\boldsymbol{x}') k^{-1}(\boldsymbol{x}, \boldsymbol{x}') \, \mathrm{d}\boldsymbol{x}' \, \mathrm{d}\boldsymbol{x} \right)^{\frac{1}{2}}.$$

$H_k(\Omega)$ satisfies Berlinet and Thomas-Agnan (2004): (i) $\forall \boldsymbol{x} \in \Omega, k(\cdot, \boldsymbol{x}) \in H_k(\Omega)$; (ii) $\forall \boldsymbol{x} \in \Omega$, $\forall f \in H_k, \langle f(\cdot), k(\cdot, \boldsymbol{x}) \rangle_k = f(\boldsymbol{x})$; (iii) $\forall \boldsymbol{x}, \boldsymbol{x}' \in \Omega, \langle k(\cdot, \boldsymbol{x}), k(\cdot, \boldsymbol{x}') \rangle_k = k(\boldsymbol{x}, \boldsymbol{x}')$.

4. Equivalent optimization problems in the NTK regime

For the completeness, we introduce the optimization problems equivalent to the training of DNNs in NTK regime, which appears explicitly or implicitly in previous works (Chizat and Bach, 2018; Oymak and Soltanolkotabi, 2018; Jacot et al., 2018; Mei et al., 2019; Banburski et al., 2019). As introduced in Section 3.2, for the analysis of gradient flow of $h_{\mathrm{DNN}}(\cdot, \boldsymbol{\theta}_{\mathrm{DNN}}(t))$ in the kernel regime, we focus on the gradient flow of its linearized model $h(\cdot, \boldsymbol{\theta}(t))$, i.e., Eqs. ((2), (3)).

We consider the gradient flow under any loss $R_S(\boldsymbol{\theta}) = \mathrm{dist}\,(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{Y})$, where dist is continuously differentiable and satisfies, for any $\boldsymbol{z} \in \mathbb{R}^n$, (i) $\mathrm{dist}(\boldsymbol{z}, \boldsymbol{z}) = 0$; (ii) $\mathrm{dist}(\boldsymbol{z}', \boldsymbol{z})$ attains minimum if and only if $\boldsymbol{z}' = \boldsymbol{z}$. (iii) $\boldsymbol{z}' = \boldsymbol{z}$ if and only if $\nabla_{\boldsymbol{z}'}\mathrm{dist}(\boldsymbol{z}', \boldsymbol{z}) = 0$. For example, $\mathrm{dist}\,(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{Y}) = \frac{1}{n} \sum_{i=1}^{n} |h(\boldsymbol{x}_i, \boldsymbol{\theta}) - y_i|^p$ for any $1 < p < \infty$. By Theorem 5 in Appendix 9, the long time solution $\boldsymbol{\theta}(\infty) = \lim_{t \to \infty} \boldsymbol{\theta}(t)$ of dynamics (2) is equivalent to the solution of the optimization problem

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2, \quad s.t., \quad \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}) = \boldsymbol{Y}. \tag{4}$$

By Theorem 6 in Appendix 9, $h(\boldsymbol{x}, \boldsymbol{\theta}(\infty))$ uniquely solves the optimization problem

$$\min_{h - h_{\mathrm{ini}} \in H_k(\Omega)} \|h - h_{\mathrm{ini}}\|_k, \quad s.t., \quad \boldsymbol{h}(\boldsymbol{X}) = \boldsymbol{Y}, \tag{5}$$

where $h_{\mathrm{ini}}(\boldsymbol{x}) = h(\boldsymbol{x}, \boldsymbol{\theta}_0)$ and the constraints $\boldsymbol{h}(\boldsymbol{X}) = \boldsymbol{Y}$ are in the sense of trace (Evans (2010), pp. 257–261). The above results hold for any initial value $\boldsymbol{\theta}_0$. We refer to *kernel-norm minimization framework* as using the optimization problem (4) or (5) to analyze the long time solution of gradient flow dynamics in (2) or (3), respectively.

With this framework, we emphasize the following results. First, for a finite set of training data, given $\boldsymbol{\theta}_0$, because dist is absent in problems (4) and (5), the output function of a well-trained DNN in the NTK regime is invariant to different choices of loss functions. Note that this result is surprising in the sense that different dist clearly leads to different trajectories of $\boldsymbol{\theta}(t)$ and $h(\cdot, \boldsymbol{\theta}(t))$. Based on this result, it is not necessary to stick to commonly used MSE loss for regression problems. For example, in practice, one can use $\mathrm{dist}\,(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{Y}) = \frac{1}{n} \sum_{i=1}^{n} |h(\boldsymbol{x}_i, \boldsymbol{\theta}) - y_i|^p$ of $1 < p < 2$ to accelerate the training of DNN near convergence or $2 < p < \infty$ to accelerate the training near initialization. One can even mixing different loss functions to further boost the training speed.

Second, among all sets of parameters that fit the training data, a DNN in the NTK regime always finds the one closest to the initialization in the parameter space with respect to the L^2 norm. Third, in the functional space, this framework shows that DNNs always seek to learn a function that has a shortest distance (with respect to the kernel norm) to the initial output function. In the following, we denote $h_k(\boldsymbol{x}; h_{\mathrm{ini}}, \boldsymbol{X}, \boldsymbol{Y})$ as the solution of problem (5) depending on k, h_{ini}, $\boldsymbol{X}$ and $\boldsymbol{Y}$. Remark that, for losses used in classification problems, e.g., cross-entropy, the global minimum is reached as $\boldsymbol{\theta} \to \infty$, resulting a nonlinear training dynamics that cannot be captured by NTK. Therefore, our above results does not extend to these cases.

5. Impact of non-zero initial output

Problems (4) and (5) explicitly incorporate the effect of initialization, thus enabling us to study quantitatively its impact to the learning of DNNs. In this section, we use the above framework to show that a random non-zero initial DNN output leads to a specific type of generalization error. We begin with a relation between the solution with zero initial output and that with non-zero initial output. Proofs of the following theorems can be found in Appendix 10.

Theorem 2 *For a fixed kernel function $k \in C(\Omega \times \Omega)$, and training set $\{\boldsymbol{X}; \boldsymbol{Y}\}$, for any initial function $h_{\mathrm{ini}} \in C(\Omega)$, $h_k(\cdot; h_{\mathrm{ini}}, \boldsymbol{X}, \boldsymbol{Y})$ can be decomposed as*

$$h_k(\cdot; h_{\mathrm{ini}}, \boldsymbol{X}, \boldsymbol{Y}) = h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}) + h_{\mathrm{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X})). \tag{6}$$

This theorem unravels quantitatively the impact of a nonzero initialization, i.e., $h_{\mathrm{ini}} \neq 0$, to the output function of a well-trained DNN in the NTK regime. Comparing the dynamics in (3) of zero and non-zero initialization, at the beginning, the difference of DNN output is h_{ini}, whereas, at the end of the training, that difference shrinks to $h_{\mathrm{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X}))$, which is the residual of fitting h_{ini} sampled at $\boldsymbol{X}$ by the same DNN. Note that Geiger et al. (2019) figures out qualitatively that h_{ini}, which does not vanish as the width of DNN tends to infinity, decreases during the training. However, they do not arrive at a quantitative relation as revealed by Theorem 2.

The expected generalization error of DNN with a random non-zero initial output can be estimated as follows.

Theorem 3 *For a target function $f \in C(\Omega)$, if h_{ini} is generated from an unbiased distribution of random functions μ such that $\mathbb{E}_{h_{\mathrm{ini}} \sim \mu} h_{\mathrm{ini}} = 0$, then the generalization error of $h_k(\cdot; h_{\mathrm{ini}}, \boldsymbol{X}, f(\boldsymbol{X}))$ can be decomposed as follows*

$$\mathbb{E}_{h_{\mathrm{ini}} \sim \mu} R_S \left(h_k(\cdot; h_{\mathrm{ini}}, \boldsymbol{X}, f(\boldsymbol{X})), f \right) = R_S \left(h_k(\cdot; 0, \boldsymbol{X}, f(\boldsymbol{X})), f \right)$$
$$+ \mathbb{E}_{h_{\mathrm{ini}} \sim \mu} R_S \left(h_k(\cdot; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X})), h_{\mathrm{ini}} \right),$$

where $R_S(h_k(\cdot; h_{\mathrm{ini}}, \boldsymbol{X}, f(\boldsymbol{X})), f) = \|h_k(\cdot; h_{\mathrm{ini}}, \boldsymbol{X}, f(\boldsymbol{X})) - f\|_{L^2(\Omega)}^2$.

