NeSSA: Near-Storage Data Selection for Accelerated Machine Learning Training

Neha Prakriya, Yu Yang, Baharan Mirzasoleiman, Cho-Jui Hsieh, Jason Cong
University of California, Los Angeles
nehaprakriya, yuyang, baharan, chohsieh, cong@cs.ucla.edu

Abstract

Large-scale machine learning (ML) models rely on extremely large datasets to learn their exponentially growing number of parameters. While these models achieve unprecedented success, the increase in training time and hardware resources required is unsustainable. Further, we find that as dataset sizes increase, data movement becomes a significant component of overall training time. We propose NeSSA, a novel SmartSSD+GPU training architecture to intelligently select important subsets of large datasets near-storage, such that training on the subset mimics training on the full dataset with a very small loss in accuracy. To the best of our knowledge, this is the first work to propose such a near-storage data selection model for efficient ML training. We have evaluated our method for the CIFAR-10, SVHN, CINIC-10, CIFAR-100, TinyImageNet, and ImageNet-100 datasets. We also test across ResNet-20, ResNet-18, and ResNet-50 models.

CCS Concepts

• **Computing methodologies → Neural networks**; • **Hardware → Emerging architectures**.

Keywords

SmartSSD, Computational Storage, Near-Storage Training, Large-Scale ML, Efficient ML Training

ACM Reference Format:
Neha Prakriya, Yu Yang, Baharan Mirzasoleiman, Cho-Jui Hsieh, Jason Cong, University of California, Los Angeles, nehaprakriya, yuyang, baharan, chohsieh, cong@cs.ucla.edu. 2023. NeSSA: Near-Storage Data Selection for Accelerated Machine Learning Training. In *15th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '23), July 9, 2023, Boston, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3599691.3603404

1 Introduction

Deep neural networks (DNNs) have achieved significant success in vision tasks over the last decade. This success is largely driven by vast improvements in GPU computational power, and the ever-increasing number of model parameters. Such over-parameterized models rely on the availability of enormous annotated datasets to achieve high accuracies [1, 2]. However, this poses a scalability challenge. In Figure 1, we demonstrate the exponential rise in training time per epoch for state-of-the-art image classification models developed in the last decade and trained over the ImageNet-1k dataset [3]. This increased training time also has a direct impact on the training costs. A study by OpenAI in 2018 showed that the training costs in terms of petaflops-s/day is doubling every 3.4 months [4]. Significantly reducing these training costs while still ensuring high accuracy is one of the grand challenges in ML [5].

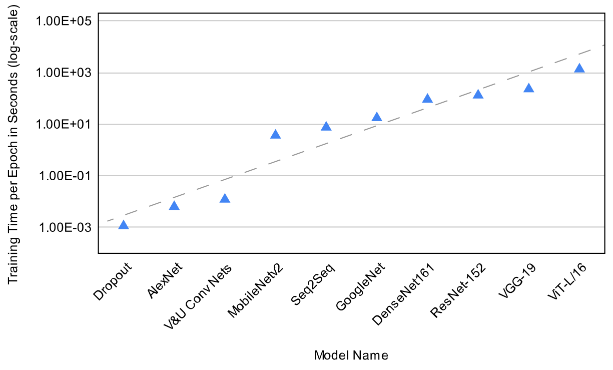

Figure 1: Training time required per epoch for different image classification models using an NVIDIA A100 GPU.

