Neural Networks with Hadamard Product: Separation on Extrapolation and Spectral Bias

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural tangent kernel (NTK) is a powerful tool to analyze training dynamics of neural networks and their generalization bounds. The study on NTK has been devoted to typical neural network architectures, but is incomplete for neural networks with Hadamard products (NNs-Hp), e.g., StyleGAN and polynomial neural networks. In this work, we derive the finite-width NTK formulation for NNs-Hp, and prove their equivalence to the kernel regression predictor with the associated NTK, which expands the application scope of NTK. Based on our results, we elucidate the difference of NNs-Hp over standard neural networks with respect to spectral bias and extrapolation. Our two key insights are that when compared to standard neural networks, a) NNs-Hp admit a slower eigenvalue decay of the respective NTK; and b) are able to fit more complicated functions in the extrapolation regime. Our empirical results validate those two separations, which provide a good justification for a deeper understanding of neural architectures.

1 Introduction

In deep learning theory, neural tangent kernel (NTK) [Jacot et al., 2018] is a powerful analysis tool that bridges the gap among the training dynamics of neural networks (NNs) trained by gradient descent and kernel regression [Jacot et al., 2018, Arora et al., 2019]. NTK provides a tractable analysis for several phenomena in deep learning, e.g., the global convergence of gradient descent [Allen-Zhu et al., 2019, Chizat et al., 2019, Du et al., 2019a, 2018], the inductive bias behind NNs [Bietti and Mairal, 2019], the spectral bias toward different frequency components [Cao et al., 2019, Choraria et al., 2022], the extrapolation behavior [Xu et al., 2021], and the generalization ability [Huang et al., 2020]. The study on NTK has been devoted to typical NNs architectures, e.g., fully-connected NNs [Jacot et al., 2018], residual NNs [Tirer et al., 2020, Huang et al., 2020], convolutional NNs [Arora et al., 2019], graph NNs [Du et al., 2019b] and recurrent NNs [Alemohammad et al., 2021].

Recently, NNs with Hadamard products (NNs-Hp), e.g., StyleGAN [Karras et al., 2019] and Π-Nets [Chrysos et al., 2020], have received increasing attention due to their expressivity and efficiency over traditional NNs [Chrysos et al., 2021a, Campbell and Broun, 2000, Su et al., 2020]. There have been several works attempting to demystify the success of NNs-Hp. Fan et al. [2021] prove that second-degree multiplicative interactions allow NNs-Hp to enlarge the set of functions that can be represented exactly with zero error. Choraria et al. [2022] reveal that NNs-Hp with second-degree multiplicative interactions can yield faster learning of high-frequency information during training in NTK regime. Yet, the theoretical analysis of NNs-Hp with high-degree multiplicative interactions is still unclear. When using NTK for analysis, the complete and rigorous proof includes showing the stability of empirical NTK during training and giving the width requirement, which is crucial to allow NTK-based analysis of typical NNs Arora et al. [2019], Tirer et al. [2020] but is still missing for NNs-Hp.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this work, we derive the NTK for NNs-Hp with high-degree multiplicative interactions and present a rigorous proof for the equivalence to the kernel regression predictor. This analysis enables us to further examine NNs-Hp in a theoretical perspective. This NTK analysis can be useful for investigating properties, such as the extrapolation. StyleGAN and Π-Nets have demonstrated stellar in-distribution performance, however implementing networks for real-world applications requires the ability to extrapolate to unseen data. As humans, we have the capacity to extrapolate beyond what we have learned [Lake et al., 2017, Webb et al., 2020]. On the contrary, NNs usually lack such ability even in solving simple arithmetic problems [Saxton et al., 2019] or learning simple functions [Haley and Soloway, 1992, Sahoo et al., 2018]. Early works demonstrate that fully-connected NNs are unable to achieve exact extrapolation when fitting low-degree polynomial functions [Haley and Soloway, 1992]. Xu et al. [2021] theoretically and empirically point out that 2-layer fully-connected NNs can only extrapolate to linear function. However, data in real-world scenarios usually lie in highly nonlinear manifolds [Van Der Maaten et al., 2009]. Our findings extend these result and highlight that NNs-Hp can extrapolate to unseen data in a non-linear way. What's more, studying the NTK of NNs-Hp also allows us to investigate its spectral bias.

Overall, our main contributions can be summarized as follows:

- We derive the NTK formulation for NNs-Hp with high-degree multiplicative interactions, and gives a concrete bound of the widths requirement for convergence to the NTK at initialization, and stability during training, which allows us to bridge the gap among NNs-Hp trained via gradient descent and kernel regression predictor.
- We provably present the extrapolation behavior of NNs-Hp and empirically show its advantage over standard NNs in learning common used functions and performing arithmetic extrapolation in real-world dataset.
- We provide a detailed analysis for the decay rate of eigenvalues of the NTK of NNs-Hp with high-degree multiplicative interactions in the task of learning spherical harmonics and scrutiny that increasing the multiplicative interactions results in a faster learning towards high-frequency information.

2 Background

In this section, we establish the notations, provide an overview of NTK, and summarize the most closely related work in NNs-Hpas well as extrapolation.

2.1 Notation

The core operators and symbols are summarized in Table 2 at Appendix A. Vectors (matrices) are symbolized by lowercase (uppercase) boldface letters, e.g., $\boldsymbol{a}$, $\boldsymbol{A}$. We use $\{\boldsymbol{x}_i\}_{i=1}^{|\mathcal{X}|}$and $\{y_i\}_{i=1}^{|\mathcal{X}|}$ to present the input features and their labels of the training set $(\mathcal{X}, \mathcal{Y})$, where $|\mathcal{X}|$ denotes the cardinality. We symbolize by $K(\boldsymbol{x}, \boldsymbol{x}')$ the neural tangent kernel with input $\boldsymbol{x}$ and $\boldsymbol{x}'$, $\boldsymbol{K} \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$, where $\boldsymbol{K}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_i)$. Next, we denote by $\boldsymbol{\theta}_t$ the parameter vector, $\|\mathrm{e}(\boldsymbol{\theta}_t)\|_2^2$ the training loss, and $\hat{\boldsymbol{K}}_t$ the empirical NTK Gram matrix at time step t. The following notation is used:

$$\|\mathrm{e}(\boldsymbol{\theta}_t)\|_2^2 = \sum_{(\boldsymbol{x}_i, y_i)\in(\mathcal{X},\mathcal{Y})} (f(\boldsymbol{x}_i; \boldsymbol{\theta}_t) - y_i)^2 \,, \quad f(\boldsymbol{\theta}_t) = \mathrm{vec}(\{f(\boldsymbol{x}_i; \boldsymbol{\theta}_t)\}_{\boldsymbol{x}_i\in\mathcal{X}}) \in \mathbb{R}^{|\mathcal{X}|},$$

$$J(\boldsymbol{\theta}_t) = \frac{\partial f(\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{|\mathcal{X}|\times|\boldsymbol{\theta}|} \,, \qquad \hat{\boldsymbol{K}}_t = J(\boldsymbol{\theta}_t) J(\boldsymbol{\theta}_t)^\top \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}.$$

