SCIENCE CHINA
Mathematics

• ARTICLES •

July 2019 Vol. 62 No. 7: 1331–1354
https://doi.org/10.1007/s11425-019-9547-2

MgNet: A unified framework of multigrid and convolutional neural network

Juncai He[1] & Jinchao Xu[2,*]

[1]*School of Mathematical Sciences, Peking University, Beijing* 100871, *China;*
[2]*Department of Mathematics, The Pennsylvania State University, University Park, PA* 16802, *USA*

Email: juncaihe@pku.edu.cn, xu@math.psu.edu

Received February 12, 2019; accepted May 8, 2019; published online May 24, 2019

Abstract We develop a unified model, known as MgNet, that simultaneously recovers some convolutional neural networks (CNN) for image classification and multigrid (MG) methods for solving discretized partial differential equations (PDEs). This model is based on close connections that we have observed and uncovered between the CNN and MG methodologies. For example, pooling operation and feature extraction in CNN correspond directly to restriction operation and iterative smoothers in MG, respectively. As the solution space is often the dual of the data space in PDEs, the analogous concept of feature space and data space (which are dual to each other) is introduced in CNN. With such connections and new concept in the unified model, the function of various convolution operations and pooling used in CNN can be better understood. As a result, modified CNN models (with fewer weights and hyperparameters) are developed that exhibit competitive and sometimes better performance in comparison with existing CNN models when applied to both CIFAR-10 and CIFAR-100 data sets.

Keywords convolutional neural network, multigrid, unified framework, network architecture

MSC(2010) 65D19, 65N55, 68T30

Citation: He J, Xu J. MgNet: A unified framework of multigrid and convolutional neural network. Sci China Math, 2019, 62: 1331–1354, https://doi.org/10.1007/s11425-019-9547-2

1 Introduction

This paper is devoted to the study of convolutional neural networks (CNN) [12, 27, 29] in machine learning by exploring their relationship with multigrid methods for numerically solving partial differential equation [15, 48, 50]. CNN has been successfully applied in many areas, especially computer vision [30]. Important examples of CNN include the LeNet-5 model of LeCun et al. [29] in 1998, the AlexNet of Hinton et al. [27] in 2012, residual network (ResNet) of He et al. [18] in 2015 and other variants of CNN in [22, 46, 47]. Given the great success of CNN models, it is of both theoretical and practical interest to understand why and how CNN works.

In 1990s, the mathematical analysis of deep neural network (DNN) mainly focus on the approximation properties for DNN and CNN models. The first approximation results for DNN are obtained for a feedforward neural network with a single hidden layer separately in [20] and [5]. From 1989 to 1999, many

* Corresponding author

results about the so-called expressive power of single hidden neural networks are derived [1,9,42]. Recently, many new DNN structures with ReLU [40] activation functions have been studied in connection with: wavelets [44], finite element [16], sparse grid [39] and polynomial expansion [8]. By using a connection of CNN and DNN that a convolution with large enough kernel can recover any linear mapping, Zhou [55] presented an approximate result with convergence rate by deep CNNs for functions in the Sobolev space $H^r(\Omega)$ with $r > 2 + d/2$ (see also most recent result of [45]).

These function approximation theories for deep learning, are far from being adequate to explain why deep neural network, especially for CNN, works and to understand the efficiency of some successful models such as ResNet. One goal of this paper is to offer some mathematical insights into CNN by using ideas from multigrid methods and by developing a theoretical framework for these two methodologies from different fields. Furthermore, such insight is used to develop more efficient CNN models.

In the existing deep learning literature, ideas and techniques from multigrid methods have been used for the development of efficient deep neural networks. As a prominent example, the ResNet and iResNet developed in [18,19], are motivated in part by the hierarchical use of "residuals" in multigrid methods as mentioned by the authors. As another example, in [38,43], a CNN model with almost the same structure as the V-cycle multigrid is proposed to deal with volumetric medical image segmentation and biomedical image segmentation. More recently, multi-resolution images have been used as the input into the neural network in [13]. Ke et al. [25] used different networks to deal with multi-resolution images separately with a CNN to glue them together.

A dynamic system viewpoint has also been explored in many papers such as [7, 13, 36] to understand the iterative structure in ResNet type models such as the iResNet model in [19]:

$$x^i = x^{i-1} + f^i(x^{i-1}). \tag{1.1}$$

Such an idea is further explored by Li and Shi [31] to use some flow model to interpret the date flow in ResNet as the solution of transport equation following the characteristic line. Chang et al. [3] proposed a multi-level training algorithm for the ResNet model by training a shallow model first and then prolongating its parameters to train a deeper model. Lu et al. [36] used the idea of time discretization in dynamic systems to interpret PloyNet [54], FractalNet [28] and RevNet [11] as different time discretization schemes. Then they proposed the LM-ResNet based on the idea of linear multi-step schemes in numerical ODEs with a stochastic learning strategy. Long et al. [34, 35] constructed the PDE-Net models to learn the PDE model from data connecting discrete differential operators and convolutions.

In a different direction, new multigrid methods for numerical PDEs can be motivated by deep learning. For example, in [24] a deep multigrid method is proposed where the restriction and prolongation matrices with a given sparsity pattern are trained by minimizing the Frobenius norm of a large power of the multigrid error propagation matrix with a sampling technique similar to what is used in machine learning. In [21], a linear U-net structure is proposed as a solver for linear PDEs on the regular mesh.

In this paper, we explore the connection between multigrid and convolutional neural networks, in several directions. First of all, we view the multi-scale of images used in CNN as piecewise (bi-)linear functions as used in multigrid methods, and we relate the pooling operation in CNN with the restriction operation in multigrid.

To examine further connections between CNN and multigrid, we introduce the so-called data and feature space for CNN, which is analogous to the function space and its duality in the theory of multigrid methods [51]. With this new concept for CNN, we propose the data-feature mapping model in every grid as

$$A(u) = f, \tag{1.2}$$

where f belongs to the data space and u belongs to the feature space. The feature extraction process can then be obtained through an iterative procedure for solving the above system, namely,

$$u^i = u^{i-1} + B^i(f - A(u^{i-1})), \quad i = 1 : \nu, \tag{1.3}$$

with $u \approx u^\nu$. The above iterative scheme (1.3) can be interpreted as both the feature extraction step in ResNet type models and the smoothing step in multigrid method.

Using the above observations and new concepts, we develop a unified framework, called MgNet, that simultaneously recovers some convolutional neural networks and multigrid methods. Furthermore, we establish connections between several ResNet type models using the MgNet framework. We provide improvements/generalizations of several ResNet type models that are as competitive as and sometimes more efficient than existing models, as demonstrated by numerically experiments for both CIFAR-10 and CIFAR-100 [26].

The remaining sections are organized as follows. In Section 2, we introduce some notation and preliminary results in supervised learning especially for the image classification problem. In Section 3, we present the idea that we need to distinguish the data and the feature space in CNN models and introduce some related mappings. In Section 4, we explore the structures and operators when we consider images as (bi-)linear functions in multilevel grids. In Section 5, we introduce multigrid by splitting it into two phases. In Section 6, we give an abstract form of MgNet as a framework for multigrid and convolutional neural network with details. In Section 7, we introduce some classical CNN structures with rigorous mathematical definition. In Section 8, we construct some relations and connections between MgNet and classic models. In Section 9, we present some numerical results to show the efficiency of MgNet. Finally in Section 10 we give concluding remarks.

2 Supervised learning on image classification

We consider a basic machine learning problem for classifying a collection of images into κ distinctive classes. As an example, we consider a two-dimensional image which is usually represented by a tensor

$$f \in \mathcal{D} := \mathbb{R}^{m\times n\times c}.$$

Here,

$$c = \begin{cases} 1, & \text{for grayscale image,} \\ 3, & \text{for color image.} \end{cases} \tag{2.1}$$

A typical supervised machine learning problem begins with a data set (training data)

$$D := \{(f_i, y_i)\}_{i=1}^N,$$

with $\{f_i\}_{i=1}^N \subset \mathcal{D}$, and $y_i \in \mathbb{R}^\kappa$ is the label for data f_i, with $[y_i]_j$ as the probability for f_i in classes j.

Roughly speaking, a supervised learning problem can be thought as a data fitting problem in a high dimensional space $\mathcal{D}$. Namely, we need to find a mapping $H : \mathbb{R}^{m\times n\times c} \mapsto \mathbb{R}^\kappa$, such that, for a given $f \in \mathcal{D}$,

$$H(f) \approx e_i \in \mathbb{R}^\kappa, \tag{2.2}$$

if f is in class i, for $1 \leqslant i \leqslant \kappa$. For the general setting above, we use a probatilistic model for understanding the output $H(f) \in \mathbb{R}^\kappa$ as a discrete distribution on $\{1, \ldots, \kappa\}$, with $[H(f)]_i$ as the probability for f in the class i, namely,

$$0 \leqslant [H(f)]_i \leqslant 1, \quad \sum_{i=1}^{\kappa} [H(f)]_i = 1. \tag{2.3}$$

At last, we finish our model with a simple strategy to choose

$$\arg\max_i \{[H(f)]_i : i = 1 : \kappa\}, \tag{2.4}$$

as the label for a test data f, which ideally is close to (2.2). The remaining key issue is the construction of the classification mapping H.

The main step in the construction of H is to construct a nonlinear mapping

$$H_0 : \mathcal{D} \mapsto V_J, \tag{2.5}$$

with

$$V_J = \mathbb{R}^{m_J \times n_J \times c_J}. \tag{2.6}$$

To be consistent with the notation for CNN which will be described below, here the subscript J refers to the number of coarsening girds in CNN. Roughly speaking, the map H_0 plays two roles. The first role is to conduct a dimensionality reduction, namely $m_J n_J c_J \ll mnc$. The second role is to map a complicated set of data into a set of data that are linearly separable. As a result, the simple logistic regression procedure can be applied.

The first step in a logistic regression is to introduce a linear mapping

$$\Theta : \mathcal{D} \to \mathbb{R}^\kappa,$$

as

$$\Theta(x) = Wx + b, \tag{2.7}$$

where $W = (w_{ij}) \in \mathbb{R}^{(m_J \times n_J \times c_J) \times \kappa}$, $b \in \mathbb{R}^\kappa$.

