Convolutional Neural Networks combined with Runge-Kutta Methods

Mai Zhu
Northeastern University
Shenyang, Liaoning, China
zhumai@stumail.neu.edu.cn

Bo Chang
University of British Columbia & Xtract Technologies Inc.
Vancouver, Canada
bchang@stat.ubc.ca

Chong Fu
Northeastern University
Shenyang, Liaoning, China
fuchong@mail.neu.edu.cn

Abstract

A convolutional neural network for image classification can be constructed mathematically since it is inspired by the ventral stream in visual cortex which can be regarded as a multi-period dynamical system. In this paper, a novel approach is proposed to construct network models from the dynamical systems view. Since a pre-activation residual network can be deemed an approximation of a time-dependent dynamical system using the Euler method, higher order Runge-Kutta methods (RK methods) can be utilized to build network models in order to achieve higher accuracy. The model constructed in such a way is referred to as the Runge-Kutta Convolutional Neural Network (RKNet). RK methods also provide an interpretation of Dense Convolutional Networks (DenseNets) from the dynamical systems view. The proposed methods are evaluated on the benchmark datasets: CIFAR-10/100 and ImageNet. The experimental results show that the RK Nets achieve similar accuracy with the state-of-the-art network models, DenseNets. Moreover, the experimental results are consistent with the theoretical properties of RK methods and support the dynamical systems interpretation.

1 Introduction

So far, many efforts have been devoted to searching for the connection between artificial neural networks and the visual system of the brain in order to gain insight into the construction of new network models. Inspired by neuroscience, a recurrent convolutional neural network (RCNN) was proposed for object recognition because recurrent connections are abundant in the visual system of the brain [16]. Residual Networks (ResNets) which are feed-forward network models with skip connections have achieved great success on several vision benchmarks in 2015 [9]. Recently, researchers have also studied the relations between ResNets, Recurrent Neural Networks (RNNs) and the primate visual cortex. The proposed multi-state time-invariant recurrent networks that reflect the multi-stage processing in the primate visual cortex achieve competitive performance as very deep residual networks [17]. Therefore, biologically-plausible recurrent architectures should be applicable in image classification. However, the state-of-the-art models such as DenseNets have no recurrent connections [12]. Therefore, a theoretically grounded interpretation is needed, not only to understand these models but also to guide future model construction.

However, only combining CNNs with neuroscience is not adequate; mathematics should play an important role in designing the network topology as well. If the ventral stream in the visual cortex is
deemed several time-dependent dynamical systems, there should be a series of ordinary differential equations (ODEs) to describe these systems. Runge-Kutta methods (RK methods) are widely-used procedures to solve ODEs in numerical analysis [1]. They are iterative methods that can represent recurrence. Consequently, these methods could be used to build network models for visual processing. Furthermore, the mathematical structure of the visual system might be better understood along this direction.

The neural network community has long been aware of the numerical methods for dynamical systems. Runge-Kutta Neural Network (RKNN) is proposed for identification of unknown dynamical systems in high accuracy [29], but it has not been used to model the visual system nor been extended to convolutional networks. In addition, Euler method, a first-order RK method, has been employed to explain ResNets with full pre-activation [10] from the dynamical systems view [3, 7]. In numerical analysis, a higher-order RK method usually achieve a lower truncation error. One of the major contributions of the paper is a novel and effective neural network architecture inspired by the RK methods.

In order to apply RK methods to the image classification problem, three assumptions are made throughout the paper. Firstly, the image classification procedure is multi-period corresponding to the several visual areas in the ventral stream. Secondly, each period in classification process is modeled by a time-dependent dynamical system. Thirdly, there is no connection among non-adjacent periods. In other words, the connections among non-adjacent visual areas are ignored. Based on these assumptions, a novel network model called RKNet is proposed.

In an RKNet, a period is composed of the recurrence of time-steps. A particular explicit RK method is adopted throughout the time-steps in a period to approximate the system state. RK methods have several stages within a time-step and utilize the ODE to calculate the increment in each step. In an RKNet, the increment is approximated by a convolutional subnetwork due to the versatility of neural networks on approximation.

Another contribution of this paper is a theoretical interpretation of DenseNets from the dynamical systems view. The dense connection in DenseNet resembles the relationship among increments in the stages in RK methods. Under some conditions, DenseNets can be formulated as approximating dynamical systems using multi-stage RK methods. We also propose a method to convert a DenseNet to an RKNet. Furthermore, DenseNets have only one time-step in each period, whereas RKNs are more general and can have multiple time-steps in each period. An RKNet with multiple time-steps induces recurrence, which is consistent with the biological motivation and the recurrent connections in the brain.

