Contents lists available at ScienceDirect
Artificial Intelligence
www.elsevier.com/locate/artint

Convolutional spectral kernel learning with generalization guarantees

Jian Li [a], Yong Liu [b,*], Weiping Wang [a]

[a] Institute of Information Engineering, Chinese Academy of Sciences, China
[b] Gaoling School of Artificial Intelligence, Renmin University of China, China

A R T I C L E I N F O

Article history:
Received 14 October 2021
Received in revised form 16 August 2022
Accepted 30 September 2022
Available online xxxx

Keywords:
Convolutional operator
Non-stationary spectral kernel
Cosine activation
Generalization analysis
Rademacher complexity

A B S T R A C T

Kernel methods are powerful tools to capture nonlinear patterns behind given data but often lead to poor performance on complicated tasks compared to convolutional neural networks. The reason is that kernel methods are still shallow and fully connected models, failing to reveal hierarchical features and local interdependencies. In this paper, to acquire hierarchical and local knowledge, we incorporate kernel methods with deep architectures and convolutional operators in a spectral kernel learning framework. Based on the inverse Fourier transform and Rademacher complexity theory, we provide the generalization error bounds for the proposed model and prove that under suitable initialization, deeper networks lead to tighter error bounds. Inspired by theoretical findings, we finally completed the convolutional spectral kernel network (CSKN) with two additional regularizers and an initialization strategy. Extensive ablation results validate the effectiveness of non-stationary spectral kernel, multiple layers, additional regularizers, and the convolutional filters, which coincide with our theoretical findings. We further devise a VGG-type 8-layers CSKN, and it outperforms the existing kernel-based networks and popular CNN models on the medium-sized image classification tasks.

1. Introduction

With solid theoretical guarantees and complete learning frameworks, kernel methods have achieved great success in various domains. However, compared to convolutional neural networks, kernel methods show inferior performance in practical applications because they failed in extracting rich representations for complex latent features. There are three problems that limit the representation ability of kernel methods:

- **Stationary representations** of kernels ignore long-range correlations between samples [1]. Conventional kernels are stationary because the kernel function is shift-invariant $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x} - \boldsymbol{x}')$ where the induced feature representations only depend on the distance $\|\boldsymbol{x} - \boldsymbol{x}'\|$ while free from inputs $\boldsymbol{x}$ themselves.
- **Kernel selection** greatly influences the empirical performance [2]. Cross-validation (CV) [3] and kernel target alignment (KTA) [2] were introduced to kernel selection, however, these methods split the process of kernel selection and model training.

* Corresponding author.
E-mail addresses: lijian9026@iie.ac.cn (J. Li), liuyonggsai@ruc.edu.cn (Y. Liu), wangweiping@iie.ac.cn (W. Wang).

https://doi.org/10.1016/j.artint.2022.103803
0004-3702/© 2022 Elsevier B.V. All rights reserved.

- **Kernel methods lack hierarchical or convolutional architecture.** Traditional kernels, e.g. Gaussian kernels, are equivalent to fully-connected neural networks with a single layer [4,5]. Thus, compared with CNN, kernel methods fail to extract the hierarchical features and local correlations.

To solve these three problems and promote the development of kernel methods, researchers have made many attempts. Non-stationary spectral kernels are proposed to construct more general kernels that learn from both the distance of inputs and the inputs themselves [6]. Approximate kernel selection accelerates kernel selection via matrix approximation [7]. Convolutional kernel networks use convolutional neural architectures to approximate kernel feature mappings [8,9]. However, the statistical properties of this method have not been fully revealed, and it lacks a general learning framework to settle these problems at the same time.

The article is organized as follows. Section 1 states the main contributions and related work. Section 2 presents preliminaries on spectral kernel methods and random features. In Section 3, we formally introduce conventional spectral kernel networks with random Fourier features and improve the algorithm motivated by theoretical results. In Section 4, we provide generalization analysis for spectral kernel networks and prove the superiority of non-stationary spectral kernels and multilayer architectures. In Section 5, we validate our approach by performing experiments on real-world datasets. Finally, we conclude this paper in Section 6. Throughout this paper, we present theoretical results in the main body and leave the proofs in the appendix.

1.1. Contributions

In this paper, we propose an effective learning framework that learns rich feature representations and optimizes kernel hyperparameters in an end-to-end way, which solves three crucial problems of kernel methods simultaneously in an end-to-end manner. Based on the learning framework, we derive a data-dependent generalization error bound via Rademacher complexity. Derived by the theoretical analysis, we devise the learning algorithm with two novel regularizers. This paper is a non-trivial extension of our previous work, published in AAAI 2020 [10]. The previous work presented the spectral kernel network with random Fourier features, to optimize the kernel hyperparameters and model weights in an end-to-end manner, and we also provided generalization analysis for the spectral kernel network. However, the previous architecture is fully connected and only with a single layer, which fails to capture hierarchical and local information for complicated tasks. Meanwhile, the generalization guarantees in the previous one only applied to the fully connected spectral kernel network with a single hidden layer. To overcome these drawbacks, we make the following significant improvements in this work:

1) **On the theoretical front.** We derived generalization error bounds of deep spectral kernel networks that reveal how the factors, including non-stationary spectral kernels, deep architecture, and initialization, affect the performance and suggest ways to improve the algorithm. Note that, we also provide a generalization interpretation of the superiority of deep neural networks over shallow networks by proving that under appropriate initializations deeper spectral kernel networks can lead to tighter generalization error bounds.
2) **On the algorithmic front.** The framework incorporates non-stationary spectral kernels with deep neural networks and convolutional operators to use their advantages of them. Intuitively, the proposed method characterizes better performance because the learned feature mappings are *input-dependent* (non-stationary spectral kernel), *output-dependent* (kernel learning), *hierarchical* (deep architecture) and *local related* (convolutional filters). Motivated by generalization results, we apply two additional regularizers in the learning objective. Furthermore, we propose `CSKN8` that adopts a VGG-type structure with convolutional filters and pooling layers but without skip connections, which achieves the state-of-the-art performance on image classification tasks.
3) **On the experimental front.** We conduct ablation experiments that validate the effectiveness of non-stationary spectral kernels, the depth of spectral kernel networks, the additional regularizers, and the convolutional filters. We then conduct comparison experiments with deep kernel networks and mainstream CNN models on six complicated image classification datasets. The experimental results illustrate that, on the medium-sized image datasets (except TinyImagenet), `CSKN8` outperforms both the existing kernel networks and popular CNN models with highest predictive accuracies and low computational burdens.

1.2. Related work

In this section, we introduce three kinds of related work that aim to relieve three bottlenecks in kernel methods.

Non-stationary Spectral Kernels. Yaglom's theorem provides spectral statements for general kernel functions via inverse Fourier transform. To break the limitation of stationary property, non-stationary spectral kernels were proposed with a concise spectral representation based on Yaglom's theorem [11,6]. Using Monte Carlo sampling, non-stationary spectral kernels were represented as neural networks [12] in Gaussian process regression, where kernel hyperparameters can be optimized together with the estimator. Then, [13,10] extended neural networks of non-stationary spectral kernels to general learning tasks. It has been proven that non-stationary spectral kernels can learn both *input-dependent* and *output-dependent* characteristics [10]. However, non-stationary spectral kernels fail to extract hierarchical features and local correlations, while deep convolutional neural networks naturally capture those characteristics and present impressive performance [14].

Kernel Learning. Kernel hyperparameters greatly decide the empirical performance of kernel methods, but kernel selection uses a lot of time to obtain favorable hyperparameters. Kinds of methods have tried to reduce the time consume via kernel learning instead of kernel selection. Multiple kernel learning (MKL) adaptively optimizes the combination of several candidate kernels together with the model parameters [15]. However, the training of MKL methods is complicated and requires suitable candidate kernels. More recently, researchers use random Fourier features to approximate kernel functions and stack random features layer by layer to update the corresponding spectral distribution of kernels via backpropagation [13,16].

Deep Convolutional Kernel Networks. Deep convolutional neural networks (CNNs) have achieved unprecedented accuracies in various domains including computer vision [14] and nature language processing [17]. Convolutional neural networks were encoded in a reproducing kernel Hilbert space (RKHS) to obtain invariance to particular transformations [8] in an unsupervised fashion. Then, combined with Nyström method, convolutional kernel networks were proposed in an end-to-end manner [18], while its stability to deformation was studied in [19]. Deep kernel learning (DKL) used neural networks as the front to extract features and Gaussian process as the backend classifier [20,21]. Recent research also explored the approximate theory of CNNs via downsampling and universality [22]. Besides, [23,9] introduced convolutional filters to spectral kernels and studied the len of spectrograms.

2. Preliminaries

Consider a supervised learning scenario where training samples $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ are drawn i.i.d. from a fixed but unknown distribution $\rho(\boldsymbol{x}, y)$ over the space $\mathcal{X} \times \mathcal{Y}$. Specifically, for general machine learning tasks, we assume the input space be $\mathcal{X} = \mathbb{R}^{d_0}$ and the output space be $\mathcal{Y} \subseteq \mathbb{R}^K$, where $K = 1$ for univariate labels (binary or regression) and $K > 1$ for multivariate labels (multi-classes or multi-labels).

Kernel methods include an implicit feature mapping $\phi : \mathcal{X} \to \mathcal{H}$ from the input space $\mathcal{X}$ to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, which is induced by a Mercer kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle$. Classical kernel methods fit the function $f : \mathcal{X} \to \mathcal{Y}$ which learns modes in the RKHS, admitting the linear form $f(\boldsymbol{x}) = \langle \boldsymbol{W}, \phi(\boldsymbol{x})\rangle_{\mathcal{H}}$. The hypothesis space is denoted by

$$H_\kappa = \Big\{ f \,|\, \boldsymbol{x} \to f(\boldsymbol{x}) = \langle \boldsymbol{W}, \phi(\boldsymbol{x})\rangle_{\mathcal{H}} \Big\}, \tag{1}$$

where $\boldsymbol{W} \in \mathcal{H} \times \mathbb{R}^K$ is the weight of the estimator and $\phi : \mathcal{X} \to \mathcal{H}$ is the feature mapping to characterize more powerful feature representations. The goal of supervised learning is to minimize the expected loss

$$\mathcal{E}(f) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(f(\boldsymbol{x}), y) d\rho(\boldsymbol{x}, y), \tag{2}$$

where ℓ is the loss function associated to specific tasks and $\rho(\boldsymbol{x}, y)$ is the joint probability distribution over $\mathcal{X} \times \mathcal{Y}$. Since the distribution $\rho(\boldsymbol{x}, y)$ is unknown, the expected loss $\mathcal{E}(f)$ cannot be computed directly. However, in practice, the learning algorithm aims to find a hypothesis under Empirical Risk Minimization (ERM) principal based on i.i.d. sample $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$:

$$\widehat{\mathcal{E}}(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(\boldsymbol{x}_i), y_i).$$

Here, we denote $f^* = \inf_{f\in H_\kappa} \mathcal{E}(f)$ the ideal learner in the hypothesis space while $\widehat{f}_n = \inf_{f\in H_\kappa} \widehat{\mathcal{E}}(f)$ the empirical estimator. Excess risk bound measures the generalization performance gap between f^* and $\widehat{f}_n$.

Shift-invariant kernels only depend on the distance $\tau = \boldsymbol{x} - \boldsymbol{x}'$, written as $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\tau)$. Commonly used kernels are shift-invariant (stationary), such as Gaussian kernels $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\tau\|_2^2)$ and Laplacian kernels $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\tau\|_1)$. According to Bochner's theorem, shift-invariant kernels are determined by its spectral density $s(\boldsymbol{\omega})$ via inverse Fourier transform [24].

Lemma 1 *(Bochner's theorem). A shift-invariant kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x} - \boldsymbol{x}')$ on $\mathcal{X}$ is positive definite if and only if it can be represented as*

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \int_{\mathcal{X}} e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')} s(\boldsymbol{\omega}) d\boldsymbol{\omega}, \tag{3}$$

where $s(\boldsymbol{\omega})$ is a non-negative probability density.

Shift-invariant kernels $\kappa(\tau) = \kappa(\boldsymbol{x} - \boldsymbol{x}')$ are stationary, which only take into account the distance $(\boldsymbol{x} - \boldsymbol{x}')$ but neglect useful information of the inputs themselves, also called stationary spectral kernels. However, the most general family of kernels are non-stationary, i.e. linear kernels $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$ and polynomial kernels $\kappa(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^\top \boldsymbol{x}' + 1)^r$.

Corollary 1 *(Stationary random Fourier features [25]). The shift-invariant kernels $\kappa(\boldsymbol{x},\boldsymbol{x}')=\int_{\mathbb{R}^d} e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')}s(\boldsymbol{\omega})d\boldsymbol{\omega}$ can be approximated by $\kappa(\boldsymbol{x},\boldsymbol{x}')\approx\langle\psi(\boldsymbol{x}),\psi(\boldsymbol{x}')\rangle$, where the random Fourier features admit the form*

$$\psi(\boldsymbol{x})=\sqrt{\frac{2}{D}}\left[\cos(\boldsymbol{\Omega}^\top\boldsymbol{x}+\boldsymbol{b})\right], \tag{4}$$

the frequency matrix $\boldsymbol{\Omega}\in\mathbb{R}^{d_0\times D}$ consists of D entries i.i.d. drawn from the spectral density $s(\boldsymbol{\omega})$ and the bias $\boldsymbol{b}\in\mathbb{R}^D$ is drawn uniformly from $[0,2\pi]^D$.

Note that, the inner product of stationary random Fourier features (4) is unbiased estimate of the shift-invariant kernels, i.e. $\kappa(\boldsymbol{x},\boldsymbol{x}')=\mathbb{E}[\psi(\boldsymbol{x})^\top\psi(\boldsymbol{x}')]$. Despite extensive literature [25–28] related to the generalization analysis of stationary random Fourier features that can only approximate shift-invariant kernels, there has been relatively few studies devoted to studying the generalization of non-stationary random features.

3. Convolutional spectral kernel learning

In this part, we first introduce the non-stationary spectral kernels, which can reveal long-range correlations and input-dependent characteristics among samples. Secondly, we extend non-stationary spectral kernels (only one hidden layer) into multilayer non-stationary spectral kernel networks, that is naturally stacked by inverse Fourier features. We finally complete the entire architecture by introducing convolutional operators instead of fully-connected operation into deep spectral kernel networks to discover local correlations.

3.1. Non-stationary spectral kernels

Lemma 2 *(Yaglom's theorem). A general kernel $\kappa(\boldsymbol{x},\boldsymbol{x}')$ is positive definite on $\mathcal{X}$ if and only if it admits the form*

$$\kappa(\boldsymbol{x},\boldsymbol{x}')=\int\limits_{\mathcal{X}\times\mathcal{X}} e^{i(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}'^\top\boldsymbol{x}')}\mu(d\boldsymbol{\omega},d\boldsymbol{\omega}'), \tag{5}$$

where $\mu(d\boldsymbol{\omega},d\boldsymbol{\omega}')$ is a Lebesgue-Stieltjes measure associated to some positive semi-definite (PSD) spectral density function $s(\boldsymbol{\omega},\boldsymbol{\omega}')$ with bounded variations.