There are two main bottlenecks in training DNNs on large datasets. First is the number of gradient computations required, which drastically increases as the dataset size increases. Second is the data movement and I/O cost incurred when training on large datasets. NeSSA addresses both these challenges using hardware-software co-optimization. For the first bottleneck, it is important to note that each training example has a different utility in aiding model convergence. Therefore, we can afford training only on the most important data examples without compromising significantly on the final model accuracy. Training on this selected subset reduces the total number of gradient computations required for convergence. For example, selecting a subset of size S out of a dataset of size V, reduces the number of gradient computations by $|V|/|S|$. Prior work on subset selection [6–21] has shown promising results in trimming down vision datasets on the CPU to 50-60% of their original size, with an accuracy loss of 5-10%. However, these selection algorithms suffer both from poor accuracy and high selection time, negating any speed-up benefits obtained by filtering the dataset. We discuss details of such work in Section 2.1 and NeSSA's optimizations to improve accuracy and selection time in Section 3. In order to gain any significant speed-up, the chosen selection model should be efficient. We profile the percentage of time spent on data movement to train on the MNIST (0.5KB/image), CIFAR-10 (3KB/image), CIFAR-100 (3KB/image), and ImageNet-100 (130KB/image) datasets in Figure 2 using an NVIDIA V100 GPU. As the dataset size increases (50K to 130K images), the time spent on data movement increases from 5.4% to 40.4% of the overall training time. Traditional data selection methods cannot address this overhead as they require loading the data from the disk to the CPU before computing selections. Prior work like [22, 23] attempt to mitigate this high I/O cost through intelligent caching, but these works still suffer from high data movements between the storage device and the cache. In recent years, there has been a resurgence in academic and industry research on near-storage acceleration [24–34]. Near-storage acceleration proposes to place computation "near data", and has emerged as a promising solution to reducing the overall data movement in the system. To address the second training bottleneck, we propose using near-storage acceleration with the recently released Samsung SmartSSD [35] for data ranking and selection operations. This ensures that data selection is carried out efficiently in the FPGA-based compute units near SSDs, so that only a small fraction of data is transmitted to GPUs for training. NeSSA achieves a critical trade-off between limited system resources and high accuracy. We achieve a large reduction in data movement (3.47x) and increase in training speed (5.37x), with a small accuracy degradation of approx. 1-2%. NeSSA outperforms prior work on subset selection, and is the closest to a model trained on all the data (Section 4.2).

We propose a storage-assisted SmartSSD+GPU system for efficient training of large-scale ML models. Our key contributions are:

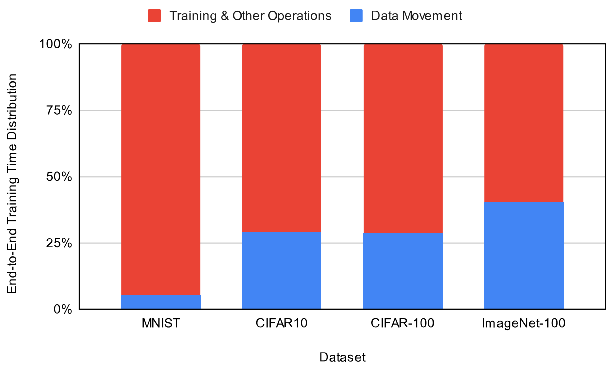

Figure 2: Time distribution of training.

(1) Lighweight FPGA-based near-storage subset selection accelerator on the SmartSSD to achieve an average of 3.47x reduction in data movement costs and end-to-end average training speed-up of 5.37x.
(2) Quantize the selection model for high selection speed.
(3) Improve the quality of subsets selected based on the model's feedback to ensure that we only train on the most important data samples.
(4) Dynamically reduce the subset size based on loss reduction rate during the training process to ensure that we train on the least required data samples.

2 Background

2.1 Subset Selection

Given a training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ of N data-label pairs indexed by $V = \{1, \cdots, N\}$, the goal of training is to learn the set of optimal parameters θ of a model $\Psi(\cdot;\theta)$ which minimize a loss function denoted by $\mathcal{L}(\cdot;\cdot)$. We can formulate the training process as follows:

$$\theta^* = \operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\Psi(x_i;\theta), y_i). \tag{1}$$

The goal of subset selection is to find a subset $S \subseteq V$ that gives a similar gradient to the entire training set V during training, but is much smaller in size. Let $\mathcal{L}_i(\theta) = \mathcal{L}(\Psi(x_i;\theta), y_i)$ be the loss associated with example $i \in V$. We can formulate the problem as follows:

$$S = \operatorname*{argmin}_{S \subseteq V} |S|, \quad \text{s.t.} \tag{2}$$

$$\max_{\theta\in\Theta} || \sum_{i\in V} \nabla\mathcal{L}_i(\theta) - \sum_{j\in S} \nabla\mathcal{L}_j(\theta) || \leq \epsilon,$$

where $\epsilon \geq 0$.

There are two main categories of work in subset selection. The first category uses trained models to infer sample importance [6–16]. That is, a sample's utility in improving a model's convergence is computed using Eq. 2 after the training process. These selection models are expensive to use as they require a full round of training (with gradient computations), negating the speed-up benefits. The second category of work uses training dynamics like loss values, gradients, and model predictions from previous epochs to infer sample importance for future epochs [17–20]. While this category is cheaper to implement than the first, choosing subsets based on limited information results in large accuracy degradation. NeSSA uses the subset selection formulation in [20] as the core component and adapts it to the SmartSSD (described in Section 3.1) to reduce data movement, selection overheads, and adds several optimizations (described in Section 3.2) to achieve high accuracy.