2.2 Neural tangent kernel

Neural networks (NNs) are relevant to kernel method, under proper initialization [Daniely et al., 2016, de G. Matthews et al., 2018]. Jacot et al. [2018] provably demonstrate the equivalence between the training dynamics by gradient descent and kernel regression induced by NTK when employing the ℓ_2 loss. Below, we recall the exact formula regarding the NTK of N-layer ($N > 2$) fully-connected NNs with ReLU activation functions σ. The corresponding NTK $K(\boldsymbol{x}, \boldsymbol{x}') = K_N(\boldsymbol{x}, \boldsymbol{x}')$ could be computed recursively by:

$$K_0(\boldsymbol{x}, \boldsymbol{x}') = \Sigma_0(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}' \,, \quad K_n(\boldsymbol{x}, \boldsymbol{x}') = \Sigma_n(\boldsymbol{x}, \boldsymbol{x}') + 2K_{i-1}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}_n(\boldsymbol{x}, \boldsymbol{x}') \,, \quad (1)$$

where the covariance Σ_n and its derivative $\dot{\Sigma}_n$ are defined as:

$$\Sigma_n(\boldsymbol{x}, \boldsymbol{x}') = 2\mathbb{E}_{(u,v)\sim\mathcal{N}(0,\boldsymbol{\Lambda}_i)}[\sigma(u)\sigma(v)]\,, \quad \dot{\Sigma}_n(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\boldsymbol{\Lambda}_i)}[\sigma'(u)\sigma'(v)]$$

$$\boldsymbol{\Lambda}_i = \begin{pmatrix} \Sigma_{i-1}(\boldsymbol{x}, \boldsymbol{x}) & \Sigma_{i-1}(\boldsymbol{x}, \boldsymbol{x}') \\ \Sigma_{i-1}(\boldsymbol{x}, \boldsymbol{x}') & \Sigma_{i-1}(\boldsymbol{x}', \boldsymbol{x}') \end{pmatrix},$$

for $i = 1, \ldots, N$. Furthermore, the aforementioned NTK is extended to residual NNs [Tirer et al., 2020, Huang et al., 2020], convolutional NNs [Arora et al., 2019], graph NNs [Du et al., 2019b], and recurrent NNs [Alemohammad et al., 2021]. One of the roles of such kernel is to analyze the training behavior of the neural network [Allen-Zhu et al., 2019, Chizat et al., 2019, Du et al., 2019a, 2018]. For instance, Lee et al. [2019] showcase that through NTK parameterization NNs trained via gradient descent of any depth evolve to linear models. Meanwhile, the inductive bias of convolutional networks, e.g., deformation stability of the images, has been studied in the NTK regime [Bietti and Mairal, 2019].

2.3 Neural networks with Hadamard product

The ideas of augmenting NNs with Hadamard products to allow multiplicative interactions can be traced back to at least to Ivakhnenko [1971] that investigate the learnable polynomial relationships. Most of early work e.g., Group Method of Data Handling [Ivakhnenko, 1971], pi-sigma network [Shin and Ghosh, 1991] do not scale well for high-dimensional signals. Chrysos et al. [2021c] factorize the weight of NNs-Hp based on tensor decompositions to reduce the number of parameters. They exhibit how to convert popular networks, such as residual networks, and convolutional NNs to the form of NNs-Hp. StyleGAN can be also considered as a special type of NN-Hp [Chrysos et al., 2019]. New efforts have recently emerged to improve the architecture of the network with Hadamard products [Chrysos et al., 2021b, Babiloni et al., 2021, Chrysos et al., 2021a]. In this work, we adopt complementary approach and focus on the extrapolation as well as the spectral bias from a theoretical perspective.

2.4 Extrapolation

The study of extrapolation properties of NNs dates at least back to the 90's [Barnard and Wessels, 1992, Kramer and Leonard, 1990]. Experimental results show poor performance of NNs in case of learning simple functions [Barnard and Wessels, 1992]. Browne [2002] also suggest that fully-connected NNs cannot extrapolate well and then illustrate how the representation of the input impact the extrapolation. Xu et al. [2021] provably present the extrapolation behavior of fully-connected NNs and Graph neural networks [Jacot et al., 2018]. Specifically, they show that 2-layer fully-connected NNs with ReLU activation function extrapolate to linear function in out-of-distribution (OOD) region. Our work exhibits that NNs-Hp can learn high degree nonlinear function. Apart from fully-connected NNs, Martius and Lampert [2016], Sahoo et al. [2018] showcase a novel family of functions with linear mapping and a non-linear transformation, which allows to use sine and cosine as nonlinearities, enabling such networks to learn well in analytical expressions. Note that there exist multiplication units in EQL, which is similar to the multiplicative interactions in NNs-Hp. Lastly, there are other type of OOD problems with specific setting among machine learning community [Shen et al., 2021]. Domain adaption assumes the source and the target domains lie in the same feature space but with different distributions [Kouw and Loog, 2019], which differs from extrapolation. Another category of methodology to solve the OOD generalization problem, called invariant learning, aims to discover high-level invariance feature from low-level observations through latent causal mechanisms [Arjovsky et al., 2019, Rosenfeld et al., 2021]. We believe our analysis can also encourage the usage of NNs-Hp in these OOD problems.

3 Analysis of NNs-Hp

Our analysis admits the following structure: we firstly study the NTK of NNs-Hp in Section 3.1, which allows us to conduct analysis towards extrapolation in Section 3.2, and spectral bias in Section 3.3.