We then use the soft-max function

$$[S(z)]_i = [\text{Solftmax}(z)]_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.8}$$

to obtain a logistic regression model

$$S \circ \Theta : \mathbb{R}^{m_J \times n_J \times c_J} \mapsto \mathbb{R}^\kappa. \tag{2.9}$$

By combining the nonlinear mapping H in (2.5) and the logistic regression (2.9), we obtain the following classifier:

$$H = S \circ \Theta \circ H_0. \tag{2.10}$$

Given the model (2.10), we finish the training phase with solving the next optimization problem

$$\min \sum_{j=1}^{N} l(H(f_j), y_j), \tag{2.11}$$

where $l(H(f_j), y_j)$ is a loss function that measures the predicted result $H(f_j)$ and the real label y_j. In logistic regression, the following cross-entropy loss function is often used:

$$l(H(f), y) = \sum_{i=1}^{\kappa} -[y]_i \log[H(f)]_i.$$

3 Data space, feature space and relevant mappings

We are given the data

$$f \in \mathbb{R}^{m \times n \times c}, \quad \text{or} \quad [f]_i \in \mathbb{R}^{m \times n}, \quad i = 1 : c, \tag{3.1}$$

where $m \times n$ is called the spatial dimension and c is the channel dimension.

For the given data f in (3.1), we look for some feature vector, denoted by u, associated with f:

$$u \in \mathbb{R}^{m \times n \times h}. \tag{3.2}$$

We make an assumption that the data f and feature u are related by a mapping (which can be either linear or nonlinear)

$$A : \mathbb{R}^{m \times n \times h} \mapsto \mathbb{R}^{m \times n \times c}, \tag{3.3}$$

so that

$$A(u) = f. \tag{3.4}$$

A mapping

$$B : \mathbb{R}^{m\times n\times c} \mapsto \mathbb{R}^{m\times n\times h},$$

is called a feature extractor if $B \approx A^{-1}$ and

$$v = B(f), \tag{3.5}$$

is such that $v \approx u$.

The data-feature relationship (3.4) or (3.5) is not unique. Different relationships give rise to different features. We can view the data-feature relationship given in (3.4) as a model that we propose. Here, the mapping A, which can be either linear or nonlinear, is unknown and needs to be trained.

We point out that the data space and feature space may have different numbers of channels.

3.1 A special linear mapping: Convolution

One important class of linear mapping is the so-called convolution

$$\theta : \mathbb{R}^{m\times n\times c} \mapsto \mathbb{R}^{m\times n\times h},$$

that can be defined by

$$[\theta(f)]_t = \sum_{i}^{c} K_{i,t} * [f]_i + b_t\mathbf{1} \in \mathbb{R}^{m\times n}, \quad t = 1 : h, \tag{3.6}$$

where $\mathbf{1} \in \mathbb{R}^{m\times n}$ is an $m \times n$ matrix with all elements being 1, and for $g \in \mathbb{R}^{m\times n}$,

$$[K * g]_{i,j} = \sum_{p,q=-k}^{k} K_{k+1+p,k+1+q} g_{i+p,j+q}, \quad i = 1 : m, \quad j = 1 : n. \tag{3.7}$$

The coefficients in (3.7) constitute a kernel matrix

$$K \in \mathbb{R}^{(2k+1)\times(2k+1)}, \tag{3.8}$$

where k is often taken as small integers. Here, padding means how to choose $X_{i+p,j+q}$ when $(i+p, j+q)$ is out of $1 : m$ or $1 : n$. Those next three choices are often used:

$$f_{i+p,j+q} = \begin{cases} 0, & \text{zero padding,} \\ f_{(i+p) \pmod m,(s+q) \pmod n}, & \text{periodic padding,} \\ f_{|i-1+p|,|j-1+q|}, & \text{reflected padding,} \end{cases} \tag{3.9}$$

if

$$i + p \notin \{1, 2, \ldots, m\} \quad \text{or} \quad j + q \notin \{1, 2, \ldots, n\}. \tag{3.10}$$

Here, $d \pmod m \in \{1, \ldots, m\}$ means the remainder when d is divided by m.

If we formally write

$$f = \begin{pmatrix} f_1 \\ \vdots \\ f_c \end{pmatrix}, \tag{3.11}$$

we can then write the operation (3.6) as

$$\theta(f) = K * f + b, \tag{3.12}$$

where

$$K = (K_{ij}) \in \mathbb{R}^{[(2k+1)\times(2k+1)]\times h\times c}$$

and

$$\boldsymbol{b} = \mathbf{1}_{m\times n} \otimes b.$$

The operation (3.12) is also called a convolution with stride 1. More generally, given an integer $s \geqslant 1$, a convolution with stride s for $f \in \mathbb{R}^{m\times n}$ is defined as

$$[K *_s f]_{i,j} = \sum_{p,q=-k}^{k} K_{p,q} f_{s(i-1)+1+p,s(j-1)+1+q}, \quad i = 1 : \left\lceil \frac{m}{s} \right\rceil, \quad j = 1 : \left\lceil \frac{n}{s} \right\rceil. \tag{3.13}$$

Here, $\lceil \frac{m}{s} \rceil$ denotes the smallest integer that is greater than $\frac{m}{s}$. In CNN, we often take $s = 2$.

3.2 Some linear and nonlinear mappings and extractors

A data-feature map A and feature extractor B can be either linear or nonlinear. The nonlinearity can be obtained from appropriate application of an activation function

$$\sigma : \mathbb{R} \to \mathbb{R}. \tag{3.14}$$

In this paper, we mainly consider a special activation function, known as the *rectified linear unit* (ReLU), which is defined by

$$\sigma(x) = \mathrm{ReLU}(x) := \max(0, x), \quad x \in \mathbb{R}. \tag{3.15}$$

By applying the function to each component, we can extend this as

$$\sigma : \mathbb{R}^{m\times n\times c} \mapsto \mathbb{R}^{m\times n\times c}. \tag{3.16}$$

A linear data-feature mapping can be simply given by a convolution as in (3.7):

$$A(u) = \xi * u. \tag{3.17}$$

A nonlinear mapping can be given by compositions of convolution and activation functions

$$A = \xi \circ \sigma \circ \eta \tag{3.18}$$

and

$$B = \sigma \circ \gamma \circ \sigma. \tag{3.19}$$

Here, ξ, η and γ are all appropriate convolution mappings.

3.3 Iterative feature extraction schemes

One key idea in this paper is that we consider different iterative processes to approximately solve (3.4) and relate them to many existing popular CNN models. Here, let us assume that the feature-data mapping (3.4) is given as a linear form (3.17). We next propose some iterative schemes to solve (3.4) for an appropriately chosen u^0.

- Residual correction method,

$$u^i = u^{i-1} + B^i(f - A(u^{i-1})), \quad i = 1 : \nu. \tag{3.20}$$

Here, B^i can be chosen as linear like $B^i(f) = \eta^i * f$ or nonlinear like (3.19). The reason why B^i is taken the nonlinear form as in (3.19) will be discussed later based on our main discovery about the relationship between MgNet and iResNet as discussed in Sections 7 and 8. We refer to [48] for more discussion on iterative schemes in the form of (3.20).

- Semi-iterative method for accelerating the residual correction iterative scheme,

$$u^i = \sum_{j=0}^{i-1} \alpha_j^i (u^j + B_j^i(f - A(u^j))), \quad i = 1 : \nu, \tag{3.21}$$

where $\alpha_j^i \geqslant 0$ and $\sum_{j=0}^{i-1} \alpha_j^i = 1$. Let the residual $r^j = f - A(u^j)$ for $j = 0 : i$. The following iterative scheme for r^j is implied by (3.21),

$$r^i = \sum_{j=0}^{i-1} \alpha_j^i (I - AB_j^i)(r^j), \tag{3.22}$$

because of the linearity of A. This scheme is analogous to the DenseNet [22] which will be discussed more in Section 7 and below. More discussion on semi-iterative method for linear system can be found in [10, 14].

- Chebyshev semi-iterative method,

$$u^i = \omega^i(u^{i-1} + B^i(f - A(u^{i-1}))) + (1 - \omega^i)u^{i-2}, \quad i = 1 : \nu. \tag{3.23}$$

The above scheme can be obtained from the above semi-iterative form by applying the Chebyshev polynomial theory [10, 14]. Similar to the previous case, by considering the iterative form of the residual $r^j = f - A(u^j)$, (3.23) implies that

$$r^i = \omega^i r^{i-1} + (1 - \omega^i)r^{i-2} - AB^i r^{i-1}. \tag{3.24}$$

This scheme corresponds to the LM-ResNet in [36] which was obtained as a linear multi-step scheme for some underlying ODEs.

4 Piecewise (bi-)linear functions on multilevel grids

An image can be viewed as a function on a grid. Images with different resolutions can then be viewed as functions on grids of different sizes. The use of such multiple-grids is a main technique used in the standard multigrid method for solving discretized partial differential equations [48, 50], and it can also be interpreted as a main ingredient used in convolutional neural networks (CNN).