2.2 Near-Storage Acceleration

In the big data era, data analytics is a major workload in warehouse-scale computers with data movement occupying an average of 80% of total time [36]. Storage read/write bandwidths have improved to 3 GBps, while commercially available Ethernet can provide about 12.5 GBps. Usually 32 drives are plugged into this network, reducing the achievable data transfer throughput. This difference is further exacerbated in case of PCIe, driving the need of near-storage acceleration. Near-storage acceleration also provides the additional benefit of freeing up system resources for other tasks. [37] shows that using data-centric computing for query processing frees up 70% of CPU cycles and reduces DRAM utilization by 60%.

Samsung's SmartSSD device is the first computational storage device fabricated in the U.2 format [38]. It has a Xilinx (AMD) Kintex KU15P FPGA with 4GB DRAM connected to a 3.84TB NAND flash over a PCIe-based peer-to-peer connection. There have been several efforts using the SmartSSD for applications like query processing [24, 25], large-scale data sorting [26–28], and near-storage ML inference [29, 30]. In this paper, we advocate for the use of SmartSSD for data filtering and selection tasks for efficient ML training.

Compared with CPU-based selection models used in prior work (Section 2.1), SmartSSD-based acceleration provides customization, fast selection, and low data movement. We discuss the speed-up obtained in Section 4.3. Compared with ASIC-based data filtering, NeSSA provides a low-cost solution which can be reconfigured to target multiple ML models and datasets. NeSSA uses the low-power FPGA on-board the SmartSSD (approx. 7.5W) for data filtering. Such energy efficiency cannot be achieved through GPU-based acceleration (Eg. Nvidia K1200 GPU: 45W, Nvidia A100 GPU: 250W). Given that we select a subset of size S from a dataset of size V,

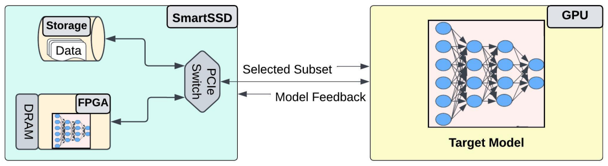

Figure 3: System setup

NeSSA reduces the data movement in the system by a factor of $|V|/|S|$ compared with an FPGA without on-board SSD. Findings by prior work also support our idea to use SmartSSD for data filtering. [33] identifies two characteristics which make a workload suitable for FPGA-based near-storage acceleration. First, the workload should display a high relative data ratio. That is, more data should be read/written to/from the storage than over the drive-host interconnect. If a subset of size S is selected from a dataset of size V where $V >> S$, then the data ratio of $|V|/|S|$ is high. Second, the workload should display low operational intensity. Here, operational intensity is the number of cycles spent on processing one input. If the operational intensity is high, then the accelerator will not saturate the available high drive bandwidth. Using a selection model based on training dynamics allows us to satisfy this condition.

3 System Design and Optimizations

We describe our system setup in Figure 3. The steps involved in our proposed training paradigm are as follows:

(1) Read the training data from the SSD to the FPGA on-board the SmartSSD using the peer-to-peer transfer feature.
(2) Run the selection model described in Section 3.1 and transfer the selected subset to the target model on the GPU.
(3) Train the target model on the subset.
(4) Use the quantized weights and the target model's loss as feedback to improve the selection model on the FPGA, and decide the number of samples to include in future subsets (discussed in Section 3.2). This feedback loop ensures that we maintain a high accuracy and train on the least required samples.
(5) Repeat for all epochs.

3.1 Selection Model to Reduce Training Costs

As directly solving (2) is NP-hard, [20] finds an upper-bound for the gradient estimation error as follows:

$$\min_{S \subseteq V} \| \sum_{i \in V} \nabla \mathcal{L}_i(\theta) - \sum_{j \in S} \nabla \mathcal{L}_j(\theta) \| \leq \sum_{i \in V} \min_{j \in S} \| \nabla \mathcal{L}_i(\theta) - \nabla \mathcal{L}_j(\theta) \|. \quad (3)$$

