Real-Valued Backpropagation is Unsuitable for Complex-Valued Neural Networks

Zhi-Hao Tan, Yi Xie, Yuan Jiang, Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
`{tanzh, xiey, jiangy, zhouzh}@lamda.nju.edu.cn`

Abstract

Recently complex-valued neural networks have received increasing attention due to successful applications in various tasks and the potential advantages of better theoretical properties and richer representational capacity. However, the training dynamics of complex networks compared to real networks remains an open problem. In this paper, we investigate the dynamics of deep complex networks during real-valued backpropagation in the infinite-width limit via neural tangent kernel (NTK). We first extend the Tensor Program to the complex domain, to show that the dynamics of any basic complex network architecture is governed by its NTK under real-valued backpropagation. Then we propose a way to investigate the comparison of training dynamics between complex and real networks by studying their NTKs. As a result, we surprisingly prove that for most complex activation functions, the commonly used real-valued backpropagation reduces the training dynamics of complex networks to that of ordinary real networks as the widths tend to infinity, thus eliminating the characteristics of complex-valued neural networks. Finally, the experiments validate our theoretical findings numerically.

1 Introduction

Recently complex-valued neural networks have been successfully applied to various tasks, such as time-series prediction [Wisdom et al., 2016], computer vision [Trabelsi et al., 2018], signal processing [Yao et al., 2020]. Compared to real-valued neural networks, it is shown that complex networks have the potential to provide richer representational capacity [Arjovsky et al., 2016], faster learning [Danihelka et al., 2016], better motivation and generalization for signal-related tasks [Hirose and Yoshida, 2012, Tygert et al., 2016]. Theoretically, there have been significant advances for complex networks regarding the universal approximation property [Voigtlaender, 2020], critical points [Nitta, 2002], local minima [Nitta, 2013] and separation results [Zhang et al., 2022].

However, training deep complex networks has been challenging because of several non-intuitive analytical properties of complex algebra. Firstly, in practice, we often deal with a real-valued cost function, which is non-analytic with respect to complex-valued parameters. Secondly, Liouville's theorem asserts that every bounded and complex-differentiable function is a constant. Thus almost all activation functions are non-analytic due to the preference for boundedness before the popularity of ReLU [Scardapane et al., 2018]. Moreover, Voigtlaender [2020] has proved that the universal approximation property holds only when using non-holomorphic activation functions.

Due to these reasons, different backpropagation algorithms in the complex domain were independently proposed for non-holomorphic networks [Bassey et al., 2021], mostly by optimizing the real and imaginary components separately. Recently, real-valued backpropagation [Nitta, 1997] is widely used due to the convenience of utilizing the real-valued deep learning library [Arjovsky et al., 2016, Trabelsi et al., 2018, Tan et al., 2020]. It optimized the complex network just like a real network by computing partial derivatives of the cost with respect to the real and imaginary parts separately and achieved state-of-the-art performance.

As a result, a natural and fundamental problem in complex-valued neural networks arises: Do complex networks have a different inductive bias from real networks during training? Do complex networks trained by gradient descent tend to learn different hypotheses from real networks? However, to the best of the authors' knowledge, the training dynamics of backpropagation for complex networks compared to real networks remains open.

In this paper, we investigate the training dynamics of deep complex networks under real-valued backpropagation from neural tangent kernel (NTK) perspective [Jacot et al., 2018], which captures the optimization behavior of neural networks in the infinite-width limit. Then we provide a way to investigate the comparison of training dynamics between complex and real networks via their NTKs. As a result, we obtain informative results which may guide the algorithm selection for complex networks in practice if people want to take full advantage of complex networks.

Our contributions. Our main contributions can be summarized as follows:

- First, we extend the Tensor Program [Yang, 2020] to the complex domain and show that for a complex-valued neural network of any basic architecture in the infinite-width limit, the training dynamics of real-valued backpropagation is determined by kernel gradient descent with a deterministic NTK at initialization.
- Second, we investigate the comparison of training dynamics between complex and real networks based on their NTKs. We surprisingly prove that the commonly used real-valued backpropagation reduces the training dynamics of complex-valued multi-layer perceptrons (MLPs) to that of ordinary real MLPs as the widths tend to infinity, thus eliminating the characteristics of complex-valued neural networks. This result holds for most commonly used complex activation functions, including all split activation functions such as $\mathbb{C}$ReLU, $\mathbb{C}$Sigmoid, $\mathbb{C}$tanh; and part of magnitude-based holomorphic activation functions.
- Finally, we study the results numerically, and the experiments verifies our findings. Specifically, in several settings with different depths and various activation functions, the NTKs of complex networks converge to the NTKs of real networks as the widths grow.

Organization. We start with some preliminaries and notations about complex networks and neural tangent kernels in Section 2. In Section 3, we investigate the NTK of complex-valued neural networks of any architecture during real-valued backpropagation in the infinite-width limit. Section 4 firstly presents the NTK of complex MLPs in any depth and then investigates the conditions that training dynamics of complex-valued MLPs reduce to that of ordinary real MLPs. We verify our results empirically in Section 5. Finally we discuss the related works and conclude the paper. Due to the limited space, all proofs are placed in the appendices.

2 Preliminaries

2.1 Complex-Valued Neural Networks

Without loss of generality, we focus on complex-valued neural networks with real-valued output $f_\theta(\boldsymbol{z}) \in \mathbb{R}^{d_{out}}$ with parameter set $\theta \in \mathbb{C}^p$, input $\boldsymbol{z} \in \mathbb{C}^d$ and $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y}i$ with $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, and trained by real-valued backpropagation algorithm (also called complex-BP or generalized complex BP, see Appendix A for more details), which is conventional in the literature [Arjovsky et al., 2016, Wisdom et al., 2016, Trabelsi et al., 2018, Zhang and Zhou, 2021, Wu et al., 2021, Zhang et al., 2022]. Our analysis can be naturally applied to complex-valued output by decomposing the real and imaginary part of output into two functions, or applied to real-valued input by treating its imaginary part as zero.

For an L-hidden layer complex network, we denote the output of last hidden layer as $\boldsymbol{h}_L \in \mathbb{C}^n$. Without loss of generality, we consider that the output of a complex network with a linear readout layer is achieved via

$$f_\theta(\boldsymbol{z}) = \Re\{\mathbf{W}_{L+1}\boldsymbol{h}_L\}$$

where $\mathbf{W}_{L+1} \in \mathbb{C}^{n \times d_{out}}$ [Wisdom et al., 2016, Zhang et al., 2022]. Note that there are other two common forms of linear readout layer to generate real-valued output, like $f_\theta(\boldsymbol{z}) = \mathbf{W}_{L+1}\Re\{\boldsymbol{h}_L\}$

where the output weight $\mathbf{W}_{L+1} \in \mathbb{R}^{n \times d_{out}}$ [Wu et al., 2021], and $f_\theta(\boldsymbol{z}) = \mathbf{W}_{L+1} \begin{bmatrix} \Re(\boldsymbol{h}_L) \\ \Im(\boldsymbol{h}_L) \end{bmatrix}$ where $\mathbf{W}_{L+1} \in \mathbb{R}^{2n \times d_{out}}$ [Arjovsky et al., 2016]. They can be treated as special cases of our settings.

For a complex-valued neural network $f_\theta(\boldsymbol{z})$, we can always decompose all the complex operations into two-dimensional real-valued operations, denoted as $f_{[\theta_R,\theta_I]}([\boldsymbol{x},\boldsymbol{y}])$ where $\theta_R, \theta_I \in \mathbb{R}^p$ are the real and imaginary parts of all complex parameters respectively. However, it would be erroneous to assume that a complex network is equivalent to an ordinary real-valued neural network, because the operation of complex multiplication limits the degree of freedom [Hirose and Yoshida, 2012].

2.2 Neural Tangent Kernel

For a real-valued deep neural network $f_{\theta_r}(\boldsymbol{x}) \in \mathbb{R}^{d_{out}}$ with parameter set $\theta_r \in \mathbb{R}^p$ and input $\boldsymbol{x} \in \mathbb{R}^d$, its Neural Tangent Kernel (NTK) under gradient descent is defined as

$$\widehat{\Theta}_r\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \left\langle \nabla_{\theta_r} f_{\theta_r}(\boldsymbol{x}), \nabla_{\theta_r} f_{\theta_r}\left(\boldsymbol{x}'\right)\right\rangle \tag{1}$$

which quantifies the functional gradient descent when taking an infinitely small gradient step on a new observation. In case f_{θ_r} corresponds to an infinite width MLP, Jacot et al. [2018] showed that $\widehat{\Theta}_r\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ converges to a limiting kernel $\mathring{\Theta}_r\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ at initialization and remains frozen during training, i.e.,

$$\lim_{n\to\infty} \widehat{\Theta}_r^t\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \lim_{n\to\infty} \widehat{\Theta}_r^0\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \mathring{\Theta}_r\left(\boldsymbol{x}, \boldsymbol{x}'\right) \quad \forall \text{ training time } t,$$

which could give an accurate description of training dynamics with kernel gradient descent trajectory. Thus a infinitely wide neural network is governed by a linear model based on its first order Taylor expansion in the parameter space [Lee et al., 2019].

Tensor Programs. After the original NTK was derived from multi-layer perceptrons, it is soon extended to a variety of network structures including convolution neural networks (CNTK) [Arora et al., 2019], recurrent neural networks (RNTK) [Yang, 2019a, Alemohammad et al., 2020], graph neural networks (GNTK) [Du et al., 2019] and so on. Importantly, Yang [2020], Yang and Littwin [2021] propose NETSOR$^\top$ program, a basic form in Tensor Programs series, and prove that for a real-valued neural network of any architecture that can be represented by NETSOR$^\top$ program language, its NTK converges to a deterministic limit and stays frozen during training in the infinite-width limit.

Neural Tangent Kernel of complex networks. For a complex-valued neural network $f_\theta(\boldsymbol{z})$, also denoted as $f_{[\theta_R,\theta_I]}([\boldsymbol{x},\boldsymbol{y}])$, when it is trained by real-valued backpropagation, the empirical neural tangent kernel is as follows due to that the real and imaginary parts are optimized separately

$$\widehat{\Theta}\left(\boldsymbol{z}, \boldsymbol{z}'\right) = \left\langle \nabla_\theta f_\theta(\boldsymbol{z}), \nabla_\theta f_\theta\left(\boldsymbol{z}'\right)\right\rangle = \left\langle \nabla_{\theta_R} f_{\theta_R}(\boldsymbol{z}), \nabla_{\theta_R} f_{\theta_R}\left(\boldsymbol{z}'\right)\right\rangle + \left\langle \nabla_{\theta_I} f_{\theta_I}(\boldsymbol{z}), \nabla_{\theta_I} f_{\theta_I}\left(\boldsymbol{z}'\right)\right\rangle.$$

We can always rewrite the NTK as $\Theta([\boldsymbol{x},\boldsymbol{y}],[\boldsymbol{x}',\boldsymbol{y}'])$. Note that at each layer of complex networks in both feed-forward and backward procedure, complex matrix multiplication structure leads to numerous interactions and weight sharing between the real and imaginary parts, which makes the analysis of the complex NTK challenging.

3 Complex Tensor Program

In this section, we show that complex-valued neural networks of any basic architecture also have NTK behavior in the infinite-width limit: the training dynamics of real-valued backpropagation is determined by kernel gradient descent with its NTK at initialization. Specifically, we extend the simplified NETSOR$^\top$ program [Yang, 2020] to the complex domain, and propose the basic Complex Tensor Program (CTP). Note that the complex networks are mostly non-holomorphic, thus we do not require the complex tensor program to represent the backward propagation of the complex networks.

Definition 1 (Complex Tensor Program) *Given an initial set $\mathcal{V}$ of random $\mathbb{C}^n$ vectors and a set $\mathcal{W}$ of random $\mathbb{C}^{n\times n}$ complex matrices, a sequence of $\mathbb{C}^n$ vectors is called a complex tensor program if they are recursively generated through one of the following ways:*

ComNonlin *Given $\phi : \mathbb{C}^k \to \mathbb{C}$ and $\boldsymbol{z}^1, \ldots, \boldsymbol{z}^k \in \mathbb{C}^n$, generate $\phi(\boldsymbol{z}^1, \ldots, \boldsymbol{z}^k) \in \mathbb{C}^n$;*

ComMatMul *Given $\mathbf{W} \in \mathbb{C}^{n\times n}$ and $\boldsymbol{z} \in \mathbb{C}^n$, generate $\mathbf{W}\boldsymbol{z} \in \mathbb{C}^n$.*

Obviously, the complex tensor program could represent the forward procedures of all basic complex network architectures, such as the generic feed-forward full-connected complex networks [Nitta, 2004], the complex-valued recurrent neural networks [Wisdom et al., 2016] and complex-valued convolutional neural networks [Trabelsi et al., 2018, Tan et al., 2020].

3.1 Complex network setup

This subsection introduces the settings and assumptions of complex networks considered. Firstly we introduce the required assumption for activation functions of complex networks, which generalizes the assumption in Yang [2020] to the complex domain.

Assumption 2 *We assume that the complex activation function $\phi : \mathbb{C}^k \to \mathbb{C}$ used in the complex networks and its derivative are polynomially-bounded, i.e., $\phi(\boldsymbol{z})$ satisfies that $|\phi(\boldsymbol{z})| \leq C\|\boldsymbol{z}\|^p + c$ for some $C, p, c > 0$ and $\boldsymbol{z} \in \mathbb{C}^k$; so is its derivative.*

It is worth mentioning that numerous activation functions have been proposed to deal with complex-valued representations and basically they satisfy the assumption. For example, the most commonly used complex sigmoid function $\mathbb{C}$Sigmoid [Nitta, 1997, 2004] and complex hyperbolic tangent function $\mathbb{C}$tanh [Nitta, 2002], which apply sigmoid and hyperbolic tangent function respectively to the real part and imaginary part separately; Moreover, the recently proposed ReLU-based complex activation functions like $\mathbb{C}$ReLU [Trabelsi et al., 2018, Tan et al., 2020], zReLU [Guberman, 2016] and modReLU [Arjovsky et al., 2016] also satisfy the assumption.

Complex NTK parametrization. For the complex weights, we initialize each $\mathbf{W} \in \mathcal{W}$ with $\mathbf{W} = \mathbf{A} + \mathbf{B}i = \frac{\sigma_A}{\sqrt{n}}A + \frac{\sigma_B}{\sqrt{n}}Bi$ where $A_{\alpha\beta}, B_{\alpha\beta} \sim \mathcal{N}(0,1)$, which we refer to as *complex NTK parametrization*. Without loss of generality, we set the variances of real and imaginary parts of all layers as σ_A and σ_B respectively in the following paper. It naturally extends the real-valued NTK parametrization [Jacot et al., 2018, Lee et al., 2019] to complex parameters. Note that this parametrization is non-vacuous because many previous works [Nitta, 1997, 2004] choose to initialize the real and imaginary parts separately.

Setup. Consider a complex-valued neural network $f_\theta(\boldsymbol{z})$ with complex NTK parametrization, its feed-forward procedure can be represented by a complex tensor program and complex activation functions all satisfy Assumption 2. Suppose that there is a multivariate Gaussian $\mathcal{N}_\mathcal{V}$ defined on $\mathbb{R}^{2|\mathcal{V}|}$ such that the real and imaginary variables of the initial set of vectors $\mathcal{V}$ are sampled like $\{\Re[q]_\alpha : q \in \mathcal{V}\} \cup \{\Im[q]_\alpha : q \in \mathcal{V}\} \sim \mathcal{N}_\mathcal{V}$ i.i.d. for each coordinate $\alpha \in [n]$. For the output readout matrix W_{L+1}, we also adopt complex NTK parametrization, and it is sampled independently from all other parameters and is not used anywhere else in the interior of the network. Without loss of generality, the network is trained by SGD with batch-size 1 and learning rate 1.