By the above theorem, $\mathbb{E}_{h_{\mathrm{ini}} \sim \mu} R_S \left(h_k(\cdot; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X})), h_{\mathrm{ini}} \right) \geq 0$ is a specific type of generalization error induced by h_{ini}. Clearly, this error decreases as the sample size n increases and as $n \to \infty$, $h_{\mathrm{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X})) \to 0$, which conforms with our intuition that if the optimization is sufficiently constrained by the training data, then the effect of initialization can be ignored. For real datasets of a limited number of training samples, this error is in general non-zero. By F-Principle (Xu et al., 2018, 2019), DNNs tend to fit training data by low frequency functions. Therefore, qualitatively, $h_{\mathrm{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X}))$ consists mainly of the high frequencies of h_{ini} which cannot be well constrained at $\boldsymbol{X}$.

6. AntiSymmetrical Initialization trick (ASI)

In general, from the Bayesian inference perspective, for fixed k, a random h_{ini} introduces a prior to the inference that is irrelevant to the target function, thus should lower the accuracy of inference. Specifically, the scaling of $1/\sqrt{m_l}$ for NTK leads to a random initial function $h_{\text{ini}} \sim O(1)$, which cannot be neglected for the analysis. To eliminate the negative impact of non-zero initial DNN output, a naive way is to set the initial parameters sufficiently small. However, because $\boldsymbol{\theta} = \mathbf{0}$ is a high order saddle point of $R_S(\boldsymbol{\theta})$, too small initial parameters can lead to a nonlinear training dynamics that cannot be captured by NTK. In practice, small initial parameters can also results in a different initial kernel of DNN with a vanishing magnitude, which slows down the training and changes the learning result. Based on our above theoretical results, we design an AntiSymmetrical Initialization trick (ASI) which can fix the initial output to zero but also keep the kernel invariant. Suppose that the activation function σ is $C^1(\mathbb{R})$. Let $h_i^{[l]}$ be the output of the i-th node of the l-th layer of a L layer DNN. Then, $h_i^{[l]}(\boldsymbol{x}) = \sigma_i^{[l]}(\boldsymbol{W}_i^{[l]} \cdot \boldsymbol{h}^{[l-1]}(\boldsymbol{x}) + b_i^{[l]})$, for $i = 1, \cdots, m_l$. For the i-th neuron of the output layer of DNN, $h_i^{[L]}(\boldsymbol{x}) = \boldsymbol{W}_i^{[L]} \cdot \boldsymbol{h}^{[L-1]} + b_i^{[L]}$. After initializing the DNN by any method, we obtain $h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}(0))$, where

$$\boldsymbol{\theta}(0) = (\boldsymbol{W}^{[L]}(0), \boldsymbol{b}^{[L]}(0), \boldsymbol{W}^{[L-1]}(0), \boldsymbol{b}^{[L-1]}(0), \cdots, \boldsymbol{b}^{[1]}(0)).$$

The ASI for general loss functions is to consider a new DNN expressed as $h_{\text{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(t)) = \frac{\sqrt{2}}{2} h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}(t)) - \frac{\sqrt{2}}{2} h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}'(t))$ where $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \boldsymbol{\theta}')$, $\boldsymbol{\Theta}$ is initialized such that $\boldsymbol{\theta}'(0) = \boldsymbol{\theta}(0)$. In the following, we prove a theorem that ASI trick eliminates the nonzero random prior without changing the kernel k (Proof can be found in Appendix 11).

Theorem 4 *For any general loss function* dist *satisfying the conditions in Sec. 4, in the NTK regime, i.e., $h(\boldsymbol{x}, \boldsymbol{\theta}) = h(\boldsymbol{x}, \theta_0) + \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \theta_0) \cdot (\boldsymbol{\theta} - \boldsymbol{\theta}_0)$, the gradient flow of both $h(\boldsymbol{x}, \boldsymbol{\theta}(t))$ and $h_{\text{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(t))$ follows the kernel dynamics*

$$\partial_t h' = -\boldsymbol{k}(\cdot, \boldsymbol{X}) \nabla_{\boldsymbol{h}(\boldsymbol{X}, t)} \text{dist}\left(\boldsymbol{h}'(\boldsymbol{X}, t), \boldsymbol{Y}\right), \tag{7}$$

with initial value $h'(\cdot, 0) = h_{\text{ini}} = h(\boldsymbol{x}, \boldsymbol{\theta}(0))$ and $h'(\cdot, 0) = 0$, respectively, where $\{\boldsymbol{X}; \boldsymbol{Y}\}$ is the training set, $k(\boldsymbol{x}, \boldsymbol{x}') = k_{\boldsymbol{\theta}_0}(\boldsymbol{x}, \boldsymbol{x}') = \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}_0)^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}', \boldsymbol{\theta}_0)$.

Note that Chizat and Bach (2018) proposes a "doubling trick" to offset the initial DNN output, that is, neurons in the last layer are duplicated, with the new neurons having the same input weights but opposite output weights. By applying the "doubling trick", $h'(\cdot, 0) = 0$. However, the kernel of layers $L - 1$ and L doubles, whereas the kernel of layers $m \leq L - 2$ completely vanishes (See Appendix 12 for the proof), which could have large impact on the training dynamics as well as the generalization performance of DNNs.

7. Experiments

Our above theoretical results are obtained using the linearized model of DNN in Eq. (1) that well approximates the behavior of DNN in the NTK regime. In this section, we will demonstrate experimentally the accuracy of these results for very wide DNNs and the effectiveness of these results for general DNNs. First, using synthetic data, we verify the invariance of DNN output after training to different loss functions as studied in Sec. 4. Then, we verify the linear relation in Eq. (6).
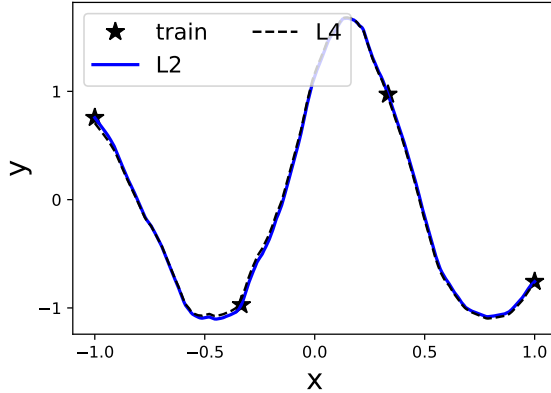
Figure 1: Invariance of DNN output to loss functions. Black stars indicate training data. Blue solid curve and the black curve indicate the outputs on the test samples of the DNN well-trained by L^2 loss and L^4 loss, respectively. The size of DNN is 1-500-500-1. $v_{\text{std}} = 5$. The training and test data are randomly sampled from $\sin(4x)$ in $[-1, 1]$ with sample size 4 and 500, respectively.

Moreover, we demonstrate the effectiveness of the ASI trick on both synthetic data and the MNIST dataset. Here is a summary of the settings of DNNs in our experiments. The activation function is ReLU, parameters are initialized by a Gaussian distribution with mean 0 and standard deviation $v_{\text{std}}\sqrt{2/(m_{\text{in}} + m_{\text{out}})}$, where m_{in} and m_{out} are for the input and the output dimension of the neuron, respectively. For Figs. (1, 2, 3), networks are trained by full gradient descent with MSE loss and the learning rate is 10^{-5}.

7.1. Invariance of DNN output to loss functions

For a DNN $h(\boldsymbol{x}, \boldsymbol{\theta}(t))$ with initialization fixed at certain $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$, we consider its gradient descent training for two loss functions: the L^2 (MSE) loss $\text{dist}(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{Y}) = \frac{1}{n}\sum_{i=1}^{n}(h(\boldsymbol{x}_i, \boldsymbol{\theta}) - y_i)^2$ and the L^4 loss $\text{dist}(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{Y}) = \frac{1}{n}\sum_{i=1}^{n}(h(\boldsymbol{x}_i, \boldsymbol{\theta}) - y_i)^4$. In Fig. 1, as Theorems 5 and 6 predict, the well-trained DNN outputs for these two losses overlap very well not only at 4 training points, but also at all the test points.

