IOWA: An I/O-Aware Adaptive Sampling Framework for Deep Learning

Shuang Hu
Zhejiang University
shu0421@zju.edu.cn

Weijian Chen
Zhejiang University
weijianchen@zju.edu.cn

Yanlong Yin
Zhejiang University
yinyanlong@gmail.com

Shuibing He
Zhejiang University
heshuibing@zju.edu.cn

***Abstract*—Training deep DNN models is time-consuming, especially when using large datasets. In the standard model training process, data instances are sampled uniformly and fed into the neural networks. However, not all instances contribute equally to the resulting model, and even the same data instance may affect the model differently in different training iterations. In addition to computational costs, I/O overhead can significantly impact the training speed, particularly for I/O-intensive processes. Given these observations, we propose an I/O-aware sampling metric in this paper. Building on this, we introduce an I/O-Aware Adaptive Sampling Framework (IOWA), which includes data profiling, adaptive data sampling, and redundant data instance replacement to accelerate the training process. Extensive experiments demonstrate that, compared to traditional DNN training processes, our approach can achieve up to a 3× speedup without compromising the resulting model.**

***Index Terms*—Deep Learning, Importance Sampling**

I. Introduction

Deep neural networks (DNNs) have achieved great success in domains such as computer vision [12], [24], speech recognition [11], and natural language processing [31]. In standard DNN training process, each data instance is treated equally during the training process. However, recent study reveals that data instances contribute differently in the model training process [4], [15], [19]. Some data are less informative, storing and training these instances has negligible benefit to improve the model accuracy.With these observations, the research community has proposed several strategies to filter out less informative data points to improve training efficiency. These strategies are broadly referred to as *importance sampling*. The general principle is to use a metric to quantify the importance of each data instance and reduce the sampling probability of those with lower importance values. Various metrics have been explored in the literature, including loss [15], [23], uncertainty [5], gradient approximation [1], [27], or self-defined boundary [17], [19], [26].

In the aforementioned sampling methods, importance values are quantified by a single criterion: their contribution to improving model accuracy. While reducing the training dataset via importance sampling can accelerate the training process by eliminating the need to compute less-informative data, there is still room for further improvement due to the following factors:

Yanlong Yin is the corresponding author.

First, *the existing metrics do not account for an instance's contribution to I/O cost*. A training process includes a data I/O phase and a data computing phase. While data instances may require the same amount of computation time, they can incur different I/O times. For example, training an instance directly from host memory requires significantly less I/O time compared to instances retrieved from remote storage, as it avoids network transmission and slow storage access. Therefore, although previous methods may be efficient for computation-intensive models, they are less effective for I/O-intensive models because I/O costs are not considered.

Second, *existing methods do not correlate data reduction with the model's learning ability*. The reduction in training data size is influenced not only by the intrinsic attributes of the data but also by the model's learning capacity. Removing the same amount of data instances affects the accuracy of a robust model less than it does for a weaker one, as demonstrated in Section II-B. Therefore, the degree of sampling selection should be adjusted during the training process.

Third, *data instances with high importance values may contain redundancy*. Storing redundant data instances incurs extra storage overhead, and loading these instances from local or remote storage for model training involves additional I/O and transmission costs. These costs can be minimized by using copies of data that are already loaded in memory or discarding similar copies.

Based on the above observations, we propose IOWA (an $\underline{\text{I}}/\underline{\text{O}}$-A$\underline{\text{w}}$are $\underline{\text{A}}$daptive Sampling Framework) to accelerate DNN model training without degrading model accuracy. More specifically, our contributions are as follows:

1) We propose a multi-criteria metric to evaluate the importance of training data instances, taking I/O-intensive scenarios into consideration.
2) Based on this multi-criteria metric, we introduce a new Bucket Sampling algorithm to select data instances for faster model training.
3) We develop a novel Redundancy Removed Importance Sampling (RRIS) algorithm to further reduce the training data size without sacrificing model accuracy.
4) We implement IOWA in PyTorch and conduct extensive experiments to evaluate the performance of our proposed approaches. The results show that our sampling methods can, at most, triple the training speed without affecting model accuracy.

TABLE I
EXPERIMENTAL SETUP FOR MOTIVATION 1 (THE BOLD TEXT SHOWS CONTROL GROUP'S SETUP.)

Device	Storage	Model	Worker
Tesla V100 PCIe 32GB	**Local HDD**	**ResNet**	**8**
48 Intel CPU @ 2.60GHz	Remote SSD	AlexNet	2

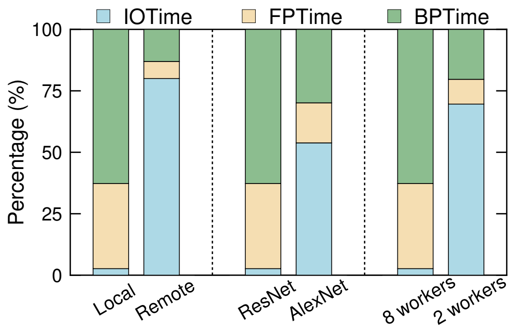


Fig. 1. The training time breakdown on forward, backward, and I/O with different storage devices, models, and number of workers.

