Butterfly-Net: Optimal Function Representation Based on Convolutional Neural Networks

Yingzhou Li
Department of Mathematics
Duke University
Durham, NC 27708
yingzhou.li@duke.edu

Xiuyuan Cheng
Department of Mathematics
Duke University
Durham, NC 27708
xiuyuan@math.duke.edu

Jianfeng Lu
Department of Mathematics
Department of Physics and Department of Chemistry
Duke University
Durham, NC 27708
jianfeng@math.duke.edu

Abstract

Deep networks, especially Convolutional Neural Networks (CNNs), have been successfully applied in various areas of machine learning as well as to challenging problems in other scientific and engineering fields. This paper introduces *Butterfly-Net*, a low-complexity CNN with structured hard-coded weights and sparse across-channel connections, which aims at an optimal hierarchical function representation of the input signal. Theoretical analysis of the approximation power of *Butterfly-Net* to the Fourier representation of input data shows that the error decays exponentially as the depth increases. Due to the ability of *Butterfly-Net* to approximate Fourier and local Fourier transforms, the result can be used for approximation upper bound for CNNs in a large class of problems. The analysis results are validated in numerical experiments on the approximation of a 1D Fourier kernel and of solving a 2D Poisson's equation.

1 Introduction

Deep network is a central tool in machine learning and data analysis nowadays [3]. In particular, Convolutional Neural Network (CNN) has been proved to be a powerful tool in image recognition and representation. Deep learning has also emerged to be successfully applied in solving PDEs [4, 19, 26] and physics problems [2, 15, 34, 38], showing the potential of becoming a tool of great use for computational mathematics and physics as well. Given the wide application of PDE and wavelet based methods in image and signal processing [6, 9, 27], an understanding of CNN's ability to approximate differential and integral operators will lead to an explanation of CNN's success in these fields, as well as possible improved network architectures.

The remarkable performance of deep networks across various fields relies on their ability to accurately represent functions of high-dimensional input data. Approximation analysis has been a central topic to the understanding of the neural networks. The classical theory developed in 80's and early 90's [1, 11, 18] approximates a target function by a linear combination of sigmoids, which is equivalent to a fully connected neural network with one hidden layer. While universal approximation theorems were established for such shallow networks, the research interest in neural networks only

Preprint. Work in progress.

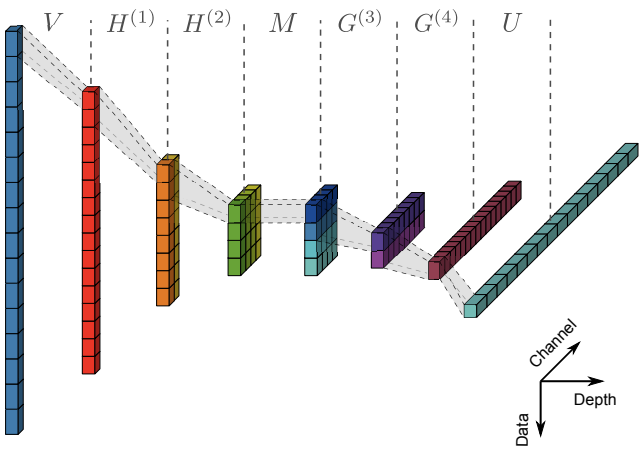


Figure 1: *Butterfly-Net* in 1D. Layer 1 and 7 are 1D convolutional layers. Layer 2, 3, 5 and 6 are 1D convolutional layers with 2 non-mixing channels for each input channel. Layer 4 is a local mixing layer, which only involves local operations. $V, H^{(\ell)}, M, G^{(\ell)}, U$ indicate the matrix representation of the convolutional layers, c.f. Supp. A. The name "butterfly" can be understood from the shape of the network.

revived in 00's after observing the successful applications of deep networks, particular the superior performance of CNNs in image and audio signal processing.

Motivated by the empirical success, the approximation advantage of deep networks over shallow ones has been theoretically analyzed in several places. However, most results assume stacked fully connected layers and do not apply to CNN which has specific geometrical constraints: (1) the convolutional scheme, namely local-supported filters with weight sharing, and (2) the hierarchical multi-scale architecture. The approximation power of deep networks with hierarchical geometrically-constrained structure has been studied recently [10, 28, 29], yet the network architecture differ from the standard CNN. We review the related literature more below.

The current paper proposes a specific architecture of convolutional neural networks based on the *Butterfly* scheme originally developed for the fast computation of special function transforms [30, 32, 37] and Fourier integral operators [7, 8, 22–24]. *Butterfly* algorithm provides a hierarchical structure with locally low-rank interpolation of kernel functions, and can be applied to solve many PDE problems. In terms of computational complexity, the scheme is near optimal for Fourier kernels and a large class of Fourier integral operators. The proposed *Butterfly-Net* explicitly hard-codes the stacked convolutional layers, which collectively computes the Fourier coefficients of the input signal with guaranteed numerical accuracy. Unlike regular CNN which is fully connected across the unordered channels, the channels in the *Butterfly-Net* has a clear correspondence with the frequency band, namely the position in the spectral representation of the signal, and meanwhile, the inter-channel weights are sparsely connected. *Butterfly-Net* is much lighter than the usual CNN: the model complexity (in terms of number of parameters in the parametrization) is $O(N)$ and computational complexity is $O(N \log N)$, where N is the length of the discrete input signal. The approximation error of *Butterfly-Net* is proved to exponentially decay as the network depth increases, also is numerically validated in Sec. 4. Due to the efficient approximation of Fourier kernels, *Butterfly-Net* thus possesses all approximation properties of the Fourier representation of input signals, which is particularly useful for solving PDEs and (local) Fourier-based algorithms in image and signal processing. Our theoretical result provides an approximation upper bound of the CNNs which are trained in practice, an exemplar case of which is shown in Sec. 4.

The contribution of our work is summarized as follows: A low-complexity CNN, motivated by the *Butterfly* algorithm, is proposed with the following properties.

1) The network has structured hard-coded weights and sparse across-channel connections. The channel indexing is in order of the frequency band, and the intermediate representations in the network have the interpretation of local Fourier transforms.

2) The networks gives a near-optimal hierarchical representation of the Fourier kernel, with model complexity $O(N)$ and computational complexity $O(N \log N)$.

3) The approximation accuracy to the Fourier Kernel is theoretically proved to be exponentially decay as the depth increases. Combined with fully-connected layers, the approximation analysis of *Butterfly-Net* provides an approximation upper bound of CNNs in a large class of problems in PDEs and image and signal processing.

Before we explain all these in more detail in the rest of the paper, we review some related works.

Classical approximation results of neural networks. Universal approximation theorems for fully-connected neural networks with one hidden layer were established in [11, 18] showing that such network can approximate a target function with arbitrary accuracy if the hidden layer is allowed to be wide enough. In theory, the family of target functions can include all measurable functions [18], when exponentially many hidden neurons are used. Gallant and White [17] proposed "Fourier network", proving universal approximation to squared-integrable functions by firstly constructing a Fourier series approximation of the target function in a hard-coded way. These theorems are firstly proved for one-dimensional input, and when generalizing to the multivariate case the complexity grows exponentially.

Using the Fourier representation of the target function supported on a sphere in $\mathbb{R}^d$, Barron [1] showed that the mean squared error of the approximation, integrated with arbitrary data distribution on the sphere, decays as $O(n^{-1})$ when n hidden nodes are used in the single hidden layer. The results for shallow networks are limited, and the approximation power of depth in neural networks has been advocated in several recent works, see below. Besides, while the connection to Fourier analysis was leveraged, at least in [17] and [1], it is different from the hierarchical function representation scheme as what we consider here.

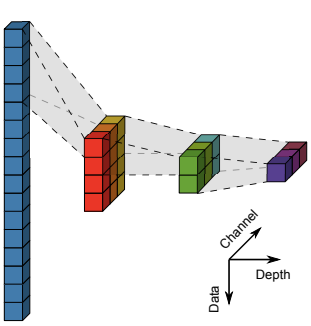
Approximation power of deep networks. The expressive power of deep networks has drawn many research interests after 00's. The approximation power of multi-layer Restricted Boltzmann Machines (RBM) was studied in [21], which showed that RBMs are universal approximators of discrete distributions and more hidden layers improves the approximation power. Relating to the classical approximation results in harmonic analysis, Bölcskei et al. [5] derived lower bounds for the uniform approximation of square-integrable functions, and proved the asymptotic optimality of the sparsely connected deep networks as a universal approximator. However, the network complexity also grows exponentially when the input dimension increases.

The advantage of deep architecture over shallow ones has been studied in several works. Delalleau and Bengio [12] identified a deep sum-product network which can only be approximated by an exponentially larger number of shallow ones. The exponential growth of linear regions as the number of layers increases was studied in [31, 35]. Eldan and Shamir [16] constructed a concrete target function which distinguishes three and two-layer networks. Liang and Srikant [25] showed that shallow networks require exponentially more neurons than deep networks to obtain a given approximation error for a large class of functions. The advantage of deep ReLU networks over the standard single-layer ones was analyzed in [36] in the context of approximation in Sobolev spaces. The above works address deep networks with fully-connected layers, instead of having geometrically-constrained constructions like CNNs.

