Cooperative Scheduling Schemes for Explainable DNN Acceleration in Satellite Image Analysis and Retraining

Woo-Joong Kim, Member, IEEE and Chan-Hyun Youn, Senior Member, IEEE

Abstract—The deep learning-based satellite image analysis and retraining systems are getting emerging technologies to enhance the capability of the sophisticated analysis of terrestrial objects. In principle, to apply the explainable DNN model for the process of satellite image analysis and retraining, we consider a new acceleration scheduling mechanism. Especially, the conventional DNN acceleration schemes cause serious performance degradation due to computational complexity and costs in satellite image analysis and retraining. In this article, to overcome the performance degradation, we propose cooperative scheduling schemes for explainable DNN acceleration in analysis and retraining process. For the purpose of it, we define the latency and energy cost modeling to derive the optimized processing time and cost required for explainable DNN acceleration. Especially, we show a minimum processing cost considered in the proposed scheduling via layer-level management of the explainable DNN on FPGA-GPU acceleration system. In addition, we evaluate the performance using an adaptive unlabeled data selection scheme with confidence threshold and a semi-supervised learning driven data parallelism scheme in accelerating retraining process. The experimental results demonstrate that the proposed schemes reduce the energy cost of the conventional DNN acceleration systems by up to about 40% while guaranteeing the latency constraints.

Index Terms—Cooperative satellite image analysis and retraining, DNN acceleration, distributed deep learning

1 INTRODUCTION

For automating reliable remote sensing and improving the analysis speed of human supervisors, it is necessary to design a satellite image analysis and retraining system based on explainable DNN. The explainable DNN generates a description of its prediction, and the human supervisors return feedbacks, such as corrections or new label annotations, for retraining [1], [2]. However, the retraining system still has several bottlenecks.

First, explainable DNN, which achieves high accuracy for reliable satellite image analysis, requires high computational complexity. In general, higher inference accuracy can be achieved with a deeper and wider network containing a greater number of network layers and channels [3]. These features significantly increase the computing complexity and memory access complexity that sophisticated hardware accelerators are required to address. Furthermore, DNN tasks computationally have a high workload with massive input data (e.g., large high-definition images, etc.)

Second, the labeling task by supervisors is cost-expensive and slow. Especially, the labeling speed is relatively too slow compared to the input data generation and explainable DNN based analysis speed. Since the image data is generated in real-time and delivered to human supervisors, the partial data is discarded without labeling. It causes overfitting on explainable DNN retraining due to scarcity of human annotations [2].

For these reasons, the analysis and model retraining process suffers from drastically long processing time and slow convergence speed [2]. To solve these bottlenecks, the conventional DNN acceleration systems attempt to schedule the DNN process for acceleration, using a large-scale accelerator cluster. However, their scheduling schemes have several problems to successfully implement DNN acceleration in the analysis and model retraining process.

Heterogeneity of Accelerator Environment. Heterogeneous accelerators, described in Table 1, should be considered for accelerating DNN processing tasks. Both GPUs and FPGAs have been deployed and utilized in datacenter infrastructure at a reasonable scale [8], [9], [10], to process a given DNN workload quickly and energy-efficiently. Current generation DNNs depend heavily on dense floating-point matrix multiplication, which is well mapped to GPUs [4]. For this reason, GPUs are widely used for DNN acceleration [6]. Meanwhile, several recent studies have attempted to configure the HPC environment with FPGA to reduce the energy cost of a large-scale DNN process by using its high processing efficiency per energy [5]. Unlike other processors that operate with a combination of predefined sets of operations, the FPGA can specify functionality at the gate level. Depending on the design method, FPGA can implement compressed neural
networks with weight quantization to accelerate certain operations.

Computational Complexity in Explainable DNN. Fig. 1 shows a typical scheduling with explainable DNN in satellite image analysis and retraining. The explainable DNN delivers object detection or classification results and their visual explanation to the human supervisor. And then, the human supervisor returns feedbacks for retraining [2]. Since labeling satellite images is time-consuming and its cost is expensive, many previous works adopt Active Learning (AL) to optimally select the data samples to be labeled from an unlabeled data pool to achieve the highest accuracy within a fixed labeling budget [17]. In general, explainable DNN is a large-scale DNN for high reliability. Thus, network layers of an explainable DNN have various computing complexity and memory demands. Explainable DNN model is composed of several components such as convolutional layer(CL), fully connected layer(FC) and region proposal network(RPN) [21]. These components have different processing time and energy cost performance depending on what type of accelerator is allocated.

In this paper, to overcome these problems, we design new explainable DNN acceleration scheduling schemes. We propose the cooperative scheduling schemes utilizing the layer-level management of the explainable DNN in image analysis as well as the confidence level criteria and data parallelism in retraining process. Our work has following contributions.

- First, we define the latency and energy cost modeling to derive the optimized processing time and cost required for explainable DNN acceleration in cooperative satellite image analysis and retraining. Especially, we propose a cooperative scheduling scheme via layer-level management of explainable DNN on FPGA-GPU to accelerate analysis process and achieve a minimum processing cost.
- In addition, we propose a confidence threshold based adaptive unlabeled data selection scheme and a semi-supervised learning driven data parallelism scheme for accelerating retraining process.
- Last, we evaluate the proposed system using a large-scale aerial image dataset for object detection or classification, such as DOTA and AID. The experimental results demonstrate that the proposed schemes effectively reduce the retraining cost compared to the conventional DNN acceleration systems, while guaranteeing the latency constraints.

## 2 A MODEL DESCRIPTION ON COOPERATIVE SCHEDULING FOR EXPLAINABLE DNN ACCELERATION IN SATELLITE IMAGE ANALYSIS AND RETRAINING

In this section, we present some limitations in applying the conventional DNN acceleration schemes to the explainable DNN acceleration in satellite image analysis and retraining. Especially, to overcome these limitations, we discuss new explainable DNN acceleration scheduling schemes.

### 2.1 Design of Cooperative Scheduling for Explainable DNN Acceleration

In satellite image analysis and retraining, the workload of human supervisors is still a big bottleneck (i.e., high labeling cost and slow labeling speed). To address this issue, we adopt AL on unlabeled data and semi-supervised learning-based retraining. Most unlabeled samples are typically ignored in AL [23]. AL selects only a few of the most informative samples (e.g., samples with very low predictive reliability) for labeling.

### TABLE 1 Specification of Heterogeneous GPU Devices (NVIDIA Products)[6], [7]

| Spec       | GTX 1060 | GTX 1080 | Quadro M2000 |
|------------|----------|----------|--------------|
| FLOPS      | 4375 GFLOPS | 8873 GFLOPS | 1812 GFLOPS  |
| Memory     | 6 GB     | 8 GB     | 4 GB         |
| Bandwidth  | 192.2 GB/sec | 320.3 GB/sec | 105.8 GB/sec |
| Power      | 120 W    | 180 W    | 75 W         |

Fig. 1. Satellite image analysis and retraining with explainable DNN.
at each training stage. It is difficult to fine-tune the DNN with these few samples of information to obtain appropriate functional representations. In semi-supervised learning, unlabeled data (often much cheaper to obtain) is also used to train DNNs [24]. Unlabeled data can be used to train some distributions, which is helpful to create more sophisticated and effective regularization. To overcome the bottleneck of human supervisor’s workload, we apply AL on unlabeled data and semi-supervised learning, which considers labeled and unlabeled data together, to the cooperative scheduling in satellite image analysis and retraining. Based on them, we organize data selection scheme on unlabeled data and semi-supervised learning based retraining scheduling scheme.

