Journal of Computational Physics 384 (2019) 1–15

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

BCR-Net: A neural network based on the nonstandard wavelet form

Yuwei Fan [a,*], Cindy Orozco Bohorquez [b], Lexing Ying [a,b,c]

[a] *Department of Mathematics, Stanford University, Stanford, CA 94305, United States of America*
[b] *Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305, United States of America*
[c] *Facebook AI Research, Menlo Park, CA 94025, United States of America*

ARTICLE INFO

Article history:
Received 20 October 2018
Accepted 12 February 2019
Available online 18 February 2019

Keywords:
Wavelet transform
Nonstandard form
Artificial neural network
Convolutional network
Locally connected network

ABSTRACT

This paper proposes a novel neural network architecture inspired by the nonstandard form proposed by Beylkin et al. (1991) [7]. The nonstandard form is a highly effective wavelet-based compression scheme for linear integral operators. In this work, we first represent the matrix-vector product algorithm of the nonstandard form as a linear neural network where every scale of the multiresolution computation is carried out by a locally connected linear sub-network. In order to address nonlinear problems, we propose an extension, called BCR-Net, by replacing each linear sub-network with a deeper and more powerful nonlinear one. Numerical results demonstrate the efficiency of the new architecture by approximating nonlinear maps that arise in homogenization theory and stochastic computation.

1. Introduction

This paper proposes a novel neural network architecture based on wavelet-based compression schemes. There has been a long history of research on representing differential and integral operators using wavelet bases. Such a representation is particularly attractive for integral operators such as pseudo-differential operators and Calderon-Zygmund operators because the wavelet-based representation is often sparse due to the vanishing moment property of the wavelet basis [11,13]. For a typical problem discretized with N unknowns, a direct application of the wavelet transform, followed by a thresholding step, results in $O(N \log N)$ significant matrix entries, with the prefactor constant depending logarithmically on the accuracy level.

In a ground-breaking paper [7], Beylkin, Coifman, and Rokhlin proposed a nonstandard form of the wavelet representation that surprisingly reduced the number of significant entries to $O(N)$. The key idea of this work is rather simple: instead of treating the matrix obtained from integral operator discretization as a linear map, view it as a two-dimensional image and hence compress it using two-dimensional wavelets. Not only drastically reducing the number of significant entries, but the resulting algorithm for matrix-vector multiplication is also significantly easier to implement and applicable to rather general pseudo-differential operators. One natural but rather unexplored question is how to extend the nonstandard wavelet form to nonlinear integral operators (see [2,44] for some related work).

[*] Corresponding author.
E-mail addresses: ywfan@stanford.edu (Y. Fan), orozcocc@stanford.edu (C. Orozco Bohorquez), lexing@stanford.edu (L. Ying).

https://doi.org/10.1016/j.jcp.2019.02.002

Recently neural networks (NNs) have experienced great successes in artificial intelligence [22,27,19,33,29,41,28,40] and even in solving PDEs [25,6,20,18,17,3,38]. One of the key reasons for these successes is that deep neural networks offer a powerful way for approximating high-dimensional functions and maps. Given a nonlinear integral operator, a fully connected NN is theoretically capable of approximating such a map [12,23,26,35]. Nonetheless, using a fully connected NN often leads to a prohibitively large number of parameters, hence long training stages and overwhelming memory footprints. To overcome these challenges, one can incorporate knowledge of the underlying structure of the problem in designing suitable network architectures. One promising and general strategy is to build NNs based on a multiscale decomposition [17,18,30,43]. The main idea, frequently used in image processing as well [4,8,9,32,39,42], is to learn increasingly coarse-grained features of a complex problem across different layers of the network structure, so that the number of parameters in each layer can be effectively controlled.

In this work, we introduce a novel NN architecture based on the nonstandard wavelet form in [7] in order to approximate certain global nonlinear operators. The paper is organized as follows. Section 2 describes the nonstandard wavelet form and the fast matrix-vector multiplication based on it. Section 3 introduces some basic NN tools and represent the matrix-vector multiplication algorithm in the form of a linear NN. Section 4 generalizes the linear NN to the nonlinear case by incorporating nonlinear activation functions and increasing the number of layers and channels of the NN. Section 5 describes implementation details and demonstrates the numerical efficiency of the NN with two applications from homogenization theory and stochastic computation. In both cases, the nonlinear mapping can be well approximated by the proposed NN, at a relative accuracy $10^{-4} \sim 10^{-3}$, with only $10^4 \sim 10^5$ parameters for 2D problems and 10^5 parameters for 3D problems.

2. Nonstandard wavelet form

In this section, we briefly summarize the nonstandard wavelet form proposed in [7], using the compactly supported orthonormal Daubechies wavelets (see [13] for details) as the basis functions.

2.1. Wavelet transform

In a one-dimensional multiresolution analysis, the starting point is a *scaling function*, or sometimes called father wavelet, $\varphi(x)$ that generates a family

$$\varphi_k^{(\ell)}(x) = 2^{\ell/2}\varphi(2^\ell x - k), \quad \ell = 0, 1, 2, \ldots, \quad k \in \mathbb{Z}, \tag{2.1}$$

such that for each fixed ℓ the functions $\{\varphi_k^{(\ell)}\}_{k\in\mathbb{Z}}$ form a Riesz basis for a space V_ℓ. The spaces $\{V_\ell\}_{\ell\geq 0}$ form a nested sequence of spaces $V_\ell \subset V_{\ell+1}$. Because of this nested condition, $\varphi(x)$ satisfies the *dilation relation*, also known as the *refinement equation*

$$\varphi(x) = \sqrt{2}\sum_{i\in\mathbb{Z}} h_i \varphi(2x - i). \tag{2.2}$$

In the case of Daubechies wavelets, $\varphi(x)$ is supported in $[0, 2p-1]$ for a positive integer p and hence the coefficients $\{h_i\}$ are nonzero only for $i = 0, \ldots, 2p-1$. From the orthonormal condition, the function $\varphi(x)$ and its integer translates satisfy an orthonormal condition

$$\int_{\mathbb{R}} \varphi(x-a)\varphi(x-b)\,\mathrm{d}x = \delta_{a,b}. \tag{2.3}$$

In terms of the coefficients h_i in (2.2), this orthonormal condition can be written as

$$\sum_{i\in\mathbb{Z}} h_i^2 = 1, \quad \sum_{i\in\mathbb{Z}} h_i h_{i+2m} = 0, \quad m \in \mathbb{Z}\backslash\{0\}, \tag{2.4}$$

following [13].

Given the scaling function $\varphi(x)$, the *mother wavelet function* $\psi(x)$ is defined as

$$\psi(x) = \sqrt{2}\sum_{i\in\mathbb{Z}} g_i \varphi(2x - i), \tag{2.5}$$

with $g_i = (-1)^{1-i}h_{1-i}$ for $i \in \mathbb{Z}$. From the support of $\varphi(x)$ and the non-zero pattern of h_i, it is clear the support of $\psi(x)$ is $[-p+1, p]$ and g_i is nonzero only for $i = -2p+2, \ldots, 1$. The *Daubechies wavelets* are defined as the scaled and shifted copies of $\psi(x)$

$$\psi_k^{(\ell)}(x) = 2^{\ell/2}\psi\left(2^\ell x - k\right), \quad \ell = 0, 1, 2, \ldots, \quad k \in \mathbb{Z}. \tag{2.6}$$

For a given function $v(x)$ defined in $\mathbb{R}$, its scaling and wavelet coefficients are given by

$$v_k^{(\ell)} := \int v(x)\varphi_k^{(\ell)}(x)\,\mathrm{d}x, \quad d_k^{(\ell)} := \int v(x)\psi_k^{(\ell)}(x)\,\mathrm{d}x, \quad \ell = 0, 1, 2, \ldots, \quad k \in \mathbb{Z}. \tag{2.7}$$

The refinement equation (2.2) implies that

$$\begin{aligned} v_k^{(\ell)} &= \int v(x)2^{\ell/2}\varphi(2^\ell x - k)\,\mathrm{d}x = \int v(x)2^{(\ell+1)/2}\sum_{i\in\mathbb{Z}} h_i\varphi(2^{\ell+1}x - 2k - i)\,\mathrm{d}x \\ &= \int v(x)\sum_{i\in\mathbb{Z}} h_i\varphi_{2k+i}^{(\ell+1)}(x)\,\mathrm{d}x = \sum_{i\in\mathbb{Z}} h_i v_{2k+i}^{(\ell+1)}, \end{aligned} \tag{2.8}$$

which is a recursive relation between the coefficients $v_k^{(\ell)}$ and $v_k^{(\ell+1)}$. A similar relation can be derived between $d_k^{(\ell)}$ and $v_k^{(\ell+1)}$:

$$d_k^{(\ell)} = \sum_{i\in\mathbb{Z}} g_i v_{2k+i}^{(\ell+1)}. \tag{2.9}$$

If $v(x) \in L^2(\mathbb{R})$, then $v^{(\ell)} := \left(v_k^{(\ell)}\right)_{k\in\mathbb{Z}}$ and $d^{(\ell)} := \left(d_k^{(\ell)}\right)_{k\in\mathbb{Z}}$ are sequences in $\ell^2(\mathbb{Z})$. In the operator form, (2.8) and (2.9) can be written as

$$v^{(\ell)} = (H^{(\ell)})^T v^{(\ell+1)}, \quad d^{(\ell)} = (G^{(\ell)})^T v^{(\ell+1)}, \tag{2.10}$$

where the operators $H^{(\ell)}, G^{(\ell)} : \ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z})$ are banded with width $2p$ due to the support size of $\{h_i\}$ and $\{g_i\}$, respectively. The orthogonality relation (2.4) can now be written compactly as

$$(H^{(\ell)})^T H^{(\ell)} = I, \quad (G^{(\ell)})^T G^{(\ell)} = I, \quad H^{(\ell)}(H^{(\ell)})^T + G^{(\ell)}(G^{(\ell)})^T = I. \tag{2.11}$$