Based on Yaglom's theorem (Lemma 2), the Fourier analysis theory has been extended to general spectral kernels, including both stationary and non-stationary cases [11,6]. Non-stationary spectral kernels depend on both the distance between inputs and inputs themselves, thus these kernels characterize more powerful representation ability and generalization performance.

To ensure a valid positive semi-definite spectral density in (5), we symmetrize spectral densities where $s(\boldsymbol{\omega},\boldsymbol{\omega}')=s(\boldsymbol{\omega}',\boldsymbol{\omega})$ and then introduce the diagonal components $s(\boldsymbol{\omega},\boldsymbol{\omega}),s(\boldsymbol{\omega}',\boldsymbol{\omega}')$, such that the kernel is defined as

$$\kappa(\boldsymbol{x},\boldsymbol{x}')=\frac{1}{4}\int\limits_{\mathcal{X}\times\mathcal{X}}\left(e^{i(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}'^\top\boldsymbol{x}')}+e^{i(\boldsymbol{\omega}'^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}')}+e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')}+e^{i\boldsymbol{\omega}'^\top(\boldsymbol{x}-\boldsymbol{x}')}\right)\mu(d\boldsymbol{\omega},d\boldsymbol{\omega}'). \tag{6}$$

Corollary 2 *(Non-stationary Random Fourier Features). For any positive define kernels defined in (6), the corresponding random Fourier features can be represented as*

$$\psi(\boldsymbol{x})=\frac{1}{\sqrt{2D}}\left[\cos(\boldsymbol{\Omega}^\top\boldsymbol{x}+\boldsymbol{b})+\cos(\boldsymbol{\Omega}'^\top\boldsymbol{x}+\boldsymbol{b})\right], \tag{7}$$

where these two frequency matrices $\boldsymbol{\Omega}=[\boldsymbol{\omega}_1,\boldsymbol{\omega}_2,\cdots,\boldsymbol{\omega}_D]\in\mathbb{R}^{d_0\times D}$, $\boldsymbol{\Omega}'=[\boldsymbol{\omega}_1',\boldsymbol{\omega}_2',\cdots,\boldsymbol{\omega}_D']\in\mathbb{R}^{d_0\times D}$ are paired Monte Carlo samples and the frequency pairs $\{(\boldsymbol{\omega}_k,\boldsymbol{\omega}_k')\}_{k=1}^D\in\mathbb{R}^{d_0}$ are drawn i.i.d. from the spectral density $s(\boldsymbol{\omega},\boldsymbol{\omega}')$. The phase vector $\boldsymbol{b}\in\mathbb{R}^D$ is drawn uniformly from $[0,2\pi]^D$.

The stationary (shift-invariant) spectral kernels (3) and corresponding random features (4) can capture the difference between examples but ignore the information of single example. Specifically, stationary spectral kernels of any example $\boldsymbol{x}\in\mathcal{X}$ are always the same $\kappa(\boldsymbol{x},\boldsymbol{x})=\int_{\mathcal{X}\times\mathcal{X}}e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x})}s(\boldsymbol{\omega})d\boldsymbol{\omega}=1$. Nevertheless, non-stationary spectral kernels (6) and random features (7) can capture the information of inputs themselves, for example the kernel of any input $\boldsymbol{x}\in\mathcal{X}$ depends on the input itself $\kappa(\boldsymbol{x},\boldsymbol{x})=\frac{1}{2}+\frac{1}{2}\int_{\mathcal{X}\times\mathcal{X}}e^{i(\boldsymbol{\omega}-\boldsymbol{\omega}')^\top\boldsymbol{x}}\mu(d\boldsymbol{\omega},d\boldsymbol{\omega}')$.

Remark 1. Yaglom's theorem illustrates that a general kernel $\kappa(\boldsymbol{x},\boldsymbol{x}')$ is associated to some positive semi-definite spectral density $s(\boldsymbol{\omega},\boldsymbol{\omega}')$ over frequencies $\boldsymbol{\omega},\boldsymbol{\omega}'$. Meanwhile, shift-invariant kernels (Bochner's theorem) is a special case of non-stationary spectral kernels (Yaglom's theorem) when the spectral measure is concentrated on the diagonal $\boldsymbol{\omega}=\boldsymbol{\omega}'$.

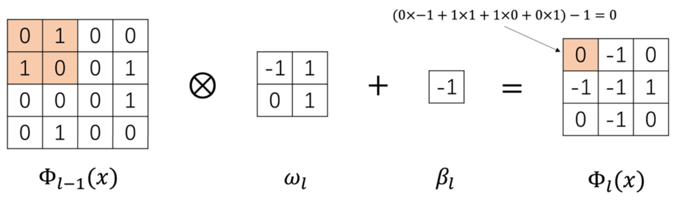

Fig. 1. The computation details of convolutional filters. For the sake of simplification, both the input and output channels are 1 and the convolutional filter is with the size 2×2. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Traditional kernel selection methods split the choice of hyperparameters and model learning. In contrast, the presented spectral kernel networks are trainable, thus we can optimize the kernel hyperparameters and model weights together, which are trained in an end-to-end manner.

3.2. Multilayer spectral kernel networks

In the view of neural networks, a non-stationary spectral kernel (6) is a single-layer neural network with infinite width, while the random Fourier approximation (7) reduces the infinite dimension to a finite width. Even though the non-stationary spectral kernels characterize input-dependent features, it is deficient in feature representations due to their shallow architecture.

In this paper, we design non-stationary spectral kernels network (SKN) by stacking random Fourier features in a hierarchical composite way:

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \Psi_L(\boldsymbol{x}), \Psi_L(\boldsymbol{x}') \rangle \qquad \text{with} \quad \Psi_L(\boldsymbol{x}) = \psi_L(\cdots \psi_2(\psi_1(\boldsymbol{x}))),$$

where the kernel κ consists of L-layers stacked spectral kernels and the feature mappings for any layer are approximated by random Fourier features (7). Based on the feature mapping of the last layer $\Psi_{l-1}(\boldsymbol{x})$, we definite the random Fourier mapping of l-th layer $\Psi_l : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}, l = 1, 2, \cdots, L$

$$\Psi_l(\boldsymbol{x}) = \frac{1}{\sqrt{2D}} \Big[\cos(\boldsymbol{\Omega}_l^\top \Psi_{l-1}(\boldsymbol{x}) + \boldsymbol{b}_l) + \ \cos(\boldsymbol{\Omega}_l'^\top \Psi_{l-1}(\boldsymbol{x}) + \boldsymbol{b}_l) \Big], \tag{8}$$

where $\Psi_0(\boldsymbol{x}) = \boldsymbol{x}$ is the input data, the frequency pairs in the l-th frequency matrices $\boldsymbol{\Omega}_l, \boldsymbol{\Omega}_l'$ are drawn i.i.d. from the l-th layer's joint spectral density $s_l(\boldsymbol{\omega}, \boldsymbol{\omega}')$. The elements in l-th phase vector $\boldsymbol{b}_l$ are drawn uniformly from $[0, 2\pi]^{d_l}$. The above architecture of deep spectral kernel networks is a kind of fully connected network (FCN), where the network includes L layers and two frequency matrices $\boldsymbol{\Omega}_l, \boldsymbol{\Omega}_l' \in \mathbb{R}^{d_l \times d_{l-1}}$ and two bias vectors $\boldsymbol{b}_l \in \mathbb{R}^{d_l}$ for the l-th layer. Therefore, the l-th layer involves $2 \times d_l \times (d_{l-1} + 1)$ parameters.

3.3. Convolutional spectral kernel networks

Even though the multilayer spectral kernel representations can learn input-dependent characteristics, long-range relationships, and hierarchical features, FCN fails to extract local correlations on the structural dataset, i.e. image and natural language. However, convolutional networks guarantee local connectivity, promising dramatic improvements in complex applications.

To characterize spatial locality, we combine spectral kernel networks with the convolutional filters, which enforces local connectivity pattern between the adjacent layers. For each channel of the l-th convolutional layer, the convolutional mapping $\Phi_l : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}$ is

$$\Phi_l(\boldsymbol{x}) = \frac{1}{\sqrt{2d_l}} \Big[\cos(\boldsymbol{\omega}_l \ \otimes \Phi_{l-1}(\boldsymbol{x}) + \boldsymbol{\beta}_l) + \ \cos(\boldsymbol{\omega}_l' \otimes \Phi_{l-1}(\boldsymbol{x}) + \boldsymbol{\beta}_l) \Big], \tag{9}$$

where $l = 1, \cdots, L$, $\Phi_0(\boldsymbol{x}) = \boldsymbol{x}$, $\otimes$ denoting convolutional operation and the l-th convolutional filters are pairwise $\boldsymbol{\omega}_l, \boldsymbol{\omega}_l' \in \mathbb{R}^{d_l'}$ in the filter size d_l'. The frequency pair $(\boldsymbol{\omega}_l, \boldsymbol{\omega}_l')$ is drawn from the joint spectral density $s(\boldsymbol{\omega}_l, \boldsymbol{\omega}_l')$ for l-th layer convolutional spectral kernel. The bias terms $\boldsymbol{\beta}_l$ is uniformly sampled from $[0, 2\pi]$ via Monte Carlo sampling.

We consider a 2D convolutional example as follows and other dimension convolutional computations are similar. Fig. 1 illustrates the computation details in the activation $\boldsymbol{\omega}_l \otimes \Phi_{l-1}(\boldsymbol{x}) + \boldsymbol{\beta}_l$, which involves a 2D convolutional filter in each layer. The convolutional operation summarizes the element-wise products on a 2×2 receptive field of last layer features $\Phi_{l-1}(\boldsymbol{x})$ and the convolutional filter $\boldsymbol{\omega}_l$, and then add the bias $\boldsymbol{b}_l$. For example, the first element in $\Phi_l(\boldsymbol{x})$ is computed by $(0 \times -1 + 1 \times 1 + 1 \times 0 + 0 \times 1) - 1 = 0$, where the corresponding block of the last layer is colored in Fig. 1. Then, the 2×2 receptive field sides to employ all features in the last layer.

As shown in Fig. 2, we define the convolutional spectral kernel network (CSKN) in a hierarchical kernel form by stacking convolutional spectral random features $\kappa(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \Phi_L(\boldsymbol{x}), \Phi_L(\boldsymbol{x}') \rangle$, which is finally followed with a FCN to model the linear

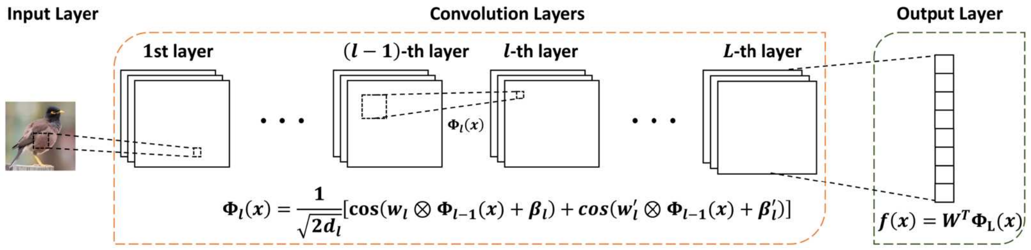
Input Layer
Convolution Layers
Output Layer
1st layer
$(l-1)$-th layer
l-th layer
L-th layer
$\Phi_l(x)$
$\Phi_l(x) = \frac{1}{\sqrt{2d_l}}[\cos(w_l \otimes \Phi_{l-1}(x) + \beta_l) + \cos(w_l' \otimes \Phi_{l-1}(x) + \beta_l')]$
$f(x) = W^T\Phi_L(x)$

Fig. 2. The structure of the proposed CSKN.

predictor $f(\boldsymbol{x}) = \boldsymbol{W}^\top \Phi_L(\boldsymbol{x})$. Moreover, the proposed CSKN can be further improved by integrating pooling layers and skip connections.

The proposed CSKN in Fig. 2 combines non-stationary spectral kernels, multilayer architectures, and convolutional operations, that help to discover long-range connections, hierarchical information, and local correlations among the training samples. Although the combination seems to be simple, it performs well in the empirical study. In the view of generalization analysis, it is a non-trivial extension of non-stationary spectral kernels due to the lack of theoretical learning on deep architectures and convolutional filters. The theoretical results in the next section can guarantee that deeper spectral kernel networks lead to tighter generalization error bounds than shallow ones, which bring a new perspective to explain the natural properties of neural networks and may inspire more ideas on the understanding of neural networks.

3.4. Improved algorithm

Motivated by the theoretical findings in Theorem 1 in the next section, we incorporate the empirical loss with two kinds of regularization terms in the minimization objective, written as

$$\mathop{arg\,min}_{\boldsymbol{W}, \Phi_L} \frac{1}{n}\sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i) + \lambda_1 \|\boldsymbol{W}\|_* + \lambda_2 \|\Phi_L(\boldsymbol{X})\|_F^2, \tag{10}$$

where the depth is L and $\Phi_L(\boldsymbol{X}) \in \mathbb{R}^{n\times d_L}$ is the feature mapping matrix on all input examples. The estimator is $f(\boldsymbol{x}_i) = \boldsymbol{W}^\top \Phi_L(\boldsymbol{x}_i) \in \mathbb{R}^K$, where the weighted matrix is $\boldsymbol{W} \in \mathbb{R}^{d_L\times K}$ and we employ the deep convolutional spectral kernel representations $\Phi_L : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ in a hierarchical composite way $\Phi_L(\boldsymbol{x}) = \Phi_L(\cdots\Phi_2(\Phi_1(\boldsymbol{x})))$ based on (9). The trace norm $\|\boldsymbol{W}\|_*$ regularizes the estimator weights and the squared Frobenius norm $\|\Phi_L(\boldsymbol{X})\|_F^2 = \sum_{i=1}^n \|\Phi_L(\boldsymbol{x}_i)\|_2^2 = \sum_{i=1}^n \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)$ is used to regularize the trace of kernel matrix. These two norms are scarcely used in conventional methods, where $\|\boldsymbol{W}\|_*$ represents the RKHS norm of primal kernel methods and $\|\Phi_L(\boldsymbol{X})\|_F^2$ regularizes the frequency pairs $(\boldsymbol{\omega}_l, \boldsymbol{\omega}_l')$.

Using backpropagation w.r.t. the objective, we update the model weights $\boldsymbol{W}$ and frequency pairs $(\boldsymbol{\omega}_l, \boldsymbol{\omega}_l')$ for convolutional layers in the objective (10), that makes the feature mappings $\Phi_l(\boldsymbol{x})$ dependent on the specific tasks. The spectral density $s(\boldsymbol{\omega}_l, \boldsymbol{\omega}_l')$ is modified via the update of frequency pairs $(\boldsymbol{\omega}_l, \boldsymbol{\omega}_l')$ in an end-to-end manner.

4. Theoretical assessment

In this section, we apply Rademacher complexity theory to spectral kernel networks, which reveals the generalization ability of kernel methods. We then explore the generalization performance of different kernels, including stationary and non-stationary spectral kernels and multilayer architectures. Finally, we improve the algorithm motivated by theoretical findings and discuss related work. We leave detailed proofs in the appendix.