II. BACKGROUND AND MOTIVATION

A. Existing Sampling Approaches in SGD

Uniform Sampling, which assigns equal importance to all the training data instances, is commonly used in stochastic gradient descent (SGD) optimization. For completeness, we briefly explain how uniform sampling is applied in SGD for DNN model training.

Given a neural network $f(w)$ parameterized by a set of weights w, and the loss function L, SGD is used to find the parameters w^* that minimize the overall loss L on dataset D, i.e.,

$$w^* = \arg\min_{w} \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{b} L_{ij}(f_w(x_j), y_j) \tag{1}$$

where B represents the total number of batches, b is the batch size, and (x_j, y_j) is the training data pair. To find the optimal w^*, SGD performs the following update until convergence:

$$w_{t+1} = w_t - \eta \sum_{i=1}^{b} \nabla_w L(f_w(x_i), y_i) \tag{2}$$

where η is the learning rate.

In both vanilla SGD (where $b = 1$) and mini-batch SGD (where $b > 1$), each training data instance is assumed to contribute equally and, therefore, has an equal chance of being sampled for model training. However, as mentioned earlier, different data instances may contributed differently during the training process [4], [15], [19]. Based on this observation, several sampling algorithms have been developed and published to filter out less informative data instances, aiming to reduce computation time and accelerate model training without significant or any accuracy loss.

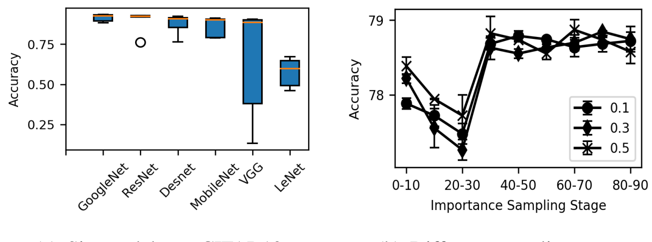


Fig. 2. Training accuracy flucturations. (a) The accuracy fluctuations when training six different models on CIFAR10 with fixed sampling rate 0.1, 0.3, and 0.5. (b) The accuracy fluctuations when training ResNet18 on Food101 with sampling rate 0.1, 0.3, and 0.5 at different stages in the course of training.

B. Motivation

Despite the advances in importance sampling approaches, the following three findings from this study suggest that these existing methods can be further improved.

First, *computation does not always dominate the training time cost*. To better understand the breakdown of DNN model training time, we conducted experiments comparing the time distribution when training ResNet and AlexNet on ImageNet under different hardware and software settings. The experimental setup is described in Table I, and the results are shown in Figure 1. We observe that, aside from computation (including forward and backward propagation), *I/O access costs are also significant*. In particular, when a lightweight model is trained on a fast computing device, or when datasets are stored on a slower storage device, data fetching can consume the majority of the time.

Second, *adaptive sampling rates should be applied when a model has higher discriminative ability*. As shown in Figure 2, reducing the training dataset size impacts models differently. More robust models, such as GoogleNet and ResNet, are less sensitive to dataset reduction, suggesting that they are less likely to forget encoded information. In contrast, models like LeNet and VGG are more vulnerable to reductions in training data size. Different phases of a single model training process also exhibit varying vulnerabilities to dataset reduction. Existing algorithms typically use a fixed sampling rate, i.e., filtering out the same proportion of the dataset in each epoch. However, the number of data instances to be sampled should depend on both the model and the data, and should be dynamically adjusted during training.

Third, *redundant data instances can lead to unnecessary I/O access costs*. To investigate redundancy in training datasets, we used perceptual hash algorithms (pHash) [32] to compute similarity scores among training instances in popular datasets: MNIST, CIFAR10, and CIFAR100. As shown in Figure 3, multiple redundant copies with high similarity scores exist, which can be removed from the training process without affecting model accuracy.

In this work, to address the first issue, we adopt a new multi-criteria metric to sample data instances, taking I/O access costs into consideration. Based on the new metric, we develop the

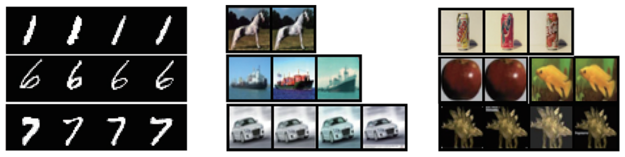


Fig. 3. The redundant images in MNIST, CIFAR10, and CIFAR100.

Bucket Sampling (BS) algorithm to filter out less informative data instances. To minimize the impact of filtering out data on model training, we integrate the Recall strategy into BS, allowing us to re-select and re-train these instances as the model approaches convergence. Finally, we use a Redundancy Removed Importance Sampling (RRIS) algorithm to eliminate redundant data copies, further speeding up the training process. It is worth noting that while diversity-based sampling has been widely studied in active learning, focusing on selecting both informative and diverse examples [1], [25], [35], to the best of our knowledge, none of these approaches has been applied to accelerating DNN model training through data instance sampling.

III. Design

A. Architecture Overview

IOWA operates at three levels: the *Storage System*, representing where the training data resides (local, remote, or cloud storage); the *I/O Aware Adaptive Sampling*, which is the core component of our framework and will be elaborated upon; and the *DL Training*, representing the deep learning process on the sampled data.