2.2 Limitation of Conventional DNN Acceleration Schemes for Explainable DNN Acceleration

Several systems are proposed to achieve optimal performance and cost-effective scheduling for DNN acceleration in an HPC environment. To perform task scheduling, execution, and visualization, the systems set up a highly-tuned computing pool by distributed resources with a common interface to an auto-run environment that can typically be applied to various types of DNN processes. Their goals are to coordinate DNN tasks on a distributed set of resources while minimizing energy costs and ensuring processing time constraints. S3DNN [12] simultaneously optimizes two conflicting objectives: new supervised streaming and scheduling frameworks, real-time accuracy and throughput that optimize the execution of DNN workloads on GPU in a real-time multitasking environment. Fang, Zhou, et al. [13] proposes QoS-aware effective heuristic scheduling of heterogeneous GPU clusters for DNN inference.

Especially, Nexus [28] is a GPU cluster engine to achieve high DNN inference throughput with latency constraints [28], as shown in Fig. 2. It balances DNN workloads and maximizes GPU utilization by performing DNN model placement, profiles-based bin packing, and scheduling batching aware DNN inference execution. It use a linear model to fit the batched latency. Its batched throughput model, derived from its linear latency model, shows about 8% performance degradation with GPU (RTX 2080). In addition, different types of GPUs have different batched latency and throughput performance even with the same model (Faster RCNN).

However, Nexus [28] only considers the GPU as an accelerator. It assumes the type of accelerators in the HPC environment is homogeneous and has identical performance characteristics in terms of throughput or latency and energy consumption. Also, it processes a DNN model as a unit of a task without considering the variety of computational complexity within a DNN.

When FPGA and GPU process Resnet [25], which consists of CL and FC, respectively, their throughput/watt performances vary on each layer. FPGA offers better throughput/watt over GPU for processing CL, since the power of GPU is usually higher than those of FPGA and the throughput of FPGA is comparable for those of GPU [5]. However, in the case of FC, GPU offers better throughput/watt over FPGA, because the memory of FPGA is insufficient to process FC. It causes a bottleneck and results in a sharp drop in computing performance.

In heterogeneous accelerator environment, where FPGA and GPU exists, the cooperative scheduling on FPGA-GPU with layer-level management utilizing the variety of complexity in explainable DNN inference, can reduce not only the energy consumption, but also the processing time. The reason is that this scheduling can maximize the energy efficiency of FPGA by allocating more CL tasks to FPGA and avoid the inefficiency by allocating less FC tasks to FPGA. The conventional DNN acceleration makes an inefficient decision to allocate more FC tasks to FPGA because the DNN tasks are assigned on accelerators in units of one DNN model.

By using or expanding the functions of Nexus, this optimal performance cannot be achieved in the target heterogeneous...
HPC environment. It is necessary to newly design a DNN acceleration system with cooperative scheduling schemes based on the layer-level management on FPGA-GPU. Besides, Nexus only focus on DNN inference, not DNN training required after completing DNN inference in the retraining system.

3 A PROPOSED COOPERATIVE SCHEDULING FOR EXPLAINABLE DNN ACCELERATION IN SATELLITE IMAGE ANALYSIS AND RETRAINING SYSTEM

In this section, we describe a cooperative scheduling for explainable DNN acceleration in satellite image analysis and retraining. Then, we define the problems for the analysis and retraining process and resolve them by cooperative scheduling schemes.

3.1 System Model and Key Features

Fig. 3 shows our target HPC environment and explainable DNN based retraining framework in satellite image acquisition scenario. A high-resolution remote sensing image (e.g., 30k x 30k) is generated and cropped to image patches in the satellite on-board system. In order to analyze the image patches, the satellite on-board system periodically transmits the image patches to the HPC ground station. We assume that the image patches, of which total size is $D$ (Bytes), are transmitted and already queued in the beginning of each period. This framework processes the given image patches within a period in order to keep the queue stable.

Our cooperative scheduling for explainable DNN acceleration performs the following three steps.

Step 1) Explainable DNN Based Analysis Process. It aims to allocate the analysis inference tasks to the heterogeneous accelerator (FPGA/GPU) with respect to energy cost minimization. The images are applied as input to the target explainable DNN model, which takes object recognition and has an explainable functionality such as Grad-CAM [11]. In order to assist human supervisors in satellite image analysis, the HPC ground station system performs the target model inference (i.e., DNN Model + Grad-CAM) that acts as an AI supervisor and delivers classification results and visual explanations for satellite images to human supervisors. It assists human supervisor’s detection ability and improves their reading speed. Faster processing on the target model

1. The length of a period can be a latency constraints.
inference can speed up the reading of human supervisors with more visual explanations.

Step 2) Confidence Threshold Based Adaptive Data Selection. It aims to select the training samples (including unlabeled samples) satisfying the thresholded confidence, which sends to the human supervisor. Training samples selected by data selection process are labeled by human supervisors.

From the total input data pool, it aims to automatically and progressively select the most informative data that human supervisors need to label. It organizes the training data $D_{tr}$ including labeled images $D_{tr}^{L}$ as well as unlabeled images $D_{tr}^{U}$. Using some of the initial labeled data, the model begins incremental learning with the data which has not yet been labeled and is informative to learn. The model will gradually be upgraded through a cycle of learning progress.

Step 3) Semi-Supervised Learning Based Retraining Process. It aims to schedule the available accelerators to achieve service deadline for retraining with minimal energy cost. With the training data $D_{tr}$ composed of the newly labeled samples $D_{tr}^{L}$ and unlabeled training samples $D_{tr}^{U}$, semi-supervised learning is performed.

Heterogeneous HPC Environment With FPGA-GPU. The HPC environment is a cluster composed of $M$ accelerator nodes, and each accelerator node is composed of a CPU and a plurality of heterogeneous accelerators such as GPU and FPGA. The data is exchanged between the host main memory and the accelerator global memory over a PCIe link. Its bandwidth is denoted by $BW_{pcie}$ (Bytes/sec). We assume that the PCIe bandwidths are same in all nodes, since the PCI express bandwidth is considerably high compared to the throughput of each accelerator. The $i$th accelerator node is denoted by $s_i$, and consists of $N_i$ heterogeneous accelerators.