4.1. Generalization guarantees for spectral kernel networks

In statistical learning theory, Rademacher complexity has been proposed to measure the richness of the hypothesis space, which has achieved great success in kernel methods (shallow network). However, it's still an open problem whether Rademacher complexity is applicative to deep neural networks [29]. We define the following Rademacher complexity for spectral kernel networks.

Definition 1. The empirical Rademacher complexity and expected Rademacher complexity of hypothesis space H_κ in (1) are defined as

$$\widehat{\mathcal{R}}(H_\kappa) = \frac{1}{n}\mathbb{E}_\xi\left[\sup_{f\in H_\kappa}\sum_{i=1}^{n}\sum_{k=1}^{K}\xi_{ik}[f(\boldsymbol{x}_i)]_k\right], \qquad \mathcal{R}(H_\kappa) = \mathbb{E}\,\widehat{\mathcal{R}}(H_\kappa),$$

where $[f(\boldsymbol{x}_i)]_k$ means the k-th value of the outputs and ξ_{ik}s is $n\times K$ independent Rademacher variables $\{\pm 1\}$ with the same probability.

The conventional Rademacher complexity was defined on a scalar output [30,31], while neural networks usually produce multi-dimensional outputs. To measure the model complexity of spectral kernel networks, we define the Rademacher complexity with K-dimensional outputs, i.e. $f : \mathcal{X} \to \mathbb{R}^K$.

Theorem 1 *(Excess risk bound for spectral kernel network). Assume the loss function ℓ is L-Lipschitz for $\mathbb{R}^K$ equipped with the 2-norm. With a probability at least $1-\delta$, the excess risk bound holds*

$$\mathcal{E}(\widehat{f}_n) - \mathcal{E}(f^*) \leq 4\sqrt{2}L\widehat{\mathcal{R}}(H_\kappa) + \mathcal{O}\Big(\sqrt{\frac{\log 1/\delta}{n}}\Big),$$

where $f^* = \inf_{f\in H_\kappa} \mathcal{E}(f)$ the target estimator and $\widehat{f}_n = \inf_{f\in H_\kappa} \widehat{\mathcal{E}}(f)$ the empirical estimator. The empirical Rademacher complexity $\widehat{\mathcal{R}}(H_\kappa)$ can be bounded by the trace of kernel matrix

$$\widehat{\mathcal{R}}(H_\kappa) \leq \frac{B}{n}\sqrt{K\sum_{i=1}^{n}\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)}, \tag{11}$$

where $B = \sup_{f\in H_\kappa} \|\boldsymbol{W}\|_* < \infty$ and K is the dimension of the outputs.

Based on Rademacher complexity, generalization error bounds of kernel methods have been well-studied [32], where the convergence depends on empirical Rademacher complexity $\widehat{\mathcal{R}}(H_\kappa)$. Meanwhile, empirical Rademacher complexity is determined by the trace of empirical kernel matrix $\sum_{i=1}^n \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)$. Therefore, the generalization performance of spectral kernel networks is relevant to the specific kernel.

Remark 2. From (11), we find that the minimization of Rademacher complexity need to minimize both B and the sum of diagonals $\sum_{i=1}^n \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)$. Since B is the upper bound of the trace norm $\|\boldsymbol{W}\|*$ and the trace holds $\sum_{i=1}^n \langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i)\rangle = \sum_{i=1}^n \|\phi(\boldsymbol{x}_i)\|_2^2 = \|\phi(\boldsymbol{X})\|_F^2$, we introduce $\|\boldsymbol{W}\|*$ and $\|\phi(\boldsymbol{X})\|_F^2$ as the regularizers to the objective in (10).

4.2. Generalization improvements from non-stationary spectral kernels

From Theorem 1, the generalization error bounds depend on the estimate of Rademacher complexity. In this part, we estimate Rademacher complexities for stationary spectral kernels and non-stationary spectral kernels. For the sake of simplification, we initialize the joint spectral density of non-stationary spectral kernels in (6) as two independent Gaussian distributions, i.e. $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and $\boldsymbol{\omega}' \sim \mathcal{N}(\mathbf{0}, \sigma'^2\mathbf{I})$.

Corollary 3 *(Rademacher complexity of shift-invariant kernels). If the kernel is shift-invariant defined in (3), the empirical Rademacher complexity $\widehat{\mathcal{R}}(H_\kappa)$ is bounded by*

$$\widehat{\mathcal{R}}(H_\kappa) \leq B\sqrt{\frac{K}{n}}.$$

For shift-invariant kernels, the diagonals of shift-invariant kernels identically equal to one regardless of the spectral density $s(\boldsymbol{\omega})$, since the fact $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) = \int_{\mathcal{X}\times\mathcal{X}} e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}_i - \boldsymbol{x}_i)} s(\boldsymbol{\omega}) d\boldsymbol{\omega} = 1$. The trace of kernel matrix is a constant $\sum_{i=1}^n \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) = n$. Then, one can obtain the above Corollary directly. Corollary 3 states that the convergence rate of Rademacher complexity is $\widehat{\mathcal{R}}(H_\kappa) \leq \mathcal{O}(\sqrt{K/n})$ when the norm of weight matrix is bounded $\|\boldsymbol{W}\|_* \leq c$ with some constant c [32].

Theorem 2 *(Rademacher complexity of non-stationary spectral kernels). If the kernel is non-stationary spectral kernel defined in (6), the empirical Rademacher complexity $\widehat{\mathcal{R}}(H_\kappa)$ is bounded by*

$$\widehat{\mathcal{R}}(H_\kappa) \leq \frac{B}{n}\sqrt{K\sum_{i=1}^{n}\frac{1}{2}\left[1 + \exp\left\{-\frac{1}{2}(\sigma^2 + \sigma'^2)\boldsymbol{x}_i^\top \boldsymbol{x}_i\right\}\right]},$$

where $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and $\boldsymbol{\omega}' \sim \mathcal{N}(\mathbf{0}, \sigma'^2\mathbf{I})$ are independent.

As shown in Theorem 2, since $(\sigma^2 + \sigma'^2) > 0$ and $\boldsymbol{x}_i^\top \boldsymbol{x}_i > 0$, the diagonal elements of non-stationary spectral kernels are always less than 1. Therefore, compared to stationary spectral kernels, non-stationary spectral kernels characterize smaller upper bounds of Rademacher complexity and thus have tighter generalization error bounds. The convergence rate of Rademacher complexity of non-stationary spectral kernel is still $\widehat{\mathcal{R}}(H_\kappa) = \mathcal{O}\big(\sqrt{K/n}\big)$, but the corresponding constant is smaller than that of stationary spectral kernels. Note that, shift-invariant kernels can be seen as a special case (the worst case) of spectral kernels with the diagonal density $\boldsymbol{\omega} = \boldsymbol{\omega}'$.

Remark 3 *(Approximation of stacked spectral kernels).* We derive above generalization analysis in the RKHS with implicit feature mappings. However, the computation of hierarchical stacked spectral kernels is intractable and optimal kernel hyperparameters are hard to estimate, so we stack explicit feature mappings via Monte Carlo approximation in (5) to approximate multilayers spectral kernels, where $\kappa(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \psi(\boldsymbol{x}), \psi(\boldsymbol{x}') \rangle$ and $\psi : \mathbb{R}^{d_0} \to \mathbb{R}^D$. According to Hoeffding's inequality, we bound the approximation error with the probability $1 - \eta$:

$$|\langle \psi(\boldsymbol{x}), \psi(\boldsymbol{x}') \rangle - \kappa(\boldsymbol{x}, \boldsymbol{x}')| \leq \sqrt{\frac{2}{D} \log \frac{2}{\eta}},$$

where $\eta \in (0, 1)$ is a small constant. The approximation error converges fast with the number of Monte Carlo samplings D. [25] has proven small approximate error ϵ is achieved by any constant probability when $D = \Omega(\frac{d_0}{\epsilon^2} \log \frac{1}{\epsilon})$. Besides, recent work revealed $D = \mathcal{O}(\sqrt{n})$ random features can achieve optimal learning rates in kernel ridge regression tasks [33,34].

4.3. Generalization improvements from deep architectures

Using the single-layer non-stationary spectral kernel (6) on each layer, we introduce multilayer non-stationary spectral kernels via

$$\kappa_L(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi_L(\boldsymbol{x}), \phi_L(\boldsymbol{x}') \rangle \qquad \text{with} \qquad \phi_L(\boldsymbol{x}) = \phi_L(\cdots \phi_2(\phi_1(\boldsymbol{x}))). \tag{12}$$

Theorem 3 *(Rademacher complexity of multilayer spectral kernel networks). If the kernel is a multilayer non-stationary spectral kernels and the kernel κ_L is defined as (12), the empirical Rademacher complexity $\widehat{\mathcal{R}}(H_\kappa)$ is bounded by*

$$\widehat{\mathcal{R}}(H_\kappa) \leq \frac{B}{n} \sqrt{K \sum_{i=1}^{n} \kappa_L(\boldsymbol{x}_i, \boldsymbol{x}_i)}.$$

For any input data $\boldsymbol{x}_i$ and $l \in [L]$, the diagonal of l-th layer $\kappa_l(\boldsymbol{x}_i, \boldsymbol{x}_i)$ is smaller than the diagonal of its previous layer $\kappa_{l-1}(\boldsymbol{x}_i, \boldsymbol{x}_i)$:

$$\kappa_l(\boldsymbol{x}_i, \boldsymbol{x}_i) \leq \kappa_{l-1}(\boldsymbol{x}_i, \boldsymbol{x}_i),$$

when $\kappa_{l-1}(\boldsymbol{x}_i, \boldsymbol{x}_i) > 1/2$ and the variance σ_l^2 satisfy $\forall i \in [n]$

$$\sigma_l^2 + {\sigma_l'}^2 \geq -\frac{2 \log\, [2\kappa_{l-1}(\boldsymbol{x}_i, \boldsymbol{x}_i) - 1]}{\kappa_{l-1}(\boldsymbol{x}_i, \boldsymbol{x}_i)}, \tag{13}$$

where σ^2 and σ'^2 are variances for random variables $\boldsymbol{\omega}_l \sim \mathcal{N}(\mathbf{0}, \sigma_l^2 \mathbf{I})$ and $\boldsymbol{\omega}_l' \sim \mathcal{N}(\mathbf{0}, {\sigma_l'}^2 \mathbf{I})$.

The results in Theorem 3 guide the design of the variance σ_l to get better generalization performance for deep neural networks. The right of inequality (13) has the decreasing property w.r.t. the diagonals $\kappa_{l-1}(\boldsymbol{x}_i, \boldsymbol{x}_i)$. To make deeper architecture available, we should ensure the decreasing on the diagonals $\kappa_l(\boldsymbol{x}_i, \boldsymbol{x}_i)$ w.r.t. the depth l, such that we enlarge σ_l for the increasing depth l. Based on the mean field theory, recent work has devised better initialization strategies [35] to improve the trainability, however, these strategies are irrelevant to the depth, ignoring the issues in generalization. It's worthy to further study the initialization schema which characterizes both good generalization ability and trainability. Under suitable initializations, the non-stationary spectral kernel $\kappa_L(\boldsymbol{x}, \boldsymbol{x})$ is closed to $1/2$ as the depth L becomes deeper. Such that, the estimate of Rademacher complexity is $\widehat{\mathcal{R}}(H_\kappa) \leq B\sqrt{\frac{K}{2n}}$, which is a half of that in Corollary 3.

Remark 4. (Generalization improvements from multilayers network) Theorem 3 holds for all diagonals $\kappa_l(\boldsymbol{x}_i, \boldsymbol{x}_i) \leq \kappa_{l-1}(\boldsymbol{x}_i, \boldsymbol{x}_i)$, thus the sum of diagonals magnifies the difference. With a favorable initialization schema, we obtain decreasing diagonals as the depth increases, which leads to tighter generalization error bounds. It's worth noting that, **we prove deeper architectures of neural networks can obtain better generalization performance** with suitable initialization. The theorem reveals the superiority of deep neural networks over shallow learning (such as conventional kernel methods) in the view of generalization. Moreover, the generalization gains from deeper architectures converge after some threshold, i.e. $\kappa(\boldsymbol{x}, \boldsymbol{x}) > 1/2$. After the threshold, it's hard to improve the generalization performance by increasing the number of layers.

Remark 5 *(Extension to general DNN).* We regard general neural networks as a composition of two parts: the feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^D$ before the last layer and the linear estimator $f(\boldsymbol{x}) = \langle \boldsymbol{W}, \phi(\boldsymbol{x}) \rangle$ in the last layer. Note that, the feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^D$ covers complicated architectures in deep neural networks (DNN), such as convolutional operators, recurrent layers, and skip connections. However, no matter how complicated the feature mappings are, Theorem 1 is still valid that the kernel with a smaller trace, in other words, the feature mapping with a smaller Frobenius norm on all examples $\|\phi(\boldsymbol{X})\|_F^2$, leads to tighter error bounds. Therefore, the techniques presented here can be applied to general DNN methods, and are of practical and theoretical interest in the understanding of general DNN.

Remark 6 *(Theoretical studies of CNN).* Deep convolutional neural networks [14,36] have achieved impressive accuracies which are often attributed to the efficient leverage of the local stationarity of natural images at multiple scales. The group invariance and stability to the action of diffeomorphisms were well-studied in [37,38,19]. Meanwhile, [22] studied the universality of deep convolutional neural networks and proved that CNNs can be used to approximate any continuous function to an arbitrary accuracy when the depth is large enough. However, the generalization ability of CNN was scarcely studied, because it's hard to extend the generalization results of FCN to CNN due to different structures. It's still not clear how to prove the superiority of convolutional networks in the view of generalization.

4.4. Comparison with related works

We compare the proposed approach and the theoretical findings with the most related works.

Non-stationary spectral kernel methods. Single hidden layer spectral kernel learning has been provided in [10], where they introduce the Rademacher complexity based error bounds for only single layer spectral kernels, with a convergence rate $\mathcal{O}(\sqrt{K/n})$ (same as Theorem 2). However, due to the complexity of stacked kernels, the error bounds for single-layer networks can not be extended trivially to multilayer cases. The work [10] also ignores the benefits of the use of multilayers and convolutional filters, leading to an inferior performance on image tasks.

Convolutional Multilayer Kernels. By optimizing the linear approximation to the Gaussian kernel rather than random Fourier features, [8] proposed the convolutional kernel network (CKN) that contained the local characteristics of the image. But, the Gaussian kernels in CKN are pre-defined while this paper jointly optimizes both kernel hyperparameters and model weights. [39,9] proposed stacked stationary random Fourier features and also extended it with convolutional layers (CRFFNet). However, CRFFNet is based on stationary random Fourier features that only captured the similarities between examples, while ours is based on non-stationary spectral kernels that capture the inputs themselves as well. More importantly, [39,9] lacks theoretical guarantees for the proposed architectures.