We evaluate the performance of RKNs converted from DenseNets on benchmark datasets including CIFAR-10, CIFAR-100 [13] and ILSVRC2012 classification dataset [25]. Experimental results show that RKNs achieve similar accuracy as the DenseNets. The accuracy of RKNs can be further improved by increasing the number of stages in RK methods.

The rest of the paper is organized as follows. The related work is reviewed in Section 2. The architecture of RKNs, the dynamical systems interpretation of DenseNets, and the conversion from DenseNets to RKNs are described in Section 3. The performance of RKNs is evaluated in Section 4. The conclusion and future work is described in Section 5.

2 Related work

ResNets have gained much attention over the past few years since they have obtained impressive performance on many challenging image tasks, such as ImageNet [25] and COCO object detection [18]. ResNets are deep feed-forward networks with the shortcuts as identity mappings. ResNets with pre-activation can be regarded as an unfolded shallow RNN, which implements a discrete dynamical system [17]. This dynamical system represents the processing of visual information through the ventral stream of the primate visual cortex. The ventral stream is associated with visual perception and passes several visual areas, i.e. V1, V2, V4, IT [6]. Each visual area can be considered a processing period. Furthermore, it has been shown that recurrent connections exist richly in the neocortex by anatomical evidence [4]. These recurrent connections within each visual area are modeled as a recurrent network in [17]. Some biologically plausible multi-state recurrent networks corresponding to the multi-stage processing in the ventral stream have been evaluated on CIFAR-10 in that paper.
Note that the state and stage in their paper both denote period. The paper [17] provides a novel point of view for explaining pre-activation ResNets via neuroscience and dynamical systems.

Recently, more work has emerged to connect dynamical systems with deep learning [5] or ResNets in particular [2, 3, 7, 15, 19, 20, 28]. The paper [5] proposes ideas about using continuous dynamical systems as a tool for machine learning. The paper [3] proposes a novel method for accelerating ResNets training based on the interpretation of ResNets from dynamical systems view in [7]. The paper [15] presents a training algorithm which can be used in the context of ResNets. The paper [20] proposes a linear multi-step architecture based on ResNets. In addition, research combining dynamical system identification and RK methods with neural networks for scientific computing has emerged recently [22, 23, 24], introducing physics informed neural networks with automatic differentiation.

DenseNets are the state-of-the-art network models after ResNets [12]. The dense connection is the main difference from the previous models. There are direct connections from a layer to all subsequent layers in a dense block in order to allow better information and gradient flow. There is no interpretation of DenseNets from dynamical systems view yet.

Given that a visual area can be thought of as a time-dependent dynamical system, there should be a set of ODEs that describes this system. Consequently, mathematical tools could be employed to construct network models. RK methods are commonly used to solve ODEs in numerical analysis [1]. Moreover, the methods are iterative; the iterations of time-steps resemble the recurrent connections within visual areas. Therefore, RK methods are ideal tools to integrate neuroscience theory and dynamical systems interpretation. Note that the Euler method used in [3, 7] is a first-order RK method. Higher order RK methods usually achieve lower truncation error.

RK methods have been adopted to construct neural networks, which are known as RKNN, for identification of unknown dynamical systems described by ODEs [29]. In that paper, neural networks are classified into two categories: (1) a network that directly learns the state trajectory of a dynamical system is called a direct-mapping neural network (DMNN); (2) a network that learns the rate of change of system states is called a RKNN. Hence, AlexNet [14], VGGNet [26], GoogLeNet [27] and ResNet [9] all belong to DMNNs. Specifically, the original ResNet [9] is a DMNN because of the ReLU layer after the addition operation. As a result, the ResNet building block learns the state trajectory directly, not the rate of change of the system states. On the contrary, a ResNet with pre-activation [10] is an RKNN.

RKNNs are proposed to eliminate several drawbacks of DMNNs, such as the difficulty in obtaining high accuracy for the multi-step prediction of state trajectories. It has been shown theoretically and experimentally that the RKNN has higher prediction accuracy and better generalization capability than the conventional DMNN [29].

Therefore, it is reasonable to believe that RK methods can be adopted to design effective network architectures for image classification problems. Additionally, the RK methods might improve the performance of image classification since the convolutional subnetworks are able to approximate the rate of change of the dynamical system states more precisely.

3 RK Nets

3.1 Runge-Kutta methods

A time-dependent dynamical system can be described by the following ODE:

\[
\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.
\]  

(1)

where \( y \) is a vector representing the system state. The dimension of \( y \) should be equal to the dimension of the dynamical system. The ODE represents the rate of change of the system states. The rate of change is a function of time and the current system state. RK methods utilize the rate of change calculated from the ODE to approximate the increment in each time-step, and then obtain the predicted final state at the end of each step. RK methods are numerical methods originated from Taylor series. There are two types of RK methods: explicit and implicit. Since the explicit RK methods have a simpler form and lower computational cost than the implicit methods, they are
employed in the RKNet. The family of explicit RK methods is given by the following equations:

\[ y_{n+1} = y_n + h \sum_{i=1}^{s} b_i z_i, \quad t_{n+1} = t_n + h, \]

(2)

where

\[ z_1 = f(t_n, y_n), \]
\[ z_2 = f(t_n + c_2 h, y_n + h (a_{21} z_1)), \]
\[ z_3 = f(t_n + c_3 h, y_n + h (a_{31} z_1 + a_{32} z_2)), \]
\[ \ldots \]
\[ z_i = f \left( t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} z_j \right). \]

(3)

In Eq. (2), \( y_{n+1} \) is an approximation of the solution to Eq. (1) at time \( t_{n+1} \), i.e. \( y(t_{n+1}) \); \( y_0 \) is the input initial value; \( h \sum_{i=1}^{s} b_i z_i \) is the increment of system state \( y \) from \( t_n \) to \( t_{n+1} \); \( \sum_{i=1}^{s} b_i z_i \) is the estimated slope which is the weighted average of the slopes computed in different stages. The positive integer \( s \) is the number of \( z_i \), i.e. the number of stages of the RK method. Eq. (3) is the general formula of \( z_i \). \( h \) is the time-step size which can be adaptive for different time-steps but must be fixed across stages within a time-step.

In numerical analysis, \( s \), \( a_{ij} \), \( b_i \) and \( c_i \) in Eq. (2) and (3) need to be prespecified for a particular RK method. Different RK methods have different truncation errors which are denoted by the order: an order \( p \) indicates that the local truncation error is \( O(h^{p+1}) \). If an explicit \( s \)-stage RK method has order \( p \), then \( s \geq p \); if \( p \geq 5 \), then \( s > p \). Therefore, more stages may achieve higher orders, i.e. lower truncation errors. The Euler method is a one-stage first-order RK method. In other words, high-order RK methods can be expected to achieve lower truncation errors than Euler method. Thus, the goal of our proposed RKNet is to improve the classification accuracy by taking advantage of high-order RK methods.

It is necessary to specify \( h \) in order to control the error of approximation in common numerical analysis. The varying time-step size can be adaptive to the regions with different rates of change. The error is lower when the average \( h \) is smaller.
3.2 From RK methods to RKNets

There are three components of RKNets: the preprocessor, the multi-periods and the postprocessor. The preprocessor manipulates the raw images and passes the results to the first period; it mainly simulates the transmission of visual information from the eyes to the visual cortex. The postprocessor deals with the output from the last period and then passes the result to the classifier to make a decision. The periods between those two components are divided by the transition layers. These periods can be modeled by time-dependent dynamical systems which are assumed to represent the visual areas in the ventral stream of visual cortex. Each period of a RKNet is divided into \( r \) time-steps as shown in Figure 1. RK methods approximate the final state of every time-step using the rate of change of the system state. Some guiding principles when applying RK methods to RKNets are listed as follows.

Firstly, dimensionality reduction is often carried out to simplify the system identification issue, when the dimension of real dynamical system is too high. The dimension of \( y \) in each period in RKNet is predefined as the multiplication of the size of feature map and the number of channels at the beginning of a period. The dimensions of \( y \) in the same periods of different RKNets can be different due to various degrees of dimensionality reduction. Nevertheless, the dimension of \( y \) is consistent within a period. Secondly, given that there is no explicit ODE for image classification, a convolutional subnetwork is employed to approximate the increment in each time-step. The number of neurons in each hidden layer can be more than the dimension of \( y \). Lastly, the number of time-steps \( r \) in each period is predefined in RKNet but the other coefficients, \( a_{ij}, b_i \) and \( c_i \) in Eq. (2) and (3) are learned by training to minimize the cross-entropy loss. The learned \( h \) is thus considered adaptive. In theory, the adaptive time-step size can achieve higher fitting precision.

A variety of RK methods can be adopted in the different periods of RKNets, but the same RK method is used for all time-steps within one period in a RKNet. The network models are named after the specific method in each period, such as RKNet-\( 3 \times 2.4 \times 1.2 \times 5.1 \times 1 \). The suffix in the name of a RKNet is composed of several \( s \times r \) terms; each stands for the method in corresponding period. \( s \) denotes the number of stages of the RK method and \( r \) is the number of time-steps. The number of such terms equals the total number of periods. \( s \) or \( r \) can vary in different periods. For example, RKNet-\( 3 \times 2.4 \times 1.2 \times 5.1 \times 1 \) has four periods: period one has 2 time-steps and each step has 3 stages; period two has 1 time-step and it has 4 stages; period three has 5 time-steps and each step has 2 stages; period four has 1 time-step and it has 1 stage. We use this notation throughout this paper.

3.3 Connection with DenseNets

In this section, we introduce the architecture of RKNets. For the purpose of constructing a RKNet, the general formula in Eq. (3) is adapted as follows:

\[
z_i = f \left( t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} z_j \right) \\
= f_i \left( y_n + h \sum_{j=1}^{i-1} a_{ij} z_j \right) \\
= F_i \left( y_n, h a_{i1} z_1, \ldots, h a_{i(i-1)} z_{i-1} \right) \\
= g_i \left( y_n, h b_1 z_1, \ldots, h b_{i-1} z_{i-1} \right).
\] (4)

The above transformation first changes the explicit dependence on the time in Eq. (3) to an implicit one. Since the time parameter \( t_n + c_i h \) is different for the different stages, it can be absorbed into \( f_i(\cdot) \), which implicitly depends on time for stage \( i \). Afterward, the summation in the input parameter of \( f_i(\cdot) \) is split into separate terms. \( F_i(\cdot) \) denotes the function of these terms for each stage. After that, adjusting the coefficients of each parameter from \( a_{ij} \) to \( b_j \) yields another function \( g_i(\cdot) \). As a result, \( h b_i z_i \) can be described by the equation below, according to Eq. (4).

\[
h b_i z_i = h b_i g_i \left( y_n, h b_1 z_1, \ldots, h b_{i-1} z_{i-1} \right) \\
= G_i \left( y_n, h b_1 z_1, \ldots, h b_{i-1} z_{i-1} \right),
\] (5)
where $G(t) := h_{b_1}g_{n}(t)$. The sum of $h_{b_1}z_i$ represents the increment in a time-step as shown in Eq. (2). It is crucial to approximate this increment in RKNet. If a convolutional subnetwork is adopted to model $G_i(t)$ in Eq. (5), the most similar network structure is the dense connections in DenseNets. However, the dense blocks must conform to the following rules in RKNets.

**Rule 1** The number of channels of $y_n$ is in the form of $mk$, where $m$ and $k$ are positive integers and $k$ is known as the growth rate in DenseNet literature. The dimension of $y_n$ is the multiplication of the size of feature map and $mk$.

**Rule 2** The $mk$ channels grown after $m$ successive growth are regarded as a group. The $i$th group corresponds to $h_{b_1}z_i$.

**Rule 3** The total number of growth is $ms$, where $s$ is number of stages of RK methods.

In the end, $y_n$ and the groups $h_{b_1}z_i$ for $i = 1, \ldots, s$ are added to obtain $y_{n+1}$. Figure 2 illustrates one time-step of RKNet.

In DenseNets, every dense block together with part of the subsequent computation can be regarded as a period using a $s$-stage RK method with $r = 1$ time-step. The transition layers and the postprocessor contain the summation operation in Eq. (2). There are three periods in some DenseNets while there are four visual areas in the ventral stream. They could be thought of as fusing the four dynamical systems into three periods. This gives an explanation of DenseNets from the dynamical systems view.

Although a general DenseNet does not necessarily meet the rules of RKNets, we can convert it into a RKNet. Since there is no explicit summation in DenseNets as shown in Eq. (2), it is necessary to add a summation layer at the end of each time-step. Furthermore, the number of channels throughout the network need to be adjusted in order to meet the aforementioned rules of dense blocks. In order to keep the original network structure of DenseNet as possible, the number of channels is calculated following the procedure below. The number of convolution filters in transition layers of the DenseNet is divided by the growth rate $k$ and then rounded down to an integer $m$. Afterwards, $mk$ is used as the number of convolution filters in the corresponding transition layers in RKNet. It is also the number of channels of $y_n$ in the following period. In particular, this number is always $2k$ in the first dense block of DenseNets. Finally, the total times of growth in a dense block in DenseNet is divided by the corresponding $m$. The computed result is rounded up to an integer as the number of stages $s$ in the corresponding period in RKNet.

Given a one time-step RKNet model, the number of stages of RK methods $s$ and the numbers of time-steps $r$ can be modified to construct more variants with the same dimensions in the corresponding periods. The growth rate $k$ and $m$ in every period control the dimension. In other words, $s$ and $r$
control depth of the network while \( m \) and \( k \) control the width of the network. More stages, more time-steps and larger dimensions usually lead to higher classification accuracy. However, the complexity of an ODE increases with the increase of dimensions. As a result, the convolutional subnetwork which approximates the increment in a time-step need be more complex for larger dimensions. Hence, the accuracy is also associated with the matching degree of the dimension and the convolutional subnetwork. The unmatched high-dimensional network model may have lower accuracy. Additionally, the training method might affect the classification accuracy too.

4 Experiments

To evaluate the performance of RKNets on image classification and explaining DenseNets from dynamical systems view, experiments are conducted using the proposed network architectures.

4.1 Experimental setup

The RKNets are evaluated on CIFAR-10, CIFAR-100 and ImageNet. The CIFAR-10 dataset contains 60,000 color images of size \( 32 \times 32 \) in 10 classes, with 5,000 training images and 1,000 test images per class. The CIFAR-100 is similar to the CIFAR-10 except that it has 100 classes and 500 training images and 100 test images per class. ImageNet, which is also known as the ILSVRC2012 classification dataset, consists of 1.28 million training images and 50,000 validation images.

To make sure that the comparisons between RKNets and DenseNets are fair, we use the Torch implementation of DenseNet\(^{[21]}\) as our base. Only the network models are modified and all the experiment settings remain the same as what were used for DenseNets. The weights are initialized as in [8]. A weight decay of 0.0001 and a momentum of 0.9 are used. The learning rate is set to 0.1 initially.

On CIFAR-10 and CIFAR-100, the models are trained using stochastic gradient descent with a mini-batch size of 64 and a standard data augmentation scheme adopted in [9]. The models are trained for 300 epochs and the learning rate is divided by 10 at epoch 150 and 225.

\(^{[21]}\)https://github.com/liuzhuang13/DenseNet/tree/master/models
Table 3: Classification errors on ImageNet validation set with a single-crop (224×224). \( k \) is growth rate and \( mk \) is the initial number of channels in each period in RKNet. For each RKNet in this table, \( m_0 = 2, m_1 = 4 \) and \( m_2 = 8 \). All the RK Nets are the variants of corresponding DenseNets in the same row. All the DenseNets in this table have bottleneck layers and their compression rate is 0.5, although their names do not contain “-BC”.

| DenseNet     | Top1 (%) | Top5 (%) | RKNet     | \( m_3 \) | Top1 (%) | Top5 (%) |
|--------------|----------|----------|-----------|-----------|----------|----------|
| DenseNet-121 \((k = 32)\) | 25.02    | 7.71     | RKNet-3×1_3×1_3×1_1×1 | 16        | 25.47    | 7.81     |
| DenseNet-169 \((k = 32)\) | 23.80    | 6.85     | RKNet-3×1_3×1_4×1_2×1 | 20        | 24.12    | 7.17     |
| DenseNet-201 \((k = 32)\) | 22.58    | 6.34     | RKNet-3×1_3×1_6×1_2×1 | 28        | 23.14    | 6.66     |

Table 4: Test errors of RK Nets with varying number of stages, evaluated on CIFAR-10 with data augmentation. All of the models have bottleneck layers. The growth rate \( k \) for dense block is 12. The initial numbers of channels in each period are \( \{2k, 9k, 12k\} \).

| RKNet       | Param (M) | Error (%) |
|-------------|-----------|-----------|
| RKNet-8×1_2×1_2×1 | 0.91      | 4.67      |
| RKNet-8×1_3×1_2×1 | 1.12      | 4.44      |
| RKNet-8×1_3×1_3×1 | 1.47      | 4.43      |
| RKNet-9×1_3×1_3×1 | 1.50      | 4.15      |

On ImageNet, the models are trained with a mini-batch size of 256 for 90 epochs. Scale and aspect ratio augmentation in [27], the standard color augmentation in [14] as well as the photometric distortions in [11] are adopted. The learning rate is divided by 10 every 30 epochs. Weight decay is applied to all weights and biases.