3.1 Neural tangent kernel of NNs-Hp

We now develop the NTK for NNs-Hp, then we bridge the gap between the NNs-Hp trained by gradient descent with respect to squared loss and the kernel regression predictor involving the

NTK. Our analysis is conducted on a specific type of NNs-Hp called polynomial neural networks (PNNs) while we still use the term NN-Hp for consistence. The goal of such networks is to learn an N-degree ($N \geq 2$) polynomial expansion that outputs $f(\boldsymbol{x}) \in \mathbb{R}$ with respect to the input $\boldsymbol{x} \in \mathbb{R}^d$. For simplifying the proof, we consider the following formulation of NNs-Hp, which is a reparameterization version of PNNs [Zhu et al., 2022]. The output is given by:

$$\boldsymbol{y}_1 = \sqrt{\frac{2}{m}}\sigma(\boldsymbol{W}_1\boldsymbol{x}), \;\; f(\boldsymbol{x}) = \sqrt{\frac{2}{m}}(\boldsymbol{W}_{N+1}\boldsymbol{y}_N), \;\; \boldsymbol{y}_n = \sqrt{\frac{2}{m}}\sigma\left(\boldsymbol{W}_i\boldsymbol{x}\right) * \boldsymbol{y}_{i-1}, \; n = 2, \ldots, N\,, \tag{2}$$

where each element in $\boldsymbol{W}_{N+1} \in \mathbb{R}^{1\times m}$ and $\boldsymbol{W}_n \in \mathbb{R}^{m\times d}$, for $n = 1, \ldots, N$ is independently sampled from $\mathcal{N}(0, 1)$. Two remarks are in place: a) We multiply by the scaling factor $\sqrt{\frac{2}{m}}$ after each degree to ensure that the norm of the network output is preserved at initialization with infinite-width setting. b) ReLU is usually required to increase the performance of NNs-Hp in experiments [Chrysos et al., 2021c].

Lemma 1. *The NTK of N-degree NNs-Hp, denoted by $K(\boldsymbol{x}, \boldsymbol{x}')$, can be derived as:*

$$K(\boldsymbol{x}, \boldsymbol{x}') = 2N \cdot \langle \boldsymbol{x}, \boldsymbol{x}'\rangle \kappa_1(\boldsymbol{x}, \boldsymbol{x}')(\kappa_2(\boldsymbol{x}, \boldsymbol{x}'))^{N-1} + 2(\kappa_2(\boldsymbol{x}, \boldsymbol{x}'))^N\,, \tag{3}$$

where κ_1 and κ_2 are defined by taking the random Gaussian vector $\boldsymbol{w} \in \mathbb{R}^d$

$$\kappa_1 = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\sqrt{\frac{2}{m}}\cdot\boldsymbol{I})}\left(\dot{\sigma}(\boldsymbol{w}^\top\boldsymbol{x}) \cdot \dot{\sigma}(\boldsymbol{w}^\top\boldsymbol{x}')\right), \kappa_2 = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\sqrt{\frac{2}{m}}\cdot\boldsymbol{I})}\left(\sigma(\boldsymbol{w}^\top\boldsymbol{x}) \cdot \sigma(\boldsymbol{w}^\top\boldsymbol{x}')\right)\,. \tag{4}$$

The proof, which provided in Appendix B.1, is based on the standard NTK calculations. Differently from fully-connected NNs, the existence of multiplicative interaction in NNs-Hp induces the product form of multiple kernel.

Next, we provide the following theorem that gives a concrete requirement for the width of the networks that is sufficient for nonasymptotic convergence to the NTK at initialization,

Theorem 1. *(Convergence to the NTK at initializatoin). Consider N-degree NNs-Hp, and assume that the width $m \geq 2^{4N-2}\log^{2N-1}(2N/\delta)$ for any $\delta \in (0, 1)$. Then, given two inputs $\boldsymbol{x}, \boldsymbol{x}'$ on the unit sphere, with probability at least $1 - \delta$ over the randomness of initialization, we have that*

$$|\langle \nabla f(\boldsymbol{x}), \nabla f(\boldsymbol{x}')\rangle - K(\boldsymbol{x}, \boldsymbol{x}')| \leq 4N\rho e\sqrt{\frac{\log(2N/\delta)}{m}},$$

where $\rho = \sqrt{2}^{2N-1}\sqrt{8}e^3(2\pi)^{1/4}e^{1/24}\left(e^{2/e}(2N-1)/2\right)^{(2N-1)/2}$.

Remark: The above result corresponds to the networks where $\boldsymbol{\theta} = (\boldsymbol{W}_1, \cdots, \boldsymbol{W}_N)$ but not $\boldsymbol{W}_{N+1}$. It should be noted that the term ρ and the width are exponential with respect to the degree N , but the degree N is not large in practice, e.g., at most 15 in Chrysos et al. [2021a]. Hence the bound is fair and reasonable.

The technical key issue of the proof is to provide probability estimates for the multiplication of several sub-exponential random variables. To this end, we rely on the concentration of sub-Weibull random variables [Zhang and Chen, 2020] to complete the proof, which is deferred to Appendix B.2.

Below, we show that under certain conditions, the limiting NTK of NNs-Hp stays constant when training with gradient descent using squared loss.

Theorem 2 (Stability of the NTK during training)**.** *Consider the NNs-Hp in Eq.* (2)*. Assume $\lambda_{\min}(\boldsymbol{K}) > 0$ and the training data $(\mathcal{X}, \mathcal{Y})$ is contained in some compact set $\boldsymbol{x} \neq \tilde{\boldsymbol{x}}$ for all $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{X}$. Then, there exist some constant $R_0 > 0$, $M > 1$, and $Q > 1$ such that for every $m > M$, when minimizing the squared loss with gradient descent and sufficient small learning rate $\eta_0 < 2(\lambda_{\min}(\boldsymbol{K}) + \lambda_{\max}(\boldsymbol{K}))^{-1}$, the following inequality holds with high probability over the random initialization of $\boldsymbol{\theta}$:*

$$\|e(\boldsymbol{\theta}_t)\|_2 \leq \left(1 - \frac{\eta_0}{3}\lambda_{\min}(\boldsymbol{K})\right)^t R_0\,, \tag{5}$$

$$\sum_{j=1}^{t}\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j-1}\|_2 \leq \frac{3QR_0}{\lambda_{\min}(\boldsymbol{K})}\,, \tag{6}$$

$$\sup_t \|\hat{\boldsymbol{K}}_t - \hat{\boldsymbol{K}}_0\|_F = \frac{6Q^3R_0}{\lambda_{\min}(\boldsymbol{K})\sqrt{m}}\,. \tag{7}$$

Regarding the proof, we firstly introduce the norm control of the Gaussian weight matrices and then derive the local boundness and local Lipschitzness. The last step is to apply the induction rules over different time steps. Details are presented in the Appendix B.3.

Remark: Eq. (5) bounds the training loss and Eq. (6) implies small perturbation of the weights. Eq. (7) shows that $\hat{\boldsymbol{K}}_t \xrightarrow{n\to\infty} \hat{\boldsymbol{K}}_0$. Combining this with Theorem 1 that states $\hat{\boldsymbol{K}}_0 \xrightarrow{n\to\infty} \boldsymbol{K}$, we have $\hat{\boldsymbol{K}}_t \xrightarrow{n\to\infty} \boldsymbol{K}$. Thus, the equivalence to the kernel regression is established. Note that Theorem 2 is an extension from the corresponding theorem of NNs with residual connection [Tirer et al., 2020] to NNs with Hadamard product. This property allows us to characterize the training process as kernel regression.

3.2 Extrapolation behavior

The derived kernel above enables us to study how NNs-Hp with ReLU activation trained by gradient descent extrapolates in OOD regime.

Theorem 3 (γ-degree ($\gamma \leq N$) extrapolation of N-degree NNs-Hp)**.** *Suppose we train N-degree ($N \geq 2$) NNs-Hp $f : \mathbb{R}^d \to \mathbb{R}$ with Eq.* (2) *with infinite-width on $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{X}|}$, and the network is optimized with squared loss in the NTK regime. For any direction $\boldsymbol{v} \in \mathbb{R}^d$ that satisfies $\|\boldsymbol{v}\|_2 = \max\{\|\boldsymbol{x}\|_i^2\}$, let $\boldsymbol{x}_0 = t\boldsymbol{v}$ and $\boldsymbol{x} = \boldsymbol{x}_0 + h\boldsymbol{v}$ with $t > 1$ and $h > 0$ be the OOD data points, the output $f(\boldsymbol{x}_0 + h\boldsymbol{v})$ follows a γ-degree ($\gamma \leq N$) function with respect to h.*

Apart from NNs-Hp, we also consummate Xu et al. [2021] that considers the extrapolation of fully-connected NNs with only 2-layer. We provide the following generalized theorem for N-layer ($N > 2$) fully-connected NNs.

Theorem 4 (Linear extrapolation of N-layer fully-connected NNs)**.** *Suppose we train N-layer ($N \geq 2$) fully-connected NNs $f : \mathbb{R}^d \to \mathbb{R}$ on $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{X}|}$. For any direction $\boldsymbol{v} \in \mathbb{R}^d$ that satisfies $\|\boldsymbol{v}\|_2 = \max\{\|\boldsymbol{x}\|_i^2\}$, $\boldsymbol{x}_0 = t\boldsymbol{v}$ and $\boldsymbol{x} = \boldsymbol{x}_0 + h\boldsymbol{v}$ with $t > 1$ and $h > 0$ are OOD data points , the output $f(\boldsymbol{x}_0 + h\boldsymbol{v})$ follows a linear function with respect to h.*

We have already shown that NNs-Hp extrapolate to a function with specific degree and is more flexible than fully-connected NNs. However, only knowing the information of degree of the extrapolation function is not enough. Naturally, we might ask under which condition NNs-Hp can achieve successful extrapolation. Below, we build our analysis in NTK regime and show how the geometry of the training set affects the behavior of NNs-Hp.

Theorem 5 (Condition for exact extrapolation of NNs-Hp)**.** *Let $f_\rho(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta} \boldsymbol{x}$ be the target function with $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{\beta} \in \mathbb{R}^{d\times d}$. Suppose that $\{\boldsymbol{x}_i\}_{i=1}^{|\mathcal{X}|}$ contains orthogonal basis $\{\boldsymbol{e}_i\}_{i=1}^d$ and $\{-\boldsymbol{e}_i\}_{i=1}^d$, then if we train 2-degree NNs-Hp f with ReLU activation function on $\{(\boldsymbol{x}_i, f_\rho(\boldsymbol{x}_i))\}_{i=1}^{|\mathcal{X}|}$ with the squared loss in the NTK regime, then we have $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta} \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^d$.*

Due to the constrained space, the proof of the aforementioned theorems can be found in Appendix C.1 to C.3.

3.3 Spectral analysis

In this section, we characterize the approximation properties of N-degree NNs-Hp in the in-distribution regime. By studying the spectral analysis in the form of a Mercer decomposition, we explicitly show the eigenvalues and eigenfunctions of NTK.

We firstly introduce some notation. Denote by $\{Y_{k,j}\}_{j=1}^{N(d,k)}$ the spherical harmonics of degree k in $d+1$ variables. $G_k^{(\gamma)}$ represents the Gegenbauer polynomials with respect to the weight function $x \mapsto (1-x^2)^{\gamma-\frac{1}{2}}$ and degree k. Finally, denote by $F(d,k) := \frac{2k+d-1}{k}\binom{k+d-2}{d-1}$.

The following lemma enables to connect spherical harmonics to Gegenbauer polynomials.