Before applying importance sampling, we begin with uniform sampling for several epochs to gather basic information, such as data loading and computation time, through the Time Profiler module. Based on this information, we adaptively select the appropriate importance criteria for I/O- and computation-intensive tasks through the Decision Maker. We then proceed with importance sampling, which consists of five primary steps, as outlined in Figure 4: ① *Data Selection*: We sort the importance values and select informative samples based on the importance criteria and sampling strategy. ② *Data Fetch*: After determining the training instances for one epoch in advance, we fetch them from the storage system. ③ *Data Preprocessing*: The fetched data is decoded, transformed, and packed into batches. ④ *Data Training*: Each batch of data is fed into the target model in one iteration, and all selected informative samples are trained within one epoch. ⑤ *Information Feedback*: During training, we continue to collect information, such as data loading times and model outputs, and feed this information back into our I/O-aware adaptive sampling layer to refine importance determination in subsequent stages.

B. I/O-Aware Multi-Criteria Importance Metric

Unlike existing sampling approaches, we propose an adaptive importance metric to handle different types of workloads,

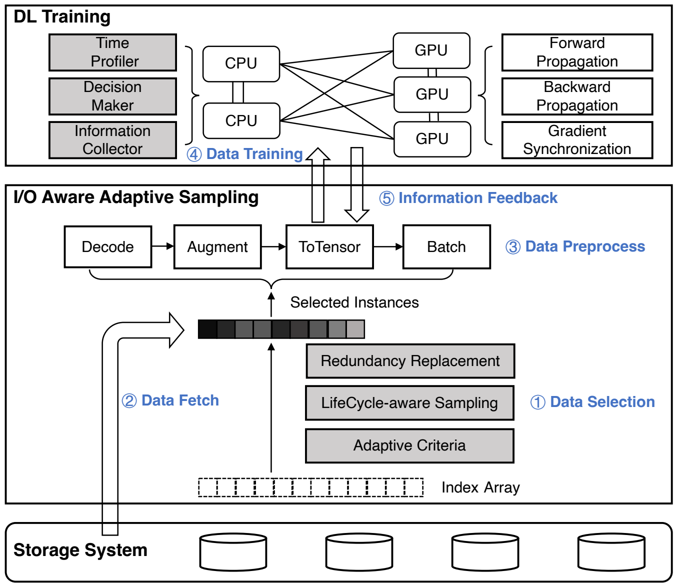


Fig. 4. System overview of IOWA.

which consists of the following three criteria: CrossEntropy Score, Margin Score, and I/O Score:

$$CrossEntropyScore = -P_\theta(y_{target}|x)log_2 P_\theta(y_{target}|x) \tag{3}$$

Where the x represents the input data, y_{target} is the target label for the input, and $P_\theta(y_{target}|x)$ is the prediction probability of the target label from the model with parameters θ. By multiplying the probability by its own log, we obtain the entropy loss for each instance, which reveals the probability distribution of the prediction. This is a commonly used criterion in many other studies [15], [18], [23], [33].

$$Margin\ Score = 1 - (P_\theta(y_{target}|x) - P_\theta(y_{max}|x)) \tag{4}$$

The margin represents the difference between the target label prediction $P_\theta(y_{target}|x)$ and the most confident prediction $P_\theta(y_{max}|x))$, excluding the target label y_{target}. A narrower margin indicates that it is more difficult to distinguish between the target label y_{target} and another label y_{max}, so we treat data instances with narrower margins as more informative. Our $Margin\ Score$ is equal to 1 minus the margin, assigning higher scores to data with narrower margins.

$$I/O\ Score = 1/(\alpha \times Time_{load} + (1-\alpha) \times Size) \tag{5}$$

$Time_{load}$ and $Size$ denote the normalized data loading time and data size for each instance, respectively. α is the weight assigned to $Time_{load}$, with a default value of 0.5. The I/O Score quantifies the I/O cost of training instances. Taking I/O overhead into account, we prioritize data with shorter access times and lower memory usage when the contributions to model convergence are similar.

TABLE II
THE SAMPLE PARTITION RESULTS FOR DIFFERENT CRITERIA

	Category 1	**Category 2**
Entropy Score	**3**, 10, 11, 0, 5, 6	9, 7, 1, 4, 2, 8
Margin Score	11, 2, 7, 6, 1, 9	4, 5, 0, **3**, 10, 8
I/O Score	5, **3**, 4, 6, 8, 11	10, 7, 9, 0, 2, 1

To calculate these criteria, we profile the time costs for data transmission, I/O (data loading and preparation), and computation during the first few epochs.

With the defined criteria, IOWA dynamically selects the appropriate metric for different workloads. For computation-intensive workload, we use only the CrossEntropy Score to evaluate the informativeness of data instances. For I/O-intensive workloads, the importance criterion is determined by majority voting across the three calculated scores: CrossEntropy Score, Margin Score, and I/O Score. First, the score lists for the training instances are partitioned into different categories based on the sampling strategies. Then, majority voting determines which category each instance belongs to. The following example illustrates the majority voting process:

Example 1: Suppose we have 12 instances in total and obtain the following scores from the information feedback stage:

IndexList = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],

EntropyScoreList = [1.05, 3.2, 4.5, 0.12, 4.40, 1.20, 1.30, 2.20, 5.02, 1.98, 0.88, 0.96],

MarginScoreList = [1.33, 1.02, 0.30, 1.77, 1.20, 1.30, 0.90, 0.55, 1.99, 1.09, 1.80, 0.12],

IOScoreList = [12.00, 45.12, 20.01, 3.22, 4.55, 2.99, 5.88, 9.07, 6.04, 10.70, 8.90, 7.66].