3.2 Acceleration Scheduling With Layer-Level Management of Explainable DNN (ASLM) on FPGA-GPU

In step 1 of the HPC ground station system, the ASLM scheme schedules the total input images $D$ for invoking the target explainable DNN model inference tasks efficiently. Its objective is to achieve the minimum energy cost while satisfying latency constraints, denoted as $L_{inf}$. To do this, it determines how to allocate the accelerator nodes in the cluster and distribute the input images $D$ to them for invoking and processing the target model inference tasks.

Structure of Explainable DNN. A target explainable DNN model inference task consists of one pre-processing component and $K$ model components, to be processed sequentially. The pre-processing component, denoted as $DL_0$, represents the pre-processing tasks, such as image I/O, decoding, and batching, to be performed in the CPU before executing the explainable DNN model in the accelerators [14]. The model components, denoted as $\{DL_1, DL_2, \ldots, DL_K\}$, represent the individual layers of the target model to be performed in the accelerators such as FPGA or GPU. Due to the nature of the explainable DNN model, there are dependencies between the model components, so the next component proceeds after the current component is processed. For example, Resnet [25] is composed of the pre-processing component, $DL_0$, the model component $DL_1$, CL for feature extraction, and the model component $DL_2$, FC for classification.

Acceleration Node Allocation. The ASLM scheme determines the acceleration node allocation strategy, denoted as $X = \{x_1, \ldots, x_M\}$, to process the input data $D$. $x_i$ indicates whether to use $i$th accelerator node or not $(x_i \in \{0, 1\})$. Then, it determines the data assignment to the nodes $D = \{D^1, \ldots, D^M\}$. The data assignment to each accelerator node will be the maximum amount of data that the accelerator node can process within a given latency service-level objective $L_{inf}$.

For the data assignment $D_i$, the accelerator node $s_i$ processes each component, $\{DL_0, DL_1, \ldots, DL_K\}$, sequentially. In this process on $D^i$, the input and output data of the component $DL_k$ are denoted as $D^i_k$ and $D^i_{tr}$, respectively. $D^i$ is input to the pre-processing component $DL_0$. That is, $D^i = D^i_0$. Since the target model structure is fixed, $D^i_k/D^i_{tr} = \alpha_s$ is fixedly determined. $\alpha_s$ is output data unit per input data in the component $DL_k$. Since $D^i_{k+1}$ is dependent to $D^i_k$, note that $D^i_k = D^i_{k+1}$.

For the input data $D^i_{tr}$, the pre-processing component $DL_0$ is processed in the CPU. In the accelerator node $s_i$, the CPU throughput for the component $DL_0$ is denoted by $Th_{0}$, the idle and active power of CPU for the component $DL_0$ in node $s_i$ are denoted by $P_{idle}^0$ and $P_{act}^0$, respectively.

For the model component $DL_k$, the input data $D^i_k$ is distributed to the $N_i$ accelerators via PCIe and is processed in them. The data assignment to the $N_i$ accelerators is denoted as $\{D^i_{k,1}, D^i_{k,2}, \ldots, D^i_{k,N_i}\}$. This satisfies $D^i_k = \sum_{j \in [N_i]} D^i_{k,j}$. The accelerator throughput for the components $DL_k$ of $N_i$ accelerators in the node $s_i$ is denoted by $[Th_{k,1}^i, Th_{k,2}^i, \ldots, Th_{k,N_i}^i]$. Since we only consider two types of accelerator, GPU and FPGA, each accelerator is one of both. The throughput of each accelerator on $DL_k$ is defined as the maximum throughput with the optimal batch size, referring to [28]. Therefore, the throughput of a certain FPGA is $Th_{k,FPGA}^i = \frac{BW_{pcie}}{Th_{k,m}^i}$, where $b^*$ is the optimal size and $\hat{b}_k^i$ is the latency of the GPU on $DL_k$ with $b^*$ and $D_{size}$ is the data size of a single input image. The throughput of a certain FPGA is $Th_{k,FPGA}^i = \frac{1BW_{pcie}}{Th_{k,m}^i}$, where $\hat{b}_k^i F_{GPU}(1)$ is the latency of the FPGA on $DL_k$ with a single input image. FPGA usually processes a single input image without batching so its batch size is 1. Besides, the active powers of a certain GPU and a certain FPGA in the node $s_i$ are denoted by $P_{act}^{GPU}$ and $P_{act}^{FPGA}$, respectively. For simplicity, the idle powers of accelerators are ignored since their idle powers are negligible compared to their active powers. It is usually known that the power of GPU, $P_{act}^{GPU}$, is higher than those of FPGA, $P_{act}^{FPGA}$ and especially, the throughput/watt of FPGA is higher than those of GPU [5] when the model component $DL_k$ is a convolutional layer. To simplify the notations, hereinafter we will omit FPGA or GPU from the throughput and power notation and replace it with the index of an accelerator.

The processing time of the $N_i$ accelerators is denoted as $[e_{k,1}^i, e_{k,2}^i, \ldots, e_{k,N_i}^i]$. Then, $e_{k,1}^i = \frac{D^i_{k,1}}{Th_{k,1}}$. This satisfies $D^i_k = \sum_{j \in [N_i]} D^i_{k,j}$. And then, the output data $D_{k}$ from the $N_i$ accelerators is transmitted to the host main memory via PCIe.

2. The target explainable DNN model is already loaded in all accelerators.
The minimum processing time $p_{k}^i$ has the following relationship with the optimal data distribution $[D_k^{i;1}, \ldots, D_k^{i;N_i}]$: 

$$p_{k}^i \approx c_{k}^i = \frac{D_{k}^{i;1}}{Th_k} + \cdots + \frac{D_{k}^{i;j}}{Th_k} + \cdots + \frac{D_{k}^{i;N_i}}{Th_k}.$$  

(3)

**Proof.** If we assume that the optimal solution $D_k^{i,j}$, which does not hold Eq. (3) and holds $D_k^{i,j} > D_k^{i,r}$, exists, $D_k^{i,r}$, which holds $D_k^{i,r} < D_k^{i,j}$, exists based on the condition $D_k = \sum_{j \in [N_i]} D_k^{i,j}$. That is, $D_k^{i,j}$ is not the optimal solution and leads to a contradiction since $p_{k}^i = \frac{D_{k}^{i;1}}{Th_k} < p_{k}^i = \frac{D_{k}^{i;N_i}}{Th_k}$. □

Based on Lemma 1, we prove that the minimum processing time $p_{k}^i$, can be achieved with the optimal data distribution, in which the processing time of each accelerator becomes equal.