The RHS of Eq. 3 is the k-medoid problem as formulated by [39] and the set S which minimizes the estimation error is given by the set of medoids. Intuitively, these medoids minimize the maximum pairwise distances between data points and themselves, making them the most representative data samples. For a specific value of k, the set of k-medoids S_t^* at iteration t can be found as:

$$S_t^* \in \arg\min_{\substack{S \subseteq V \\ |S| \leq k}} \sum_{i \in V} \min_{j \in S} \| \nabla \mathcal{L}_i(\theta_t) - \nabla \mathcal{L}_j(\theta_t) \|_2. \quad (4)$$

The minimization problem (4) can be turned into maximizing a submodular facility location objective as described in [20] and is the subset selection model we use:

$$S_t^* \in \arg\min_{S \subseteq V} |S|, \quad s.t. \quad (5)$$
$$F(S) = \sum_{i \in V} \max_{j \in S} (c_0 - \| \nabla \mathcal{L}_i(\theta_t) - \nabla \mathcal{L}_j(\theta_t) \|_2),$$

where c_0 is a constant satisfying $c_0 \geq \| \nabla \mathcal{L}_i(\theta_t) - \nabla \mathcal{L}_j(\theta_t) \|_2$, for all $i, j \in V$. The computational complexity of running this selection model is $O(N)$ using stochastic methods [40], and can be further improved using lazy evaluation [41] and distributed implementations [42].

3.2 Optimizations to Ensure High Accuracy

As discussed in Section 2, selection methods depending only on training dynamics suffer from poor accuracy and increased data movement. While adapting the data selection to the SmartSSD reduces data movement, we propose the following optimizations to improve the understanding of the dataset and accuracy:

3.2.1 Feedback of Quantized Weights : The selection model on the FPGA requires predictions (forward pass) before computing the set of medoids (Section 3.1) for which we use a quantized version of the target model. After training the target model on the subset, we quantize its weights and transfer back to the FPGA forming a feedback loop. This ensures that the selection model is continuously updated to best infer the relative importance of training examples over time, and only select the most informative samples. This optimization allows us to achieve higher accuracy than prior work which do not use such a feedback loop (Section 4.2).

3.2.2 Selecting from Samples That are Not Learned - Subset Biasing : As more data samples are learned, an efficient selection model should focus more on the harder examples that are difficult for the target model to learn and produce large gradients. Samples producing small gradients are already learned and can be ignored. We record losses of the current training examples from the most recent five epochs, mark the samples with small values, and drop the marked samples from the training set every twenty epochs. We found that dropping samples every twenty epochs was a conservative trade-off between training on the smallest subset, and ensuring that we still give the model sufficient time to learn all the data points. Then, we select subsets based on Eq. 5 only from the remaining elements. This ensures that we train on the least required training examples. We discuss the impact of this optimization in Section 4.2.

3.2.3 Selecting Subsets from Smaller Data Partitions - Dataset Partitioning : The subset selection algorithm discussed in Section 3.1 computes pairwise similarities between all examples from the same class label at the beginning of each epoch. When the training set becomes larger, the cost of computing these pairwise similarities scales up quadratically. To efficiently select the subset on an FPGA with limited on-chip memory (4.32MB), we randomly partition the training set into several chunks, and select a smaller subset from each chunk. This way, we do not need to fit the gradients from an entire class onto the on-chip memory, but only the gradients from a single chunk. For example, for a mini-batch size of m and subset size k to be selected from N data points, we partition the dataset into k/m random chunks and select m examples from each chunk to get a total k number of examples in the subset. We discuss the impact of this optimization in Section 4.2.

4 Evaluation

In this Section, we first describe our experimental setup and then the performance results obtained. We evaluate NeSSA to address three main topics: (1) Accuracy comparison with models trained on the full dataset and prior work on subset selection (Section 4.2), (2) End-to-end training speed-up (Section 4.3), (3) Quantitative benefits of using our storage-assisted SmartSSD+GPU training setup (Section 4.4).

4.1 Experimental Setup

The datasets and models used in our experiments are listed in Table 1. We train the models for 200 epochs with batch size of 128, initial learning rate of 0.1 divided by 5 at the 60th, 120th, and 160th epochs, weight decay of $5e-4$, and Nesterov momentum of 0.9.

Table 1: Dataset overview

Dataset	Classes	Train	Network
CIFAR-10 [43]	10	50k	ResNet-20
SVHN [44]	10	73k	ResNet-18
CINIC-10 [45]	10	90k	ResNet-18
CIFAR-100 [43]	100	50k	ResNet-18
TinyImageNet [46]	200	100k	ResNet-18
ImageNet-100 [3]	100	130k	ResNet-50

Table 2: Accuracy and data ratio trained by NeSSA compared with the same model trained on the full dataset.

