Human Interaction and Emerging Technologies (IHIET-AI 2025), Vol. 161, 2025, 72–83
https://doi.org/10.54941/ahfe1005898

Evaluating Training Acceleration Through Selective Workload Skipping: Methods and Benchmarks

Kareem Ibrahim[1], Milos Nikolic[1], Nicholas Giamblanco[1], Ali Hadi Zadeh[1], Enrique Torres Sanchez[1], and Andreas Moshovos[1,2]

[1]University of Toronto, Toronto, Ontario, Canada
[2]Vector Institute for Artificial Intelligence, Toronto, Ontario, Canada

ABSTRACT

Work-skipping methods accelerate neural network training by selectively skipping work that is deemed not to contribute significantly to learning. The goal of such methods is to reduce training time while incurring little or negligible reduction in output accuracy. We identify a "blind-spot" in current best-practice methodologies used to evaluate the effectiveness of work-skipping methods. Current methodologies fail to establish objective ways of determining whether a time reduction vs. accuracy drop trade-off is indeed beneficial. We propose a set of guidelines for evaluating the effectiveness of workload skipping techniques. Our guidelines emphasize the importance of using wall clock time, comparing with random skipping baselines, incorporating early stopping or time-to-accuracy measures, and utilizing Pareto curves. By providing a structured framework, we aim to assist practitioners in accurately determining the true speed advantages of training acceleration algorithms that involve workload skipping. To illustrate the appropriateness of our guidelines we study two work-skipping methods: GSkip, which skips complete layer's gradient computations and weight updates based on their relative changes, and DeadHorse, which selects data samples for backpropagation according to output confidence. We demonstrate how our methodology can establish when these methods are indeed beneficial. We find that on many occasions, random skipping, early termination, or hyperparameter tuning may be as effective if not more.

Keywords: Neural network training acceleration, Workload-skipping, Gradient skipping, Sample skipping, Performance evaluation

INTRODUCTION

Training deep neural networks is computationally and memory-intensive, often requiring extensive data processing. As these models evolve, their increasing demands create bottlenecks in both research and applications, motivating efforts to reduce training costs in computation, memory, and ultimately time (Thompson et al., 2020).

Training proceeds incrementally across layers, batches, epochs, and individual samples. This naturally prompts the question of whether every single step is necessary for achieving high performance—leading to the

emergence of **work-skipping algorithms** (Shen et al., 2023). These methods selectively omit parts of the workload considered non-essential, thereby reducing computational overhead without substantially compromising model accuracy (Coquelin et al., 2022).

The **main contribution of this work** is introducing an **evaluation methodology** to assess the effectiveness of such training acceleration methods. Our methodology addresses pitfalls in prior evaluations, ensuring that claimed efficiency improvements are thoroughly vetted under various conditions. This structured framework sets reliable benchmarks, crucial for evaluating the practical value of workload-skipping algorithms in real-world applications.

We illustrate the usefulness of our methodology by presenting two new work-skipping methods, **GSkip** and **DeadHorse**. GSkip selectively skips gradient updates for layers deemed to have minimal impact based on gradient similarity and loss stability, while DeadHorse omits updates from samples that offer limited informational value. Although initial experiments indicated promising speedups with minor accuracy trade-offs, we found that existing evaluations relied on subjective judgments of what constitutes an "acceptable" accuracy drop. By introducing control experiments such as **random skipping** and **early stopping**, we discovered that simpler strategies can often match or even surpass the performance of these advanced skipping methods. These findings emphasize the need for robust baselines and objective metrics when assessing workload-skipping algorithms.

BACKGROUND

Neural Network Training and Costs

Neural network training is a trial-and-error process that optimizes a loss function to improve model performance. It involves two core phases: the **forward pass**, which multiplies inputs by learned weights and stores activations, and the **backward pass**, which computes gradients and updates the weights. Because it repeats these cycles across many samples and epochs, training is significantly more resource-intensive than inference, often requiring around 70% more computations and substantial off-chip memory transfers (Goli & Aamodt, 2020).