By introducing an orthogonal operator $W^{(\ell)} = \begin{pmatrix} G^{(\ell)} & H^{(\ell)} \end{pmatrix}$, one can rewrite (2.10) as

$$\begin{pmatrix} d^{(\ell)} \\ v^{(\ell)} \end{pmatrix} = (W^{(\ell)})^T v^{(\ell+1)}, \quad v^{(\ell+1)} = W^{(\ell)} \begin{pmatrix} d^{(\ell)} \\ v^{(\ell)} \end{pmatrix}, \quad \ell = 0, 1, 2, \ldots, \tag{2.12}$$

which are the decomposition and reconstruction steps of the wavelet analysis, respectively. This process can be illustrated by the following diagram

$$\begin{tikzcd} \cdots \arrow[r] \arrow[dr] & v^{(\ell)} \arrow[r] \arrow[dr] & v^{(\ell-1)} \arrow[r] \arrow[dr] & v^{(\ell-2)} \arrow[r] \arrow[dr] & \cdots \arrow[r] \arrow[dr] & v^{(2)} \arrow[r] \arrow[dr] & v^{(1)} \arrow[r] \arrow[dr] & v^{(0)} \\ & d^{(\ell)} & d^{(\ell-1)} & d^{(\ell-2)} & \cdots & d^{(2)} & d^{(1)} & d^{(0)} \end{tikzcd}. \tag{2.13}$$

Although, until this point our discussion is concerned with wavelets defined on $\mathbb{R}$, the definitions can be easily extended to the case of periodic functions defined on a finite interval, say $[0, 1]$. The only modification is that all the shifts and scalings in the x variable are now done modulus the integers. In addition, instead of going all the way to level 0, the above multiresolution analysis stops at a coarse level $L_0 = O(\log_2 p)$ before the wavelet and scaling functions start to overlap itself, which is diagramed as

$$\begin{tikzcd} \cdots \arrow[r] \arrow[dr] & v^{(\ell)} \arrow[r] \arrow[dr] & v^{(\ell-1)} \arrow[r] \arrow[dr] & v^{(\ell-2)} \arrow[r] \arrow[dr] & \cdots \arrow[r] \arrow[dr] & v^{(L_0)} \\ & d^{(\ell)} & d^{(\ell-1)} & d^{(\ell-2)} & \cdots & d^{(L_0)} \end{tikzcd}. \tag{2.14}$$

Finally, the Daubechies wavelet $\psi(x)$ described above has exactly p vanishing moments [13], i.e.,

$$\int x^j \psi(x)\,\mathrm{d}x = 0, \quad j = 0, \ldots, p-1. \tag{2.15}$$

Since the space of polynomials up to degree $p-1$ is invariant under scaling and translation, (2.15) is true also for any wavelet $\psi_k^{(\ell)}(x)$.

2.2. Integral operator compression

The nonstandard form is concerned with wavelet based compression of integral operators. Suppose that A is an integral operator on the periodic interval $[0,1]$ with kernel $a(x,y)$. We consider the Galerkin projection of A in the space V_L for a sufficient deep level L. The $2^L \times 2^L$ matrix $A^{(L)} = (A^{(L)}_{k_1,k_2})_{k_1,k_2=0,\dots,2^L-1}$ with entries given by

$$A^{(L)}_{k_1,k_2} := \iint \varphi^{(L)}_{k_1}(x) a(x,y) \varphi^{(L)}_{k_2}(y)\,\mathrm{d}x\,\mathrm{d}y$$

is a reasonably accurate approximation of the operator A. The nonstandard form is essentially a data-sparse representation of $A^{(L)}$ using the 2D multiresolution wavelet basis. Let us now introduce

$$\begin{aligned} D^{(\ell)}_{1,k_1,k_2} &:= \iint \psi^{(\ell)}_{k_1}(x) a(x,y) \psi^{(\ell)}_{k_2}(y)\,\mathrm{d}x\,\mathrm{d}y, & D^{(\ell)}_{2,k_1,k_2} &:= \iint \psi^{(\ell)}_{k_1}(x) a(x,y) \varphi^{(\ell)}_{k_2}(y)\,\mathrm{d}x\,\mathrm{d}y, \\ D^{(\ell)}_{3,k_1,k_2} &:= \iint \varphi^{(\ell)}_{k_1}(x) a(x,y) \psi^{(\ell)}_{k_2}(y)\,\mathrm{d}x\,\mathrm{d}y, & A^{(\ell)}_{k_1,k_2} &:= \iint \varphi^{(\ell)}_{k_1}(x) a(x,y) \varphi^{(\ell)}_{k_2}(y)\,\mathrm{d}x\,\mathrm{d}y, \end{aligned} \tag{2.16}$$

for $\ell = L_0, \dots, L-1$, and $k_1, k_2 = 0, \dots, 2^\ell - 1$. In what follows, it is convenient to organize these coefficients into the following matrices:

$$A^{(\ell)} = (A^{(\ell)}_{k_1,k_2})_{k_1,k_2=0,\dots,2^\ell-1}, \qquad D^{(\ell)}_j = (D^{(\ell)}_{j,k_1,k_2})_{k_1,k_2=0,\dots,2^\ell-1}, \; j = 1,2,3. \tag{2.17}$$

Using (2.12), these matrices can be computed from level L down to level L_0 via the recursive relation

$$\begin{pmatrix} D^{(\ell)}_1 & D^{(\ell)}_2 \\ D^{(\ell)}_3 & A^{(\ell)} \end{pmatrix} = (W^{(\ell)})^T A^{(\ell+1)} W^{(\ell)}. \tag{2.18}$$

A key feature of (2.18) is that the entries of the matrices $D^{(\ell)}_j (j = 1,2,3)$ decay rapidly away from the diagonal if the kernel function $a(x,y)$ satisfies certain smoothness conditions. More precisely, if A is a Calderon-Zygmund operator with kernel $a(x,y)$ that satisfies

$$|a(x,y)| \lesssim \frac{1}{|x-y|}, \quad |\partial^p_x a(x,y)| + |\partial^p_y a(x,y)| \lesssim_p \frac{1}{|x-y|^{1+p}}, \tag{2.19}$$

then the entries of $D^{(\ell)}_j$ satisfy the following estimate [7, Proposition 4.1]

$$|D^{(\ell)}_{j,k_1,k_2}| \lesssim_p \frac{C_p}{1 + |k_1 - k_2|^{p+1}}, \quad \text{for all } j \text{ and } |k_1 - k_2| \geq 2p. \tag{2.20}$$

Similarly, suppose that A is a pseudo-differential operator with symbol $\sigma(x,\xi)$

$$(Av)(x) = \int e^{\mathrm{i}x\xi} \sigma(x,\xi) \hat{v}(\xi)\,\mathrm{d}\xi = \int a(x,y) v(y)\,\mathrm{d}y. \tag{2.21}$$

If the symbols σ of A and σ^* of A^* satisfy the conditions

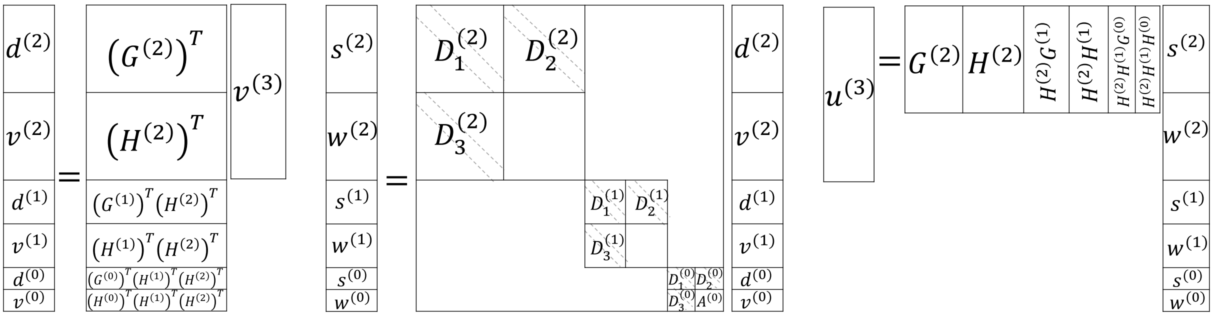
$$|\partial^{m_1}_\xi \partial^{m_2}_x \sigma(x,\xi)| \lesssim_{m_1,m_2} (1+|\xi|)^{\lambda-m_1+m_2}, \quad |\partial^{m_1}_\xi \partial^{m_2}_x \sigma^*(x,\xi)| \lesssim_{m_1,m_2} (1+|\xi|)^{\lambda-m_1+m_2}, \tag{2.22}$$

then [7, page 155]

$$\frac{1}{2^{\lambda\ell}} |D^{(\ell)}_{j,k_1,k_2}| \lesssim_p \frac{1}{(1+|k_1-k_2|)^{p+1}}, \quad \text{for all } j \text{ and } k_1, k_2. \tag{2.23}$$

Because of the rapid decay of the entries of $D^{(\ell)}_{j,k_1,k_2}$, one can approximate these matrices by truncating at a band of width $n_b = O(\log(1/\varepsilon))$ for a prescribed *relative accuracy* ε. Note that, the value of n_b is independent of the specific choices of $\ell = L_0, \dots, L-1$, $j = 1,2,3$, or the mesh size N. From now on, we assume that the matrices $D^{(\ell)}_{j,k_1,k_2}$ are pre-truncated already. The number of non-zero entries of these matrices at level ℓ is clearly $O(2^\ell)$.