Without loss of generality, for simplicity, we assume that the initial grid, $\mathcal{T}$, is of size

$$m = 2^s + 1, \quad n = 2^t + 1,$$

for some integers $s, t \geqslant 1$. Starting from $\mathcal{T}_1 = \mathcal{T}$, we consider a sequence of coarse grids (as depicted in Figure 1 with $J = 4$):

$$\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_J, \tag{4.1}$$

such that $\mathcal{T}_\ell$ ($\ell = 1, \ldots, J$) consist of $m_\ell \times n_\ell$ grid points, with

$$m_\ell = 2^{s-\ell+1} + 1, \quad n_\ell = 2^{t-\ell+1} + 1. \tag{4.2}$$

The grid points of these grids can be given by

$$x_i^\ell = ih_{1,\ell}, \quad y_j^\ell = jh_{2,\ell}, \quad i = 1, \ldots, m_\ell, \quad j = 1, \ldots, n_\ell.$$

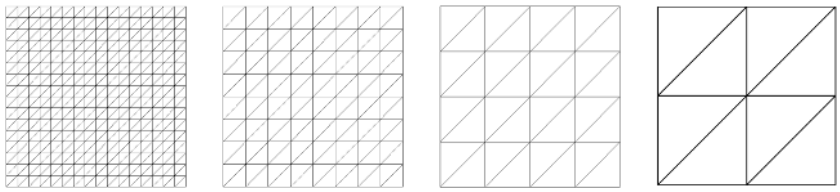

Figure 1 Multilevel grids for piecewise linear functions

Here, $h_{1,\ell}=2^{-s+\ell-1}a$ and $h_{2,\ell}=2^{-t+\ell-1}b$, for some $a,b>0$. The above geometric coordinates (x_i^ℓ,y_i^ℓ) are usually not used in image precess literatures, but they are relevant in the context of multigrid method for the numerical solution of PDEs. We now consider piecewise linear functions on the sequence of grids (4.1) and we obtain a nested sequence of linear vector spaces

$$\mathcal{V}_1\supset\mathcal{V}_2\supset\cdots\supset\mathcal{V}_J. \tag{4.3}$$

Here, each $\mathcal{V}_\ell$ consists of all piecewise bilinear (or linear) functions with respect to the grid (4.1) and (4.2). Each $\mathcal{V}_\ell$ has a set of basis functions: $\phi_{ij}^\ell\in\mathcal{V}_\ell$ satisfying

$$\phi_{ij}^\ell(x_p,y_q)=\delta_{(i,j),(p,q)}=\begin{cases}1, & \text{if} \quad (p,q)=(i,j),\\ 0, & \text{if} \quad (p,q)\neq(i,j).\end{cases}$$

Thus, for each $v\in\mathcal{V}_\ell$, we have

$$v(x,y)=\sum_{i,j}v_{ij}^\ell\phi_{ij}^\ell(x,y). \tag{4.4}$$

4.1 Prolongation

Given a piecewise (bi-)linear function $\boldsymbol{v}\in\mathcal{V}_{\ell+1}$, the nodal values of $\boldsymbol{v}$ on $m_{\ell+1}\times n_{\ell+1}$ grids point constitute a tensor

$$v^{\ell+1}\in\mathbb{R}^{m_{\ell+1}\times n_{\ell+1}}.$$

We note that $\boldsymbol{v}\in\mathcal{V}_\ell$ thanks to (4.3) and the nodal values of $\boldsymbol{v}$ on $\mathcal{T}_\ell$ constitute a tensor

$$v^\ell\in\mathbb{R}^{m_\ell\times n_\ell}.$$

By using the property of piecewise (bi-)linear functions, it is easy to see that

$$v^\ell=\bar{P}_{\ell+1}^\ell v^{\ell+1}, \tag{4.5}$$

where

$$\bar{P}_{\ell+1}^\ell:\mathbb{R}^{m_{\ell+1}\times n_{\ell+1}}\mapsto\mathbb{R}^{m_\ell\times n_\ell}, \tag{4.6}$$

which is called a prolongation in multigrid terminology. More specifically,

$$v_{2i-1,2j-1}^\ell=v_{i,j}^{\ell+1} \tag{4.7}$$

with

$$v_{2i-1,2j}^\ell=\frac{1}{2}(v_{i,j}^{\ell+1}+v_{i,j+1}^{\ell+1}),\quad v_{2i,2j-1}^\ell=\frac{1}{2}(v_{i,j}^{\ell+1}+v_{i+1,j}^{\ell+1}) \tag{4.8}$$

and

$$v_{2i,2j}^\ell=\begin{cases}\dfrac{1}{4}(v_{i,j}^{\ell+1}+v_{i+1,j}^{\ell+1}+v_{i,j+1}^{\ell+1}+v_{i+1,j+1}^{\ell+1}), & \text{if } v^\ell \text{ is piecewise bilinear},\\ \dfrac{1}{2}(v_{i+1,j}^{\ell+1}+v_{i,j+1}^{\ell+1}), & \text{if } v^\ell \text{ is piecewise linear}.\end{cases} \tag{4.9}$$

4.2 Pooling, restriction and interpolation

The prolongation given by (4.6) can be used to transfer feature from a coarse grid to a fine grid. On the other hand, we also need a mapping, known as restriction, that transfer data from fine grid to corse grid

$$\bar{R}_\ell^{\ell+1}:\mathbb{R}^{m_\ell\times n_\ell}\mapsto\mathbb{R}^{m_{\ell+1}\times n_{\ell+1}}. \tag{4.10}$$

In multigrid for solving the discretized partial differential equation, the restriction is often taken to be transpose of the prolongation given by (4.6):

$$\bar{R}_\ell^{\ell+1}=(\bar{P}_{\ell+1}^\ell)^{\mathrm{T}}. \tag{4.11}$$

Lemma 4.1. *If $\tilde{P}_{\ell+1}^{\ell}$ takes the form of prolongation in multigrid methods for linear finite element functions on the above grids, then $\tilde{R}_{\ell}^{\ell+1}$ is a convolution with stride 2 and a 3×3 kernel as*

$$R_{\ell}^{\ell+1}f = K_R *_2 f, \tag{4.12}$$

where, if $\mathcal{V}_\ell$ is piecewise bilinear,

$$K_R = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}, \tag{4.13}$$

or, if $\mathcal{V}_\ell$ is piecewise linear,

$$K_R = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}. \tag{4.14}$$

In addition, all these convolutions are applied with zero padding as in (3.9), which is consistent with the Neumann boundary condition for applying finite element method (FEM) to numerical PDEs. More details will be discussed in Subsection 4.2.

In the deep learning literature, the restriction such as (4.10) is often known as pooling operation. One popular pooling is a convolution with stride s, with some small integer $s>1$.

Some other fixed (or untrained) poolings are also often used. One popular pooling is the so-called average pooling R_{avr} which can be a convolution with stride 2 or bigger using the kernel K in the form of

$$K = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{4.15}$$

Nonlinear pooling operator is also used, for the example the $(2k+1)\times(2k+1)$ max-pooling operator with stride s as follows:

$$[R_{\max}(f)]_{i,j} = \max_{-k\leqslant p,q\leqslant k}\{f_{s(i-1)+1+p,s(j-1)+1+q}\}. \tag{4.16}$$

Another approach to the construction of restriction of pooling can be obtained by using interpolation. Given

$$v^{\ell}\in\mathbb{R}^{m_\ell\times n_\ell},$$

let $\boldsymbol{v}\in\mathcal{V}_\ell$ be the function whose nodal values are precisely given by v^ℓ as in (4.4). Any reasonable linear operator

$$\Pi : \mathcal{V}_\ell \mapsto \mathcal{V}_{\ell+1}, \tag{4.17}$$

such as nodal value interpolation, Scott-Zhang interpolation and L^2 projection [49], would give rise to a mapping

$$\Pi_{\ell}^{\ell+1} : \mathbb{R}^{m_\ell\times n_\ell}\mapsto\mathbb{R}^{m_{\ell+1}\times n_{\ell+1}} \tag{4.18}$$

such that

$$v^{\ell+1} = \Pi_{\ell}^{\ell+1}v^{\ell}.$$

As situations permit, we can use these a priori given restrictions to replace unknown pooling operators to reduce the number of parameters.

5 Multigrid methods for numerical PDEs

Let us first briefly describe a geometric multigrid method used to solve the following boundary value problem:

$$-\Delta u = f, \quad \text{in} \quad \Omega, \quad \frac{\partial u}{\partial \boldsymbol{n}} = 0 \quad \text{on} \quad \partial\Omega, \quad \Omega = (0,1)^2. \tag{5.1}$$

We consider a continuous linear finite element discretization of (5.1) on a nested sequence of grids of sizes $n_\ell \times n_\ell$ with $n_\ell = 2^{J-\ell+1}+1$, as shown in the left part of Figure 1 and the corresponding sequence of finite element spaces (4.3).

Based on the grid $\mathcal{T} = \mathcal{T}_\ell$, the discretized system is

$$Au = f. \tag{5.2}$$

Here, $A : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is a tensor satisfying

$$(Au)_{i,j} = 4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}, \tag{5.3}$$

which holds for $1 \leqslant i,j \leqslant n$ with zero padding. Here we notice that, there exists a 3×3 kernel as

$$K_A = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{5.4}$$

with

$$Au = K_A * u, \tag{5.5}$$

where $*$ is the standard convolution operation with zero padding like (3.7). We now briefly describe a simple multigrid method by a mixed use of the terminologies from deep learning [12] and multigrid methods.

The first main ingredient in geometric multigrid (GMG) method is a smoother. A commonly used smoother is a damped Jacobi with damped coefficient ω with $\omega \in (0,2)$, which can be written as $S_0 : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ satisfying

$$(S_0 f)_{i,j} = \frac{\omega}{4} f_{i,j}, \tag{5.6}$$

for (5.2) with initial guess zero. If we apply the Jacobian iteration twice, then

$$S_1(f) = S_0 f + S(f - A(S_0 f))$$

with element-wise form

$$[S_1(f)]_{i,j} = \frac{1}{4}\omega(2-\omega) f_{i,j} + \frac{\omega^2}{16}(f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}). \tag{5.7}$$

Then we have

$$K_{S_0} = \frac{\omega}{4} \tag{5.8}$$

and

$$K_{S_1} = \begin{pmatrix} 0 & \frac{\omega^2}{16} & 0 \\ \frac{\omega^2}{16} & \frac{\omega(2-\omega)}{4} & \frac{\omega^2}{16} \\ 0 & \frac{\omega^2}{16} & 0 \end{pmatrix}, \tag{5.9}$$

such that

$$S_0 f = K_{S_0} * f, \quad S_1 f = K_{S_1} * f. \tag{5.10}$$

Similarly, we can define

$$S^\ell : \mathbb{R}^{n_\ell \times n_\ell} \mapsto \mathbb{R}^{n_\ell \times n_\ell}.$$

We use prolongation

$$P_{\ell+1}^{\ell} : \mathbb{R}^{n_{\ell+1}\times n_{\ell+1}} \mapsto \mathbb{R}^{n_\ell \times n_\ell}$$

as defined in (4.6) and the restriction $R_\ell^{\ell+1} = (P_{\ell+1}^{\ell})^{\mathrm{T}}$. Furthermore, we use the following relationship to define coarse operation:

$$A^{\ell+1} = R_\ell^{\ell+1} A^\ell P_{\ell+1}^{\ell}, \quad \ell = 1 : J-1 \tag{5.11}$$

with $A^1 = A$.