During the forward pass, the network produces outputs that must be cached for gradient calculations in the subsequent backward pass. The backward pass then compares these outputs to ground truth labels, computing gradients that guide weight adjustments. Each incremental step is meant to refine the network's parameters, but not all updates contribute equally to the final model accuracy.

Training Acceleration via Work-Skipping

To reduce the heavy computational load of training, **work-skipping methods** selectively bypass parts of the training process that are less impactful. By identifying layers or samples that contribute minimally to learning, these approaches can eliminate redundant operations and lighten memory usage.

Such selective skipping leverages the iterative nature of training to focus resources on critical updates, thereby aiming to accelerate convergence without incurring unacceptable accuracy losses. However, evaluating the true benefit of these approaches demands comprehensive comparisons against simpler baselines and control strategies—a central focus of this paper.

RELATED WORK

Work-Skipping Training Acceleration Methods

MeProp retains only the top-k gradients during the backward pass, achieving up to 95% gradient sparsity and significant speedups (Sun et al., 2019; Wei et al., 2017). However, MeProp exhibits training divergence on more complex tasks.

ResProp accelerates CNN training by reusing sparse gradients, yielding a 2× speedup at the cost of about 1% accuracy reduction (Goli & Aamodt, 2020).

Backprop with Approximate Activations reduces memory footprint by storing approximate activations, cutting memory usage by up to 8× while keeping accuracy loss below 0.5% (Chakrabarti & Moseley, 2019).

Weight Update Skipping (WUS) and its variant WUS with Learning Rate (WUSLR) omit weight updates during periods of minimal accuracy improvement, focusing only on bias updates (Safayenikoo & Akturk, 2020). They reduce training time by over 50% on certain benchmarks, with accuracy drops below 1% in many cases.

Collectively, these methods demonstrate the potential of selective skipping to accelerate training. Yet each approach involves some trade-off, typically sacrificing a fraction of accuracy for computational gains. Evaluations often rely on subjective judgment to deem these trade-offs "acceptable." Our proposed methodology addresses this limitation by isolating whether a method's intelligence truly outperforms simpler strategies—especially given that a large portion of training time often yields only marginal gains in accuracy.

Evaluation Methodologies

The **AlgoPerf benchmark** provides a comprehensive framework for comparing training algorithms across different neural network architectures (Dahl et al., 2023). It covers essential challenges: determining when training should stop, measuring exact training time, and handling variations in workloads. It also addresses hyperparameter tuning, offering a fair platform for algorithmic comparisons. Our work builds upon and complements these excellent guidelines by proposing specific and additional control experiments that we believe should be conducted for any work-skipping training acceleration proposal.

LAYER-WISE GRADIENT SKIPPING (GSKIP)

Our first work-skipping method, GSkip, is based on the principle that not all gradients contribute equally to a model's learning, particularly in

earlier layers or in later stages of training. By selectively disabling gradient computations for layers expected to have minimal impact, GSkip aims to reduce training time and memory usage without significantly compromising accuracy. To illustrate why rigorous control experiments are critical, GSkip initially appeared promising but, as we later show, must be evaluated against strong baselines.

We analyzed gradient distributions across various networks (ResNet, DenseNet, MobileNetV2, VGG) and datasets (CIFAR10, CIFAR100, ImageNet) (He et al., 2016; Krizhevsky et al., 2009; Sandler et al., 2018; Zhu & Newsam, 2017). Our findings confirm that gradients often cluster around zero as training progresses. Notably, gradients in layers closer to the input are generally smaller.

This observation suggested that if a layer's gradients barely change and overall loss is stable, skipping the computations for that layer would likely have negligible impact on accuracy. To operationalize these ideas, GSkip checks two conditions after each backward pass:

1) **High gradient similarity:** If the cosine similarity between the current gradient $G^P$and the previous iteration's gradient G^P_{prev} exceeds a threshold s^p_t, GSkip flags that layer's gradients as potentially skippable. The threshold $s^P_t = \frac{1}{1 + e^{-0.2 \times \text{ASC}}}$ is maintained by an **Adaptive Saturating Counter (ASC)**, which adjusts automatically based on recent gradient similarity patterns.
2) **Loss stagnation:** GSkip uses an Exponential Moving Average (LEMA) of the loss to detect if the network's performance has plateaued. A configurable threshold l_t determines whether each new loss measurement E indicates stagnation. If the loss is consistently near the LEMA (i.e., E≈LEMA), GSkip considers the network's learning progress marginal.