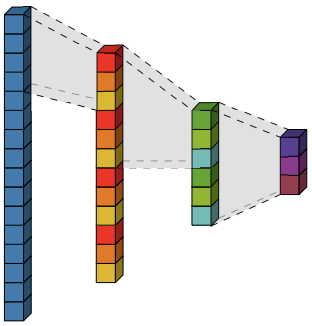
Deep networks with these constraints are relatively less analyzed. The approximation power of a hierarchical binary tree network was studied in [28, 29] which supports the potential advantage of deep CNNs. Cohen et al. [10] used convolutional arithmetic circuits to show the equivalence between a deep network and a hierarchical Tucker decomposition of tensors, and proved the advantage of depth in function approximation. The networks being studied differ from the regular CNNs widely used in the typical real world applications.

2 *Butterfly-Net*

Butterfly-Net is a novel structured CNN which requires far less number of parameters to represent functions that can be expressed in the frequency domain. The essential building block of *Butterfly-Net* is the interpolation convolutional layer, illustrated in Fig. 2 and described in Sec. 2.1, which by itself is also interesting as it gives another way of interpreting channel mixing. Sec. 2.2 assembles interpolation convolutional layers together and form the *Butterfly-Net*. For simplicity of the notation and indexing, we will limit the discussion to 1D signals, while the extension to 2D and higher dimensional data is straightforward by using a tensor product construction.



(a) 1D interpolation convolutional layers with folded channel representation



(b) 1D interpolation convolutional layers with unfolded channel representation

Figure 2: 1D interpolation convolutional layers with input size 16 and channel size 1. The first layer is a 1D conv with filter size and stride size being 4 and contains 3 output channels. All later layers are 1D convs with filter size and stride size being 2 and the input and output channels sizes are 3.

2.1 Interpolation convolutional layer

To introduce the interpolation convolutional operation, we first introduce an equivalent formula of the usual convolutional layer by "channel unfolding", and then illustrate the layer through a hierarchical interpolation example.

Let $\{f(i,k_1) \mid i = 1,2,\ldots,n; k_1 = 1,2,\ldots,c_1\}$ be the input data with length n and c_1 channels. Assume the 1D convolutional layer maps c_1 input channels to c_2 output channels and the convolution filter is of size w, hence the parameters can be written as W_{k_2,i,k_1} where $i = 1,\ldots,w$, $k_1 = 1,\ldots,c_1$, and $k_2 = 1,\ldots,c_2$. The output data, under these notations, is written as

$$g(j,k_2) = \sum_{1\le i\le w,\, 1\le k_1\le c_1} W_{k_2,i,k_1} f(i+s(j-1),k_1), \tag{1}$$

where $s \ge 1$ is the stride size. In many problems, it is more convenient to unfold the channel index into the data length, i.e., $f[(i-1)c_1+k_1] = f(i,k_1)$, $g[(j-1)c_2+k_2] = g(j,k_2)$, and $W[k_2,(i-1)w+k_1] = W_{k_2,i-s(j-1),k_1}$. Hence one site of g with all channels can be represented as the matrix vector product,

$$g[(j-1)c_2+(1:c_2)] = W[1:c_2, 1:wc_1] f[sc_1(j-1)+(1:wc_1)]. \tag{2}$$

Without considering the weight sharing in the convolution layer, all channel direction can be unfolded into the data dimension and the convolution is modified as a block convolution. Such an unfolded convolutional layer will be called the interpolation convolutional layer.

The representation of interpolation convolutional layer is motivated by the observation that function interpolation (source transfer) can be naturally represented as multi-channel convolution. In this setting, unfolding channels is natural. Let $B_1, B_2, \ldots, B_J$ be a uniform partition of $[0,1)$, i.e., $B_i = [{}^{(i-1)}/_J, {}^{i}/_J)$, and $x_{k_1}^{B_j}$ denote the k_1-th discretization point in B_j for $k_1 = 1,\ldots,c_1$. We further assume that the locations of $x_{k_1}^{B_j}$ relative to B_j are the same for all j.

The input data will be viewed as the function value of $f(x)$ evaluated at the points $x_{k_1}^{B_j}$, i.e., $f(j,k_1) = f(x_{k_1}^{B_j})$. Let $z_{k_2}^{B_j}$ be the interpolation points on B_j for $k_2 = 1,\ldots,c_2$, with the Lagrange basis polynomial given by

$$\mathcal{L}_{k_2}(x) = \prod_{p\ne k_2} \frac{x - z_p^{B_j}}{z_{k_2}^{B_j} - z_p^{B_j}}. \tag{3}$$

The equivalent source of $f(x)$ at $z_{k_2}^{B_j}$ is then defined as,

$$g(z_{k_2}^{B_j}) = g(j,k_2) = \sum_{k_1=1}^{c_1} \mathcal{L}_{k_2}(x_{k_1}^{B_j}) f(x_{k_1}^{B_j}) = \sum_{k_1=1}^{c_1} \prod_{p\ne k_2} \frac{x_{k_1}^{B_j} - z_p^{B_j}}{z_{k_2}^{B_j} - z_p^{B_j}} f(x_{k_1}^{B_j}). \tag{4}$$

We note that the product of the fractions in (4) depends only on the relative distance and is thus independent of B_j thanks to our assumption on the interpolation points. Therefore, we could

denote $W_{k_2,i,k_1} = \prod_{p\neq k_2} {}^{(x_{k_1}^{B_1} - z_p^{B_1})}/_{(z_{k_2}^{B_1} - z_p^{B_1})}$, and thus the source transfer formula (4) can be interpreted as convolution (1) with the stride size being the same as the filter size, i.e., $s = w$. In this representation, the two channel indices k_1 and k_2 denote the original points and interpolation points within each interval B_j. Therefore, unfolding the channel index of both f and g leads to the natural ordering of the index of points on $[0, 1)$.

For a CNN with multiple convolutional layers, the unfolding of the channel index could be done recursively. Fig. 2 (a) illustrates 1D interpolation convolutional layers whereas Fig. 2 (b) shows its unfolded representation. Gray zones in both figures indicate instances of the data dependency between layers.

Fig. 2 (b) can also be understood in the view of function interpolation. The domain is first divided into four parts and the first layer interpolates the input function within each part to its three interpolation points. The layer afterwards merges two adjacent parts into one and interpolates the function defined on the previous 6 grid points to the new 3 interpolation points on the merged part.

2.2 *Butterfly-Net* Architecture

Let $f(t)$ be the input data viewed as signal in time (extension to higher dimension signals is carried out through tensor product construction). Time-frequency analysis usually splits the signal into different modes according to frequency range, e.g., high-, medium-, low-frequency modes. Most importantly, once the signal is decomposed into different modes, they will be analyzed separately and will not be mixed again. This motivates us to propose non-mixing channels in our *Butterfly-Net*, since the channel has a correspondence with frequency modes in our setting. Assume that the input vector is of length N and the output is a feature vector of length K. Let L denote the number of major layers in the *Butterfly-Net*, r denote the size mixing channels. Without loss of generality, we assume that L is an even integer such that $L \leq \log N$.

We propose the *Butterfly-Net* architecture as follows (see Fig. 1). The butterfly shape of the network, which is motivated by the hierarchical structure of the *Butterfly* scheme, results from the complementary low-rank structure of time-frequency analysis. It plays an essential role in the approximation power of the *Butterfly-Net*, as will be shown in the theoretical analysis.

- *Preparation Layer.* Assume N is multiple of 2^L, i.e., $N = m2^L$ and the input vector is $f = [f_1 \quad \cdots f_N]$. A 1D convolution layer with filter size m, stride m and output channel r is added with ReLU activation. The output vector of the preparation layer is denoted as $f^{(0)}(j, k, 1)$, where $j = 1, \ldots, 2^L$ is the index of data and $k = 1, \ldots, r$ is the index of channel. Both second and third indices correspond to channels, the former denotes the mixing channel whereas the later denotes the non-mixing channel (with a single channel for the preparation layer).
- *Layer* $\ell = 1, \ldots, {}^L/_2$. The input vector of Layer ℓ is $f^{(\ell-1)}(j, k, i)$ for $j = 1, \ldots, 2^{L-\ell+1}$, $k = 1, \ldots, r$, and $i = 1, \ldots, 2^{\ell-1}$. For each of the non-mixing channel i, two 1D convolution layer with filter size 2, stride 2 and output channel r is added with ReLU activation. The output vector for each i is then denoted as $f^{(\ell)}(j, k, 2i-1)$ and $f^{(\ell)}(j, k, 2i)$. Hence, the output vector of Layer ℓ is denoted as $f^{(\ell)}(j, k, i)$ where $j = 1, \ldots, 2^{L-\ell}$ is the index of data, $k = 1, \ldots, r$ is the index of mixing channel, and $i = 1, \ldots, 2^\ell$ is the index of non-mixing channel. The output vector matches the input vector at next layer.
- *Layer M.* The middle layer, Layer M, is a special layer of local operations. Denote the input and output vector of Layer M is $f^{({}^L/_2)}(j, k, i)$ and $g^{({}^L/_2)}(j, k, i)$ resp. At this level, for each i and j, $g^{({}^L/_2)}(j, i, :) = W_{j,i}^{(M)} f^{({}^L/_2)}(j, :, i)$, where $W_{j,i}^{(M)}$ is a matrix of size $r \times r$.
- *Layer* $\ell = {}^L/_2 + 1, \ldots, L$. The input vector of Layer ℓ is $g^{(\ell-1)}(j, i, k)$ for $j = 1, \ldots, 2^{L-\ell+1}$, $i = 1, \ldots, 2^{\ell-1}$, and $k = 1, \ldots, r$. For each of the non-mixing channel i, two 1D convolution layers with filter size 2, stride 2 and output channel r is added and then the ReLU activations are applied. The output vector for each i is then denoted as $g^{(\ell)}(j, 2i-1, k)$ and $g^{(\ell)}(j, 2i, k)$. Hence, the output vector of Layer ℓ is denoted as $g^{(\ell)}(j, i, k)$ where $j = 1, \ldots, 2^{L-\ell}$ is the index of data, $i = 1, \ldots, 2^\ell$ is the index of non-mixing channel, and $k = 1, \ldots, r$ is the index of mixing channel.
- *Feature Layer.* The input vector is $g^{(L)}(1, i, k)$ where $i = 1, \ldots, 2^L$ and $k = 1, \ldots, r$. A 1D convolution layer interpolates the g^L back into output vector of length K.