Concerning the matrix $A^{(\ell)}$ for $\ell = L_0, \dots, L-1$, though they are dense in general, the recursive relation (2.17) shows that one only needs to store the matrix $A^{(L_0)}$ at the top level L_0. Since there are only $O(p)$ wavelets and scaling functions at level L_0, $A^{(L_0)}$ is of constant size. Combining this with the estimate for matrices $D^{(\ell)}_{j,k_1,k_2}$ over all levels, one concludes that the total number of non-zero coefficients in the nonstandard form is $O(N)$. This is the *nonstandard form* proposed by Beylkin, Coifman, and Rokhlin.



(a) wavelet transform (b) matrix-vector multiplication (c) inverse wavelet transform

Fig. 1. Illustration of matrix-vector multiplication based on the nonstandard form (with $L_0 = 0$ and $L = 3$). $D_j^{(\ell)}$, $j = 1, 2, 3$ are all band matrices.

2.3. Matrix-vector multiplication

For an integral operator A, a fundamental operation is to apply A to an arbitrary function v, i.e.,

$$u = Av, \quad u(x) = \int_\Omega a(x, y)v(y)\,\mathrm{d}y, \quad \Omega = [0, 1], \tag{2.24}$$

where v and u are periodic functions defined on Ω. With the Galerkin discretization described above at level L, this is simply a matrix vector multiplication.

$$u^{(L)} = A^{(L)}v^{(L)}. \tag{2.25}$$

This matrix-vector multiplication is computationally intensive since $A^{(\ell)}$ is a dense matrix. However, the nonstandard form introduced above offers an $O(N)$ (linear complexity) algorithm for carrying out the matrix vector multiplication. The key observation of this process is the following recurrence relation

$$A^{(\ell+1)} = W^{(\ell)} \begin{pmatrix} D_1^{(\ell)} & D_2^{(\ell)} \\ D_3^{(\ell)} & A^{(\ell)} \end{pmatrix} (W^{(\ell)})^T, \quad \ell = 0, \ldots, L-1, \tag{2.26}$$

obtained from transposing (2.18). This means that, modulus wavelet transforms, at each level ℓ, one only needs to perform matrix-vector multiplication with the sparse matrices $D_{j,k_1,k_2}^{(\ell)}$. The dense matrix multiplication is only carried out with $A^{(L_0)}$ at the top level L_0. More precisely, the algorithm follows the three steps illustrated in Fig. 1:

(a) Transforming $v^{(L)}$ to obtain its scaling and wavelet coefficients of the nonstandard form;
(b) Multiplying with banded matrices at each scale;
(c) Transforming the scaling and wavelet coefficients of the nonstandard form to obtain $u^{(L)}$.

By following (2.12), (2.18), (2.26) and introducing auxiliary vectors $u^{(\ell)}$ at all levels, this calculation can be written more compactly as

$$u^{(\ell+1)} = A^{(\ell+1)}v^{(\ell+1)} = W^{(\ell)} \begin{pmatrix} D_1^{(\ell)} & D_2^{(\ell)} \\ D_3^{(\ell)} & A^{(\ell)} \end{pmatrix} \begin{pmatrix} d^{(\ell)} \\ v^{(\ell)} \end{pmatrix} = W^{(\ell)} \left[\begin{pmatrix} D_1^{(\ell)} & D_2^{(\ell)} \\ D_3^{(\ell)} & D_4^{(\ell)} \end{pmatrix} \begin{pmatrix} d^{(\ell)} \\ v^{(\ell)} \end{pmatrix} + \begin{pmatrix} 0 \\ u^{(\ell)} \end{pmatrix} \right], \tag{2.27}$$

for $\ell = L-1, L-2, \ldots, L_0$ with $D_4^{(\ell)}$ defined to be the zero matrix. Algorithm 1 provides the pseudo code for computing (2.27). Fig. 2 illustrates the multiresolution structure of the computation, which will turn out to be useful when we introduce the NN. Since the matrices $D_j^{(\ell)}$ are band matrices, the arithmetic complexity of evaluating $u^{(\ell+1)}$ given $u^{(\ell)}$ is $O(n_b N/2^\ell)$, where $N = 2^L$. Therefore, the overall complexity of (2.25) is $O(N)$.

2.4. Multidimensional case

The above discussion can be easily extended to higher dimensions by using the multidimensional orthonormal wavelets. The discussion here will focus on the two-dimensional case. For the two-dimensional wavelet analysis (see for example [36, Chapter 8] and [34]), there are three different types of wavelets at each scale.

Algorithm 1 $u^{(L)} = A^{(L)} v^{(L)}$ in the nonstandard form.

```
for $\ell$ from $L-1$ to $L_0$ by $-1$ do
    $\begin{pmatrix} d^{(\ell)} \\ v^{(\ell)} \end{pmatrix} = (W^{(\ell)})^T v^{(\ell+1)}$;
end for
$u^{(L_0)} = A^{(L_0)} v^{(L_0)}$;
for $\ell$ from $L_0$ to $L-1$ do
    $\begin{pmatrix} s^{(\ell)} \\ w^{(\ell)} \end{pmatrix} = \begin{pmatrix} D_1^{(\ell)} & D_2^{(\ell)} \\ D_3^{(\ell)} & D_4^{(\ell)} \end{pmatrix} \begin{pmatrix} d^{(\ell)} \\ v^{(\ell)} \end{pmatrix}$;
    $u^{(\ell+1)} = W^{(\ell)} \begin{pmatrix} s^{(\ell)} \\ w^{(\ell)} + u^{(\ell)} \end{pmatrix}$;
end for
```

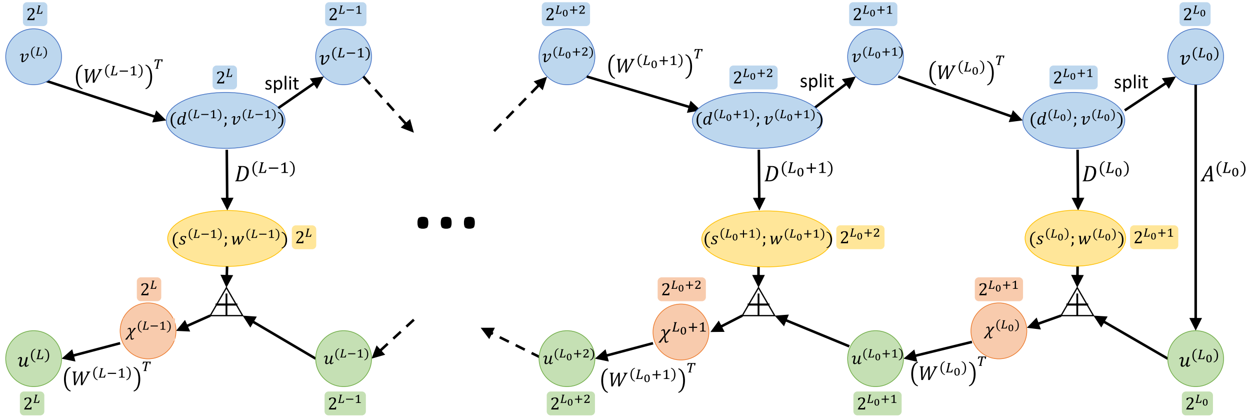


Fig. 2. Diagram of the matrix-vector multiplication in the nonstandard form. The notation ⨹ denotes the summation $\chi^{(\ell)} = \begin{pmatrix} s^{(\ell)} \\ w^{(\ell)} \end{pmatrix} + \begin{pmatrix} 0 \\ u^{(\ell)} \end{pmatrix}$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

The 2D analog of the recursive relation (2.26) takes the form

$$\begin{pmatrix} D_1^{(\ell)} & D_2^{(\ell)} & D_3^{(\ell)} & D_4^{(\ell)} \\ D_5^{(\ell)} & D_6^{(\ell)} & D_7^{(\ell)} & D_8^{(\ell)} \\ D_9^{(\ell)} & D_{10}^{(\ell)} & D_{11}^{(\ell)} & D_{12}^{(\ell)} \\ D_{13}^{(\ell)} & D_{14}^{(\ell)} & D_{15}^{(\ell)} & A^{(\ell)} \end{pmatrix} = (W^{(\ell)})^T A^{(\ell+1)} W^{(\ell)} \tag{2.28}$$

and all $D_j^{(\ell)}$ matrices are sparse with only $O(2^\ell)$ non-negligible entries after thresholding. The matrix-vector multiplication takes exactly the same form as Algorithm 1 except obvious changes due to matrix sizes.