When both conditions are met, GSkip skips updating the gradient for that layer and all preceding layers in the subsequent iteration, imposing a "waterfall" skip that can significantly reduce redundant computations. Otherwise, it proceeds with standard gradient calculations. GSkip begins with a **warm-up phase** of full training—allowing the network to stabilize—then adjusts behavior using a **Skip-History** of recent decisions. If the network has not skipped for multiple steps, GSkip loosens its similarity threshold to encourage skipping; if it has skipped recently, it tightens the threshold, ensuring that only highly similar gradients are repeatedly bypassed. After each epoch, GSkip reviews whether the loss has degraded or improved. A degraded or stagnant loss prompts more conservative skipping (tightening l_t and resetting ASC), while an improving loss allows for more skipping (relaxing l_t). GSkip also continuously rechecks skipped layers; if conditions no longer justify skipping, those layers are reactivated.

By inhibiting gradient calculations and activation storage for less impactful layers, GSkip cuts down on memory transfers and computational overhead. Although it adds minor history-keeping costs (maintaining past gradients, counters, and the LEMA), these are typically outweighed by the savings in both time and resources.

EXPERIMENTAL SETUP AND INITIAL RESULTS

GSkip was implemented in Python, integrated with PyTorch v1.13, and tested on a NVIDIA RTX 3090 GPU with CUDA 11.7 (Paszke et al., 2019). We conducted experiments on ResNet18, ResNet50, DenseNet121, and MobileNetV2 with the CIFAR10 and CIFAR100 datasets, as well as on the ImageNet dataset with ResNet18. For CIFAR10 and CIFAR100, we employed Stochastic Gradient Descent (SGD) with momentum 0.9, weight decay of 0.0001, and an initial learning rate of 0.1, across 200 epochs with a batch size of 128. For ImageNet, we trained for 90 epochs, reducing the learning rate by a factor of 10 at epochs 0, 30, and 60, with a batch size of 256. GSkip was configured to consider the past four network losses, using a threshold $l_t = 0.00005$ and a burn-in period of 750 steps.

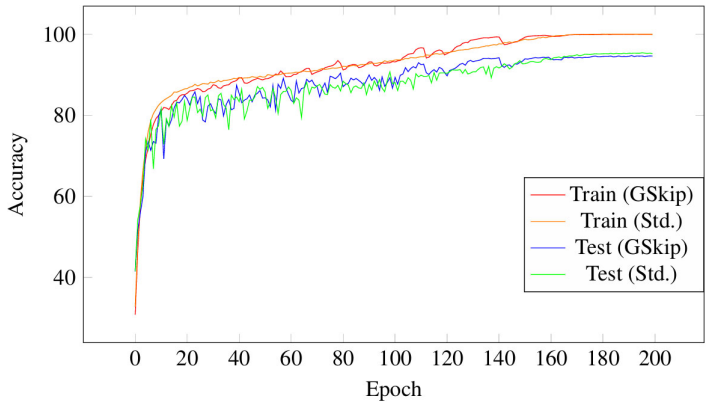

Figure 1: Training vs. validation accuracy for ResNet18 on CIFAR10 with GSkip and standard training.

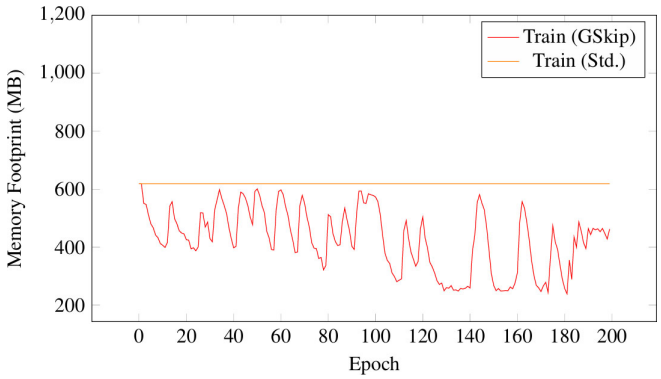

Figure 2: Average memory footprint for ResNet18 on CIFAR10 with GSkip and standard training.