The output of the feature layers can be used as efficient low-dimensional representation of the input function; the approximation power of the *Butterfly-Net* and its applications will be discussed in Sec. 3.

Fig. 1 demonstrates an example of the *Butterfly-Net* with input vector being partitioned into 16 parts. We adopts the unfolded representation of the mixing channel as in Fig. 2 (b), and the channel direction only contains non-mixing channels. A matrix representation of the *Butterfly-Net* is given in Supp. A to facilitate the theoretical analysis of approximation power.

The main advantage of the proposed *Butterfly-Net* is the reduction of model size and computational complexity. As can be seen from a simple calculation, the overall number of parameters involved is $O(r^2N)$, where N is the size of input, and r is the size of mixing channels. Such a parametrization is near optimal for many well-known kernels, for instance, discrete Fourier kernel, discrete smooth Fourier integral operator, etc., since it gives almost linear scaling algorithms up to logarithmic factors. And the overall computational cost for a evaluation of the *Butterfly-Net* is $O(N \log N)$.

The extension of the *Butterfly-Net* to d dimensional input signals, $f(t)$ for $t \in \mathbb{R}^d$, is straightforward. Under the same architecture, we can simply replace the index i, j, k by multi dimensional index, e.g., $i \to (i_1, \ldots, i_d)$. This means instead of 1D mixing channels and 1D non-mixing channels, now we have dD mixing channels and dD non-mixing channels. The filter size and stride size should also be adapted to d dimension. The overall number of parameters in this case is $O(r^{2d}N)$, where N is the total size of the d dimensional input. If filters are assumed to maintain tensor product structure, then both the number of parameters and computational cost can be reduced.

3 Analysis of Approximation Power

In this section, we analyze the approximation power of the *Butterfly-Net* on a specific kernel, discrete Fourier kernel, whose matrix entry is defined as $\mathcal{K}_{ij} = e^{-2\pi\imath\xi_i \cdot t_j}$ where t_j and ξ_i are uniformly distributed on $[0, 1)$ and $[-\frac{K}{2}, \frac{K}{2})$ ($K \leq N$) respectively. The analysis result shows that though *Butterfly-Net* by construction has very low complexity as the number of parameters is on the order of the input data size, it exhibits full approximation power in terms of function representations. The generalization to a wider arrange of function representations is possible and will be commented in the sequel.

Theorem 3.1. *Let N denote the size of the input and K denote the length of the domain of the output in the Butterfly-Net. L and r are two parameters such that $\pi e K \leq r2^L$. L is the depth of the Butterfly-Net and r is the size of mixing channels. There exists a parametrized Butterfly-Net, $\mathcal{B}(\cdot)$, approximating the discrete Fourier kernel such that for any bounded input vector f, the error of the output of the Butterfly-Net satisfies that for any $p \in [1, \infty]$*

$$\|\mathcal{K}f - \mathcal{B}(f)\|_p \leq m^{1-\frac{1}{p}} C_{r,K} \left(\frac{\sqrt{r}\left(\frac{2}{\pi}\ln r + 1\right)}{2^{r-2}} \right)^L \|f\|_p, \tag{5}$$

where $m = N/2^L$, and $C_{r,K} = {(2+2/\pi \ln r)^3 (\pi e K)^r}/{(2r)^{r-2}}$ is a constant depending only on r and K.

The proof of Theorem 3.1 is constructive. We first fill the *Butterfly-Net* with a specific set of parameters based on the complementary low-rank property of the discrete Fourier kernel (see Theorem B.1 in Supp. B for a precise statement of the complementary low-rank property). Once the parameters of *Butterfly-Net* are constructed in this way, which means entries in the matrix representation of the *Butterfly-Net* are explicitly known, 1-norm and ∞-norm of each matrix can be bounded. Combined with the low-rank approximation error at different levels, we derive the 1-norm and ∞-norm upper bound for the *Butterfly-Net* matrix representation. Applying Riese-Thorin interpolation theorem, we reach to the conclusion of Theorem 3.1 for general index p.

Previously, in the context of fast algorithms, Kunis and Melzer [20] analyzed the approximations of a simplified *Butterfly* scheme and Demanet et al. [13] analyzed general *Butterfly* scheme under different error measures on the input and output. While as a side product of our proof, we also obtain the error estimate of the matrix approximation of the general *Butterfly* schemes in terms of matrix norms. Supp. B provides the detailed proof of the theorem.

For a problem with fixed input and output size, we can tune two parameters r and L to reach desired accuracy. As r increases, which corresponds to increase the width of each layer, two parts of the

error bound, the explicit power and the constant $C_{r,K}$, both decrease. The base of the power decays exponentially as the increase of r whereas the constant $C_{r,K}$ decays as r^{-r}. Interestingly, when L increases, which corresponds to increase the depth of the *Butterfly-Net*, the error bound decays exponentially in L when $r > 3$ as can be verified by a simple calculation. Hence we conclude that the error of the *Butterfly-Net* decays exponentially in L for any $r > 3$.

Generalization of the approximation result. While Theorem 3.1 only considers the Fourier kernel, the result in it can be much generalized by a careful investigation of the proof. First the domain of the discrete Fourier kernel, interval $[0, 1)$ and $[-K/2, K/2)$, can be scaled and shifted. Theorem 3.1 holds as long as the product of the length of two intervals remain K. Thus local Fourier transforms and wavelets type multi-resolution analysis can be easily accommodated. Further, through a parallel reading of the proof of Theorem 3.1 and [8, 22], we can show that similar theorem can be provided for smooth Fourier integral operators (FIOs), i.e., $e^{-2\pi\imath\Phi(\xi,t)}$ with smooth $\Phi(\xi, t)$ satisfying homogeneity condition of degree 1. In image and signal processing, Laplace operator, which corresponds to discrete Fourier kernel, is often used for diffusion related process. The usage of *Butterfly-Net* further enables processing with elliptic operators, whose symbols can be written as smooth FIOs [14], so that to obtain better performance in image and signal processing. The wide applicability of the *Butterfly-Net* could provide insights of the success of the CNN in image and signal processing.

Butterfly-Net can also be used as a module in a larger network, serving to efficiently approximate a discrete Fourier kernel or Fourier integral operator by a low-complexity CNN sub-network. Many layers can be added before/after *Butterfly-Net*. For example, in order to estimate the energy functional of a Poisson's equation, which can be approximated by a quadratic form of the low-frequency Fourier components of the input function, we can add a fully connected layer on the top of the *Butterfly-Net* to approximate the quadratic form thanks to the power of the universal approximation theorem [1, 11, 18]. In this case, we can claim that the *Butterfly-Net* with a fully connected layer provides more efficient representation than plain fully connected layers. For many other functionals involving the Fourier components, the *Butterfly-Net* with fully connected layer is capable to represent them. The proof is simply a combined usage of Theorem 3.1 and the universal approximation theorem of shallow networks.

4 Numerical Results

We present two numerical results to demonstrate the approximation power of the *Butterfly-Net* and its application to energy functional. The first numerical experiment shows that the approximation error of an initialized *Butterfly-Net* decays exponentially as the increases of the network depth L, which verifies the conclusion of Theorem 3.1. In the second experiment, we construct a CNN with two convolutional layers and a fully connect layer interlacing with ReLU layers, which is a over parametrized version of *Butterfly-Net* plus fully connected layers. The neural network well approximates the energy functional of a 2D Poisson's equation.

***Butterfly-Net* for discrete Fourier transform.** The first numerical result aims to verify the exponential decay of the approximation error of the *Butterfly-Net* as the depth L increases. We construct a *Butterfly-Net* to approximate the discrete Fourier kernel with fixed number of mixing channels, $r = 4$, which is sufficient large for the decay factor in Theorem 3.1 being smaller than one. The *Butterfly-Net* is filled with the parameters in the constructive proof of Theorem 3.1. The input vector in this example is of size $N = 16384$ whereas different sizes of the output vector are tested. The output vector represents integer frequency of the input function in the frequency domain $[-K/2, K/2)$. The approximation error of the *Butterfly-Net* is measured against the dense discrete Fourier kernel matrix, and relative matrix p-norm error is reported, $\epsilon_p = \|\mathcal{K}-\mathcal{B}\|_p / \|\mathcal{K}\|_p$, where $\mathcal{B}$ denotes the *Butterfly-Net* matrix.

Tab. 1 shows for both choices of K, the relative approximation errors measured in matrix 1-norm, 2-norm, and ∞-norm decay exponentially as L increases. The decay factors for different K remain similar, while the prefactor is much larger for large K. All of these observations agree with the error bound in Theorem 3.1.