Lemma 2 (Eq. (24) in Choraria et al. [2022])**.** *For any dot product Mercer kernel K, when the data $\boldsymbol{x}$ and $\boldsymbol{x}'$ are uniform on the unit sphere, we can apply the following Mercer's decomposition in terms*

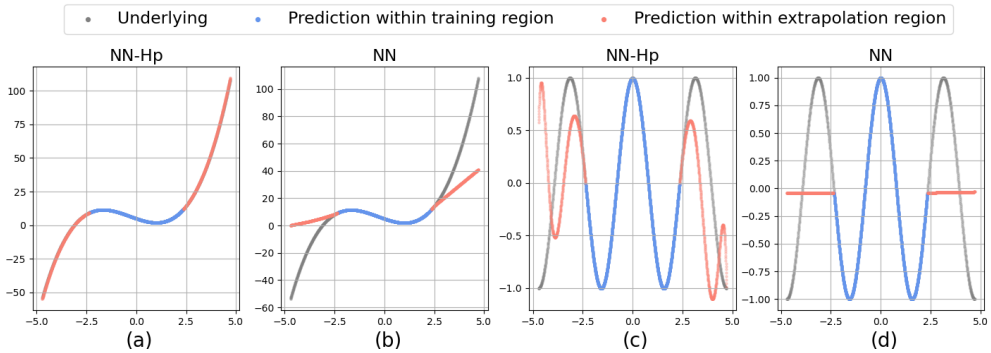


Figure 1: **Extrapolation function.** The horizontal coordinate of the blue curve indicate the input of the training regime. The extrapolation prediction is symbolized by pink color. (a) and (b) show the fitting results towards $f_\rho(x) = x^3 + x^2 - 10x + 5$. We can see that NN extrapolates linearly without the Hadamard product (Hp) while NN-Hp is able to extrapolate to the underlying non-linear function nearly. (c) and (d) present the fitting results towards $f_\rho(x) = \cos(2x)$. Notably, NN-Hp is more flexible to learn the non-linear function outsides the training region.

of spherical harmonics

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \sum_{k=0}^{\infty} \mu_k \sum_{j=1}^{N(d,k)} Y_{k,j}(\boldsymbol{x}) Y_{k,j}(\boldsymbol{x'}) = \sum_{k=0}^{\infty} \mu_k F(d,k) G_k^{(\frac{d-1}{2})}(\langle \boldsymbol{x}, \boldsymbol{x'} \rangle). \tag{8}$$

Theorem 6. *NNs-Hp with N-degree ($N \geq 2$) multiplicative interactions is considered. Denote by $\{\mu_1, \mu_2, \dots\}$ the eigenvalues associated to the NTK. Then for $k \gg d$, we have $\mu_k = \Omega((k)^{-d/2})$.*

The proof can be found in Appendix D.1. As a comparison, the decay rate for both deep fully-connected NNs and residual NNs is $\Omega((k)^{-d})$ [Belfer et al., 2021]. Thus, we can see a slower decay rate when inserting Hadamard product into standard NNs, which leads to a faster learning towards high-frequency information.

4 Experiments

Our experiment is organized as follows: We firstly showcase the extrapolation of NNs-Hp in learning some common used functions in Section 4.1. Next, we access the extrapolation performance on non-synthetic datasets in Section 4.2. Lastly, we conduct the experiment in learning spherical harmonics to study the effect of Hadamard product in Section 4.3.

4.1 Extrapolation in learning simple functions

These experiments aim to examine the extrapolation behavior of NNs-Hp in regression tasks and validate Theorem 3 and Theorem 4. Our first experiment includes training the networks via the squared loss to fit several well-known and simple underlying functions. Unless specified otherwise, N-layer NNs is compared against NN-Hp with $N-1$ degree multiplicative interactions in each experiment, the reason is explained in Appendix E.1. During prediction, we sample data points beyond the training regime and observe the extrapolation performance. More details on implementation can be found in Appendix E.1.

We set the target function as $f_\rho(x) = x^3 + x^2 - 10x + 5$ and use 4-layer fully-connected NN. As presented in Figure 1(a) and Figure 1(b) fully-connected NN extrapolates linearly while NN-Hp approximates better the out-of-distribution part of the underlying non-linear function, which are consistent with Theorem 3 and Theorem 4.

Learning $f_\rho(x) = \cos(2x)$. We choose 11-layer fully-connected NN. The training set and testing set are the same as in the previous experiment. Observing Figure 1(c) and Figure 1(d), we find that NN-Hp is more flexible to learn the non-linear function outside the training region while fully-connected NN still extrapolates linearly.

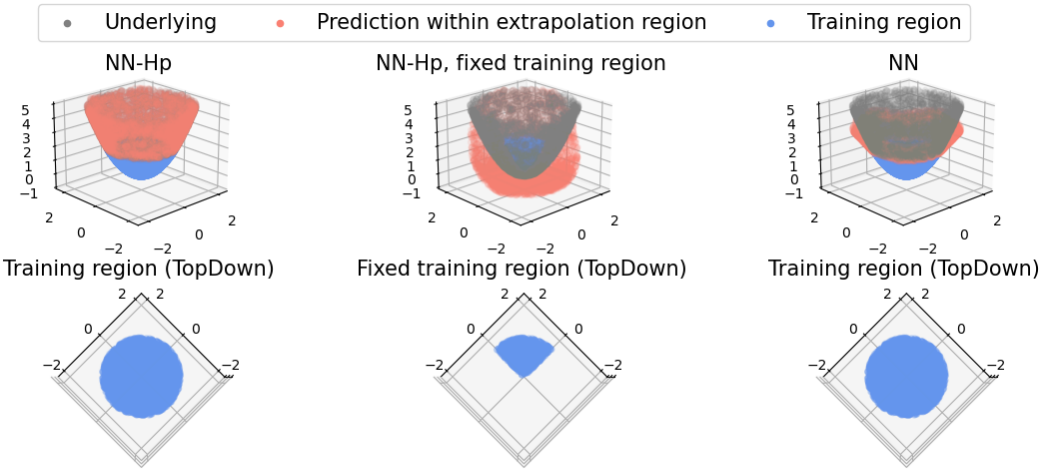


Figure 2: Fitting results for the underlying function $f_\rho(\boldsymbol{x}) = \boldsymbol{x}_{(1)}^2 + \boldsymbol{x}_{(2)}^2$, where $\boldsymbol{x}_{(1)}$ and $\boldsymbol{x}_{(2)}$ is the first and second dimension of $\boldsymbol{x} \in R^2$. **Left**: We train NNs-Hp with the training set contains support in all directions, the network is able to extrapolate successfully. **Middle**: We train NN-Hp with the training set where two dimension are fixed to be positive, NN-Hp fails to extrapolate. **Right**. We remove the Hadamard product of the network, which leads to linear extrapolation.