**Theorem 1.** For the input image $D_i$ and the target explainable DNN model, the minimum processing time $p_{k}^i$ of the accelerator node $s^i$ exists and is derived as follows:

$$p_{k}^i \approx D^i \left( \frac{1}{Th_0} + \sum_{j \in [N_i]} \frac{\sigma_k}{Th_k^i} \right).$$  

(4)

**Proof.** Based on Lemma 1, the optimal input data of $D_k^{i,j}$ for the model Component $k$ of Accelerator $j$ is $D_k^{i,j} = c_{k}^i * Th_k^j$. Since $D_k = \sum_{j \in [N_i]} D_k^{i,j} = \sum_{j \in [N_i]} c_{k}^i * Th_k^j = c_{k}^i \sum_{j \in [N_i]} Th_k^j$, $c_{k}^i = \frac{D_k}{\sum_{j \in [N_i]} Th_k^j}$. For the input image $D_k^i$ and the component $DL_k$, the PCI express bandwidth $BW_{pci}$ is considerably higher compared to the throughput of accelerators. The processing time $p_{k}^i$ of the accelerator node $s^i$ can be approximated by $c_{k}^i$.

$$i \text{ if } Th_k^j \ll BW_{pci}, \forall j \in [N_i], \quad p_{k}^i \approx c_{k}^i.$$  

(5)
Each \( c_k \) is the largest value among the processing time of each accelerator. As a result, the total minimal processing time \( p_t \) of the accelerator node \( s \) on the input data \( D \) derived as follows:

\[
p_t = p_t^0 + \sum_{k \in [K]} P_{k,act}^0 \cdot p_t^0 + \sum_{k \in [K]} P_{k,id}^0 \cdot p_t^0 + \sum_{k \in [K]} \sum_{j \in [J]} P_{k,act}^{j,0} + c_k^{i,j},
\]

where \( P_{act}^0 \) is active power of CPU for pre-processing in node \( s \) [15, 16]. \( P_{id}^0 \) is idle power of CPU for pre-processing in node \( s \). \( P_{act}^{j,0} \) is active power of \( j \)-th accelerator (FPGA or GPU) for inferencing in node \( s \). \( p_t_i \) is pre-processing time on CPU in node \( s \). \( p_t_i^{i,j} \) is processing time on accelerators in node \( s \). \( c_k^{i,j} \) is main processing time on \( j \)-th accelerator in node \( s \).

Hence, the total energy consumption in all the nodes is defined as

\[
E = \sum_i E^i \cdot x^i, \tag{8}
\]

where \( x^i \) is the variable of \( X \).

**ASLM Scheme on FPGA-GPU.** In the explainable DNN model inference process for the total input image size \( D \), the goal of the adaptive scheduler is to achieve the minimum energy cost while satisfying the latency constraints \( L^{inf} \).

The decision variable is \( X = [x^1, \ldots, x^i, \ldots, x^M] \) and the objective function is defined as follows:

\[
\mathcal{P}_1: \min_{X,D} E \tag{9}
\]

subject to \( C_1: p_t^i \leq L^{inf}, \forall i \in [M]. \tag{10}\)

The constraints \( C_1 \) means that The processing time \( p_t^i \) for data \( D \) in the accelerator node \( s \) does not exceed the latency constraints \( L^{inf} \).

This problem is a mixed-integer linear programming (MILP) problem with integer variables, the allocation strategy \( \mathcal{X} = [x^1, \ldots, x^i, \ldots, x^M] \) and real variables, the data assignment \( \mathcal{D} = [D^1, \ldots, D^i, \ldots, D^M] \). We can solve it with some optimization methods, such as branch and cut and simplex algorithm, provided by CPLEX [22].

However, this MILP problem is NP-hard, so we propose a simple and intuitive heuristic algorithm that finds the near-optimal solution. In the performance evaluation Section 4, we will show that our heuristic algorithm achieves almost near-optimal solutions.

First, \( C_1 \) can be removed by assuming the data assignment to each accelerator node is the maximum data size that the accelerator node can handle until the latency constraint \( L^{inf} \). That is \( p_t^i = L^{inf}, \forall i \).

Based on Eq. (11), the data assignment to the nodes, \([D^1, \ldots, D^i, \ldots, D^M]\), is determined

\[
D^i := \frac{L^{inf}}{\frac{1}{P_0^0} + \sum_{k \in [K]} \sigma_k^0}, \forall i \in [M]. \tag{11}
\]

Second, it constructs the ranked ratio of energy cost per maximum data size, denoted as \( S \), for every accelerator node

\[
\begin{align*}
u & = \arg\min \frac{E^i}{D^i}, \text{ where } \frac{L^{inf}}{\frac{1}{P_0^0} + \sum_{k \in [K]} \sigma_k^0}. \tag{12}
\end{align*}
\]

Last, the accelerator nodes with low-efficiency are allocated one by one until the given input \( D \) can be processed within the latency constraints \( L^{inf} \).

\[
\sum_{i \in [M]} D^i \cdot x^i < D. \tag{13}
\]

The entire process of the ASLM scheme is described in Algorithm 1.

**Algorithm 1. Acceleration Scheduling Scheme With Layer-Level Management of Explainable DNN on FPGA-GPU**

**Input:**

- \( E^i \): Energy consumption in accelerator node \( s_i \),
- \( D^i \): Data assignment to accelerator node \( s_i \)

**Output:**

- \( S \): Ranked ratio of “energy cost per maximum data size”

**Function ConstructRatio\((E^i, D^i)\):**

\[
\begin{align*}
S & = \emptyset, \\
T & = M = [M] \\
\text{Repeat} & \text{ } \\
& \quad u = \arg\min_i \frac{E^i}{D^i}, \text{ where } \frac{L^{inf}}{\frac{1}{P_0^0} + \sum_{k \in [K]} \sigma_k^0} \\
& \quad S = (S \cup \{u\}) \\
& \quad T = T - \{u\} \\
& \text{Until } T = \emptyset
\end{align*}
\]

**End Function**

**Input:**

- \( S \): Ranked ratio of “energy cost per maximum data size”

**Output:**

- \( \mathcal{X} \): Acceleration Node Allocation Strategy

**Function NodeAssignment\((S)\):**

\[
\mathcal{X} = 0 \\
\text{Repeat} & \text{ } \\
x^{S[0]} & = 1 \\
S & = S - S[0] \\
\text{Until } \sum_{i \in [M]} D^i \cdot x^i < D
\]

**End Function**
3.3 Adaptive Unlabeled Data Selection (AUDS) With Confidence Threshold for DNN Retraining Cost Reduction

In step 2 of the HPC ground station system, we adopt the algorithm, proposed in [23], for the AUDS scheme which selects the most informative unlabeled data on retraining. The selected data is passed to human supervisors for labeling. The total input data pool of $m$ classes and $n$ samples is denoted by $I = \{x_i | i \in [n]\}$. The label of $x_i$ is denoted as $y_i$. If $x_i$ is in the $j$th class, $y_i = j, j \in [M]$. While the data scale continues to grow, almost all data cannot be unlabeled due to high labeling cost and slow labeling speed. Utilizing informative unlabeled data is important to accelerate the retraining. Therefore, the AUDS scheme organizes the training data $D_{tr}$ composed of unlabeled images $D_{tr}^u$ and labeled images $D_{tr}^l$.

The DNN retraining problem is defined as follows:

$$P_1 : \min_{W; y_i \in D_{tr}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} 1(y_i = j) \log p(y_i = j|x_i; W),$$

where $1(\cdot)$ is the function that $1(\text{true}) = 1$ and $1(\text{false}) = 0$, and $W$ is the model parameters. $p(y_i = j|x_i; W)$ is the softmax output of the model $W$ on $x_i$, for the $j$th class.