Deep kernel learning. Wilson et al. proposed Deep Kernel Learning (DKL) [20] that uses neural networks as the front feature exactor and the Gaussian process as the backend classifier. Nevertheless, only the Gaussian process in the last layer is related to kernel methods, and thus the front neural networks rather than the kernel layer play the dominant role in DKL. Meanwhile, DKL lacks generalization guarantees and the computing of the Gaussian process is time-consuming. Xue et al. presented a learning framework of multilayer spectral kernels [13], but they focused on the design of theoretical algorithms to develop spectral kernels for multilayer and the theoretical guarantees have not been explored. Meanwhile, they also failed to bridge the convolutional filters with the spectral kernels.

However, the existing kernel-based networks have not achieved state-of-the-art empirical results in complicated tasks, but also their theoretical analysis has been only marginally studied. In this paper, we provide theoretical guarantees for spectral kernel networks in this section and validate the empirical effectiveness of the proposed algorithms in the next section.

5. Experiments

To validate theoretical findings in Section 4 and the effectiveness of the proposed algorithm, we first conduct experiments for exploring the impacts of key factors for CSKN. Then, we compare CSKN with both convolutional kernel networks and popular CNN models on real-world image datasets, and report testing accuracy, running time, and computational complexities.

We implement compared methods on Pytorch [40] and conduct experiments on Nvidia RTX 2080 Ti (11G). We use Adam [41] as the optimizer and tune the learning rate η for different algorithms and datasets. Using 5-folds cross-validation, we tune regularized hyperparameters $\lambda_1, \lambda_2 \in \{10^{-10}, 10^{-9}, \cdots, 10^{-1}\}$ and optimizer hyperparameter $\eta \in \{10^{-5}, 10^{-4}, \cdots, 1\}$. We use the cross-entropy loss for multi-class classification tasks. All the code used in experiments is now publicly available.[1]

5.1. Ablation experiments

From the theoretical findings in section 4 and the refined algorithm CSKN, there are key factors that affect the performance, including the non-stationary spectral kernel, the depth of the network, convolutional filters, and two additional regularizers in (10). Based on the CIFAR100 dataset,[2] we explore the effects of key factors as follows. The CIFAR100 dataset consists of 32×32 colorful images in 100 classes, where 60000 images for training and 10000 images for testing. Using these factors, the predictive accuracy is improved from 28.83% to 40.84% for the CIFAR100 dataset.

[1] https://github.com/superlj666/CSKN/.
[2] https://www.cs.toronto.edu/~kriz/cifar.html.

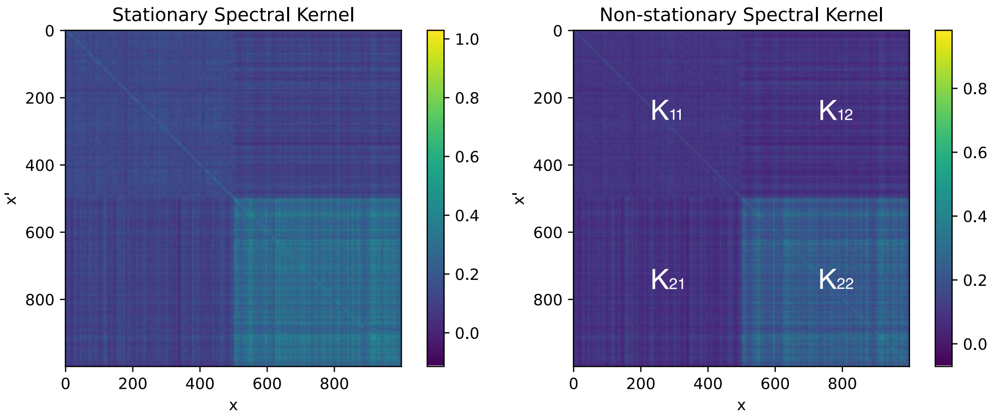

Fig. 3. Kernel matrices of stationary spectral kernel (left) and non-stationary spectral kernel (right). Here, the first 1000 images with the label 'beaver' and the rest with label 'hamster'.

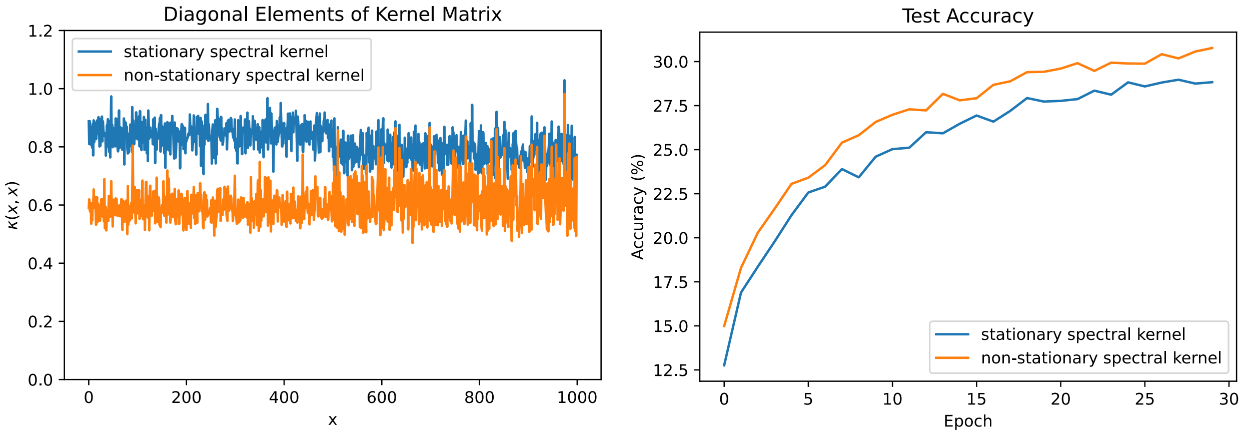

Fig. 4. The diagonal elements (left) and the accuracies (right) of stationary and non-stationary spectral kernels on the CIFAR100 dataset, respectively.

5.1.1. *The effect of non-stationary spectral kernels*

For stationary spectral kernel, we use random Fourier features to approximate Gaussian kernel $\kappa(\boldsymbol{x},\boldsymbol{x}')=\exp(-\sigma^2\|\boldsymbol{x}-\boldsymbol{x}'\|^2/2)\approx\langle\psi(\boldsymbol{x}),\psi(\boldsymbol{x}')\rangle$ from (4), where $\psi(\boldsymbol{x})=\sqrt{2/D}\cos(\boldsymbol{\Omega}^\top\boldsymbol{x}+\boldsymbol{b})$, where $\boldsymbol{\Omega}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I}_D)$ and $\boldsymbol{b}\sim[0,2\pi]^D$. For non-stationary spectral kernel, we approximate the constructed kernel in (6) by $\psi(\boldsymbol{x})=\sqrt{1/2D}\left[\cos(\boldsymbol{\Omega}^\top\boldsymbol{x}+\boldsymbol{b})+\cos(\boldsymbol{\Omega}'^\top\boldsymbol{x}+\boldsymbol{b})\right]$, where $\boldsymbol{\Omega}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I}_D), \boldsymbol{\Omega}'\sim\mathcal{N}(\mathbf{0},\sigma'^2\mathbf{I}_D)$, and $\boldsymbol{b}\sim[0,2\pi]^D$. We employ a single-hidden layer for both stationary and non-stationary spectral kernels and fixed the number of hidden units as $D=2000$. We set the epoch $T=30$, batch size 128, the learning rate $\eta_1=1e-5$ for non-stationary spectral kernel layers and $\eta_2=1e-2$ for the linear layer, and the kernel hyperparameters as $\sigma=1e-7,\sigma'=1e-2$.

As shown in Fig. 3, $\mathbf{K}_{11}$ and $\mathbf{K}_{22}$ measure the similarities among examples with the same labels while the other blocks measure the distance between examples with different labels. For the non-stationary spectral kernel, the distinctions between blocks are more obvious. We compute the average distance by $\|\mathbf{K}_{11}+\mathbf{K}_{22}-\mathbf{K}_{12}-\mathbf{K}_{21}\|/\mathrm{tr}(\mathbf{K})$ that the distance is 0.1717 for the stationary spectral kernel and 0.1807 for the non-stationary spectral kernel. Therefore, the non-stationary spectral kernel can distinguish examples with different labels more easily than the stationary spectral kernel.

In the left of Fig. 4, the diagonals of stationary are usually higher than that of non-stationary spectral kernels. Note that, since the Rademacher complexity depends on the trace of kernel matrix from Theorem 1, smaller diagonals can lead to better generalization performance, for example, the empirical predictive accuracies 28.83% v.s. 30.77% in the right of Fig. 4.

These experimental results validate the superiority of non-stationary spectral kernels and the theoretical findings in Theorem 2.

5.1.2. *The effect of multilayers spectral network*

Theorem 3 illustrates that multiple-layer spectral kernel networks can guarantee smaller with appropriate initializations where the variance should be bigger as the increase of network depth. To validate Theorem 3, we construct deep spectral kernel networks (DSKN) $f(\boldsymbol{x})=\langle\boldsymbol{W},\Psi_L(\boldsymbol{x})\rangle$ by stacking the non-stationary spectral random features $\Psi_l(\boldsymbol{x})=$

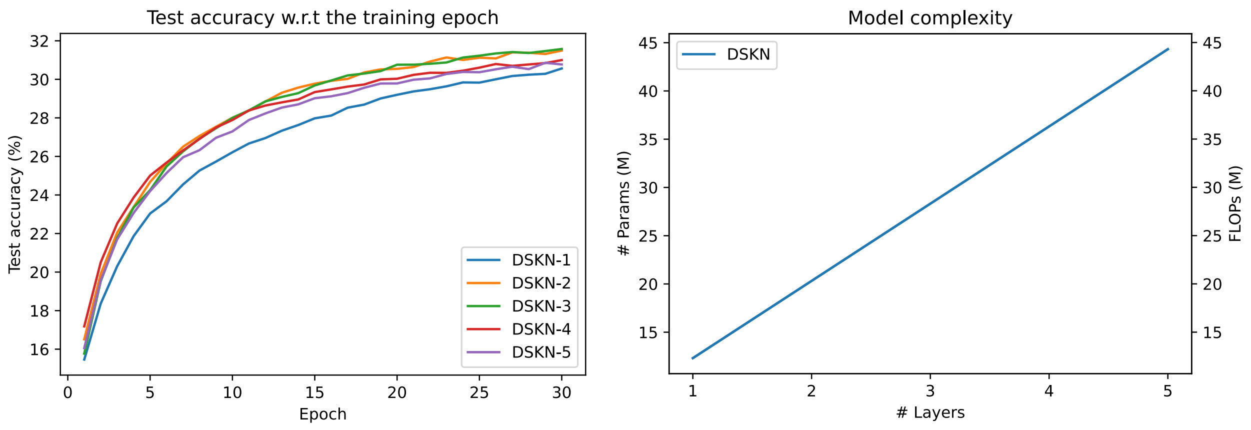

Fig. 5. The test accuracy w.r.t. the number of epochs (left) and the computational/storage costs versus the number of layers (right) on the CIFAR100 dataset.

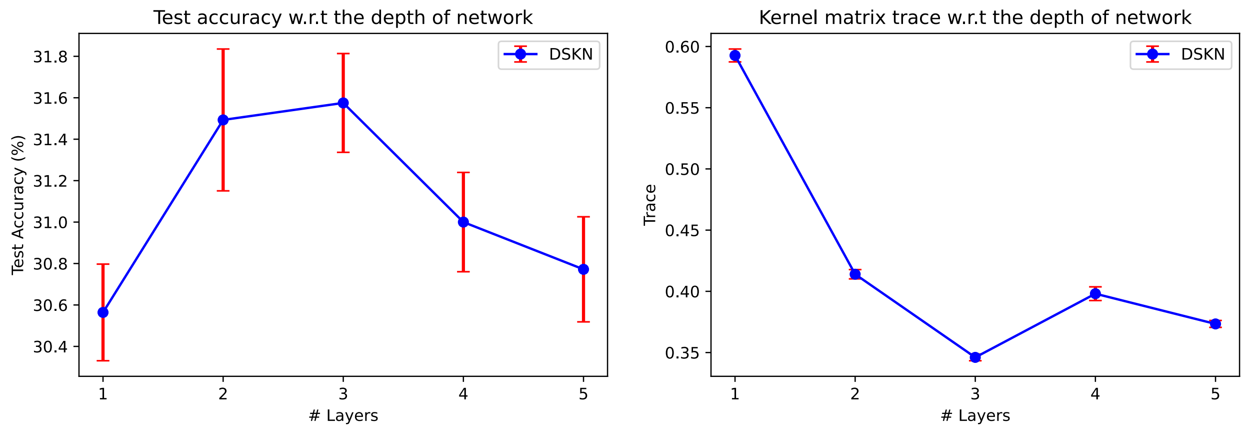

Fig. 6. The mean test accuracies at epoch 30 (left) and the trace of kernel matrix (right) versus the number of layers on the CIFAR100 dataset. DSKN-l represent non-stationary spectral networks (8) with the depth l.

$\frac{1}{\sqrt{2D}}\left[\cos(\boldsymbol{\Omega}_l^\top \Psi_{l-1}(\boldsymbol{x})+\boldsymbol{b}_l)+\cos(\boldsymbol{\Omega}_l'^\top \Psi_{l-1}(\boldsymbol{x})+\boldsymbol{b}_l)\right]$ according to (8) where $\boldsymbol{\Omega}_l \sim \mathcal{N}(\mathbf{0}, \sigma_l^2\mathbf{I})$, $\boldsymbol{\Omega}_l' \sim \mathcal{N}(\mathbf{0}, \sigma_l'^2\mathbf{I})$ and $\boldsymbol{b} \sim [0, 2\pi]^{d_l}$, and tune the increasing variances $\sigma_l^2, \sigma_l'^2$ for different the numbers of layers l. The features of initial layer are $\Psi_0(\boldsymbol{x}) = \boldsymbol{x}$.

According to Theorem 3, the initialization variance for multilayer CSKN should increase as the growth of the depth for obtaining smaller kernel traces. We use the same learning rates as the previous and set $\sigma_l = 1e-7 * \alpha^l, \sigma_l' = 1e-2 * \alpha^l$ to guarantee exponential growth, where α is the augment factor tuned by 5-folds cross-validation and l is the number of layers in the network. We set $\alpha = \{256, 32, 16, 8\}$ are suitable for 2-layers, 3-layers, 4-layers and 5-layers, respectively. We repeat the 10 trials for all depths to obtain the statistical significance of DSKN with different depths.

In terms of the number of layers, the left in Fig. 5 records the mean test accuracies of DSKN with different depths, while the right illustrates the increase of computational/storage burdens. Specifically, the mean test accuracies after 30 training epochs are 30.56%, 31.49%, 31.58%, 31.00%, and 30.77% for the number of layers of DSKN from 1 to 5, respectively. Since DSKN is a kind of fully connected network without skip connections and pooling layers, the computational burdens (FLOPs) and the storage costs (the number of parameters) are similar. As shown in the right of Fig. 5, the FLOPs and the number of parameters grow linearly as the increase of DSKN. Since the generalization gains are smaller when $L > 3$ but the computational costs increase linearly, the depth $L = 3$ achieves a tradeoff between generalization ability and computational efficiency.