***Butterfly-Net* for Discrete Laplace Operator.** The second numerical example aims to construct an approximation of the energy functional of 2D Poisson's equation, as explained in the Section 3 about the general function representation power of the *Butterfly-Net*. Here we use a regular CNN which can be viewed as *Butterfly-Net* with all channels mixed. We aims to obtain the energy of Poisson's

Table 1: Approximation accuracy of the Fourier kernel by the *Butterfly-Net*. Mixing channels are of size $r = 4$.

N	$K = 64$				$K = 256$			
	L	ϵ_1	ϵ_2	ϵ_∞	L	ϵ_1	ϵ_2	ϵ_∞
16384	6	5.0e-2	6.8e-2	5.7e-2	8	6.4e-2	8.9e-2	6.6e-2
16384	8	1.9e-4	3.0e-4	2.4e-4	10	2.4e-4	3.8e-4	2.7e-4
16384	10	1.2e-6	1.3e-6	1.0e-6	12	8.6e-7	1.5e-6	1.2e-6

equation $\Delta u = f$ with periodic boundary condition, where Δ is the Laplace operator, f is the input function, and u is the solution function. The energy functional of Poisson's equation is defined as the negative inner product of u and f, which can also be approximated by a quadratic form of the leading low-frequency Fourier components, which can be rewritten as,

$$\mathcal{E}(f) = -\langle f, u\rangle \approx \sum_{k\in[-K/2,K/2-1)^2} \frac{1}{|k|^2}\left|\widehat{f}_k\right|^2, \tag{6}$$

where $\widehat{f}_k$ is the k-th Fourier component of f. If the input function f is a linear combination of the Fourier components with $k \in [-K/2, K/2)^2$, equality is achieved in (6). In this numerical example, we assume the domain of f, $[0,1)^2$, is discredited by a uniform grid with 64 points on each dimension. f is a smooth periodic random function. It is generated from the Fourier interpolation of a fully random function defined on 8×8 grid. The reference energy is calculated via the discrete Laplace operator. 10^4 random instances of f is used as training data whereas 10^3 random instances are used as testing data. Detailed parameters of the network and training parameters can be found in Supp. C.

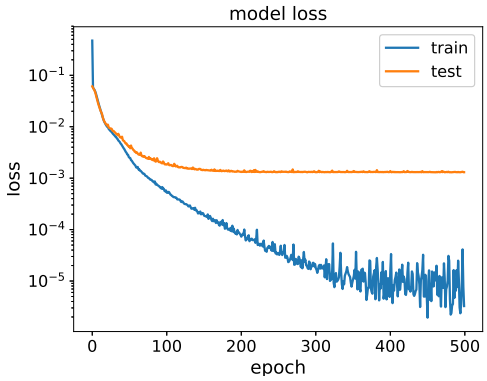


Figure 3: Mean square error of the trained CNN representing the energy functional of a 2D Poisson's equation. The network contains 3 convolution layers and a fully connected layer. Adam method is adopted to train the model with initial stepsize 10^{-4}.

Fig. 3 shows the decay of relative mean square error (MSE) of the network, which is the mean square error divided by the square of the averaged energy values. The loss of training data flatted after about 300 epochs and the loss of testing data stop improving after 150 epochs. The CNN is able to achieve 3 digits accuracy of the relative MSE, that is about 97% accuracy of the value of the energy.

5 Conclusion and Future Works

A low-complexity CNN with structured hard-coded weights and sparse across-channel connections is proposed, motivated by the *Butterfly* scheme. The hierarchical functional representation by *Butterfly-Net* is optimal in the sense that the model complexity is $O(N)$ and the computational complexity is $O(N \log N)$. The approximation accuracy to the Fourier kernel is proved to exponentially converge as the depths of the *Butterfly-Net* increases, which provides an approximation upper bound for CNNs in a large class of problems in scientific computing and image and signal processing.

The work can be extended in several directions. First, more applications of the *Butterfly-Net* should be explored such as those in image analysis and to compare the performance with other CNN architectures. Second, our current theoretical analysis does not address the case when the input data contains noise. In particular, adding rectified layers in *Butterfly-Net* can be interpreted as a thresholding denoising operation applied to the intermediate representations; a statistical analysis is desired. Finally, the training of the proposed *Butterfly-Net* with large data set should be addressed, in particular, the convergence of the training and the analysis of generalization of the trained network.

References

[1] Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945.

[2] Behler, J. and Parrinello, M. (2007). Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401.

[3] Bengio, Y., Goodfellow, I. J., and Courville, A. (2015). Deep learning. *Nature*, 521(7553):436–444.

[4] Berg, J. and Nyström, K. (2017). A unified deep artificial neural network approach to partial differential equations in complex geometries. *arXiv preprint arXiv:1711.06464*.

[5] Bölcskei, H., Grohs, P., Kutyniok, G., and Petersen, P. (2017). Optimal approximation with sparsely connected deep neural networks. *arXiv preprint arXiv:1705.01714*.

[6] Cai, J.-F., Dong, B., Osher, S., and Shen, Z. (2012). Image restoration: total variation, wavelet frames, and beyond. *Journal of the American Mathematical Society*, 25(4):1033–1089.

[7] Candès, E. J., Demanet, L., and Ying, L. (2007). Fast computation of Fourier integral operators. *SIAM J. Sci. Comput.*, 29(6):2464–2493.

[8] Candès, E. J., Demanet, L., and Ying, L. (2009). A fast butterfly algorithm for the computation of Fourier integral operators. *Multiscale Model. Simul.*, 7(4):1727–1750.

[9] Chan, T. F. and Shen, J. J. (2005). *Image processing and analysis: variational, PDE, wavelet, and stochastic methods*, volume 94. SIAM.

[10] Cohen, N., Sharir, O., and Shashua, A. (2016). On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728.

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

[12] Delalleau, O. and Bengio, Y. (2011). Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674.

[13] Demanet, L., Ferrara, M., Maxwell, N., Poulson, J., and Ying, L. (2012). A butterfly algorithm for synthetic aperture radar imaging. *SIAM Journal on Imaging Sciences*, 5(1):203–243.

[14] Demanet, L. and Ying, L. (2011). Discrete symbol calculus. *SIAM Rev.*, 53(1):71–104.

[15] E, W., Han, J., and Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380.

[16] Eldan, R. and Shamir, O. (2016). The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940.

[17] Gallant, A. R. and White, H. (1988). There exists a neural network that does not make avoidable mistakes. In *Proceedings of the Second Annual IEEE Conference on Neural Networks, San Diego, CA, I*.

[18] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

[19] Khoo, Y., Lu, J., and Ying, L. (2018). Solving for high dimensional committor functions using artificial neural networks. *arXiv preprint arXiv:1802.10275*.

[20] Kunis, S. and Melzer, I. (2012). A stable and accurate butterfly sparse Fourier transform. *SIAM J. Numer. Anal.*, 50(3):1777–1800.

[21] Le Roux, N. and Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural computation*, 20(6):1631–1649.

[22] Li, Y. and Yang, H. (2017). Interpolative butterfly factorization. *SIAM J. Sci. Comput.*, 39(2):A503–A531.

[23] Li, Y., Yang, H., Martin, E. R., Ho, K. L., and Ying, L. (2015a). Butterfly factorization. *Multiscale Model. Simul.*, 13(2):714–732.

[24] Li, Y., Yang, H., and Ying, L. (2015b). A multiscale butterfly algorithm for multidimensional Fourier integral operators. *Multiscale Model. Simul.*, 13(2):1–18.

[25] Liang, S. and Srikant, R. (2016). Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*.

[26] Long, Z., Lu, Y., Ma, X., and Dong, B. (2017). PDE-Net: Learning PDEs from data. *arXiv preprint arXiv:1710.09668*.

[27] Mallat, S. (2008). *A wavelet tour of signal processing: the sparse way*. Academic press.

[28] Mhaskar, H., Liao, Q., and Poggio, T. (2016). Learning functions: when is deep better than shallow. *arXiv preprint arXiv:1603.00988*.

[29] Mhaskar, H. N. and Poggio, T. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848.

[30] Michielssen, E. and Boag, A. (1996). A multilevel matrix decomposition algorithm for analyzing scattering from large structures. *IEEE Trans. Antennas Propag.*, 44(8):1086–1093.

[31] Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932.

[32] O'Neil, M., Woolfe, F., and Rokhlin, V. (2010). An algorithm for the rapid evaluation of special function transforms. *Appl. Comput. Harmon. Anal.*, 28(2):203–226.

[33] Rivlin, T. J. (1990). *Chebyshev polynomials: from approximation theory to algebra and number theory*. Wiley-Interscience, 2nd edition.

[34] Schneider, E., Dai, L., Topper, R. Q., Drechsel-Grau, C., and Tuckerman, M. E. (2017). Stochastic neural network approach for learning high-dimensional free energy surfaces. *Physical review letters*, 119(15):150601.

[35] Telgarsky, M. (2016). Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*.

[36] Yarotsky, D. (2017). Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114.

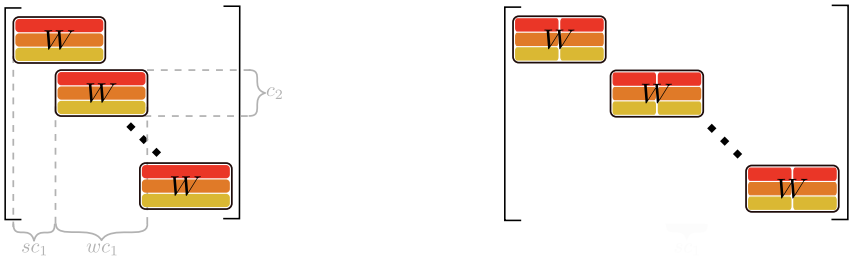
[37] Ying, L. (2009). Sparse Fourier transform via butterfly algorithm. *SIAM J. Sci. Comput.*, 31(3):1678–1694.