First, we sort the score lists in descending order.

Next, using a pre-computed threshold (as described in Section III-C), we classify the samples into two categories: important and unimportant. Samples with scores higher than the threshold are classified as important, while those with lower scores are categorized as unimportant. The classification results are shown in Table II.

Finally, we apply a majority voting strategy to obtain the final results. For example, the data with index 3 receives two votes for the important category and one for the unimportant category, so it is classified as an important instance. In contrast, the data with index 10 receives one vote for the important category and two for the unimportant category, resulting in its classification as unimportant. The remaining samples are categorized in the same manner. As a result, the important category includes $\{3, 6, 11, 5\}$, while the unimportant category consists of $\{0, 7, 1, 2, 4, 8, 10, 9\}$.

C. Learning-Ability-Conscious Sampling Strategies

In this section, we present our proposed I/O-aware adaptive sampling strategies. The general idea is to reduce the I/O and computation costs of less informative data instances to the utmost extent while minimizing the impact on model accuracy. The key steps are as follows:

1) *Model profiling*: Train the model for a few epochs to collect data information, including computation and I/O time.
2) *Importance metric selection*: Analyze the time breakdown and select the appropriate importance metric for the profiled training workload.
3) *Continuous data sampling*: Use the optimized sampling algorithm to determine which data samples to train at each epoch. To reduce memory access overhead, we rely on historical values rather than current ones to evaluate the importance of instances. This approach is reasonable because the importance of data fluctuates only slightly across neighboring epochs.

The process follows these steps:

Step 1 - Regular Training: In the Bucket Sampling algorithm, all training data instances are initially placed in the in-process bucket.

Step 2 - Sampling Out: During training, an empirical importance value threshold is used to gradually move some data instances from the in-process bucket to the waiting bucket. To accommodate the dynamically changing importance values across different epochs, the threshold metric is defined as follows:

$$thr_t = \beta \times thr_{t-1} + (1 - \beta) \times par_t \tag{6}$$

where thr_t denotes the threshold value at epoch t, thr_{t-1} represents the threshold at epoch $t-1$, and par_t is computed by multiplying the value of equal points $score_e$ in importance list by the relax factor γ at epoch t, as listed in Equation 7. At the start of Bucket Sampling, since thr_{t-1} is unavailable, we will only get the par_0 as initial thresholds thr_0.

$$par_t = score_e \times \gamma \tag{7}$$

Step 3 - re-Sampling: The re-sampling stage is introduced to reduce I/O and computation costs at each epoch while minimizing data loss during training. We begin re-sampling data instances from the waiting bucket randomly when model training accuracy stops improving for k consecutive epochs (we use $k = 3$ in our experiments). This indicates that the selected samples are no longer sufficient for model convergence. At the same time, the temporarily removed samples in the waiting bucket gradually become more important, so we randomly re-sample training instances rather than selecting a specific portion.

D. Redundancy Removed Importance Sampling

The above strategies are designed to accelerate the model training by gradually filtering out *unimportant* data instances and keep *important* ones in the training loop. Inspired by research on data redundancy [3], [10], [20], [28], we also remove redundant *important* data instances in the training process to further reduce I/O and computation costs. To quantify the similarity between data instances, we adopt the

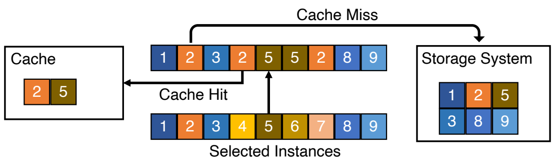


Fig. 5. The redundancy replacement process for importance sampling. Instances {2, 4, 7} and {5, 6} are two groups with higher similar scores. For instance, when we need to read data instances #4 and #7, we can replace them with instance #2 which is cached in memory rather than loading them from remote storage system.

perceptual hashing algorithm (pHash) [32]. pHash is a widely-used algorithm in search engines (e.g., Google) to identify similar or duplicate images based on pixels. We chose it because it can be easily applied to large-scale image processing and is model-independent.

Using similarity scores, we identify redundant important data instances. To reduce computation and I/O costs while maintaining high model accuracy, we propose two approaches to leverage this feature: (1) removing redundant images from the important instances directly for computation-intensive tasks, and (2) replacing instances with another from the similar group for I/O-intensive tasks. To minimize small random accesses to the same image, we cache the frequently selected image in memory and process it with different augmentations during training, as demonstrated in Figure 5.

IV. Evaluation

In this section, we evaluate the effectiveness of our proposed sampling framework with the following configurations:

1) *BS-norecall*: In this configuration, once instances are filtered out during training, they are never re-sampled.
2) *BS*: This sampling algorithm uses a waiting bucket to store less informative instances that fall below an empirical threshold. We explore several variations: a) *BS-random*: When a plateau in accuracy is observed, instances from the waiting bucket are re-sampled randomly. b) *BS-random-RRemove*: An extension of BS-random, where redundant images are removed if a similar instance has already been trained. c) *BS-random-RRepeat*: Another extension of BS-random, where similar samples in the cache are re-trained if a new instance is unavailable but a similar one exists in the cache.