3. Matrix-vector multiplication as a neural network

The goal of this section is to represent the matrix-vector multiplication in Algorithm 1 as a linear neural network. We start by introducing several basic tools in Section 3.1 and then present the neural network representation of Algorithm 1 in Section 3.2. The presentation will be focused on the 1D case first in order to illustrate the main ideas clearly.

3.1. Neural network tools

Throughout the discussion below, the input, output, and intermediate data are all represented with 2-tensor. For a 2-tensor of size $N_x \times \alpha$, we call N_x the *spatial dimension* and α the *channel dimension*.

To perform the operations appeared in Algorithm 1, we first introduce a few common NN layers. The first one is the well-known *convolutional layer* (Conv) where the output of each location depends only locally on the input. Given an input tensor ξ of size $N_{\text{in}} \times \alpha_{\text{in}}$ and an output tensor ζ of size $N_{\text{out}} \times \alpha_{\text{out}}$, the convolutional layer performs the computation

$$\zeta_{i,c'} = \phi\left(\sum_{j=is}^{is+w-1} \sum_{c=0}^{\alpha_{\text{in}}-1} W_{j;c',c} \xi_{j,c} + b_{c'} \right), \quad i = 0, \dots, N_{\text{out}} - 1,\ c' = 0, \dots, \alpha_{\text{out}} - 1, \tag{3.1}$$

where ϕ is a pre-specified function, called *activation*, usually chosen to be a linear function, a rectified-linear unit (ReLU) function, or a sigmoid function. The parameters w and s are called the *kernel window size* and *stride*, respectively. Here we assume that $N_{\text{out}} = N_{\text{in}}/s$ and the tensor ξ is periodically padded if the index is out of range. Note that this differs from the definition of the convolution layer in TensorFlow [1] for which zero padding is the default behavior. In what follows, such a convolution layer is denoted as

$$\zeta = \mathsf{Conv}[\alpha_{\text{out}}, w, s, \phi](\xi), \tag{3.2}$$

where the values of N_{in}, α_{in}, and N_{out} are inferred from the input tensor ξ.

Note that the weight $W_{j;c',c}$ in (3.1) is independent on the position i of ζ, thus Conv is translation invariant. When the weight are required to depend on the position i (i.e. $W_{i,j;c',c}$), the natural extension of Conv is the so-called *locally connected* (LC) layer. This layer is denoted by

$$\zeta = \mathsf{LC}[\alpha_{\text{out}}, w, s, \phi](\xi).$$

Finally, when the output data tensor depends on every entry of the input tensor, this is called a *dense* layer, denoted by

$$\zeta = \mathsf{Dense}[\alpha_{\text{out}}, \phi](\xi).$$

Here we assume implicitly that the spatial dimensions of the input and output tensors are same, i.e., $N_{\text{out}} = N_{\text{in}}$.

With these basic tools, we can implement the key steps of Algorithm 1 in the NN framework:

- Multiply $(W^{(\ell)})^T$ with a vector: this step takes the form

  $$\zeta = \mathsf{Conv}[2, 2p, 2, \mathsf{id}](\xi), \tag{3.3}$$

  where id is the identity operator. The size of ζ is $M/2 \times 2$ if the size of ξ is $M \times 1$. The convention adopted is that the first channel is for the wavelet coefficients and the second channel is for the scaling function coefficients.
- Multiply $\begin{pmatrix} D_1^{(\ell)} & D_2^{(\ell)} \\ D_3^{(\ell)} & D_4^{(\ell)} \end{pmatrix}$ with a vector: this step takes the form

  $$\zeta = \mathsf{LC}[2, n_b, 1, \mathsf{id}](\xi). \tag{3.4}$$

  The size of ζ is $M \times 2$ if the size of ξ is $M \times 2$. Notice that the width of LC layer corresponds to the band width of the banded matrices.
- Multiply $A^{(L_0)}$ with a vector: this step takes the form

  $$\zeta = \mathsf{Dense}[1, \mathsf{id}](\xi).$$

  Both ζ and ξ are of size $2^{Lc} \times 1$.
- Multiply $W^{(\ell)}$ with a vector: this step first computes

  $$\zeta = \mathsf{Conv}[2, p, 1, \mathsf{id}](\xi), \tag{3.5}$$

  followed by a reshape that goes through the channel dimension first. The output is of size $2M \times 1$ if the input is $M \times 2$.

In all these NN operations, the bias b is set to be zero.

3.2. Neural network representation

Combining the basic tools introduced above, we can translate Algorithm 1 into an NN. The resulting algorithm is summarized in Algorithm 2. Fig. 3 illustrates the multiresolution structure of the NN. It has the same structure as Fig. 2 except the basic operations are replaced with the NN layers introduced in Section 3.1.

Algorithm 2 NN architecture for $u^{(L)} = A^{(L)} v^{(L)}$ in the nonstandard form.

```
for $\ell$ from $L-1$ to $L_0$ by $-1$ do
    $\xi^{(\ell)} = \mathsf{Conv}[2, 2p, 2, \mathsf{id}](v^{(\ell+1)})$;
    $v^{(\ell)}$ is the last channel of $\xi^{(\ell)}$;
end for
$u^{(L_0)} = \mathsf{Dense}[1, \mathsf{id}](v^{(L_0)})$;
for $\ell$ from $L_0$ to $L-1$ do
    $\zeta^{(\ell)} = \mathsf{LC}[2, n_b, 1, \mathsf{id}](\xi^{(\ell)})$;
    Adding $u^{(\ell)}$ to the last channel of $\zeta^{(\ell)}$ gives $\chi^{(\ell)}$;
    $u^{(\ell+1)} = \mathsf{Conv}[2, p, 1, \mathsf{id}](\chi^{(\ell)})$;
    Reshape $u^{(\ell+1)}$ to a 2-tensor of size $2^{\ell+1} \times 1$ by following channel dimension first;
end for
```

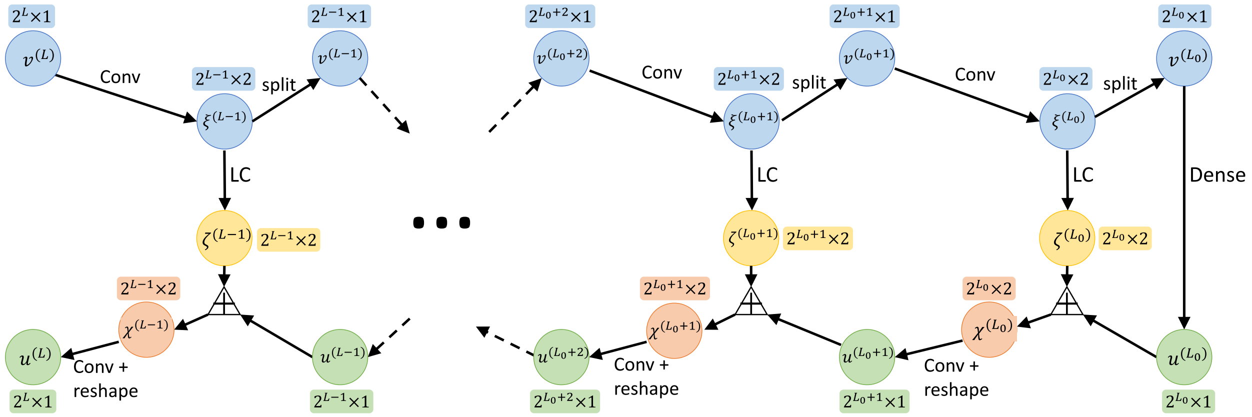


Fig. 3. Neural network architecture of the matrix-vector multiplication in the nonstandard form. "split" means extracting the last channel of $\xi^{(\ell)}$ to obtain $v^{(\ell)}$. ⨹ means adding $u^{(\ell)}$ to the last channel of $\zeta^{(\ell)}$ to obtain $\chi^{(\ell)}$.

Let us now count the number of parameters used in the network in Algorithm 2. Notice that the number of parameters in Conv and LC are $w\alpha_{\text{in}}\alpha_{\text{out}}$ and $N_{\text{out}}w\alpha_{\text{in}}\alpha_{\text{out}}$, respectively. The total number of parameters in Algorithm 2 is

$$N_{\text{params}} = \sum_{\ell=L_0}^{L-1}\left(4p + 2^{\ell}4n_b + 4p\right) + 4^{L_0} \approx 4Nn_b + 8p(L-L_0) + 4^{L_0}. \tag{3.6}$$

Since 2^{L_0} is a small constant, the total number of parameters is $O(Nn_b)$.