Our evaluation began with **ResNet18 on CIFAR10**, to see whether GSkip could effectively identify and skip non-critical updates in earlier layers. **Figure 1** compares top-1 training ("Train") and validation ("Test") accuracies between standard (baseline) and GSkip-enabled training (Gskip).

GSkip achieves a test accuracy of 94.7%, just a 0.7% drop from the baseline's 95.4%, while skipping 28.84% of gradient computations (i.e., the proportion of skipped updates relative to the total). **Figure 2** shows that GSkip reduces memory footprint by approximately 1.67×, with the savings becoming more pronounced in later stages of training.

Evaluation of GSkip Across Datasets and Networks

Effect on CIFAR10: Table 1 summarizes results for ResNet18, ResNet50, DenseNet121, and MobileNetV2 on CIFAR10. GSkip attains similar validation accuracies, sometimes with minor degradation (up to –0.75%) for ResNet50 and MobileNetV2, or even a slight improvement (+0.55%) for DenseNet121. Memory footprint reductions average around 1.5×, and skip rates remain above 24%.

Effect on CIFAR100: On the more challenging CIFAR100 dataset, GSkip maintains accuracy within about 1% of the baseline, occasionally surpassing it (e.g., DenseNet121, MobileNetV2) as shown in **Table 1.** Memory usage drops by around 1.29×, reflecting gradient skipping rates of at least 19.4%.

Table 1. Training statistics for CIFAR10 and CIFAR100.

CNN	CIFAR10				CIFAR100			
	Std	GSkip			Std	GSkip		
	Acc. %	Acc. %	Mem. Dec.	Skip Rate %	Acc. %	Acc. %	Mem. Dec.	Skip Rate %
ResNet18	95.41	94.7	1.45×	28.84	75.6	74.82	1.26	24.21
ResNet50	94.68	95.23	1.51×	35.47	76.64	75.28	1.36	29.74
DenseNet-121	95.67	94.92	1.7×	42.551	79.14	79.34	1.37	30.17
MobileNet-V2	92.63	92.94	1.36×	24.344	71.78	71.88	1.15	19.4

Effect on ImageNet: For ImageNet, we evaluated ResNet18 training. GSkip maintained a final accuracy of 69.13% compared to the baseline's 69.87%, skipping 16.1% of gradients and reducing memory usage by 1.18×. **Figure 3** illustrates these findings, demonstrating GSkip's potential to lower computational overhead on larger-scale tasks while preserving most of the accuracy.

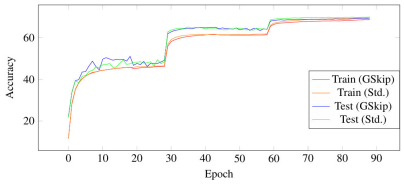

Figure 3: Training vs. validation accuracy for ResNet18 on ImageNet with GSkip and standard training.

Overall, these findings indicate GSkip's potential, guiding further investigation into its capabilities and limitations.

Control Experiments

At this point, a standard evaluation would have declared GSkip a moderate success. However, we questioned whether these experiments truly demonstrated the "intelligence" of GSkip's decisions or if simpler approaches could achieve similar effort reductions given the observed accuracy drop. We also investigated how training time correlates with accuracy improvements, providing insights into the real efficiency of work-skipping methods like GSkip. We settled on two control experiments:

1) **Random Skipping:** Mini-batches were omitted at random, without regard for their potential impact on training accuracy. We set the skipping rate to mirror GSkip's proportion of skipped workload, noting that backpropagation typically consumes about two-thirds of the computational load (Hobbhahn & Sevilla, 2021). For example, on CIFAR100 with ResNet18, we applied a 14% random skipping rate, approximating GSkip's average.

Table 2 compares final accuracies for GSkip and random skipping under the same amount of skipped workload. Surprisingly, ResNet18 achieved 75.15% accuracy using random skipping, slightly outperforming GSkip's 74.82%. Similar trends were observed for ResNet50 and MobileNetV2, while GSkip only outperformed both random skipping and standard training on DenseNet-121.