A. Experimental Setup

1) Experimental Platform: We conduct experiments on a server equipped with 8-core Intel Xeon CPUs, 128GB of RAM, and an NVIDIA Tesla V100 GPU with 32GB of GPU memory to perform single-machine tests. Additionally, we evaluate on a platform with 64-core AMD CPUs, 256GB of RAM, and 4 A100-SXM GPUs, each with 40GB of GPU memory, to perform data-parallel distributed training. The storage system used is a 2TB SSD. Our sampling framework is implemented in PyTorch 1.6.0.

2) Workloads and Datasets: For the image classification task, we use four datasets: MNIST, CIFAR10, CIFAR100, and ImageNet, representing small, medium, medium, and large datasets, respectively. For MNIST, we train three DNN models: MLPNet, ConvNet, and AlexNet. For CIFAR10 and CIFAR100, we train three other DNN models: ResNet18, MobileNetV2, and VGG16. Additionally, we train ResNet18 on ImageNet.

3) Baseline Algorithms: To evaluate the effectiveness of our proposed approach, we select the following well-known sampling algorithms as baselines:

0. The Origin: This represents the uniform sampling process, where every data instance is fed into the model once per training epoch.

1. Online [23]: A rank-based importance sampling algorithm that samples data instances according to a loss-rank-based probability.

2. Active Bias [5]: This algorithm re-weights data instances during model training to improve accuracy. The prediction variance is used to measure the uncertainty of each data instance, with those having high variance values sampled for training.

3. Biggest Loser [15]: This algorithm uses the latest and previous loss to measure the importance of samples. All samples are forwarded at least once per epoch, and those with high loss values are selected for backpropagation.

4. Select via Proxy [9]: This algorithm pre-trains a lightweight model first, and then trains the target model using the coreset selected by the pre-trained light model. Unlike our proposed adaptive sampling approach, this algorithm does not incorporate a re-sampling phase during training.

In our experiments, we adopt the same parameter settings as in the aforementioned literature when implementing the baseline algorithms for comparison. All experiments are run three times independently, and the averaged results are reported.

B. Speedup with a Single GPU

In our comparisons, we run different DNNs, including MLPNet, ConvNet, and AlexNet on the MNIST dataset. The results are illustrated in Table III. Additionally, the experimental results of using ResNet18, MobileNetV2, and VGG16 on CIFAR10 and CIFAR100 are shown in Table IV, V, respectively.

From the above results, we can see that model training with Bucket Sampling can achieve up to a 3.2× speedup for the ResNet model on the CIFAR10 dataset compared to the origin, with less than a 1% impact on performance. It is worth noting that while BS without re-sampling achieves better acceleration, it also has a more significant impact on accuracy.

Among the baseline approaches, *Biggest Loser* and *Online*, which focus on reducing backward time or the frequency of loss recomputation, achieve inferior speedup performance compared to our proposed algorithms. For *Active Bias*, it can achieve a slighter model accuracy improvement but with no time speedup. *SelectViaProxy* can gain a 2× speedup but with significant model accuracy loss (up to 3%), as shown in Figure

TABLE III
THE ACCURACY AND SPEEDUP COMPARISON OF DIFFERENT FRAMEWORKS ON MNIST

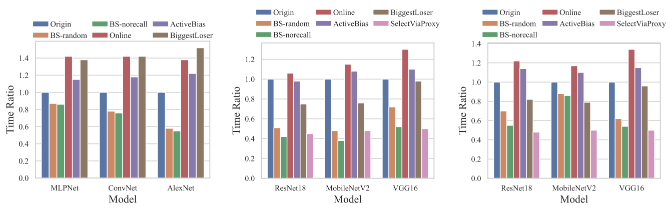
Model	Sample Method	Accuracy	DAcc	Speedup
AlexNet	Origin	0.9922±0.0004		1.00
	BS-norecall	0.992±0.0003	-0.0002	2.51
	BS-random	0.9918±0.0003	-0.0004	2.01
	BS-random-RRemove	0.992±0.0004	-0.0002	2.37
	BS-random-RRepeat	0.9918±0.0011	-0.0004	1.97
ConvNet	Origin	0.9915±0.0007		1.00
	BS-norecall	0.9917±0.0005	0.0002	1.98
	BS-random	0.9912±0.0003	-0.0003	1.83
	BS-random-RRemove	0.9915±0.0003	0	1.81
	BS-random-RRepeat	0.9914±0.0003	-0.0001	1.77
MLPNet	Origin	0.9788±0.0		1.00
	BS-norecall	0.9783±0.0004	-0.0005	1.67
	BS-random	0.9786±0.0011	-0.0002	1.51
	BS-random-RRemove	0.9784±0.0004	-0.0004	1.61
	BS-random-RRepeat	0.9781±0.0	-0.0007	1.52