7.2. Linear relation and the effectiveness of ASI trick

7.2.1. 1-D SYNTHETIC DATA

In this sub-section, we use 1-d data, which is convenient for visualization, to train DNNs of a large width. As shown in Fig. 2(a), without applying any trick, the original DNN initialized with a large weight learns/interpolates the training data in a fluctuating manner (blue solid). Both the ASI trick (cyan dashed dot) and the "doubling trick" (green dashed) enable the DNN to interpolate the training data in a more "flat" way. As shown by the red dashed curve, the output computed by the right hand side (RHS) of Eq. (6) accurately predicts the final output of the original DNN on test points. In our experiments $h_k(x; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X}))$, $h_k(x; 0, \boldsymbol{X}, \boldsymbol{Y})$, $h_k(x; h_{\text{ini}}, \boldsymbol{X}, \boldsymbol{Y})$ are always obtained using very wide DNNs with or without the ASI trick applied. From Eq. (6), a non-zero initialization adds
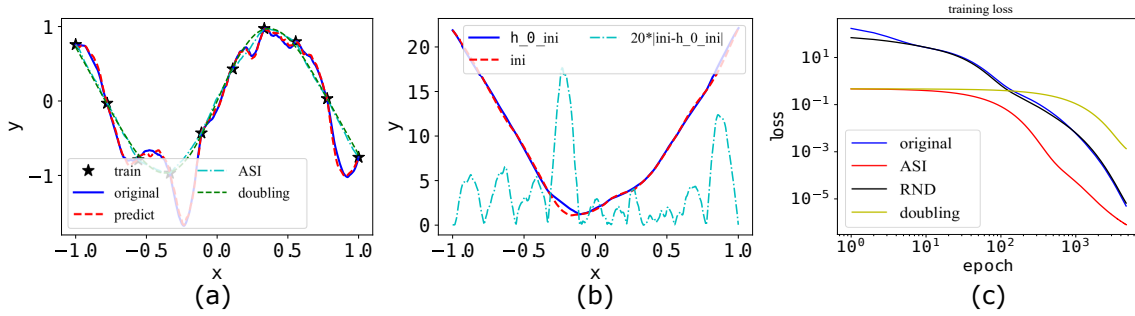
Figure 2: Synthetic data. (a) Black stars indicate the training data. Other curves indicate final outputs of different DNNs evaluated at the test points. Blue solid: the original DNN (without tricks); cyan dashed dot: DNN with ASI trick applied; green dashed: DNN with "doubling trick" applied; red dashed: the RHS of Eq. (6). (b) Blue: $h_k(x; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X}))$; red: h_{ini}; cyan: $20|h_{\mathrm{ini}} - h_k(x; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X}))|$. (c) Evolution of loss functions of different DNNs during the training. Blue: the original DNN; red: DNN with ASI trick applied; black: DNN with RND applied; yellow: DNN with the "doubling trick" applied. The width of the original DNN is 1-5000-5000-1. $v_{\mathrm{std}} = 10$. We sample training and test data randomly from $\sin(4x)$ in $[-1, 1]$ with size 10 and 500, respectively.

a prior $h_{\mathrm{ini}} - h_k(x; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X}))$ to the final DNN output. As shown by the cyan dashed curve in Fig. 2(b), this prior fluctuates a lot, thus, leading to an oscillatory output of DNN after training. Note that this experiment also support the prediction of F-Principle (Xu et al., 2018, 2019) that it is the high frequencies of h_{ini} (red dashed) that remains in the final output of DNN. Concerning the training speed, as shown in Fig. 2(c), the loss function of the DNN with the ASI trick applied decreases much faster than that of the original DNN or the one with the "doubling trick" applied. For a reference, we double the original network similar to the ASI trick, then initialize it randomly following the same distribution as the original. We refer to this trick as RND. As shown by the black curve in Fig. 2(c), the loss function of the DNN with the RND trick applied also decreases much slower than the one with the ASI trick applied.

In summary, for 1-d problem, the linear relation holds well and the ASI trick is effective in removing the artificial prior induced by h_{ini} and accelerating the training speed. In the following, we further investigate the generalization performance of DNN with the ASI trick for real datasets.

7.2.2. BOSTON HOUSE PRICE DATASET

We verify our theoretical results for high dimensional regression problems using Boston house price dataset (Harrison Jr and Rubinfeld, 1978), in which we normalize the value of each property and the price to $[-0.5, 0.5]$. We choose 400 samples as the training data, and the other 106 samples as the test data. As illustrated by the red dots concentrating near the black line of an identity relation in Fig. 3a, the RHS of Eq. (6) well predicts the final output of the original DNN without any trick, which is significant different from the final output of DNN with the ASI trick applied as shown by the blue dots deviating from the black line. As shown in 3b, similar to the experiments on 1-d synthetic data, the ASI trick accelerates the training. In addition, conforming with Theorem 3, the generalization error of the DNN with ASI trick applied is much smaller than that of the original DNN.
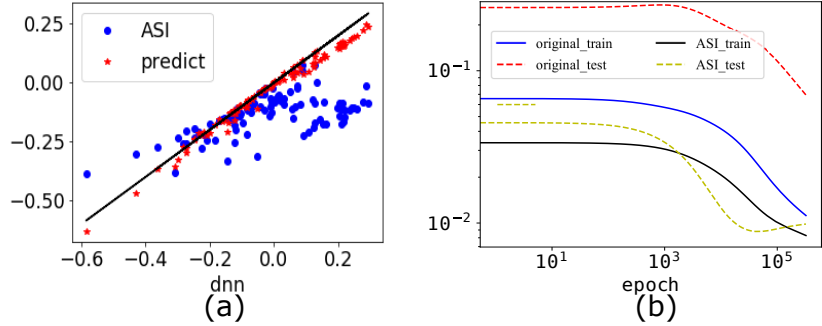
Figure 3: Boston house price dataset. (a) Each dot represents outputs evaluated at one test point. The abscissa is $h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, \boldsymbol{Y})$ obtained using the original DNN. The ordinate for each blue dot is $h_k(x; 0, \boldsymbol{X}, \boldsymbol{Y})$ obtained using DNN with the ASI trick applied, whereas for each red dot is the RHS of Eq. (6). The black line indicates the identity function $y = x$. (b) The evolution of training loss (blue solid) and test loss (red dashed) of the original DNN, and the training loss (black solid) and test loss (yellow dashed) of DNN with ASI trick applied. The width of DNN is 13-100000-1. $v_{\text{std}} = 5$.
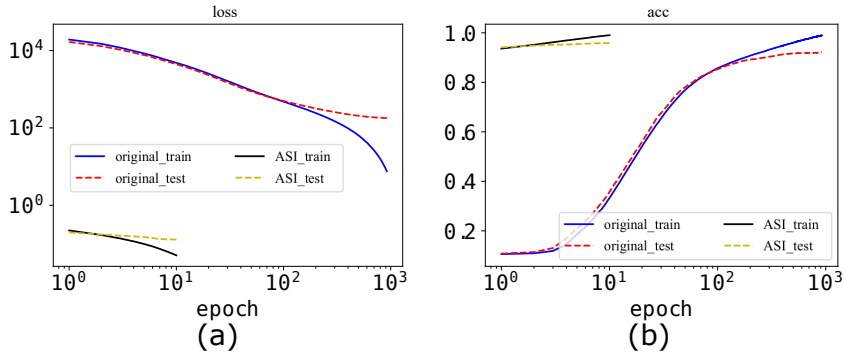

Figure 4: Effectiveness of ASI trick for MNIST dataset in the non-NTK regime of DNN. (a) Evolution of loss functions with the same legend as in Fig. 3(b). (b) Evolution of the corresponding accuracy. The learning rate is 2×10^{-7}. See main text for other settings.

7.2.3. MNIST DATASET AND THE NON-NTK REGIME OF DNN

Next, we use the MNIST dataset to examine the effectiveness of ASI trick in the non-NTK regime of DNNs. We use a DNN with a more realistic setting of width 784-400-400-400-400-10, cross-entropy loss, batch size 512, and Adam optimizer (Kingma and Ba, 2014). In such a case, as shown in Fig. 4, the ASI trick still effectively eliminate h_{ini}, accelerate the training speed and improve the generalization. In Fig. 4(b), with the ASI trick applied, both training and test accuracy exceeds 90% after only 1 epoch of training. This phenomenon further demonstrate that, without the interference of h_{ini}, DNNs can capture very efficiently and accurately the behavior of the training data.