First, the AUDS scheme sorts the total input data pool according to the AL criteria that uncertain samples are often the most informative for model update. And then, those most uncertain samples are manually labeled by human supervisors and added into labeled training data $D_{tr}^l$. Those most certain samples are pseudo-labeled and added into unlabeled training data $D_{tr}^u$.

For $x_i$, the model predicts its label as a probability vector $p(y_i = j|x_i; W) \in [0, 1]^m$. Let $s(x)$ be the confidence of $x_i$. Measuring the maximum confidence that the model has in any class

$$s(x) = \max_j(p(y_i = j|x_i; W)).$$

**Definition 2.** Configuration Threshold. Let $\theta$ be the confidence threshold of $s(x)$, described in Fig. 5. The high confidence

3.4 Retraining Acceleration Scheduling (RAS) Based on Semi-Supervised Learning With Data Parallelism

In step 3 of the HPC ground station system, the RAS scheme schedules the retraining with the training data $D_{tr}$ composed of unlabeled images $D_{tr}^u$ and labeled images $D_{tr}^l$. Its objective is to achieve the minimum energy cost while satisfying latency constraints, denoted as $L_{tr}$. To do this, it determines how to allocate the accelerators in the cluster and distribute the training data $D_{tr}$ to them for invoking the target explainable DNN model retraining tasks. Structure of Semi-Supervised Learning Based Retraining. The retraining task is based on
Synchronous SGD, which is one of the data parallelism (DP) methods [18]. The DP reduces one iteration time by dividing the mini-batch size into the allocated accelerators [19]. The mini-batch size of i th GPU is denoted as $b_i^r$. The total batch size is $B = \sum_b b_i^r$ in an iteration. The batch size $B$ is randomly sampled from the training data $D_{tr}$. The number of iterations in an epoch is given to $Iter = \frac{B}{2}$ and the number of epochs $ep$ is determined by system operators. We cannot use Synchronous SGD directly because of the presence of unlabeled data. As described in Section 3.3, we utilize the informative unlabeled data to accelerate the retraining and overcome the bottleneck due to high labeling cost and slow labeling speed. Therefore, we adopt the semi-supervised learning method, Label Guessing [24], which makes pseudo-labels by invoking the target model inference regarding unlabeled data.

**GPU Resource Allocation.** For retraining, we use GPUs. The available GPUs after GPU allocation for inference in step 1 are denoted as $X_{tr}$, which is the allocation vector on GPUs for retraining, as follows:

$$X_{tr} = \{x_{tr}^1, \ldots, x_{tr}^C\}, \text{ where } x_{tr}^i = \{0, 1\},$$

where $C$ is the number of available GPUs. $x_{tr}^i = 1$ means that the GPU $i$ is allocated for retraining and otherwise $x_{tr}^i = 0$.

**Latency Modeling in Retraining Process.** For batch size $B$, the retraining latency with allocation Strategy $X_{tr}$ is defined as follows:

$$l_{tr} = l_{fb} * Iter * ep + l_{lg},$$

where $l_{fb}$ is the label guessing latency.

According to unlabeled training images $D_{tr}^l$, the label guessing latency with allocation Strategy $X_{tr}$ is defined given by Eq. (22)

$$l_{lg} = \frac{D_{tr}^l}{\sum_{i \in [C]} T_{th}^i * x_{tr}^i},$$

where $T_{th}^i$ is the throughput of $i$ th-GPU. We show the label guessing latency through Theorem 1.

Based on the mechanism of Synchronous SGD as described in Fig. 7, an iteration executes one Feed-Forward and Backpropagation (FB) with the batch $B$ of training data $D_{tr}$ on $C$ GPUs. The RAS scheme determines the optimal mini-batch $b_i$ to be processed by each GPU for training on $B$ in each FB

$$b_{tr} = (b_{tr}^1, \ldots, b_{tr}^C), \text{ then } B = \sum b_{tr}^i.$$  (23)

The FB Latency at i th GPU consists of the processing time $l_{p}^i(b_{tr}^i)$ for mini-batch $b_{tr}^i$ and the transfer time $l_{tr}^i$ to send the gradient to the parameter server and receive updated models from the parameter server, respectively [20]

$$l_{p}^i(b_{tr}^i) = l_{p}^i(b_{tr}^i) + l_{tr}^i.$$  (24)

The processing time $l_{p}^i(b_{tr}^i)$ is defined with the parameters $\alpha^i$ and $\beta^i$ implying the performance of i th GPU [20]

$$l_{p}^i(b_{tr}^i) = \alpha^i * b_{tr}^i + \beta^i.$$  (25)

In the heterogeneous accelerator environment, each GPU has different processing speed characteristics per batch, so if the same mini-batch is determined, straggler occurs and FB operations are getting slow down [20]. $b_{tr}^i$ is adjusted to minimize the FB Latency of the accelerator with maximum FB Latency to obtain the minimized FB performance time $l_{fb}$ and mini-batch of each accelerator. It is given to Eq. (26)

$$\min \max_i l_{fb}^i(b_{tr}^i).$$  (26)

If $b_{tr}^i$ is a continuous variable, the optimal solution of Eq. (26) on the FB latency is derived as follows:

$$l_{fb}^i = \frac{B + \sum_i (\frac{\beta^i}{\alpha^i}) * x_{tr}^i}{\sum_i (\frac{\beta^i}{\alpha^i}) * x_{tr}^i} + l_e$$

$$l_{p}^i = \frac{B + \sum_i (\frac{\beta^i}{\alpha^i}) * x_{tr}^i}{\sum_i (\frac{\beta^i}{\alpha^i}) * x_{tr}^i}$$

$$b_{tr}^i = l_{fb}^i - \alpha^i - l_{tr}^i, \forall i.$$  (29)

where we assume that the communication latencies of $X_{tr}$ are same as $l_e$.

**Energy Cost Modeling in Retraining Process.** During retraining process, the active processing time and power in GPU are given to $l_{act}^i * Iter * ep + l_{lg}$ and $P_{act}^i$ and the idle time and power spent on transmission in GPU are $l_{idle}^i * Iter * ep$ and $P_{idle}^i$. Pipelining between processing and transmission cannot be applied because the training can be continued for the next batch after synchronization has been established.