2) **Early Termination:** We measured "time to accuracy" rather than focusing solely on final accuracy. Deep learning models often exhibit diminishing returns on accuracy gains in later epochs (Yarally et al., 2023). For instance, halting baseline training of ResNet18 on ImageNet at epoch 64 achieved 69.13% accuracy—identical to GSkip's final accuracy after 90 epochs—indicating that 26 additional epochs yielded just a 0.75% boost over baseline training.

Table 2. Control experiments with CIFAR100.

CNN	Baseline/GSkip/Random Testing % Accuracy	Baseline/GSkip/Random Workload Ratio %
ResNet18	75.60 / 74.82 / 75.15	100 / 83 / 83
ResNet50	76.64 / 75.28 / 75.76	100 / 80 / 80
DenseNet-121	79.14 / 79.14 / 78.87	100 / 87 / 87
MobileNetV2	71.78 / 71.88 / 72.07	100 / 79 / 79

These findings suggest that the benefits attributed to GSkip and other methods might have been mischaracterized due to inadequate baseline comparisons. This underscores the importance of rigorous evaluation

methodologies, which we will address next by proposing robust evaluation guidelines.

Evaluation Guidelines

To ensure a fair evaluation of workload skipping techniques, we propose the following guidelines, which reflect real-world implications of these methods:

1. **Comparison with Random Workload Skipping:** By omitting a fixed percentage of batches, layer updates, or samples randomly during training (including random omission at the sample, batch, step, or layer level to align with the target method), we can assess objectively if techniques offer genuine improvements.
2. **Incorporation of Early Stopping or Time to Accuracy:** Using early stopping or tracking the time to reach a specified accuracy helps gauge the effectiveness of a skipping method in accelerating training. Using it, researchers can determine if the computational savings from skipping methods are substantial enough to justify their complexity.
3. **Utilization of Pareto Curves for Accuracy vs. Time Analysis:** Building on the concept of time to accuracy, Pareto curves illustrate the trade-off between accuracy and training time across different budgets and help identify the most efficient balance for various algorithms. Conventionally, generating these curves requires multiple training runs, which is resource-intensive. Using multiplicative cyclic learning rates can construct these curves in a single session (Portes et al., 2022).
4. **Wall Clock Time as the Primary Metric:** Traditional measures such as accuracy at a certain epoch count can be misleading because they do not account for the actual time used during training. Using wall clock time ensures that reported improvements in computational efficiency reflect the total training duration to a specified accuracy level.
5. **Memory Usage Analysis:** Besides reducing training time, effective algorithms should optimize memory consumption. Evaluations must consider hardware differences, network architectures, and batch sizes, but detailed exploration of these factors is outside the scope of this paper.

Applying Guidelines to GSkip

Figure 4 refers to the Pareto curves for CIFAR100 training on ResNet50, showing validation accuracy versus training time for standard training, GSkip, and random mini-batch skipping. The dotted lines indicate polynomial fits, illustrating the Pareto front for each method. While GSkip reduces training time, its advantage over both standard training and random skipping is not consistently significant across all accuracy levels. Although GSkip performs better in certain ranges, these differences are not significant enough to conclusively assert its effectiveness. This finding underscores the importance of applying these evaluation guidelines to accurately measure the benefits of any workload skipping algorithm.