8. Discussion

In this work, focusing on the regression problem, we propose using a kernel-norm minimization framework to study theoretically the role of loss function and initialization for DNNs in the NTK regime. We prove that, given initialization, DNNs of different loss functions in a general class find the same global minimum. Regarding initialization, we find that a non-zero initial output of DNN leads to a specific type of generalization error. We then propose the ASI trick to eliminate this error without changing the neural tangent kernel. Experimentally, we find that ASI trick significantly accelerates the training and improves the generalization performance. Moreover, ASI trick remains effective for classification problems as well as for DNNs in the non-NTK regime. Because the error of DNN output induced by random initialization shrinks during the training, the advantage of ASI trick is much more significant at the early stage of the training. Based on above results, we suggest incorporating ASI trick in the design of controlled experiments for the quantitative study of DNNs.

From the perspective of training flexibility, ASI trick can alleviate the sensitivity of generalization and training speed to different random initializations of DNNs, thus expand the range of well-generalized initializations. This property could be especially helpful for finding a well-generalized solution of a new problem when empirical guidance is not available. We also remarks that, from Eq. (6), a particular prior of h_{ini}, such as the one from meta learning (Rabinowitz, 2019), could decrease the generalization error. However, when meta learning is not available, a zero h_{ini} is in general the best choice for generalization.

Cross-entropy loss is commonly used in classification problems, for which the DNN outputs are often transformed by a softmax function to stay in $(0, 1)$. Theoretically, to obtain a zero cross-entropy loss given that labels of the training data take 1 or 0, weights of the DNN should approach infinity. In such a case, it is impossible for a DNN to stay in the NTK regime, which requires a small variation of weights throughout the training. However, in practice, training of a DNN often stops by meeting certain criteria of training accuracy or validation accuracy. Therefore, it is possible that weights of a sufficiently wide DNN stay in a small neighborhood of the initialization during the training. By setting a proper tolerance for the cross-entropy loss, we will analyze in the future the behavior of DNNs in kernel regime for classification problems with cross-entropy loss.

Acknowledgments

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019a.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019b.

Andrzej Banburski, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Bob Liang, Jack Hidary, and Tomaso Poggio. Theory iii: Dynamics and generalization in deep networks. *arXiv preprint arXiv:1903.04991*, 2019.

A Berlinet and C Thomas-Agnan. Reproducing kernel hilbert spaces in probability and, 2004.

Yuan Cao and Quanquan Gu. A generalization theory of gradient descent for learning over-parameterized deep relu networks. *arXiv preprint arXiv:1902.01384*, 2019.

Lenaic Chizat and Francis Bach. A note on lazy training in supervised differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Weinan E, Chao Ma, Qingcan Wang, and Lei Wu. Analysis of the gradient descent algorithm for a deep neural network model with skip-connections. *arXiv preprint arXiv:1904.05263*, 2019a.

Weinan E, Chao Ma, and Lei Wu. A comparative analysis of the optimization and generalization property of two-layer neural network and random feature models under gradient descent dynamics. *arXiv preprint arXiv:1904.04326*, 2019b.

Lawrence C Evans. Partial differential equations. 2010.

Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *arXiv preprint arXiv:1901.01608*, 2019.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1832–1841, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. *arXiv preprint arXiv:1902.06015*, 2019.

James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.

Samet Oymak and Mahdi Soltanolkotabi. Overparameterized nonlinear learning: Gradient descent takes the shortest path? *arXiv preprint arXiv:1812.10004*, 2018.

T Poggio, K Kawaguchi, Q Liao, B Miranda, L Rosasco, X Boix, J Hidary, and HN Mhaskar. Theory of deep learning iii: the non-overfitting puzzle. Technical report, Technical report, CBMM memo 073, 2018.

Xueheng Qiu, Le Zhang, Ye Ren, Ponnuthurai N Suganthan, and Gehan Amaratunga. Ensemble deep learning for regression and time series forecasting. In *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pages 1–6. IEEE, 2014.

Neil C Rabinowitz. Meta-learners' learning dynamics are unlike learners'. *arXiv preprint arXiv:1905.01320*, 2019.

Nasim Rahaman, Devansh Arpit, Aristide Baratin, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of deep neural networks. *arXiv preprint arXiv:1806.08734*, 2018.

Karthik A Sankararaman, Soham De, Zheng Xu, W Ronny Huang, and Tom Goldstein. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. *arXiv preprint arXiv:1904.06963*, 2019.

Zhi-Qin J Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. *arXiv preprint arXiv:1807.01251*, 2018.

Zhi-Qin John Xu. Frequency principle in deep learning with general loss functions and its potential application. *arXiv preprint arXiv:1811.10146*, 2018a.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

Zhiqin John Xu. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018b.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and E Weinan. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical review letters*, 120 (14):143001, 2018.

Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. *arXiv preprint arXiv:1811.08888*, 2018.

Appendix

9. Theorems for the kernel-norm minimization framework

Theorem 5 *Let $\boldsymbol{\theta}(t)$ be the solution of gradient flow dynamics*

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\theta}(t) = -\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0)\nabla_{\boldsymbol{h}(\boldsymbol{X},\boldsymbol{\theta}(t))}\mathrm{dist}\left(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t)), \boldsymbol{Y}\right) \tag{8}$$

with initial value $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$, where $\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0)$ is a full rank (rank n) matrix of size $m \times n$ with $m > n$. Suppose that the limit $\boldsymbol{\theta}(\infty) = \lim_{t\to\infty}\boldsymbol{\theta}(t)$ exists. Then $\boldsymbol{\theta}(\infty)$ solves the constrained optimization problem

$$\min_{\boldsymbol{\theta}}\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2, \ s.t., \ \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}) = \boldsymbol{Y}. \tag{9}$$

Remark *Compared with the nonlinear gradient flow of DNN, the linearization in Eq. (8) is only performed on the hypothesis function h but not on the loss function or the gradient flow.*

Proof Because $\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0)$ is a full rank matrix,

$$\lim_{t\to\infty} \nabla_{\boldsymbol{h}(\boldsymbol{X},\boldsymbol{\theta}(t))}\mathrm{dist}\left(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t)), \boldsymbol{Y}\right) = \boldsymbol{0}.$$

Recall that $\nabla_{\boldsymbol{z}'}\mathrm{dist}(\boldsymbol{z}', \boldsymbol{z}) = \boldsymbol{0}$ if and only if $\boldsymbol{z}' = \boldsymbol{z}$. Thus

$$\lim_{t\to\infty} \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t)) = \boldsymbol{Y}.$$

By applying singular value decomposition to $\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0)$, we obtain $\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\mathsf{T}$, where $\boldsymbol{V}$ and $\boldsymbol{U}$ are orthonormal matrix of size $m \times m$ and $n \times n$ respectively, $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 \\ \boldsymbol{0} \end{bmatrix}$ of size $m \times n$, where $\boldsymbol{\Sigma}_1$ is a full rank diagonal matrix of size $n \times n$. $\boldsymbol{V}$ can be split into two part as $\boldsymbol{V} = [\boldsymbol{V}_1, \boldsymbol{V}_2]$, where $\boldsymbol{V}_1$ takes the first n columns and $\boldsymbol{V}_2$ takes the last $m - n$ columns of $\boldsymbol{V}$. Then

$$\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\mathsf{T} = [\boldsymbol{V}_1, \boldsymbol{V}_2]\begin{bmatrix} \boldsymbol{\Sigma}_1 \\ \boldsymbol{0} \end{bmatrix}\boldsymbol{U}^\mathsf{T} = \boldsymbol{V}_1\boldsymbol{\Sigma}_1\boldsymbol{U}^\mathsf{T},$$

$$\boldsymbol{V}_2^\mathsf{T}\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) = \boldsymbol{V}_2^\mathsf{T}\boldsymbol{V}_1\boldsymbol{\Sigma}_1\boldsymbol{U}^\mathsf{T} = \boldsymbol{0}.$$