3.3. Multidimensional case

Our discussion here focuses on the 2D case. Each piece of data in the 2D algorithm can be represented by a 3-tensor of size $N_{x,1}\times N_{x,2}\times\alpha$, where $N_x=(N_{x,1},N_{x,2})$ is the size in the spatial dimension and α is the channel number. If a tensor ξ of size $N_{\text{in},1}\times N_{\text{in},2}\times\alpha_{\text{in}}$ is connected to a tensor ζ of size $N_{\text{out},1}\times N_{\text{out},2}\times\alpha_{\text{out}}$ by a convolution layer, the computation takes the form

$$\zeta_{i,c'} = \phi\left(\sum_{j_1=i_1 s}^{i_1 s+w-1}\sum_{j_2=i_2 s}^{i_2 s+w-1}\sum_{c=0}^{\alpha_{\text{in}}-1} W_{j;c',c}\xi_{j,c} + b_{c'}\right), \quad i_1=0,\dots,N_{\text{out},1}-1, i_2=0,\dots,N_{\text{out},2}-1, c'=0,\dots,\alpha_{\text{out}}-1. \tag{3.7}$$

By denoting such a layer by Conv2, we can write (3.7) concisely as Conv2$[\alpha,w,s,\phi](\xi)$. Similar to the 1D case, we also define the locally connected layer, denoted by LC2$[\alpha,w,s,\phi](\xi)$ and the dense layer, denoted by Dense2$[\alpha,\phi](\xi)$.

With these tools, one can readily extend the Algorithm 2 to the 2D case, shown in Algorithm 3.

Algorithm 3 NN architecture for $u^{(L)}=A^{(L)}v^{(L)}$ in the nonstandard form for the 2D case.

```
for $\ell$ from $L-1$ to $L_0$ by $-1$ do
    $\xi^{(\ell)} = \text{Conv2}[4, 2p, 2, \text{id}](v^{(\ell+1)})$;
    $v^{(\ell)}$ is the last channel of $\xi^{(\ell)}$;
end for
$u^{(L_0)} = \text{Dense2}[1, \text{id}](v^{(L_0)})$;
for $\ell$ from $L_0$ to $L-1$ do
    $\zeta^{(\ell)} = \text{LC2}[4, n_b, 1, \text{id}](\xi^{(\ell)})$;
    Adding $u^{(\ell)}$ to the last channel of $\zeta^{(\ell)}$ gives $\chi^{(\ell)}$;
    $u^{(\ell+1)} = \text{Conv2}[4, p, 1, \text{id}](\chi^{(\ell)})$;
    Reshape $u^{(\ell+1)}$ to a 3-tensor of size $2^{(\ell+1)}\times 2^{(\ell+1)}\times 1$;
end for
```

The reshape step at the end of Algorithm 3 deserves some comments. The input tensor $u^{(\ell+1)}$ is of size $2^{\ell}\times2^{\ell}\times4$. The reshape process first change the input to a $2^{\ell}\times2^{\ell}\times2\times2$ tensor by splitting the last dimension. It then permutes the second and their third dimension to obtain a 4-tensor of size $2^{\ell}\times2\times2^{\ell}\times2$. By grouping the first and the second dimensions as well as the third and the fourth dimension, one obtains a $2^{\ell+1}\times2^{\ell+1}$ tensor. Finally, this tensor is regarded as a 3-tensor of size $2^{\ell+1}\times2^{\ell+1}\times1$, i.e., with a single component in the channel (last) dimension.

4. BCR-Net

For many nonlinear map of form

$$u = \mathcal{M}(v), \quad u, v \in \mathbb{R}^{N^d}, \tag{4.1}$$

when the singularity of u only appears at the singularity of v, such a operator can be viewed as a nonlinear generalization of pseudo-differential operators. By leveraging the representation power of the neural networks, we extend the architecture constructed in Algorithm 2 to represent the nonlinear maps (4.1). The resulting NN architecture is referred to as the BCR-Net. In order to simplify the presentation, we will focus on the 1D case in this section as all the results here can be easily extended to the multi-dimensional case by following the discussion in Section 3.3.

Two changes are made in order to extend Algorithm 2 to the nonlinear case. The first is to replace some of the identity activation functions id with the nonlinear activation functions. More precisely, the activation function in LC and Dense is replaced with a nonlinear one (either being ReLU or Sigmoid function), denoted by ϕ. On the other hand, since each Conv layer corresponds to a step of the wavelet transform, its activation function is kept as id.

The second modification is to increase the "width" and "depth" of the network. The NN in Algorithm 2 is rather narrow (the number of channels in LC is only 2) and shallow (the number of LC layers is only 1). As a result, its representation power is limited. In order to represent more general nonlinear maps, we increase the number of channels from 2 to 2α, with α wavelet channels and α scaling function channels. Here α is a user-specified parameter. Moreover, the network also becomes deeper by increasing number of LC and Dense layers. The resulting algorithm is summarized in Algorithm 4 and illustrated in Fig. 4.

Algorithm 4 BCR-Net applied to an input $v \in \mathbb{R}^N$ with $N = 2^L$.

```
$v^{(L)} = v$;
for $\ell$ from $L-1$ to $L_0$ by $-1$ do
    $\xi^{(\ell)} = \mathrm{Conv}[2\alpha, 2p, 2, \mathrm{id}](v^{(\ell+1)})$;
    $v^{(\ell)}$ is the last $\alpha$ channels of $\xi^{(\ell)}$;
end for
$u_0^{(L_0)} = v^{(L_0)}$;
for $k$ from 1 to $K$ do do
    $u_k^{(L_0)} = \mathrm{Dense}[\alpha, \phi](u_{k-1}^{(L_0)})$;
end for
$u^{(L_0)} = u_K^{(L_0)}$;
for $\ell$ from $L_0$ to $L-1$ do
    $\zeta_0^{(\ell)} = \xi^{(\ell)}$;
    for $k$ from 1 to $K$ do
        $\zeta_k^{(\ell)} = \mathrm{LC}[2\alpha, n_b, 1, \phi](\zeta_{k-1}^{(\ell)})$;
    end for
    Adding $u^{(\ell)}$ to the last $\alpha$ channels of $\zeta_K^{(\ell)}$ gives $\chi^{(\ell)}$;
    $u^{(\ell+1)} = \mathrm{Conv}[2\alpha, p, 1, \mathrm{id}](\chi^{(\ell)})$;
    Reshape $u^{(\ell+1)}$ to a 2-tensor of size $2^{\ell+1} \times \alpha$ by following channel dimension first;
end for
Average over the channel direction of $u^{(L)}$ to give $u$;
```

Similar to the linear case, the number of parameters of BCR-Net can be estimated as follows:

$$\begin{aligned} N_{\text{params}} &= \sum_{\ell=L_0}^{L-1} \left(4\alpha^2 p + \sum_{k=1}^{K} (2\alpha)^2 n_b 2^\ell + 4\alpha^2 p \right) + \sum_{k=1}^{K} 4^{L_0}\alpha^2 \\ &\approx 4\alpha^2 n_b K N + 8\alpha^2 p(L - L_0) + 4^{L_0} K\alpha^2. \end{aligned} \tag{4.2}$$

As L_0 is small, thus the total number of parameters is $O(n_b\alpha^2 K N)$.

Translation-equivariant case For the linear system (2.24), if the kernel is of convolution type, i.e., $a(x, y) = a(x - y)$, then A is a cyclic matrix. So are the matrices $D_j^{(\ell)}$ and $A^{(L)}$. Therefore, the LC layer in Algorithm 1 can be replaced with a Conv layer.

In the nonlinear case, the operator A is *translation-equivariant* if

$$\mathcal{T}A(v) = A(\mathcal{T}v) \tag{4.3}$$

holds for any translation operator $\mathcal{T}$. In this case, each LC layer in Algorithm 4 is replaced with a Conv layer with the same window size, while each Dense layer is replaced with a Conv layer with window size equal to the input size. Since the number of parameters of a Conv layer is $\alpha_{\text{out}}\alpha_{\text{in}}w$, the number of parameters of Algorithm 4 is