3.2 NTK for any complex network

Theorem 3 (Complex NTK at initialization) *Consider a complex-valued neural network $f_\theta(\boldsymbol{z})$ with above setup, then as its widths go to infinity, its NTK $\widehat{\Theta}(\boldsymbol{z}, \boldsymbol{z}')$ at initialization converges almost surely to a deterministic limiting kernel $\mathring{\Theta}(\boldsymbol{z}, \boldsymbol{z}')$ over any finite set of inputs.*

Corollary 4 (Complex NTK during training) *Consider training a complex-valued neural network $f_\theta(\boldsymbol{z})$ with above setup. At training time t, denote the input sample as $\boldsymbol{z}_t$ and the loss function as $\mathcal{L}_t : \mathbb{R} \to \mathbb{R}$. Suppose $\mathcal{L}_t$ is continuous for all t. Then as widths approach infinity, for any $\boldsymbol{z} \in \mathbb{C}^d$ and training time t, $f_t(\boldsymbol{z})$ converges almost surely to a random variable $\mathring{f}_t(\boldsymbol{z})$ and*

$$\mathring{f}_{t+1}(\boldsymbol{z}) - \mathring{f}_t(\boldsymbol{z}) = -\mathring{\Theta}\left(\boldsymbol{z}, \boldsymbol{z}_t\right) \mathcal{L}_t'\left(\mathring{f}_t\left(\boldsymbol{z}_t\right)\right), \tag{2}$$

where $\mathring{\Theta}\left(\boldsymbol{z}, \boldsymbol{z}_t\right)$ is the limiting NTK of the complex network at initialization.

The proofs of Theorem 3 and Corollary 4 are given in Appendix B. Theorem 3 and Corollary 4 show that for a complex-valued neural network of any architecture trained by real-valued backpropagation, its NTK at initialization converges to a deterministic limiting kernel in the infinite-width limit, and the training dynamics is determined by kernel gradient descent with the NTK at initialization.

4 Comparison of training dynamics between complex and real networks

In this section, we focus on the important problem: when will complex networks have different inductive bias during training from real networks? The problem can not be solved unless the training dynamics of complex networks could be captured. Based on the NTK theory obtained in Section 3, we could provide a way to compare the training dynamics of complex and real networks by comparing their NTKs. Specifically, we have derived the NTK formula of complex multi-layer perceptrons (MLPs) and investigated the conditions under which complex MLPs trained by real-valued backpropagation will reduce to ordinary real MLPs in the infinite-width limit.

4.1 Neural Tangent Kernel of complex multi-layer perceptrons

This subsection presents the NTK formula of the most generic complex network, i.e., the L-hidden layer complex MLPs.

The network performs the following computation at layer $l \in [1, L]$

$$\boldsymbol{h}_l = \boldsymbol{s}_l + \boldsymbol{r}_l i = \phi(\mathbf{W}_l \boldsymbol{h}_{l-1}) = \phi\left((\mathbf{A}_l + \mathbf{B}_l i)(\boldsymbol{s}_{l-1} + \boldsymbol{r}_{l-1} i)\right), \tag{3}$$

where $\mathbf{W}_l = \mathbf{A}_l + \mathbf{B}_l i$ is the complex weight matrix and $\boldsymbol{h}_l = \boldsymbol{s}_l + \boldsymbol{r}_l i$ with $\boldsymbol{h}_l \in \mathbb{C}^n$ is the output of l-th layer. For the first layer, we set $\boldsymbol{s}_0 = \boldsymbol{x}$ and $\boldsymbol{r}_0 = \boldsymbol{y}$ where $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y} i$ and $\boldsymbol{z} \in \mathbb{C}^d$. Besides, the output of the complex network with a linear read-out layer is achieved via $f_\theta(\boldsymbol{x}) = \mathrm{Re}\{\mathbf{W}_{L+1}\boldsymbol{h}_L\}$ where $\mathbf{W}_{L+1} \in \mathbb{C}^{n\times d_{out}}$. Suppose all activation functions ϕ satisfy the Assumption 2. Complex NTK parametrization is applied for all complex parameters $\mathbf{W}_1 \in \mathbb{C}^{d\times n}$, $\mathbf{W}_{L+1} \in \mathbb{C}^{n\times d_{out}}$ and $\mathbf{W}_l \in \mathbb{C}^{n\times n}$ for $l \in [2, L]$.

We denote the real part of the l-th hidden layer pre-activation as $\boldsymbol{\alpha}_l(\boldsymbol{z})$ and the imaginary part as $\boldsymbol{\beta}_l(\boldsymbol{z})$. In the feed-forward procedure, we denote the covariance kernel functions between the real and imaginary part of the pre-activations respectively as

$$\Sigma^l_\alpha(\boldsymbol{z}, \boldsymbol{z}') = \mathop{\mathbb{E}}_{\theta\sim\mathcal{N}}\Big[\boldsymbol{\alpha}_l(\boldsymbol{z})^\top \boldsymbol{\alpha}_l(\boldsymbol{z}')/n\Big], \qquad \Sigma^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}') = \mathop{\mathbb{E}}_{\theta\sim\mathcal{N}}\Big[\boldsymbol{\alpha}_l(\boldsymbol{z})^\top \boldsymbol{\beta}_l(\boldsymbol{z}')/n\Big], \tag{4}$$

$$\Sigma^l_\beta(\boldsymbol{z}, \boldsymbol{z}') = \mathop{\mathbb{E}}_{\theta\sim\mathcal{N}}\left[\boldsymbol{\beta}_l(\boldsymbol{z})^\top \boldsymbol{\beta}_l(\boldsymbol{z}')/n\right], \qquad \Sigma^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}') = \mathop{\mathbb{E}}_{\theta\sim\mathcal{N}}\left[\boldsymbol{\beta}_l(\boldsymbol{z})^\top \boldsymbol{\alpha}_l(\boldsymbol{z}')/n\right]. \tag{5}$$

In the backward procedure, we denote the gradient vector of the real part of the l-th hidden layer pre-activation as $\boldsymbol{\delta}^l_\alpha(\boldsymbol{z}) := \sqrt{n}\left(\nabla_{\boldsymbol{\alpha}_l(\boldsymbol{z})} f_\theta(\boldsymbol{z})\right)$ and that of the imaginary part as $\boldsymbol{\delta}^l_\beta(\boldsymbol{z}) := \sqrt{n}\left(\nabla_{\boldsymbol{\beta}_l(\boldsymbol{z})} f_\theta(\boldsymbol{z})\right)$. Similarly, we denote the covariance kernel functions of the gradient vector between the real and imaginary part of the pre-activations respectively as

$$\Pi^l_\alpha(\boldsymbol{z}, \boldsymbol{z}') = \mathop{\mathbb{E}}_{\theta\sim\mathcal{N}}\left[\boldsymbol{\delta}^l_\alpha(\boldsymbol{z})^\top \boldsymbol{\delta}^l_\alpha(\boldsymbol{z}')/n\right], \qquad \Pi^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}') = \mathop{\mathbb{E}}_{\theta\sim\mathcal{N}}\left[\boldsymbol{\delta}^l_\alpha(\boldsymbol{z})^\top \boldsymbol{\delta}^l_\beta(\boldsymbol{z}')/n\right], \tag{6}$$

$$\Pi^l_\beta(\boldsymbol{z}, \boldsymbol{z}') = \mathop{\mathbb{E}}_{\theta\sim\mathcal{N}}\left[\boldsymbol{\delta}^l_\beta(\boldsymbol{z})^\top \boldsymbol{\delta}^l_\beta(\boldsymbol{z}')/n\right], \qquad \Pi^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}') = \mathop{\mathbb{E}}_{\theta\sim\mathcal{N}}\left[\boldsymbol{\delta}^l_\beta(\boldsymbol{z})^\top \boldsymbol{\delta}^l_\alpha(\boldsymbol{z}')/n\right]. \tag{7}$$

Theorem 5 *For a L-hidden layer complex MLP, with all activation functions satisfying Assumption 2, in the limit as all widths $n \to \infty$, the empirical NTK at initialization converges to the following limiting kernel*

$$\lim_{n\to\infty} \widehat{\Theta}(\boldsymbol{z}, \boldsymbol{z}') = \mathring{\Theta}(\boldsymbol{z}, \boldsymbol{z}') = \Theta(\boldsymbol{z}, \boldsymbol{z}') \otimes \mathbf{I}_{d_{out}} \tag{8}$$

where

$$\Theta(\boldsymbol{z}, \boldsymbol{z}') = \sum_{l=1}^{L} \big(\Pi^l_\alpha(\boldsymbol{z}, \boldsymbol{z}')\Sigma^l_\alpha(\boldsymbol{z}, \boldsymbol{z}') + \Pi^l_\beta(\boldsymbol{z}, \boldsymbol{z}')\Sigma^l_\beta(\boldsymbol{z}, \boldsymbol{z}') \tag{9}$$

$$+\Pi^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}')\Sigma^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}') + \Pi^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}')\Sigma^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}')\big) + \Sigma^{L+1}_\alpha(\boldsymbol{z}, \boldsymbol{z}') \tag{10}$$

where the covariance functions $\Sigma^l_\alpha, \Sigma^l_\beta, \Pi^l_\alpha, \Pi^l_\beta$ are defined in Eq. 4-7.

The result is proved in the Appendix C, where the detailed recursions of intermediate kernels for the NTK calculation are also presented.

Note that the NTK formula of a complex MLP looks very different from the NTK of a real MLP given by Jacot et al. [2018] due to the existence of interaction between real and imaginary parts, which is caused by joint weight sharing in complex matrix multiplication. However, if we go deeper, does there exist situations that the NTK of a complex MLP will reduce to that of a real MLP?

4.2 Asymptotic equivalence of training dynamics

In this subsection, we provide our main results. Surprisingly, we show that, for commonly used complex activation functions, complex networks trained by real-valued backpropagation have the same inductive bias as real networks during training in the infinite-width limit.

We first define asymptotic equivalence between neural networks, which represents a perspective to investigate when will a complex network have different inductive bias from a real network during training. It also helps if we change the network structure or backpropagation algorithms.

Definition 6 *Two neural networks trained by gradient descent are asymptotic equivalent , if as all widths go to infinity, their neural tangent kernels $\widehat{\Theta}$ converge to the same deterministic limit $\mathring{\Theta}$ at initialization and have the same optimization trajectory during training.*

The following theorem is our main result: if we train complex networks with real-valued back propagation, under very common conditions, the complex MLPs are asymptotic equivalent with real MLPs, thus they have the same training dynamics.

Theorem 7 *Consider a complex MLP in Eq.* (3) *and an ordinary real MLP with L hidden layers trained by real-valued backpropagation. Suppose $\sigma_A = \sigma_B$ at initialization and the activation functions satisfy Assumption 2. As the widths go to infinity, they are asymptotic equivalent if the activation functions satisfy one of the following conditions*

***Condition** 1 All split activation functions ϕ satisfying $\phi(\alpha, \beta) = \phi_R(\alpha) + \phi_R(\beta)i$, where ϕ_R is a real-valued activation function;*

***Condition** 2 A subset of holomorphic activation functions ϕ satisfying $\phi_2(\alpha, \beta) = \phi_1(\beta, -\alpha)$ and $\frac{\partial \phi_1(\alpha,\beta)}{\partial \beta} = \frac{\partial \phi_2(\alpha,\beta)}{\partial \alpha} = 0$,*

where the general complex activation function is denoted as $\phi(\alpha, \beta) = \phi_1(\alpha, \beta) + \phi_2(\alpha, \beta)i$ with input pre-activations $\alpha + \beta i$.

In the Appendix D the result is proved and we also give the sufficient and necessary conditions. For simplicity, here we only show the most informative conditions. The key idea of the proof is transforming asymptotic equivalence into four complex conditions and find the common solutions based on Rules.F.13 in Appendix F.

Discussion about the Condition 1. Note that most commonly used complex activation functions satisfy the Condition 1:

$$\phi(z) = \phi_R(\Re(z)) + \phi_R(\Im(z))i$$

like complex sigmoid function [Benvenuto and Piazza, 1992, Nitta, 1997], complex hyperbolic tangent function [Hirose and Yoshida, 2012], etc. It is also worth mentioning that the recently proposed ReLU-based complex activation function $\mathbb{C}$ReLU also satisfies the Condition 1, which has achieved the best performance in feed-forward complex networks in image processing tasks [Trabelsi et al., 2018, Tan et al., 2020] among all ReLU-based complex activation functions.

Remark 8 *Because of Liouville's theorem, the only complex-valued functions that are bounded and analytic everywhere are constants. Thus in practice, one must choose between boundedness and analyticity for a complex activation function. Before the popularity of ReLU, almost all activation functions in the real case were bounded. Consequently, previous works about complex networks always preferred non-analytic functions to preserve boundedness. Most commonly they applied split activation functions separately to the real and imaginary parts, as investigated in Bassey et al. [2021] and Scardapane et al. [2018]. So the condition contains most complex networks in practice.*

As a result, the theorem demonstrates that for complex networks with all these common complex activations, if they are trained by real-valued BP, then these complex networks reduce to real networks as widths grow, despite the joint interaction weight sharing caused by complex matrix multiplication structure. Consequently, real-valued backpropagation totally eliminates the characteristics of complex networks at infinite width. This may guide the selection of training algorithm in practice if people want to take full advantage of complex networks, and encourage people to explore learning algorithms specially designed for complex networks.

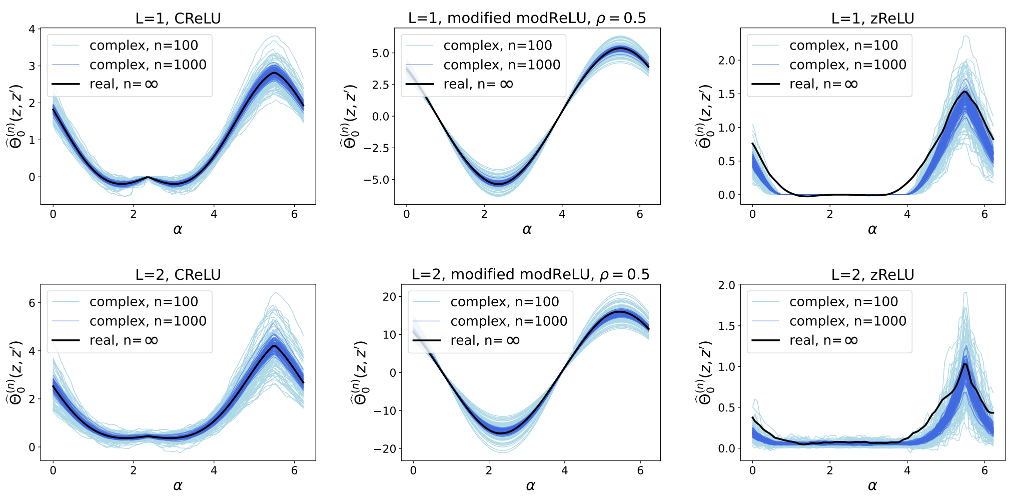

Figure 1: **Convergence of complex NTKs to corresponding real NTKs at initialization.** One input $z = 1 - i$ is fixed and the other input $z' = \cos\alpha + \sin\alpha i$ varies with $\alpha \in [0, 2\pi]$. The black line is the limiting NTKs of real MLPs $\mathring{\Theta}_r(z, z')$ while the light blue and blue ones are empirical NTKs of complex MLPs $\widehat{\Theta}_0^{(n)}(x, x')$ with widths $n = 100$ and 1000. For each width, $\widehat{\Theta}_0^{(n)}(z, z')$ is calculated 100 times randomly, corresponding to 100 lines in the figure.

Discussion about the Condition 2. Condition 2 is also non-vacuous since it is a subset of Cauchy-Riemann condition. It includes all the magnitude-based ReLU-type complex activation functions. For example, we can easily obtain a modified modReLU [Arjovsky et al., 2016] and a modified phase-based ReLU satisfying Condition 2 as follows

$$\phi(z) = \begin{cases} z & \text{if } |z| \geq \rho, \\ 0 & \text{otherwise.} \end{cases} \qquad \phi(z) = \begin{cases} z & \text{if } g(\cos\theta_z) \leq \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

where $g(\cos\theta_z)$ can be any function of phase $\cos\theta_z$. Note that, these activation functions are analytic almost everywhere, and according to previous theory, they enjoy better theoretical results like separation results [Zhang et al., 2022] and local minima [Wu et al., 2021]. However, our results indicate that real-valued backpropagation eliminates these advantages of complex networks in the infinite-width limit, which further illustrates the inappropriateness of real-valued backpropagation.