Therefore

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{V}_2^\mathsf{T}\boldsymbol{\theta}(t) = -\boldsymbol{V}_2^\mathsf{T}\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0)^\mathsf{T}\nabla_{\boldsymbol{h}(\boldsymbol{X},\boldsymbol{\theta}(t))}\mathrm{dist}\left(\boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t)), \boldsymbol{Y}\right) = \boldsymbol{0},$$

which leads to

$$\boldsymbol{V}_2^\mathsf{T}\left(\boldsymbol{\theta}(t) - \boldsymbol{\theta}_0\right) = \boldsymbol{0}, \text{ for any } t \geq 0. \tag{10}$$

By Eq. (1), $\lim_{t\to\infty} \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}(t)) = \boldsymbol{Y}$ yields

$$\lim_{t\to\infty} \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0)^\mathsf{T} (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_0) = \boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0),$$

which can be written as

$$\lim_{t\to\infty} \boldsymbol{U}\boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\mathsf{T} (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_0) = \boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0)$$

hence

$$\lim_{t\to\infty} \boldsymbol{V}_1^\mathsf{T} (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_0) = \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right]. \tag{11}$$

Combining Eq. (10) and (11), $\boldsymbol{\theta}(\infty) = \lim_{t\to\infty} \boldsymbol{\theta}(t)$ is uniquely determined as

$$\boldsymbol{V}^\mathsf{T} (\boldsymbol{\theta}(\infty) - \boldsymbol{\theta}_0) = \left[\begin{array}{c} \boldsymbol{V}_1^\mathsf{T} \\ \boldsymbol{V}_2^\mathsf{T} \end{array} \right] (\boldsymbol{\theta}(\infty) - \boldsymbol{\theta}_0)$$

$$= \left[\begin{array}{c} \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right] \\ \boldsymbol{0} \end{array} \right],$$

$$\boldsymbol{\theta}(\infty) - \boldsymbol{\theta}_0 = \boldsymbol{V} \left[\begin{array}{c} \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right] \\ \boldsymbol{0} \end{array} \right] = \boldsymbol{V}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right],$$

which leads to

$$\boldsymbol{\theta}(\infty) = \boldsymbol{V}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right] + \boldsymbol{\theta}_0.$$

On the other hand, by the above analysis, problem (9) can be formulated as

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2 \quad \text{s.t.} \quad \boldsymbol{V}_1^\mathsf{T} (\boldsymbol{\theta} - \boldsymbol{\theta}_0) = \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right].$$

Any $\boldsymbol{\theta}$ satisfies above constraint can be expressed as

$$\boldsymbol{\theta} = \boldsymbol{V}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right] + \boldsymbol{V}_2 \boldsymbol{\xi} + \boldsymbol{\theta}_0,$$

where $\boldsymbol{\xi} \in \mathbb{R}^{m-n}$. Then

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2^2 = \|\boldsymbol{V}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right]\|_2^2 + \|\boldsymbol{V}_2 \boldsymbol{\xi}\|_2^2.$$

Clearly, $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2$ attains minimum if and only if $\boldsymbol{\xi} = \boldsymbol{0}$. Therefore $\boldsymbol{\theta}(\infty) = \boldsymbol{V}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{U}^\mathsf{T} \left[\boldsymbol{Y} - \boldsymbol{h}(\boldsymbol{X}, \boldsymbol{\theta}_0) \right] + \boldsymbol{\theta}_0$ uniquely solves problem (9). ∎

For the proof of Theorem 6, we first introduce the following two lemmas.

Lemma 1 *For any $h' \in H_k(\Omega)$, there exist $\boldsymbol{\theta}' = \langle h'(\cdot), \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0) \rangle_k$ such that $h' = \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}_0)^\mathsf{T} \boldsymbol{\theta}'$.*

Proof For any $h' \in H_k(\Omega)$,

$$\left\langle h'(\cdot), k(\cdot, \boldsymbol{z}) \right\rangle_k = \left\langle h'(\cdot), \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^\mathsf{T} \nabla_{\boldsymbol{\theta}} h(\boldsymbol{z}, \boldsymbol{\theta}_0) \right\rangle_k$$

$$= \left\langle h'(\cdot), \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0) \right\rangle_k^\mathsf{T} \nabla_{\boldsymbol{\theta}} h(\boldsymbol{z}, \boldsymbol{\theta}_0).$$

For $\boldsymbol{\theta}' = \langle h'(\cdot), \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0) \rangle_k$, by the property of reproducing kernel k,

$$h'(\boldsymbol{x}) = \left\langle h'(\cdot), k(\cdot, \boldsymbol{x}) \right\rangle_k = \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}_0)^\mathsf{T} \boldsymbol{\theta}'.$$

∎

Lemma 2 *For any $\boldsymbol{\theta}' \in \mathbb{R}^m$, $\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \boldsymbol{\theta}' \in H_k(\Omega)$.*

Proof By the Mercer's theorem,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{\infty} \sigma_j \phi_j(\boldsymbol{x}) \phi_j(\boldsymbol{x}'),$$

where $\{\phi_j\}_{j=1}^{\infty}$ are orthonormal basis of $L^2(\Omega)$. Suppose that $\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \boldsymbol{\theta}' \notin H_k(\Omega)$.

If for all positive integer j, $\sigma_j \neq 0$, then $\sum_{i=1}^{\infty} \sigma_i^{-1} \langle \phi_j, \phi_i \rangle_{L^2(\Omega)}^2 = \sigma_j^{-1} < \infty$. Thus $\phi_j \in H_k(\Omega)$ for all j. Therefore $H_k(\Omega) = L^2(\Omega)$. This implies that $\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \boldsymbol{\theta}' \in H_k(\Omega)$.

If for some positive integer j_1, $\sigma_{j_1} = 0$ and $\langle \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \boldsymbol{\theta}', \phi_{j_1} \rangle_{L^2(\Omega)} \neq 0$. Then there exists j_2 such that $\left\langle \nabla_{\theta_{j_2}} h(\cdot, \boldsymbol{\theta}_0), \phi_{j_1} \right\rangle_{L^2(\Omega)} \neq 0$, where θ_{j_2} is the j_2-th component of $\boldsymbol{\theta}$. Then

$$\int_{\Omega} \phi_{j_1}(\boldsymbol{x}) k(\boldsymbol{x}, \boldsymbol{x}') \phi_{j_1}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{x}' = \int_{\Omega} \phi_{j_1}(\boldsymbol{x}) \left(\nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}_0)^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}', \boldsymbol{\theta}_0) \right) \phi_{j_1}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{x}'$$
$$= \sum_i \langle \partial_{\theta_i} h(\cdot, \boldsymbol{\theta}_0), \phi_{j_1} \rangle_{L^2(\Omega)}^2$$
$$\geq \left\langle \partial_{\theta_{j_2}} h(\cdot, \boldsymbol{\theta}_0), \phi_{j_1} \right\rangle_{L^2(\Omega)}^2 > 0.$$

However, on the other hand,

$$\int_{\Omega} \phi_{j_1}(\boldsymbol{x}) k(\boldsymbol{x}, \boldsymbol{x}') \phi_{j_1}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{x}' = \int_{\Omega} \phi_{j_1}(\boldsymbol{x}) \sum_{j=1}^{\infty} \sigma_j \phi_j(\boldsymbol{x}) \phi_j(\boldsymbol{x}') \phi_{j_1}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{x}'$$
$$= \sum_j \sigma_j \langle \phi_j, \phi_{j_1} \rangle_{L^2(\Omega)}^2 = \sigma_{j_1} = 0,$$

which leads to an contradiction. Therefore, $\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \boldsymbol{\theta}' \in H_k(\Omega)$. ∎

Theorem 6 *Let $\boldsymbol{\theta}$ be the solution of problem (9), then $h(\boldsymbol{x}, \boldsymbol{\theta})$ uniquely solves the optimization problem*

$$\min_{h - h_{\mathrm{ini}} \in H_k(\Omega)} \|h - h_{\mathrm{ini}}\|_k \quad \text{s.t.} \quad \boldsymbol{h}(\boldsymbol{X}) = \boldsymbol{Y}, \tag{12}$$

where $h_{\mathrm{ini}} = h(\boldsymbol{x}, \boldsymbol{\theta}_0)$ and the constraints $\boldsymbol{h}(\boldsymbol{X}) = \boldsymbol{Y}$ are in the sense of trace (Evans, 2010) if the Hilbert space $H_k(\Omega)$ is good enough for the existence of trace of $\boldsymbol{h}$ at $\boldsymbol{X}$.