TABLE IV
THE ACCURACY AND SPEEDUP COMPARISON OF DIFFERENT FRAMEWORKS ON CIFAR10

Model	Sample Method	Accuracy	DAcc	Speedup
MobileNetV2	Origin	0.923±0.002		1.00
	BS-norecall	0.9115±0.0014	-0.0115	2.71
	BS-random	0.9159±0.0021	-0.0071	2.10
	BS-random-RRemove	0.9169±0.0008	-0.0061	2.11
	BS-random-RRepeat	0.9146±0.0034	-0.0084	2.09
ResNet18	Origin	0.9327±0.0014		1.00
	BS-norecall	0.9175±0.0015	-0.0152	3.25
	BS-random	0.9243±0.002	-0.0084	2.43
	BS-random-RRemove	0.9222±0.0013	-0.0105	2.49
	BS-random-RRepeat	0.9224±0.0022	-0.0103	2.51
VGG16	Origin	0.916±0.0023		1.00
	BS-norecall	0.9033±0.0006	-0.0127	2.13
	BS-random	0.9059±0.0029	-0.0101	1.50
	BS-random-RRemove	0.9057±0.0023	-0.0103	1.55
	BS-random-RRepeat	0.9089±0.0034	-0.0071	1.61

TABLE V
THE ACCURACY AND SPEEDUP COMPARISON OF DIFFERENT FRAMEWORKS ON CIFAR100

Model	Sample Method	Accuracy	DAcc	Speedup
MobileNetV2	Origin	0.685±0.0032		1.00
	BS-norecall	0.679±0.001	-0.006	1.15
	BS-random	0.689±0.0012	0.004	1.04
	BS-random-RRemove	0.6845±0.0025	-0.0005	1.04
	BS-random-RRepeat	0.6864±0.0036	0.0014	1.04
ResNet18	Origin	0.762±0.0022		1.00
	BS-norecall	0.7125±0.0036	-0.0495	2.15
	BS-random	0.7518±0.0006	-0.0102	1.27
	BS-random-RRemove	0.754±0.002	-0.008	1.29
	BS-random-RRepeat	0.7539±0.0021	-0.0081	1.29
VGG16	Origin	0.7262±0.0029		1.00
	BS-norecall	0.6586±0.0033	-0.0676	1.59
	BS-random	0.718±0.0029	-0.0082	1.14
	BS-random-RRemove	0.72±0.0019	-0.0062	1.15
	BS-random-RRepeat	0.7194±0.0017	-0.0068	1.15



(a) MNIST (b) CIFAR10 (c) CIFAR100

Fig. 6. The time speedup of different algorithms.

TABLE VI
THE ACCURACY AND SPEEDUP COMPARISON OF DIFFERENT NUMBERS OF GPUS

Model (GPU number)	Methods	Accuracy	Speedup
ResNet18 (with 8 GPUs)	Origin	69.90%	-
	BS-random	69.51%	1.21 ×
	BS-norecall	69.56%	1.23 ×
ResNet18 (with 4 GPUs)	Origin	69.78%	-
	BS-random	69.53%	1.21 ×
	BS-norecall	69.35%	1.23 ×
ResNet18 (with 2 GPUs)	Origin	69.79%	-
	BS-random	69.37%	1.18 ×
	BS-norecall	69.21%	1.22 ×

6. This is because *SelectViaProxy* trains the target model only on the coreset and without re-sampling from the removed data instances.

C. Speedup with Distributed GPUs

We conducted further experiments on distributed GPUs to explore the impact of I/O overhead, including network data transmission, on our algorithms. On each GPU, we start one worker for training. We train ResNet18 on the ImageNet dataset, and the results are summarized in Table VI.

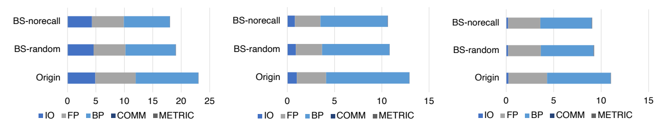
From Table VI, we can observe that our introduced algorithms still achieve up to a 1.2× speedup compared to the original uniform sampling. However, the speedup is less significant when training on ImageNet compared to MNIST, CIFAR10, and CIFAR100. This is because ImageNet presents a higher learning difficulty and lower redundancy, leaving less room for sampling optimization.

D. Time Breakdown

To further explore the contributions of our algorithms during different phases of model training, we plot the time breakdown of training ResNet18 on ImageNet, which includes I/O, forward, backward (gradient communication and backpropagation), metric computation, and metric communication. The results are shown in Figure 7. We can see that BS slightly increases the metric computation and data transmission time needed to synchronize the metric score values, but significantly reduces the I/O, forward, and backward time.

E. Overhead Analysis

For Bucket Sampling, the only overhead is computing the threshold and partitioning the training instances, which costs



(a) With 2 GPUs (b) With 4 GPUs (c) With 8 GPUs

Fig. 7. The time breakdown of training ResNet on ImageNet.

O(1) time. Additionally, the time cost of redundancy removal can be ignored, as it is performed offline. In a distributed training environment, the extra broadcasting time for the importance metric is O(N), where N represents the total data volume.

V. Related Work

A. Learning Strategy

Curriculum Learning (CL) [2] is a strategy inspired by the human learning process, where easier examples are trained first, and harder samples are gradually introduced in the later stages of training. CL labels samples as easy or hard based on prior knowledge, without considering feedback from the learner. Unlike CL, Self-paced learning (SPL) [21] dynamically adjusts the learner's learning space according to the loss value, which may result in overfitting as prior knowledge is not taken into account. To address the shortcomings of CL and SPL, [16] proposes Self-paced Curriculum Learning (SPCL), which combines CL and SPL. SPCL leverages static information before training and dynamic knowledge during training, showing improved performance.