Fig. 6 demonstrates that the test accuracy increases first and drops after $L = 3$ while the kernel trace exhibits the opposite curve, which coincides with Theorem 1 that smaller kernel traces can lead to better generalization performance. Indeed, the kernel trace does not decrease even with decreasing initialized variances as shown in Theorem 3. When the depth is bigger than 3, the kernel trace is bigger than that of $L = 3$, and the predictive ability of DSKN-4 and DSKN-5 is worse than DSKN-3. This observation coincides with the theoretical finding in Theorem 3 that the generalization performance of multilayer DSKN converges to a certain level after the depth is bigger than some thresholding depth, for example, the thresholding depth is $L = 3$ on the CIFAR100 dataset.

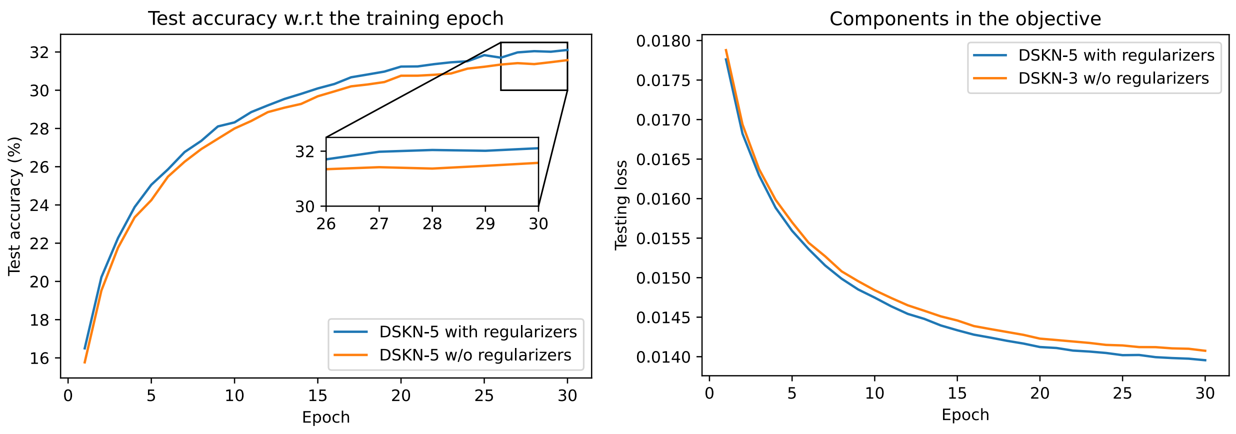

Fig. 7. The testing accuracy (left) and the test loss (right) of DSKN-3 with and without regularizers versus the number of training epochs on the CIFAR100 dataset.

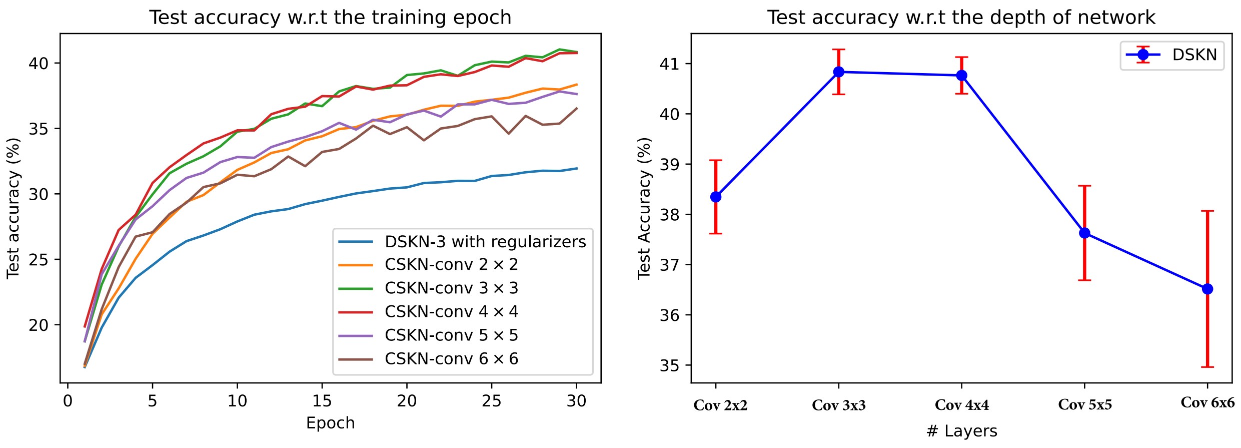

Fig. 8. The test accuracy w.r.t. the training epochs (left) and the mean test accuracies at epoch 30 w.r.t. different convolutional filters (right) on the CIFAR100 dataset.

5.1.3. The effect of additional regularizers

Motivated by the theoretical findings, we modify the proposed algorithms with two additional regularizers $\|\boldsymbol{W}\|_*$ and $\|\Phi_L(\boldsymbol{X})\|_F^2$ in (10). As illustrated in the last experiment, DSKN-3 achieves a good tradeoff between predictive accuracy and computational efficiency, and we thus use the 3-layers spectral network and the same initialization strategy and learning rates as DSKN-3. We set regularization hyperparameters as $\lambda_1 = 1e-8$, $\lambda_2 = 1e-4$ by tuning them via cross-validation and repeat experiments 10 times with random splits of training and testing set. Fig. 7 reports the mean testing accuracy and the mean testing loss w.r.t. the training epochs.

In the left of Fig. 7, with the additional regularizers, the test accuracy of DSKN-3 is improved from 31.58% to 32.11%. The right of Fig. 7 implies that the additional regularizers guide the DSKN-3 model to better fit the unseen data. These experimental results validate the effectiveness of the proposed learning framework in (10).

5.1.4. The effect of convolutional filters

To directly illustrate the effect of convolutional filters, we only consider convolutional operations in (9) while omitting pooling layers and skip connections. We compare the generalization performance of DSKN-3 and CSKN with different kernel sizes 2×2, 3×3, 4×4, 5×5, and 6×6. For a fair comparison, we stack 3 convolutional filters before the FCN layer and fix the number of out channels as 20.

As shown in Fig. 8, there are significant performance gaps between fully connected DSKN-3 (32.11%) and convolutional filters-based CSKN (36.52%+). This illustrates that CSKN can exploit spatial locality using convolutional filters for image tasks. The predictive accuracy rises first and then decreases as the range of the receptive field increases, where the average predictive accuracies are 38.35%, 40.84%, 40.77%, 37.63% and 36.52% for the filters 2×2, 3×3, 4×4, 5×5, and 6×6, respectively. Note that, CSKN with the filter 3×3 achieves the optimal predictive accuracy. When the receptive field is small, i.e. the filter 2×2, it can only capture relatively limited local information and ignores more local connectivity. When

Table 1
Model complexities of DSKN-3 and CSKN with different convolutional filters. Here, FLOPs are computed for one example rather than the entire dataset.

Complexity	DSKN-3	2×2	3×3	4×4	5×5	6×6
FLOPs	28.49M	7.82M	18.02M	25.92M	46.20M	56.08M
# Params	28.50M	1.69M	1.71M	1.74M	2.06M	2.09M

the receptive field is too large, for example, 5×5 and 6×6 convolutional filters, CSKN enforces local connectivity patterns on too many adjacent neurons that also fail to achieve optimal performance.

Since the CSKN with the kernel size 3×3 achieves the optimal predictive accuracy 40.84% with favorable computational burdens, CSKN with 3×3 convolutional filters is more suitable for the CIFAR100 dataset. Meanwhile, as shown in Table 1, the fully connected DSKN-3 owns higher computational and storage complexities than CSKN with the filter 3×3, which guarantees CSKN is especially suitable for high computational performance devices, e.g., GPUs. Therefore, the proposed CSKN cannot only significantly improve the predictive performance but also reduce the computational burdens.

5.2. Results on real-world datasets

In this part, to handle large image classification tasks, we incorporate popular techniques in neural networks including pooling layers and batch normalization with CSKN and propose an 8-layers VGG-type model named CSKN8. We compare the modified model with both kernel-based and convolutional neural networks based models as follows:

- **CSKN8** uses a similar architecture as VGG-11 [42] without skip connections. It consists of 7 convolutional hidden-layers with 3×3 kernel size for non-linear feature mappings and a FCN layer for the linear classification. The size of feature maps (the number of out channels) in convolutional layers is $64-64-128-128-256-256-512$. We use the batch normalization for all convolutional layers and 2×2 max-pooling layers after the 1st, 3rd, 5th and 7th convolutional layers. Before the last FCN layer, we also apply a 1×1 average-pooling layer.
- **CRFFNet8 [39,9]** stacks stationary random Fourier features (4) rather than non-stationary spectral random features (7) in the convolutional layers. For a fair comparison, we apply the same architecture for CRFFNet8 as CSKN8 but differ only in convolutional layers.
- **DKL [20,21]** trains an exact/approximate inference Gaussian process with a neural network based feature exactor in the end-to-end manner. Based on its implementation in GpyTorch,[3] we also use a medium-sized DenseNet (34-layers) as the front feature exactor.
- **Popular CNN models** include VGG-11 [42], ResNet-18 [43], DenseNet-121 [44], and ShuffleNetV2 [45]. We use their implementations and the pre-trained weights as the initialization in Pytorch.[4] The optimizer is Adam and the learning rate is widely-used $3e-4$.

Remark 7 *(The design of CSKN8).* The proposed CSKN is a flexible learning framework and can be incorporated with modern neural networks. For example, one can use (9) as the activation function instead of the Relu $\Psi_l(\boldsymbol{x}) = \max(0, \boldsymbol{x})$ in CNN models. Note that, VGG [42] is a sequential model and performs well without skip connections and VGG is compatible with our theoretical findings. ResNet [43] and DenseNet [44] with skip connections and deeper architectures, can improve the computational efficiency over VGG and relieve the overfitting, but these techniques are beyond the scope of our theoretical results. For the sake of fair comparison and coincidence with theoretical findings, we devise a VGG-type CSKN8 of the feature dimensions $64-64-128-128-256-256-512$ without skip connections. Compared to VGG-11 with hidden layers $64-128-256-256-512-512-512-512$, our proposed CSKN8 and CRFFNet8 employ fewer hidden layers and smaller feature dimensions, which reduce the computational and storage costs. Even though the proposed CSKN8 is smaller than VGG-11, the generalization performance of CSKN8 always outperforms that of VGG-11 as shown in Fig. 9 and Tables 3-5, which implies the superiority of non-stationary spectral random features over neural networks with the Relu activation. Moreover, CSKN8 also defeats other CNN models including ResNet-18 and DenseNet-121 on mediums-sized datasets and CSKN8 is comparable to ResNet on a large dataset, i.e. TinyImagenet.

We conduct comparison experiments on publicly available image classification datasets without data augmentation, including MNIST, FashionMNIST, SVHN, CIFAR10, CIFAR100, and TinyImagenet. Table 2 reports statistical information of these image classification datasets, where the first four datasets are also built-in in Pytorch.[5] Due to the limitation of computation resources, we use a subset of the ImageNet dataset, a.k.a. TinyImagenet,[6] instead of the entire ImageNet dataset. We train compared models on all datasets with a mini-batch size of 128. The maximum epochs are 300 for the medium-sized

[3] https://gpytorch.ai/.
[4] https://pytorch.org/vision/stable/models.html.
[5] https://pytorch.org/vision/stable/datasets.html.
[6] https://www.kaggle.com/c/tiny-imagenet.

Table 2
The statistics of image classification datasets. One channel means each example is a grayscale image, while 3 channels correspond to colored image with RGB values.

Dataset	# Train	# Testing	# Channels	Images size	# classes
MNIST	60000	10000	1	28×28	10
FashionMNIST	60000	10000	1	28×28	10
SVHN	73257	26032	3	32×32	10
CIFAR10	50000	10000	3	32×32	10
CIFAR100	50000	10000	3	32×32	100
TinyImagenet	100000	10000	3	64×64	200

Table 3
Average test accuracy (%) of the last 50 training epochs for each method on image classification datasets without data augmentation. Here, we bold the highest predictive accuracies for each dataset.

Dataset	**CSKN8**	CRFFNet8	DKL	ResNet	DenseNet	VGG	ShuffleNet
MNIST	**99.54**	99.40	99.20	99.48	99.28	99.20	98.87
FashionMNIST	**92.70**	92.03	91.86	91.90	92.19	92.03	89.29
SVHN	**94.29**	93.76	93.95	93.26	94.24	92.04	84.75
CIFAR10	**89.12**	88.42	86.88	84.87	86.57	86.90	71.17
CIFAR100	**64.50**	60.72	60.97	54.89	59.60	55.71	42.51
TinyImagenet	45.46	43.04	36.29	47.32	**54.49**	36.27	37.59

Table 4
Mean training time (seconds) of one training epoch / testing time (seconds) on the testing set for each method.

Dataset	CSKN8	CRFFNet8	DKL	ResNet	DenseNet	VGG	ShuffleNet
MNIST	7.78/0.74	6.59/0.71	36.28/1.01	7.98/0.85	32.68/0.99	24.17/0.90	13.00/0.84
FashionMNIST	8.86/0.75	6.50/0.71	36.74/1.02	7.90/0.85	31.50/0.96	24.39/0.89	12.62/0.83
SVHN	12.67/1.64	6.99/1.58	41.05/2.30	8.93/1.56	38.05/1.73	31.60/1.64	14.66/1.59
CIFAR10	10.95/0.90	6.25/0.89	28.31/1.16	6.90/0.90	26.58/0.98	21.59/0.90	10.84/0.89
CIFAR100	10.09/0.89	9.89/0.88	31.42/1.16	9.99/0.90	27.15/0.97	22.28/0.91	11.03/0.92
TinyImagenet	61.83/1.93	48.67/1.90	148.18/3.34	30.94/1.93	83.39/1.98	67.76/1.94	26.65/1.91

datasets, including MNIST, FashionMNIST, SVHN, CIFAR10, and CIFAR100, while we set the number of epochs as 500 for TinyImagenet.