5 Empirical study

In this section, we empirically verify the relationship between NTKs of the complex networks and real networks, and investigate the network widths required for the establishment of our results. We consider complex-valued MLPs with one or two hidden layers and we use $\mathbb{C}$ReLU, modified modReLU, $\mathbb{C}$Sigmoid, $\mathbb{C}$tanh and zReLU as the activation functions. Note that all these activation functions satisfy the conditions of our theorem except zReLU. Through the following experiments, we want to check whether the empirical complex NTKs $\widehat{\Theta}_t^{(n)}$ converge to the corresponding real NTKs $\mathring{\Theta}_r$ with different activation functions as the widths n grow.

For real networks, the input is the concatenated vector of the real and imaginary parts of the complex-valued input. Corresponded to the complex-valued fully-connected layer, we use the commonly used real-valued fully-connected layer without complex matrix multiplication structure. To implement the corresponding activation functions, complex-valued activation functions are transformed to real-valued ones in the following way: we divide the pre-activation vector into two half, treat the first half as real parts and the second as imaginary parts, as the input of $\phi(\alpha, \beta)$. Then we concatenate the real and imaginary parts after activation. For split activation functions like $\mathbb{C}$ReLU, it can just correspond to real ReLU activation. In NTK initialization, the standard deviations are set as 1 for complex networks and scaled to $\sqrt{2}$ for real networks. All empirical NTKs of complex networks are calculated based on the Neural Tangents library [Novak et al., 2019].

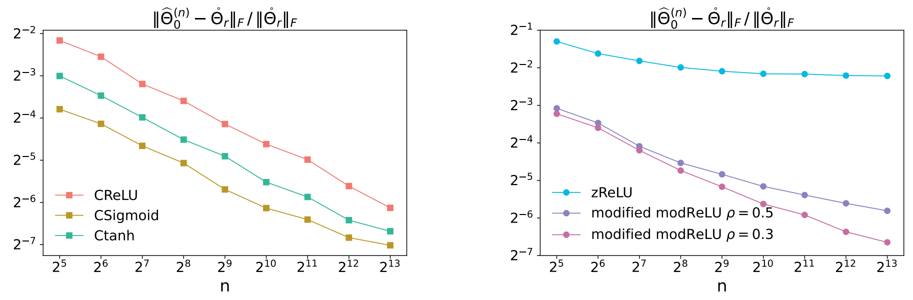

Figure 2: **Convergence of complex NTKs to corresponding real NTKs under various activation functions with a larger range of widths.** The Y-axis is the difference between empirical complex NTKs and the corresponding real NTKs in terms of the relative Frobenius norm. At each point the relative Frobenius norm is calculated 20 times and the mean value is shown. **Left:** three kinds of split activation functions satisfying our conditions. **Right:** other complex-valued activation functions where modified modReLU satisfies conditions and zReLU does not.

Verifying asymptotic equivalence at initialization. The first experiment shows the distribution of empirical NTKs of complex MLPs $\widehat{\Theta}_t^{(n)}(z, z')$ and analytic NTKs of real MLPs $\mathring{\Theta}_r(z, z')$ with different z' at initialization on a synthetic dataset. We define $z = 1 - i$ and $z' = \cos\alpha + \sin\alpha i$ for $\alpha \in [0, 2\pi]$. Then we can view $\widehat{\Theta}_t^{(n)}(z, z')$ and $\mathring{\Theta}_r(z, z')$ as functions of α. For the complex networks, we calculate empirical NTKs for hidden layer widths $n = 100, 1000$ at initialization and hidden layer number $l = 1, 2$. In each case, we calculate $\widehat{\Theta}_t^{(n)}(z, z')$ 100 times with different random NTK initialization. We compare these empirical complex NTKs with corresponding real NTKs. In the case of $\mathbb{C}$ReLU, we calculate $\mathring{\Theta}_r$ by the analytic form solution of NTK [Cho and Saul, 2009]; in the case of zReLU and modified modReLU, it's hard to get the closed form solution, so we use the average of a large mount of wide empirical NTKs to approximate $\mathring{\Theta}_r$. The results are shown in Figure 1. In the figure we observe that for $\mathbb{C}$ReLU and modified modReLU, which satisfy our conditions, their NTKs $\widehat{\Theta}_0^{(n)}$ concentrate to the NTKs of real MLPs $\mathring{\Theta}_r$ perfectly, and when $n = 1000$, the convergence is more concentrated and the complex NTKs almost equal to real NTKs; for zReLU which does not satisfy the conditions, there's a gap between complex and real NTKs. Therefore, the results verify our theorem perfectly and demonstrate our results are non-vacuous.

Verifying asymptotic equivalence with more activation functions as widths grow much larger. In the first experiment, although for zReLU there is a gap between complex and real networks, the tendency is still similar. For the second experiment, we do the similar experiment on the same synthetic dataset, to see what will happen when n becomes much larger, so that it can be more convincing to verify whether it converges. Besides, we consider more different activation functions which satisfy our conditions including $\mathbb{C}$ReLU, $\mathbb{C}$Sigmoid, $\mathbb{C}$tanh and modified modReLU with different hyperparameters ρ. We calculate relative Frobenius norm $\|\widehat{\Theta}_0^{(n)}(X, X) - \mathring{\Theta}_r(X, X)\|_F / \|\mathring{\Theta}_r(X, X)\|_F$ on set X with widths n ranging from 2^5 to 2^{13}, which measures the difference between complex NTKs $\widehat{\Theta}_0^{(n)}$ and real NTKs $\mathring{\Theta}_r$. Figure 2 shows the result. For all those split activation functions($\mathbb{C}$ReLU, $\mathbb{C}$Sigmoid and $\mathbb{C}$tanh), the tendency of convergence remains unchanged even when the width n goes to quite a large number 2^{13}. The two curves of modified modReLU act similarly with that of split activation functions; However, the curve of zReLU does not converge at all at initialization.

Verifying asymptotic equivalence during training. The third experiment investigates the convergence of difference between complex NTKs $\widehat{\Theta}_t^{(n)}$ and real NTKs $\mathring{\Theta}_r$ during training as the widths go to infinity on MNIST [LeCun et al., 1998]. We randomly choose a subset of MNIST as training set $\mathcal{D} = (X, Y)$ ($|\mathcal{D}| = 128$), and treat the first half of features as real parts, the second half as imaginary parts. Then we calculate relative Frobenius norm between empirical NTKs of complex networks at time t and real NTKs at initialization with widths n ranging from 2^5 to 2^{10} at initialization ($t = 0$) and during training ($t = 1000$). Due to the memory limitations, we cannot try larger n. The learning rate η is 0.5 for $l = 1$ and 0.2 for $l = 2$. The results are shown in Figure 3. We can see that in all these cases, the relative Frobenius norm decreases as n goes up, regardless of the training steps and

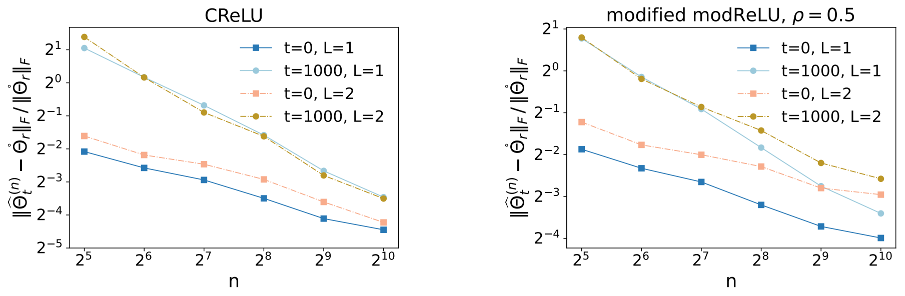

Figure 3: **Convergence of complex NTKs to corresponding real NTKs as widths grow during training.** The Y-axis is the difference between empirical complex NTKs and the corresponding real NTKs in terms of the relative Frobenius norm. At each point the relative Frobenius norm is calculated 20 times and the mean value is shown. The solid line indicates one hidden layer ($L = 1$) while the dashdot line indicates two hidden layers ($L = 2$). The square marker indicates $t = 0$ (at initialization) while the circle marker indicates $t = 1000$ (during training).

hidden layer numbers. Therefore, our theoretical results hold during training. More experiments verifying asymptotic equivalence can be found in Appendix E.

Empirical results. Overall, in the case of complex activation functions satisfying our theorems, such as $\mathbb{C}$ReLU and modified modReLU, complex NTKs $\widehat{\Theta}_t^{(n)}$ converges to real NTKs $\mathring{\Theta}_r$ quite well even the widths are about 1000; in the case of complex activation function that does satisfy our theorems like zReLU, the convergence from the complex NTK $\widehat{\Theta}_0^{(n)}$ to the real NTK $\mathring{\Theta}_r$ does not occur. This validates our theory perfectly and demonstrates that our conditions are non-vacuous.

6 Related work

Long before the popularity of deep learning, there have been many investigations on complex-valued neural networks [Hirose, 1992, Benvenuto and Piazza, 1992, Nitta, 1997]. However, It is always challenging to train complex networks due to some non-intuitive analytical properties, of which the most notable reason is that almost all cost functions are real-valued and thus non-holomorphic. In order to perform backpropagation for complex networks, the conventional approach to overcome the limitation is to use separate derivatives with respect to the real-imaginary parts of a non-analytic function [Nitta, 2004], or split amplitude-phase parts [Hirose, 1992]. Hirose and Yoshida [2012] has shown that split backpropagation for amplitude-phase parts could achieve better generalization than real networks on signal processing tasks. Recently Wirtinger calculus [Wirtinger, 1927] has received increasing attention [Adali et al., 2011, Bassey et al., 2021], which presents an elegant formulation to derive complex-valued gradient, Jacobian, and Hessian. The real-valued backpropagation [Nitta, 1997] is most widely used, and it becomes essential for training deep complex networks due to the convenience of utilizing the real-valued deep learning library [Arjovsky et al., 2016, Trabelsi et al., 2018, Tan et al., 2020] and achieves state-of-the-art performance. Theoretically, there have been important advances for complex networks regarding the universal approximation property [Voigtlaender, 2020], local minima [Nitta, 2013, Wu et al., 2021], separation results [Zhang et al., 2022]. However, to our best knowledge, there is still no theoretical analysis of the training dynamics of complex networks and the equivalence after training between complex and real networks.

7 Conclusion

In this paper, we propose a way to compare the training dynamics between complex and real networks based on their neural tangent kernels (NTKs). Surprisingly, we find that the commonly used real-valued backpropagation reduces the training dynamics of complex-valued MLPs to that of ordinary real MLPs as the widths tend to infinity, thus eliminating the characteristics of complex-valued neural networks. Empirical study verifies that our results are practical for commonly used complex activation functions. The results encourage the design of new training algorithms for complex networks in the future. Besides, the proposed asymptotic equivalence of training dynamics between networks provides a new perspective for theoretical analysis of neural networks, which may offer a possibility to divide various neural network architectures into equivalence classes.

Acknowledgment

This research was supported by NSFC (62176116, 61921006) and Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Program B for Outstanding PhD candidate of Nanjing University. The authors would like to thank Jin-Hui Wu and Peng Zhao for helpful discussions. We are also grateful for the anonymous reviewers for their valuable comments.

References

Tülay Adali, Peter J Schreier, and Louis L Scharf. Complex-valued signal processing: The proper way to deal with impropriety. *IEEE Transactions on Signal Processing*, 59(11):5101–5125, 2011.

Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. In *International Conference on Learning Representations*, 2020.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8139–8148, 2019.

Joshua Bassey, Lijun Qian, and Xianfang Li. A survey of complex-valued neural networks. *arXiv preprint arXiv:2101.12249*, 2021.

Nevio Benvenuto and Francesco Piazza. On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 40(4):967–969, 1992.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pages 342–350, 2009.

Ivo Danihelka, Greg Wayne, Benigno Uria, Nal Kalchbrenner, and Alex Graves. Associative long short-term memory. In *International Conference on Machine Learning*, pages 1986–1994, 2016.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems*, pages 5724–5734, 2019.

Nitzan Guberman. On complex valued convolutional neural networks. *arXiv preprint arXiv:1602.09046*, 2016.

Akira Hirose. Continuous complex-valued back-propagation learning. *Electronics Letters*, 28(20): 1854–1855, 1992.

Akira Hirose and Shotaro Yoshida. Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence. *IEEE Transactions on Neural Networks and Learning Systems*, 23(4):541–551, 2012.

Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pages 8580–8589, 2018.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, pages 8572–8583, 2019.

Tohru Nitta. An extension of the back-propagation algorithm to complex numbers. *Neural Networks*, 10(8):1391–1415, 1997.

Tohru Nitta. On the critical points of the complex-valued neural network. In *Proceedings of the 9th International Conference on Neural Information Processing*, pages 1099–1103, 2002.

Tohru Nitta. Orthogonality of decision boundaries in complex-valued neural networks. *Neural Computation*, 16(1):73–97, 2004.

Tohru Nitta. Local minima in hierarchical structures of complex-valued neural networks. *Neural Networks*, 43:1–7, 2013.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A Alemi, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2019.

Simone Scardapane, Steven Van Vaerenbergh, Amir Hussain, and Aurelio Uncini. Complex-valued neural networks with nonparametric activation functions. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2):140–150, 2018.

Xiaofeng Tan, Ming Li, Peng Zhang, Yan Wu, and Wanying Song. Complex-valued 3-d convolutional neural network for polsar image classification. *IEEE Geoscience and Remote Sensing Letters*, 17 (6):1022–1026, 2020.

Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018.

Mark Tygert, Joan Bruna, Soumith Chintala, Yann LeCun, Serkan Piantino, and Arthur Szlam. A mathematical motivation for complex-valued convolutional networks. *Neural Computation*, 28(5): 815–825, 2016.

Felix Voigtlaender. The universal approximation theorem for complex-valued neural networks. *arXiv preprint arXiv:2012.03351*, 2020.

Wilhelm Wirtinger. Zur formalen theorie der funktionen von mehr komplexen veränderlichen. *Mathematische Annalen*, 97(1):357–375, 1927.

Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 4880–4888, 2016.

Jin-Hui Wu, Shao-Qun Zhang, Yuan Jiang, and Zhi-Hua Zhou. Towards theoretical understanding of flexible transmitter networks via approximation and local minima. *arXiv preprint arXiv:2111.06027*, 2021.

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019a.

Greg Yang. Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *Advances in Neural Information Processing Systems*, pages 9947–9960, 2019b.

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020.

Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. In *International Conference on Machine Learning*, pages 11762–11772, 2021.

Xin Yao, Xiaoran Shi, and Feng Zhou. Human activities classification based on complex-value convolutional neural network. *IEEE Sensors Journal*, 20(13):7169–7180, 2020.

Shao-Qun Zhang and Zhi-Hua Zhou. Flexible transmitter network. *Neural Computation*, 33(11): 2951–2970, 2021.

Shao-Qun Zhang, Wei Gao, and Zhi-Hua Zhou. Towards understanding theoretical advantages of complex-reaction networks. *Neural Networks*, 151:80–93, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] **See Section 3 for assumptions.**
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] **See Section 3 for assumptions.**
   (b) Did you include complete proofs of all theoretical results? [Yes] **See Appendix.**
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] **The numerical experiments only aim to verify the theoretical results.**
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] **The numerical experiments only aim to verify the theoretical results.**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] **MNIST.**
   (b) Did you mention the license of the assets? [N/A] **MNIST. GNU General Public License v3.0**
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Complex Backpropagation

A.1 Holomorphic Functions

Consider first a complex-valued function $f(z) = u(x, y) + v(x, y)i$ where $z = x + yi$. The classical definition of *complex differentiability* requires that the derivatives defined as the limit

$$f'(z_0) = \lim_{\Delta z \to 0} \frac{f(z_0 + \Delta z) - f(z_0)}{\Delta z}$$

are independent of the direction in which Δz approaches 0 in the complex plane. In order to be complex differentiable, $f(z)$ should satisfy

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \text{ and } \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

They are called the Cauchy-Riemann equations, which give a necessary condition for complex differentiability. If the partial derivatives of $u(x, y)$ and $v(x, y)$ are continuous, then the Cauchy-Riemann equations become a sufficient condition as well. A function that is complex differentiable on its entire domain is called *holomorphic* or *analytic*.