For batch size $B$ and training data $D_{tr}$, the retraining energy with allocation Strategy $X_{tr}$ is defined as processing time ($E_{tr}$) [15], [16]

$$E_{tr} = \sum_{i \in [C]} E_{tr}^i * x_{tr}^i$$

$$E_{tr}^i = P_{act}^i * l_{act}^i + P_{idle}^i * l_{idle}^i * C $$

RAS Scheme Based on Semi-Supervised Learning With Data Parallelism. In the retraining process for the training data $D_{tr}$, the goal of the RAS scheme is to achieve the minimum energy cost while satisfying the latency constraints $L_{fb}^i$. The decision variables are an accelerator allocation strategy $X_{tr} = [x_{tr}^1, \ldots, x_{tr}^C]$ and a confidence threshold $\theta$. 
First, the RAS scheme determines the confidence threshold \( \theta \) adjusting several iterations (\( \text{Iter} \)) with the training data \( D_{tr} \).

\[
\text{Iter} = \frac{D_{tr}}{B} = \frac{\{x|s(x) > \theta, x \in I\}}{B}. \tag{32}
\]

The initial confidence threshold is obtained using the best empirical value. If the training data \( D_{tr} \) selected by \( \theta \) cannot be processed within \( L^{tr} \) with whole available GPUs, the RAS scheme reduces \( D_{tr} \) and \( \text{Iter} \)

\[
l_{tr} = l_{\text{fb}}(X_{tr}) \ast \text{Iter}(\theta) \ast \text{ep} + l_{\text{fb}} \rangle L^{tr},
\]

where \( x_{tr}^i = 1, \forall i \in [C] \).

The RAS scheme increases the initial confidence threshold \( \theta \) to reduce the training data size \( D_{tr} \) calculated in the AL process (step 2) until \( L^{tr} \) is satisfied.

(a) Increase Threshold of Uncertainty

\[
\Delta_{\text{unc}} = l_{tr} - L^{tr} \tag{34}
\]

\[
\theta_{\text{new}} = \theta + \varepsilon \ast \Delta_{\text{unc}} = \theta + \varepsilon \ast (l_{tr} - L^{tr}). \tag{35}
\]

(b) Reduce Training Data Size

\[
D_{tr} \downarrow = \{x|s(x) > \theta, x \in I\}. \tag{36}
\]

(c) Reduce the number of iterations and Epoch Latency

\[
\text{Iter} \downarrow = \frac{D_{tr}}{B} \tag{37}
\]

(d) Repeat Until

\[
l_{tr} \leq L^{tr}. \tag{38}
\]

Contrary, if the training data \( D_{tr} \) selected by \( \theta \) can be processed within \( L^{tr} \) with whole available GPUs, the RAS scheme additionally reduces retraining cost \( E_{tr} \) with the latency constraints \( L^{tr} \) by controlling the accelerator allocation strategy \( X_{tr} = \{x_1^{tr}, \ldots, x_{tr}^{i}, \ldots, x_{tr}^{C}\} \) and the objective function is defined as follows:

\[
P_2 : \text{minimize} \quad E_{tr}
\]

subject to \( C_1 : l_{tr} \leq L^{tr} \)

\[
P_2' : \text{minimize} \quad \sum_{i \in [C]} E_{tr i}^{i} \ast x_{tr i}^{i}
\]

\[
= \left\{ \sum_{i \in [C]} P_{act}^{i} \ast x_{tr i}^{i} (l_{tr}) \right\} + \left\{ \sum_{i \in [C]} P_{idle}^{i} \ast x_{tr i}^{i} \right\} \ast (l_{tr} \ast \text{Iter} \ast \text{ep})
\]

subject to \( C_1 : l_{tr} < L^{tr}. \) \tag{40}

However, since it is hard to solve (NP-hard problem), we propose a heuristic algorithm to reduce computational complexity. The objective function \( P_2 \) to find the optimal retraining cost can be configured in the form of GPU active Power \( \sum_{i \in [C]} P_{act}^{i} \ast x_{tr i}^{i} \) \times One Iteration Latency \( l_{tr} \). For simplicity, the GPU idle power is ignored. We propose a heuristic on accelerator allocation \( X_{tr} \).

First, it starts with setting all items of \( X_{tr} \) as 1. It means that we consider using all of available GPUs.

Second, it obtains the energy cost reduction for the processing time increase, when one GPU among \( X_{tr} \) is decided not to use

\[
\frac{\Delta E_{tr}}{\Delta t_{tr}} = \frac{E_{tr}' - E_{tr}}{l_{tr}' - l_{tr}}. \tag{41}
\]

Third, it excludes one inefficient GPU which has the highest value on \( \Delta E_{tr} \). If the GPU-\( i \) is chosen, it set as \( x_{tr i} = 0 \). It repeats this process right before the latency constraints \( L^{tr} \) is exceeded. This algorithm is finished and uses the result of \( X_{tr} \) as a solution to problem \( P_2 \). The entire process of the RAS scheme is described in Algorithm 2.

### Algorithm 2: Semi-Supervised Learning Based Retraining Acceleration Scheduling Scheme With Data Parallelism

**Input:**

- \( D_{tr} \): Input Retraining Data,
- \( B \): Batch Size,
- \( C \): Available GPUs

**Output:**

- \( X_{tr} \): GPU Allocation Strategy

**Function:**

if \( l_{tr} > L^{tr} \) then

\[
X_{tr} \leftarrow \{1\}
\]

Repeat

\[
\Delta_{\text{unc}} = l_{tr} - L^{tr}
\]

\[
\theta_{\text{new}} = \theta + \varepsilon \ast \Delta_{\text{unc}}
\]

\[
D_{tr} \downarrow = \{x|s(x) > \theta, x \in I\}
\]

\[
\text{Iter} \downarrow = \frac{D_{tr}}{B}
\]

Until \( l_{tr} \leq L^{tr} \)

else

Repeat

Calculate \( \frac{\Delta E_{tr}}{\Delta t_{tr}} = \frac{E_{tr}' - E_{tr}}{l_{tr}' - l_{tr}} \)

Exclude the GPU of the highest \( \frac{\Delta E_{tr}}{\Delta t_{tr}} \) value

Until \( l_{tr} > L^{tr} \)

end

Return \( X_{tr} \)

End Function

## 4 PERFORMANCE EVALUATION AND DISCUSSION

In this section, we evaluate the processing time and energy cost of the proposed cooperative scheduling schemes for explainable DNN acceleration in satellite image analysis and retraining, comparing with the conventional DNN acceleration schemes. The proposed schemes optimize the pre-processing component as well, reflecting the heterogeneity of CPU, and the evaluation results of the proposed schemes definitely include the effect of the pre-processing tasks. However, we focus on the processing time and energy consumption performance of the main model components processed in accelerators such as CL or FC since the heterogeneity of CPU in our experimental environment is negligible compared to the heterogeneity of accelerators. In this evaluation, the scheduling scheme of Nexus [28] is used as the conventional scheme for satellite image analysis and Homogeneous DP [18] and Heterogeneous DP [20] are used as the conventional schemes for
satellite image retraining. The inference scheduling schemes of Nexus, assuming a homogeneous HPC environment, pack the workload each accelerator maximally and minimize the resource usage considering the latency constraints. Homogeneous DP is the training scheme distributing data evenly across all of available resource [18]. Heterogeneous DP is the training scheme distributing data optimally across all of available resources for straggler mitigation [20].

In addition, we also compare with the optimal solution of our problems, derived from the MILP optimization algorithm provided by CPLEX [22], and show how close the proposed schemes are to the optimal result.