All three of these algorithms focus on improving model accuracy. During the training process, CL or SPL involve more and more samples until all samples have been considered according to the learning curriculum. In contrast, our proposed adaptive sampling algorithm involves fewer and fewer samples to accelerate the training process.

B. Importance Sampling

The optimal sampling distribution is proven to sample proportionally to the gradient norm of the sample. For an unbiased estimation of the stochastic gradient, classic importance sampling techniques weight the items inversely proportional to the probability that they are selected. However, computing the sampling distribution via gradient norms introduces significant computational overhead. To reduce this cost, Biggest Loser [15], Biased Importance Sampling [18], Online [23], AutoAssist [33] use *loss* to calculate the sampling distribution. Among them, Online [23] and Biggest Loser [15] maintain a history of losses for previously seen samples, and sample either proportionally to the loss or based on the loss ranking. The method proposed by Online [23] samples instances with a probability exponential to their last known loss value. Biased Importance Sampling [18] and AutoAssist [33] train a separate model to predict the loss, avoiding the need to forward-compute the loss for every sample. In addition to loss, [19] introduces a novel upper bound to the gradient norm, which provides a better approximation than loss-based methods, though it is more challenging to compute. Recently, Select via Proxy [9] uses a proxy model to select a core set of training data to speed up the training process. Dataset Condensation [34] compresses the dataset into synthetic samples; however, this method currently only works on a small subset of images.

The sampling method in IOWA falls under the category of *historical value rank-based* sampling strategies, which use loss as an approximation of sample importance. However, our scheme differs from other rank-based approaches in terms of the importance criterion and pacing strategy. Our method uses adaptive criteria targeted at different performance bottlenecks, achieving a better trade-off between model accuracy and speedup compared to a single criterion. The comparisons with our proposed approach are also summarized in Table VII.

C. Training Acceleration

To accelerate the training process, Wavelet-like Auto-Encoder (WAE) [6] reduces computational complexity by decomposing images into low-resolution channels and then using these decomposed channels as inputs to the CNN. Budgeted Training [22] proposes a linear learning rate schedule to accelerate training under a resource budget constraint. Data echoing, proposed by [8], accelerates model training through data reuse. The idea is to train with the data on hand multiple times rather than waiting for new data to become available. Besides optimizing training from an algorithmic perspective, there are also relevant studies focused on memory management [7], [14] and the use of new hardware, such as processing-in-memory (PIM) [29], [30], to further enhance training efficiency. Orthogonal to these works, our proposed approach minimizes the number of training data instances in each iteration, which can be combined with these existing strategies to achieve better performance.

VI. Conclusion

In this paper, we propose IOWA, an I/O-aware adaptive sampling framework designed to accelerate the model training process without sacrificing the resulting model accuracy. Recognizing that I/O overhead can be significant in I/O-intensive training processes, we first introduce an I/O-aware sampling metric. Based on this metric, we develop the Bucket Sampling algorithm, which gradually samples out training data instances in each iteration to achieve speedup. To further reduce I/O and computation costs, we also incorporate a redundant sample removal strategy into our proposed framework. To evaluate the effectiveness of our approach, we conduct extensive experiments across various DNN architectures and datasets. The experimental results show that our approach achieves up to a $3\times$ speedup with negligible model accuracy loss.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB4502100, the Major Projects of Zhejiang Province under Grant LD24F020012, the National Science Foundation of China under Grant 62172361, the Program of Zhejiang

TABLE VII
COMPARISON OF EXISTING DATA SELECTION METHODS FOR DEEP LEARNING

Technique	Importance Metric	I/O Reduction	Redundency Removing	Lifecycle-Aware Sampling Rate
CL and variation [2], [13], [16], [21]	loss	✗	✗	✓
Online [23]	loss	✗	✗	✓
Active Bias [5]	variance of prediction probabilities	✗	✗	✗
Biggest Loser [15]	loss	✗	✗	✗
Not all samples [19]	upper bound	✗	✗	✗
AutoAssist [33]	another model	✗	✗	✗
SelectViaProxy [9]	core set	✓	✗	✗
IOWA	adaptive criterion	✓	✓	✓

Province Science and Technology under Grant 2022C01044, and the National Key Research and Development Program of China under Grant 2021ZD0110700.

REFERENCES

[1] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds," *arXiv preprint arXiv:1906.03671*, 2019.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum Learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[3] V. Birodkar, H. Mobahi, and S. Bengio, "Semantic Redundancies in Image-Classification Datasets: The 10% You Don't Need," *arXiv preprint arXiv:1901.11409*, 2019.

[4] O. Canévet, C. Jose, and F. Fleuret, "Importance Sampling Tree for Large-Scale Empirical Expectation," in *International Conference on Machine Learning*, 2016, pp. 1454–1462.

[5] H.-S. Chang, E. Learned-Miller, and A. McCallum, "Active Bias: Training More Accurate Neural Networks by Emphasizing High Variance Samples," in *Advances in Neural Information Processing Systems*, 2017, pp. 1002–1012.