Dataset	All Data (%)	NeSSA (%)	Subset (%)
CIFAR-10	92.02	90.17	28
SVHN	95.81	95.18	15
CINIC-10	81.49	80.26	30
CIFAR-100	70.98	69.23	38
TinyImageNet	63.40	63.66	34
ImageNet-100	84.60	83.76	28

4.2 Performance Evaluation

We compare the performance of NeSSA for all tested datasets with a model trained on the full dataset in Table 2. NeSSA achieves comparable accuracy by training on just 15-38% of the dataset. The rest of the dataset is automatically pruned away during training by our optimizations described in Section 3.2. To quantify the improvement in accuracy obtained using each of these optimizations, we analyze the performance for the CIFAR-10 dataset in Table 3. Here, "Vanilla" refers to NeSSA without subset biasing (Section 3.2.2) and dataset partitioning (Section 3.2.3). "SB" and "PA" refer to the implementations with subset biasing and partitioning respectively. "Goal" refers to a model trained with the full dataset. We also compare the accuracy obtained on different subset sizes by NeSSA and prior CPU-based state-of-the-art subset selection work [20][17] in Table 3. Overall, NeSSA outperforms prior work and is the closest to the goal accuracy. Compared with [17], NeSSA uses a submodular-optimization based selection methods which attempts to minimize the total dissimilarity between data points instead of the total squared error. Compared with [20], NeSSA includes several optimizations as described in Section 3.2 which significantly improve the quality of the selected subsets. Specifically, adding feedback between the target model and the selection model allows the selection of only the most important training examples. We also achieve higher performance with the same subset size due to the subset biasing and partitioning optimizations.

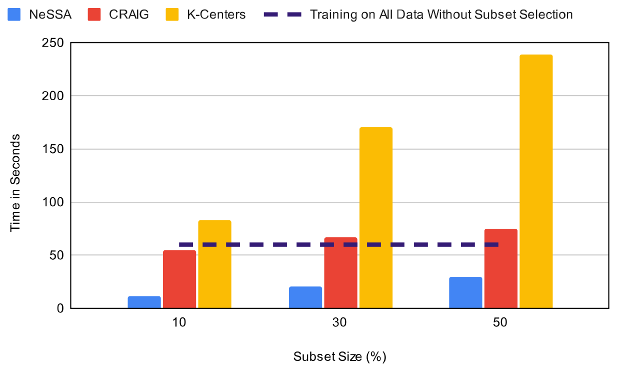

Figure 4: Training time averaged across epochs for NeSSA, prior work, and a model trained on the full dataset.

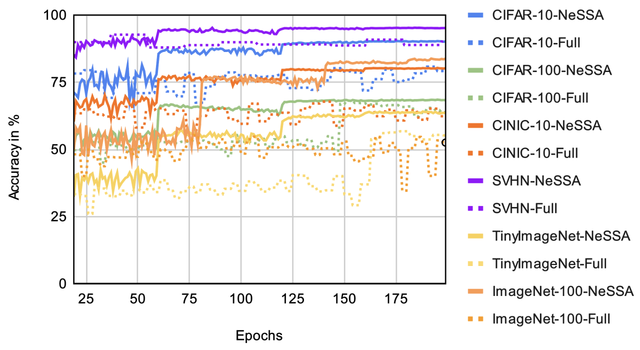

Figure 5: Accuracy of NeSSA and a model trained on the full dataset over the training process.

4.3 End-to-End Training Speed-Up

In Figure 4, we compare the average time taken to train one epoch for the CIFAR-10 dataset with the ResNet-20 model using NeSSA, prior CPU-based subset selection techniques like [17, 20], and a model trained on the full dataset. Overall across datasets, NeSSA gains an end-to-end training speed-up of 5.37x compared with training on the full dataset, 4.3x compared with [20], and 8.1x compared with [17]. We also find that NeSSA converges close to the optimal solution faster than a model trained on the entire dataset. This is because NeSSA ensures that the model constantly trains on examples which will accelerate it's convergence the most, rather than selecting random batches of data. We demonstrate the faster convergence on all datasets in Table 1 in Figure 5. NeSSA reaches closer to convergence within the first 30 epochs of training for all datasets compared with a model trained on the full dataset (solid series is higher than dotted series of the same color).

Table 3: Accuracy comparison of NeSSA with different optimizations and prior work. The column names marked in bold is NeSSA and the values marked in bold is the setting with best performance (closest to "Goal").