A.2 Real-valued backpropagation for deep complex networks

Obviously, for complex neural networks with real-valued output, the network function $f_\theta(z)$ is not analytic because $v_\theta(x, y) \equiv 0$ and thus the Cauchy-Riemann conditions do not hold. Even for complex neural networks with complex-valued output, in general the Cauchy-Riemann conditions do not hold either because the loss function is real-valued. In order to perform backpropagation in a complex neural network, the conventional approach to overcome this limitation is to use the separate derivatives with respect to the real and imaginary parts of a non-analytic function [Nitta, 1997, Arjovsky et al., 2016, Trabelsi et al., 2018, Zhang and Zhou, 2021]. Thus we only require that the output functions of each layer have continuous partial derivatives with respect to the real and imaginary parts. Such functions are called real-differentiable. For this purpose, it is needed that all the activation functions in the complex networks are real-differentiable.

If $f_\theta(\boldsymbol{z})$ is a complex network function and v is a complex weight with $v = v_1 + v_2 i$ where $v_1, v_2 \in \mathbb{R}$, then

$$\nabla_v f(\boldsymbol{z}) = \frac{\partial f}{\partial v} = \frac{\partial f}{\partial v_1} + \frac{\partial f}{\partial v_2} i = \Re(\nabla_v f(\boldsymbol{z})) + \Im(\nabla_v f(\boldsymbol{z}))i.$$

If we have another complex weight $w = w_1 + w_2 i$ where $w_1, w_2 \in \mathbb{R}$ and v could be expressed in terms of w, then we have the generalized complex chain rule as follows

$$\begin{aligned}
\nabla_w f(\boldsymbol{z}) &= \frac{\partial f}{\partial w} = \frac{\partial f}{\partial w_1} + \frac{\partial f}{\partial w_2} i = \frac{\partial f}{\partial v}\frac{\partial v}{\partial w_1} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial w_2} i \\
&= \frac{\partial f}{\partial v_1}\frac{\partial v_1}{\partial w_1} + \frac{\partial f}{\partial v_2}\frac{\partial v_2}{\partial w_1} + \left(\frac{\partial f}{\partial v_1}\frac{\partial v_1}{\partial w_2} + \frac{\partial f}{\partial v_2}\frac{\partial v_2}{\partial w_2}\right) i \\
&= \frac{\partial f}{\partial v_1}\left(\frac{\partial v_1}{\partial w_1} + \frac{\partial v_1}{\partial w_2} i\right) + \frac{\partial f}{\partial v_2}\left(\frac{\partial v_2}{\partial w_1} + \frac{\partial v_2}{\partial w_2} i\right) \\
&= \Re(\nabla_w f(\boldsymbol{z}))\left(\frac{\partial v_1}{\partial w_1} + \frac{\partial v_1}{\partial w_2} i\right) + \Im(\nabla_w f(\boldsymbol{z}))\left(\frac{\partial v_2}{\partial w_1} + \frac{\partial v_2}{\partial w_2} i\right)
\end{aligned}$$

B Proof of the Theorem 3 and the Corollary 4: Complex Tensor Program

For real-valued neural networks that can be expressed by tensor program NETSOR$^\top$ (See Appendix F for details), we have the following theorem from Yang [2020], Corollary 7.3:

Theorem B.9 *Let f_{θ_R} be a real-valued (possibly recurrent) neural network with scalar output, which can be expressed by NETSOR$^\top$ and satisfies Condition F.12. If its nonlinearities have polynomially bounded weak derivatives, then its empirical NTK $\widehat{\Theta}$ at initialization converges almost surely, over any finite set of inputs, to a deterministic kernel Θ as its widths go to infinity and each elements of its factored weights W are randomly initialized as zero-mean Gaussian variables.*

The key idea of the proof is to reduce the complex tensor operations and backward propagation to real-valued tensor program NETSOR$^\top$. First we can easily show that for complex activation functions under Assumption 2 , the corresponding real-valued functions are also polynomially-bounded.

Proposition B.10 *Consider a complex activation function $\phi : \mathbb{C}^k \to \mathbb{C}$ with the input complex vector $\boldsymbol{z} = [z_1, \ldots, z_k]$ where $z_j = x_j + y_j i$ for $\forall j \in [k]$ and $x_j, y_j \in \mathbb{R}$. When $\phi(\boldsymbol{z})$ are polynomially-bounded in the complex domain, if we treat the complex activation function $\phi(\boldsymbol{z})$ as $\phi = \phi_1 + \phi_2 i$ where ϕ_1, ϕ_2 are two real-valued functions with real-valued input $[\boldsymbol{x}, \boldsymbol{y}] \in \mathbb{R}^{2k}$, then the real-valued functions $\phi_1(\boldsymbol{x}, \boldsymbol{y})$, $\phi_2(\boldsymbol{x}, \boldsymbol{y})$ are both polynomially-bounded.*

Proof The proposition can be easily proved if we treat all complex numbers using real values in terms of their real and imaginary parts. According to that $\phi(z)$ satisfies $|\phi(\boldsymbol{z})| \leq C\|\boldsymbol{z}\|^p + c$ for some $C, p, c > 0$ and $\boldsymbol{z} \in \mathbb{C}^k$, let $\boldsymbol{e} = [\boldsymbol{x}, \boldsymbol{y}]$ be the real-valued concatenating vector, we have

$$|\phi(\boldsymbol{z})| = |\phi_1(\boldsymbol{e}) + \phi_2(\boldsymbol{e})i| = \sqrt{\phi_1(\boldsymbol{e})^2 + \phi_2(\boldsymbol{e})^2} \leq C\|\boldsymbol{z}\|^p + c = C\|\boldsymbol{e}\|^p + c$$

Thus obviously we have $|\phi_1(\boldsymbol{e})|, |\phi_2(\boldsymbol{e})| \leq \sqrt{\phi_1(\boldsymbol{e})^2 + \phi_2(\boldsymbol{e})^2} = C\|\boldsymbol{e}\|^p + c$. ■

Initial set of random vectors. Without loss of generality, considering the feed-forward and backward procedure of a complex network, we have initial set of vectors $\{\boldsymbol{\alpha}_1 = \mathbf{A}_1\boldsymbol{x} - \mathbf{B}_1\boldsymbol{y}\} \cup \{\boldsymbol{\beta}_1 = \mathbf{A}_1\boldsymbol{y} + \mathbf{B}_1\boldsymbol{x}\} \cup \{\boldsymbol{\delta}_s^L(\boldsymbol{z})\} \cup \{\boldsymbol{\delta}_r^L(\boldsymbol{z})\}$ where the gradient vectors $\boldsymbol{\delta}_s^L(\boldsymbol{z})$ and $\boldsymbol{\delta}_r^L(\boldsymbol{z})$ are defined as $\boldsymbol{\delta}_s^L(\boldsymbol{z}) = \sqrt{n}\nabla_{\boldsymbol{s}_L} f(\boldsymbol{z})$ and $\boldsymbol{\delta}_r^L(\boldsymbol{z}) = \sqrt{n}\nabla_{\boldsymbol{r}_L} f(\boldsymbol{z})$ respectively like gradient vectors $\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})$ and $\boldsymbol{\delta}_\beta^l(\boldsymbol{z})$. Since complex NTK parametrization is applied for the weight matrices, it is easy to show that $Z^{\boldsymbol{\alpha}_1}$ and $Z^{\boldsymbol{\beta}_1}$ are distributed as multivariate gaussian over any input instances as follows

$$\begin{aligned}
\left\{Z^{\boldsymbol{\alpha}_1}, Z^{\boldsymbol{\alpha}_1'}\right\} &= \left\{Z^{\mathbf{A}_1\boldsymbol{x}-\mathbf{B}_1\boldsymbol{y}}, Z^{\mathbf{A}_1\boldsymbol{x}'-\mathbf{B}_1\boldsymbol{y}'}\right\} \\
&\sim \mathcal{N}\left(0, \frac{\sigma_A^2}{d}\begin{pmatrix}\boldsymbol{x}^\top\boldsymbol{x} & \boldsymbol{x}^\top\boldsymbol{x}' \\ \boldsymbol{x}^\top\boldsymbol{x}' & \boldsymbol{x}'^\top\boldsymbol{x}'\end{pmatrix} + \frac{\sigma_B^2}{d}\begin{pmatrix}\boldsymbol{y}^\top\boldsymbol{y} & \boldsymbol{y}^\top\boldsymbol{y}' \\ \boldsymbol{y}^\top\boldsymbol{y}' & \boldsymbol{y}'^\top\boldsymbol{y}'\end{pmatrix}\right). \\
\left\{Z^{\boldsymbol{\beta}_1}, Z^{\boldsymbol{\beta}_1'}\right\} &= \left\{Z^{\mathbf{A}_1\boldsymbol{y}+\mathbf{B}_1\boldsymbol{x}}, Z^{\mathbf{A}_1\boldsymbol{y}'+\mathbf{B}_1\boldsymbol{x}'}\right\} \\
&\sim \mathcal{N}\left(0, \frac{\sigma_B^2}{d}\begin{pmatrix}\boldsymbol{x}^\top\boldsymbol{x} & \boldsymbol{x}^\top\boldsymbol{x}' \\ \boldsymbol{x}^\top\boldsymbol{x}' & \boldsymbol{x}'^\top\boldsymbol{x}'\end{pmatrix} + \frac{\sigma_A^2}{d}\begin{pmatrix}\boldsymbol{y}^\top\boldsymbol{y} & \boldsymbol{y}^\top\boldsymbol{y}' \\ \boldsymbol{y}^\top\boldsymbol{y}' & \boldsymbol{y}'^\top\boldsymbol{y}'\end{pmatrix}\right).
\end{aligned}$$

where $Z^{\boldsymbol{\alpha}_1}$ and $Z^{\boldsymbol{\beta}_1}$ are random variables corresponding to the iid coordinate of $\boldsymbol{\alpha}_1$ and $\boldsymbol{\beta}_1$. (See details in Rules F.13). In addition, each coordinate of the backward initial vectors $\boldsymbol{\delta}_s^L(\boldsymbol{z}) = \sqrt{n}\nabla_{\boldsymbol{s}_L} f(\boldsymbol{z}) = \sigma_A A_{L+1}$ and $\boldsymbol{\delta}_r^L(\boldsymbol{z}) = \sqrt{n}\nabla_{\boldsymbol{r}_L} f(\boldsymbol{z}) = -\sigma_B B_{L+1}$ are also both gaussian distributed. Therefore the assumptions of Theorem F.14 are satisfied.

Forward and backward procedure of complex tensor programs. For a complex-valued neural network $f_\theta(\boldsymbol{z})$, we can always decompose all the complex operations into two-dimensional real-valued operations, denoted as $f_{[\theta_R, \theta_I]}([\boldsymbol{x}, \boldsymbol{y}])$, where $\theta_R, \theta_I \in \mathbb{R}^p$ are the real and imaginary parts of all complex parameters respectively. For the forward procedure of complex tensor programs, given $\mathbf{W} = \mathbf{A} + \mathbf{B}i$ and $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y}i$, the complex tensor operation **ComMatMul** $\mathbf{W}\boldsymbol{z} = (\mathbf{A}\boldsymbol{x} - \mathbf{B}\boldsymbol{y}) + (\mathbf{A}\boldsymbol{y} + \mathbf{B}\boldsymbol{x})\, i$ can be decomposed into four real-valued tensor operations **MatMul** in NETSOR$^\top$ in Definition F.11. And based on the Proposition B.10, a complex activation function can be seen as two real-valued functions with input $(\boldsymbol{x}, \boldsymbol{y})$, thus **ComNonLin** also can be rewritten with real-valued tensor operations **NonLin** in NETSOR$^\top$.

For the backward procedure, without loss of generality, we have the following for the backpropagation of **ComMatMul** and **ComNonLin**

Nonlin:

$$\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}) = \boldsymbol{\delta}_s^l(\boldsymbol{z}) \odot \partial\phi_1(\boldsymbol{\alpha}_l + \boldsymbol{\beta}_l i)/\partial\alpha + \boldsymbol{\delta}_r^l(\boldsymbol{z}) \odot \partial\phi_2(\boldsymbol{\alpha}_l + \boldsymbol{\beta}_l i)/\partial\alpha;$$
$$\boldsymbol{\delta}_\beta^l(\boldsymbol{z}) = \boldsymbol{\delta}_s^l(\boldsymbol{z}) \odot \partial\phi_1(\boldsymbol{\alpha}_l + \boldsymbol{\beta}_l i)/\partial\beta + \boldsymbol{\delta}_r^l(\boldsymbol{z}) \odot \partial\phi_2(\boldsymbol{\alpha}_l + \boldsymbol{\beta}_l i)/\partial\beta;$$

MatMul:

$$\boldsymbol{\delta}_s^{l-1}(\boldsymbol{z}) = \mathbf{A}_l^\top\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}) + \mathbf{B}_l^\top\boldsymbol{\delta}_\beta^l(\boldsymbol{z});$$
$$\boldsymbol{\delta}_r^{l-1}(\boldsymbol{z}) = -\mathbf{B}_l^\top\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}) + \mathbf{A}_l^\top\boldsymbol{\delta}_\beta^l(\boldsymbol{z}).$$

Therefore, the backward procedure of complex neural networks can also be expressed by NETSOR$^\top$.

Denote the real function corresponding to the complex network as $f_{\theta_r}(\boldsymbol{\tau}) : \mathbb{R}^{2d} \to \mathbb{R}^{d_{out}}$ where the input $\boldsymbol{\tau} = [\Re\{\boldsymbol{z}\}, \Im\{\boldsymbol{z}\}]$ is a real-valued concatenating vector and all weights are decomposed to real-valued matrices. According to the Theorem B.9, for this real-valued network whose forward and backward procedure could be represented by NETSOR$^\top$, its empirical NTK $\widehat{\Theta}_r : \mathbb{R}^{2d} \times \mathbb{R}^{2d} \to \mathbb{R}^{d_{out}}$ defined as

$$\widehat{\Theta}_r(\boldsymbol{\tau}, \boldsymbol{\tau}') = \langle \nabla_{\theta_r} f_{\theta_r}(\boldsymbol{\tau}), \nabla_{\theta_r} f_{\theta_r}(\boldsymbol{\tau}') \rangle$$

converges to a deterministic limiting kernel as widths go to infinity. Therefore, the equivalent complex NTK with corresponding complex input $\widehat{\Theta} : \mathbb{C}^d \times \mathbb{C}^d \to \mathbb{R}^{d_{out}}$ also converges as widths go to infinity. And the limiting NTK could be calculated by Rules F.13. This concludes the proof of Theorem 3.

Based on the Theorem 3 and the main theorem of Yang and Littwin [2021], the proof of Corollary 4 is straight. Due to that complex networks can be decomposed into real operations and represented by NETSOR$^\top$, based on the main theorem in **NTKTRAIN** under the assumptions in Setup D.1 in Yang and Littwin [2021], it can be obtained that the corresponding real function $f_{\theta_r}(\boldsymbol{\tau}) : \mathbb{R}^{2d} \to \mathbb{R}^{d_{out}}$ satisfy Eq. (2)

$$\mathring{f}_{\theta_r,t+1}(\boldsymbol{\tau}) - \mathring{f}_{\theta_r,t}(\boldsymbol{\tau}) = -\mathring{\Theta}_r(\boldsymbol{\tau}, \boldsymbol{\tau}_t)\, \mathcal{L}'_t\left(\mathring{f}_{\theta_r,t}(\boldsymbol{\tau}_t)\right), \tag{11}$$

where $\Theta_r(\boldsymbol{\tau}, \boldsymbol{\tau}') : \mathbb{R}^{2d} \times \mathbb{R}^{2d} \to \mathbb{R}^{d_{out}}$ is the infinite-width NTK (at initialization) of the corresponding real neural network. Due to the original complex network and complex NTK have the same value as real version, the Corollary 4 is proved.