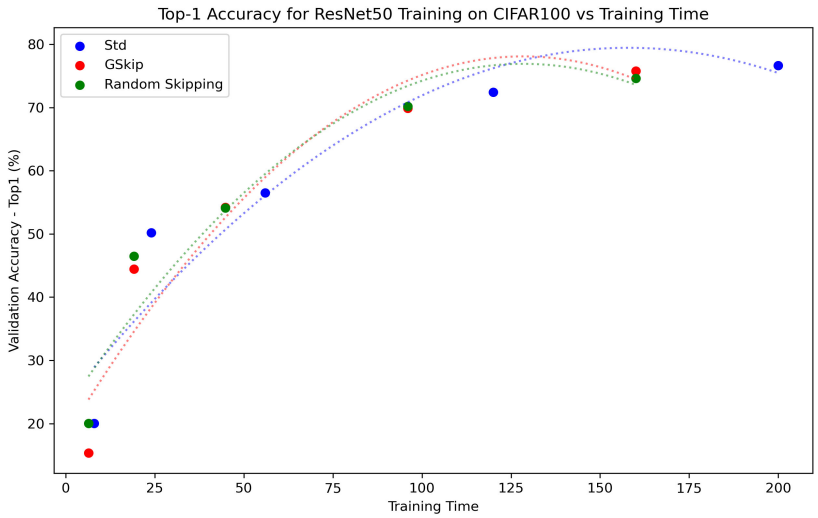

Figure 4: Pareto curves showing validation accuracy vs. training time for standard training, GSkip, and random mini-batch skipping on CIFAR100 with ResNet50.

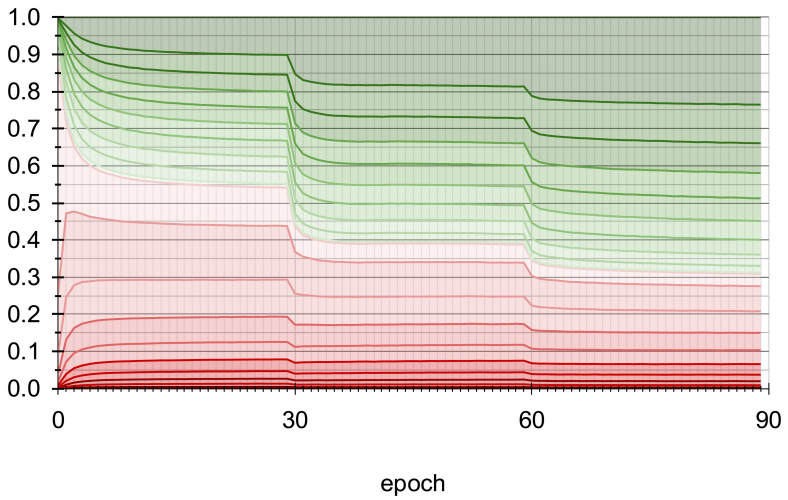

Figure 5: Confidence of prediction during training with ResNet18 on ImageNet. Correct/incorrect predictions are marked with green/red, with darker shades indicating higher confidence.

DeadHorse

Typically, each epoch processes every sample in the training dataset, bundling them into batches to ensure diversity. DeadHorse performs *sample skipping*, deciding when to exclude an input sample from an epoch's processing under the hypothesis that not all samples require constant revisiting—particularly if the network consistently succeeds or fails to learn them.

DeadHorse posits that a sample's **prediction confidence** can signal its necessity in the backward pass. High-confidence, correct predictions induce

minimal parameter updates, while low-confidence ones drive learning. However, ignoring high-confidence samples risks "unlearning," disrupting the balance between well-learned and under-learned samples. **Figure 5** illustrates a breakdown of predictions during ResNet18's full ImageNet training, with correct samples in green and incorrect ones in red, ordered by confidence bands from 0.0 to 1.0. The figure shows that as training progresses, some initially incorrect, low-confidence predictions become correct, whereas others remain persistently "unlearnable."

General Approach: DeadHorse uses softmax confidence S and compare it to a threshold T to decide whether to skip a sample's backward pass. Several **Reactive** and **Predictive** policies that decide how samples are skipped were explored. Reactive policies calculate confidence during the forward pass and selectively skip the backward pass based on this. Predictive policies potentially skip both the forward and backward passes using historical data.

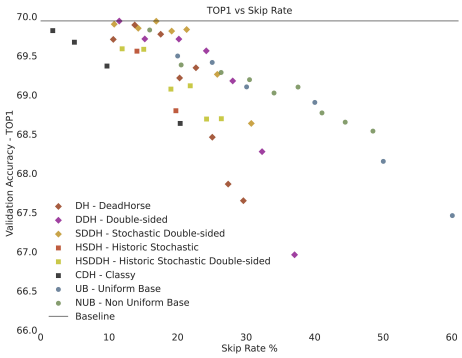

Figure 6: Deadhorse operations count: skip rate vs. validation accuracy of ResNet18 on ImageNet.