Subset (%)	**Vanilla** (%)	**SB** (%)	**PA** (%)	**SB+PA** (%)	CRAIG[20]	K-Centers [17]	Goal (%)
10	82.76	87.61	83.56	**87.75**	87.07	65.72	92.44
30	89.51	90.42	**90.68**	90.49	89.12	88.49	92.44
50	90.59	91.89	91.81	**91.92**	90.32	90.14	92.44

Table 4: Resource utilization

Resource	Available	Util (%)
LUT	432k	67.53
FF	919k	23.14
BRAM	738	50.30
DSP	1962	42.67

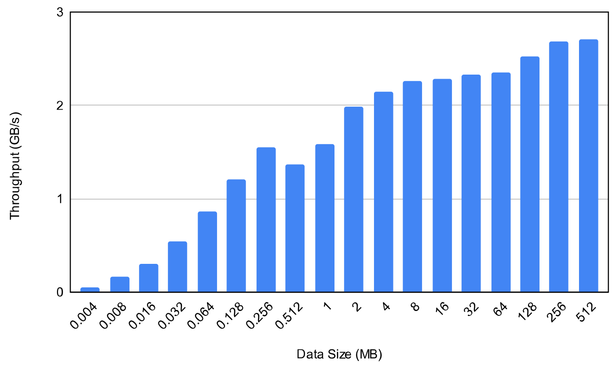

Figure 6: Data transfer throughput between FGPA and on-board SSD (average of read/writes)

4.4 Benefits of using Storage-Assisted Training

SSD to FPGA transfers on-board the SmartSSD can theoretically achieve up to 3GBps data transfer rates. In a conventional setting where FPGA does not have direct access to the on-board SSD and uses CPU memory as temporary storage, the effective bandwidth is reduced to 1.4 GBps. Therefore, data transfer rates are on average 2.14x faster using the SmartSSD. Along with this speed-up, reducing the dataset size by selecting subsets also reduces the volume of data transferred over the interconnect. Overall, our method achieves an average data movement reduction of 3.47x across datasets. We profile the effective data transfer rates between the FPGA and on-board SSD (average of reading and writing from/to on-board SSD) in Figure 6. For the CIFAR-10 dataset where each image size is 0.003MB, using a batch size of 128 images, data transfer rates achieved are 1.46 GBps. As the dataset and image sizes increase, the available bandwidth is better saturated. In case of the ImageNet-100 dataset where each image is 0.126 MB, using a batch size of 128 images achieves a data transfer throughput of 2.28 GBps. Therefore, as the dataset size increases, using storage-assisted ML training becomes more effective and necessary. Particularly using FPGA-based storage-assisted training provides a low-cost solution (Table 4) which allows low selection and data transfer time (Figure 4) compared with CPU-based selection algorithms and training on the full dataset.

5 Conclusion

We present a SmartSSD+GPU training setup to intelligently select subsets of large datasets near-storage and train only on the selected data while maintaining high accuracy. NeSSA achieves 3.47x reduction in data movement and end-to-end training speed-up average of 5.37x with a negligible loss in accuracy compared with a model trained on the full dataset. We are currently working on extending this work for larger datasets and models scaling over multiple SmartSSDs and GPUs.

Acknowledgments

This research was supported by PRISM Award CC OTH 00541340 2022 TR and CRISP Award GI18518.156870. We would like to thank Samsung Electronics Co. for providing access to the SmartSSD devices. This work was supported in part by the Semiconductor Research Corporation (SRC) and DARPA. BM was supported by the National Science Foundation CAREER Award 2146492. CJH was supported by the National Science Foundation Award IIS-2008173, IIS-2048280 and Sony Research Award.

References

[1] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *CoRR*, abs/1707.02968, 2017. URL http://arxiv.org/abs/1707.02968.

[2] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. Beyond Neural Scaling Laws: Beating Power Law Scaling Via Data Pruning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=UmvSlP-PyV.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[4] Dario Amodei, Danny Hernandez, Girish Sastry, Jack Clark, Greg Brockman, and Ilya Sutskever. AI and Compute. 2018.

[5] Hilal Asi and John C. Duchi. The Importance of Better Models in Stochastic Optimization. *Proceedings of the National Academy of Sciences*, 116(46):22924–22930, oct 2019. doi: 10.1073/pnas.1908018116. URL https://doi.org/10.1073%2Fpnas.1908018116.