C Proof of Theorem 5: NTK of complex MLPs

First, we consider the case that output dimension $d_{out} = 1$. In this case, the output weight matrix can be written as $\mathbf{W}_{L+1} = \mathbf{A}_{L+1} + \mathbf{B}_{L+1} i = \frac{\sigma_A}{\sqrt{n}} A_{L+1} + \frac{\sigma_B}{\sqrt{n}} B_{L+1} i = \frac{\sigma_A}{\sqrt{n}} \boldsymbol{a} + \frac{\sigma_B}{\sqrt{n}} \boldsymbol{b} i$ for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^{n\times 1}$ and $f_\theta(\boldsymbol{z}) = \Re\{\mathbf{W}_{L+1}\boldsymbol{h}_L\} = \frac{\sigma_A}{\sqrt{n}} \boldsymbol{a}^\top \boldsymbol{s}_L - \frac{\sigma_B}{\sqrt{n}} \boldsymbol{b}^\top \boldsymbol{r}_L$. Without loss of generality, here we set all the variance of real part and imaginary part as σ_A and σ_B respectively. We will show that the results of single-dimensional output can be easily extended to multi-dimensional output case.

Decomposing NTK: The Canonical Decomposition. Due to that the complex networks are optimized by the generalized BP algorithm, which decomposes the complex weight matrices into real-valued matrices, like Yang [2020], we can first decompose the NTK into contributions from each real-valued parameter's gradient as

$$\begin{aligned}\widehat{\Theta}(\boldsymbol{z}, \boldsymbol{z}') &= \langle \nabla_\theta f_\theta(\boldsymbol{z}), \nabla_\theta f_\theta(\boldsymbol{z}') \rangle \\ &= \sum_{l=1}^{L} \langle \nabla_{A_l} f(\boldsymbol{z}), \nabla_{A_l} f(\boldsymbol{z}') \rangle + \sum_{l=1}^{L} \langle \nabla_{B_l} f(\boldsymbol{z}), \nabla_{B_l} f(\boldsymbol{z}') \rangle \\ &\quad + \left\langle \nabla_{A_{L+1}} f(\boldsymbol{z}), \nabla_{A_{L+1}} f(\boldsymbol{z}') \right\rangle + \left\langle \nabla_{B_{L+1}} f(\boldsymbol{z}), \nabla_{B_{L+1}} f(\boldsymbol{z}') \right\rangle \end{aligned} \tag{12}$$

Considering that for $l \in [1, L]$, we have

$$\boldsymbol{s}_l + \boldsymbol{r}_l i = \phi\left((\mathbf{A}_l + \mathbf{B}_l i)(\boldsymbol{s}_{l-1} + \boldsymbol{r}_{l-1} i)\right) = \phi\left((\mathbf{A}_l \boldsymbol{s}_{l-1} - \mathbf{B}_l \boldsymbol{r}_{l-1}) + (\mathbf{A}_l \boldsymbol{r}_{l-1} + \mathbf{B}_l \boldsymbol{s}_{l-1} i)\right) \tag{13}$$

Based on gradient vectors $\boldsymbol{\delta}^l_\alpha(\boldsymbol{z})$ and $\boldsymbol{\delta}^l_\beta(\boldsymbol{z})$, when $l \in [2, L]$, the first term of Eq. (12) can be written as

$$\begin{aligned}\langle \nabla_{A_l} f(\boldsymbol{z}), \nabla_{A_l} f(\boldsymbol{z}') \rangle &= \frac{\sigma_A^2}{n} \langle \nabla_{\mathbf{A}_l} f(\boldsymbol{z}), \nabla_{\mathbf{A}_l} f(\boldsymbol{z}') \rangle \\ &= \frac{\sigma_A^2}{n^2} \left\langle \boldsymbol{\delta}^l_\alpha(\boldsymbol{z}) \boldsymbol{s}_{l-1}^\top + \boldsymbol{\delta}^l_\beta(\boldsymbol{z}) \boldsymbol{r}_{l-1}^\top, \boldsymbol{\delta}^l_\alpha(\boldsymbol{z}') {\boldsymbol{s}'_{l-1}}^\top + \boldsymbol{\delta}^l_\beta(\boldsymbol{z}') {\boldsymbol{r}'_{l-1}}^\top \right\rangle \\ &= \sigma_A^2 \frac{\boldsymbol{\delta}^l_\alpha(\boldsymbol{z})^\top \boldsymbol{\delta}^l_\alpha(\boldsymbol{z}')}{n} \frac{\boldsymbol{s}_{l-1}^\top \boldsymbol{s}'_{l-1}}{n} + \sigma_A^2 \frac{\boldsymbol{\delta}^l_\beta(\boldsymbol{z})^\top \boldsymbol{\delta}^l_\beta(\boldsymbol{z}')}{n} \frac{\boldsymbol{r}_{l-1}^\top \boldsymbol{r}'_{l-1}}{n} \\ &\quad + \sigma_A^2 \frac{\boldsymbol{\delta}^l_\alpha(\boldsymbol{z})^\top \boldsymbol{\delta}^l_\beta(\boldsymbol{z}')}{n} \frac{\boldsymbol{s}_{l-1}^\top \boldsymbol{r}'_{l-1}}{n} + \sigma_A^2 \frac{\boldsymbol{\delta}^l_\beta(\boldsymbol{z})^\top \boldsymbol{\delta}^l_\alpha(\boldsymbol{z}')}{n} \frac{\boldsymbol{r}_{l-1}^\top \boldsymbol{s}'_{l-1}}{n} \end{aligned} \tag{14}$$

Note that for simplicity, here we abbreviate $\boldsymbol{s}_l(\boldsymbol{z}')$ and $\boldsymbol{r}_l(\boldsymbol{z}')$ as $\boldsymbol{s}_l'$ and $\boldsymbol{r}_l'$ respectively. Then for the second term, when $l \in [2, L]$, we have decomposition as follows

$$\begin{aligned}
\langle \nabla_{B_l} f(\boldsymbol{z}), \nabla_{B_l} f(\boldsymbol{z}') \rangle &= \frac{\sigma_B^2}{n} \langle \nabla_{\mathbf{B}_l} f(\boldsymbol{z}), \nabla_{\mathbf{B}_l} f(\boldsymbol{z}') \rangle \\
&= \frac{\sigma_B^2}{n^2} \left\langle -\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})\boldsymbol{r}_{l-1}^\top + \boldsymbol{\delta}_\beta^l(\boldsymbol{z})\boldsymbol{s}_{l-1}^\top, -\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')\boldsymbol{r}_{l-1}'^\top + \boldsymbol{\delta}_\beta^l(\boldsymbol{z}')\boldsymbol{s}_{l-1}'^\top \right\rangle \\
&= \sigma_B^2 \frac{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})^\top \boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')}{n} \frac{\boldsymbol{r}_{l-1}^\top \boldsymbol{r}_{l-1}'}{n} + \sigma_B^2 \frac{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})^\top \boldsymbol{\delta}_\beta^l(\boldsymbol{z}')}{n} \frac{\boldsymbol{s}_{l-1}^\top \boldsymbol{s}_{l-1}'}{n} \\
&\quad - \sigma_B^2 \frac{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})^\top \boldsymbol{\delta}_\beta^l(\boldsymbol{z}')}{n} \frac{\boldsymbol{r}_{l-1}^\top \boldsymbol{s}_{l-1}'}{n} - \sigma_B^2 \frac{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})^\top \boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')}{n} \frac{\boldsymbol{s}_{l-1}^\top \boldsymbol{r}_{l-1}'}{n} \qquad (15)
\end{aligned}$$

When $l = 1$, i.e., for the input layer, since the weight matrix is $\mathbb{C}^{n\times d}$, thus we have the following decomposition for the first and second term in Eq. (12)

$$\begin{aligned}
\langle \nabla_{A_1} f(\boldsymbol{z}), \nabla_{A_1} f(\boldsymbol{z}') \rangle &= \sigma_A^2 \frac{\boldsymbol{\delta}_\alpha^1(\boldsymbol{z})^\top \boldsymbol{\delta}_\alpha^1(\boldsymbol{z}')}{n} \frac{\boldsymbol{x}^\top \boldsymbol{x}'}{d} + \sigma_A^2 \frac{\boldsymbol{\delta}_\beta^1(\boldsymbol{z})^\top \boldsymbol{\delta}_\beta^1(\boldsymbol{z}')}{n} \frac{\boldsymbol{y}^\top \boldsymbol{y}'}{d} \\
&\quad + \sigma_A^2 \frac{\boldsymbol{\delta}_\alpha^1(\boldsymbol{z})^\top \boldsymbol{\delta}_\beta^1(\boldsymbol{z}')}{n} \frac{\boldsymbol{x}^\top \boldsymbol{y}'}{d} + \sigma_A^2 \frac{\boldsymbol{\delta}_\beta^1(\boldsymbol{z})^\top \boldsymbol{\delta}_\alpha^1(\boldsymbol{z}')}{n} \frac{\boldsymbol{y}^\top \boldsymbol{x}'}{d} \qquad (16) \\
\langle \nabla_{B_1} f(\boldsymbol{z}), \nabla_{B_1} f(\boldsymbol{z}') \rangle &= \sigma_B^2 \frac{\boldsymbol{\delta}_\alpha^1(\boldsymbol{z})^\top \boldsymbol{\delta}_\alpha^1(\boldsymbol{z}')}{n} \frac{\boldsymbol{y}^\top \boldsymbol{y}'}{d} + \sigma_B^2 \frac{\boldsymbol{\delta}_\beta^1(\boldsymbol{z})^\top \boldsymbol{\delta}_\beta^1(\boldsymbol{z}')}{n} \frac{\boldsymbol{x}^\top \boldsymbol{x}'}{d} \\
&\quad - \sigma_B^2 \frac{\boldsymbol{\delta}_\alpha^1(\boldsymbol{z})^\top \boldsymbol{\delta}_\beta^1(\boldsymbol{z}')}{n} \frac{\boldsymbol{y}^\top \boldsymbol{x}'}{d} - \sigma_B^2 \frac{\boldsymbol{\delta}_\beta^1(\boldsymbol{z})^\top \boldsymbol{\delta}_\alpha^1(\boldsymbol{z}')}{n} \frac{\boldsymbol{x}^\top \boldsymbol{y}'}{d} \qquad (17)
\end{aligned}$$

Finally, for the term of output layer, we can obtain

$$\left\langle \nabla_{A_{L+1}} f(\boldsymbol{z}), \nabla_{A_{L+1}} f(\boldsymbol{z}') \right\rangle = \frac{\sigma_A^2}{n} \left\langle \nabla_{\mathbf{A}_{L+1}} f(\boldsymbol{z}), \nabla_{\mathbf{A}_{L+1}} f(\boldsymbol{z}') \right\rangle = \sigma_A^2 \frac{\boldsymbol{s}_L^\top \boldsymbol{s}_L'}{n} \qquad (18)$$

$$\left\langle \nabla_{B_{L+1}} f(\boldsymbol{z}), \nabla_{B_{L+1}} f(\boldsymbol{z}') \right\rangle = \frac{\sigma_B^2}{n} \left\langle \nabla_{\mathbf{B}_{L+1}} f(\boldsymbol{z}), \nabla_{\mathbf{B}_{L+1}} f(\boldsymbol{z}') \right\rangle = \sigma_B^2 \frac{\boldsymbol{r}_L^\top \boldsymbol{r}_L'}{n} \qquad (19)$$

Initial set of random vectors. Considering the feed-forward and backward procedure of complex full-connected networks, we have initial set of vectors $\{\boldsymbol{\alpha}_1 = \mathbf{A}_1\boldsymbol{x} - \mathbf{B}_1\boldsymbol{y}\} \cup \{\boldsymbol{\beta}_1 = \mathbf{A}_1\boldsymbol{y} + \mathbf{B}_1\boldsymbol{x}\} \cup \{\boldsymbol{\delta}_s^L(\boldsymbol{z})\} \cup \{\boldsymbol{\delta}_r^L(\boldsymbol{z})\}$ where the gradient vectors $\boldsymbol{\delta}_s^L(\boldsymbol{z})$ and $\boldsymbol{\delta}_r^L(\boldsymbol{z})$ are defined as $\boldsymbol{\delta}_s^L(\boldsymbol{z}) = \sqrt{n}\nabla_{\boldsymbol{s}_L} f(\boldsymbol{z})$ and $\boldsymbol{\delta}_r^L(\boldsymbol{z}) = \sqrt{n}\nabla_{\boldsymbol{r}_L} f(\boldsymbol{z})$ respectively like gradient vectors $\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})$ and $\boldsymbol{\delta}_\beta^l(\boldsymbol{z})$. Since complex NTK initialization is applied for the weight matrices, it is easy to show that $Z^{\boldsymbol{\alpha}_1}$ and $Z^{\boldsymbol{\beta}_1}$ are distributed as multivariate gaussian over any input instances as follows

$$\begin{aligned}
\left\{ Z^{\boldsymbol{\alpha}_1}, Z^{\boldsymbol{\alpha}_1'} \right\} &= \left\{ Z^{\mathbf{A}_1\boldsymbol{x} - \mathbf{B}_1\boldsymbol{y}}, Z^{\mathbf{A}_1\boldsymbol{x}' - \mathbf{B}_1\boldsymbol{y}'} \right\} \\
&\sim \mathcal{N}\left(0, \frac{\sigma_A^2}{d}\begin{pmatrix} \boldsymbol{x}^\top\boldsymbol{x} & \boldsymbol{x}^\top\boldsymbol{x}' \\ \boldsymbol{x}^\top\boldsymbol{x}' & \boldsymbol{x}'^\top\boldsymbol{x}' \end{pmatrix} + \frac{\sigma_B^2}{d}\begin{pmatrix} \boldsymbol{y}^\top\boldsymbol{y} & \boldsymbol{y}^\top\boldsymbol{y}' \\ \boldsymbol{y}^\top\boldsymbol{y}' & \boldsymbol{y}'^\top\boldsymbol{y}' \end{pmatrix}\right). \\
\left\{ Z^{\boldsymbol{\beta}_1}, Z^{\boldsymbol{\beta}_1'} \right\} &= \left\{ Z^{\mathbf{A}_1\boldsymbol{y} + \mathbf{B}_1\boldsymbol{x}}, Z^{\mathbf{A}_1\boldsymbol{y}' + \mathbf{B}_1\boldsymbol{x}'} \right\} \\
&\sim \mathcal{N}\left(0, \frac{\sigma_B^2}{d}\begin{pmatrix} \boldsymbol{x}^\top\boldsymbol{x} & \boldsymbol{x}^\top\boldsymbol{x}' \\ \boldsymbol{x}^\top\boldsymbol{x}' & \boldsymbol{x}'^\top\boldsymbol{x}' \end{pmatrix} + \frac{\sigma_A^2}{d}\begin{pmatrix} \boldsymbol{y}^\top\boldsymbol{y} & \boldsymbol{y}^\top\boldsymbol{y}' \\ \boldsymbol{y}^\top\boldsymbol{y}' & \boldsymbol{y}'^\top\boldsymbol{y}' \end{pmatrix}\right).
\end{aligned}$$

where $Z^{\boldsymbol{\alpha}_1}$ and $Z^{\boldsymbol{\beta}_1}$ are random variables corresponding to the iid coordinate of $\boldsymbol{\alpha}_1$ and $\boldsymbol{\beta}_1$. (See details in Rules F.13). In addition, each coordinate of the backward initial vectors $\boldsymbol{\delta}_s^L(\boldsymbol{z}) = \sqrt{n}\nabla_{\boldsymbol{s}_L} f(\boldsymbol{z}) = \sigma_A A_{L+1}$ and $\boldsymbol{\delta}_r^L(\boldsymbol{z}) = \sqrt{n}\nabla_{\boldsymbol{r}_L} f(\boldsymbol{z}) = -\sigma_B B_{L+1}$ are also both gaussian distributed. Therefore the assumptions of Theorem F.14 are satisfied.

Formulation of corresponding NETSOR$^\top$ program. As proved in the Theorem 3, a complex network which can be represented by complex tensor program could also be expressed by the NETSOR$^\top$. The complex full-connected network can be represented as the following NETSOR$^\top$ program, where we denote the corresponding real activation functions with real-valued input $[\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l]$ as $\widetilde{\phi}_1$ and $\widetilde{\phi}_2$.