Learning $f_\rho(\boldsymbol{x}) = \boldsymbol{x}_{(1)}^2 + \boldsymbol{x}_{(2)}^2$, where $\boldsymbol{x}_{(1)}$ and $\boldsymbol{x}_{(2)}$ is the first and second dimension of $\boldsymbol{x} \in R^2$. In this task, we choose 3-layer NNs. Each model is trained with different data distribution, i.e., the training set contains support in all directions in the first case while two dimension of the training set are fixed to be positive in the second case. The result is visually depicted in Figure 2, which shows that NN-Hp can achieve exact extrapolation if the training set contains support in all directions and thereby validates Theorem 5. On the other hand, NN without the Hadamard product fails to extrapolate to the underlying function due to its linear extrapolation.

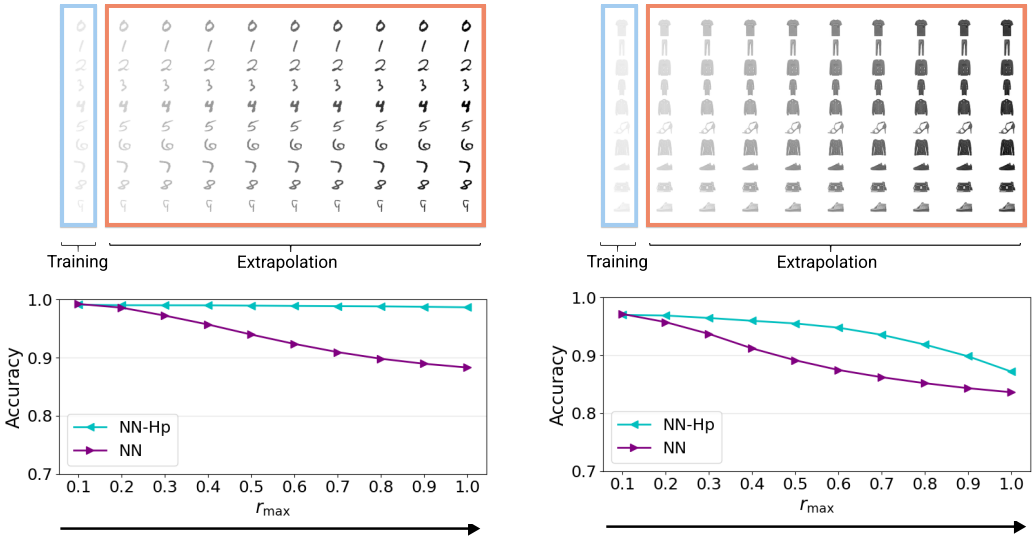
4.2 Extrapolation in real-world dataset

Previous experiments focus exclusively on learning simple functions and show that augmenting the Hadamard product makes NN becomes more flexible during extrapolation In this section, we access the extrapolation performance beyond synthetic datasets.

Variation of brightness. This experiment is preformed on two well-known grayscale image datasets: MNIST dataset [LeCun et al., 1998] and Fashion-MNIST dataset Xiao et al. [2017]. For these two datasets, the original range of the pixel of each image is $[0, 1]$, we divided it by 10 for the raw training set to construct the new one where the pixels range from 0 to 0.1. During extrapolation, we limit the range of the original testing set to $[0, r_{\text{max}}]$ through division, where $r_{\text{max}} \in \{0.1, 0.2, 0.3, ..., 1.0\}$, as illustrated in the top two panels in Figure 3. Then we feed these images into the trained network and evaluate the accuracy. We access 5-layer NNs in this task. More details on the implementation can be found in Appendix E.2. The accuracy is summarized in the two bottom plots of Figure 3. We find that both networks achieve similar accuracy in the case $r_{\text{max}} = 0.1$ while inserting Hadamard product (Hp) into NN improves the performance during extrapolation.

Arithmetic extrapolation. Now we turn to a more challenging task. As human we can usually extrapolate to arbitrarily large numbers in arithmetic. How do the neural networks perform during extrapolation? Following the setup of Bloice et al. [2020], we use MNIST dataset, where there are 100 different 2-image combinations of the digits $0 \sim 9$. We randomly pick up 90 combinations as the training set and the remaining 10 combinations as the extrapolation set. This problem is treated as regression instead of classification for higher error tolerance following Bloice et al. [2020]. In addition, if we design the network as a classifier, the number of the class will vary as the change of the splitting for the training set and testing set. The network only outputs one single discrete value. However, we still measure the accuracy by rounding the network output. 5-layer fully-connected NNs and convolution NNs are chosen as the baselines. For comparison, we implement NN-Hp with dense

layers and NN-Hp with convolution layers, respectively. More details on the implementation can be found in the Appendix E.3. The results obtained by a three-fold cross validation are summarized in Table 1, where we can see NN-Hp has a better extrapolation behavior in such more difficult task.



(a) Examples and results on MNIST dataset.

(b) Examples and results on Fashion-MNIST dataset.

Figure 3: The top two panels shows the examples of extrapolation in MNIST dataset and Fashion-MNIST dataset. $r_{\max}$ varies from 0.1 to 1.0. from left to right, indicating the variation of the darkness of the image. The bottom two panels shows the accuracy as $r_{\max}$ increasing. Both networks achieve similar accuracy in the case $r_{\max} = 0.1$ while inserting Hadamard product (Hp) into NN improves the performance during extrapolation.

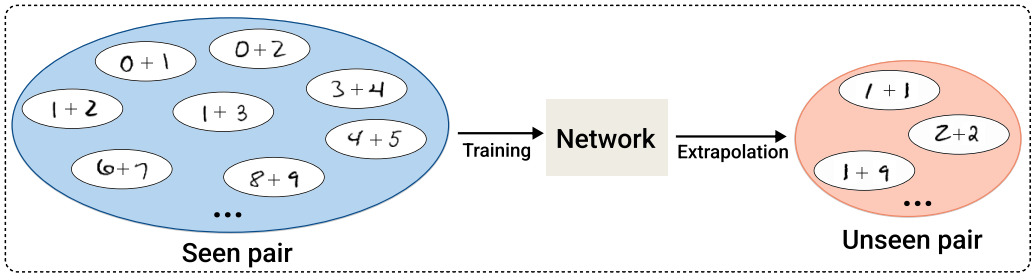


Figure 4: A schematic illustration for the task of arithmetic extrapolation. For the MNIST dataset, there are 100 different 2-image combinations of the digits $0 \sim 9$. We randomly pick up 90 combinations as the 'seen pair' and the remaining 10 combinations as the 'unseen pair'. During training, the networks are fed with some pair sampled from the 'seen pair'. During extrapolation, the network need to predict the sum of those samples from 'unseen pair'.