Using the smoother S^ℓ, prolongation $P_{\ell+1}^{\ell}$, restriction $R_\ell^{\ell+1}$ and mapping A^ℓ as given in (5.11), we can formulate the following algorithm as a major component of a multigrid algorithm:

Algorithm 1 $(u^{\ell,\nu_\ell} : \ell = 1 : J) = \mathrm{MG0}(f; J, \nu_1, \ldots, \nu_J)$

1: Set up

$$f^1 = f, \quad u^{1,0} = 0.$$

2: Smoothing and restriction from fine to coarse level (nested)
3: **for** $\ell = 1 : J$ **do**
4: Pre-smoothing:
5: **for** $i = 1 : \nu_\ell$ **do**
6:

$$u^{\ell,i} = u^{\ell,i-1} + S^\ell(f^\ell - A^\ell u^{\ell,i-1}). \tag{5.12}$$

7: **end for**
8: Form restricted residual and set initial guess:

$$u^{\ell+1,0} = 0, \quad f^{\ell+1} = R_\ell^{\ell+1}(f^\ell - A^\ell u^{\ell,\nu_\ell}).$$

9: **end for**

Using Algorithm 1, there are different multigrid algorithms such as: \-cycle, V-cycle and W-cycle. Let us now only give one special form of multigrid algorithm as follows (see Algorithm 2):

Algorithm 2 $u = \backslash\text{-MG}(f; J, \nu_1, \ldots, \nu_J)$

1: Call Algorithm 1,

$$(u^{\ell,\nu_\ell} : \ell = 1 : J) = \mathrm{MG0}(f; J, \nu_1, \ldots, \nu_J).$$

2: Prolongation and restriction from coarse to fine level
3: **for** $\ell = J-1 : 1$ **do**
4: Coarse grid correction (residual)

$$u^{\ell,\nu_\ell} \leftarrow u^{\ell,\nu_\ell} + P_{\ell+1}^{\ell} u^{\ell+1,\nu_{\ell+1}}. \tag{5.13}$$

5: **end for**
6: Output

$$u = u^{1,\nu_1}.$$

6 MgNet: A new network structure

In this section, we introduce a new neural network structure, named as MgNet, motivated by the multigrid algorithm, Algorithm 1, as discussed in the previous section.

First, given the data-feature equation (3.4), we consider its restrictions to grid ℓ as follows:

$$A^\ell(u^\ell) = f^\ell, \quad \ell = 1 : J, \tag{6.1}$$

where

$$f^\ell \in \mathbb{R}^{m_\ell \times n_\ell \times c_{f,\ell}} \tag{6.2}$$

and

$$u^\ell \in \mathbb{R}^{m_\ell \times n_\ell \times c_{u,\ell}}. \tag{6.3}$$

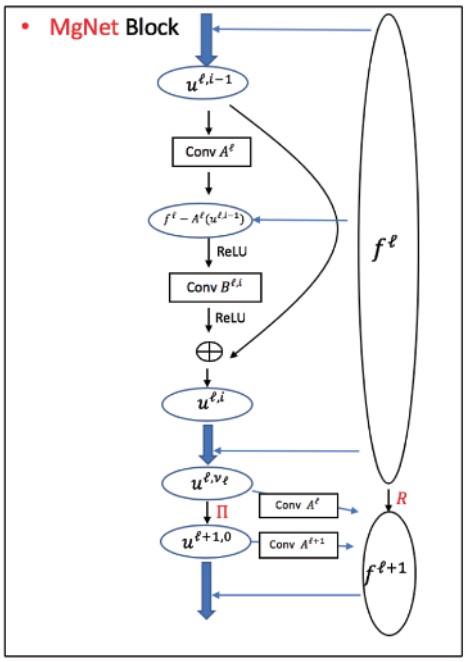

Figure 2 (Color online) Structure of MgNet

We are now in a position to state the main algorithm, namely MgNet as follows:

Algorithm 3 $u^J = \mathrm{MgNet}(f; J, \nu_1, \ldots, \nu_J)$

Initialization: $f^1 = f_{\mathrm{in}}(f)$, $u^{1,0} = 0$
for $\ell = 1 : J$ **do**
  for $i = 1 : \nu_\ell$ **do**
    Feature extraction (smoothing):

$$u^{\ell,i} = u^{\ell,i-1} + B^{\ell,i}(f^\ell - A^\ell(u^{\ell,i-1})). \tag{6.4}$$

  end for
  Note: $u^\ell = u^{\ell,\nu_\ell}$
  Interpolation and restriction:

$$u^{\ell+1,0} = \Pi_\ell^{\ell+1} u^\ell, \tag{6.5}$$

$$f^{\ell+1} = R_\ell^{\ell+1}(f^\ell - A^\ell(u^\ell)) + A^{\ell+1}(u^{\ell+1,0}). \tag{6.6}$$

end for

Here, $f_{\mathrm{in}}(\cdot)$ is the data initialization process as a usual step in many classical CNNs [18, 19, 22, 27]. It may depend on different data sets and problems. We will discuss it later in Subsection 6.1 and Section 7. For the main structure, the next diagram (see Figure 2) gives a brief illustration for the schema of MgNet as shown in Algorithm 3 with (3.17) and (3.19).

Here, we may have some more general MgNet structures by replacing the feature extraction (smoothing) step (6.4) with some other iterative schemes such as:

(Single step) MgNet.

$$u^{\ell,i} = u^{\ell,i-1} + B^{\ell,i}(f^\ell - A^\ell(u^{\ell,i-1})), \quad i = 1 : \nu_\ell. \tag{6.7}$$

Multi-step MgNet.

$$u^{\ell,i} = \sum_{j=0}^{i-1} \alpha_j^{\ell,i}(u^{\ell,j} + B_j^{\ell,i}(f^\ell - A^\ell(u^{\ell,j}))), \quad i = 1 : \nu_\ell. \tag{6.8}$$

Chebyshev-semi MgNet.

$$u^{\ell,i} = \omega^{\ell,i}(u^{\ell,i-1} + B^{\ell,i}(f^\ell - A^\ell(u^{\ell,i-1}))) + (1-\omega^{\ell,i})u^{\ell,i-2}, \quad i = 1:\nu_\ell, \tag{6.9}$$

where $B^{\ell,i}$ and $B_j^{\ell,i}$ can be some appropriate nonlinear forms such as (3.19) in the basic MgNet in Algorithm 3 which can relate to iResNet model naturally. Roughly speaking, multi-step MgNet structure and Chebyshev-semi MgNet may be related to DenseNet [22] and LM-ResNet [36] with a special choice of the nonlinear form of $B_j^{\ell,i}$ and $B^{\ell,i}$.

Let us focus on the basic MgNet form in Algorithms 3. The first important property of MgNet is that it recovers the fine to coarse process of multigrid methods as in Algorithm 1.

Theorem 6.1. *If A^ℓ, $R_\ell^{\ell+1}$ and $B^{\ell,i} = S^\ell$ are all linear operations as described in multigrid method in Section* 5*, then Algorithm* 1 *is equivalent to Algorithm* 3 *with any choice of $\Pi_\ell^{\ell+1}$.*

Proof. Here, we replace $u^{\ell,i}$ and f^ℓ by $\tilde{u}^{\ell,i}$ and $\tilde{f}^\ell$ in MgNet. What we want to prove are

$$\tilde{f}^\ell = f^\ell + A_\ell \tilde{u}^{\ell,0} \quad \text{and} \quad u^{\ell,i} = \tilde{u}^{\ell,i} - \tilde{u}^{\ell,0}, \tag{6.10}$$

with $u^{\ell,i}$, f^ℓ in Algorithm 1 and $\tilde{u}^{\ell,i}$, $\tilde{f}^\ell$ in Algorithm 3 for any choice of $\Pi_\ell^{\ell+1}$. We prove this result by induction.

- It is easy to check that $\ell = 1$ is right by taking $\theta = \mathrm{id}$.
- Once the above equation (6.10) is right for ℓ, let us prove the corresponding result for $\ell+1$.
- For $\tilde{f}^{\ell+1}$, as the definition in Algorithm 3, we have

$$\begin{aligned}
\tilde{f}^{\ell+1} &= R_\ell^{\ell+1}(\tilde{f}^\ell - A^\ell \tilde{u}^{\ell,\nu_\ell}) + A^{\ell+1}\tilde{u}^{\ell+1,0} \\
&= R_\ell^{\ell+1}(f^\ell + A^\ell \tilde{u}^{\ell,0} - A^\ell \tilde{u}^{\ell,\nu_\ell}) + A^{\ell+1}\tilde{u}^{\ell+1,0} \\
&= R_\ell^{\ell+1}(f^\ell - A^\ell(\tilde{u}^{\ell,\nu_\ell} - u^{\ell,0})) + A^{\ell+1}\tilde{u}^{\ell+1,0} \\
&= R_\ell^{\ell+1}(f^\ell - A^\ell u^{\ell,\nu_\ell}) + A^{\ell+1}\tilde{u}^{\ell+1,0} \\
&= f^{\ell+1} + A^{\ell+1}\tilde{u}^{\ell+1,0}.
\end{aligned} \tag{6.11}$$

- For $u^{\ell+1,i}$, first we have

$$u^{\ell+1,0} = 0 = \tilde{u}^{\ell+1,0} - \tilde{u}^{\ell+1,0}, \tag{6.12}$$

and then we prove

$$u^{\ell+1,i} = \tilde{u}^{\ell+1,i} - \tilde{u}^{\ell+1,0} \tag{6.13}$$

by induction for i.