Nonlin:

$$\boldsymbol{s}_l = \phi_1(\boldsymbol{\alpha}_l + \boldsymbol{\beta}_l i) = \phi_1((\boldsymbol{\alpha}^l_A - \boldsymbol{\alpha}^l_B) + (\boldsymbol{\beta}^l_A + \boldsymbol{\beta}^l_B))i) = \widetilde{\phi}_1(\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l);$$
$$\boldsymbol{r}_l = \phi_2(\boldsymbol{\alpha}_l + \boldsymbol{\beta}_l i) = \phi_2((\boldsymbol{\alpha}^l_A - \boldsymbol{\alpha}^l_B) + (\boldsymbol{\beta}^l_A + \boldsymbol{\beta}^l_B))i) = \widetilde{\phi}_2(\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l);$$

MatMul:

$$\boldsymbol{\alpha}^l_A = \mathbf{A}_l \boldsymbol{s}_{l-1}; \quad \boldsymbol{\alpha}^l_B = \mathbf{B}_l \boldsymbol{r}_{l-1}; \quad \boldsymbol{\beta}^l_A = \mathbf{A}_l \boldsymbol{r}_{l-1}; \quad \boldsymbol{\beta}^l_B = \mathbf{B}_l \boldsymbol{s}_{l-1};$$

Nonlin:

$$\boldsymbol{\delta}^l_\alpha(\boldsymbol{z}) = \boldsymbol{\delta}^l_s(\boldsymbol{z}) \odot \partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l + \boldsymbol{\delta}^l_r(\boldsymbol{z}) \odot \partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l;$$
$$\boldsymbol{\delta}^l_\beta(\boldsymbol{z}) = \boldsymbol{\delta}^l_s(\boldsymbol{z}) \odot \partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)/\partial\boldsymbol{\beta}_l + \boldsymbol{\delta}^l_r(\boldsymbol{z}) \odot \partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)/\partial\boldsymbol{\beta}_l;$$

MatMul:

$$\boldsymbol{\delta}^{l-1}_s(\boldsymbol{z}) = \mathbf{A}_l^\top \boldsymbol{\delta}^l_\alpha(\boldsymbol{z}) + \mathbf{B}_l^\top \boldsymbol{\delta}^l_\beta(\boldsymbol{z});$$
$$\boldsymbol{\delta}^{l-1}_r(\boldsymbol{z}) = -\mathbf{B}_l^\top \boldsymbol{\delta}^l_\alpha(\boldsymbol{z}) + \mathbf{A}_l^\top \boldsymbol{\delta}^l_\beta(\boldsymbol{z}).$$

Convergence of intermediate kernels. Based on Proposition B.10, it is known that the activation functions under Assumption 2 satisfies that the corresponding real activation functions ϕ_1 and ϕ_2 are polynomially bounded. The corresponding NETSOR$^\top$ program satisfies the Simple GIA Check (See details in Condition F.12), thus by recursively applying the Master Theorem F.14 we can obtain that as the width $n \to \infty$, all the intermediate kernels $\Sigma^l_\alpha, \Sigma^l_\beta, \Pi^l_\alpha, \Pi^l_\beta, \Sigma^l_{\alpha,\beta}, \Sigma^l_{\beta,\alpha}, \Pi^l_{\alpha,\beta}, \Pi^l_{\beta,\alpha}$ converge to the limits defined in Rules F.13. Specifically, for $\forall l \in [L]$, we have the following limits for the intermediate kernels:

$$\lim_{n\to\infty} \Sigma^l_\alpha(\boldsymbol{z}, \boldsymbol{z}') \xrightarrow{\text{a.s.}} \mathbb{E} Z^{\boldsymbol{\alpha}_l} Z^{\boldsymbol{\alpha}'_l} \qquad \lim_{n\to\infty} \Sigma^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}') \xrightarrow{\text{a.s.}} \mathbb{E} Z^{\boldsymbol{\alpha}_l} Z^{\boldsymbol{\beta}'_l}$$
$$\lim_{n\to\infty} \Sigma^l_\beta(\boldsymbol{z}, \boldsymbol{z}') \xrightarrow{\text{a.s.}} \mathbb{E} Z^{\boldsymbol{\beta}_l} Z^{\boldsymbol{\beta}'_l} \qquad \lim_{n\to\infty} \Sigma^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}') \xrightarrow{\text{a.s.}} \mathbb{E} Z^{\boldsymbol{\beta}_l} Z^{\boldsymbol{\alpha}'_l}$$
$$\lim_{n\to\infty} \Pi^l_\alpha(\boldsymbol{z}, \boldsymbol{z}') \xrightarrow{\text{a.s.}} \mathbb{E} Z^{\boldsymbol{\delta}^l_\alpha(\boldsymbol{z})} Z^{\boldsymbol{\delta}^l_\alpha(\boldsymbol{z}')} \qquad \lim_{n\to\infty} \Pi^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}') \xrightarrow{\text{a.s.}} \mathbb{E} Z^{\boldsymbol{\delta}^l_\alpha(\boldsymbol{z})} Z^{\boldsymbol{\delta}^l_\beta(\boldsymbol{z}')}$$
$$\lim_{n\to\infty} \Pi^l_\beta(\boldsymbol{z}, \boldsymbol{z}') \xrightarrow{\text{a.s.}} \mathbb{E} Z^{\boldsymbol{\delta}^l_\beta(\boldsymbol{z})} Z^{\boldsymbol{\delta}^l_\beta(\boldsymbol{z}')} \qquad \lim_{n\to\infty} \Pi^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}') \xrightarrow{\text{a.s.}} \mathbb{E} Z^{\boldsymbol{\delta}^l_\beta(\boldsymbol{z})} Z^{\boldsymbol{\delta}^l_\alpha(\boldsymbol{z}')}$$

where $Z^{\boldsymbol{\alpha}_l}$, $Z^{\boldsymbol{\beta}_l}$, $Z^{\boldsymbol{\delta}^l_\alpha(\boldsymbol{z})}$ and $Z^{\boldsymbol{\delta}^l_\beta(\boldsymbol{z})}$ are random variables corresponding to the iid coordinate defined in Rules F.13.

Intermediate kernels of forward procedure. The vectors $\boldsymbol{s}_l$, $\boldsymbol{r}_l$, $\boldsymbol{\alpha}_l$ and $\boldsymbol{\beta}_l$ have roughly iid coordinates distributed as $Z^{\boldsymbol{s}_l}$, $Z^{\boldsymbol{r}_l}$, $Z^{\boldsymbol{\alpha}_l}$ and $Z^{\boldsymbol{\beta}_l}$ respectively. As all widths go to infinity, based on the Rules F.13, we can make the following calculations for the forward iteration of kernels.

$$\Sigma^l_\alpha(\boldsymbol{z}, \boldsymbol{z}') = \mathbb{E} Z^{\boldsymbol{\alpha}_l} Z^{\boldsymbol{\alpha}'_l} = \sigma_A^2 \mathbb{E} Z^{\boldsymbol{s}_{l-1}} Z^{\boldsymbol{s}'_{l-1}} + \sigma_B^2 \mathbb{E} Z^{\boldsymbol{r}_{l-1}} Z^{\boldsymbol{r}'_{l-1}} \tag{20}$$
$$\Sigma^l_\beta(\boldsymbol{z}, \boldsymbol{z}') = \mathbb{E} Z^{\boldsymbol{\beta}_l} Z^{\boldsymbol{\beta}'_l} = \sigma_A^2 \mathbb{E} Z^{\boldsymbol{r}_{l-1}} Z^{\boldsymbol{r}'_{l-1}} + \sigma_B^2 \mathbb{E} Z^{\boldsymbol{s}_{l-1}} Z^{\boldsymbol{s}'_{l-1}} \tag{21}$$
$$\Sigma^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}') = \mathbb{E} Z^{\boldsymbol{\alpha}_l} Z^{\boldsymbol{\beta}'_l} = \sigma_A^2 \mathbb{E} Z^{\boldsymbol{s}_{l-1}} Z^{\boldsymbol{r}'_{l-1}} - \sigma_B^2 \mathbb{E} Z^{\boldsymbol{r}_{l-1}} Z^{\boldsymbol{s}'_{l-1}} \tag{22}$$
$$\Sigma^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}') = \mathbb{E} Z^{\boldsymbol{\beta}_l} Z^{\boldsymbol{\alpha}'_l} = \sigma_A^2 \mathbb{E} Z^{\boldsymbol{r}_{l-1}} Z^{\boldsymbol{s}'_{l-1}} - \sigma_B^2 \mathbb{E} Z^{\boldsymbol{s}_{l-1}} Z^{\boldsymbol{r}'_{l-1}} \tag{23}$$

Then, for $Z^{\boldsymbol{s}_l}$ and $Z^{\boldsymbol{r}_l}$, we can obtain

$$\begin{aligned}\mathbb{E} Z^{\boldsymbol{s}_l} Z^{\boldsymbol{s}'_l} &= \mathbb{E} Z^{\phi_1(\boldsymbol{\alpha}_l + \boldsymbol{\beta}_l i)} Z^{\phi_1(\boldsymbol{\alpha}'_l + \boldsymbol{\beta}'_l i)} = \mathbb{E} Z^{\widetilde{\phi}_1(\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)} Z^{\widetilde{\phi}_1(\boldsymbol{\alpha}'_l, \boldsymbol{\beta}'_l)} \\ &= \mathbb{E}\widetilde{\phi}_1(Z^{\boldsymbol{\alpha}_l}, Z^{\boldsymbol{\beta}_l})\widetilde{\phi}_1(Z^{\boldsymbol{\alpha}'_l}, Z^{\boldsymbol{\beta}'_l}) = \mathbb{E}\widetilde{\phi}_1(\xi_l, \zeta_l)\widetilde{\phi}_1(\xi'_l, \zeta'_l)\end{aligned} \tag{24}$$
$$\begin{aligned}\mathbb{E} Z^{\boldsymbol{r}_l} Z^{\boldsymbol{r}'_l} &= \mathbb{E} Z^{\phi_2(\boldsymbol{\alpha}_l + \boldsymbol{\beta}_l i)} Z^{\phi_2(\boldsymbol{\alpha}'_l + \boldsymbol{\beta}'_l i)} = \mathbb{E} Z^{\widetilde{\phi}_2(\boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)} Z^{\widetilde{\phi}_2(\boldsymbol{\alpha}'_l, \boldsymbol{\beta}'_l)} \\ &= \mathbb{E}\widetilde{\phi}_2(Z^{\boldsymbol{\alpha}_l}, Z^{\boldsymbol{\beta}_l})\widetilde{\phi}_2(Z^{\boldsymbol{\alpha}'_l}, Z^{\boldsymbol{\beta}'_l}) = \mathbb{E}\widetilde{\phi}_2(\xi_l, \zeta_l)\widetilde{\phi}_2(\xi'_l, \zeta'_l)\end{aligned} \tag{25}$$

Similarly, we have $\mathbb{E} Z^{\boldsymbol{s}_l} Z^{\boldsymbol{r}'_l} = \mathbb{E}\widetilde{\phi}_1(\xi_l, \zeta_l)\widetilde{\phi}_2(\xi'_l, \zeta'_l)$ and $\mathbb{E} Z^{\boldsymbol{r}_l} Z^{\boldsymbol{s}'_l} = \mathbb{E}\widetilde{\phi}_2(\xi_l, \zeta_l)\widetilde{\phi}_1(\xi'_l, \zeta'_l)$ where the random variables $\xi_l, \zeta_l, \xi'_l, \zeta'_l$ satisfy

$$(\xi_l, \zeta_l, \xi'_l, \zeta'_l) \sim \mathcal{N}\left(0, \begin{pmatrix} \Sigma^l_\alpha(\boldsymbol{z}, \boldsymbol{z}) & \Sigma^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}) & \Sigma^l_\alpha(\boldsymbol{z}, \boldsymbol{z}') & \Sigma^l_{\alpha,\beta}(\boldsymbol{z}, \boldsymbol{z}') \\ \Sigma^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}) & \Sigma^l_\beta(\boldsymbol{z}, \boldsymbol{z}) & \Sigma^l_{\beta,\alpha}(\boldsymbol{z}, \boldsymbol{z}') & \Sigma^l_\beta(\boldsymbol{z}, \boldsymbol{z}') \\ \Sigma^l_\alpha(\boldsymbol{z}', \boldsymbol{z}) & \Sigma^l_{\alpha,\beta}(\boldsymbol{z}', \boldsymbol{z}) & \Sigma^l_\alpha(\boldsymbol{z}', \boldsymbol{z}') & \Sigma^l_{\alpha,\beta}(\boldsymbol{z}', \boldsymbol{z}') \\ \Sigma^l_{\beta,\alpha}(\boldsymbol{z}', \boldsymbol{z}) & \Sigma^l_\beta(\boldsymbol{z}', \boldsymbol{z}) & \Sigma^l_{\beta,\alpha}(\boldsymbol{z}', \boldsymbol{z}') & \Sigma^l_\beta(\boldsymbol{z}', \boldsymbol{z}') \end{pmatrix}\right) \tag{26}$$

Thus the covariance matrix of each two variables among $\xi_l, \zeta_l, \xi_l', \zeta_l'$ can be directly obtained from Eq.(26).

Intermediate kernels of backward procedure. Following the derivation of the feed-forward limiting kernels, it is known that the vectors $\boldsymbol{\delta}_s^l(\boldsymbol{z})$, $\boldsymbol{\delta}_r^l(\boldsymbol{z})$, $\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})$ and $\boldsymbol{\delta}_\beta^l(\boldsymbol{z})$ have roughly iid coordinates distributed as $Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}$, $Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}$, $Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})}$ and $Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})}$ respectively. As all widths go to infinity, based on the Rules F.13, we can make the following calculations for the backward iteration of kernels.

$$\mathbb{E}Z^{\boldsymbol{\delta}_s^{l-1}(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^{l-1}(\boldsymbol{z}')} = \sigma_A^2\mathbb{E}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')} + \sigma_B^2\mathbb{E}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z}')} \tag{27}$$
$$\mathbb{E}Z^{\boldsymbol{\delta}_r^{l-1}(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^{l-1}(\boldsymbol{z}')} = \sigma_A^2\mathbb{E}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z}')} + \sigma_B^2\mathbb{E}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')} \tag{28}$$
$$\mathbb{E}Z^{\boldsymbol{\delta}_s^{l-1}(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^{l-1}(\boldsymbol{z}')} = \sigma_A^2\mathbb{E}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z}')} - \sigma_B^2\mathbb{E}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')} \tag{29}$$
$$\mathbb{E}Z^{\boldsymbol{\delta}_r^{l-1}(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^{l-1}(\boldsymbol{z}')} = \sigma_A^2\mathbb{E}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')} - \sigma_B^2\mathbb{E}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z}')} \tag{30}$$

Then, for $Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})}$ and $Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})}$, we can obtain

$$\begin{aligned}
&\Pi_\alpha^l(\boldsymbol{z},\boldsymbol{z}') = \mathbb{E}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')}\\
&= \mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})\odot\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l+\boldsymbol{\delta}_r^l(\boldsymbol{z})\odot\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')\odot\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l',\boldsymbol{\beta}_l')/\partial\boldsymbol{\alpha}_l'+\boldsymbol{\delta}_r^l(\boldsymbol{z}')\odot\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l',\boldsymbol{\beta}_l')/\partial\boldsymbol{\alpha}_l'}\\
&= \mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')}\mathbb{E}Z^{\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l}Z^{\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l',\boldsymbol{\beta}_l')/\partial\boldsymbol{\alpha}_l'}\\
&\qquad + \mathbb{E}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z}')}\mathbb{E}Z^{\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l}Z^{\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l',\boldsymbol{\beta}_l')/\partial\boldsymbol{\alpha}_l'}\\
&\qquad + \mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z}')}\mathbb{E}Z^{\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l}Z^{\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l',\boldsymbol{\beta}_l')/\partial\boldsymbol{\alpha}_l'}\\
&\qquad + \mathbb{E}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')}\mathbb{E}Z^{\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l}Z^{\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l',\boldsymbol{\beta}_l')/\partial\boldsymbol{\alpha}_l'} &(31)\\
&= \mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_1(\xi_l',\zeta_l')}{\partial\xi_l'} + \mathbb{E}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_2(\xi_l',\zeta_l')}{\partial\xi_l'}\\
&\quad + \mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_2(\xi_l',\zeta_l')}{\partial\xi_l'} + \mathbb{E}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_1(\xi_l',\zeta_l')}{\partial\xi_l'} &(32)
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
&\Pi_\beta^l(\boldsymbol{z},\boldsymbol{z}') = \mathbb{E}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z}')}\\
&= \mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})\odot\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\beta}_l+\boldsymbol{\delta}_r^l(\boldsymbol{z})\odot\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\beta}_l}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')\odot\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l',\boldsymbol{\beta}_l')/\partial\boldsymbol{\beta}_l'+\boldsymbol{\delta}_r^l(\boldsymbol{z}')\odot\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l',\boldsymbol{\beta}_l')/\partial\boldsymbol{\beta}_l'}\\
&= \mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_1(\xi_l',\zeta_l')}{\partial\zeta_l'} + \mathbb{E}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_2(\xi_l',\zeta_l')}{\partial\zeta_l'}\\
&\quad + \mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_2(\xi_l',\zeta_l')}{\partial\zeta_l'} + \mathbb{E}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_1(\xi_l',\zeta_l')}{\partial\zeta_l'} &(33)
\end{aligned}$$

We can calculate the recursive formulation of $\Pi_{\alpha,\beta}^l(\boldsymbol{z},\boldsymbol{z}')$ and $\Pi_{\beta,\alpha}^l(\boldsymbol{z},\boldsymbol{z}')$ in the same way, and the random variables $\xi_l, \zeta_l, \xi_l', \zeta_l'$ satisfy Eq.(26).