Proof By Eq. (1) $h(\boldsymbol{x}, \boldsymbol{\theta}) - h_{\mathrm{ini}} = \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}_0) \cdot (\boldsymbol{\theta} - \boldsymbol{\theta}_0)$. By Lemma 2, $h(\cdot, \boldsymbol{\theta}) - h_{\mathrm{ini}} \in H_k(\Omega)$. For any $h - h_{\mathrm{ini}} \in H_k(\Omega)$, by Lemma 1, for $\boldsymbol{\theta}' = \langle h - h_{\mathrm{ini}}, \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0) \rangle_k$, $h - h_{\mathrm{ini}} = \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}_0)^{\mathsf{T}} \boldsymbol{\theta}'$. Then

$$\|h - h_{\mathrm{ini}}\|_k = \|\nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \boldsymbol{\theta}'\|_k$$
$$= \sqrt{\langle h - h_{\mathrm{ini}}, \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0)^{\mathsf{T}} \boldsymbol{\theta}' \rangle_k}$$
$$= \sqrt{\langle h - h_{\mathrm{ini}}, \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0) \rangle_k^{\mathsf{T}} \boldsymbol{\theta}'}$$
$$= \sqrt{\boldsymbol{\theta}'^{\mathsf{T}} \boldsymbol{\theta}'} = \|\boldsymbol{\theta}'\|_2.$$

By Problem (9), for any $\boldsymbol{\theta}_1 \neq \boldsymbol{\theta} - \boldsymbol{\theta}_0$ that satisfies $h(\boldsymbol{X}, \boldsymbol{\theta}_1 + \boldsymbol{\theta}_0) = \boldsymbol{Y}$, we have

$$\|\boldsymbol{\theta}_1\|_2 > \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2.$$

Then, for problem (12), for any h_1 satisfying $h_1 - h_{\mathrm{ini}} \in H_k(\Omega)$, $h_1(\boldsymbol{X}) = \boldsymbol{Y}$ and $h_1(\boldsymbol{x}) \neq h(\boldsymbol{x}, \boldsymbol{\theta})$, let $\boldsymbol{\theta}_1 = \langle h_1 - h_{\mathrm{ini}}, \nabla_{\boldsymbol{\theta}} h(\cdot, \boldsymbol{\theta}_0) \rangle_k$. Clearly, $\boldsymbol{\theta}_1 \neq \boldsymbol{\theta} - \boldsymbol{\theta}_0$, which leads to

$$\|h_1 - h_{\mathrm{ini}}\|_k = \|\boldsymbol{\theta}_1\|_2 > \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2 = \|h(\boldsymbol{x}, \boldsymbol{\theta}) - h_{\mathrm{ini}}\|_k.$$

Therefore $h(\boldsymbol{x}, \boldsymbol{\theta})$ uniquely solves problem (12). $\blacksquare$

Now, we obtain the equivalence between the long time solution of dynamics (13) and the solution of optimization problem (14) as follows.

Corollary 1 *Let $h(\boldsymbol{x}, t)$ be the solution of dynamics*

$$\frac{\mathrm{d}}{\mathrm{d}t} h(\boldsymbol{x}, t) = -\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}) \nabla_{\boldsymbol{h}(\boldsymbol{X}, t)} \mathrm{dist}\left(\boldsymbol{h}(\boldsymbol{X}, t), \boldsymbol{Y}\right), \tag{13}$$

with $h(\boldsymbol{x}, 0) = h(\boldsymbol{x}, \boldsymbol{\theta}_0)$ for certain $\boldsymbol{\theta}_0$. Then $h(\boldsymbol{x}, \infty)$ uniquely solves optimization problem

$$\min_{h - h_{\mathrm{ini}} \in H_k(\Omega)} \|h - h_{\mathrm{ini}}\|_k, \quad s.t., \quad \boldsymbol{h}(\boldsymbol{X}) = \boldsymbol{Y}. \tag{14}$$

Proof Notice that dynamics (13) is the same as dynamics (3) obtained from (2). Therefore, for $h(\boldsymbol{x}, 0) = h(\boldsymbol{x}, \boldsymbol{\theta}_0)$, $h(\boldsymbol{x}, t) = h(\boldsymbol{x}, \boldsymbol{\theta}(t))$ where $\boldsymbol{\theta}(t)$ is the solution of dynamics (2) with $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$. By Theorem 5 and 6, $h(\boldsymbol{x}, \infty) = h(\boldsymbol{x}, \boldsymbol{\theta}(\infty))$ uniquely solves dynamics (14). $\blacksquare$

10. Impact of non-zero initial output

In this section, we use the above framework to show that a random non-zero initial DNN output leads to a specific type of generalization error. We begin with a lemma showing the linear composition property of the final DNN outputs in the NTK regime.

Lemma 3 *For a fixed kernel function $k \in L^2(\Omega \times \Omega)$, for any two training sets $\{\boldsymbol{X}; \boldsymbol{Y}_1\}$ and $\{\boldsymbol{X}; \boldsymbol{Y}_2\}$, where $\boldsymbol{Y}_1 = (y_1^{(1)}, \cdots, y_n^{(1)})^\intercal$ and $\boldsymbol{Y}_2 = (y_1^{(2)}, \cdots, y_n^{(2)})^\intercal$, the following linear relation holds*

$$h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}_1 + \boldsymbol{Y}_2) = h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}_1) + h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}_2). \tag{15}$$

Proof Let $h_1(\boldsymbol{x}, t), h_2(\boldsymbol{x}, t)$ be the solutions of the gradient flow dynamics with respect to a MSE loss $\mathrm{dist}\left(\boldsymbol{h}(\boldsymbol{X}, t), \boldsymbol{Y}\right) = \frac{1}{2} \sum_{i=1}^n (h(\boldsymbol{x}_i, t) - y_i)^2$

$$\frac{\partial}{\partial t} h(\boldsymbol{x}, t) = -\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}) \left(\boldsymbol{h}(\boldsymbol{X}, t) - \boldsymbol{Y}\right) \tag{16}$$

with training labels $\boldsymbol{Y} = \boldsymbol{Y}_1$ and $\boldsymbol{Y} = \boldsymbol{Y}_2$, respectively, and $h_1(\boldsymbol{x}, 0) = h_2(\boldsymbol{x}, 0) = h_{\mathrm{ini}} = 0$. Then

$$\partial_t \left(h_1 + h_2\right) = -\boldsymbol{k}(\cdot, \boldsymbol{X}) \left(\boldsymbol{h}_1(\boldsymbol{X}, \boldsymbol{\theta}(t)) - \boldsymbol{Y}_1\right) - \boldsymbol{k}(\cdot, \boldsymbol{X}) \left(\boldsymbol{h}_2(\boldsymbol{X}, \boldsymbol{\theta}(t)) - \boldsymbol{Y}_2\right).$$
$$= -\boldsymbol{k}(\cdot, \boldsymbol{X}) \left[\left(\boldsymbol{h}_1 + \boldsymbol{h}_2\right)(\boldsymbol{X}, \boldsymbol{\theta}(t)) - \left(\boldsymbol{Y}_1 + \boldsymbol{Y}_2\right)\right]$$

with initial value $(h_1 + h_2)(\cdot, 0) = 0$. Therefore $h_1 + h_2$ solves dynamics (16) for $\boldsymbol{Y} = \boldsymbol{Y}_1 + \boldsymbol{Y}_2$ and $h_{\text{ini}} = 0$. Then, by Corollary 1, we obtain

$$h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}_1 + \boldsymbol{Y}_2) = h_1(\cdot, \infty) + h_2(\cdot, \infty) = h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}_1) + h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}_2).$$

∎

Using Lemma (3), we obtain the following quantitative relation between the solution with zero initial output and that with non-zero initial output.