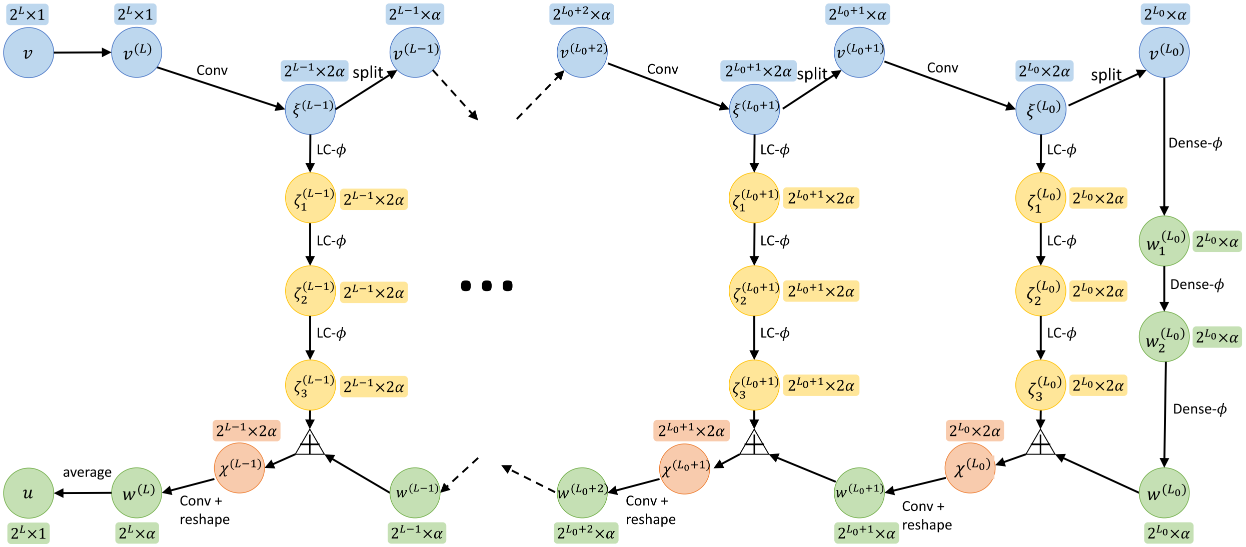


Fig. 4. Architecture of BCR-Net. "split" means extracting the last α channels of $\xi^{(\ell)}$ to obtain $v^{(\ell)}$. △ means adding $u^{(\ell)}$ to the last α channels of $\zeta^{(\ell)}$ to obtain $\chi^{(\ell)}$.

$$
\begin{aligned}
N_{\text{params}} &= \sum_{\ell=L_0}^{L-1}\left(4\alpha^2 p + \sum_{k=1}^{K}(2\alpha)^2 n_b + 4\alpha^2 p\right) + \sum_{k=1}^{K} 4^{L_0}\alpha^2 \\
&\approx 4\alpha^2 n_b K(L-L_0) + 8\alpha^2 p(L-L_0) + 4^{L_0}K\alpha^2 \\
&\approx O\left(n_b\alpha^2 K\log(N)\right),
\end{aligned}
\tag{4.4}
$$

which is only logarithmic in N.

5. Applications

We implement BCR-Net with Keras [10] (running on top of TensorFlow [1]) using Nadam as the optimizer [14] and the mean squared error as the loss function. The parameters in BCR-Net are initialized randomly from the normal distribution, and the batch size is always set as two percent of the size of the training set.

In this section, we study the performance of BCR-Net using a few examples. In the experiments, the support of the scaling function $\varphi(x)$ is chosen to be $[0, 2p-1]$ with $p=3$. The activation function in wavelet transform is set to be the identity as we mentioned, while ReLU is used in the LC and Dense layers. As discussed in Section 4, when the operator is translation-equivariant, the LC layers and Dense layers in Algorithm 4 are replaced by Conv and fully connected Conv layers, respectively. The selection of parameters α (number of channels) and K (number of LC layers in Algorithm 4) are problem dependent.

For each sample, let u be the *exact* solution generated by numerical discretization of PDEs and u_{NN} be the prediction from BCR-Net. Then the error of this specific sample is calculated by the relative error measured in the ℓ^2 norm:

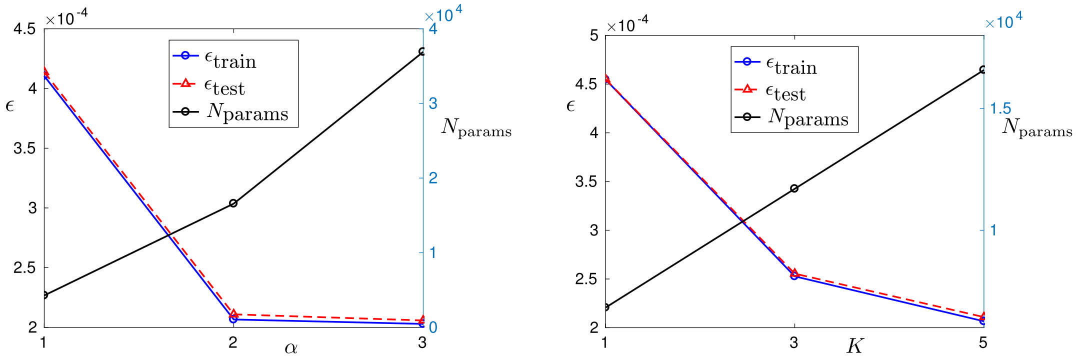
$$
\epsilon = \frac{\|u-u_{\text{NN}}\|_{\ell^2}}{\|u\|_{\ell^2}}. \tag{5.1}
$$

The training error ϵ_{train} and test error ϵ_{test} are then obtained by averaging (5.1) over a given set of training or testing samples, respectively. The numerical results presented in this section are obtained by repeating the training process five times, using different random seeds.

5.1. Diagonal of the inverse matrix of elliptic operator

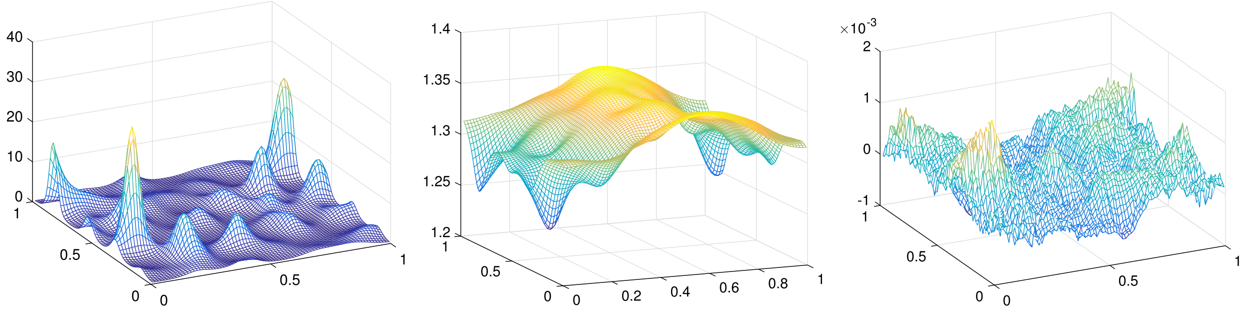
Elliptic operators of form $H = -\Delta + v(x)$ appear in many mathematical models. In quantum mechanics, the Hamiltonian of the Schrödinger equation takes this form and it is fundamental in describing the dynamics of quantum particles. In probability theory, this operator describes for example the behavior of a random walk particle that interacts with the environment.

Here, we are interested the Green function $G = H^{-1} = (-\Delta + v(x))^{-1}$ and, more specifically, the dependence of its diagonal $G(x,x)$ on the potential $v(x)$. The diagonal of the Green's function plays an important role in several applica-



(a) $K = 5$ (b) $\alpha = 2$

Fig. 5. The relative error and the number of parameters of BCR-Net for (5.2) with $N^{\text{train}}_{\text{samples}} = N^{\text{test}}_{\text{samples}} = 20000$. (a) Different channel numbers (α). (b) Different LC layer numbers (K).



(a) $v(x)$ (b) $g_{\text{NN}}(x)$ (c) $g(x) - g_{\text{NN}}(x)$

Fig. 6. A random sample potential $v(x)$, the prediction $g_{\text{NN}}(x)$ from BCR-Net with $\alpha = 2$ and $K = 5$, and the error from the reference solution of (5.2).

tions. For example, in density function theory, when combined with appropriate rational expansions, the diagonal of the (shifted) Green's function allows one to compute the Kohn-Sham map efficiently [31]. As another example, for random walk particles that decay at a spatially dependent rate given by $v(x) > 0$, the diagonal $G(x, x)$ gives the expected local time (or the number of visits in the discrete setting) at position x for a particle starting from x, via the Feynman-Kac formula [16].

In the following numerical studies, we work with the operator $H = -\Delta + v(x)$ on $[0, 1]^2$ in 2D with the periodic boundary condition. The differential operator is discretized using the standard five-point central difference with 80 points per dimension. The potential v is generated by randomly sampling on a 10×10 grid independently from the standard Gaussian distribution $\mathcal{N}(0, 1)$, interpolating it to the 80×80 grid via Fourier interpolation, and finally performing a pointwise exponentiation.

By denoting the diagonal of $G = H^{-1}$ by $g(x)$, we apply BCR-Net to learn the nonlinear map from the potential $v(x)$ to $g(x)$:

$$v(x) \to g(x) := G(x, x). \tag{5.2}$$

The approximation to $g(x)$ produced by BCR-Net will be denoted by $g_{\text{NN}}(x)$.

Fig. 5 presents the numerical results with different choices for α (channel number) and K (layer number) in Algorithm 4. The band of the $D^{(\ell)}_j$ matrices is set to be $n_b = 3$. When α or K increases, the error decreases consistently. Fig. 5a plots the results for different α values with K fixed. Note that $\alpha = 2$ already achieves a fairly accurate result. Fig. 5b shows the results for different K values with α fixed, showing that deeper networks clearly give better results. In each case, the test error is close to the training error, implying that there is no over-fitting in our model. As discussed in Section 4, the number of parameters is proportional to $\alpha^2 K$, which agrees with the curves in Fig. 5. In Fig. 6, we plot a sample of the potential $v(x)$, along with the prediction from BCR-Net and its error in comparison with the reference solution ($\alpha = 2$ and $K = 5$).