Final NTK formulation of complex full-connected network. Based on the above intermediate limiting kernels, we can calculate the final NTK of complex full-connected networks recursively according to the NTK decomposition Eq. (12). As all widths go to infinity, we can rewrite the NTK decomposition with intermediate kernels as follows.

$$\begin{aligned}
\langle\nabla_{A_l}f(\boldsymbol{z}),\nabla_{A_l}f(\boldsymbol{z}')\rangle &= \sigma_A^2\Pi_\alpha^l(\boldsymbol{z},\boldsymbol{z}')\Sigma_s^{l-1}(\boldsymbol{z},\boldsymbol{z}') + \sigma_A^2\Pi_\beta^l(\boldsymbol{z},\boldsymbol{z}')\Sigma_r^{l-1}(\boldsymbol{z},\boldsymbol{z}')\\
&\quad + \sigma_A^2\Pi_{\alpha,\beta}^l(\boldsymbol{z},\boldsymbol{z}')\Sigma_{s,r}^{l-1}(\boldsymbol{z},\boldsymbol{z}') + \sigma_A^2\Pi_{\beta,\alpha}^l(\boldsymbol{z},\boldsymbol{z}')\Sigma_{r,s}^{l-1}(\boldsymbol{z},\boldsymbol{z}') &(34)\\
\langle\nabla_{B_l}f(\boldsymbol{z}),\nabla_{B_l}f(\boldsymbol{z}')\rangle &= \sigma_B^2\Pi_\alpha^l(\boldsymbol{z},\boldsymbol{z}')\Sigma_r^{l-1}(\boldsymbol{z},\boldsymbol{z}') + \sigma_B^2\Pi_\beta^l(\boldsymbol{z},\boldsymbol{z}')\Sigma_s^{l-1}(\boldsymbol{z},\boldsymbol{z}')\\
&\quad - \sigma_B^2\Pi_{\alpha,\beta}^l(\boldsymbol{z},\boldsymbol{z}')\Sigma_{r,s}^{l-1}(\boldsymbol{z},\boldsymbol{z}') - \sigma_B^2\Pi_{\beta,\alpha}^l(\boldsymbol{z},\boldsymbol{z}')\Sigma_{s,r}^{l-1}(\boldsymbol{z},\boldsymbol{z}') &(35)\\
\left\langle\nabla_{A_{L+1}}f(\boldsymbol{z}),\nabla_{A_{L+1}}f(\boldsymbol{z}')\right\rangle &+ \left\langle\nabla_{B_{L+1}}f(\boldsymbol{z}),\nabla_{B_{L+1}}f(\boldsymbol{z}')\right\rangle = \sigma_A^2\Sigma_s^L + \sigma_B^2\Sigma_r^L &(36)
\end{aligned}$$

Thus the final NTK of complex full-connected networks can be represented as

$$\begin{aligned}\widehat{\Theta}\left(\boldsymbol{z}, \boldsymbol{z}'\right) &= \sum_{l=1}^{L}\left\langle\nabla_{A_l} f(\boldsymbol{z}), \nabla_{A_l} f\left(\boldsymbol{z}'\right)\right\rangle + \sum_{l=1}^{L}\left\langle\nabla_{B_l} f(\boldsymbol{z}), \nabla_{B_l} f\left(\boldsymbol{z}'\right)\right\rangle \\ &= \sum_{l=1}^{L}\big(\Pi_\alpha^l(\boldsymbol{z}, \boldsymbol{z}')\Sigma_\alpha^l(\boldsymbol{z}, \boldsymbol{z}') + \Pi_\beta^l(\boldsymbol{z}, \boldsymbol{z}')\Sigma_\beta^l(\boldsymbol{z}, \boldsymbol{z}') \\ &\quad + \Pi_{\alpha,\beta}^l(\boldsymbol{z}, \boldsymbol{z}')\Sigma_{\alpha,\beta}^l(\boldsymbol{z}, \boldsymbol{z}') + \Pi_{\beta,\alpha}^l(\boldsymbol{z}, \boldsymbol{z}')\Sigma_{\beta,\alpha}^l(\boldsymbol{z}, \boldsymbol{z}')\big) + \Sigma_\alpha^{L+1}(\boldsymbol{z}, \boldsymbol{z}') \end{aligned} \tag{37}$$

Multi-dimensional output. We denote the above limiting NTK corresponding to the single output complex full-connected network as $\Theta(\boldsymbol{z}, \boldsymbol{z}')$. For multi-dimensional output case, i.e., $f_\theta(\boldsymbol{z}) \in \mathbb{R}^{d_{out}}$, the i-th output for $i \in [d_{out}]$ is obtained via

$$\left[f_\theta(\boldsymbol{z})\right]_i = \left[\Re\left\{\mathbf{W}_{L+1}\boldsymbol{h}_L\right\}\right]_i = \frac{\sigma_A}{\sqrt{n}}\boldsymbol{a}_i^\top \boldsymbol{s}_L - \frac{\sigma_B}{\sqrt{n}}\boldsymbol{b}_i^\top \boldsymbol{r}_L, \tag{38}$$

where $\boldsymbol{a}_i, \boldsymbol{b}_i \in \mathbb{R}^n$ is the i-th row of corresponding real output matrices A and B; and $\boldsymbol{a}_i, \boldsymbol{b}_i$ are independent of $\boldsymbol{a}_j, \boldsymbol{b}_j$ for $i \neq j$. As a result, for the multi-dimensional output NTK $\widehat{\mathring{\Theta}}(\boldsymbol{z}, \boldsymbol{z}') \in \mathbb{R}^{d_{out}\times d_{out}}$ we have

$$\left[\widehat{\mathring{\Theta}}\left(\boldsymbol{z}, \boldsymbol{z}'\right)\right]_{i,j} = \left\langle \nabla_\theta\left[f_\theta(\boldsymbol{z})\right]_i, \nabla_\theta\left[f_\theta\left(\boldsymbol{z}'\right)\right]_j\right\rangle. \tag{39}$$

Thus for $i = j$, i.e., the diagonal elements of the kernel matrix, the value satisfies

$$\left[\mathring{\Theta}(\boldsymbol{z}, \boldsymbol{z}')\right]_{i,i} = \lim_{n\to\infty}\left[\widehat{\mathring{\Theta}}(\boldsymbol{z}, \boldsymbol{z}')\right]_{i,i} = \Theta(\boldsymbol{z}, \boldsymbol{z}'), \text{ for } i = j; \tag{40}$$

but for $i \neq j$, since $\boldsymbol{a}_i, \boldsymbol{b}_i$ are independent of $\boldsymbol{a}_j, \boldsymbol{b}_j$ for $i \neq j$ and all the gradient vectors in $\nabla_\theta\left[f_\theta(\boldsymbol{x})\right]_i$ contains $\boldsymbol{a}_i, \boldsymbol{b}_i$ due to the backpropagation, we have

$$\left[\mathring{\Theta}(\boldsymbol{z}, \boldsymbol{z}')\right]_{i,j} = \lim_{n\to\infty}\left\langle \nabla_\theta\left[f_\theta(\boldsymbol{z})\right]_i, \nabla_\theta\left[f_\theta\left(\boldsymbol{z}'\right)\right]_j\right\rangle = 0, \text{ for } i \neq j. \tag{41}$$

Therefore, we have the following limiting NTK for the multi-dimensional output complex full-connected network

$$\mathring{\Theta}(\boldsymbol{z}, \boldsymbol{z}') = \Theta(\boldsymbol{z}, \boldsymbol{z}') \otimes \mathbf{I}_{d_{out}}. \tag{42}$$

This concludes the proof of Theorem 5.

D Proof of Theorem 7: Asymptotic equivalence for deep complex networks

To prove Theorem 7, we need to analyze the conditions that the NTK of complex MLP reduces to that of real MLP when $\sigma_A = \sigma_B$. Based on the expansion of final NTK form of complex MLPs (Eq. (34)-(36)), we could conclude that the following four subconditions need to be satisfied at the same time.

Subcondition 1 $\forall l$, the interaction terms in the NTK decomposition are eliminated.

$$\begin{aligned}&\Pi_{\alpha,\beta}^l(\boldsymbol{z}, \boldsymbol{z}')\Sigma_{s,r}^{l-1}(\boldsymbol{z}, \boldsymbol{z}') + \Pi_{\beta,\alpha}^l(\boldsymbol{z}, \boldsymbol{z}')\Sigma_{r,s}^{l-1}(\boldsymbol{z}, \boldsymbol{z}') \\ &\qquad -\Pi_{\alpha,\beta}^l(\boldsymbol{z}, \boldsymbol{z}')\Sigma_{r,s}^{l-1}(\boldsymbol{z}, \boldsymbol{z}') - \Pi_{\beta,\alpha}^l(\boldsymbol{z}, \boldsymbol{z}')\Sigma_{s,r}^{l-1}(\boldsymbol{z}, \boldsymbol{z}') = 0\end{aligned} \tag{43}$$

Subcondition 2 $\forall l$, the interaction terms in $\Pi_\alpha^l(\boldsymbol{z}, \boldsymbol{z}')$ and $\Pi_\beta^l(\boldsymbol{z}, \boldsymbol{z}')$ are eliminated.

Subcondition 3 $\forall l$, we have $\Pi_\alpha^l(\boldsymbol{z}, \boldsymbol{z}') = \Pi_\beta^l(\boldsymbol{z}, \boldsymbol{z}')$.

Subcondition 4 $\forall l$, we have $\Sigma_s^l(\boldsymbol{z}, \boldsymbol{z}') = \Sigma_r^l(\boldsymbol{z}, \boldsymbol{z}')$.

For Subcondition 1, we can obtain the equivalent formulation as follows

$$\Pi_{\alpha,\beta}^l(\boldsymbol{z}, \boldsymbol{z}') - \Pi_{\beta,\alpha}^l(\boldsymbol{z}, \boldsymbol{z}') = \mathbb{E}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z}')} - \mathbb{E}Z^{\boldsymbol{\delta}_\beta^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_\alpha^l(\boldsymbol{z}')} = 0. \tag{44}$$

because $\Sigma^l_{\alpha,\beta}(\boldsymbol{z},\boldsymbol{z}') = \Sigma^{l-1}_{s,r}(\boldsymbol{z},\boldsymbol{z}') - \Sigma^{l-1}_{r,s}(\boldsymbol{z},\boldsymbol{z}') = 0$ and $\Sigma^l_{\beta,\alpha}(\boldsymbol{z},\boldsymbol{z}') = 0$ are unachievable. Specifically, for $l = 1$, they hold only when the input $[\boldsymbol{x},\boldsymbol{y}] = c[\boldsymbol{x}',\boldsymbol{y}']$ for some $c \in \mathbb{R}$. Then we can transform the Subcondition 1 to the following:

$$\begin{aligned}&\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})\odot\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l+\boldsymbol{\delta}^l_r(\boldsymbol{z})\odot\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\alpha}_l}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')\odot\partial\widetilde{\phi}_1(\boldsymbol{\alpha}'_l,\boldsymbol{\beta}'_l)/\partial\boldsymbol{\beta}'_l+\boldsymbol{\delta}^l_r(\boldsymbol{z}')\odot\partial\widetilde{\phi}_2(\boldsymbol{\alpha}'_l,\boldsymbol{\beta}'_l)/\partial\boldsymbol{\beta}'_l}\\&-\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})\odot\partial\widetilde{\phi}_1(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\beta}_l+\boldsymbol{\delta}^l_r(\boldsymbol{z})\odot\partial\widetilde{\phi}_2(\boldsymbol{\alpha}_l,\boldsymbol{\beta}_l)/\partial\boldsymbol{\beta}_l}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')\odot\partial\widetilde{\phi}_1(\boldsymbol{\alpha}'_l,\boldsymbol{\beta}'_l)/\partial\boldsymbol{\alpha}'_l+\boldsymbol{\delta}^l_r(\boldsymbol{z}')\odot\partial\widetilde{\phi}_2(\boldsymbol{\alpha}'_l,\boldsymbol{\beta}'_l)/\partial\boldsymbol{\alpha}'_l}=0\end{aligned} \tag{45}$$

Based on the Rules. F.13, we can obtain

$$\begin{aligned}&\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\zeta'_l}+\mathbb{E}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\zeta'_l}\\&+\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\zeta'_l}+\mathbb{E}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\zeta'_l}\\&-\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\xi'_l}-\mathbb{E}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\xi'_l}\\&-\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\xi'_l}-\mathbb{E}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\xi'_l}=0\end{aligned} \tag{46}$$

For Subcondition 2, based on Eq. (32) and Eq. (33), we have the expansion as following

$$\begin{aligned}&\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\xi'_l}+\mathbb{E}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\xi'_l}\\&\quad+\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\zeta'_l}+\mathbb{E}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\zeta'_l}=0\end{aligned}$$

According to Eq. (29) and Eq. (30), considering $\sigma_A = \sigma_B$, thus $Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z}')} = -Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')}$, then we can obtain

$$\begin{aligned}&\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\xi'_l}+\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\zeta'_l}\\&\quad-\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\xi'_l}-\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\zeta'_l}=0\end{aligned} \tag{47}$$

For Subcondition 3, given the Subcondition 2 holds, we can obtain that

$$\begin{aligned}&\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\xi'_l}+\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\xi'_l}\\&\quad-\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\zeta'_l}-\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\zeta'_l}=0\end{aligned} \tag{48}$$

For Subcondition 4, we can obtain that

$$\mathbb{E}Z^{s_l}Z^{s'_l}-\mathbb{E}Z^{r_l}Z^{r'_l}=\mathbb{E}\widetilde{\phi}_1(\xi_l,\zeta_l)\widetilde{\phi}_1(\xi'_l,\zeta'_l)-\mathbb{E}\widetilde{\phi}_2(\xi_l,\zeta_l)\widetilde{\phi}_2(\xi'_l,\zeta'_l)=0 \tag{49}$$

where the random variables $\xi_l, \zeta_l, \xi'_l, \zeta'_l$ are distributed as Eq. (26).

It could be verified that there is no common solution for all these four subconditions if we assume that $\partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\xi_l, \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\xi_l, \partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\zeta_l, \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\zeta_l$ are all nonzero (this can be verified by substituting all solutions of the Subcondition 1 into the other Subconditions). However, when we allow $\partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\zeta_l, \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\xi_l$ to be zero, which is common in popular activation functions, we can find the solution that satisfies all the subconditions.