[38] Zhang, L., Han, J., Wang, H., Car, R., and E, W. (2018). DeePCG: constructing coarse-grained models via deep neural networks. *arXiv preprint arXiv:1802.08549*.

Supplementary Materials

A Matrix Representation of *Butterfly-Net*

We first show the matrix representation of the interpolation convolutional layer and the matrix representation of the *Butterfly-Net* simply stacks the interpolation convolutional layer together with an extra middle level local connection. Fig. 4 (a) represents (2) when both g and f are vectorized. If we permute the row ordering of the matrix, we would result blocks of convolution matrix with the number of blocks being the size of the output channels. Fig. 4 (b) assumes that $s = w$ and the matrix is further simplified to be a block diagonal matrix. According to the figures, we note that when $s = w$, the transpose of the matrix is also a representation of a interpolation convolutional layer with W replaced by W^T.



(a) Matrix of 1D interp. conv. layer (b) Matrix of 1D interp. conv. layer with $s = w$

Figure 4: Matrix representation of 1D interpolation convolutional layers

Since the CNN with mixing channel can already be represented as Fig. 4, *Butterfly-Net* simply stack interpolation convolutional layer together and drives non-mixing channel, which is equivalent to stack the matrix row-wise. We would explain the matrix representation for each step of the *Butterfly-Net*.

- *Preparation Layer.* The matrix representation is Fig. 4 (b) with 2^L diagonal blocks and each block is of size $r \times m$. The matrix is denoted as V.
- *Layer* $\ell = 1, \ldots, L/2$. Let the convolution kernel from $f^{(\ell-1)}(2j-1, k, i)$ and $f^{(\ell-1)}(2j, k, i)$ to $f^{(\ell)}(j, k, 2i-1)$ is denoted as $W^{(\ell)}_{2i-1}$. Similarly the convolution kernel from $f^{(\ell-1)}(2j-1, k, i)$ and $f^{(\ell-1)}(2j, k, i)$ to $f^{(\ell)}(j, k, 2i)$ is denoted as $W^{(\ell)}_{2i}$. Assemble the above small matrices together to get

$$H^{(\ell)}_i = \begin{pmatrix} W^{(\ell)}_{2i-1} & & \\ & \ddots & \\ & & W^{(\ell)}_{2i-1} \\ \hline W^{(\ell)}_{2i} & & \\ & \ddots & \\ & & W^{(\ell)}_{2i} \end{pmatrix}, \quad \text{and } H^{(\ell)} = \begin{pmatrix} H^{(\ell)}_1 & & & \\ & H^{(\ell)}_2 & & \\ & & \ddots & \\ & & & H^{(\ell)}_{2^{L-\ell}} \end{pmatrix}.$$

  Each block $W^{(\ell)}_{2i-1}$ and $W^{(\ell)}_{2i}$ of the top and bottom part of $H^{(\ell)}_i$ is of form Fig. 4 (b).
- *Layer M.* The matrix representation of layer M is

$$M = \begin{pmatrix} M_1 & & \\ & \ddots & \\ & & M_{2^{L/2}} \end{pmatrix}, \quad \text{and } M_j = \begin{pmatrix} W_{j,1} & & \\ & \ddots & \\ & & W_{j,2^{L/2}} \end{pmatrix}.$$

- *Layer* $\ell = L/2 + 1, \ldots, L$. The matrix representation here, denoted as $G^{(\ell)}$ has exactly the same structure as that of $H^{(\ell)}$.
- *Feature Layer.* The matrix representation of this layer is a block diagonal matrix, denoted as U.

Based on the matrix representation of each level of the *Butterfly-Net*, we could write down the overall matrix representation together with ReLU activation layer as,

$$g = \mathcal{B}(f) = UF\left[G^{(L)}F\left[\cdots G^{(L/2+1)}F\left[MF\left[H^{(L/2)}F\left[\cdots H^{(1)}F\left[Vf\right]\right]\right]\right]\right]\right], \tag{7}$$

where $U, G^{(\ell)}, M, H^{(\ell)}, V$ are defined in Supp. and $F[\cdot]$ denotes the operation of adding the bias and apply the ReLU activation.

B Proof of Theorem 3.1

This section first derives a low-rank interpolation of the discrete Fourier kernel, which can be understood as a convolution with mixing and non-mixing channels. Then in Sec. B.2, we fill the matrix representation of *Butterfly-Net* with the interpolation representation. This implies that the discrete Fourier transform (DFT) matrix can be approximately written as *Butterfly-Net*. Later in Sec. B.3, we prove Theorem 3.1

B.1 Low-rank approximation of Fourier kernel

It is well-known that DFT matrix has orthonormal rows and hence, is a full rank matrix. In order to obtain the local convolutional property, we first define the L level hierarchical partition of $[0, 1)$ and $[-K/2, K/2)$, respectively. Let $B_0^0 = [0, 1)$ be the domain on level 0. On level 1, the domain B_0^0 is evenly partitioned into $B_0^1 = [0, 1/2)$ and $B_1^1 = [1/2, 1)$. We conduct the partition recursively, i.e., $B_i^{\ell-1}$ is evenly partitioned into B_{2i}^{ℓ} and B_{2i+1}^{ℓ}. In the end, the partition on level ℓ is $\{B_i^\ell, i = 0, \ldots, 2^\ell - 1\}$ and each B_i^ℓ is of length $2^{-\ell}$. Similarly, we conduct the hierarchical bipartition on $[-K/2, K/2)$, denoted them as A_j^ℓ, and A_j^ℓ is of length $K \cdot 2^{-\ell}$. As we will see, the partition of domain will be used in a complementary way, A_i^ℓ will be paired with $B_j^{L-\ell}$ for all choices of i and j, such that the Fourier kernel restricted to A_i^ℓ and $B_j^{L-\ell}$ permits a low-rank approximation, see Theorem B.1. Theorem B.1 actually depends only on the length of the domains of t and ξ, but does not depend on the location.

We first give a brief introduction of the Chebyshev interpolation with r Chebyshev points. The Chebyshev grid of order r on $[-1/2, 1/2]$ is defined as,

$$\left\{ z_i = \frac{1}{2}\cos\left(\frac{(i-1)\pi}{r}\right)\right\}_{i=1}^{r}. \tag{8}$$

The r points Chebyshev interpolation of any function $f(x)$ on $[-1/2, 1/2]$ is defined as,

$$\Pi_r f(x) = \sum_{k=1}^{r} f(z_k)\mathcal{L}_k(x), \tag{9}$$

where $\mathcal{L}_k(x)$ is the Lagrange polynomial as in (3).

One important property is that for any fixed k, $\mathcal{L}_k(x)$ depends only on the relative location of x and $\{z_i\}$ and is independent of the length and location of the interval. This is essential for convolutional representation.

Several earlier works [7, 8, 24] proved the Chebyshev interpolation representation for Fourier integral kernel, which is a generalized Fourier kernel. Theorem B.1 is a special case of these earlier work but with more precise estimation of the prefactors.

Theorem B.1. *Let L and r be two parameters such that $\pi e K \le r 2^L$. For any $\xi \in A_i^\ell$ and $t \in B_j^{L-\ell}$, there exists an Chebyshev interpolation representation of the Fourier kernel,*

$$\left| e^{-2\pi\imath\xi\cdot t} - \sum_{k=1}^{r} e^{-2\pi\imath\xi\cdot t_k} e^{-2\pi\imath\xi_0\cdot(t-t_k)}\mathcal{L}_k(t)\right| \le \left(2 + \frac{2}{\pi}\ln r\right)\left(\frac{2\pi e K}{4r2^L}\right)^r, \tag{10}$$

and

$$\left| e^{-2\pi\imath\xi\cdot t} - \sum_{k=1}^{r} e^{-2\pi\imath(\xi-\xi_k)\cdot t_0}\mathcal{L}_k(\xi) e^{-2\pi\imath\xi_k\cdot t}\right| \le \left(2 + \frac{2}{\pi}\ln r\right)\left(\frac{2\pi e K}{4r2^L}\right)^r, \tag{11}$$

where ξ_0 and t_0 are the centers of A_i^ℓ and $B_j^{L-\ell}$ respectively, ξ_k and t_k are Chebyshev points on A_i^ℓ and $B_j^{L-\ell}$ respectively.

Lemma B.2. *Let $f(y) \in C_{[a,b]}$ and $\mathcal{P}_r$ be the space spanned by the monomials y^r. The projection operator Π_r mapping f into its Lagrange interpolation on the r Chebyshev grid obeys,*

$$\|f - \Pi_r f\|_\infty \leq \left(2 + \frac{2}{\pi} \ln r\right) \inf_{g \in \mathcal{P}_r} \|f - g\|_\infty. \tag{12}$$

The proof of Lemma B.2 can be found in [33].

Proof of Theorem B.1. At level ℓ, we assume $\xi \in A_i^\ell \subseteq [-K/2, K/2)$ and $t \in B_j^{L-\ell} \subseteq [0, 1)$. According to the definition of domain partition, at level ℓ, we have $w(A_i^\ell) = K/2^\ell$ and $w(B_j^{L-\ell}) = 1/2^{L-\ell}$, which implies $w(A_i^\ell) \cdot w(B_j^{L-\ell}) = K/2^\ell 2^{L-\ell} = K \cdot 2^{-L}$, where $w(\cdot)$ denote the function of length.