We assume (6.13) holds for $0, 1, \ldots, i-1$. Let us miner $\tilde{u}^{\ell+1,0}$ on both sides of the smoothing process (6.4) in Algorithm 3. Then we have

$$\begin{aligned}
\tilde{u}^{\ell+1,i} - \tilde{u}^{\ell+1,0} &= \tilde{u}^{\ell+1,i-1} - \tilde{u}^{\ell+1,0} + B^{\ell+1,i}(\tilde{f}^{\ell+1} - A^{\ell+1}\tilde{u}^{\ell+1,i-1}) \\
&= \tilde{u}^{\ell+1,i-1} - \tilde{u}^{\ell+1,0} + B^{\ell+1,i}(f^{\ell+1} + A^{\ell+1}\tilde{u}^{\ell+1,0} - A^{\ell+1}\tilde{u}^{\ell+1,i-1}) \\
&= u^{\ell+1,i-1} + B^{\ell+1,i}(f^{\ell+1} - A^{\ell+1}u^{\ell+1,i-1}).
\end{aligned} \tag{6.14}$$

This is exactly the smoothing process in Algorithm 1 as we take $B^{\ell+1,i} = S^{\ell+1}$. □

Similar to Algorithm 2 in \-MG or the corresponding version in V-cycle multigrid, there exists a related V-MgNet (see Algorithm 4) that includes a process from coarse to fine grids. This type of V-MgNet makes use of prolongation operators that correspond directly to the co-called deconvolution operations in CNN models [41]. In addition, the correction steps such as (6.15) correspond directly to the symmetric skip connection in many autoencoder type models such as U-net [43] and others [32, 33, 37]. Furthermore, we can actually recover these U-net type CNN models from V-MgNet with similar situation to that as in MgNet and iResNet which we will discuss later in Section 8.

Despite of the simplicity look of Algorithm 3, there are rich mathematical structures and variants which we briefly discuss below.

Algorithm 4 $u^1 = \text{V-MgNet}(f; J, \nu_1, \dots, \nu_J; \nu'_1, \dots, \nu'_J)$

1: Call Algorithm 3,
$$(\bar{u}^{1,0}, \bar{u}^1, f^1, \bar{u}^{2,0}, \bar{u}^2, f^2, \dots, \bar{u}^{J,0}, \bar{u}^J, f^J) = \text{MgNet}(f; J, \nu_1, \dots, \nu_J).$$

2: **for** $\ell = J-1:1$ **do**

3: Prolongation (deconvolution) and correction (shortcut connection)
$$u^{\ell,0} \leftarrow \bar{u}^\ell + P^\ell_{\ell+1}(u^{\ell+1} - \bar{u}^{\ell+1,0}). \tag{6.15}$$

4: **for** $i = 1:\nu'_\ell$ **do**

5: Feature extraction (post-smoothing)
$$u^{\ell,i} \leftarrow u^{\ell,i-1} + B'_{\ell,i}(f^\ell - A^\ell(u^{\ell,i-1})). \tag{6.16}$$

6: **end for**

7:
$$u^\ell \leftarrow u^{\ell,\nu'_\ell}.$$

8: **end for**

6.1 Initialization: Feature space channels

Initially for $\ell = 1$, we take $m_1 = m$ and $n_1 = n$ and we may define the linear mapping

$$\theta : \mathbb{R}^{m\times n\times c} \mapsto \mathbb{R}^{m_1\times n_1\times c_1} \tag{6.17}$$

to obtain $f^1 = f_{\text{in}}(f) = \theta(f)$ with c given in (2.1) changed to the channel of the initial data space to c_1. Usually,

$$c_1 \geqslant c. \tag{6.18}$$

One possibility is that we choose $c_1 = c$. In this case, we choose $\theta =$ identity. But in general, we may need to choose $c_1 \gg c$. One possible advantage of preprocessing the red, green and blue colors ($c = 3$) to different color spaces is that we can better choose what kind of features the CNN can detect, and under what conditions those detections will be invariant.

One possibility of understanding and modifying this step is to decompose the data f into a number of more specialized data

$$f = \sum_{k=1}^{c_1} \xi_k f^1_k = \xi^T f^1. \tag{6.19}$$

We may use some knowledge from image processing or physics to design a procedure to obtain the right decomposition of (6.19), or we can just train it. Conceivably, we may view $f^1 = \theta(f)$ as a special approximation solution of (6.19) with the same sparsity pattern to ξ.

6.2 Extracted units: u^ℓ and channels

The first new feature and the main new ingredient in the proposed neural network is the introduction of feature variables u^ℓ in (6.3), which will be known as the extracted units.

One main ingredient in our MgNet in addition to the data variables is the introduction of feature variables u^ℓ in (6.3), known as the extracted-units. The so-called dual path networks (DPN) model in [4] also makes use of additional variables. DPN is a special CNN obtained by combining two different CNN models such as ResNet and DenseNet. If we view $u^{\ell,i}$ and f^ℓ as two different paths, MgNet can be related to DPN model. We note that, $u^{\ell,i}$ and f^ℓ communicate to each other with a special version as in (6.4) with a special restriction form as in (6.6). We can recover DPN from MgNet by using two different smoothing processes and combining them.

We emphasize that the extracted-units $u^{\ell,i}$ and the data f^ℓ can have different numbers of channels:

$$u^{\ell,i} \in \mathbb{R}^{m_\ell\times n_\ell\times c_{u,\ell}}, \quad f^\ell \in \mathbb{R}^{m_\ell\times n_\ell\times c_{f,\ell}}. \tag{6.20}$$

One possibility is that the number of channels for both u and f remains unchanged in different grids:

$$c_{f,\ell} = c_f, \quad \ell = 1 : J \tag{6.21}$$

and

$$c_{u,\ell} = c_u, \quad \ell = 1 : J. \tag{6.22}$$

Both c_f and c_u are two super-parameters that need to be tuned, and we may even take $c_u = c_f$.

6.3 Poolings: $\Pi_\ell^{\ell+1}$ and $R_\ell^{\ell+1}$

The pooling $\Pi_\ell^{\ell+1}$ in (4.12) and $R_\ell^{\ell+1}$ in (6.6) are in general different. They can be trained in general, but they may be a priori chosen.

There are many different possibilities to choose $\Pi_\ell^{\ell+1}$. The simplest choice of $\Pi_\ell^{\ell+1}$ is

$$\Pi_\ell^{\ell+1} = 0. \tag{6.23}$$

A more sophisticated choice can be obtained by considering an interpolation from fine grid to coarse (that, for example preserves linear function locally). Namely,

$$\Pi_\ell^{\ell+1} = \bar{\Pi}_\ell^{\ell+1} \otimes I_{c_\ell \times c_\ell} \tag{6.24}$$

with $\bar{\Pi}_\ell^{\ell+1}$ given by (4.18). It can be implemented by group convolution [53] with channels as groups number.

6.4 Data-feature mapping: A^ℓ

The second new feature of MgNet is that this data-feature mapping only depends on the grid $\mathcal{T}_\ell$, and it does not depend on layers within the same grid. This amounts to a significant saving of the number of parameters especially for deep ResNet models. In comparison, the existing CNN, such as iResNet, can be interpreted as a network related to the case that A^ℓ is replaced by $A^{\ell,i}$, namely,

$$u^{\ell,i} = u^{\ell,i-1} + B^{\ell,i}(f^\ell - A^{\ell,i}(u^{\ell,i-1})), \tag{6.25}$$

which will be discussed later in Section 8.

The data-feature mapping: A^ℓ can be either linear (3.17), or nonlinear (3.18). The underlying convolution kernels can be different on different grids and they can all be trained.

6.5 Feature extractors: $B^{\ell,i}$

There are some freedoms in choosing these feature extrators. One common choice of extractors is given by (3.19), namely,

$$B^{\ell,i} = \sigma \circ \eta^{\ell,i} \circ \sigma. \tag{6.26}$$

Other than the level dependent extractors, the following different strategies can be used:

Constant extractors. $B^{\ell,i} = B^\ell$ for $i = 1 : \nu_\ell$.

Scaled extractors. $B^{\ell,i} = \alpha_i B^\ell$ for $i = 1 : \nu_\ell$.

Variable extractors. $B^{\ell,i}$.

This brief framework gives us the basic principle on designing a CNN models for classification. All models are seen as the special choice of data-feature mapping A^ℓ, feature extractors $B^{\ell,i}$ and the pooling operators $\Pi_\ell^{\ell+1}$ with $R_\ell^{\ell+1}$.

7 Some classic CNN models

In this section, we will use the notation introduced above to give a brief description of some classic CNN models.

7.1 LeNet-5, AlexNet and VGG

The LeNet-5 [29], AlexNet [27] and VGG [46] can be written as:

$$\begin{cases} f^{1,0} = \theta^0(f), \\ \textbf{for} \quad \ell = 1 : J \\ \qquad \textbf{for} \quad i = 1 : \nu_\ell \\ \qquad f^{\ell,i} = \theta^{\ell,i} \circ \sigma(f^{\ell,j-1}), \\ \qquad \textbf{end for} \\ \qquad f^{\ell+1,0} = R_\ell^{\ell+1}(f^{\ell,m+\ell}), \\ \textbf{end for}, \end{cases} \tag{7.1}$$

where $R_\ell^{\ell+1}$ can be general pooling operators and $\theta^{\ell,i}$ can be convolution with stride 1, or fully connected operators. Then the CNN model will be defined by

$$H_0(f) = f^{J,\nu_J}. \tag{7.2}$$

In these three classic CNN models, they still need some extra fully connected layers after $H_0(f)$ but before the logistic regression (2.9). These fully connected layers are removed in ResNet to be described below.

7.2 ResNet

The ResNet [18] can be written as

$$\begin{cases} f^{1,0} = f_{\text{in}}(f), \\ \textbf{for} \quad \ell = 1 : J \\ \qquad \textbf{for} \quad i = 1 : \nu_\ell \\ \qquad f^{\ell,i} = \sigma(f^{\ell,i-1} + \mathcal{F}^{\ell,i}(f^{\ell,i-1})), \\ \qquad \textbf{end for} \\ \qquad f^{\ell+1,0} = \sigma(R_\ell^{\ell+1}(f^{\ell,\nu_\ell}) + \mathcal{F}^{\ell,0}(f^{\ell,\nu_\ell})), \\ \textbf{end for} \\ H_0(f) = R_{\text{ave}}(f^{L,\nu_\ell}). \end{cases} \tag{7.3}$$

Here, $f_{\text{in}}(\cdot)$ may depend on different data set and problems such as $f_{\text{in}}(f) = \sigma \circ \theta^0(f)$ for CIFAR [26] and

$$f_{\text{in}}(f) = R_{\max} \circ \sigma \circ \theta^0(f)$$

for ImageNet [6] as in [18]. In addition,

$$\sigma(f^{\ell,i-1} + \mathcal{F}^{\ell,i}(f^{\ell,i-1}))$$

is often called the basic ResNet block with

$$\mathcal{F}^{\ell,i}(f^{i-1}) = \xi^i \circ \sigma \circ \eta^i(f^{i-1}).$$

Generally, $\xi^{\ell,i}$ and $\eta^{\ell,i}$ take the form of with zero padding and stride 1, except, $\eta^{\ell,0}$ is taken as convolution with stride 2 with the same output dimension of $R_\ell^{\ell+1}$.