For the testbed establishment, we implement the proposed cooperative scheduling schemes, as shown in Fig. 8. We build the heterogeneous HPC environment with 7 accelerator nodes, each of which contains multiple heterogeneous CPUs, memory, and accelerators (FPGA-GPU) (Geforce series, GTX1080/GTX1080Ti/RTX2080Super, FPGA, Arria 10 GX) as shown in Table 2. The ASLM, AUDS, and RAS schemes determine the best accelerator node, data selection, and GPU allocation, respectively, for a given workload with input images \( D \) in a certain period. After the images \( D \) are analyzed with the ASLM scheme, the delay occurred from the AUDS scheme and the supervisor's labeling are ignored in this experiment. We assume that the training data \( D_{tr} \) composed of unlabeled images \( D_{tr}^{l} \), and labeled images \( D_{tr}^{l} \), is generated without delay and is directly applied to the RAS scheme for retraining of the target explainable DNN model. Besides, We build a monitoring module for each accelerator node using Nvidia-smi [31] and Xilinx xbutil tool [32] to monitor the GPU and FPGA power usage periodically in real-time.

For evaluation, we use two models as a target explainable DNN model; Resnet-152 (classification) [25], and Faster-RCNN (Object Detection) [21] (but, to evaluate the RAS scheme, we only use Resnet-152). Also, we attach Grad-CAM on the models for visual explanation. We use large-scale image datasets such as DOTA [26] and AID [27], as shown in Fig. 9. We run Pytorch for deep learning framework [29]. We use CUDA 9.0 and cuDNN 7.0 [30] to accelerate the DNN inference speed based on GPU. We measure two metrics: processing time (sec) and energy consumption (J). The processing time means the total time to complete the given workload and the energy consumption means the total energy consumption in the accelerator nodes during processing.

### 4.1 Experimental Results and Analysis

Figs. 10 and 11 shows the results of the ASLM scheme on Resnet-152 and Faster-RCNN, respectively, in terms of processing time (sec) and energy consumption (J) with various workloads. The workloads are set as \([3200, 4800, 6400, 8000]\) (Number of Requests). A request is composed of an input image to invoke the target model inference. The latency constraints \( L_{inf}^{max} \) is set as 20s for Resnet-152 and 70s for Faster-RCNN, respectively.

In Figs. 10 and 11, the ASLM scheme shows better performance on energy consumption compared to the conventional
schemes in every cases while guaranteeing the latency constraints, 20s and 70s. Especially, in Fig. 10, the ASLM scheme reduces energy consumption by \(39, 29, 22, 10\)% over workloads compared to Nexus. As the workload gets bigger, its effectiveness decreases because the options it can choose to reduce energy consumption become smaller. In the same way, in Fig. 11, it reduces energy consumption by \(5.72, 4.36, 3.09, 1.28\)% over workloads compared to Nexus while guaranteeing the latency constraints. Its effectiveness becomes smaller than the results on Resnet-152, because the workload of Faster-RCNN is larger than that of Resnet-152.

Last, the ASLM scheme nearly achieves the optimal result on processing time and energy consumption, derived from the MILP optimization algorithm provided by CPLEX [22].

The results show that the ASLM scheme effectively optimizes the analysis performance with the latency and energy cost modeling, Eqs. (1) and (7), reflecting the heterogeneity of accelerator environment. Especially, its layer-level management on explainable DNN with the latency and energy cost modeling removes the inefficiency of Nexus. Nexus packs the workload into each accelerator as much as possible within the given latency constraints in order to minimize the number of used accelerators. However, the modeling of Nexus, which do not consider the energy consumption of accelerators and assumes the accelerators are homogeneous, makes the inefficient decisions to use the bad throughput/watt accelerators. In addition, since it processes a DNN model as a unit of a task, it cannot avoid allocating FC to FPGA and causes performance degradation.

Fig. 12 shows the results of the RAS scheme on retraining Resnet-152, in terms of processing time (sec) and energy consumption (J) with various batch sizes. For simplicity, we shows the average result of an iteration. Also, in this evaluation, we do not consider the label guessing latency because it is negligible in the retraining latency. The batch sizes are set as \([64, 128, 256]\]. The latency constraints \(L_{tr}\), the training data \(D_{tr}\) and the number of epochs \(ep\) are set as \(300s, 6400\) and \(40\), respectively. Then, the latency constraints on an iteration is set as \(0.3s\).

In the same way with the ASLM scheme’s evaluation, the RAS scheme shows better performance on energy consumption in retraining compared to the conventional schemes
entirely while guaranteeing the latency constraints, 0.3s. In Fig. 12, the RAS scheme reduces energy consumption by \(41.54, 15.83, 2.8\)% over batch sizes compared to Heterogeneous DP. Homogeneous DP and Heterogeneous DP just use all of available accelerators and decide the data distribution to minimize the processing time. They do not consider the remaining time within the latency constraints which can be utilized to save the energy consumption. Meanwhile, the RAS scheme additionally reduces the accelerator usage to minimize the energy consumption, utilizing the remaining time within the latency constraints. It effectively optimizes the retraining performance with the latency and energy cost modeling, Eqs. (33) and (30), considering the heterogeneity of accelerator environment.

Fig. 13 shows the results of the AUDS scheme in terms of accuracy (%) and Number of Training Data (#) and elapsed time(sec). In this experiment, we fix the number of labeled data as 1000 and the unlabeled data might be input into the AUDS scheme in an incremental way. We evaluate it with respect to random sampling based on the fixed threshold. The AUDS scheme achieves competitive accuracy with a smaller number of training samples in comparison to the fixed threshold approach. It provides higher accuracy with the same number of training samples in comparison to the fixed threshold approach.

5 CONCLUSION

In this paper, we addressed the limitations of the conventional DNN acceleration systems which cause serious performance degradation on energy cost in satellite image analysis and retraining. To overcome these problems, we discussed new explainable DNN acceleration scheduling schemes. Utilizing the latency and energy cost modeling that reflects the layer-level management of explainable DNN in analysis and the confidence level criteria and data parallelism in retraining, we propose cooperative scheduling schemes to minimize the analysis or retraining cost and guarantee the latency constraints. We implemented the cooperative scheduling for explainable DNN acceleration in heterogeneous HPC environment based on FPGA-
GPU and conducted real satellite image analysis and retraining experiments with a large-scale aerial image dataset, such as DOTA and AID. In these experiments, the results showed that the ASLM and TAS schemes provide the optimized processing time and cost performance with respect to explainable DNN acceleration, utilizing the latency and energy cost modeling reflecting the heterogeneity of accelerator environment. In the case of ResNet-152, the ASLM and TAS schemes reduced the energy cost of their conventional schemes by up to about 40% while guaranteeing the latency constraints. Furthermore, the results showed that the AUDS scheme achieved competitive accuracy with a smaller number of training samples in comparison to the fixed threshold approach. We showed that the AUDS scheme alleviated the bottleneck of supervisors workload and realize the fast processing and convergence of explainable DNN in satellite image analysis and retraining.