The Fourier kernel $\mathcal{K}(\xi, t) = e^{-2\pi\imath\xi\cdot t}$ can be factorized as,

$$\begin{aligned}\mathcal{K}(\xi, t) =& e^{-2\pi\imath(\xi\cdot t - \xi_0\cdot t - \xi\cdot t_0 + \xi_0\cdot t_0)} \cdot e^{-2\pi\imath\xi_0\cdot t} \cdot e^{-2\pi\imath\xi\cdot t_0} \cdot e^{2\pi\imath\xi_0\cdot t_0} \\ =& e^{-2\pi\imath R(\xi,t)} \cdot e^{-2\pi\imath\xi_0\cdot t} \cdot e^{-2\pi\imath\xi\cdot t_0} \cdot e^{2\pi\imath\xi_0\cdot t_0},\end{aligned} \tag{13}$$

where $R(\xi, t) = (\xi - \xi_0) \cdot (t - t_0)$.

Next, we show the r-term truncation error of the first term in the second line of (13). Based on the power expansion of $e^{-2\pi\imath R(\xi,t)}$, i.e.,

$$e^{-2\pi\imath R(\xi,t)} = \sum_{k=0}^{\infty} \frac{(-2\pi\imath R(\xi, t))^k}{k!}, \tag{14}$$

the r-term truncation error can be bounded as,

$$\begin{aligned}\delta =& \left| e^{-2\pi\imath R(\xi,t)} - \sum_{k=0}^{r} \frac{(-2\pi\imath R(\xi, t))^k)}{k!} \right| = \left| \sum_{k=r+1}^{\infty} \frac{(-2\pi\imath R(\xi, t))^k}{k!} \right| \\ \leq& \sum_{k=r+1}^{\infty} \frac{1}{k!} \left(\frac{2\pi K}{4 \cdot 2^L}\right)^k \leq \sum_{k=r+1}^{\infty} \frac{e^k}{k^k} \left(\frac{2\pi K}{4 \cdot 2^L}\right)^k \leq \sum_{k=r+1}^{\infty} \left(\frac{2\pi e K}{4r2^L}\right)^k \leq \left(\frac{2\pi e K}{4r2^L}\right)^r,\end{aligned} \tag{15}$$

where the last inequality uses $\pi e K \leq r 2^L$. We also notice that, for any fixed ξ, $\sum_{k=0}^{r} \frac{(2\pi\imath R(\xi,\cdot))^k}{k!} \in \mathcal{P}_r$. Applying Lemma B.2, we obtain,

$$\left\| e^{-2\pi\imath R(\xi,t)} - \sum_{k=1}^{r} e^{-2\pi\imath R(\xi,t_k)} \mathcal{L}_k(t) \right\| \leq \left(2 + \frac{2}{\pi} \ln r\right) \delta. \tag{16}$$

By substituting the explicit expression of $R(\xi, t)$, we obtain one of the conclusion,

$$\left\| e^{-2\pi\imath\xi\cdot t} - \sum_{k=1}^{r} e^{-2\pi\imath\xi\cdot t_k} e^{-2\pi\imath\xi_0\cdot(t-t_k)} \mathcal{L}_k(t) \right\| \leq \left(2 + \frac{2}{\pi} \ln r\right) \left(\frac{2\pi e K}{4r2^L}\right)^r, \tag{17}$$

for any $\xi \in A_i^\ell$ and $t \in B_j^{L-\ell}$.

Similarly, for any fixed t, we have $\sum_{k=0}^{r} \frac{(2\pi\imath R(\xi,t))^k}{k!} \in \mathcal{P}_r(\xi)$. Hence the second conclusion can be obtained through the same procedure. □

B.2 Forward *Butterfly-Net*

Theorem B.1 provides a low-rank approximation of the Fourier kernel restricted in A_i^ℓ and $B_j^{L-\ell}$. When $\ell \leq L/2$, we adopt (10) and $e^{-2\pi\imath\xi_0\cdot(t-t_k)}\mathcal{L}_k(t)$ depends only on relative difference $t - t_k$. Summing over all $t \in B_j^{L-\ell}$ and all $B_j^{L-\ell}$, $e^{-2\pi\imath\xi_0\cdot(t-t_k)}\mathcal{L}_k(t)$ is a convolution kernel with filter size

being equal to the stride size. When $\ell \geq {}^{L}/_{2}+1$, we adopt (11) and $e^{-2\pi\imath(\xi-\xi_k)\cdot t_0}\mathcal{L}_k(\xi)$ depends only on relative difference $\xi-\xi_k$. Summing over all $\xi \in A_i^\ell$ and all A_i^ℓ, $e^{-2\pi\imath(\xi-\xi_k)\cdot t_0}\mathcal{L}_k(\xi)$ is a block diagonal matrix as the transpose of Fig. 4 (b) and hence is a convolution. We assume the input data is evaluated at t equally distributed on B_j^L and the output data is provided at ξ equally distributed on A_i^L. Further, we denote $\xi_0^{A_i^\ell}$ and $t_0^{B_j^\ell}$ as the centers of A_i^ℓ and B_j^ℓ respectively; $\xi_k^{A_i^\ell}$ and $t_k^{B_j^\ell}$ as the Chebyshev points of A_i^ℓ and B_j^ℓ respectively; and $\xi^{A_i^L}$ and $t^{B_j^L}$ as the data points in A_i^L and B_j^L respectively.

- *Preparation Layer.* Let the diagonal block of V be V_j. Then V_j is a matrix of size $r \times m$ with entry $e^{-2\pi\imath\xi_0^{A_0^0}\cdot(t^{B_j^L}-t_k^{B_j^L})}\mathcal{L}_k(t^{B_j^L})$, where $t_k^{B_j^L}$ and $t^{B_j^L}$ are row and column index respectively.
- *Layer* $\ell = 1,\ldots,{}^{L}/_{2}$. $W_{2i-1}^{(\ell)}$ is a matrix of size $r \times 2r$ with entry $e^{-2\pi\imath\xi_0^{A_{2i-1}^\ell}\cdot(t_{k'}^{B_1^{L-\ell+1}}-t_k^{B_1^{L-\ell}})}\mathcal{L}_k(t_{k'}^{B_1^{L-\ell+1}})$ and $e^{-2\pi\imath\xi_0^{A_{2i-1}^\ell}\cdot(t_{k'}^{B_2^{L-\ell+1}}-t_k^{B_1^{L-\ell}})}\mathcal{L}_k(t_{k'}^{B_2^{L-\ell+1}})$, where $t_k^{B_1^{L-\ell}}$ is the row index, and $t_{k'}^{B_1^{L-\ell+1}}$ and $t_{k'}^{B_2^{L-\ell+1}}$ are column index. The former generates the left $r \times r$ submatrix of $W_{2i-1}^{(\ell)}$, whereas the later generates the right half submatrix. $W_{2i}^{(\ell)}$ is of the same format with A_{2i-1}^ℓ being replaced by A_{2i}^ℓ.
- *Layer M.* $W_{2i-1}^{(M)}$ is a matrix of size $r \times r$ with entry $e^{-2\pi\imath\xi_{k'}^{A_i^{L/2}}\cdot t_k^{B_j^{L/2}}}$.
- *Layer* $\ell = {}^{L}/_{2}+1,\ldots,L$. $W_{2j-1}^{(\ell)}$ is a matrix of size $r \times 2r$ with entry $e^{-2\pi\imath(\xi_{k'}^{A_1^{\ell-1}}-\xi_k^{A_1^\ell})\cdot t_0^{B_{2j-1}^{L-\ell}}}\mathcal{L}_k(\xi_{k'}^{A_1^{\ell-1}})$ and $e^{-2\pi\imath(\xi_{k'}^{A_1^{\ell-1}}-\xi_k^{A_1^\ell})\cdot t_0^{B_{2j}^{L-\ell}}}\mathcal{L}_k(\xi_{k'}^{A_1^{\ell-1}})$, where $\xi_{k'}^{A_1^{\ell-1}}$ is the row index, and $\xi_k^{A_1^\ell}$ is column index. The former generates the left $r \times r$ submatrix of $W_{2i-1}^{(\ell)}$, whereas the later generates the right half submatrix. $W_{2i}^{(\ell)}$ is of the same format with $A_1^{\ell-1}$ being replaced by $A_2^{\ell-1}$.
- *Feature Layer.* Let the diagonal block of U be U_i. Then U_i is a matrix of size $K/2^L \times r$ with entry $e^{-2\pi\imath(\xi^{A_j^L}-\xi_k^{A_j^L})\cdot t_0^{B_0^0}}\mathcal{L}_k(\xi^{A_j^L})$, where $\xi^{A_j^L}$ and $\xi_k^{A_j^L}$ are row and column index respectively.

B.3 Proof

Lemma B.3. *Let $\{z_i\}_{i=1}^r$ be r Chebyshev points and $\mathcal{L}_k(x)$ be the Lagrange polynomial of order r. For any r, the Lebesgue constant Λ_r is bounded as*

$$\Lambda_r = \max_{-1\leq x\leq 1}\sum_{i=1}^{r}|\mathcal{L}_i(x)| \leq \frac{2}{\pi}\ln r + 1.$$

Lemma B.3 is a standard result of Chebyshev interpolation and the proof can be found in [33].