We report the testing accuracies versus the training epochs in Fig. 9 and the mean accuracies of the last 50 epochs in Table 3. As shown in Fig. 9 and Table 3, we find that: 1) The proposed CSKN8 achieves promising predictive accuracies on image classification tasks, ranking first in 5 datasets except TinyImagenet. It outperforms other kernel-based models on all datasets and CNN models on medium-sized datasets. Specifically, the proposed CSKN8 is far ahead of the compared methods on the CIFAR100 dataset, of which the predictive accuracy is 3.53% higher than that of the second method. 2) On the TinyImagenet dataset, even though the predictive accuracy of CSKN8 is worse than DenseNet and ResNet at epoch 300, its performance is still on the rise and the gap shrinks as the increase of epochs and closed to that of ResNet at epoch 500. Meanwhile, although CSKN8 is similar but simpler than VGG-11, it performs much better than VGG on the TinyImagenet dataset. On the TinyImagenet dataset, the performance of VGG and DKL is not stable since they are prone to optimization problems for training them on large datasets. 3) CSKN8 adopts a similar architecture as VGG-11 with fewer convolutional and FCN layers, but CSKN8 achieves much higher predictive accuracy than VGG-11, which validates the effectiveness of non-stationary spectral kernels and additional regularizers. With skip connections, ResNet and DenseNet outperform VGG, which validates the effectiveness of skip connections. 4) Compared with other kernel-based methods CRFFNet8 and DKL, CSKN8 obtain higher predictive accuracies and perform usually more stable. Our implemented CRFFNet8 performs better than their primal implementations in [39,9]. On relatively hard tasks, i.e. CIFAR100 and TinyImagenet, the performance gaps between CSKN8 and CRFFNet8 are more obvious. The predictive performance of DKL is more unstable and it performs worst on the TinyImagenet dataset. 5) The kernel-based models (CSKN8, CRFFNet8, and DKL) achieve higher accuracies than advanced CNN models on medium-sized datasets but lower accuracies on the TinyImagenet dataset. The cause of this observation comes from the initialization and the optimization: kernel-based networks make use of periodic activation functions, i.e. cosine, characterize better expressive ability on small to medium-sized datasets with suitable initialization [46], but it is hard to tune suitable initialization and optimization hyperparameters for kernel-based networks on large datasets. However, advanced CNN models consist of Relu activation are much easier to be optimized and they are not very sensitive to initialization strategy [47,48].

We also report the training/inference times in Table 4 and FLOPs and the number of parameters in Table 5, which illustrate 1) The training time of CSKN8 is usually less than ShuffleNet on most datasets except TinyImagenet, while DKL owns the longest training time due to the complicated Gaussian process in the last layer. 2) VGG characterizes the largest number of parameters and FLOPs, while with fewer layers the proposed CSKN8 owns much fewer parameters, FLOPs, and

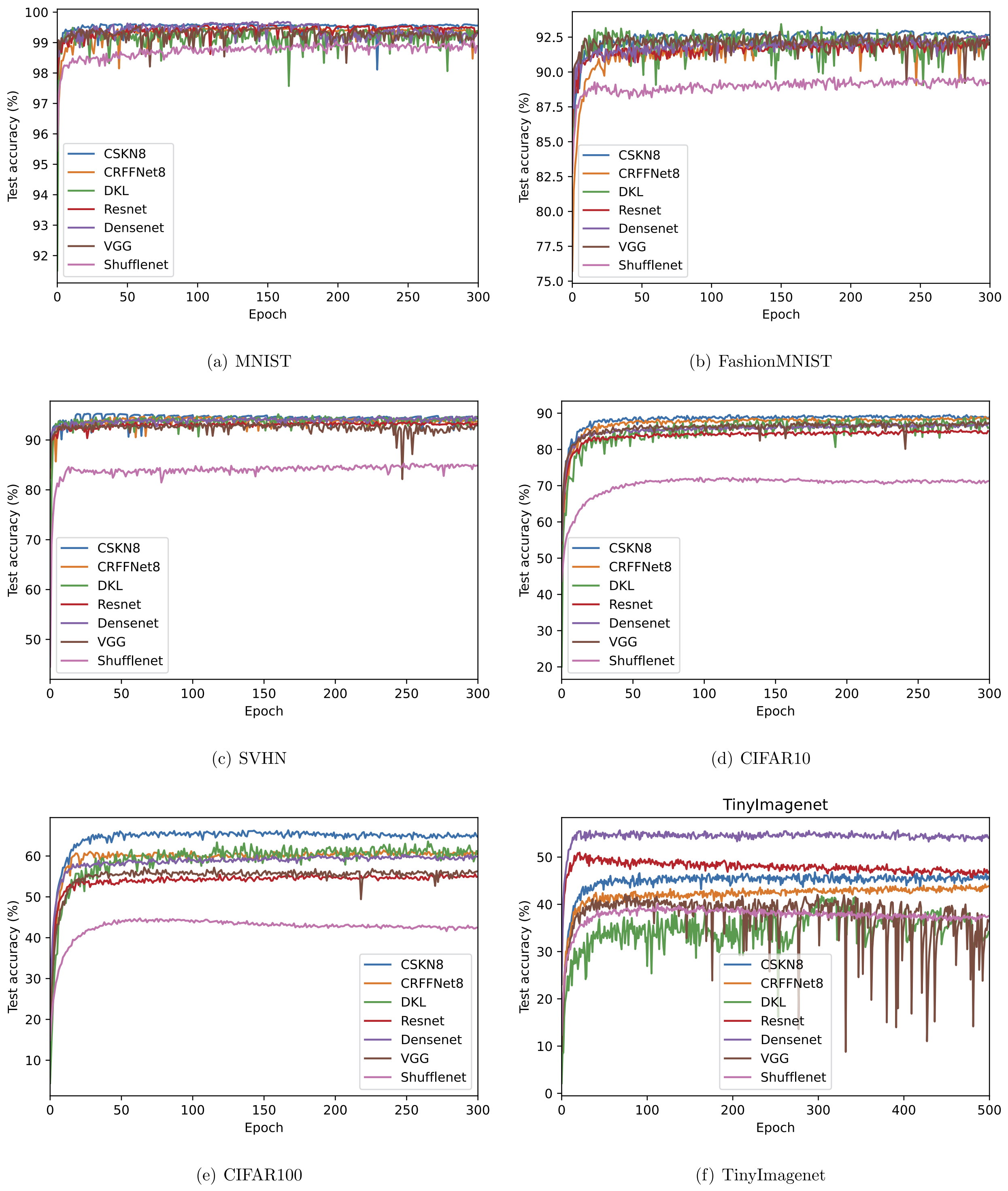

Fig. 9. The test accuracies versus the training epochs of compared methods on MNIST (a), FashionMNIST (b), SVHN (c), CIFAR10 (d), CIFAR100 (e), and TinyImagenet (f).

training time. 3) The ranking of FLOPs is not consistent with the training time due to the degree of parallel computation. For example, skip connections and the Gaussian process are relatively unfriendly to parallel computation and thus consume more training time. 4) The use of skip connections in ResNet and DenseNet reduces computational complexity for much deeper architectures. 5) The computational complexity of ShuffleNet is the lowest, but its predictive accuracy is also the lowest.

Table 5
FLOPs and the number of parameters for compared methods. FLOPs[†] represents the FLOPs of a MNIST example with the size $1 \times 28 \times 28$, while FLOPs[‡] is the FLOPs of a TinyImagenet example with the size $3 \times 64 \times 64$.

Dataset	CSKN8	CRFFNet8	DKL	ResNet	DenseNet	VGG	ShuffleNet
FLOPs[†]	119.92M	60.08M	73.96M	35.51M	56.87M	271.34M	0.75M
FLOPs[‡]	696.15M	348.73M	297.61M	148.46M	233.86M	731.27M	3.40M
# Params	4.74M	2.41M	0.73M	11.18M	6.96M	128.81M	0.35M

Indeed, on medium-size datasets, the proposed VGG-type `CSKN8` can lead to remarkably stronger learning ability than other kernel-based methods and popular CNN models, while its computational complexities are similar to ShuffleNet which are much lower than that of DKL and other CNN models. In future, ResNet-type / DenseNet-type `CSKN` with skip connections may lead to higher predictive accuracy, while ShuffleNet-type `CSKN` may achieve better tradeoffs between predictive accuracy and computational efficiency.

6. Conclusions

In this paper, we first integrate the non-stationary spectral kernel with deep convolutional neural network architecture, which optimizes the spectral density and the estimator together in an end-to-end manner. Then, from the perspective of generalization analysis, we prove non-stationary spectral kernel characterizes better generalization ability and deeper architectures lead to tighter error bounds with suitable initialization. Generalization analysis interprets the superiority of deep architectures and can be applied to general DNNs to improve their interpretability. To verify the theoretical findings, we conduct extensive ablation experiments to explore the effect of key factors. Besides, we design a VGG-type 8-layer model, a.k.a. `CSKN8`, and compare it with the existing deep kernel networks and popular CNN models. Experiments show that `CSKN8` achieves state-of-the-art predictive accuracies on medium-sized image datasets with low computational burdens.

6.1. Discussion and future work

The generalization ability of spectral kernel networks was rarely studied. Using Rademacher complexity, the generalization ability of spectral kernels was studied in [10]. The RKHS norm and spectral norm were considered to improve the generalization ability of neural networks [49,29,50]. Furthermore, [51,52] proposed that the learnability of deep modes involves both generalization ability and trainability.

Based on the mean field theory, [35,53] revealed that initialization schema determines both the trainability and the expressivity. Generalization analysis interprets the superiority of deep architectures and can be applied to general DNNs to improve their interpretability. Intuitively, the generating feature mappings enjoy the following benefits:

- 1) *input-dependent* (non-stationary spectral kernels).
- 2) *output-dependent* (backpropagation towards the objective).
- 3) *hierarchical represented* (deep architecture).
- 4) *local correlated* (convolutional operators).

Intuitively, the generating feature mappings enjoy the following benefits: 1) *input-dependent* (non-stationary spectral kernels), 2) *output-dependent* (backpropagation towards the objective), 3) *hierarchical represented* (deep architecture), 4) *local correlated* (convolutional operators).

While deep convolutional networks succeed in practice, theoretical analyses are overly pessimistic. In this paper, we apply convolutional operators to the spectral kernel network. We provide the learning algorithm and generalization guarantee for this network, which are beneficial from the excellent empirical performance of convolutional networks and solid theoretical foundation of kernel methods. We introduce the generalization theory of kernel methods into deep convolutional networks, which may inspire the research on the generalization of deep neural networks.

However, there are still a few tackle problems to be settled. For convolutional networks, current theoretical work focus on group invariance [37], stability [19] and approximation ability [22]. However, these theories can not explain why convolutional architectures work better than fully connected networks. In future work, we will try to explain the generalization ability of convolutional networks using downsampling [54] and locality. Besides, generalization analysis indicates that the initialization variance σ_l should be increased for the growth of depth, while σ_l decreases as l increases in current mean field theory work [53,55]. It's worth exploring the tradeoffs between generalization and optimization in terms of random initialization. Our work also can be incorporated with *Neural Tangent Kernel* (NTK) [56] to capture the dynamics of signals and conduct a simpler kernel.

On the TinyImagenet dataset, the predictive performance of `CSKN8` is better than the base model VGG but worse than DenseNet and ResNet. The reason is that without skip connections VGG-type models cannot be applied to deeper architectures due to heavy computational costs. In future work, we consider DenseNet-type / ResNet type `CSKN` with skip connections to obtain stronger learning ability on complicated datasets, for example ImageNet.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported in part by the Excellent Talents Program of the Institute of Information Engineering, CAS, Special Research Assistant Project of CAS (No. E0YY231114), Beijing Outstanding Young Scientist Program (No. BJJWZYJH012019100020098), and National Natural Science Foundation of China (No. 62076234, No. 62106257).

Appendix A. Proofs

In this section, we first provide proofs for random Fourier features of stationary spectral kernels and non-stationary spectral kernels, respectively. Then, we derive the generalization error bounds for spectral kernels. Finally, we estimate the Rademacher complexities for non-stationary spectral kernel and multilayer spectral kernel networks, respectively.

A.1. Random Fourier features approximation

We first prove the approximation between (4) and stationary spectral kernel (5) in Corollary 1.

Proof of Corollary 1. Using the Euler's formula, we have

$$e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')} = \cos\left(\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')\right) + i\ \sin\left(\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')\right).$$

Since the kernel function is real-valued, we discard the imaginary part

$$\kappa(\boldsymbol{x},\boldsymbol{x}') = \int_{\mathcal{X}} e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')} s(\boldsymbol{\omega})d\boldsymbol{\omega} = \int_{\mathcal{X}} \cos\left(\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')\right) s(\boldsymbol{\omega})d\boldsymbol{\omega}.$$

Based on Bochner's theorem, shift-invariant kernels can be represented by

$$\begin{aligned}
&\kappa(\boldsymbol{x},\boldsymbol{x}')\\
=&\int_{\mathcal{X}} \cos\left(\boldsymbol{\omega}^\top\boldsymbol{x} - \boldsymbol{\omega}^\top\boldsymbol{x}'\right) s(\boldsymbol{\omega})d\boldsymbol{\omega}\\
=&\frac{1}{2\pi}\int_0^{2\pi}\int_{\mathcal{X}} \left(\cos\left(\boldsymbol{\omega}^\top\boldsymbol{x} - \boldsymbol{\omega}^\top\boldsymbol{x}'\right) + \cos\left(\boldsymbol{\omega}^\top\boldsymbol{x} + \boldsymbol{\omega}^\top\boldsymbol{x}' + 2\theta\right)\right) s(\boldsymbol{\omega})d\boldsymbol{\omega}d\theta\\
=&\frac{1}{2\pi}\int_0^{2\pi}\int_{\mathcal{X}} 2\cos\left(\boldsymbol{\omega}^\top\boldsymbol{x} + \theta\right)\cos\left(\boldsymbol{\omega}^\top\boldsymbol{x}' + \theta\right) s(\boldsymbol{\omega})d\boldsymbol{\omega}d\theta\\
=&2\int_{\mathcal{X}} \cos\left(\boldsymbol{\omega}^\top\boldsymbol{x} + b\right)\cos\left(\boldsymbol{\omega}^\top\boldsymbol{x}' + b\right) s(\boldsymbol{\omega})d\boldsymbol{\omega},
\end{aligned}$$

where b is uniformly drawn from $[0, 2\pi]$. The last two steps use a Trigonometric identity $\cos(\alpha)\cos(\beta) = \frac{1}{2}\cos(\alpha-\beta) + \frac{1}{2}\cos(\alpha+\beta)$ for any $\alpha, \beta \in \mathbb{R}$.

Using Monte Carlo integration, we approximate the shift-invariant kernels with D random samples w.r.t. the distribution

$$\begin{aligned}
\kappa(\boldsymbol{x},\boldsymbol{x}') &\approx \frac{2}{D}\sum_{k=1}^{D}\left[\cos\left(\boldsymbol{\omega}_k^\top\boldsymbol{x} + b_k\right)^\top \cos\left(\boldsymbol{\omega}_k^\top\boldsymbol{x}' + b_k\right)\right]\\
&= \Bigg\langle \sqrt{\frac{2}{D}}[\cos(\boldsymbol{\omega}_1^\top\boldsymbol{x} + b_1), \cdots, \cos(\boldsymbol{\omega}_M^\top\boldsymbol{x} + b_M)],\\
&\qquad \sqrt{\frac{2}{D}}[\cos(\boldsymbol{\omega}_1^\top\boldsymbol{x}' + b_1), \cdots, \cos(\boldsymbol{\omega}_M^\top\boldsymbol{x}' + b_M)]\Bigg\rangle
\end{aligned}$$

$$
\begin{aligned}
&= \left\langle \sqrt{\frac{2}{D}}\Big[\cos(\boldsymbol{\Omega}^\top \boldsymbol{x}+\boldsymbol{b})\Big], \sqrt{\frac{2}{D}}\Big[\cos(\boldsymbol{\Omega}^\top \boldsymbol{x}'+\boldsymbol{b})\Big]\right\rangle \\
&= \langle \psi(\boldsymbol{x}), \psi(\boldsymbol{x}')\rangle,
\end{aligned}
$$

where $\{\boldsymbol{\omega}_k\}_{k=1}^D \overset{i.i.d.}{\sim} p(\boldsymbol{\omega})$, $\{b_k\}_{k=1}^D \overset{i.i.d.}{\sim} U(0,2\pi)$ and we denote the frequency matrix $\boldsymbol{\Omega}=[\boldsymbol{\omega}_1,\cdots,\boldsymbol{\omega}_D]$ and the phase vector $\boldsymbol{b}=[b_1,\cdots,b_D]$. $\square$

Using similar proof techniques as in the proofs of Corollary 1, we then prove the approximation between non-stationary spectral kernel (6) and random Fourier features (7) in Corollary 2.