[6] Agata Lapedriza, Hamed Pirsiavash, Zoya Bylinskii, and Antonio Torralba. Are All Training Examples Equally Valuable? *arXiv preprint arXiv:1311.6510*, 2013.

[7] Linnan Wang, Yi Yang, Renqiang Min, and Srimat Chakradhar. Accelerating Deep Neural Network Training with Inconsistent Stochastic Gradient Descent. *Neural Networks*, 93:219–229, 2017.

[8] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset Distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[9] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An Empirical Study of Example Forgetting during Deep Neural Network Learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

[10] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via Proxy: Efficient Data Selection for Deep Learning. *arXiv preprint arXiv:1906.11829*, 2019.

[11] Jiong Zhang, Hsiang-Fu Yu, and Inderjit S Dhillon. AutoAssist: A Framework to Accelerate Training of Deep Neural Networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/9bd5ee6fe55aaeb673025dbcb8f939c1-Paper.pdf.

[12] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryghZJBKPS.

[13] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset Condensation with Gradient Matching. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=mSAKhLYLSsl.

[14] Srikumar Ramalingam, Daniel Glasner, Kaushal Patel, Raviteja Vemulapalli, Sadeep Jayasumana, and Sanjiv Kumar. Less is More: Selecting Informative and Diverse Subsets With Balancing Constraints. *arXiv preprint arXiv:2104.12835*, 2021.

[15] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep Learning on a Data Diet: Finding Important Examples Early in Training. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20596–20607. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/ac56f8fe9eea3e4a365f29f0f1957c55-Paper.pdf.

[16] Prioritized Training on Points That Are Learnable, Worth Learning, And Not Yet Learnt, author=Mindermann, Sören and Brauner, Jan M and Razzak, Muhammed T and Sharma, Mrinank and Kirsch, Andreas and Xu, Winnie and Höltgen, Benedikt and Gomez, Aidan N and Morisot, Adrien and Farquhar, Sebastian and others, booktitle=International Conference on Machine Learning, pages=15630–15649, year=2022, organization=PMLR.

[17] Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. *arXiv preprint arXiv:1708.00489*, 2017.

[18] Angelos Katharopoulos and François Fleuret. Not All Samples Are Created Equal: Deep Learning With Importance Sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[19] Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating Deep Learning By Focusing On The Biggest Losers. *arXiv preprint arXiv:1910.00762*, 2019.

[20] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for Data-Efficient Training of Machine Learning Models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

[21] Yu Yang, Hao Kang, and Baharan Mirzasoleiman. Towards sustainable learning: Coresets for data-efficient deep learning, 2023.

[22] Redwan Ibne Seraj Khan, Ahmad Hossein Yazdani, Yuqi Fu, Arnab K. Paul, Bo Ji, Xun Jian, Yue Cheng, and Ali R. Butt. SHADE: Enable fundamental cacheability for distributed deep learning training. In *21st USENIX Conference on File and Storage Technologies (FAST 23)*, pages 135–152, Santa Clara, CA, February 2023. USENIX Association. ISBN 978-1-939133-32-8. URL https://www.usenix.org/conference/fast23/presentation/khan.

[23] Weijian Chen, Shuibing He, Yaowen Xu, Xuechen Zhang, Siling Yang, Shuang Hu, Xian-He Sun, and Gang 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)*, pages 220–232, 2023. doi: 10.1109/HPCA56546.2023.10070964.

[24] Jaeyoung Do, Yang-Suk Kee, Jignesh M. Patel, Chanik Park, Kwanghyun Park, and David J. DeWitt. Query Processing on Smart SSDs: Opportunities and Challenges. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, page 1221–1230, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320375. doi: 10.1145/2463676.2465295. URL https://doi.org/10.1145/2463676.2465295.

[25] Mohammadreza Soltaniyeh, Veronica Lagrange Moutinho Dos Reis, Matthew Bryson, Richard Martin, and Santosh Nagarakatte. Near-Storage Acceleration of Database Query Processing with SmartSSDs. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 265–265, 2021. doi: 10.1109/FCCM51124.2021.00052.

[26] Weikang Qiao, Jihun Oh, Licheng Guo, Mau-Chung Frank Chang, and Jason Cong. FANS: FPGA-Accelerated Near-Storage Sorting. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 106–114, 2021. doi: 10.1109/FCCM51124.2021.00020.