It is worth pointing out that the test error is quite small (around 2.1×10^{-4} with $\alpha = 2$ and $K = 5$), while the number of parameters is $N_{\text{params}} = 1.6 \times 10^4$. Comparing with millions of parameters in the applications on images [27,39], BCR-Net only uses tens of thousands of parameters.

5.2. Nonlinear homogenization theory

The second example is concerned with homogenization theory, which studies effective models for differential and integral equations with oscillatory coefficients. In the simplest setting, consider the linear second order elliptic PDE in a domain Ω

$$-\nabla \cdot \left(a\left(\frac{x}{\varepsilon}\right) \nabla u^{\varepsilon}(x) \right) = 0$$

with appropriate boundary conditions on $\partial\Omega$, where $a(\cdot)$ is a periodic function on the unit cube $[0,1]^d$. The homogenization theory [5,15,24,37] states that, when ε goes to zero, the solution $u^{\varepsilon}(x)$ exhibits a multiscale decomposition

$$u^{\varepsilon}(x) = u_0(x) + \varepsilon u_1(x) + \varepsilon^2 u_2(x) + \cdots .$$

Here $u_0(x)$ is the solution of a constant elliptic PDE

$$-\nabla \cdot (A_0 \nabla u_0(x)) = 0$$

where the *constant* matrix A_0 is called the *effective coefficient tensor*. The next term $u_1(x)$ that depends on the gradient of $u_0(x)$ can be written as

$$u_1(x) = \sum_{i=1}^{d} \eta_i \left(\frac{x}{\varepsilon}\right) \nabla_i u_0(x)$$

in terms of the so-called *corrector functions* $\{\eta_i(x)\}_{i=1,\ldots,d}$. For each $i \in \{1,\ldots,d\}$, the corrector function $\eta_i(x)$ is the solution of the periodic problem

$$-\nabla \cdot (a(x)(\nabla \eta_i(x) + e_i)) = 0, \qquad \int\limits_{[0,1]^d} \eta_i(x)\, \mathrm{d}x = 0, \tag{5.3}$$

where e_i is the canonical basis vector in the i-th coordinate. Both the effective coefficient tensor A_0 and the correctors $\eta_i(x)$ are important for studying multiscale problems in engineering applications. Note from (5.3) that computing the corrector $\eta_i(x)$ for each i requires a single PDE solve over the periodic cube.

For many nonlinear problems, a similar homogenization theory holds. Consider for example the variational problem

$$u_{\varepsilon} = \operatorname{argmin}_v \int\limits_{\Omega} f\left(\frac{x}{\varepsilon}, \nabla v\right) \mathrm{d}x$$

with appropriate boundary conditions on $\partial\Omega$, where $f\left(\frac{x}{\varepsilon}, \nabla v\right)$ is given by

$$f\left(\frac{x}{\varepsilon}, \nabla u\right) = a\left(\frac{x}{\varepsilon}\right) |\nabla u|^p$$

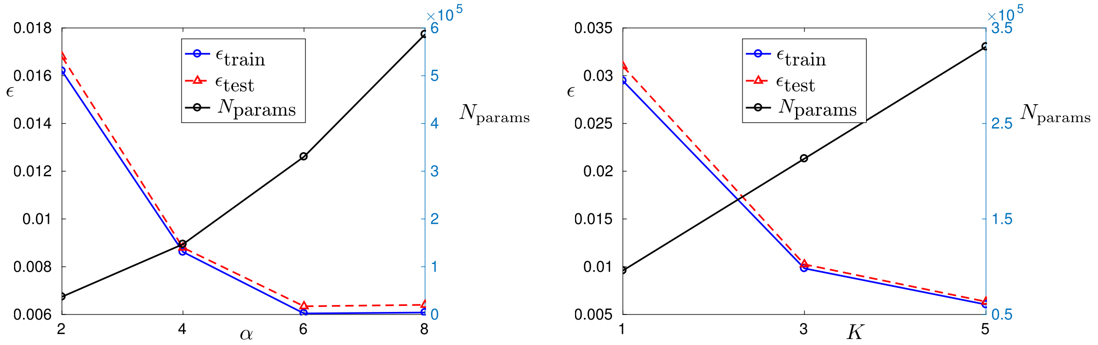
for some constant $p > 1$. In this case, the corrector function is parameterized by a unit vector $\xi \in \mathbb{S}^{d-1}$: for a fixed unit vector ξ, the corrector function $\chi_{p,\xi}(x)$ satisfies

$$-\nabla \cdot \left(a(x) |\nabla \chi_{p,\xi}(x) + \xi|^{p-2} (\nabla \chi_{p,\xi}(x) + \xi) \right) = 0, \qquad \int\limits_{[0,1)^d} \chi_{p,\xi}(x)\, \mathrm{d}x = 0, \tag{5.4}$$

with the periodic boundary condition. Since (5.4) is a nonlinear PDE, the numerical solution of the corrector function in the nonlinear case is computationally more challenging.

Here, we apply BCR-Net to learn the map from the periodic oscillatory coefficient function $a(x)$ to the corrector $\chi_{p,\xi}(x)$ for a given $\xi = (\xi_1, \ldots, \xi_d)^T$. Note that, for the case $p = 2$, (5.4) reduced to (5.3). It is noticed that, though quite different numerically, the solution of the linear problem (5.3) serves as a reasonable baseline for the nonlinear problem (5.4). Motivated by this observation, we compute $\eta_{\xi}(x) := \sum_{i=1}^{d} \xi_i \eta_i(x)$ by solving the linear system efficiently and use BCR-Net only to learn the map from $a(x)$ to the difference $g(x)$ between the nonlinear and linear correctors

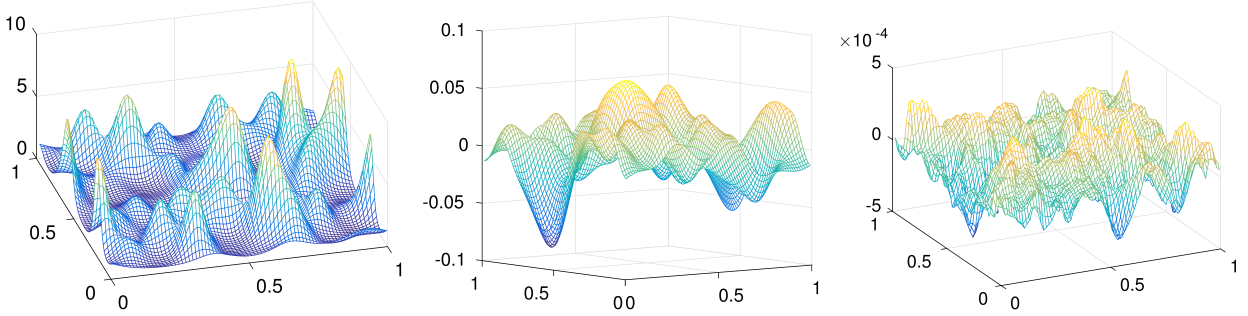
$$a(x) \to g(x) := \chi_{p,\xi}(x) - \eta_{\xi}(x) = \chi_{p,\xi}(x) - \sum_{i=1}^{d} \xi_i \eta_i(x). \tag{5.5}$$



(a) $K = 5$

(b) $\alpha = 6$

Fig. 7. Relative error and number of parameters of BCR-Net for (5.5) with $N_{\text{samples}}^{\text{train}} = N_{\text{samples}}^{\text{test}} = 20000$ in 2D. (a) Different channel numbers (α). (b) Different LC layers numbers (K).



(a) $a(x)$

(b) $\chi_{\xi,p,\text{NN}}(x)$

(c) $\chi_{\xi,p}(x) - \chi_{\xi,p,\text{NN}}(x)$

Fig. 8. A random coefficient $a(x)$, the approximation $\chi_{\xi,p,\text{NN}}(x)$ produced by BCR-Net with $\alpha = 6$ and $K = 5$, and the error from the reference solution of (5.5) in 2D.

Two-dimensional case In this test, we set $p = 3$ and discretize the coefficient field $a(x)$ with an 80×80 Cartesian grid. Each realization of $a(x)$ is generated by randomly sampling on a 10×10 grid with respect to $\mathcal{N}(0, 1)$, interpolating it to a 80×80 grid via the Fourier transform, and finally taking pointwise exponential.