REFERENCES

[1] H. Fahmy, F. Pastore, M. Bagherzadeh, and L. Briand, “Supporting DNN safety analysis and retraining through heat-map-based unsupervised learning,” IEEE Trans. Rel., 2021, doi: 10.1109/TR.2021.3074750.

[2] J. Heo et al., “Cost-effective interactive attention learning with neural attention processes,” in Proc. 37th Int. Conf. Mach. Learn., 2020.

[3] X. Hou and T. Han, “TrustServing: A quality inspection sampling approach for remote DNN services,” in Proc. 17th Annu. IEEE Int. Conf. Sens. Commun. Netw., 2020, pp. 1–9.

[4] A. Putnam et al., “A reconfigurable fabric for accelerating large-scale datacenter services,” in Proc. ACM/IEEE 41st Int. Symp. Comput. Archit., 2014, pp. 13–24.

[5] E. Nurvitalidi et al., “Can FPGAs beat GPUs in accelerating next-generation deep neural networks?,” in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays, 2017, pp. 3–14.

[6] NVIDIA NVIDIA DGX-1, 2017. [Online]. Available: http://www.nvidia.com/dg

[7] A. M. Caulfield et al., “A cloud-scale acceleration architecture” in Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchit., 2016, pp. 1–13.

[8] M. Staveley, “Applications that scale using GPU Compute,” in AzureCon 2015, Microsoft, Aug, 2015.

[9] J. Barks, “Build 3D streaming applications with EC2s New G2 instance Type,” Amazon Web Service, Nov. 2013.

[10] J. Ouyang, S. Lin, W. Qi, Y. Wang, B. Yu, and S. Jiang, “SDA: Software defined accelerator for large-scale DNN systems,” in Proc. IEEE Hot Chips 26 Symp., 2014, pp. 1–23.

[11] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual explanations from deep networks,” in Proc. ACM Conf. Inf. Knowl. Manage., 2017, pp. 2067–2070.

[12] H. Zhou, S. Bateni, and C. Liu, “8-bit DNN: Supervised streaming and scheduling for GPU-accelerated real-time DNN workloads,” in Proc. IEEE Real-Time Embedded Techn. Appl. Symp., 2018, pp. 190–201.

[13] Z. Fang et al., “QoS-Aware scheduling of heterogeneous servers for inference in deep neural networks,” in Proc. ACM Conf. Inf. Knowl. Manage., 2017, pp. 2067–2070.

[14] D. Batra et al., “Jointly optimizing preprocessing and inference for DNN-based visual analytics,” in Proc. VLDB Endow., vol. 14, no. 10, Oct. 2020, pp. 87–100, doi: https://doi.org/10.14778/3425879.3425881.

[15] X. Mei, X. Chu, H. Liu, Y. Leung, and Z. Li, “Energy efficient real-time task scheduling on CPU-GPU hybrid clusters,” in Proc. IEEE Conf. Comput. Commun., 2017, pp. 1–9.

[16] R. A. Bridges, N. Imam, and T. M. Mintz, “Understanding GPU performance: A survey of profiling, modeling, and simulation methods,” ACM Comput. Surv., vol. 49, no. 3, pp. 1–27, 2016.

[17] O. Sener and S. Savarese, “Active learning for convolutional neural networks: A core-set approach,” in Proc. Int. Conf. Learn. Representations, 2018.

[18] J. Dean et al., “Large scale distributed deep networks,” in Proc. 25th Int. Conf. Neural Inf. Process. Syst., 2012, pp. 1223–1231.

[19] P. Goyal et al., “Accurate, large minibatch SGD: Training ImageNet in 1 hour,” 2017, arXiv:1706.02677.

[20] E. Yang, D.-K. Kang, and C.-H. Youn, “BOA: Batch orchestration algorithm for straggler mitigation of distributed DL training in heterogeneous GPU cluster,” J. Supercomput., vol. 76, no. 1, pp. 47–67, 2020.

[21] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 1, 2017, doi: 10.1109/TPAMI.2016.2577651.

[22] Manual, C.U., “IBM ILOG CPLEX optimization studio,” Version 1987, 12, 1987–2018.

[23] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin, “Cost-effective active learning for deep image classification,” IEEE Trans. Circuits Syst. Video Technol., vol. 27, no. 12, pp. 2591–2600, Dec. 2017.

[24] S. Song, D. Berthelot, and A. Rostamizadeh, “Combining Mix-Match and active learning for better accuracy with fewer labels,” 2019, arXiv:1912.00594.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[26] G.-S. Xia et al., “DOTA: A large-scale dataset for object detection in aerial images,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 3974–3983.

[27] G.-S. Xia et al., “AID: A benchmark data set for performance evaluation of aerial scene classification,” IEEE Trans. Geosci. Remote Sens., vol. 55, no. 7, pp. 3965–3981, Jul. 2017.

[28] H. Shen et al., “Nexus: A GPU cluster engine for accelerating DNN-based video analysis,” in Proc. 27th ACM Symp. Operating Syst. Princ., 2019, pp. 322–337.

[29] I. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” Adv. Neural Inf. Process. Syst., vol. 32, pp. 8026–8037, 2019.

[30] S. Cook. “CUDA programming: a developer’s guide to parallel computing with GPUs,” Newnes, 2012. Accessed: Dec. 2018. [Online]. Available: https://developer.nvidia.com/

[31] NVIDIA Management Library NVML. Accessed: Apr. 2019. [Online]. Available: https://developer.nvidia.com/nvidia-managementlibrary-nvml

[32] Xilinx, Xilinx Board Utility, 2020. [Online]. Available: https://www.xilinx.com/htmldocs/xilinx2020 1/vitis doc/xbutilutility.html

Woo-Joong Kim (Member, IEEE) received the MS degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1981 and 1985, respectively, and the PhD degree in electrical engineering from the Korea Advanced Institute of Technology (KAIST), Daejeon, South Korea. He is currently a staff engineer with Samsung Electronics, Memory Business Division. His research interests include cloud computing, cloud storage service, and data center management.

Chan-Hyun Youn (Senior Member, IEEE) received the BSc and MSc degrees in electronics engineering from Kyungpook National University, Daegu, South Korea, in 1981 and 1985, respectively, and the PhD degree in electrical and communications engineering from Tohoku University, Sendai, Japan, in 1994. Before joining the University, from 1986 to 1997, he was the leader of High-Speed Networking Team, KT Telecommunications Network Research Laboratories, where he had been involved in the research and developments of centralized switching maintenance system, high-speed networking, and ATM network. Since 2009, he has been a professor with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. He was an associate professor of office of planning and budgets in KAIST from 2013 to 2017. He also is a director of Grid Middleware Research Center and XAI Acceleration Technology Research Center, KAIST, where he is developing core technologies that are in the areas of high performance computing, edge-cloud computing, AI acceleration system and developments of centralized switching maintenance system, high-speed networking, and ATM network. For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdii.