Parareal neural networks emulating a parallel-in-time algorithm

Youngkyu Lee, Jongho Park, Chang-Ock Lee
Department of Mathematical Sciences, KAIST, Daejeon, Korea
lyk92@kaist.ac.kr, jongho.park@kaist.ac.kr, colee@kaist.edu

Korea Advanced Institute of Science and Technology
Department of Mathematical Sciences

Abstract

As deep neural networks (DNNs) become deeper, the training time increases. In this perspective, multi-GPU parallel computing has become a key tool in accelerating the training of DNNs. In this presentation, we introduce a novel methodology to construct a parallel neural network that can utilize multiple GPUs simultaneously from a given DNN. We observe that layers of the DNN can be interpreted as time steps of a time-dependent problem and can be parallelized by emulating a parallel-in-time algorithm called parareal. The parareal algorithm consists of fine structures which can be implemented in parallel and a coarse structure which gives suitable approximations to the fine structures. By emulating it, the layers of the DNN are torn to form a parallel structure, which is connected using a suitable coarse network. We report accelerated and accuracy-preserved results of the proposed methodology applied to ResNet-1001 on the datasets CIFAR-10 and CIFAR-100.

1 Introduction

- DNN training is time-consuming and there are demands to reduce training time these days.
- DNN has a feed-forward architecture whose sequential computations can be interpreted as time steps of a time-dependent problem.
- 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 proposed methodology is easy to program since there is no data structures lying on the subdomain interfaces.

2 The parareal algorithm

The parareal algorithm, proposed by Lions et al. [1], 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:

$$\dot{\mathbf{u}}(t) = A\mathbf{u}(t) \text{ in } [0,T], \quad \mathbf{u}(0) = \mathbf{u}_0, \qquad (2.1)$$

where A: $\mathbb{R}^m \to \mathbb{R}^m$ is an operator, $T > 0$, and $\mathbf{u}_0 \in \mathbb{R}^m$.

Algorithm 1: The parareal algorithm

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 $\dfrac{\mathbf{U}^1_{j+1} - \mathbf{U}^1_j}{\Delta T_j} = A\mathbf{U}^1_{j+1}, \ \mathbf{U}^1_0 = \mathbf{u}_0$
end
for $k \leftarrow 1, 2, \ldots$ **do**
  for $j \leftarrow 0$ **to** N ***in parallel*** **do**
    Solve $\dot{\mathbf{u}}^k_j(t) = A\mathbf{u}^k_j(t)$ in $[T_j, T_{j+1}]$, $\mathbf{u}^k_j(T_j) = \mathbf{U}^k_j$.
  end
  for $j \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{S}^k_{j+1} = \mathbf{u}^k_j(T_{j+1}) - \mathbf{U}^k_{j+1}$.
    Solve $\dfrac{\boldsymbol{\delta}^k_{j+1} - \boldsymbol{\delta}^k_j}{\Delta T_j} = A\boldsymbol{\delta}^k_{j+1} + \mathbf{S}^k_j, \ \boldsymbol{\delta}^k_0 = 0$.
    $\mathbf{U}^{k+1}_{j+1} = \mathbf{U}^k_{j+1} + \boldsymbol{\delta}^k_{j+1}$.
  end
end

3 Parareal neural networks

Neural network setting

- f_θ: $X \to Y$ is a neural network.
- $f_\theta = h_\varepsilon \circ g_\phi \circ C_\delta, \quad \theta = \delta \oplus \phi \oplus \varepsilon$,
- C_δ is a preprocessing operator and h_ε is a postprocessing operator.
- g_ϕ is a block repetitive sturcture.

Parareal setting

- $\{C^j_{\delta_j}\}$ is defined to satisfy $C^1_{\delta_1} = C_\delta$ and $C^j_{\delta_j}$ for $j = 2, \ldots, N$ play similar roles to C_δ.
- $\{g^i_{\phi_i}\}$ is a collection of parallel subnetworks such that $g_\phi = g^N_{\phi_N} \circ g^{N-1}_{\phi_{N-1}} \circ \cdots \circ g^1_{\phi_1}$.
- $\{F^j_{\eta_j}\}$ is a coarse network satisfying that $F^j_{\eta_j} \approx g^{j+1}_{\phi_{j+1}}$ and $\dim(\eta_j) \ll \dim(\phi_{j+1})$.

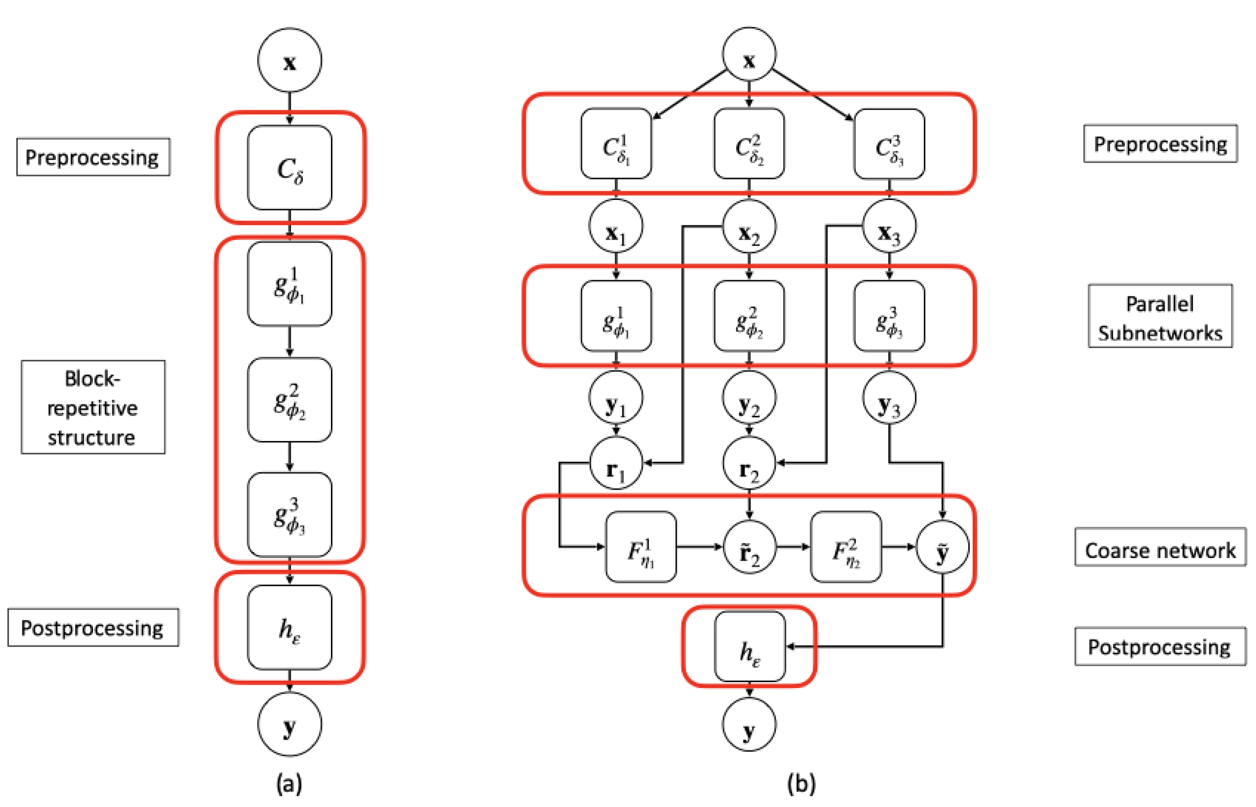

Figure 1: **(a)** Feed-forward neural network f_θ, **(b)** Parareal neural network $\bar{f}_{\bar{\theta}}$ with N parallel subnetworks ($N = 3$).

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

4 Application to ResNet-1001

We present an application of the proposed methodology to ResNet-1001, which is a typical convolutional neural network for classification problems.

ResNet-1001 description

- C_δ is a single 3×3 convolution layer.
- $C^1_{\delta_1} = C_\delta$ and $C^j_{\delta_j}$ is a 1×1 convolution with stride for $j > 1$.
- g_ϕ consists of 333 residual units(RUs) which are decomposed into $\{g^j_{\phi_j}\}$ having $\lceil 333/N \rceil$ RUs.
- $\{F^j_{\eta_j}\}$ is constructed using $N_c := \lceil \frac{12}{N} \rceil$ RUs, which can have similar coverage to the parallel subnetwork $g^j_{\phi_j}$.
- h_ε consists of global average pooling and fully connected layer.

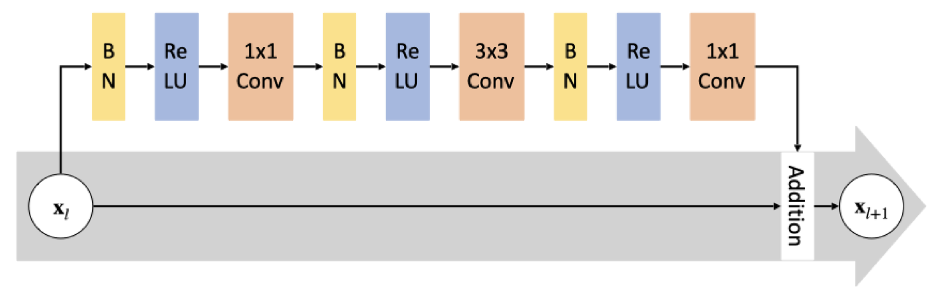

Figure 2: Bottleneck structure of RU used in ResNet-1001.

5 Numerical results

Environment: Python, PyTorch
Machine: Intel Xeon Gold 5515, NVIDIA Titan RTX
Dataset: CIFAR-10, CIFAR-100
Training Strategy: Batchsize 128, Epoch 200, SGD with weight decay 0.0005, momentum 0.9, learning rate 0.1 which is reduced by a factor of 10 in the 80th and 120th epochs.

5.1 Performance comparison

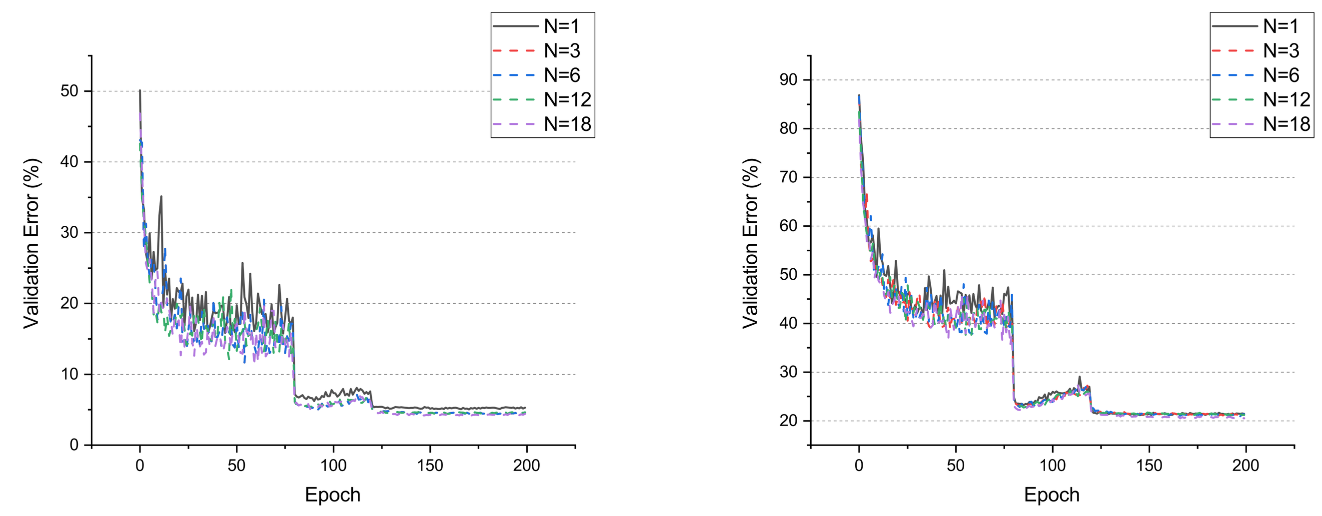

Figure 3: Comparison of the validation error rates(%) for ResNet-1001 and its parareal neural networks on CIFAR datasets: **[Left]** CIFAR-10 and **[Right]** CIFAR-100 results.

Network	CIFAR-10	CIFAR-100
ResNet-1001	5.03	21.13
Parareal ResNet-3	4.51	21.02
Parareal ResNet-6	4.35	20.94
Parareal ResNet-12	4.37	20.77
Parareal ResNet-18	4.11	20.46

Table 1: Error rates(%) on CIFAR datasets. Parareal ResNet-N denotes the parareal neural network using N subnetworks with $N_c = \lceil \frac{12}{N} \rceil$.

5.2 Gradient calculating time

	Virtual wall-clock time				
N	Preprocessing	Parallel subnetworks	Coarse network	Postprocessing	Total
1	0.0010	1.4588	-	0.0009	1.4607
3	0.0012	0.5966	0.0543	0.0008	0.6529
6	0.0019	0.3062	0.0638	0.0007	0.3726
12	0.0023	0.1583	0.0797	0.0009	0.2412
18	0.0025	0.1067	0.1224	0.0010	0.2326
24	0.0020	0.0797	0.1701	0.0008	0.2526

Table 2: Gradient calculating time for ResNet-1001 ($N = 1$) and Parareal ResNet with ($N = 3, 6, 12, 18, 24$). It is a measure of the time taken in one iteration for CIFAR dataset input $\mathbf{x} \in \mathbb{R}^{3\times 32\times 32}$ with batch size 128.

6 Conclusion

- We proposed a novel methodology to construct a parallel neural network called the parareal neural network which is suitable for parallel computation using multiple GPUs from a given feed-forward neural network.
- The coarse network that corrects differences at the interfaces among subnetworks was introduced, and it was proven both theoretically and numerically that the performance of the resulting parareal network agrees with the original network.
- 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] J.-L. LIONS, Y. MADAY, and G. TURINICI: *Résolution d'EDP par un schéma en temps «pararéel»*, Comptes Rendus de l'Académie des Sciences-Series I-Mathematics, 332, pp.661–668 (2001)