Proof of Corollary 2. Based on Yaglom's theorem and Euler's formula, non-stationary spectral kernels can be represented by

$$
\begin{aligned}
&\kappa(\boldsymbol{x},\boldsymbol{x}') \\
=&\frac{1}{4}\int\limits_{\mathcal{X}\times\mathcal{X}} \Big(e^{i(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}'^\top\boldsymbol{x}')}+e^{i(\boldsymbol{\omega}'^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}')}+e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')}+e^{i\boldsymbol{\omega}'^\top(\boldsymbol{x}-\boldsymbol{x}')}\Big)s(\boldsymbol{\omega},\boldsymbol{\omega}')d\boldsymbol{\omega}d\boldsymbol{\omega}' \\
=&\frac{1}{4}\int\limits_{\mathcal{X}\times\mathcal{X}} \Big(\cos(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}'^\top\boldsymbol{x}')+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}') \\
&\quad+\cos(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}')+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}-\boldsymbol{\omega}'^\top\boldsymbol{x}')\Big)s(\boldsymbol{\omega},\boldsymbol{\omega}')d\boldsymbol{\omega}d\boldsymbol{\omega}' \\
=&\frac{1}{8\pi}\int_0^{2\pi}\int\limits_{\mathcal{X}\times\mathcal{X}} \Big(\cos(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}'^\top\boldsymbol{x}')+\cos(\boldsymbol{\omega}^\top\boldsymbol{x}+\boldsymbol{\omega}'^\top\boldsymbol{x}'+2\theta) \\
&\quad+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}')+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}'+2\theta) \\
&\quad+\cos(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}')+\cos(\boldsymbol{\omega}^\top\boldsymbol{x}+\boldsymbol{\omega}^\top\boldsymbol{x}'+2\theta) \\
&\quad+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}-\boldsymbol{\omega}'^\top\boldsymbol{x}')+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}-\boldsymbol{\omega}'^\top\boldsymbol{x}'+2\theta)\Big)s(\boldsymbol{\omega},\boldsymbol{\omega}')d\boldsymbol{\omega}d\boldsymbol{\omega}'d\theta.
\end{aligned}
$$

The last step is due to the fact $\frac{1}{2\pi}\int_0^{2\pi}cos(\alpha+2\theta)d\theta=0$ for any $\alpha\in\mathbb{R}$. Using Trigonometric identity $\cos(\alpha)\cos(\beta)=\frac{1}{2}\cos(\alpha-\beta)+\frac{1}{2}\cos(\alpha+\beta)$ for any $\alpha,\beta\in\mathbb{R}$, we then have

$$
\begin{aligned}
&\kappa(\boldsymbol{x},\boldsymbol{x}') \\
=&\frac{1}{4\pi}\int_0^{2\pi}\int\limits_{\mathcal{X}\times\mathcal{X}} \cos(\boldsymbol{\omega}^\top\boldsymbol{x}+\theta)\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}'+\theta)+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}+\theta)\cos(\boldsymbol{\omega}^\top\boldsymbol{x}'+\theta) \\
&+\cos(\boldsymbol{\omega}^\top\boldsymbol{x}+\theta)\cos(\boldsymbol{\omega}^\top\boldsymbol{x}'+\theta)+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}+\theta)\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}'+\theta)d\boldsymbol{\omega}d\boldsymbol{\omega}'d\theta \\
=&\frac{1}{2}\int\limits_{\mathcal{X}\times\mathcal{X}} \Big[\cos(\boldsymbol{\omega}^\top\boldsymbol{x}+b)+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}+b)\Big]\Big[\cos(\boldsymbol{\omega}^\top\boldsymbol{x}'+b)+\cos(\boldsymbol{\omega}'^\top\boldsymbol{x}'+b)\Big]d\boldsymbol{\omega}d\boldsymbol{\omega}',
\end{aligned}
$$

where b is drawn uniformly from $[0,2\pi]$. Then, using Monte Carlo sampling, we approximate the kernel by

$$
\begin{aligned}
&\kappa(\boldsymbol{x},\boldsymbol{x}') \\
\approx&\frac{1}{2D}\sum_{k=1}^{D}\Big[\cos(\boldsymbol{\omega}_k^\top\boldsymbol{x}+b)+\cos(\boldsymbol{\omega}_k'^\top\boldsymbol{x}+b)\Big]\Big[\cos(\boldsymbol{\omega}_k^\top\boldsymbol{x}'+b)+\cos(\boldsymbol{\omega}_k'^\top\boldsymbol{x}'+b)\Big] \\
=&\Bigg\langle\sqrt{\frac{1}{2D}}\cos(\boldsymbol{\omega}_1^\top\boldsymbol{x}+b)+\cos(\boldsymbol{\omega}_1'^\top\boldsymbol{x}+b),\cdots,\cos(\boldsymbol{\omega}_D^\top\boldsymbol{x}+b)+\cos(\boldsymbol{\omega}_D'^\top\boldsymbol{x}+b), \\
&\sqrt{\frac{1}{2D}}\cos(\boldsymbol{\omega}_1^\top\boldsymbol{x}'+b)+\cos(\boldsymbol{\omega}_1'^\top\boldsymbol{x}'+b),\cdots,\cos(\boldsymbol{\omega}_D^\top\boldsymbol{x}'+b)+\cos(\boldsymbol{\omega}_D'^\top\boldsymbol{x}'+b)\Bigg\rangle \\
=&\left\langle\frac{1}{\sqrt{2D}}\Big[\cos(\boldsymbol{\Omega}^\top\boldsymbol{x}+\boldsymbol{b})+\cos(\boldsymbol{\Omega}'^\top\boldsymbol{x}+\boldsymbol{b})\Big],\frac{1}{\sqrt{2D}}\Big[\cos(\boldsymbol{\Omega}^\top\boldsymbol{x}'+\boldsymbol{b})+\cos(\boldsymbol{\Omega}'^\top\boldsymbol{x}'+\boldsymbol{b})\Big]\right\rangle
\end{aligned}
$$

$$= \langle \psi(\boldsymbol{x}), \psi(\boldsymbol{x}')\rangle,$$

where $\{(\boldsymbol{\omega}_k, \boldsymbol{\omega}'_k)\}_{k=1}^D \overset{i.i.d.}{\sim} s(\boldsymbol{\omega}, \boldsymbol{\omega}')$, $\{b_k\}_{k=1}^D \overset{i.i.d.}{\sim} U(0, 2\pi)$ and we denote $\boldsymbol{\Omega} = [\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_D]$, $\boldsymbol{\Omega}' = [\boldsymbol{\omega}'_1, \cdots, \boldsymbol{\omega}'_D]$ and $\boldsymbol{b} = [b_1, \cdots, b_D]$. □

A.2. Generalization guarantees for spectral kernel networks

In this section, we provide detailed proofs for the generalization error bounds of kernel methods, and the empirical estimates of Rademacher complexities of non-stationary spectral kernels and multilayer spectral networks, respectively. Rademacher complexity on the hypothesis space is given in Definition 1, but the measure of generalization error bounds is directly related to the loss function rather than the estimator. Thus, we also define Rademacher complexity on the loss space as follows.

Definition 2. We define the empirical Rademacher complexity over the loss space $\mathcal{L}_\kappa = \{\ell(f(\boldsymbol{x}), y) \mid f \in H_\kappa\}$ as

$$\widehat{\mathcal{R}}(\mathcal{L}_\kappa) = \frac{1}{n}\,\mathbb{E}_\xi \left[\sup_{f \in H_\kappa} \sum_{i=1}^n \xi_i \ell(f(\boldsymbol{x}_i), y_i)\right],$$

where ξ_i are Rademacher variables uniformly distributed over $\{\pm 1\}$.

Proposition 1 *(Uniform deviation, Lemma A.5 of [31]). For the hypothesis space, it holds*

$$\max\left(\mathbb{E}\sup_{f\in H_\kappa}(\mathcal{E}(f) - \widehat{\mathcal{E}}(f)), \mathbb{E}\sup_{f\in H_\kappa}(\widehat{\mathcal{E}}(f) - \mathcal{E}(f))\right) \le 2\mathbb{E}\,\widehat{\mathcal{R}}(\mathcal{L}_\kappa).$$

Proposition 2 *(Lemma A.2 of [57]). According to McDiarmid's inequality, we can state that, for any $\delta \in (0, 1)$ with probability $1 - \delta$:*

$$\mathbb{E}\,\widehat{\mathcal{R}}(\mathcal{L}_\kappa) \le \widehat{\mathcal{R}}(\mathcal{L}_\kappa) + \sqrt{\frac{2\log(1/\delta)}{n}},$$

$$\sup_{f\in H_\kappa}\left[\widehat{\mathcal{E}}(f) - \mathcal{E}(f)\right] \le \mathbb{E}\sup_{f\in H_\kappa}\left[\widehat{\mathcal{E}}(f) - \mathcal{E}(f)\right] + \sqrt{\frac{2\log(1/\delta)}{n}}.$$

Proposition 3 *(Lemma 5 of [58]). Assume the loss function ℓ is L-Lipschitz for $\mathbb{R}^K$ equipped with the 2-norm. Then, the following inequality holds*

$$\widehat{\mathcal{R}}(\mathcal{L}_\kappa) \le \sqrt{2}L\widehat{\mathcal{R}}(H_\kappa).$$

Proof of Theorem 1. Since $\widehat{f}_n = \arg\min_{f\in H_\kappa}\widehat{\mathcal{E}}(f)$, the excess risk bound can be decomposed as follows

$$\begin{aligned}
\mathcal{E}(\widehat{f}_n) - \mathcal{E}(f^*) &= \mathcal{E}(\widehat{f}_n) - \widehat{\mathcal{E}}(\widehat{f}_n) + \widehat{\mathcal{E}}(\widehat{f}_n) - \widehat{\mathcal{E}}(f^*) + \widehat{\mathcal{E}}(f^*) - \mathcal{E}(f^*) \\
&\le \mathcal{E}(\widehat{f}_n) - \widehat{\mathcal{E}}(\widehat{f}_n) + \widehat{\mathcal{E}}(f^*) - \mathcal{E}(f^*) \\
&\le 2\sup_{f\in H_\kappa}\left[\widehat{\mathcal{E}}(f) - \mathcal{E}(f)\right].
\end{aligned}$$

Using Proposition 2 and Proposition 1, we have

$$\begin{aligned}
\mathcal{E}(\widehat{f}_n) - \mathcal{E}(f^*) &\le 2\,\mathbb{E}\sup_{f\in H_\kappa}\left[\widehat{\mathcal{E}}(f) - \mathcal{E}(f)\right] + 2\sqrt{\frac{2\log(1/\delta)}{n}} \\
&\le 4\,\mathbb{E}\,\widehat{\mathcal{R}}(\mathcal{L}_\kappa) + 2\sqrt{\frac{2\log(1/\delta)}{n}} \\
&\le 4\,\widehat{\mathcal{R}}(\mathcal{L}_\kappa) + 6\sqrt{\frac{2\log(1/\delta)}{n}}.
\end{aligned}$$

Then, with a probability at least $1 - \delta$, by Proposition 3 there holds

$$\mathcal{E}(\widehat{f}_n) - \mathcal{E}(f^*) \le 4\sqrt{2}L\widehat{\mathcal{R}}(H_\kappa) + \mathcal{O}\left(\sqrt{\frac{\log 1/\delta}{n}}\right). \tag{A.1}$$

We estimate empirical Rademacher complexity via

$$\begin{aligned}\widehat{\mathcal{R}}(H_\kappa) &= \frac{1}{n}\,\mathbb{E}_\xi\left[\sup_{f\in H_\kappa}\sum_{i=1}^{n}\sum_{k=1}^{K}\xi_{ik}[f(\boldsymbol{x}_i)]_k\right] \\ &= \frac{1}{n}\,\mathbb{E}_\xi\left[\sup_{f\in H_\kappa}\langle \boldsymbol{W},\boldsymbol{\Phi}_\xi\rangle\right],\end{aligned} \tag{A.2}$$

where $\boldsymbol{W},\boldsymbol{\Phi}_\xi \in \mathcal{H}\times\mathbb{R}^K$ and $\langle \boldsymbol{W},\boldsymbol{\Phi}_\xi\rangle = \mathrm{Tr}(\boldsymbol{W}^\top\boldsymbol{\Phi}_\xi)$ and the matrix $\boldsymbol{\Phi}_\xi$ is defined as follows:

$$\boldsymbol{\Phi}_\xi := \left[\sum_{i=1}^{n}\xi_{i1}\phi(\boldsymbol{x}_i), \sum_{i=1}^{n}\xi_{i2}\phi(\boldsymbol{x}_i), \cdots, \sum_{i=1}^{n}\xi_{iK}\phi(\boldsymbol{x}_i)\right].$$

Applying Hölder's inequality and $\|\boldsymbol{W}\|_*$ bounded by a constant B to (A.2), there holds

$$\begin{aligned}\widehat{\mathcal{R}}(H_\kappa) &= \frac{1}{n}\,\mathbb{E}_\xi\left[\sup_{f\in H_\kappa}\langle \boldsymbol{W},\boldsymbol{\Phi}_\xi\rangle\right] \\ &\le \frac{1}{n}\,\mathbb{E}_\xi\left[\sup_{f\in H_\kappa}\|\boldsymbol{W}\|_*\|\boldsymbol{\Phi}_\xi\|_F\right] \le \frac{B}{n}\,\mathbb{E}_\xi\left[\|\boldsymbol{\Phi}_\xi\|_F\right] \\ &\le \frac{B}{n}\,\mathbb{E}_\xi\left[\sqrt{\|\boldsymbol{\Phi}_\xi\|_F^2}\right] \le \frac{B}{n}\sqrt{\mathbb{E}_\xi\,\|\boldsymbol{\Phi}_\xi\|_F^2}.\end{aligned}$$

Then, we bound $\mathbb{E}_\xi\,\|\boldsymbol{\Phi}_\xi\|_F^2$ as follows

$$\begin{aligned}\mathbb{E}_\xi\,\|\boldsymbol{\Phi}_\xi\|_F^2 &\le \mathbb{E}_\xi\sum_{k=1}^{K}\left\|\sum_{i=1}^{n}\xi_{ik}\phi(\boldsymbol{x}_i)\right\|_2^2 \\ &\le \sum_{k=1}^{K}\mathbb{E}_\xi\left\|\sum_{i=1}^{n}\xi_{ik}\phi(\boldsymbol{x}_i)\right\|_2^2 \\ &\le \sum_{k=1}^{K}\mathbb{E}_\xi\sum_{i,k=1}^{n}\xi_{ik}\xi_{jk}\big[\langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\rangle\big] \\ &= K\sum_{i=1}^{n}\langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_i)\rangle.\end{aligned}$$

The last step is due to the symmetry of the kernel $\kappa(\boldsymbol{x},\boldsymbol{x}') = \langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_i)\rangle$. We finally bound the empirical Rademacher complexity

$$\widehat{\mathcal{R}}(H_\kappa) \le \frac{B}{n}\sqrt{K\sum_{i=1}^{n}\langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_i)\rangle} \tag{A.3}$$

where $B = \sup_{f\in H_\kappa}\|\boldsymbol{W}\|_*$. Substituting the above in equation (A.3) to (A.1), we complete the proof. □

Suppose the sample is drawn from a norm distribution $X \sim N(\mu,\Sigma)$, then its characteristic function holds

$$\mathbb{E}[e^{it^\top X}] = \exp\left(i\mu^\top t - \frac{1}{2}t^\top\Sigma t\right) \tag{A.4}$$

We then estimate the empirical Rademacher complexity for single-layer non-stationary spectral kernels.