Theorem 7 *(Theorem 2 in main text) For a fixed kernel function $k \in C(\Omega \times \Omega)$, and training set $\{\boldsymbol{X}; \boldsymbol{Y}\}$, for any initial function $h_{\text{ini}} \in C(\Omega)$, $h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, \boldsymbol{Y})$ can be decomposed as*

$$h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, \boldsymbol{Y}) = h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}) + h_{\text{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X})). \tag{17}$$

Proof Because $h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, \boldsymbol{Y})$ is the solution of problem (5). Then $h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, \boldsymbol{Y}) - h_{\text{ini}}$ is the solution of problem

$$\min_{h \in H_k(\Omega)} \|h\|_k, \ s.t., \ \boldsymbol{h}(\boldsymbol{X}) = \boldsymbol{Y} - h_{\text{ini}}(\boldsymbol{X}),$$

whose solution is denoted as $h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y} - h_{\text{ini}}(\boldsymbol{X}))$. By Lemma 3,

$$h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y} - h_{\text{ini}}(\boldsymbol{X})) = h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}) - h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X})). \tag{18}$$

Therefore

$$\begin{aligned}
h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, \boldsymbol{Y}) &= h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y} - h_{\text{ini}}(\boldsymbol{X})) + h_{\text{ini}} \\
&= h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}) + h_{\text{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X})).
\end{aligned} \tag{19}$$

∎

The generalization error of DNN contributed by a random initial output can be estimated as follows.

Theorem 8 *(Theorem 3 in main text) For a target function $f \in C(\Omega)$, if h_{ini} is generated from an unbiased distribution of random functions μ such that $\mathbb{E}_{h_{\text{ini}} \sim \mu} h_{\text{ini}} = 0$, then the generalization error of $h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, f(\boldsymbol{X}))$ can be decomposed as follows*

$$\begin{aligned}
\mathbb{E}_{h_{\text{ini}} \sim \mu} R_S\left(h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, f(\boldsymbol{X})), f\right) &= R_S\left(h_k(\cdot; 0, \boldsymbol{X}, f(\boldsymbol{X})), f\right) \\
&\quad + \mathbb{E}_{h_{\text{ini}} \sim \mu} R_S\left(h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X})), h_{\text{ini}}\right),
\end{aligned}$$

where $R_S(h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, f(\boldsymbol{X})), f) = \|h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, f(\boldsymbol{X})) - f\|_{L^2(\Omega)}^2$.

Proof By Theorem 7,

$$\begin{aligned}
&\|h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, f(\boldsymbol{X})) - f\|_{L^2(\Omega)}^2 \\
&= \|[h_k(\cdot; 0, \boldsymbol{X}, f(\boldsymbol{X})) - f] + [h_{\text{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X}))]\|_{L^2(\Omega)}^2 \\
&= \|h_k(\cdot; 0, \boldsymbol{X}, f(\boldsymbol{X})) - f\|_{L^2(\Omega)}^2 + \|h_{\text{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X}))\|_{L^2(\Omega)}^2 \\
&\quad + 2 \langle h_k(\cdot; 0, \boldsymbol{X}, f(\boldsymbol{X})) - f, h_{\text{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X})) \rangle_{L^2(\Omega)}.
\end{aligned} \tag{20}$$

161

Because $\mathbb{E}_{h_{\text{ini}}\sim\mu}h_{\text{ini}} = 0$, by Lemma 3, $\mathbb{E}_{h_{\text{ini}}\sim\mu}\left[h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X}))\right] = \left[h_k(\cdot; 0, \boldsymbol{X}, \mathbb{E}_{h_{\text{ini}}\sim\mu}h_{\text{ini}}(\boldsymbol{X}))\right] = 0$,

$$\mathbb{E}_{h_{\text{ini}}\sim\mu}\left\langle h_k(\cdot; 0, \boldsymbol{X}, f(\boldsymbol{X})) - f, h_{\text{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X}))\right\rangle_{L^2(\Omega)}$$
$$= \left\langle h_k(\cdot; 0, \boldsymbol{X}, f(\boldsymbol{X})) - f, \mathbb{E}_{h_{\text{ini}}\sim\mu}\left[h_{\text{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X}))\right]\right\rangle_{L^2(\Omega)} = 0. \qquad (21)$$

Then we obtain

$$\mathbb{E}_{h_{\text{ini}}\sim\mu}R_S\left(h_k(\cdot; h_{\text{ini}}, \boldsymbol{X}, f(\boldsymbol{X})), f\right) = R_S\left(h_k(\cdot; 0, \boldsymbol{X}, f(\boldsymbol{X})), f\right)$$
$$+ \mathbb{E}_{h_{\text{ini}}\sim\mu}R_S\left(h_k(\cdot; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X})), h_{\text{ini}}\right)$$

∎

11. AntiSymmetrical Initialization trick (ASI)

We design an AntiSymmetrical Initialization trick (ASI) which can make the initial output zero but also keep the kernel invariant. Let $h_i^{[l]}$ be the output of the ith node of the lth layer of a L layer DNN. Then, $h_i^{[l]}(\boldsymbol{x}) = \sigma_i^{[l]}(\boldsymbol{W}_i^{[l]} \cdot \boldsymbol{h}^{[l-1]}(\boldsymbol{x}) + b_i^{[l]})$, for $i = 1, \cdots, m_l$. For the ith neuron of the output layer of DNN $h_i^{[L]}(\boldsymbol{x}) = \boldsymbol{W}_i^{[L]} \cdot \boldsymbol{h}^{[L-1]} + b_i^{[L]}$. After initializing the network with any conventional method, we obtain $h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}(0))$, where

$$\boldsymbol{\theta}(0) = (\boldsymbol{W}^{[L]}(0), \boldsymbol{b}^{[L]}(0), \boldsymbol{W}^{[L-1]}(0), \boldsymbol{b}^{[L-1]}(0), \cdots, \boldsymbol{b}^{[1]}(0)).$$

The ASI for general loss functions is to consider a new DNN with output $h_{\text{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(t)) = \frac{\sqrt{2}}{2}h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}(t)) - \frac{\sqrt{2}}{2}h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}'(t))$ where $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \boldsymbol{\theta}')$, $\boldsymbol{\Theta}$ is initialized such that $\boldsymbol{\theta}'(0) = \boldsymbol{\theta}(0)$. We will prove that ASI trick eliminates the nonzero prior without changing the kernel k.

Theorem 9 *(Theorem 4 in main text) By applying trick ASI to any DNN $h(\boldsymbol{x}, \boldsymbol{\theta}(t))$ initialized by $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$ such that $h_{\text{ini}} = h(\boldsymbol{x}, \boldsymbol{\theta}_0) \neq 0$, we obtain a new DNN $h_{\text{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(t)) = \frac{\sqrt{2}}{2}h(\boldsymbol{x}, \boldsymbol{\theta}_1(t)) - \frac{\sqrt{2}}{2}h(\boldsymbol{x}, \boldsymbol{\theta}_2(t))$ ($\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$) with initial value $\boldsymbol{\theta}_1(0) = \boldsymbol{\theta}_2(0) = \boldsymbol{\theta}_0$. Then, for any general loss function dist, in the NTK regime, i.e., $h(\boldsymbol{x}, \boldsymbol{\theta}) = h(\boldsymbol{x}, \theta_0) + \nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}, \boldsymbol{\theta}_0) \cdot (\boldsymbol{\theta} - \boldsymbol{\theta}_0)$, the evolution of both $h(\boldsymbol{x}, \boldsymbol{\theta}(t))$ and $h_{\text{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(t))$ under gradient flow of both DNNs follows kernel dynamics*

$$\partial_t h' = -\boldsymbol{k}(\cdot, \boldsymbol{X})\nabla_{\boldsymbol{h}(\boldsymbol{X}, t)}\text{dist}\left(\boldsymbol{h}'(\boldsymbol{X}, t), \boldsymbol{Y}\right), \qquad (22)$$

with initial value $h'(\cdot, 0) = h_{\text{ini}}$ and $h'(\cdot, 0) = 0$, respectively, where $\{\boldsymbol{X}; \boldsymbol{Y}\}$ is the training set, $k(\boldsymbol{x}, \boldsymbol{x}') = k_{\boldsymbol{\theta}_0}(\boldsymbol{x}, \boldsymbol{x}') = \nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}, \boldsymbol{\theta}_0)^{\mathsf{T}}\nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}', \boldsymbol{\theta}_0)$.