[27] Sahand Salamat, Armin Haj Aboutalebi, Behnam Khaleghi, Joo Hwan Lee, Yang Seok Ki, and Tajana Rosing. NASCENT: Near-Storage Acceleration of Database Sort on SmartSSD. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '21, page 262–272, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382182. doi: 10.1145/3431920.3439298. URL https://doi.org/10.1145/3431920.3439298.

[28] Sahand Salamat, Hui Zhang, Yang Seok Ki, and Tajana Rosing. NASCENT2: Generic Near-Storage Sort Accelerator for Data Analytics on SmartSSD. *ACM Trans. Reconfigurable Technol. Syst.*, 15 (2), jan 2022. ISSN 1936-7406. doi: 10.1145/3472769. URL https://doi.org/10.1145/3472769.

[29] Mohammadreza Soltaniyeh, Veronica Lagrange Moutinho Dos Reis, Matt Bryson, Xuebin Yao, Richard P. Martin, and Santosh Nagarakatte. Near-Storage Processing for Solid State Drive Based Recommendation Inference with SmartSSDs. In *Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering*, ICPE '22, page

177–186, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391436. doi: 10.1145/3489525.3511672. URL https://doi.org/10.1145/3489525.3511672.

[30] Erfan Bank Tavakoli, Amir Beygi, and Xuebin Yao. RPkNN: An OpenCL-Based FPGA Implementation of the Dimensionality-Reduced kNN Algorithm Using Random Projection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(4):549–552, 2022. doi: 10.1109/TVLSI.2022.3147743.

[31] Zhenyuan Ruan, Tong He, and Jason Cong. INSIDER: Designing In-Storage computing system for emerging High-Performance drive. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 379–394, Renton, WA, July 2019. USENIX Association. ISBN 978-1-939133-03-8. URL https://www.usenix.org/conference/atc19/presentation/ruan.

[32] Ian F. Adams, Neha Agrawal, and Michael P. Mesnier. Enabling near-data processing in distributed object storage systems. In *Proceedings of the 13th ACM Workshop on Hot Topics in Storage and File Systems*, HotStorage '21, page 28–34, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385503. doi: 10.1145/3465332.3470881. URL https://doi.org/10.1145/3465332.3470881.

[33] Zhenyuan Ruan, Tong He, and Jason Cong. Analyzing and Modeling In-Storage Computing Workloads On EISC — An FPGA-Based System-Level Emulation Platform. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019. doi: 10.1109/ICCAD45719.2019.8942135.

[34] Weikang Qiao, Jieqiong Du, Zhenman Fang, Michael Lo, Mau-Chung Frank Chang, and Jason Cong. High-throughput lossless compression on tightly coupled cpu-fpga platforms. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 37–44, 2018. doi: 10.1109/FCCM.2018.00015.

[35] Joo Hwan Lee, Hui Zhang, Veronica Lagrange, Praveen Krishnamoorthy, Xiaodong Zhao, and Yang Seok Ki. SmartSSD: FPGA Accelerated Near-Storage Data Analytics on SSD. *IEEE Computer Architecture Letters*, 19(2):110–113, 2020. doi: 10.1109/LCA.2020.3009347.

[36] Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks. Profiling a Warehouse-Scale Computer. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pages 158–169, 2015. doi: 10.1145/2749469.2750392.

[37] Shuotao Xu, Thomas Bourgeat, Tianhao Huang, Hojun Kim, Sungjin Lee, and Arvind Arvind. AQUOMAN: An Analytic-Query Offloading Machine. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 386–399, 2020. doi: 10.1109/MICRO50266.2020.00041.

[38] SmartSSD Computational Storage Drive: Installation and User Guide. 2021. URL https://www.xilinx.com/content/dam/xilinx/support/documents/boards_and_kits/accelerator-cards/1_3/ug1382-smartssd-csd.pdf.

[39] Leonard Kaufman and Peter Rousseeuw. Clustering by Means of Medoids. 1987.

[40] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier Than Lazy Greedy. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[41] Michel Minoux. Accelerated Greedy Algorithms for Maximizing Submodular Set Functions. In *Optimization techniques*, pages 234–243. Springer, 1978.

[42] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed Submodular Maximization: Identifying Representative Elements in Massive Data. In *Advances in Neural Information Processing Systems*, pages 2049–2057, 2013.

[43] A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[44] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[45] Luke N. Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. CINIC-10 is not ImageNet or CIFAR-10, 2018. URL https://arxiv.org/abs/1810.03505.

[46] Ya Le and Xuan S. Yang. Tiny ImageNet Visual Recognition Challenge. 2015.