When $\partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\zeta_l = \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\xi_l = 0$ and $\partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\xi_l = \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\zeta_l$, the Subcondition 2 and Subcondition 3 naturally hold. For Subcondition 1, it is transformed to

$$\mathbb{E}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l}\frac{\partial\widetilde{\phi}_2(\xi'_l,\zeta'_l)}{\partial\zeta'_l}-\mathbb{E}Z^{\boldsymbol{\delta}^l_r(\boldsymbol{z})}Z^{\boldsymbol{\delta}^l_s(\boldsymbol{z}')}\mathbb{E}\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}\frac{\partial\widetilde{\phi}_1(\xi'_l,\zeta'_l)}{\partial\xi'_l}=0$$

which needs $\mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z}')} - \mathbb{E}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')} = 0$, and this is only satisfied when $\Pi^l_{\alpha,\beta}(\boldsymbol{z},\boldsymbol{z}') = \Pi^l_{\beta,\alpha}(\boldsymbol{z},\boldsymbol{z}')$, thus this constructs a recursion: $\mathbb{E}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z}')} - \mathbb{E}Z^{\boldsymbol{\delta}_r^l(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^l(\boldsymbol{z}')} = 0$ only when $\mathbb{E}Z^{\boldsymbol{\delta}_s^{l+1}(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^{l+1}(\boldsymbol{z}')} - \mathbb{E}Z^{\boldsymbol{\delta}_r^{l+1}(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^{l+1}(\boldsymbol{z}')} = 0$. Finally, due to that for the output layer, $Z^{\boldsymbol{\delta}_s^L(\boldsymbol{z})}$ and $Z^{\boldsymbol{\delta}_r^L(\boldsymbol{z}')}$ are independent, i.e., $\mathbb{E}Z^{\boldsymbol{\delta}_s^L(\boldsymbol{z})}Z^{\boldsymbol{\delta}_r^L(\boldsymbol{z}')} = \mathbb{E}Z^{\boldsymbol{\delta}_r^L(\boldsymbol{z})}Z^{\boldsymbol{\delta}_s^L(\boldsymbol{z}')} = 0$, thus this concludes the proof of satisfying Subcondition 1.

For Subcondition 4, when $\widetilde{\phi}_2(a,b) = \widetilde{\phi}_1(b,-a)$, which could also induce that $\partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\xi_l = \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\zeta_l$, we can obtain

$$\mathbb{E}\widetilde{\phi}_2(\xi_l,\zeta_l)\widetilde{\phi}_2(\xi_l',\zeta_l') = \mathbb{E}\widetilde{\phi}_1(\zeta_l,-\xi_l)\widetilde{\phi}_1(\zeta_l',-\xi_l') \tag{50}$$

Notice that

$$(\zeta_l,-\xi_l,\zeta_l',-\xi_l') \sim \mathcal{N}\left(0, \begin{pmatrix} \Sigma^l_\alpha(\boldsymbol{z},\boldsymbol{z}) & -\Sigma^l_{\beta,\alpha}(\boldsymbol{z},\boldsymbol{z}) & \Sigma^l_\alpha(\boldsymbol{z},\boldsymbol{z}') & -\Sigma^l_{\beta,\alpha}(\boldsymbol{z},\boldsymbol{z}') \\ -\Sigma^l_{\alpha,\beta}(\boldsymbol{z},\boldsymbol{z}) & \Sigma^l_\beta(\boldsymbol{z},\boldsymbol{z}) & -\Sigma^l_{\alpha,\beta}(\boldsymbol{z},\boldsymbol{z}') & \Sigma^l_\beta(\boldsymbol{z},\boldsymbol{z}') \\ \Sigma^l_\alpha(\boldsymbol{z}',\boldsymbol{z}) & -\Sigma^l_{\beta,\alpha}(\boldsymbol{z}',\boldsymbol{z}) & \Sigma^l_\alpha(\boldsymbol{z}',\boldsymbol{z}') & -\Sigma^l_{\beta,\alpha}(\boldsymbol{z}',\boldsymbol{z}') \\ -\Sigma^l_{\alpha,\beta}(\boldsymbol{z}',\boldsymbol{z}) & \Sigma^l_\beta(\boldsymbol{z}',\boldsymbol{z}) & -\Sigma^l_{\alpha,\beta}(\boldsymbol{z}',\boldsymbol{z}') & \Sigma^l_\beta(\boldsymbol{z}',\boldsymbol{z}') \end{pmatrix}\right) \tag{51}$$

due to that $\Sigma^l_{\alpha,\beta}(\boldsymbol{z}',\boldsymbol{z}) = -\Sigma^l_{\beta,\alpha}(\boldsymbol{z},\boldsymbol{z}')$, thus the distribution is the same as Eq.(26). Then we can obtain that

$$\mathbb{E}\widetilde{\phi}_2(\xi_l,\zeta_l)\widetilde{\phi}_2(\xi_l',\zeta_l') = \mathbb{E}\widetilde{\phi}_1(\zeta_l,-\xi_l)\widetilde{\phi}_1(\zeta_l',-\xi_l') = \mathbb{E}\widetilde{\phi}_1(\xi_l,\zeta_l)\widetilde{\phi}_1(\xi_l',\zeta_l') \tag{52}$$

In summary, when $\widetilde{\phi}_2(a,b) = \widetilde{\phi}_1(b,-a)$ and $\partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\zeta_l = \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\xi_l = 0$ the NTK of complex MLP reduces to real MLP.

As for the necessary and sufficient conditions for the asymptotic equivalence, we consider all solutions of Subcondition 4: $\widetilde{\phi}_2(a,b) = \widetilde{\phi}_1(-b,a)$ and $\widetilde{\phi}_2(a,b) = \widetilde{\phi}_1(b,-a)$, and verify them with Subcondition 1-3. We can obtain that all solutions satisfying all the conditions as follows $\widetilde{\phi}_2(a,b) = \widetilde{\phi}_1(-b,a)$.

$$\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l} = \frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l} = 0, \quad \frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l} = \pm\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l}; \tag{53}$$

$$\frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\xi_l} = \frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\zeta_l} = 0, \quad \frac{\partial\widetilde{\phi}_1(\xi_l,\zeta_l)}{\partial\zeta_l} = \pm\frac{\partial\widetilde{\phi}_2(\xi_l,\zeta_l)}{\partial\xi_l}. \tag{54}$$

This concludes the proof of the Condition 2 in Theorem 7.

For split activation functions, i.e.,

$$\phi(z) = \phi_1(z) + \phi_2(z)i = \phi_R(\Re(z)) + \phi_R(\Im(z))i$$

we naturally have $\partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\zeta_l = \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\xi_l = 0$ and $\partial\widetilde{\phi}_1(\xi_l,\zeta_l)/\partial\xi_l = \partial\widetilde{\phi}_2(\xi_l,\zeta_l)/\partial\zeta_l$. Thus Subcondition 1-3 hold as analyzed above. For Subcondition 4, we have $\mathbb{E}Z^{\boldsymbol{s}_l}Z^{\boldsymbol{s}_l'} = \mathbb{E}\widetilde{\phi}_1(\xi)\widetilde{\phi}_1(\xi')$ and $\mathbb{E}Z^{\boldsymbol{r}_l}Z^{\boldsymbol{r}_l'} = \mathbb{E}\widetilde{\phi}_2(\zeta)\widetilde{\phi}_2(\zeta')$, where

$$(\xi,\xi') \sim \mathcal{N}\left(0, \begin{pmatrix} \Sigma^l_\alpha(\boldsymbol{z},\boldsymbol{z}) & \Sigma^l_\alpha(\boldsymbol{z},\boldsymbol{z}') \\ \Sigma^l_\alpha(\boldsymbol{z},\boldsymbol{z}') & \Sigma^l_\alpha(\boldsymbol{z}',\boldsymbol{z}') \end{pmatrix}\right), \tag{55}$$

$$(\zeta,\zeta') \sim \mathcal{N}\left(0, \begin{pmatrix} \Sigma^l_\beta(\boldsymbol{z},\boldsymbol{z}) & \Sigma^l_\beta(\boldsymbol{z},\boldsymbol{z}') \\ \Sigma^l_\beta(\boldsymbol{z},\boldsymbol{z}') & \Sigma^l_\beta(\boldsymbol{z}',\boldsymbol{z}') \end{pmatrix}\right). \tag{56}$$

Note that $\Sigma^l_\alpha(\boldsymbol{z},\boldsymbol{z}') = \Sigma^l_\beta(\boldsymbol{z},\boldsymbol{z}')$ for $\forall \boldsymbol{z},\boldsymbol{z}' \in \mathbb{C}^d$ since $\sigma_A = \sigma_B$. This concludes the proof of the Condition 1 in Theorem 7.

E Additional experiments for complex NTK during training

Verifying complex NTK during training. In this experiment, we investigate the difference of complex NTK before and after training on MNIST, i.e., we compare the empirical complex NTKs

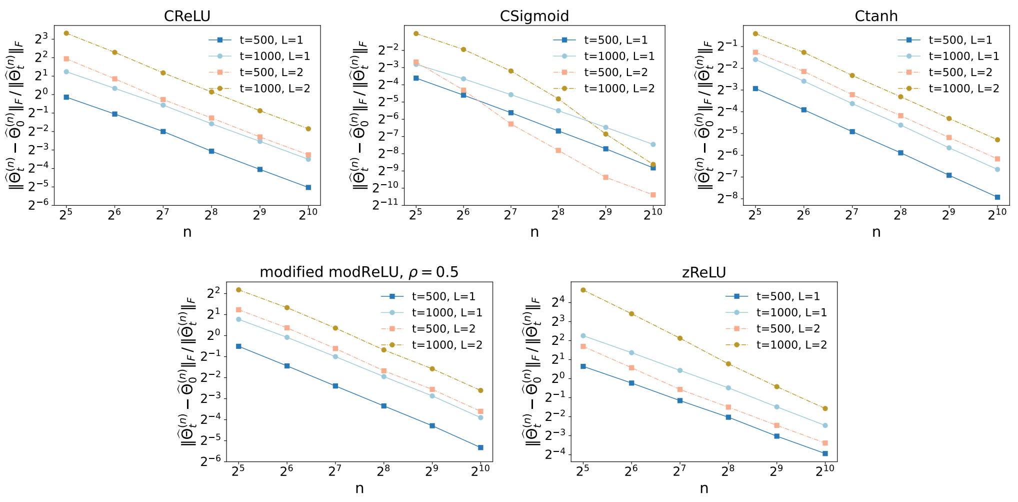

Figure 4: **Convergence of the difference of empirical complex-valued NTKs before and during training as widths grow.** Similar to Figure 3, but the Y-axis is the relative Frobenius norm of the change of empirical NTKs during training.

during training $\hat{\Theta}_t^{(n)}$ with empirical complex NTK at initialization $\hat{\Theta}_0^{(n)}$. The results are shown in Figure 4. We observe that the relative norm converges fast in all these cases, regardless of the training steps, hidden layer number and activation functions. So the empirical complex NTK during training $\hat{\Theta}_t^{(n)}$ converges to the empirical complex NTK at initialization $\hat{\Theta}_0^{(n)}$ as the widths grows, which means the complex NTK remains frozen during training. This experiment perfectly verifies our Theorem 4, because all these activation functions satisfy the the Assumption 2 of complex tensor program.

In conclusion, the first two experiments focus on the difference between complex NTK and real NTK at initialization, which verify our theoretical results about real-valued backpropagation and demonstrate that our conditions in Theorem 7 are non-vacuous. And this experiment focuses on the change of NTKs during training, which verifies our Theorem 4; and on the other hand, it indicates that in order to figure out the relationship between complex NTKs and real NTKs, we could just focus on the NTKs at initialization.

F NETSOR$^\top$ Program

For self-containedness, in this section we will highlight some important theoretical tools used in our proofs.

For MLPs and CNNs, it has been shown that the pre-activation of each layer tends to a Gaussian process as width $n \to \infty$ based on the Central Limit Theorem (CLT) [Lee et al., 2018] and the neural tangent kernel (NTK) at initialization tends to a limiting kernel as width $n \to \infty$ [Jacot et al., 2018, Arora et al., 2019]. However, the analysis can not be extended to the neural networks with weight sharing, like recurrent neural networks (RNN), because the sequential limit is not possible and conditioning on previous layers does not result in iid weights. To deal with it, Yang [2019b, 2020] presented a novel proof using Gaussian conditioning trick which allows the recurrent weights in a network. Further, it has been demonstrated that any architecture that can be expressed as NETSOR programs converges to Gaussian process as all widths go to infinity in the same rate and any architecture whose feed-forward and backward procedure can be expressed as NETSOR$^\top$ programs has a convergent NTK as all widths go to infinity in the same rate. Besides, to ensure readability, we did not use the original Tensor Programs [Yang, 2019a], which has been superceded by the NETSOR$^\top$, a more concise and readable version. We list the important theoretical tools used in our proofs including definitions, theorems and conditions from [Yang, 2019b, 2020] as follows.

Definition F.11 (Simplified NETSOR$^\top$ Program) *A NETSOR$^\top$ program is just a sequence of $\mathbb{R}^n$ vectors inductively generated via one of the following ways from an initial set $\mathcal{V}$ of random $\mathbb{R}^n$ vectors and a set $\mathcal{W}$ of random $n \times n$ matrices*

Nonlin *Given $\phi : \mathbb{R}^k \to \mathbb{R}$ and $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k \in \mathbb{R}^n$, we can generate $\phi(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \in \mathbb{R}^n$.*

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

Condition F.12 (Simple GIA Check) *The output layer is sampled independently and with zero mean from all other parameters and is not used anywhere else in the interior of the network.*

Rules F.13 (Intuitions for Computing the Limits) *When the width $n \gg 1$, every (pre-)actication vector $\boldsymbol{x} \in \mathbb{R}^n$ has roughly i.i.d. coordinates distributed as some random variable denoted $Z^{\boldsymbol{x}}$. Thus for any vector $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, as $n \to \infty$,*

$$\boldsymbol{x}^\top \boldsymbol{y}/n = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{y}_i \to \mathbb{E} Z^{\boldsymbol{x}} Z^{\boldsymbol{y}},$$

Then we can use the following rules to compute $Z^{\boldsymbol{x}}$ of a NETSOR$^\top$ program recursively.

- **Nonlin** *For any fixed k and ϕ: $\mathbb{R}_k \to \mathbb{R}$, we have*
$$Z^{\phi(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)} = \phi(Z^{\boldsymbol{x}_1}, \cdots, Z^{\boldsymbol{x}_k})$$
- **MatMul** *For any set of $\mathbb{R}^n$ vectors $\mathcal{X}$ and a matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ with $\mathbf{W}_{ij} \sim \mathcal{N}(0, \sigma_W^2/n)$, the set of random variables $\{Z^{\mathbf{W}\boldsymbol{x}} : \boldsymbol{x} \in \mathcal{X}\}$ is jointly Gaussian with zero mean and covariance*
$$Cov(Z^{\mathbf{W}\boldsymbol{x}}, Z^{\mathbf{W}\overline{\boldsymbol{x}}}) = \sigma_W^2 \mathbb{E} Z^{\boldsymbol{x}} Z^{\overline{\boldsymbol{x}}}, \ \ \forall \boldsymbol{x}, \overline{\boldsymbol{x}} \in \mathcal{X}.$$
If $\mathcal{Y}$ is any set of $\mathbb{R}^n$ vectors and $\overline{\mathbf{W}} \neq \mathbf{W}$, then $\{Z^{\mathbf{W}\boldsymbol{x}} : \boldsymbol{x} \in \mathcal{X}\}$ is independent from $\{Z^{\overline{\mathbf{W}}\boldsymbol{y}} : \boldsymbol{y} \in \mathcal{Y}\}$.

Theorem F.14 (NETSOR$^\top$ Master Theorem) *Consider a NETSOR$^\top$ program. Suppose:*

- *for each intial $\mathbf{W} \in \mathcal{W}$, $\mathbf{W}_{ij} \in \mathcal{N}(0, \sigma_W^2/n)$ for an associate variance σ_W^2;*
- *there is a multivariate Gaussian $Z^{\mathcal{V}} = \{Z^{\boldsymbol{\nu}} : \boldsymbol{\nu} \in \mathcal{V} \in \mathbb{R}^{|\mathcal{V}|}\}$ such that the initial set of vectors $\mathcal{V}$ are sampled like $\{\boldsymbol{\nu}_i : \boldsymbol{\nu} \in \mathcal{V}\} \sim Z^{\mathcal{V}}$ i.i.d. for each $i \in [n]$.*

if the program satisfies the simple GIA check and all ϕ used in **Nonlin** *are polynomially bounded, then*

$$\frac{1}{n}\sum_{i=1}^{n} \psi\left(h_i^1, \ldots, h_i^k\right) \overset{a.s.}{\to} \mathbb{E}\psi\left(Z^{h^1}, \ldots, Z^{h^k}\right), \quad as \quad n \to \infty$$

for any collection of vectors $h^1, \ldots, h^k$ in the program and any polynomially bounded $\psi : \mathbb{R}^k \to \mathbb{R}$, where Z^{h^i} are defined in **Rules F.13**.