Corollary B.4. *Let $\{z_i\}_{i=1}^r$ be r Chebyshev points and $\mathcal{L}_k(x)$ be the Lagrange polynomial of order r. For any r and $i \leq r$,*

$$\max_{-1\leq x\leq 1}|\mathcal{L}_i(x)| \leq \frac{2}{\pi}\ln r + 1.$$

Corollary B.4 is an immediate result of Lemma B.3.

Lemma B.5. *Let V be the block diagonal matrix defined in the preparation layer, then*

$$\|V\|_1 \leq \frac{2}{\pi}\ln r + 1 \text{ and } \|V\|_\infty \leq m\left(\frac{2}{\pi}\ln r + 1\right)$$

where r is the number of Chebyshev points and $m = N/2^L$.

Proof. V is a block diagonal matrix with block V_j for $j = 1, 2, \ldots, 2^L$. V_j are the same $r \times m$ with entry $e^{-2\pi\imath\xi_0^{A_0^0}\cdot(t^{B_1^L}-t_k^{B_1^L})}\mathcal{L}_k(t^{B_j^L})$. By the definition of matrix 1-norm, we have

$$\|V\|_1 = \|V_1\|_1 \leq \max_{t\in B_1^L}\sum_{t_k^{B_1^L}}\left|e^{-2\pi\imath\xi_0^{A_0^0}\cdot(t-t_k^{B_1^L})}\mathcal{L}_k(t)\right| \leq \max_{t\in B_1^L}\sum_{t_k^{B_1^L}}|\mathcal{L}_k(t)| \leq \frac{2}{\pi}\ln r + 1. \quad (18)$$

By the definition of matrix ∞-norm, we have

$$\|V\|_\infty = \|V_1\|_\infty \le \sum_{t\in B_1^L}\sum_{t_k^{B_1^L}} \left| e^{-2\pi\imath\xi_0^{A_0^0}\cdot(t-t_k^{B_1^L})}\mathcal{L}_k(t)\right| \le m\left(\frac{2}{\pi}\ln r+1\right). \tag{19}$$

□

Lemma B.6. *Let M be the block diagonal matrix defined in the Layer M, then*

$$\|M\|_1 \le r \text{ and } \|M\|_\infty \le r,$$

where r is the number of Chebyshev points.

Proof. Based on the structure of M and the definition of matrix 1-norm, we have

$$\|M\|_1 = \max_j \|M_j\|_1 \le \max_j \max_i \|W_{j,i}\|_1 = \max_{j,i}\max_{t_k^{B_j^{L/2}}} \sum_{\xi_{k'}^{A_i^{L/2}}} \left| e^{-2\pi\imath\xi_{k'}^{A_i^{L/2}}\cdot t_k^{B_j^{L/2}}}\right| = r. \tag{20}$$

Based on the structure of M and the definition of matrix ∞-norm, we have

$$\|M\|_\infty = \max_j \|M_j\|_\infty \le \max_j \max_i \|W_{j,i}\|_\infty = \max_{j,i}\max_{t_k^{B_j^{L/2}}} \sum_{t_k^{B_j^{L/2}}} \left| e^{-2\pi\imath\xi_{k'}^{A_i^{L/2}}\cdot t_k^{B_j^{L/2}}}\right| = r. \tag{21}$$

□

Lemma B.7. *Let $H^{(\ell)}$ be the block diagonal matrix defined in the Layer $\ell = 1,\dots,L/2$, then*

$$\left\|H^{(\ell)}\right\|_1 \le 2\left(\frac{2}{\pi}\ln r+1\right),$$

where r is the number of Chebyshev points.

Proof. Based on the structure of $H^{(\ell)}$ and the definition of matrix 1-norm, we have

$$\begin{aligned}
\left\|H^{\ell}\right\|_1 =& \max_i \left\|H_i^{(\ell)}\right\|_1 \le \max_i \left(\left\|W_{2i-1}^{(\ell)}\right\|_1 + \left\|W_{2i}^{(\ell)}\right\|_1\right)\\
\le& \max_i \left(\max_{t\in B_1^{L-\ell}} \sum_{t_k^{B_1^{L-\ell}}} \left| e^{-2\pi\imath\xi_0^{A_{2i-1}^{\ell}}\cdot(t-t_k^{B_1^{L-\ell}})}\mathcal{L}_k(t)\right| \right.\\
&\left. + \max_{t\in B_1^{L-\ell}} \sum_{t_k^{B_1^{L-\ell}}} \left| e^{-2\pi\imath\xi_0^{A_{2i}^{\ell}}\cdot(t-t_k^{B_1^{L-\ell}})}\mathcal{L}_k(t)\right| \right)\\
\le& \max_i \left(\max_{t\in B_1^{L-\ell}} \sum_{t_k^{B_1^{L-\ell}}} |\mathcal{L}_k(t)| + \max_{t\in B_1^{L-\ell}} \sum_{t_k^{B_1^{L-\ell}}} |\mathcal{L}_k(t)| \right)\\
\le& 2\left(\frac{2}{\pi}\ln r+1\right).
\end{aligned} \tag{22}$$

□

Lemma B.8. *Let $G^{(\ell)}$ be the block diagonal matrix defined in the Layer $\ell = L/2+1,\dots,L$, then*

$$\left\|G^{(\ell)}\right\|_1 \le 2r\left(\frac{2}{\pi}\ln r+1\right),$$

where r is the number of Chebyshev points.

Proof. Based on the structure of $G^{(\ell)}$ and the definition of matrix 1-norm, we have

$$
\begin{aligned}
\left\|G^{\ell}\right\|_1 =& \max_j \left\|G_j^{(\ell)}\right\|_1 \leq \max_j \left(\left\|W_{2j-1}^{(\ell)}\right\|_1 + \left\|W_{2j}^{(\ell)}\right\|_1\right) \\
\leq& \max_j \left\{ \max_k \left(\sum_{\xi_{k'}^{A_1^{\ell-1}}} \left| e^{-2\pi\imath(\xi_{k'}^{A_1^{\ell-1}} - \xi_k^{A_1^{\ell}} \cdot t_1^{B_{2j-1}^{L-\ell}})} \mathcal{L}_k(\xi_{k'}^{A_1^{\ell-1}}) \right| \right.\right. \\
& \left.\left. + \sum_{\xi_{k'}^{A_2^{\ell-1}}} \left| e^{-2\pi\imath(\xi_{k'}^{A_2^{\ell-1}} - \xi_k^{A_1^{\ell}} \cdot t_1^{B_{2j-1}^{L-\ell}})} \mathcal{L}_k(\xi_{k'}^{A_2^{\ell-1}}) \right| \right) \right. \\
& + \max_k \left(\sum_{\xi_{k'}^{A_1^{\ell-1}}} \left| e^{-2\pi\imath(\xi_{k'}^{A_1^{\ell-1}} - \xi_k^{A_1^{\ell}} \cdot t_1^{B_{2j}^{L-\ell}})} \mathcal{L}_k(\xi_{k'}^{A_1^{\ell-1}}) \right| \right. \\
& \left. \left. \left. + \sum_{\xi_{k'}^{A_2^{\ell-1}}} \left| e^{-2\pi\imath(\xi_{k'}^{A_2^{\ell-1}} - \xi_k^{A_1^{\ell}} \cdot t_1^{B_{2j}^{L-\ell}})} \mathcal{L}_k(\xi_{k'}^{A_2^{\ell-1}}) \right| \right) \right\} \\
\leq& \max_j \left\{ \max_k \left(\sum_{\xi_{k'}^{A_1^{\ell-1}}} \left| \mathcal{L}_k(\xi_{k'}^{A_1^{\ell-1}}) \right| + \sum_{\xi_{k'}^{A_2^{\ell-1}}} \left| \mathcal{L}_k(\xi_{k'}^{A_2^{\ell-1}}) \right| \right) \right. \\
& \left. + \max_k \left(\sum_{\xi_{k'}^{A_1^{\ell-1}}} \left| \mathcal{L}_k(\xi_{k'}^{A_1^{\ell-1}}) \right| + \sum_{\xi_{k'}^{A_2^{\ell-1}}} \left| \mathcal{L}_k(\xi_{k'}^{A_2^{\ell-1}}) \right| \right) \right\} \\
\leq& 2r \left(\frac{2}{\pi} \ln r + 1 \right).
\end{aligned}
\tag{23}
$$

□

Lemma B.9. *Let $H^{(\ell)}$ be the block diagonal matrix defined in the Layer $\ell = 1, \ldots, L/2$, then*

$$
\left\|H^{(\ell)}\right\|_\infty \leq 2r \left(\frac{2}{\pi} \ln r + 1 \right),
$$

where r is the number of Chebyshev points.

Lemma B.10. *Let $G^{(\ell)}$ be the block diagonal matrix defined in the Layer $\ell = L/2 + 1, \ldots, L$, then*

$$
\left\|G^{(\ell)}\right\|_\infty \leq 2 \left(\frac{2}{\pi} \ln r + 1 \right),
$$

where r is the number of Chebyshev points.

The proofs of Lemma B.9 and Lemma B.10 follow that of Lemma B.8 and Lemma B.7 respectively.