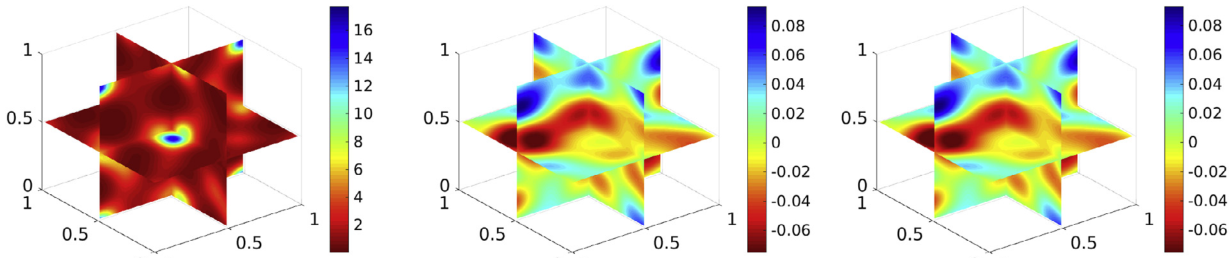
Fig. 7 summarizes the numerical results for different choices of α and K in Algorithm 4 with $n_b = 5$. Notice that the error behavior is comparable to the one shown in Section 5.1. The relative error decreases consistently as α or K increases. In addition, there is no sign of over-fitting and the number of parameters grows proportional to $\alpha^2 K$ in agreement with the complexity analysis. From Fig. 7, we notice that the best choice of the parameters for this problem is $\alpha = 6$ and $K = 5$. Fig. 8 shows a random sample of the coefficient field $a(x)$, the approximation to $\chi_{p,\xi}(x)$ predicted by BCR-Net, and the error when compared with the reference solution.

Three-dimensional case We set the discretization grid for $a(x)$ to be $40 \times 40 \times 40$ and p to be 3. The coefficient field $a(x)$ is generated by randomly sampling on a $5 \times 5 \times 5$ grid from $\mathcal{N}(0, 1)$, interpolating to the full grid via Fourier transformation, and finally taking pointwise exponential. The BCR-Net is trained with $n_b = 3$, $\alpha = 4$, and $K = 5$ using $N_{\text{samples}}^{\text{train}} = N_{\text{samples}}^{\text{test}} = 10000$ samples. The number of parameters is 5.7×10^5 and the test error is 8.7×10^{-3}. As Fig. 9 shows the results for one random realization of $a(x)$, BCR-Net is able to reproduce the corrector function quite accurately.

6. Conclusion

In this paper, inspired by the nonstandard wavelet form proposed by Beylkin, Coifman, and Rokhlin in [7], we developed a novel neural network architecture, called BCR-Net, to approximate certain nonlinear generalization of pseudo-differential operators. This NN demonstrates promising results while approximating the nonlinear maps arising from homogenization theory and stochastic computation, using only tens of thousands of parameters.

The BCR-Net architecture can be naturally extended in several ways. For instance, the LC layers can be altered to include other network structures, such as parallel sub-networks or the ResNet architecture [21]. The Conv layers corresponding the wavelet transforms can also be replaced with other types of building blocks.



(a) $a(x)$ (b) $\chi_{\xi,p,\mathrm{NN}}(x)$ (c) $\chi_{\xi,p}(x)$

Fig. 9. A random coefficient $a(x)$ in 3D, the prediction $\chi_{\xi,p,\mathrm{NN}}(x)$ from BCR-Net with $\alpha = 4$ and $K = 5$, and the reference solution of (5.5).

Acknowledgements

The work of Y.F. and L.Y. is partially supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program. The work of L.Y. is also partially supported by the National Science Foundation under award DMS-1818449. The work of C. O. is partially supported by the Stanford Graduate Fellowship in Science & Engineering. This work is also supported by the GCP Research Credits Program from Google.

References

[1] M. Abadi, et al., Tensorflow: a system for large-scale machine learning, in: OSDI, vol. 16, 2016, pp. 265–283.
[2] B. Alpert, G. Beylkin, D. Gines, L. Vozovoi, Adaptive solution of partial differential equations in multiwavelet bases, J. Comput. Phys. 182 (1) (2002) 149–190.
[3] M. Araya-Polo, J. Jennings, A. Adler, T. Dahlke, Deep-learning tomography, Lead. Edge 37 (1) (2018) 58–66.
[4] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: a deep convolutional encoder-decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. (2017).
[5] A. Bensoussan, J.-L. Lions, G. Papanicolaou, Asymptotic Analysis for Periodic Structures, AMS Chelsea Publishing, Providence, RI, 2011. Corrected reprint of the 1978 original [MR0503330].
[6] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.
[7] G. Beylkin, R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms I, Commun. Pure Appl. Math. 44 (2) (1991) 141–183.
[8] J. Bruna, S. Mallat, Invariant scattering convolution networks, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1872–1886.
[9] L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 834–848.
[10] F. Chollet, et al., Keras, https://keras.io, 2015.
[11] A. Cohen, Numerical Analysis of Wavelet Methods, vol. 32, Elsevier, 2003.
[12] N. Cohen, O. Sharir, A. Shashua, On the expressive power of deep learning: a tensor analysis, in: Conference on Learning Theory, 2016, pp. 698–728.
[13] I. Daubechies, Orthonormal bases of compactly supported wavelets, Commun. Pure Appl. Math. 41 (7) (1988) 909–996.
[14] T. Dozat, Incorporating Nesterov momentum into Adam, in: International Conference on Learning Representations, 2016.
[15] B. Engquist, P.E. Souganidis, Asymptotic and numerical homogenization, Acta Numer. 17 (2008) 147–190.
[16] L.C. Evans, An Introduction to Stochastic Differential Equations, American Mathematical Society, Providence, RI, 2013.
[17] Y. Fan, J. Feliu-Fabà, L. Lin, L. Ying, L. Zepeda-Núñez, A multiscale neural network based on hierarchical nested bases, arXiv preprint, arXiv:1808.02376, 2018.
[18] Y. Fan, L. Lin, L. Ying, L. Zepeda-Núñez, A multiscale neural network based on hierarchical matrices, arXiv preprint, arXiv:1807.01883, 2018.
[19] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, vol. 1, MIT Press, Cambridge, 2016.
[20] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (34) (2018) 8505–8510.
[21] 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.
[22] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.
[23] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[24] V.V. Jikov, S.M. Kozlov, O.A. Oleinik, Homogenization of Differential Operators and Integral Functionals, Springer-Verlag, Berlin, 1994. Translated from the Russian by G. A. Yosifian [G. A. Iosif'yan].
[25] Y. Khoo, J. Lu, L. Ying, Solving parametric PDE problems with artificial neural networks, arXiv preprint, arXiv:1707.03351, 2017.
[26] V. Khrulkov, A. Novikov, I. Oseledets, Expressive power of recurrent neural networks, arXiv preprint, arXiv:1711.00811, 2017.
[27] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems – Volume 1, NIPS'12, Curran Associates Inc., USA, 2012, pp. 1097–1105.
[28] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (436) (2015).
[29] M.K.K. Leung, H.Y. Xiong, L.J. Lee, B.J. Frey, Deep learning of the tissue-regulated splicing code, Bioinformatics 30 (12) (2014) i121–i129.
[30] Y. Li, X. Cheng, J. Lu, Butterfly-Net: optimal function representation based on convolutional neural networks, arXiv preprint, arXiv:1805.07451, 2018.
[31] L. Lin, J. Lu, L. Ying, R. Car, W. E, Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems, Commun. Math. Sci. 7 (3) (2009) 755–777.
[32] G. Litjens, T. Kooi, B.E. Bejnordi, A.A.A. Setio, F. Ciompi, M. Ghafoorian, J.A.W.M. van der Laak, B. van Ginneken, C.I. Sánchez, A survey on deep learning in medical image analysis, Med. Image Anal. 42 (2017) 60–88.
[33] J. Ma, R.P. Sheridan, A. Liaw, G.E. Dahl, V. Svetnik, Deep neural nets as a method for quantitative structure-activity relationships, J. Chem. Inf. Model. 55 (2) (2015) 263–274.

[34] S. Mallat, A Wavelet Tour of Signal Processing: The Sparse Way, third edition, Academic Press, Boston, 2008.

[35] H. Mhaskar, Q. Liao, T. Poggio, Learning functions: when is deep better than shallow, arXiv preprint, arXiv:1603.00988, 2016.

[36] M. Misiti, Y. Misiti, G. Oppenheim, J.-M. Poggi, Wavelets and Their Applications, John Wiley & Sons, 2013.

[37] G.A. Pavliotis, A.M. Stuart, Multiscale Methods – Averaging and Homogenization, Texts in Applied Mathematics, vol. 53, Springer, New York, 2008.

[38] M. Raissi, G.E. Karniadakis, Hidden physics models: machine learning of nonlinear partial differential equations, J. Comput. Phys. 357 (2018) 125–141.

[39] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention, MICCAI 2015, Springer International Publishing, Cham, 2015, pp. 234–241.

[40] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.

[41] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 27, Curran Associates, Inc., 2014, pp. 3104–3112.

[42] D. Ulyanov, A. Vedaldi, V. Lempitsky, Deep image prior, arXiv:1711.10925, 2018.

[43] Y. Wang, C.W. Siu, E.T. Chung, Y. Efendiev, M. Wang, Deep multiscale model learning, arXiv preprint, arXiv:1806.04830, 2018.

[44] I. Yavneh, G. Dardyk, A multilevel nonlinear method, SIAM J. Sci. Comput. 28 (1) (2006) 24–46.