[6] T. Chen, L. Lin, W. Zuo, X. Luo, and L. Zhang, "Learning a Wavelet-Like Auto-Encoder to Accelerate Deep Neural Networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] W. Chen, S. He, Y. Xu, X. Zhang, S. Yang, S. Hu, X.-H. Sun, and G. Chen, "icache: An importance-sampling-informed cache for accelerating i/o-bound dnn model training," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 220–232.

[8] D. Choi, A. Passos, C. J. Shallue, and G. E. Dahl, "Faster Neural Network Training with Data Echoing," *arXiv preprint:1907.05550*, 2019.

[9] C. Coleman, S. Mussmann, B. Mirzasoleiman, P. Bailis, P. Liang, J. Leskovec, and M. Zaharia, "Select Via Proxy: Efficient Data Selection for Training Deep Networks," 2018.

[10] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-Balanced Loss Based on Effective Number of Samples," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9268–9277.

[11] L. Deng, G. Hinton, and B. Kingsbury, "New Types of Deep Neural Network Learning for Speech Recognition and Related Applications: An Overview," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8599–8603.

[12] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-Digit Number Recognition from Street View Imagery Using Deep Convolutional Neural Networks," *arXiv preprint arXiv:1312.6082*, 2013.

[13] G. Hacohen and D. Weinshall, "On The Power of Curriculum Learning in Training Deep Networks," *arXiv preprint arXiv:1904.03626*, 2019.

[14] S. He, P. Chen, S. Chen, Z. Li, S. Yang, W. Chen, and L. Shou, "Home: A holistic gpu memory management framework for deep learning," *IEEE Transactions on Computers*, vol. 72, no. 3, pp. 826–838, 2022.

[15] A. H. Jiang, D. L.-K. Wong, G. Zhou, D. G. Andersen, J. Dean, G. R. Ganger, G. Joshi, M. Kaminksy, M. Kozuch, Z. C. Lipton *et al.*, "Accelerating Deep Learning by Focusing on The Biggest Losers," *arXiv preprint arXiv:1910.00762*, 2019.

[16] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann, "Self-Paced Curriculum Learning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[17] T. B. Johnson and C. Guestrin, "Training Deep Models Faster with Robust, Approximate Importance Sampling," in *Advances in Neural Information Processing Systems*, 2018, pp. 7265–7275.

[18] A. Katharopoulos and F. Fleuret, "Biased Importance Sampling for Deep Neural Network Training," *arXiv preprint arXiv:1706.00043*, 2017.

[19] ——, "Not All Samples Are Created Equal: Deep Learning with Importance Sampling," *arXiv preprint arXiv:1803.00942*, 2018.

[20] V. Kaushal, R. Iyer, S. Kothawade, R. Mahadev, K. Doctor, and G. Ramakrishnan, "Learning from Less Data: A Unified Data Subset Selection and Active Learning Framework for Computer Vision," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1289–1299.

[21] M. P. Kumar, B. Packer, and D. Koller, "Self-Paced Learning for Latent Variable Models," in *Advances in neural information processing systems*, 2010, pp. 1189–1197.

[22] M. Li, E. Yumer, and D. Ramanan, "Budgeted Training: Rethinking Deep Neural Network Training under Resource Constraints," *arXiv preprint:1905.04753*, 2019.

[23] I. Loshchilov and F. Hutter, "Online Batch Selection for Faster Training of Neural Networks," *In International Conference on Learning Representation (ICLR)*, 2016.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet Large Scale Visual Recognition Challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[25] O. Sener and S. Savarese, "Active Learning for Convolutional Neural Networks: A Core-Set Approach," *arXiv preprint:1708.00489*, 2017.

[26] H. Song, S. Kim, M. Kim, and J.-G. Lee, "Ada-Boundary: Accelerating DNN Training via Adaptive Boundary Batch Selection," *Machine Learning*, vol. 109, no. 9, pp. 1837–1853, 2020.

[27] S. U. Stich, A. Raj, and M. Jaggi, "Safe Adaptive Importance Sampling," in *Advances in Neural Information Processing Systems*, 2017, pp. 4381–4391.

[28] K. Vodrahalli, K. Li, and J. Malik, "Are All Training Examples Created Equal? An Empirical Study," *arXiv preprint arXiv:1811.12569*, 2018.

[29] T. Wu, S. He, J. Zhu, W. Chen, S. Yang, P. Chen, Y. Yin, X. Zhang, X.-H. Sun, and G. Chen, "Autohet: An automated heterogeneous reram-based accelerator for dnn inference," in *Proceedings of the 53rd International Conference on Parallel Processing*, 2024, pp. 1052–1061.

[30] S. Yang, W. Chen, X. Zhang, S. He, Y. Yin, and X.-H. Sun, "Auto-prune: Automated dnn pruning and mapping for reram-based accelerator," in *Proceedings of the ACM International Conference on Supercomputing*, 2021, pp. 304–315.

[31] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[32] C. Zauner, "Implementation and Benchmarking of Perceptual Image Hash Functions," 2010.

[33] J. Zhang, H.-F. Yu, and I. S. Dhillon, "Autoassist: A Framework to Accelerate Training of Deep Neural Networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 5998–6008.

[34] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset Condensation with Gradient Matching," *In International Conference on Learning Representation (ICLR)*, 2021.

[35] F. Zhdanov, "Diverse Mini-Batch Active Learning," *arXiv preprint arXiv:1901.05954*, 2019.