A Parareal Architecture for Very Deep Convolutional Neural Network

Chang-Ock Lee, Youngkyu Lee, and Jongho Park

1 Introduction

Due to the large number of layers in deep neural networks (DNNs) [11, 12], DNN training is time-consuming and there are demands to reduce training time these days. Recently, multi-GPU parallel computing has become an important topic for accelerating DNN training [2, 6]. In particular, Günther et al. [6] considered the layer structure of ResNet [8] as the forward Euler discretization of a specific ODE and applied a nonlinear in-time multigrid method [3] by regarding the learning process of the network as an optimal control problem.

In this work, we propose a novel paradigm of multi-GPU parallel computing for DNNs, called *parareal neural network*. In general, DNN has a feed-forward architecture. That is, the output of DNN is obtained from the input by sequential compositions of functions representing layers. We observe that sequential computations can be interpreted as time steps of a time-dependent problem. In the field of numerical analysis, after a pioneering work of Lions et al. [14], there have been numerous researches on parallel-in-time algorithms to solve time-dependent problems in parallel; see, e.g., [5, 15, 16]. Motivated by these works, we present a methodology to transform a given feed-forward neural network to another neural network called parareal neural network which naturally adopts parallel computing. The parareal neural network consists of fine structures which can be processed in parallel and

Chang-Ock Lee
Department of Mathematical Sciences, KAIST, Daejeon 34141, Korea
e-mail: colee@kaist.edu

Youngkyu Lee
Department of Mathematical Sciences, KAIST, Daejeon 34141, Korea
e-mail: lyk92@kaist.ac.kr

Jongho Park
Natural Science Research Institute, KAIST, Daejeon 34141, Korea
e-mail: jongho.park@kaist.ac.kr

a coarse structure which approximates the fine structures by emulating one of the parallel-in-time algorithms called parareal [14].

Note that both the proposed parareal neural network and the work of Günther et al. [6] seem to be very closely related in that they parallelize and accelerate the training of neural networks using a parallel time integration approach. However, unlike the work of Günther et al., the proposed network has the advantage of being more general by focusing on layer propagation in an arbitrary feed-forward network.

The parareal neural network can significantly reduce the time for inter-GPU communication because the fine structures do not communicate with each other but communicate only with the coarse structure. Therefore, the proposed methodology is effective in reducing the elapsed time for dealing with very deep neural networks. Numerical results confirm that the parareal neural network gives similar or better performance to the original network even with less training time.

2 The parareal algorithm

The parareal algorithm proposed by Lions et al. [14] is a parallel-in-time algorithm to solve time-dependent differential equations. For the purpose of description, the following system of ordinary differential equations is considered:

$$\dot{\mathbf{u}}(t) = A\mathbf{u}(t) \ \text{ in } [0,T], \ \ \mathbf{u}(0) = \mathbf{u}_0, \tag{1}$$

where $A\colon \mathbb{R}^m \to \mathbb{R}^m$ is an operator, $T > 0$, and $\mathbf{u}_0 \in \mathbb{R}^m$. The time interval $[0,T]$ is decomposed into N subintervals $0 = T_0 < T_1 < \cdots < T_N = T$. First, an approximated solution $\{U_j^1\}_{j=0}^N$ on the coarse grid $\{T_j\}_{j=0}^N$ is obtained by the backward Euler method. The key step of the parareal algorithm is to correct residuals $\{S_j^k\}_{j=1}^{N-1}$ occurring in each interface. It is well-known that the algorithm converges to the exact solution uniformly [1, 4]. We briefly summarize the parareal in Algorithm 1.

3 Parareal neural networks

In this section, we propose a methodology to design a *parareal neural network* by emulating the parareal algorithm introduced in Section 2 from a given feed-forward neural network. The resulting parareal neural network has an intrinsic parallel structure and is suitable for parallel computation using multiple GPUs with distributed memory simultaneously.

Let $f_\theta\colon X \to Y$ be a feed-forward neural network, where X and Y are the spaces of inputs and outputs, respectively, and θ is a vector consisting of parameters. Since many modern neural networks such as [9, 10, 17] have block-repetitive substructures, we may assume that f_θ can be written as the composition of three functions $C_\delta\colon X \to W_0$, $g_\varphi\colon W_0 \to W_1$, and $h_\varepsilon\colon W_1 \to Y$, i.e.,

Algorithm 1: The parareal algorithm for (1)

Let $\Delta T_j = T_{j+1} - T_j$ and $0 = T_0 < T_1 < \cdots < T_N = T$.
for $j \leftarrow 0$ **to** $N-1$ **do**
  Solve $\dfrac{\mathbf{U}_{j+1}^1 - \mathbf{U}_j^1}{\Delta T_j} = A\mathbf{U}_{j+1}^1, \ \ \mathbf{U}_0^1 = \mathbf{u}_0$
end
for $k \leftarrow 1, 2, \ldots$ **do**
  for $j \leftarrow 0$ **to** $N-1$ ***in parallel*** **do**
    Solve $\dot{\mathbf{u}}_j^k(t) = A\mathbf{u}_j^k(t)$ in $[T_j, T_{j+1}]$, $\mathbf{u}_j^k(T_j) = \mathbf{U}_j^k$.
  end
  for $j \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{S}_{j+1}^k = \mathbf{u}_j^k(T_{j+1}) - \mathbf{U}_{j+1}^k, \mathbf{S}_0^k = 0.$
    Solve $\dfrac{\boldsymbol{\delta}_{j+1}^k - \boldsymbol{\delta}_j^k}{\Delta T_j} = A\boldsymbol{\delta}_{j+1}^k + \mathbf{S}_j^k, \ \boldsymbol{\delta}_0^k = 0.$
    $\mathbf{U}_{j+1}^{k+1} = \mathbf{U}_{j+1}^k + \boldsymbol{\delta}_{j+1}^k.$
  end
end

$$f_\theta = h_\varepsilon \circ g_\varphi \circ C_\delta, \quad \theta = \delta \oplus \varphi \oplus \varepsilon,$$

where W_0 and W_1 are vector spaces, g_φ is a block-repetitive substructure of f_θ with parameters φ, C_δ is a *preprocessing operator* with parameters δ, and h_ε is a *postprocessing operator* with parameters ε. Note that $\oplus$ represents a concatenation.

For appropriate vector spaces $X_0, X_1, \ldots, X_N$, we further assume that g_φ can be partitioned into N subnetworks $\{g_{\varphi_j}^j \colon X_{j-1} \to X_j\}_{j=1}^N$ which satisfy the followings:

- $X_0 = W_0$ and $X_N = W_1$,
- $\varphi = \bigoplus_{j=1}^N \varphi_j$,
- $g_\varphi = g_{\varphi_N}^N \circ g_{\varphi_{N-1}}^{N-1} \circ \cdots \circ g_{\varphi_1}^1$.

In forward and backward propagations through g_φ, propagations are done sequentially through the subnetworks $\{g_{\varphi_j}^j\}_{j=1}^N$. Regarding the subnetworks as subintervals of a time-dependent problem and adopting the idea of the parareal algorithm introduced in Section 2, we construct a new neural network $\bar{f}_{\bar{\theta}} \colon X \to Y$ which contains $\{g_{\varphi_j}^j\}_{j=1}^N$ as parallel subnetworks; the precise definition for parameters $\bar{\theta}$ will be given in (4).

Since the dimensions of the spaces $\{X_j\}_{j=0}^{N-1}$ are different for each j in general, we introduce preprocessing operators $C_{\delta_j}^j \colon X \to X_{j-1}$ such that $C_{\delta_1}^1 = C_\delta$ and $C_{\delta_j}^j$ for $j = 2, \ldots, N$ play similar roles to C_δ; particular examples will be given in Section 4. We write $\mathbf{x}_j \in X_{j-1}$ and $\mathbf{y}_j \in X_j$ as follows:

$$\mathbf{x}_j = C_{\delta_j}^j(\mathbf{x}) \text{ for } \mathbf{x} \in X, \quad \mathbf{y}_j = g_{\varphi_j}^j(\mathbf{x}_j). \tag{2}$$

Then, we consider neural networks $F^j_{\eta_j}\colon X_j \to X_{j+1}$ with parameters η_j for $j \geq 1$ such that it approximates $g^{j+1}_{\varphi_{j+1}}$ well while it has a cheaper computational cost than $g^{j+1}_{\varphi_{j+1}}$, i.e., $F^j_{\eta_j} \approx g^{j+1}_{\varphi_{j+1}}$ and $\dim(\eta_j) \ll \dim(\varphi_{j+1})$. Emulating the coarse grid correction of the parareal algorithm, we assemble a network called *coarse network* with building blocks $F^j_{\eta_j}$. With inputs $\mathbf{x}_{j+1}$, $\mathbf{y}_j$, and an output $\mathbf{y} \in Y$, the coarse network is described as follows:

$$\mathbf{r}_N = \mathbf{0}, \quad \mathbf{r}_j = \mathbf{y}_j - \mathbf{x}_{j+1} \quad \text{for } j = 1, \dots, N-1, \tag{3a}$$

$$\tilde{\mathbf{r}}_1 = \mathbf{r}_1, \quad \tilde{\mathbf{r}}_{j+1} = \mathbf{r}_{j+1} + F^j_{\eta_j}(\tilde{\mathbf{r}}_j) \quad \text{for } j = 1, \dots, N-1, \tag{3b}$$

$$\tilde{\mathbf{y}} = \mathbf{y}_N + \tilde{\mathbf{r}}_N. \tag{3c}$$

That is, in the coarse network, the residual $\mathbf{r}_j$ at the interface between layers $g^j_{\varphi_j}$ and $g^{j+1}_{\varphi_{j+1}}$ propagates through shallow neural networks $F^1_{\eta_1}, \dots, F^{N-1}_{\eta_{N-1}}$. Then the propagated residual is added to the output.

Finally, the parareal neural network $\bar{f}_{\bar{\theta}}$ corresponding to the original network f_θ is defined as

$$\bar{f}_{\bar{\theta}}(\mathbf{x}) = h_\varepsilon(\tilde{\mathbf{y}}), \quad \bar{\theta} = \left(\bigoplus_{j=1}^{N} (\delta_j \oplus \varphi_j) \right) \oplus \left(\bigoplus_{j=1}^{N-1} \eta_j \right) \oplus \varepsilon. \tag{4}$$

That is, $\bar{f}_{\bar{\theta}}$ is composed of the preprocessing operators $\{C^j_{\delta_j}\}$, parallel subnetworks $\{g^j_{\varphi_j}\}$, the coarse network $\{F^j_{\eta_j}\}$, and the postprocessing operator h_ε. Figure 1(b) illustrates $\bar{f}_{\bar{\theta}}$.

Since each $g^j_{\varphi_j} \circ C^j_{\delta_j}$ lies in parallel, all computations related to $g^j_{\varphi_j} \circ C^j_{\delta_j}$ can be done independently. Therefore, multiple GPUs can be utilized to process $\{g^j_{\varphi_j} \circ C^j_{\delta_j}\}$ simultaneously for each j. In this case, one may expect significant decrease of the elapsed time for training $\bar{f}_{\bar{\theta}}$ compared to the original network f_θ. On the other hand, the coarse network cannot be parallelized since $\{F^j_{\eta_j}\}$ is computed in the sequential manner. One should choose $F^j_{\eta_j}$ whose computational cost is as cheap as possible in order to reduce the bottleneck effect of the coarse network.

Now, we want show that the proposed parareal neural network $\bar{f}_{\bar{\theta}}$ is consistently constructed in the sense that it recovers the original neural network f_θ in the setting where nonlinearity is removed. By collecting all the residuals in each interface and dealing with them sequentially, the following proposition is obtained.

Proposition 1 (Consistency)

Assume that the original network f_θ is linear and $F^j_{\eta_j} = g^{j+1}_{\varphi_{j+1}}$ for $j = 1, \dots, N-1$. Then we have $\bar{f}_{\bar{\theta}}(\mathbf{x}) = f_\theta(\mathbf{x})$ for all $\mathbf{x} \in X$.

Proposition 1 presents a guideline on how to design the coarse network of $\bar{f}_{\bar{\theta}}$. Under the assumption that f_θ is linear, a sufficient condition to ensure that $\bar{f}_{\bar{\theta}} = f_\theta$

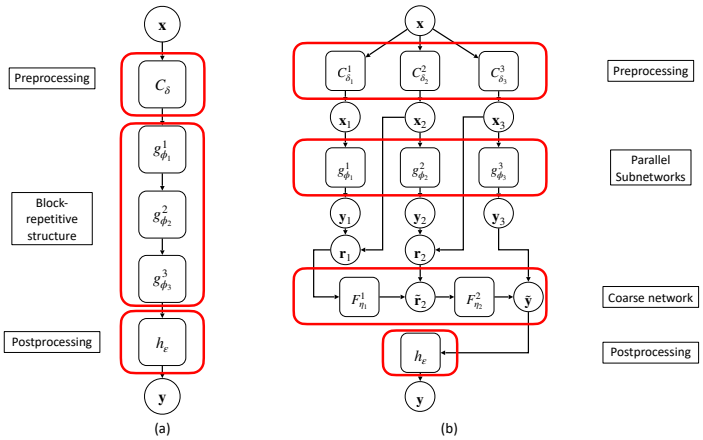

Fig. 1: A feed-forward neural network and its corresponding parareal neural network: **(a)** Feed-forward neural network f_θ, **(b)** Parareal neural network $\bar{f}_{\bar{\theta}}$ with N parallel subnetworks ($N = 3$).

is $F^j_{\eta_j} = g^{j+1}_{\varphi_{j+1}}$ for all j. Therefore, we can say that it is essential to design the coarse network with $F^j_{\eta_j} \approx g^{j+1}_{\varphi_{j+1}}$ to ensure that the performance of $\bar{f}_{\bar{\theta}}$ is as good as that of f_θ. Detailed examples will be given in Section 4.

4 Application to ResNet-1001

The proposed parareal neural network can be applied to a general feed-forward neural network. However, since most of the current very deep neural networks have residual structures, we applied it to ResNet-1001 [9], which is one of the typical very deep convolutional neural network for classification problems. First, we describe the structure of ResNet-1001 with the terminology introduced in Section 3. Inputs for ResNet-1001 are 3-channel images with 32×32 pixels, i.e., $X = \mathbb{R}^{3\times32\times32}$. The output space Y is given by $Y = \mathbb{R}^m$, where m is the number of classes of images. ResNet-1001 has a block-repetitive substructure consisting of 333 residual units (RUs), so that we may set $g_\varphi \colon W_0 \to W_1$ as the composition of those RUs with $W_0 = \mathbb{R}^{16\times32\times32}$ and $W_1 = \mathbb{R}^{256\times8\times8}$. Then the preprocessing operator $C_\delta \colon X \to W_0$ is a single 3×3 convolution layer and the postprocessing operator $h_\varepsilon \colon W_1 \to Y$ consists of global average pooling and fully connected layers.

The design of a parareal neural network with N parallel subnetworks for ResNet-1001, denoted as *Parareal ResNet-N*, can be completed by specifying the structures $g^j_{\varphi_j}$, $C^j_{\delta_j}$, and $F^j_{\eta_j}$. For convenience, the original neural network ResNet-1001 is called Parareal ResNet-1. We assume that $N = 3N_0$ for some positive integer N_0. We note that g_φ can be decomposed as

$$g_\varphi = g^N_{\varphi_N} \circ \cdots \circ g^{2N_0+1}_{\varphi_{2N_0+1}} \circ g^{2N_0}_{\varphi_{2N_0}} \circ \cdots \circ g^{N_0+1}_{\varphi_{N_0+1}} \circ g^{N_0}_{\varphi_{N_0}} \circ \cdots \circ g^1_{\varphi_1},$$

where each of $g^j_{\varphi_j} \colon X_{j-1} \to X_j$ consists of $\lceil 333/N \rceil$ RUs with

$$X_j = \begin{cases} \mathbb{R}^{64\times 32\times 32} & \text{for } j = 1, \ldots, N_0, \\ \mathbb{R}^{128\times 16\times 16} & \text{for } j = N_0+1, \ldots, 2N_0, \\ \mathbb{R}^{256\times 8\times 8} & \text{for } j = 2N_0+1, \ldots, N, \end{cases} \qquad \varphi = \bigoplus_{j=1}^{N} \varphi_j.$$

The main role of the preprocessing operator $C^j_{\delta_j} \colon X \to X_{j-1}$ is to transform an input $\mathbf{x} \in X$ to fit in the space X_{j-1}. In this perspective, we simply set $C^1_{\delta_1} = C_\delta$ and $C^j_{\delta_j}$ for $j > 1$ consists of a 1×1 convolution to match the number of channels after appropriate number of 3×3 max pooling layers with stride 2 to match the image size. For the coarse network, we first define a coarse RU consisting of two 3×3 convolutions and skip-connection. If the downsampling is needed, then the stride of first convolution in coarse RU is set to 2. We want to define $F^j_{\eta_j} \colon X_j \to X_{j+1}$ having smaller number of (coarse) RUs than $g^{j+1}_{\varphi_{j+1}}$ but a similar coverage to $g^{j+1}_{\varphi_{j+1}}$. Let N_c be the number of coarse RUs in $F^j_{\eta_j}$ of the coarse network. Note that the receptive field of $g^j_{\varphi_j}$ covers the input size 32×32. In the case of $N = 3$, even if we construct $F^j_{\eta_j}$ with $N_c = 4$ coarse RUs, it can cover 31×31 pixels which are similar coverage to the parallel subnetwork $g^j_{\varphi_j}$. Generally, if we use N parallel subnetworks ($N \geq 3$), each $333/N$ RUs in $g^j_{\varphi_j}$ can be approximated by the N_c RUs in $F^j_{\eta_j}$ whenever we select $N_c = \lceil 12/N \rceil$.

5 Numerical results

In this section, we present numerical results of the Parareal ResNet-N with various N. First, we present details on the datasets we used. The CIFAR-m ($m = 10, 100$) dataset consists of 32×32 colored natural images and includes 50,000 training and 10,000 test samples with m classes. The SVHN dataset is composed of 32×32 colored digit images; there are 73,257 and 26,032 samples for training and test, respectively, with additional 531,131 training samples. However, we did not use the additional ones for training. MNIST is a classic dataset which contains handwritten digits encoded in 28×28 grayscale images. It includes 55,000 training, 5,000 validation, and 10,000 test samples. In our experiments, the training and validation samples are used as training data and the test samples as test data. We adopted a data augmentation technique in [13] for CIFAR datasets; four pixels are padded on each side of images, and 32×32 crops are randomly sampled from the padded images and their horizontal flips.

All neural networks in this section were trained using the stochastic gradient descent with the batch size 128, weight decay 0.0005, momentum 0.9, and weights initialized as in [7]. The initial learning rate is set to 0.1, and is reduced by a factor

Table 1: Error rates (%) on the CIFAR-10, CIFAR-100, MNIST, and SVHN datasets of Parareal ResNet-N ($N = 1, 3, 6, 12, 18$) with $N_c = \lceil 12/N \rceil$.

N	Parameters per subnetwork	Parameters of coarse network	Total Parameters	CIFAR-10	CIFAR-100	MNIST	SVHN
1	-	-	10.3M	4.96	21.13	0.34	3.17
3	3.4M	5.6M	15.9M	4.61	21.14	0.31	3.11
6	1.7M	5.7M	16.1M	4.20	20.87	0.31	3.21
12	0.9M	5.8M	16.2M	4.37	20.42	0.28	3.25
18	0.6M	8.9M	19.4M	4.02	20.40	0.33	3.29

Table 2: Forward/backward computation time for Parareal ResNet-N ($N = 1, 3, 6, 12, 18$). The time is measured in one iteration for CIFAR-100 dataset input $\mathbf{x} \in \mathbb{R}^{3\times32\times32}$ with batch size 128.

Virtual wall-clock time (ms)					
N	Preprocessing	Parallel subnetworks	Coarse network	Postprocessing	Total
1	0.25/6.46	443.81/1387.62	-	0.06/3.18	444.12/1397.26
3	0.25/6.45	131.92/458.87	10.01/97.60	0.06/3.71	142.24/566.63
6	0.27/6.42	67.59/219.72	14.68/137.08	0.06/3.33	82.60/366.55
12	0.28/6.59	48.47/113.33	17.97/149.52	0.06/3.63	66.78/273.07
18	0.29/6.17	30.40/77.84	27.87/163.25	0.06/3.64	58.62/250.90
24	0.29/6.58	22.71/58.04	41.03/242.87	0.06/3.54	64.09/311.03

of 10 in the 80th and 120th epochs. All networks were implemented in Python with PyTorch and all computations were performed on a cluster equipped with Intel Xeon Gold 5515 (2.4GHz, 20C), NVIDIA Titan RTX and the operating system Ubuntu 18.04 64bit.

With fixed $N_c = \lceil 12/N \rceil$, we report the classification results of Parareal ResNet with respect to various N on datasets CIFAR-10, CIFAR-100, SVHN, and MNIST. Table 1 shows that the error rates of Parareal ResNet-N are usually smaller than ResNet-1001.

Next, we investigate the elapsed time for forward and backward propagations of parareal neural networks. Table 2 shows the virtual wall-clock time for forward and backward computation of Parareal ResNet-N with various N for the input $\mathbf{x} \in \mathbb{R}^{3\times32\times32}$. As shown in Table 2, the larger N, the shorter the computing time of the parallel subnetworks $g^j_{\varphi_j}$, while the longer the computing time of the coarse network. This is because as N increases, the depth of each parallel subnetwork $g^j_{\varphi_j}$ becomes shallower while the number of $F^j_{\eta_j}$ in the coarse network increases. On the other hand, each preprocessing operator $C^j_{\delta_j}$ is designed to be the same as or similar to the preprocessing operator C_δ of the original neural network and the postprocessing operator h_ε is the same as the original one. Therefore, the computation time for the pre- and postprocessing operators does not increase even as N increases.

Table 3: The wall-clock time and relative speedup on the CIFAR-100 dataset. The wall-clock time is the total time taken to train a given network by 200 epochs.

Network	Parameters	Wall-clock time (h:m:s)	Relative speedup (%)
ResNet-1001	10.3M	22:44:53	0.0
Parareal ResNet-3	15.9M	16:28:38	27.6
Parareal ResNet-6	16.1M	11:48:13	48.1

Finally, we meausure the wall-clock time of the Parareal ResNet with the CIFAR-100 dataset. Table 3 shows that Parareal ResNet's wall clock time is reduced by about half as N increases to 6.

In conclusion, despite the large number of layers, the parareal neural network can accelerate the training of the very deep CNN using multiple-GPU. To the best of our knowledge, the proposed methodology is a new kind of multi-GPU parallelism in the field of deep learning.

References

1. Bal, G.: On the convergence and the stability of the parareal algorithm to solve partial differential equations. In: Domain Decomposition Methods in Science and Engineering, pp. 425–432. Springer, Berlin (2005)
2. Chen, C.C., Yang, C.L., Cheng, H.Y.: Efficient and robust parallel DNN training through model parallelism on multi-GPU platform (2018). ArXiv:1809.02839
3. Falgout, R.D., Friedhoff, S., Kolev, T.V., MacLachlan, S.P., Schroder, J.B.: Parallel time integration with multigrid. SIAM Journal on Scientific Computing **36**(6), C635–C661 (2014)
4. Gander, M.J., Hairer, E.: Nonlinear convergence analysis for the parareal algorithm. In: Domain Decomposition Methods in Science and Engineering XVII, pp. 45–56. Springer, Berlin (2008)
5. Gander, M.J., Vandewalle, S.: Analysis of the parareal time-parallel time-integration method. SIAM Journal on Scientific Computing **29**(2), 556–578 (2007)
6. Günther, S., Ruthotto, L., Schroder, J.B., Cyr, E.C., Gauger, N.R.: Layer-parallel training of deep residual neural networks. SIAM Journal on Mathematics of Data Science **2**(1), 1–23 (2020)
7. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
9. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision, pp. 630–645. Springer, Cham (2016)
10. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)

12. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989)
13. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Artificial Intelligence and Statistics, pp. 562–570 (2015)
14. Lions, J.L., Maday, Y., Turinici, G.: Résolution d'EDP par un schéma en temps «pararéel». Comptes Rendus de l'Académie des Sciences-Series I-Mathematics **332**(7), 661–668 (2001)
15. Maday, Y., Turinici, G.: A parareal in time procedure for the control of partial differential equations. Comptes Rendus Mathematique **335**(4), 387–392 (2002)
16. Minion, M.: A hybrid parareal spectral deferred corrections method. Communications in Applied Mathematics and Computational Science **5**(2), 265–301 (2011)
17. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1492–1500 (2017)