Proof of Theorem 3.1. Assume the *Butterfly-Net* is full filled with the Forward *Butterfly-Net* as in Sec. B.2 with all rectifiers are deactivated. The kernel matrix is approximated by a product of sparse matrices, $U, G^{(\ell)}, M, H^{(\ell)}$, and V. We write the exact matrix product with error matrix $E^{(\ell)}$,

$$
\begin{aligned}
\mathcal{K} = \Big[\Big[\Big[\Big[\Big[\Big[\Big[U + E^{(L)} \Big] G^{(L)} + E^{(L-1)} \Big] \cdots \Big] G^{(L/2+1)} + E^{(M)} \\
+ \ M \Big] H^{(L/2)} + E^{(L/2)} \Big] \cdots \Big] H^{(1)} + E^{(1)} \Big] V + E^{(0)}.
\end{aligned}
\tag{24}
$$

In the following derivation, we adopt the notation, $\Lambda_r = \frac{2}{\pi}\ln r + 1$. Then, we have,

$$
\begin{aligned}
&\left\|\mathcal{K} - UG^{(L)}\cdots G^{(L/2+1)}MH^{(L/2)}\cdots H^{(1)}V\right\|_1 \\
\leq&\left\|E^{(L/2)}G^{(L)}\cdots G^{(L/2+1)}MH^{(L/2)}\cdots H^{(1)}V\right\|_1 \\
&+\cdots+\left\|E^{(M)}MH^{(L/2)}\cdots H^{(1)}V\right\|_1+\cdots+\left\|E^{(1)}V\right\|_1+\left\|E^{(0)}\right\|_1 \\
\leq&\left\|E^{(L/2)}\right\|_1\left(\prod_{\ell=L/2+1}^{L}\left\|G^{(\ell)}\right\|_1\right)\|M\|_1\left(\prod_{\ell=1}^{L/2}\left\|H^{(\ell)}\right\|_1\right)\|V\|_1 \\
&+\cdots+\left\|E^{(M)}\right\|_1\|M\|_1\left(\prod_{\ell=1}^{L/2}\left\|H^{(\ell)}\right\|_1\right)\|V\|_1+\cdots+\left\|E^{(1)}\right\|_1\|V\|_1+\left\|E^{(0)}\right\|_1 \\
\leq&\delta_1\sum_{\ell=0}^{L/2}\left(2r\Lambda_r\right)^{\ell}r\left(2\Lambda_r\right)^{L/2}\Lambda_r+\delta_1\sum_{\ell=0}^{L/2}\left(2\Lambda_r\right)^{\ell}\Lambda_r+\delta_1 \\
\leq&\left(\frac{\left(2r\Lambda_r\right)^{L/2+1}-1}{2r\Lambda_r-1}r\left(2\Lambda_r\right)^{L/2}\Lambda_r+\frac{\left(2\Lambda_r\right)^{L/2+1}-1}{2\Lambda_r-1}\Lambda_r+1\right)\delta_1 \\
\leq&\left(\left(2r\Lambda_r\right)^{L/2+1}\left(2\Lambda_r\right)^{L/2}+\left(2\Lambda_r\right)^{L/2+1}+1\right)\delta_1 \\
\leq&\left(4r\Lambda_r^2\right)^{L/2+1}\delta_1,
\end{aligned}
\tag{25}
$$

where the last few inequalities adopt the fact that $r > 1$ and $\Lambda_r > 1$, δ_1 is a uniform upper bound for $\left\|E^{(\ell)}\right\|_1$. Theorem B.1 provides an upper bound for each entry of $E^{(\ell)}$. Hence, the uniform upper bound for the 1-norm is

$$
\delta_1 = \max_{\ell}\left\|E_1^{(\ell)}\right\|_1 \leq 2^L r\left(1+\Lambda_r\right)\left(\frac{2\pi eK}{4r2^L}\right)^r. \tag{26}
$$

Substituting the upper bound of δ_1 into (25), we obtain,

$$
\begin{aligned}
&\left\|\mathcal{K} - UG^{(L)}\cdots G^{(L/2+1)}MH^{(L/2)}\cdots H^{(1)}V\right\|_1 \\
\leq&\left(4r\Lambda_r^2\right)^{L/2+1}2^L r\left(1+\Lambda_r\right)\left(\frac{2\pi eK}{4r2^L}\right)^r \\
\leq&C_{r,K}\left(\frac{r\Lambda_r^2}{4^{r-2}}\right)^{L/2} = C_{r,K}\left(\frac{r\left(\frac{2}{\pi}\ln r+1\right)^2}{4^{r-2}}\right)^{L/2},
\end{aligned}
\tag{27}
$$

where $C_{r,K} = {4(2+2/\pi\ln r)^3 r^2(\pi eK)^r}/{(2r)^r}$ is a constant depends on r and K, and is independent of L. Equation (27) says that when r is chosen such that $\frac{r\Lambda_r^2}{4^{r-2}} < 1$ ($r > 3$ is sufficient), increasing L reduces the error exponentially. Here L is depth of the neural network.

On the other side, measuring the error under matrix ∞-norm, we have,

$$\begin{aligned}
&\left\|\mathcal{K} - UG^{(L)}\cdots G^{(L/2+1)}MH^{(L/2)}\cdots H^{(1)}V\right\|_\infty \\
\leq&\left\|E^{(L/2)}\right\|_\infty\left(\prod_{\ell=L/2+1}^{L}\left\|G^{(\ell)}\right\|_\infty\right)\|M\|_\infty\left(\prod_{\ell=1}^{L/2}\left\|H^{(\ell)}\right\|_\infty\right)\|V\|_1 \\
&+\cdots+\left\|E^{(M)}\right\|_\infty\|M\|_\infty\left(\prod_{\ell=1}^{L/2}\left\|H^{(\ell)}\right\|_\infty\right)\|V\|_\infty+\cdots+\left\|E^{(1)}\right\|_\infty\|V\|_\infty+\left\|E^{(0)}\right\|_\infty \\
\leq&\delta_\infty\sum_{\ell=0}^{L/2}\left(2\Lambda_r\right)^\ell r\left(2r\Lambda_r\right)^{L/2}m\Lambda_r+\delta_\infty\sum_{\ell=0}^{L/2}\left(2r\Lambda_r\right)^\ell m\Lambda_r+\delta_\infty \\
\leq&\left(\frac{\left(2\Lambda_r\right)^{L/2+1}-1}{2\Lambda_r-1}r\left(2r\Lambda_r\right)^{L/2}m\Lambda_r+\frac{\left(2r\Lambda_r\right)^{L/2+1}-1}{2r\Lambda_r-1}m\Lambda_r+1\right)\delta_\infty \\
\leq&\left(m(2r\Lambda_r)^{L/2+1}\left(2\Lambda_r\right)^{L/2}+m(2r\Lambda_r)^{L/2+1}+1\right)\delta_\infty \\
\leq&(4r\Lambda_r^2)^{L/2+1}m\delta_\infty,
\end{aligned} \tag{28}$$

where δ_∞ is the uniform upper bound for $\left\|E^{(\ell)}\right\|_\infty$ satisfying

$$\delta_\infty=\max_\ell\left\|E_1^{(\ell)}\right\|_\infty\leq 2^L r\left(1+\Lambda_r\right)\left(\frac{2\pi eK}{4r2^L}\right)^r. \tag{29}$$

Therefore, we reach to the ∞-norm bound of the error,

$$\begin{aligned}
&\left\|\mathcal{K} - UG^{(L)}\cdots G^{(L/2+1)}MH^{(L/2)}\cdots H^{(1)}V\right\|_\infty \\
\leq&mC_{r,K}\left(\frac{r\Lambda_r^2}{4^{r-2}}\right)^{L/2}=mC_{r,K}\left(\frac{r\left(\frac{2}{\pi}\ln r+1\right)^2}{4^{r-2}}\right)^{L/2},
\end{aligned} \tag{30}$$

where $C_{r,K}$ is the same constant as in (27).

Applying Riesz-Thorin interpolation theorem together with (27) and (30), we obtain the error bound under the matrix p-norm,

$$\left\|\mathcal{K} - UG^{(L)}\cdots G^{(L/2+1)}MH^{(L/2)}\cdots H^{(1)}V\right\|_p\leq m^{1-\frac{1}{p}}C_{r,K}\left(\frac{r\left(\frac{2}{\pi}\ln r+1\right)^2}{4^{r-2}}\right)^{L/2}, \tag{31}$$

for any $1\leq p\leq\infty$.

Since the input vector f is bounded, we can also pick the shift parameters associated with every activation layer such that all of them are deactivated. Hence, we obtain, for any bounded vector f, the conclusion of Theorem 3.1, which is the direct result of (31). □

C Network Configuration for Numerical Results

```
import numpy as np
import scipy.io as spio
from keras.models import Sequential
from keras.layers import Conv2D, Activation, \
        MaxPooling2D, AveragePooling2D, Flatten, Dense
from keras import optimizers as opts

#========================================================
# Construct CNN model
#--------------------------------------------------------
r = 4
```

```
levels = 3
init_n = int(n/(2**levels))

model = Sequential()
model.add(Conv2D(r*r, (init_n,init_n), strides=(init_n,init_n),
        padding='valid', input_shape=(n, n, 1)))
model.add(Activation('relu'))

for it in range(0,levels-1):
    model.add(Conv2D(r*r*(4**(it+1)), (2,2), strides=(2,2),
            padding='valid'))
    model.add(Activation('relu'))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dense(1))

model.summary()

#=============================================================
# Compile and Fit CNN model
#-------------------------------------------------------------

adamopt = opts.Adam(lr=0.0001)
model.compile(optimizer=adamopt, loss='mse')

history = model.fit( X_train, Y_train, batch_size=100, epochs=500,
            verbose=1, validation_data=(X_test, Y_test))
```