7.3 iResNet

The iResNet [19] can be written as:

$$\begin{cases} f^{1,0} = f_{\text{in}}(f), \\ \textbf{for} \quad \ell = 1 : J \\ \qquad \textbf{for} \quad i = 1 : \nu_\ell \\ \qquad f^{\ell,i} = f^{\ell,i-1} + \mathcal{F}^{\ell,i}(f^{\ell,i-1}), \\ \qquad \textbf{end for} \\ \qquad f^{\ell+1,0} = R_\ell^{\ell+1}(f^{\ell,\nu_\ell}) + \mathcal{F}^{\ell,0}(f^{\ell,\nu_\ell}), \\ \textbf{end for} \\ H_0(f) = R_{\text{ave}}(f^{L,\nu_\ell}), \end{cases} \tag{7.4}$$

where $f_{\text{in}}(\cdot)$ shares the same setup with ResNet but $\mathcal{F}^{\ell,i}(f^{\ell,i-1}) = \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i}\sigma(f^{\ell,i-1})$. The only difference between ResNet and iResNet can be viewed as putting a σ in different places.

7.4 DenseNet

The DenseNet [22] model can be written as:

$$\begin{cases} f^{1,0} = f_{\text{in}}(f), \\ \textbf{for} \quad \ell = 1 : J \\ \qquad \textbf{for} \quad i = 1 : \nu_\ell \\ \qquad f^{\ell,i} = \sigma\Big(\sum_{j=0}^{i-1} [\theta^{\ell,i}]_j * f^{\ell,j}\Big), \\ \qquad \textbf{end for} \\ \qquad f^{\ell+1,0} = R_\ell^{\ell+1}([f^{\ell,0}, \ldots, f^{\ell,\nu_\ell}]), \\ \textbf{end for} \\ H_0(f) = R_{\text{ave}}(f^{L,\nu_\ell}). \end{cases} \tag{7.5}$$

Here, $[f^{\ell,0}, \ldots, f^{\ell,i}]$ represents the collection of all the previous output in ℓ-th grids after i-th smoother in the channel dimension, and

$$\theta^{\ell,i} = ([\theta^{\ell,i}]_0, \ldots, [\theta^{\ell,i}]_{i-1}) : \mathbb{R}^{m_\ell \times n_\ell \times (\sum_{j=0}^{i-1} k_j)} \mapsto \mathbb{R}^{m_\ell \times n_\ell \times k_i}, \tag{7.6}$$

where $[\theta^{\ell,i}]_j : \mathbb{R}^{m_\ell \times n_\ell \times k_j} \mapsto \mathbb{R}^{m_\ell \times n_\ell \times k_i}$ for $j = 0 : i-1$. Roughly speaking, the main iterative step in DenseNet is almost the same as the semi-iterative iterative process (3.21) if we ignore the nonlinear activation function σ and fix the channel dimension k_j.

In our paper, we mainly consider the connection between MgNet and ResNet type models from the viewpoint of single step (residual correction) iterative scheme. In addition, we also make some discussion about the relationship between Multi-step MgNet and DenseNet using the idea of multi-iterative method.

The development of the first three models is often shown with the following diagrams (see Figure 3):

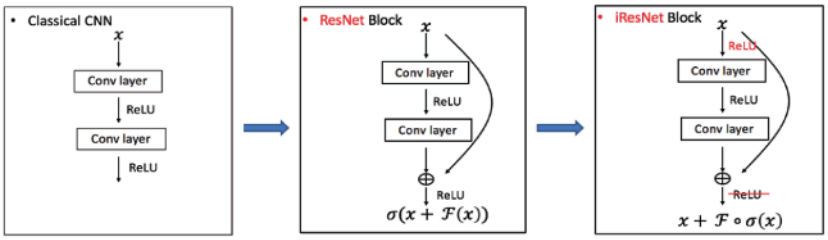

Figure 3 (Color online) Comparison of CNN Structures

Without loss of generality, we extract the key feedforward steps on the same grid in different CNN models as follows.

Classic CNN.

$$f^{\ell,i} = \xi^i \circ \sigma(f^{\ell,i-1}) \quad \text{or} \quad f^{\ell,i} = \sigma \circ \xi^i(f^{\ell,i-1}). \tag{7.7}$$

ResNet.

$$f^{\ell,i} = \sigma(f^{\ell,i-1} + \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i}(f^{\ell,i-1})). \tag{7.8}$$

iResNet.

$$f^{\ell,i} = f^{\ell,i-1} + \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1}). \tag{7.9}$$

DenseNet.

$$f^{\ell,i} = \sigma\bigg(\sum_{j=0}^{i-1} [\theta^{\ell,i}]_j * f^{\ell,j}\bigg). \tag{7.10}$$

8 Variants and generalizations of MgNet

The MgNet model algorithm is one very basic algorithm and it can be generalized in many different ways. It can also be used as a guidance to modify and extend many existing CNN models.

The following result show how MgNet is related to the iResNet [19].

Theorem 8.1. *The MgNet model Algorithm* 3*, with* $A = \xi^\ell$ *and* $B^{\ell,i} = \sigma \circ \eta^{\ell,i} \circ \sigma$*, admits the following identities*:

$$f^{\ell,i} = f^{\ell,i-1} - \xi^\ell \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1}), \quad i = 1 : \nu_\ell, \tag{8.1}$$

where

$$f^{\ell,i} = f^\ell - \xi^\ell(u^{\ell,i}). \tag{8.2}$$

Furthermore, (8.1) *represents iResNet* [19] *as shown in* (7.9)*.*

Proof. Because of the linearity of ξ^ℓ and invariant within the same grid ℓ, we can apply ξ^ℓ on both sides of (6.4) and minus with f^ℓ. Thus we have

$$f^\ell - \xi^\ell(u^{\ell,i}) = f^\ell - \xi^\ell(u^{\ell,i-1}) - \xi^\ell \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^\ell - \xi^\ell(u^{\ell,i-1})).$$

This finishes the proof with definition in (8.2). □

The above result is very simple but critically important. In view of Theorem 8.1, it shows how multigrid and CNN are intimately related. Furthermore, it provides a different version of iResNet, which can be viewed as the dual version of the original iResNet. This relation is quit similar to the dual relation of u and f in multigrid method [51].

Lemma 8.2. *The ResNet* [18] *step as in* (7.8) *admits the following relation*:

$$\tilde{f}^{\ell,i} = \sigma(\tilde{f}^{\ell,i-1}) - \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(\tilde{f}^{\ell,i-1}), \tag{8.3}$$

where

$$\tilde{f}^{\ell,i} = f^{\ell,i-1} - \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i}(f^{\ell,i-1}). \tag{8.4}$$

Proof. First, we apply $\xi^{\ell,i+1} \circ \sigma \circ \eta^{\ell,i+1}$ on both sides of (7.8) and get

$$\xi^{\ell,i+1} \circ \sigma \circ \eta^{\ell,i+1}(f^{\ell,i}) = \xi^{\ell,i+1} \circ \sigma \circ \eta^{\ell,i+1} \circ \sigma(\tilde{f}^{\ell,i}). \tag{8.5}$$

Subtract $f^{\ell,i}$ from both sides of the above equation and recall the definition in (8.4). We have

$$\tilde{f}^{\ell,i+1} = f^{\ell,i} - \xi^{\ell,i+1} \circ \sigma \circ \eta^{\ell,i+1} \circ \sigma(\tilde{f}^{\ell,i}).$$

By the definition of $f^{\ell,i} = \sigma(\tilde{f}^{\ell,i})$, we finish this proof. □

We call the above form (8.3) as σ-ResNet. Similar to the MgNet we replace $\xi^{\ell,i}$ by ξ^{ℓ} and get the next Mg-ResNet form as:

$$f^{\ell,i} = \sigma(f^{\ell,i-1}) - \xi^{\ell} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1}). \tag{8.6}$$

If we take these pooling and prolongation operators as discussed in the previous sections and focus on the iterative forms on a certain grid ℓ, we may compare them as below (see Table 1).

We can have these connections for all iterative scheme in data space:

$$\text{ResNet} \xleftrightarrow{(8.4)} \sigma\text{-ResNet} \xleftrightarrow{\xi^{\ell,i}\leftrightarrow\xi^{\ell}} \text{Mg-ResNet} \xleftrightarrow{\sigma(f^{\ell,i-1})\leftrightarrow f^{\ell,i-1}} \text{Mg-iResNet} \xleftrightarrow{\xi^{\ell}\leftrightarrow\xi^{\ell,i}} \text{iResNet}. \tag{8.7}$$

In this sense, these MgNet related models can be understood as models between iResNet and ResNet. In addition, all these models can be understood as iteration in the data space as a dual relationship with feature space as MgNet.

The rationality of replacing $\xi^{\ell,i}$ by layer independent ξ^{ℓ} may be justified by the following theorem.

Theorem 8.3. *On each grid $\mathcal{T}_\ell$, the following hold*:

(1) *Any CNN model with*

$$f^{\ell,i} = \chi^{\ell,i} \circ \sigma(f^{\ell,i-1}) \tag{8.8}$$

can be written as

$$f^{\ell,i} = \sigma(f^{\ell,i-1}) - \xi^{\ell} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1}). \tag{8.9}$$

(2) *Any CNN model with*

$$f^{\ell,i} = \sigma \circ \chi^{\ell,i}(f^{\ell,i-1}) \tag{8.10}$$

can be written as

$$f^{\ell,i} = \sigma(f^{\ell,i-1} - \xi^{\ell} \circ \sigma \circ \eta^{\ell,i}(f^{\ell,i-1})). \tag{8.11}$$

Proof. Let us prove the first case as an example. The second case can be proven with the same process.