4.2 Experimental results

RK Nets that are converted from DenseNets according to the transformation procedure described in Section 3.3 are evaluated. The network models reported in [12] are used as the base DenseNet models. There are two types of models depending on whether they have bottleneck layers and compression. The bottleneck layer is a \( 1\times1 \) convolution layer which is placed before each \( 3\times3 \) convolution layer. The compression is to compress the number of channels in transition layers at a compression rate that is less than one. A model with bottleneck layers and compression is denoted by DenseNet-BC. In particular, all the DenseNet-BC models for ImageNet dataset is denoted as DenseNet-<depth>. A compression rate of 0.5 is used in the DenseNet-BC models [12]. In addition, the bottleneck layer in them outputs \( 4k \) channels to the following \( 3\times3 \) convolution layer.

The test error of DenseNets/DenseNet-BCs and RK Nets on CIFAR-10 and CIFAR-100 are shown in Table 1 and Table 2 respectively. The top-1 and top-5 errors on ImageNet validation set with a single-crop \((224 \times 224)\) are shown in Table 3. According to the experimental results, all the RK Nets have the similar accuracy with the corresponding DenseNets/DenseNet-BCs on three datasets.

According to the theoretical results, a RK method with more stages has a higher order and a lower truncation error. Therefore, as the number of stages increases, a more precise approximation of the system states in every period leads to more accurate classification. Table 4 shows the number of parameters and classification error on CIFAR-10 for RK Nets with varying number of stages in each period. The empirical results are consistent with the theoretical interpretation.

5 Conclusion

We propose to employ a type of numerical ODE method, the RK method, to construct CNNs for image classification tasks. This gives a theoretical interpretation of the state-of-the-art DenseNet model via the dynamical systems view. The model constructed using the RK methods is referred to as the RKNet, which can be converted from a DenseNet model by enforcing theoretical constraints. The experimental results demonstrate that RK Nets have similar classification accuracy as DenseNets.
Moreover, the experimental results validate the theoretical properties of RK methods and support the dynamical systems interpretation.

The mathematical formulation in this paper is consistent with the biological evidence of the visual system in the brain. With the help of the dynamical systems view and various numerical ODE methods including RK methods, more general neural networks can be constructed. Many aspects of RKNets and the dynamical systems view still require further investigation. We hope this work inspires future research directions.

References

[1] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2008.

[2] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

[3] Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level residual networks from dynamical systems view. In *International Conference on Learning Representations*, 2018.

[4] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience*, volume 806. Cambridge, MA: MIT Press, 2001.

[5] Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[6] Melvyn A Goodale and A David Milner. Separate visual pathways for perception and action. *Trends in neurosciences*, 15(1):20–25, 1992.

[7] Eldad Haber, Lars Ruthotto, Elliot Holtham, and Seong-Hwan Jun. Learning across scales - multiscale methods for convolution neural networks. *arXiv preprint arXiv:1703.02009*, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[11] Andrew G Howard. Some improvements on deep convolutional neural network based image classification. In *International Conference on Learning Representations*, 2014.

[12] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] Quanxiao Li, Long Chen, Cheng Tai, and E Weinan. Maximum principle based algorithms for deep learning. *arXiv preprint arXiv:1710.09513*, 2017.

[16] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[17] Q. Liao and T. Poggio. Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex. *ArXiv e-prints*, April 2016.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[19] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. *arXiv preprint arXiv:1710.09668*, 2017.

[20] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*, 2017.

[21] Geoff Pleiss, Danlu Chen, Gao Huang, Tongcheng Li, Laurens van der Maaten, and Kilian Q Weinberger. Memory-efficient implementation of densenets. *arXiv preprint arXiv:1707.06990*, 2017.
[22] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *arXiv preprint arXiv:1801.06637*, 2018.

[23] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[24] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[28] Bao Wang, Xiyang Luo, Zhen Li, Wei Zhu, Zuoqiang Shi, and Stanley J Osher. Deep learning with data dependent implicit activation function. *arXiv preprint arXiv:1802.00168*, 2018.

[29] Yi-Jen Wang and Chin-Teng Lin. Runge-kutta neural network for identification of dynamical systems in high accuracy. *IEEE Transactions on Neural Networks*, 9(2):294–307, 1998.