Proof of Theorem 2. From the non-stationary spectral kernels (6), the diagonals admit

$$\begin{aligned}\kappa(\boldsymbol{x},\boldsymbol{x}) &= \frac{1}{2} + \frac{1}{2}\int_{\mathcal{X}\times\mathcal{X}} e^{i(\boldsymbol{\omega}-\boldsymbol{\omega}')^\top\boldsymbol{x}}\mu(d\boldsymbol{\omega},d\boldsymbol{\omega}') \\ &= \frac{1}{2} + \frac{1}{2}\,\mathbb{E}_{\boldsymbol{\omega},\boldsymbol{\omega}'}\left[e^{i(\boldsymbol{\omega}-\boldsymbol{\omega}')^\top\boldsymbol{x}}\right].\end{aligned} \tag{A.5}$$

Since two multivariate normal random variables $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma'^2\mathbf{I})$ are independent, we can obtain

$$\boldsymbol{\omega} - \boldsymbol{\omega}' \sim \mathcal{N}\Big(\mathbf{0}, (\sigma^2 + \sigma'^2)\mathbf{I}\Big).$$

Using the characteristic function (A.4), we have

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\omega},\boldsymbol{\omega}'}\left[e^{i(\boldsymbol{\omega}-\boldsymbol{\omega}')^\top \boldsymbol{x}}\right] \\
&= \exp\left\{i\,\mathbf{0}^\top \boldsymbol{x} - \frac{1}{2}(\sigma^2 + \sigma'^2)\boldsymbol{x}^\top \boldsymbol{x}\right\} \\
&= \exp\left\{-\frac{1}{2}(\sigma^2 + \sigma'^2)\boldsymbol{x}^\top \boldsymbol{x}\right\}.
\end{aligned} \tag{A.6}$$

Combining (A.5) and (A.6), we can compute the trace of non-stationary spectral kernel matrix

$$\sum_{i=1}^{n} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) = \sum_{i=1}^{n} \frac{1}{2}\left[1 + \exp\left\{-\frac{1}{2}(\sigma^2 + \sigma'^2)\boldsymbol{x}_i^\top \boldsymbol{x}_i\right\}\right].$$

Substituting the above equation to (11), we finish the proof. □

Finally, we derive the upper bounds for deep non-stationary spectral kernels that illustrate the superiority of deeper architectures.

Proof of Theorem 3. Similar to the proof of Theorem 2, by initializing $\boldsymbol{\omega}_l \sim \mathcal{N}(\mathbf{0}, \sigma_l^2\mathbf{I})$ and $\boldsymbol{\omega}_l' \sim \mathcal{N}(\mathbf{0}, \sigma_l'^2\mathbf{I})$, we obtain the distribution of two normally distributed variables $\boldsymbol{\omega}_l$ and $\boldsymbol{\omega}_l'$, such that

$$\boldsymbol{\omega}_l - \boldsymbol{\omega}_l' \sim \mathcal{N}\Big(\mathbf{0}, (\sigma_l^2 + \sigma_l'^2)\mathbf{I}\Big).$$

Therefore, using the characteristic function (A.4), we estimate the spectral kernel for the l-th layer for any $\boldsymbol{x} \in \mathcal{X}$

$$\begin{aligned}
\kappa_l(\boldsymbol{x}, \boldsymbol{x}) &= \frac{1}{2} + \frac{1}{2}\int_{\mathcal{X}\times\mathcal{X}} e^{i(\boldsymbol{\omega}_l - \boldsymbol{\omega}_l')^\top \phi_{l-1}(\boldsymbol{x})}\mu(d\boldsymbol{\omega}_l, d\boldsymbol{\omega}_l') \\
&= \frac{1}{2} + \frac{1}{2}\exp\left\{-\frac{1}{2}(\sigma_l^2 + \sigma_l'^2)\phi_{l-1}(\boldsymbol{x})^\top \phi_{l-1}(\boldsymbol{x})\right\} \\
&= \frac{1}{2} + \frac{1}{2}\exp\left\{-\frac{1}{2}(\sigma_l^2 + \sigma_l'^2)\kappa_{l-1}(\boldsymbol{x}, \boldsymbol{x})\right\}.
\end{aligned} \tag{A.7}$$

By setting the variances of independent Gaussian distributions as

$$(\sigma_l^2 + \sigma_l'^2) \geq -\frac{2\log\left[2\kappa_{l-1}(\boldsymbol{x}, \boldsymbol{x}) - 1\right]}{\kappa_{l-1}(\boldsymbol{x}, \boldsymbol{x})},$$

we have

$$\kappa_l(\boldsymbol{x}, \boldsymbol{x}) = \frac{1}{2} + \frac{1}{2}\exp\left\{-\frac{1}{2}(\sigma_l^2 + \sigma_l'^2)\kappa_{l-1}(\boldsymbol{x}, \boldsymbol{x})\right\} \leq \kappa_{l-1}(\boldsymbol{x}, \boldsymbol{x}).$$

Therefore, we complete the proof. □

References

[1] Y. Bengio, O. Delalleau, N.L. Roux, The curse of highly variable functions for local kernel machines, in: Advances in Neural Information Processing Systems (NeurIPS), 2006, pp. 107–114.
[2] C. Cortes, M. Mohri, A. Rostamizadeh, Two-stage learning kernel algorithms, in: 27th International Conference on Machine Learning, ICML 2010, 2010, pp. 239–246.
[3] G.C. Cawley, Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2006, pp. 1661–1668.
[4] R.M. Neal, Bayesian learning for neural networks, Ph.D. thesis, Citeseer, 1995.
[5] J. Lee, Y. Bahri, R. Novak, S.S. Schoenholz, J. Pennington, J. Sohl-Dickstein, Deep neural networks as Gaussian processes, in: International Conference on Learning Representations, 2018.
[6] S. Remes, M. Heinonen, S. Kaski, Non-stationary spectral kernels, in: Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, pp. 4642–4651.
[7] L. Ding, S. Liao, Y. Liu, L. Liu, F. Zhu, Y. Yao, L. Shao, X. Gao, Approximate kernel selection via matrix approximation, IEEE Trans. Neural Netw. Learn. Syst. (2020).

[8] J. Mairal, P. Koniusz, Z. Harchaoui, C. Schmid, Convolutional kernel networks, in: Advances in Neural Information Processing Systems (NeurIPS), 2014, pp. 2627–2635.
[9] J. Xie, F. Liu, K. Wang, X. Huang, Deep kernel learning via random Fourier features, arXiv preprint, arXiv:1910.02660, 2019.
[10] J. Li, Y. Liu, W. Wang, Automated spectral kernel learning, in: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI), 2020.
[11] Y.-L.K. Samo, S. Roberts, Generalized spectral kernels, arXiv preprint, arXiv:1506.02236, 2015.
[12] S. Sun, G. Zhang, J. Shi, R. Grosse, Functional variational Bayesian neural networks, in: International Conference on Learning Representations, 2019.
[13] H. Xue, Z.-F. Wu, W.-X. Sun, Deep spectral kernel learning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), 2019, pp. 4019–4025.
[14] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[15] S. Sonnenburg, G. Rätsch, C. Schäfer, B. Schölkopf, Large scale multiple kernel learning, J. Mach. Learn. Res. 7 (2006) 1531–1565.
[16] K. Fang, X. Huang, F. Liu, J. Yang, End-to-end kernel learning via generative random Fourier features, arXiv preprint, arXiv:2009.04614, 2020.
[17] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.
[18] J. Mairal, End-to-end kernel learning with supervised convolutional kernel networks, in: Advances in Neural Information Processing Systems (NeurIPS), 2016, pp. 1399–1407.
[19] A. Bietti, J. Mairal, Group invariance, stability to deformations, and complexity of deep convolutional representations, J. Mach. Learn. Res. 20 (1) (2019) 876–924.
[20] A.G. Wilson, Z. Hu, R. Salakhutdinov, E.P. Xing, Deep kernel learning, in: Artificial Intelligence and Statistics, PMLR, 2016, pp. 370–378.
[21] A.G. Wilson, Z. Hu, R.R. Salakhutdinov, E.P. Xing, Stochastic variational deep kernel learning, Adv. Neural Inf. Process. Syst. 29 (2016).
[22] D.-X. Zhou, Universality of deep convolutional neural networks, Appl. Comput. Harmon. Anal. 48 (2) (2020) 787–794.
[23] Z. Shen, M. Heinonen, S. Kaski, Learning spectrograms with convolutional spectral kernels, arXiv preprint, arXiv:1905.09917, 2019.
[24] M.L. Stein, Interpolation of Spatial Data: Some Theory for Kriging, Springer Science & Business Media, 1999.
[25] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems (NeurIPS), vol. 21, 2007, pp. 1177–1184.
[26] B. Sriperumbudur, Z. Szabó, Optimal rates for random Fourier features, in: Advances in Neural Information Processing Systems (NeurIPS), vol. 28, 2015, pp. 1144–1152.
[27] H. Avron, M. Kapralov, C. Musco, C. Musco, A. Velingker, A. Zandieh, Random Fourier features for kernel ridge regression: approximation bounds and statistical guarantees, in: International Conference on Machine Learning, PMLR, 2017, pp. 253–262.
[28] Z. Liao, R. Couillet, M.W. Mahoney, A random matrix analysis of random Fourier features: beyond the Gaussian kernel, a precise phase transition, and the corresponding double descent, Adv. Neural Inf. Process. Syst. 33 (2020) 13939–13950.
[29] M. Belkin, S. Ma, S. Mandal, To understand deep learning we need to understand kernel learning, arXiv preprint, arXiv:1802.01396, 2018.
[30] V. Koltchinskii, Rademacher penalties and structural risk minimization, IEEE Trans. Inf. Theory 47 (5) (2001) 1902–1914.
[31] P.L. Bartlett, O. Bousquet, S. Mendelson, Local Rademacher complexities, Ann. Stat. 33 (4) (2005) 1497–1537.
[32] P.L. Bartlett, S. Mendelson, Rademacher and Gaussian complexities: risk bounds and structural results, J. Mach. Learn. Res. 3 (Nov 2002) 463–482.
[33] A. Rudi, L. Rosasco, Generalization properties of learning with random features, in: Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, pp. 3215–3225.
[34] F. Bach, On the equivalence between kernel quadrature rules and random feature expansions, J. Mach. Learn. Res. 18 (21) (2017) 1–38.
[35] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, S. Ganguli, Exponential expressivity in deep neural networks through transient chaos, in: Advances in Neural Information Processing Systems (NeurIPS), vol. 29, 2016, pp. 3360–3368.
[36] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems (NeurIPS), 2012, pp. 1097–1105.
[37] S. Mallat, Group invariant scattering, Commun. Pure Appl. Math. 65 (10) (2012) 1331–1398.
[38] T. Wiatowski, H. Bölcskei, A mathematical theory of deep convolutional neural networks for feature extraction, IEEE Trans. Inf. Theory 64 (3) (2017) 1845–1866.
[39] S. Zhang, J. Li, P. Xie, Y. Zhang, M. Shao, H. Zhou, M. Yan, Stacked kernel network, arXiv preprint, arXiv:1711.09219, 2017.
[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: an imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems (NeurIPS), 2019, pp. 8024–8035.
[41] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[42] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556, 2014.
[43] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[44] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[45] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, ShuffleNet V2: practical guidelines for efficient CNN architecture design, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 116–131.
[46] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, Adv. Neural Inf. Process. Syst. 33 (2020) 7462–7473.
[47] B. Ghorbani, S. Mei, T. Misiakiewicz, A. Montanari, When do neural networks outperform kernel methods?, Adv. Neural Inf. Process. Syst. 33 (2020) 14820–14830.
[48] M. Refinetti, S. Goldt, F. Krzakala, L. Zdeborová, Classifying high-dimensional Gaussian mixtures: where kernel methods fail and neural networks succeed, in: International Conference on Machine Learning, PMLR, 2021, pp. 8936–8947.
[49] P.L. Bartlett, D.J. Foster, M.J. Telgarsky, Spectrally-normalized margin bounds for neural networks, in: Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 6240–6249.
[50] A. Bietti, G. Mialon, D. Chen, J. Mairal, A kernel perspective for regularizing deep neural networks, in: International Conference on Machine Learning (ICML), 2019, pp. 664–674.
[51] Z. Allen-Zhu, Y. Li, Y. Liang, Learning and generalization in overparameterized neural networks, going beyond two layers, in: Advances in Neural Information Processing Systems (NeurIPS), 2019, pp. 6155–6166.
[52] S. Arora, S. Du, W. Hu, Z. Li, R. Wang, Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks, in: International Conference on Machine Learning (ICML), 2019, pp. 322–332.
[53] S.S. Schoenholz, J. Gilmer, S. Ganguli, J. Sohl-Dickstein, Deep information propagation, in: International Conference on Learning Representations, 2017.
[54] D.-X. Zhou, Theory of deep convolutional neural networks: downsampling, Neural Netw. (2020).
[55] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, J. Pennington, Dynamical isometry and a mean field theory of CNNs: how to train 10,000-layer vanilla convolutional neural networks, in: Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 5393–5402.
[56] A. Jacot, F. Gabriel, C. Hongler, Neural tangent kernel: convergence and generalization in neural networks, in: Advances in Neural Information Processing Systems (NeurIPS), 2018, pp. 8571–8580.

[57] L. Oneto, A. Ghio, S. Ridella, D. Anguita, Local Rademacher complexity: sharper risk bounds with and without unlabeled samples, Neural Netw. 65 (2015) 115–125.
[58] C. Cortes, V. Kuznetsov, M. Mohri, S. Yang, Structured prediction theory based on factor graph complexity, in: Advances in Neural Information Processing Systems (NeurIPS), vol. 29, 2016, pp. 2514–2522.

Sponsor names

Do not correct this page. Please mark corrections to sponsor names and grant numbers in the main text.

CAS, *country*=China, *grants*=E0YY231114

National Natural Science Foundation of China, *country*=China, *grants*=62076234, 62106257