With the similar structure in MgNet, we can take

$$\xi^{\ell} = \hat{\delta}^{\ell} := [\hat{\delta}_1, \ldots, \hat{\delta}_{c_\ell}] \tag{8.12}$$

and

$$\eta^{\ell,i} = [\mathrm{id}_{c_\ell}, -\mathrm{id}_{c_\ell}] \circ (\chi^{\ell,i} - \mathrm{id}_{c_\ell}). \tag{8.13}$$

Here,

$$\mathrm{id}_{c_\ell} : \mathbb{R}^{n_\ell \times n_\ell \times c_\ell} \mapsto \mathbb{R}^{n_\ell \times n_\ell \times c_\ell} \tag{8.14}$$

is the identity map and

$$\hat{\delta}_k : \mathbb{R}^{n_\ell \times n_\ell \times 2c_\ell} \mapsto \mathbb{R}^{n_\ell \times n_\ell} \tag{8.15}$$

Table 1 Comparison for MgNet and ResNet type iterative forms

Primal-Dual	Model	Iterative forms
Feature space	Abstract-MgNet	Solving $A^{\ell}(u^{\ell}) = f^{\ell}$
	Single step MgNet	$u^{\ell,i} = u^{\ell,i-1} + B^{\ell,i}(f^{\ell} - A^{\ell}(u^{\ell,i-1}))$
	Multi-step MgNet	$u^{\ell,i} = \sum_{j=0}^{i-1} \alpha_j^{\ell,i}(u^{\ell,j} + B_j^{\ell,i}(f^{\ell} - A^{\ell}(u^{\ell,j})))$
	Chebyshev-semi MgNet	$u^{\ell,i} = \omega^{\ell,i}(u^{\ell,i-1} + B^{\ell,i}(f^{\ell} - A^{\ell}(u^{\ell,i-1}))) + (1-\omega^{\ell,i})u^{\ell,i-2}$
	MgNet	$u^{\ell,i} = u^{\ell,i-1} + \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell} - \xi^{\ell}(u^{\ell,i-1}))$
Data space	iResNet	$f^{\ell,i} = f^{\ell,i-1} - \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1})$
	Mg-iResNet	$f^{\ell,i} = f^{\ell,i-1} - \xi^{\ell} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1})$
	Mg-ResNet	$f^{\ell,i} = \sigma(f^{\ell,i-1}) - \xi^{\ell} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1})$
	σ-ResNet	$f^{\ell,i} = \sigma(f^{\ell,i-1}) - \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1})$
	ResNet	$f^{\ell,i} = \sigma(f^{\ell,i-1} - \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i}(f^{\ell,i-1}))$

with

$$\hat{\delta}_k([X,Y]) = -([X]_k + [Y]_k), \tag{8.16}$$

for any $X, Y \in \mathbb{R}^{n_\ell \times n_\ell \times c_\ell}$ and $[X,Y] \in \mathbb{R}^{n_\ell \times n_\ell \times 2c_\ell}$.

First, we see that $\eta^{\ell,i}$ with the above form is a convolution from $\mathbb{R}^{n_\ell \times n_\ell \times c_\ell}$ to $\mathbb{R}^{n_\ell \times n_\ell \times 2c_\ell}$. Then we have a special MgNet model because of the identity

$$\mathrm{ReLU}(x) + \mathrm{ReLU}(-x) = x, \tag{8.17}$$

and the definition of ξ^ℓ, i.e.,

$$\xi^\ell = \hat{\delta}^\ell. \tag{8.18}$$

For more details, we can give an exact form of $\hat{\delta}_k$ as in (8.16) with

$$\hat{\delta}_k = [0, \ldots, 0, -\delta, \ldots, 0; 0, \ldots, 0, -\delta, \ldots, 0], \quad k = 1 : c_\ell, \tag{8.19}$$

where δ is the identity kernel in one channel.

Furthermore, we have

$$\begin{aligned}
[\xi^\ell \circ \sigma \circ [\mathrm{id}_{c_\ell}, -\mathrm{id}_{c_\ell}](x)]_k &= [\xi^\ell \circ \sigma \circ [x, -x]]_k \\
&= \hat{\delta}_k([\sigma(x), \sigma(-x)]) \\
&= -\delta([\sigma(x)]_k) - \delta([\sigma(-x)]_k) \\
&= -(\sigma([x]_k) + \sigma(-[x]_k)) \\
&= -[x]_k.
\end{aligned} \tag{8.20}$$

That is to say,

$$\xi^\ell \circ \sigma \circ [\mathrm{id}_{c_\ell}, -\mathrm{id}_{c_\ell}] = -\mathrm{id}_{c_\ell}. \tag{8.21}$$

Then the modified dual form of MgNet in (8.3) becomes

$$\begin{aligned}
f^{\ell,i} &= \sigma(f^{\ell,i-1}) - \xi^{\ell,i} \circ \sigma \circ \eta^{\ell,i} \circ \sigma(f^{\ell,i-1}) \\
&= \sigma(f^{\ell,i-1}) - (\xi^\ell \circ \sigma \circ [\mathrm{id}_{c_\ell}, -\mathrm{id}_{c_\ell}]) \circ (\chi^{\ell,i} - \mathrm{id}_{c_\ell}) \circ \sigma(f^{\ell,i-1}) \\
&= \sigma(f^{\ell,i-1}) + (\chi^{\ell,i} - \mathrm{id}_{c_\ell}) \circ \sigma(f^{\ell,i-1}) \\
&= \chi^{\ell,i} \circ \sigma(f^{\ell,i-1}).
\end{aligned} \tag{8.22}$$

This covers (8.9). □

Remark 8.4. Theorem 8.3 shows that general CNN in the forms of either (8.8) or (8.10) can be written recast as (8.9) or (8.11) with the data-feature mapping $A^\ell = \xi^\ell$ that is not only independent of the layers, but is actually given a priori as in (8.12). In view of Theorems 8.1 and 8.3, the classic CNN models can be essentially recovered from MgNet by choosing ξ^ℓ a priori as in (8.12). We believe that general and well-defined mathematical structure of MgNet would provide mathematical insights for understanding and developing these CNN models.

9 Numerical experiments

In this section, we present some numerical results to illustrate the efficiency and potential of MgNet as described in Algorithm 3.

9.1 Data sets and model structure

We choose CIFAR-10 and CIFAR-100 [26] as two data sets for numerical tests. Here, the CIFAR-10 dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. We split these two data sets with 50,000 training images and 10,000 test images.

We will mainly carry out a comparison with study between MgNet and ResNet [18] on these two data sets, so we choose some similar process techniques in ResNet such as there will be an average pooling before linear regression layers

$$R_{\text{ave}} : \mathbb{R}^{m_{J-1}\times n_{J-1}\times c_{J-1}} \mapsto \mathbb{R}^{c_{J-1}}. \tag{9.1}$$

Here, we can recover this average operator by taking $\nu_J = 0$ in MgNet and

$$u^J = u^{J,0} = \Pi_{J-1}^J u^{J-1,\nu_{J-1}} \in \mathbb{R}^{c_{J-1}}$$

with $\Pi_{J-1}^J = R_{\text{ave}}$. This can be true also thanks to our structure that

$$c_{u,\ell} = c_u, \quad 1 \leqslant \ell \leqslant J. \tag{9.2}$$

Given an image f, similar to ResNet, we apply our MgNet as follows:

$$y = S \circ \theta \circ u^J(f), \tag{9.3}$$

where $u^J(f)$ is the output from our MgNet as described in Algorithm 3, S is the soft-max mapping in (2.8) and

$$\theta : \mathbb{R}^{c_u} \mapsto \mathbb{R}^{\kappa}, \tag{9.4}$$

represents a fully linear layer with $\kappa = 10$ for CIFAR-10 and $\kappa = 100$ for CIFAR-100.

We will make the following choice of hyperparameters for the MgNet:

• f_{in}: data initialization process. Similar to ResNet, we take $f_{\text{in}}(f) = \sigma \circ \theta^0(f)$ as discussed in Subsection 6.1 and Section 7.

• J: the number of grids. As all images in CIFAR-10 or CIFAR-100 are $32 \times 32 \times 3$, we choose $J = 5$ to be consistent with ResNet.

• ν_ℓ: the number of smoothings in each grids. To be consistent with ResNet-18 or ResNet-34 we choose $\nu_\ell = 2$ or $\nu_\ell = 4$.

• c_u and c_f: the number of feature and data channels.

• A^ℓ: the data-feature mapping. We choose the linear case in (3.17).

• $B^{\ell,i}$: the feature extractor. We choose the constant extractors as in Subsection 6.5.

• $R_\ell^{\ell+1}$: the restriction operator in (6.6). Here, we choose it as a convolution with stride 2 which needs to be trained.

• $\Pi_\ell^{\ell+1}$: the interpolation operator in (6.5). Here, we compare these next three different choices:

1. Π_0: zero interpolation, i.e., $\Pi_\ell^{\ell+1} = 0$;
2. Π_1: convolution with stride 2 which needs to be trained;
3. Π_2: channel-wise interpolation as in (6.24) with $\bar{\Pi}_\ell^{\ell+1}$ as a convolution with one channel and stride 2 which also needs to be trained.

9.2 Training algorithm

While there are many different choices of training algorithms [2], in our test, we adopt the popular stochastic gradient descent (SGD) with mini-batch and momentum for cross-entropy loss function (see Algorithm 5).

Here, we have

$$h_i(w_t) = l(H(f_i; w_t), y_i)$$

as defined in (2.11), where w_t denotes all free parameters in MgNet and θ in (9.4). We use the SGD with momentum of 0.9. The mini-batch size is chosen as $m = 128$. The learning rate starts from 0.1 and is divided by 10 for every 30 epochs, and the models are trained for up to $K = 120$ epochs. We adopt batch normalization (BN) after each convolution and before activation, following [23]. Initialization strategy is the same with ResNet as in [17]. We do not use weight decay and dropout. The final Top-1 test accuracy is shown in Table 2.