4.3 Spectral bias in learning spherical harmonics.

This experiment is to learn the linear combinations of spherical Harmonics where inputs is sampled from the uniform distribution on the unit sphere. The target function is defined by: $f^*(\boldsymbol{x}) = \frac{1}{N(\mathcal{K})}\sum_{k\in K} A_k P_k(\langle \boldsymbol{x}, \zeta_k\rangle), \quad \mathcal{K} = \{1, 3, 4, 5, 8, 12\}$, where $P_k(t)$ denotes the k -degree Gegenbauer polynomial, ζ_k are fixed vectors that are independently sampled from uniform distribution on unit sphere, and $N(\mathcal{K})$ is the normalizing constant. The error residuals with different $\mathcal{K}$ are compared during the training process. Implementation details are deferred to Appendix E.4. Note that Theorem 6 is a supplementary for the paper Choraria et al. [2022] wherein the theorem only

Table 1: Results in the task of arithmetic extrapolation, which aims to predict the target label with regression. 'Interpolation' indicates the accuracy in the seen pairs during training while 'Extrapolation' indicates the accuracy tested in those unseen pairs. Three ways are used for the network output: (a) Rounding, the output is rounded to the the nearest integer. (b) Floor/ceiling, A floor and ceiling function is applied for the output and if one of those equal to the ground truth label, we treat it as correct prediction. (c) ± 1. An error of ± 1 is allowed. We can observe that NN-Hp has a better extrapolation behavior compared with the baselines.

Method		Rounding	Floor/ceiling	± 1
NN(Dense)	Interpolation	0.980 ± 0.002	0.999 ± 0.000	0.999 ± 0.000
	Extrapolation	0.436 ± 0.065	$\mathbf{0.805 \pm 0.042}$	0.887 ± 0.011
NN-Hp (Dense)	Interpolation	0.926 ± 0.031	0.996 ± 0.001	0.999 ± 0.000
	Extrapolation	$\mathbf{0.554 \pm 0.011}$	0.802 ± 0.010	$\mathbf{0.889 \pm 0.008}$
NN(Conv)	Interpolation	0.945 ± 0.983	0.983 ± 0.021	0.994 ± 0.007
	Extrapolation	0.617 ± 0.103	0.918 ± 0.016	0.953 ± 0.006
NN-Hp (Conv)	Interpolation	0.991 ± 0.002	0.998 ± 0.000	0.999 ± 0.000
	Extrapolation	$\mathbf{0.825 \pm 0.109}$	$\mathbf{0.948 \pm 0.006}$	$\mathbf{0.963 \pm 0.007}$

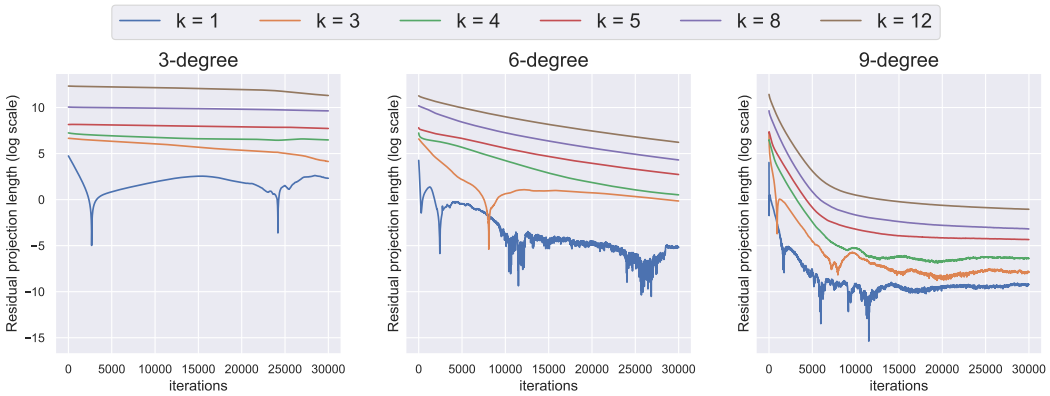


Figure 5: Comparison of convergence curve of error projection lengths for NNs-Hp with N order multiplicative interactions, where $N \in \{3, 6, 9\}$ for different order harmonics with $K \in \{1, 3, 4, 5, 8, 12\}$ We can see the improvement for NNs-Hp with high-degree multiplicative interactions in the rate of convergence of error.

considers NNs with only 2-order multiplicative interactions. In this experiment, we turn to show that increasing the number of multiplicative interactions can improve the rate of convergence of error, as presented in Figure 5.

5 Conclusion

This paper examines a specific type of neural network with Hadamard product, called, NNs-Hp, from a theoretical perspective. The analysis of the NTK paves the way for knowing interesting proporties of the networks, such as extrapolation behavior and spectral bias. Experimental results in learning simple functions validate our hypothesis. We further conduct several experiments in real-world datasets and demonstrate the advantage of inserting Hadamard product into standard neural networks. We believe not only our framework provides a good justification for a deeper understanding of neural architecture, but it also lays the foundations to investigate other more complicated OOD problems such as domain adaption and invariant learning in future work.

References

Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. In *International Conference on Learning Representations (ICLR)*, 2021.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning (ICML)*, pages 242–252. PMLR, 2019.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

Francesca Babiloni, Ioannis Marras, Filippos Kokkinos, Jiankang Deng, Grigorios Chrysos, and Stefanos Zafeiriou. Poly-nl: Linear complexity non-local layers with 3rd order polynomials. In *International Conference on Computer Vision (ICCV)*, pages 10518–10528, 2021.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

Etienne Barnard and LFA Wessels. Extrapolation and interpolation in neural network classifiers. *IEEE Control Systems Magazine*, 12(5):50–53, 1992.

Yuval Belfer, Amnon Geifman, Meirav Galun, and Ronen Basri. Spectral analysis of the neural tangent kernel for deep residual networks. *arXiv preprint arXiv:2104.03093*, 2021.

Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in neural information processing systems (NeurIPS)*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/c4ef9c39b300931b69a36fb3dbb8d60e-Paper.pdf`.

Marcus D Bloice, Peter M Roth, and Andreas Holzinger. Performing arithmetic using a neural network trained on digit permutation pairs. In *International Symposium on Methodologies for Intelligent Systems*, pages 255–264. Springer, 2020.

Antony Browne. Representation and extrapolation in multilayer perceptrons. *Neural computation*, 14 (7):1739–1754, 2002.

W.M. Campbell and C.C. Broun. Using polynomial networks for speech recognition. In *Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No.00TH8501)*, volume 2, pages 795–803 vol.2, 2000. doi: 10.1109/NNSP.2000.890159.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. *arXiv preprint arXiv:1912.01198*, 2019.

L. Carlitz. The product of two ultraspherical polynomials. *Glasgow Mathematical Journal*, 5(2): 76–79, July 1961. ISSN 2051-2104, 2040-6185. doi: 10.1017/S204061850003433X. URL `https://www.cambridge.org/core/journals/glasgow-mathematical-journal/article/product-of-two-ultraspherical-polynomials/BF734D19E2D88047AFD196033D20766A`. Publisher: Cambridge University Press.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

Moulik Choraria, Leello Tadesse Dadi, Grigorios Chrysos, Julien Mairal, and Volkan Cevher. The spectral bias of polynomial neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

Grigorios Chrysos, Stylianos Moschoglou, Yannis Panagakis, and Stefanos Zafeiriou. Polygan: High-order polynomial generators. *arXiv preprint arXiv:1908.06571*, 2019.

Grigorios Chrysos, Markos Georgopoulos, and Yannis Panagakis. Conditional generation using polynomial expansions. In *Advances in neural information processing systems (NeurIPS)*, 2021a.

Grigorios G. Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. P-nets: Deep polynomial neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Grigorios G Chrysos, Markos Georgopoulos, Jiankang Deng, and Yannis Panagakis. Polynomial networks in deep classifiers. *arXiv preprint arXiv:2104.07916*, 2021b.

Grigorios G. Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Jiankang Deng, Yannis Panagakis, and Stefanos P Zafeiriou. Deep polynomial neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2021c.

Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in neural information processing systems (NeurIPS)*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/abea47ba24142ed16b7d8fbf2c740e0d-Paper.pdf`.

Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning (ICML)*, pages 1675–1685. PMLR, 2019a.

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

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. *Advances in neural information processing systems (NeurIPS)*, 32, 2019b.

Feng-Lei Fan, Mengzhou Li, Fei Wang, Rongjie Lai, and Ge Wang. Expressivity and trainability of quadratic networks. *arXiv preprint arXiv:2110.06081*, 2021.

Pamela J Haley and DONALD Soloway. Extrapolation limitations of multilayer feedforward neural networks. In *International Joint Conference on Neural Networks*, volume 4, pages 25–30. IEEE, 1992.

Kaixuan Huang, Yuqing Wang, Molei Tao, and Tuo Zhao. Why do deep residual networks generalize better than deep feedforward networks?—a neural tangent kernel perspective. *Advances in neural information processing systems (NeurIPS)*, 33:2698–2709, 2020.

Alexey Grigorevich Ivakhnenko. Polynomial theory of complex systems. *IEEE transactions on Systems, Man, and Cybernetics*, 1971.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in neural information processing systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf`.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 2009.

Wouter M Kouw and Marco Loog. A review of domain adaptation without target labels. *IEEE transactions on pattern analysis and machine intelligence*, 43(3):766–785, 2019.

Mark A Kramer and JA Leonard. Diagnosis using backpropagation neural networks—analysis and criticism. *Computers & chemical engineering*, 14(12):1323–1338, 1990.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

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

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. 32, 2019.

Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016.

Quynh Nguyen, Marco Mondelli, and Guido F Montufar. Tight bounds on the smallest eigenvalue of the neural tangent kernel for deep relu networks. In *International Conference on Machine Learning (ICML)*, pages 8119–8129, 2021.

Elan Rosenfeld, Pradeep Kumar Ravikumar, and Andrej Risteski. The risks of invariant risk minimization. In *International Conference on Learning Representations (ICLR)*, 2021.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning (ICML)*, pages 4442–4450, 2018.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

Yoan Shin and Joydeep Ghosh. The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation. 1991.

Jiahao Su, Wonmin Byeon, Jean Kossaifi, Furong Huang, Jan Kautz, and Anima Anandkumar. Convolutional tensor-train lstm for spatio-temporal learning. *Advances in neural information processing systems (NeurIPS)*, 33:13714–13726, 2020.

Tom Tirer, Joan Bruna, and Raja Giryes. Kernel-based smoothness analysis of residual networks. *arXiv preprint arXiv:2009.10008*, 2020.

Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Taylor Webb, Zachary Dulberg, Steven Frankland, Alexander Petrov, Randall O'Reilly, and Jonathan Cohen. Learning representations that support extrapolation. In *International Conference on Machine Learning (ICML)*, pages 10136–10146. PMLR, 2020.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Keyulu Xu, Mozhi Zhang, Jingling Li, Simon Shaolei Du, Ken-Ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

Huiming Zhang and Song Xi Chen. Concentration inequalities for statistical inference. *arXiv preprint arXiv:2011.02258*, 2020.

Zhenyu Zhu, Fabian Latorre, Grigorios Chrysos, and Volkan Cevher. Controlling the complexity and lipschitz constant improves polynomial nets. In *International Conference on Learning Representations (ICLR)*, 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 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix F
   (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]
   (b) Did you include complete proofs of all theoretical results? [Yes] We provide detailed derivations in the 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] The datasets we use are all publicly available and commonly used in image-related tasks. We intend to release the source code upon the acceptance of the paper, which should enable interested practitioners to further improve our framework and verify our results.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We conduct three runs in our quantitative experiments, e.g., Table 1.
   (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)? [Yes] All our experiments are conducted on a single GPU, which is part of our cluster.
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]
   (b) Did you mention the license of the assets? [N/A] This is a theoretical work; we either use synthetic experiments, e.g. analytic functions as the target, or standard image-based datasets. All of the public datasets are publicly available and we cite their respecting papers, where the licenses are mentioned.
   (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]