Proof Clearly, $h(\cdot, \boldsymbol{\theta}(t))$ under gradient flow follows dynamics (22) with initial function $h'(\cdot, 0) = h_{\text{ini}}$. For the evolution of $h_{\text{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(t))$, it is easy to see that it follows dynamics (22) with initial function $h'(\cdot, 0) = 0$ if and only if $h_{\text{ASI}}(\cdot, \boldsymbol{\Theta}(0)) = 0$ and $k_{\boldsymbol{\Theta}_0} = k_{\boldsymbol{\theta}_0}$. By the definition of $k_{\boldsymbol{\Theta}(0)}$,

$$k_{\boldsymbol{\Theta}_0}(\boldsymbol{x}, \boldsymbol{x}') = \nabla_{\boldsymbol{\Theta}}h_{\text{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(0)) \cdot \nabla_{\boldsymbol{\Theta}}h_{\text{ASI}}(\boldsymbol{x}', \boldsymbol{\Theta}(0))$$
$$= \frac{1}{2}\nabla_{\boldsymbol{\theta}_1}h(\boldsymbol{x}, \boldsymbol{\theta}_1(0)) \cdot \nabla_{\boldsymbol{\theta}_1}h(\boldsymbol{x}', \boldsymbol{\theta}_1(0)) + \frac{1}{2}\nabla_{\boldsymbol{\theta}_2}h(\boldsymbol{x}, \boldsymbol{\theta}_2(0)) \cdot \nabla_{\boldsymbol{\theta}_2}h(\boldsymbol{x}', \boldsymbol{\theta}_2(0))$$
$$= \frac{1}{2}\nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}, \boldsymbol{\theta}_0) \cdot \nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}', \boldsymbol{\theta}_0) + \frac{1}{2}\nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}, \boldsymbol{\theta}_0) \cdot \nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}', \boldsymbol{\theta}_0)$$
$$= \nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}, \boldsymbol{\theta}_0) \cdot \nabla_{\boldsymbol{\theta}}h(\boldsymbol{x}', \boldsymbol{\theta}_0) = k_{\boldsymbol{\theta}(0)}.$$

Moreover,

$$h_{\mathrm{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(0)) = \frac{\sqrt{2}}{2}h(\boldsymbol{x}, \boldsymbol{\theta}_1(0)) - \frac{\sqrt{2}}{2}h(\boldsymbol{x}, \boldsymbol{\theta}_2(0))$$

$$= \frac{\sqrt{2}}{2}h(\boldsymbol{x}, \boldsymbol{\theta}_0) - \frac{\sqrt{2}}{2}h(\boldsymbol{x}, \boldsymbol{\theta}_0) = 0.$$

This completes the proof. ∎

12. "doubling trick"

By applying the "doubling trick" (Note that, in Chizat and Bach (2018), there is no bias term in the last layer), we obtain a new network with network parameters $\boldsymbol{\theta}' = (\boldsymbol{W}'^{[L]}, \boldsymbol{W}'^{[L-1]}, \boldsymbol{b}'^{[L-1]}, \cdots, \boldsymbol{b}'^{[1]})$ initialized as $\boldsymbol{W}'^{[L]}(0) = (\boldsymbol{W}^{[L]}(0), -\boldsymbol{W}^{[L]}(0)), \boldsymbol{W}'^{[L-1]}(0) = (\boldsymbol{W}^{[L-1]}(0), \boldsymbol{W}^{[L-1]}(0)), \boldsymbol{b}'^{[L-1]}(0) = (\boldsymbol{b}^{[L-1]}(0), \boldsymbol{b}^{[L-1]}(0))$ and $\boldsymbol{W}'^{[l]}(0) = \boldsymbol{W}^{[l]}(0), \boldsymbol{b}'^{[l]}(0) = \boldsymbol{b}^{[l]}(0)$ for any $l = 1, \cdots, L - 2$.

In general, the kernel can be decomposed as the summation of kernels with respect the tangent space of parameters of the neural network in each layer, that is

$$k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}, \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x}', \boldsymbol{\theta})$$

$$= \sum_{l=1}^{L} \left[\nabla_{\boldsymbol{W}^{[l]}} h(\boldsymbol{x}, \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{W}^{[l]}} h(\boldsymbol{x}, \boldsymbol{\theta}) + \nabla_{\boldsymbol{b}^{[l]}} h(\boldsymbol{x}, \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{b}^{[l]}} h(\boldsymbol{x}, \boldsymbol{\theta}) \right].$$

Theorem 10 *For the DNN initialized by $\boldsymbol{\theta}'$, by applying the "doubling trick", for any $m \leq L - 2$,*

$$k_{\boldsymbol{W}'^{[m]}}(\boldsymbol{x}, \boldsymbol{x}') = 0, \quad k_{\boldsymbol{b}'^{[m]}}(\boldsymbol{x}, \boldsymbol{x}') = 0.$$

For $m = L - 1, L$, and $\boldsymbol{\Theta} = \boldsymbol{W}^{[L-1]}, \boldsymbol{b}^{[L-1]}, \boldsymbol{W}^{[L]}$,

$$k_{\boldsymbol{\Theta}'}(\boldsymbol{x}, \boldsymbol{x}') = 2k_{\boldsymbol{\Theta}}(\boldsymbol{x}, \boldsymbol{x}'),$$

Proof For any $m \leq L - 2$,

$$\nabla_{\boldsymbol{W}'^{[m]}_{i,j}} h'(\boldsymbol{x}, \boldsymbol{\theta}'(0)) = \left(\prod_{l=m+1}^{L-1} \boldsymbol{W}'^{[l+1]}(0) \boldsymbol{s}'^{[l]}(\boldsymbol{x}, 0) \right) \boldsymbol{W}'^{[m+1]}_j(0) s'^{[m]}_j(\boldsymbol{x}, 0) h'^{[m-1]}_i(\boldsymbol{x}, 0),$$

$$\nabla_{\boldsymbol{b}'^{[m]}_j} h'(\boldsymbol{x}, \boldsymbol{\theta}'(0)) = \left(\prod_{l=m+1}^{L-1} \boldsymbol{W}'^{[l+1]}(0) \boldsymbol{s}'^{[l]}(\boldsymbol{x}, 0) \right) \boldsymbol{W}'^{[m+1]}_j(0) s'^{[m]}_j(\boldsymbol{x}, 0),$$

where $s'^{[l]}_i(\boldsymbol{x}, t) = s(\boldsymbol{W}^{[l]}_i(t) \cdot \boldsymbol{h}^{[l-1]}(\boldsymbol{x}) + b^{[l]}_i(t))$, for $i = 1, \cdots, m_l$, $s(\boldsymbol{x}) = \frac{\mathrm{d}\sigma(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}}$. Because

$$\boldsymbol{W}'^{[L]}(0) \boldsymbol{s}'^{[L-1]}(\boldsymbol{x}, 0) = \boldsymbol{W}^{[L]}(0) \boldsymbol{s}^{[L-1]}(\boldsymbol{x}, 0) - \boldsymbol{W}^{[L]}(0) \boldsymbol{s}^{[L-1]}(\boldsymbol{x}, 0) = 0,$$

for any $m \leq L - 2$, we obtain $\nabla_{\boldsymbol{W}'^{[m]}_{i,j}} h'(\boldsymbol{x}, \boldsymbol{\theta}'(0)) = \boldsymbol{0}$ and $\nabla_{\boldsymbol{b}'^{[m]}_j} h'(\boldsymbol{x}, \boldsymbol{\theta}'(0)) = \boldsymbol{0}$, which leads to $k_{\boldsymbol{W}'^{[m]}}(\boldsymbol{x}, \boldsymbol{x}') = 0$ and $k_{\boldsymbol{b}'^{[m]}}(\boldsymbol{x}, \boldsymbol{x}') = 0$. For layer $L - 1$ and L, similarly, we have

$$k_{\boldsymbol{W}'^{[L-1]}}(\boldsymbol{x}, \boldsymbol{x}') = 2k_{\boldsymbol{W}^{[L-1]}}(\boldsymbol{x}, \boldsymbol{x}'),$$

$$k_{\boldsymbol{b}'^{[L-1]}}(\boldsymbol{x}, \boldsymbol{x}') = 2k_{\boldsymbol{b}^{[L-1]}}(\boldsymbol{x}, \boldsymbol{x}'),$$
$$k_{\boldsymbol{W}'^{[L]}}(\boldsymbol{x}, \boldsymbol{x}') = 2k_{\boldsymbol{W}^{[L]}}(\boldsymbol{x}, \boldsymbol{x}').$$

∎

Therefore, by applying the "doubling trick", $h'(\boldsymbol{x}, \boldsymbol{\theta}(0))$ is offset to 0. However, the kernel of layers $L-1$ and L doubles, whereas the kernel of layers $m \le L-2$ completely vanishes, which could have large impact on the training dynamics as well as the generalization performance of DNN output.