Algorithm 5 SGD with mini-batch and momentum

1: **Input**: learning rate η_t, batch size m, parameter Initialization w_0, number of epochs K.
2: **for** Epoch $k = 1 : K$ **do**
3: Shuffle data and get mini-batch $B_1, \ldots, B_{\frac{N}{m}}$, choose mini-batch as: B_{i_t} with

$$i_t \equiv t \mod \left(\frac{N}{m}\right).$$

4: Compute the gradient on B_{i_t}:

$$g_t = \nabla_w \frac{1}{m} \sum_{i \in B_{i_t}} h_i(w_t).$$

5: Compute the momentum:

$$v_t = \alpha v_{t-1} - \eta_t g_t, \quad v_0 = 0. \tag{9.5}$$

6: Update w:

$$w_{t+1} = w_t + v_t. \tag{9.6}$$

7: **end for**

Table 2 ResNet and MgNet on CIFAR-10 and CIFAR-100. Our methods are named with ν_ℓ, $(c_u,\ c_f)$ and $\Pi_\ell^{\ell+1}$ by definition above

Models	CIFAR-10	CIFAR-100	Params
ResNet-18	92.24	71.96	11.2M
ResNet-34	92.80	71.93	21.3M
$2, (256, 256), \Pi_0$	92.02	68.29	7.1M
$2, (256, 256), \Pi_1$	93.04	72.32	8.9M
$2, (256, 512), \Pi_1$	93.20	72.42	19.5M
$2, (256, 512), \Pi_2$	93.53	74.26	17.7M

From the above numerical results, we find that the modified CNN models based on MgNet structure have competitive and sometimes better performance in comparison with standard ResNet models when applied to both CIFAR-10 and CIFAR-100 data sets. Generally speaking, the more channels the better performance we can achieve (see WideResNet [52] for similar observation). Furthermore, Π_1 and Π_2 work better than Π_0, and Π_2 can even work better than Π_1 with fewer parameters for big enough channel numbers.

10 Concluding remarks

By carefully studying the connections between the traditional multigrid method and the convolutional neural network (especially the ResNet type) models, the MgNet established in this paper provides a unified framework that connects both multigrid and CNN in a technical level. Comparing with other existing works that discuss the connection between multigrid and CNN, MgNet goes beyond formal or qualitative comparisons and identifies key model components that play the same corresponding roles, from an abstract viewpoint, for these two different methodologies. As a result, how and why CNN models work can be mathematically understood in a similar fashion as for multigrid method which has a much more mature and better developed theory. Motivated from various known techniques from multigrid method, many variants and improvements of CNN can then be naturally obtained. For example, as demonstrated from our preliminary numerical experiments, the resulting modified CNN models equipped with fewer weights and hyperparameters actually exhibit competitive and sometimes better performance than standard ResNet models.

The MgNet framework opens a new door to the mathematical understanding, analysis and improvements of deep learning models. The very preliminary results presented in this paper have demonstrated the great potential of MgNet from both theoretical and practical viewpoints. Obviously many aspects of

MgNet should be further explored and expect to be much improved. In fact, only very few techniques from multigrid method have been tried in this paper and many more in-depth techniques from multigrid require further study for deep neural networks, especially CNN. In particular, we believe that the MgNet framework will lead to improved CNN that only has a small fraction of the number of weights that are required by the current CNN. On the other hand, the techniques in CNN can also be used to develop new generation of multigrid and especially algebraic multigrid methods [51] for solving partial differential equations. Our ongoing works have demonstrated great potentials for research in these directions and many more results will be reported in future papers.

Acknowledgements The first author was supported by the Elite Program of Computational and Applied Mathematics for PhD Candidates of Peking University. The second author was supported in part by the National Science Foundation of USA (Grant No. DMS-1819157) and the US Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics Program (Grant No. DE-SC0014400). The authors thank Xiaodong Jia for his help with the numerical experiments.

References

1 Barron A R. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans Inform Theory, 1993, 39: 930–945
2 Bottou L, Curtis F E, Nocedal J. Optimization methods for large-scale machine learning. SIAM Rev, 2018, 60: 223–311
3 Chang B, Meng L, Haber E, et al. Multi-level residual networks from dynamical systems view. ArXiv:1710.10348, 2017
4 Chen Y, Li J, Xiao H, et al. Dual path networks. In: Advances in Neural Information Processing Systems, vol. 30. Long Beach: Neural Information Processing Systems Foundation, 2017, 4467–4475
5 Cybenko G. Approximation by superpositions of a sigmoidal function. Math Control Signals Systems, 1989, 2: 303–314
6 Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database. In: Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition. Long Beach: IEEE, 2009, 248–255
7 E W. A proposal on machine learning via dynamical systems. Commun Math Stat, 2017, 5: 1–11
8 E W, Wang Q. Exponential convergence of the deep neural network approximation for analytic functions. Sci China Math, 2018, 61: 1733–1740
9 Ellacott S W. Aspects of the numerical analysis of neural networks. Acta Numer, 1994, 3: 145–202
10 Golub G H, Van Loan C F. Matrix Computations, 3rd ed. Baltimore: Johns Hopkins University Press, 2012
11 Gomez A N, Ren M, Urtasun R, et al. The reversible residual network: Backpropagation without storing activations. In: Advances in Neural Information Processing Systems, vol. 30. Long Beach: Neural Information Processing Systems Foundation, 2017, 2214–2224
12 Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge: MIT Press, 2017
13 Haber E, Ruthotto L, Holtham E. Learning across scales—A multiscale method for convolution neural networks. ArXiv:1703.02009, 2017
14 Hackbusch W. Iterative Solution of Large Sparse Systems of Equations. New York: Springer, 1994
15 Hackbusch W. Multi-grid Methods and Applications. Heidelberg: Springer, 2013
16 He J, Li L, Xu J, et al. ReLU deep neural networks and linear finite elements. ArXiv:1807.03973, 2018
17 He K, Zhang X, Ren S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the 2015 IEEE International Conference on Computer Vision. Santiago: IEEE, 2015, 1026–1034
18 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016, 770–778
19 He K, Zhang X, Ren S, et al. Identity mappings in deep residual networks. In: Proceedings of the 14th European Conference on Computer Vision. Amsterdam: Springer, 2016, 630–645
20 Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks, 1989, 2: 359–366
21 Hsieh J T, Zhao S, Eismann S, et al. Learning neural PDE solvers with convergence guarantees. In: Proceedings of the 7th International Conference on Learning Representations. Https://openreview.net/forum?id=rklaWn0qK7, 2019
22 Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks. In: Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition. Honolulu: IEEE, 2017, 4700–4708
23 Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ArXiv:1502.03167, 2015

24 Katrutsa A, Daulbaev T, Oseledets I. Deep multigrid: Learning prolongation and restriction matrices. ArXiv:1711.03825, 2017
25 Ke T W, Maire M, Stella X Y. Multigrid neural architectures. ArXiv:1611.07661, 2016
26 Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. Technical report. Toronto: University of Toronto, 2009
27 Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25. Lake Tahoe: Neural Information Processing Systems Foundation, 2012, 1097–1105
28 Larsson G, Maire M, Shakhnarovich S. Fractalnet: Ultra-deep neural networks without residuals. ArXiv:1605.07648, 2016
29 LeCun L, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, vol. 86. New York: IEEE, 1998, 2278–2324
30 LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436
31 Li Z, Shi Z. A flow model of neural networks. ArXiv:1708.06257v2, 2017
32 Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection. In: Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition. Honolulu: IEEE, 2017, 2117–2125
33 Liu D, Wen B, Liu X, et al. When image denoising meets high-level vision tasks: A deep learning approach. ArXiv:1706.04284, 2017
34 Long Z, Lu Y, Dong B. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. ArXiv:1812.04426, 2018
35 Long Z, Lu Y, Ma X, et al. PDE-Net: Learning PDEs from data. In: Proceedings of the 35th International Conference on Machine Learning. Stockholm: PMLR, 2018, 3214–3222
36 Lu Y, Zhong A, Li Q, et al. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In: Proceedings of the 35th International Conference on Machine Learning. Stockholm: PMLR, 2018, 3282–3291
37 Mao X, Shen C, Yang Y B. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: Advances in Neural Information Processing Systems, vol. 29. Barcelona: Neural Information Processing Systems Foundation, 2016, 2802–2810
38 Milletari F, Navab N, Ahmadi S A. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: Proceedings of 2016 4th International Conference on 3D Vision. Stanford: IEEE, 2016, 565–571
39 Montanelli H, Du Q. Deep ReLU networks lessen the curse of dimensionality. ArXiv:1712.08688, 2017
40 Nair V, Hinton G E. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning. Haifa: PMLR, 2010, 807–814
41 Noh H, Hong S, Han B. Learning deconvolution network for semantic segmentation. In: Proceedings of the 2015 IEEE International Conference on Computer Vision. Santiago: IEEE, 2015, 1520–1528
42 Pinkus A. Approximation theory of the MLP model in neural networks. Acta Numer, 1999, 8: 143–195
43 Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: Proceedings of Medical Image Computing and Computer-Assisted Intervention. Munich: Springer, 2015, 234–241
44 Shaham U, Cloninger A, Coifman R R. Provable approximation properties for deep neural networks. Appl Comput Harmon Anal, 2018, 44: 537–557
45 Siegel J W, Xu J. On the approximation properties of neural networks. ArXiv:1904.02311, 2019
46 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. ArXiv:1409.1556, 2014
47 Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition. Boston: IEEE, 2015, 1–9
48 Xu J. Iterative methods by space decomposition and subspace correction. SIAM Rev, 1992, 34: 581–613
49 Xu J. The Finite Element Methods. Http://www.multigrid.org/wiki, 2019
50 Xu J, Zikatanov L. The method of alternating projections and the method of subspace corrections in Hilbert space. J Amer Math Soc, 2002, 15: 573–597
51 Xu J, Zikatanov L. Algebraic multigrid methods. Acta Numer, 2017, 26: 591–721
52 Zagoruyko S, Komodakis N. Wide residual networks. ArXiv:1605.07146, 2016
53 Zhang T, Qi G J, Xiao B, et al. Interleaved group convolutions. In: Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition. Honolulu: IEEE, 2017, 4373–4382
54 Zhang X, Li Z, Change Loy C, et al. Polynet: A pursuit of structural diversity in very deep networks. In: Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition. Honolulu: IEEE, 2017, 3900–3908
55 Zhou D X. Universality of deep convolutional neural networks. ArXiv:1805.10769, 2018