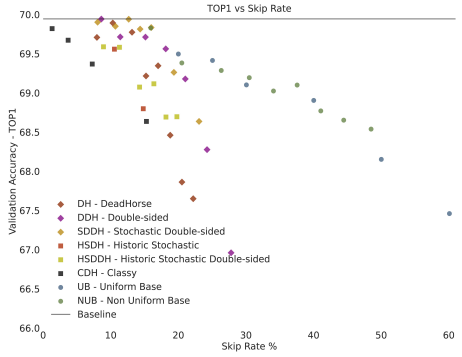

Figure 7: Deadhorse off-chip memory transfers: skip rate vs. validation accuracy of ResNet18 on ImageNet.

- **Reactive Policies:** (1) Basic DeadHorse (DH) skips correct, high-confidence samples. (2) Double-Sided DeadHorse (DDH) relies solely on confidence level, regardless of correctness. (3) Stochastic Double-Sided DeadHorse (SDDH) adds randomness to reduce overfitting.

- **Predictive Policies:** (1) Historic Stochastic Deadhorse (HSDH) uses a previous pass's results to skip correct, high-confidence samples. (2) Historic Stochastic Double-sided Deadhorse (HSDDH) skips any high-confidence samples, correct or not. (3) Classy Deadhorse (CDH) focuses on entire classes that display consistently high confidence across epochs.

Metadata (confidence and correctness) is tracked per sample or class, depending on the policy. We evaluated DeadHorse on ResNet18/ImageNet for 90 epochs, adjusting learning rates at epochs 0, 30, and 60, then applied "time to accuracy" to compare each policy's speed against the baseline.

DeadHorse's "intelligence" was also compared against two random-skipping baselines: **Uniform**, which randomly skips samples at a fixed rate, and **Non-Uniform**, which matches DeadHorse's average skip probability per 30-epoch segment. **Figure 6** shows the operation count for our experiments and indicates that only a few Stochastic Double-sided DH thresholds (0.8, 0.7, 0.6) and Double-sided DH thresholds (0.8, 0.7) outperform the baseline slightly. In **Figure 7**, comparing off-chip memory transfers shows no decisive advantage for any DeadHorse policy; the marginal gain at threshold 0.8 for Stochastic Double-sided DH is inconclusive.

All in all, our results show all DH methods fail to outperform the baseline consistently. Future changes in algorithms, tiling, or trade-offs might make this approach more relevant. For now, the benefits of DeadHorse are marginal at best.

CONCLUSION

Our work advances the evaluation of training acceleration methods, particularly work-skipping techniques. We establish guidelines to assess whether a work-skipping policy effectively balances accuracy and acceleration. While training often requires significant effort for small accuracy gains, it is easy to reduce work with minimal impact on accuracy. By proposing and evaluating two new work-skipping techniques, we provide best-practice guidelines for future research. Notably, our GSkip method reduces work without affecting accuracy.

REFERENCES

Chakrabarti, A., & Moseley, B. (2019). Backprop with Approximate Activations for Memory-efficient Network Training. Proceedings of the 33rd International Conference on Neural Information Processing Systems.

Coquelin, D., Debus, C., Götz, M., von der Lehr, F., Kahn, J., Siggel, M., & Streit, A. (2022). Accelerating neural network training with distributed asynchronous and selective optimization (DASO). Journal of Big Data, 9(1).

Dahl, G. E., Schneider, F., Nado, Z., Agarwal, N., Sastry, C. S., Hennig, P., Medapati, S., Eschenhagen, R., Kasimbeg, P., Suo, D., Bae, J., Gilmer, J., Peirson, A. L., Khan, B., Anil, R., Rabbat, M., Krishnan, S., Snider, D., Amid, E., … Mattson, P. (2023). Benchmarking neural network training algorithms.

Goli, N., & Aamodt, T. M. (2020). Resprop: Reuse sparsified backpropagation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1548–1558.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778.

Hobbhahn, M., & Sevilla, J. (2021). What's the backward-forward FLOP ratio for neural networks? https://epochai.org/blog/backward-forward-FLOP-ratio

Krizhevsky, A., Hinton, G., & others. (2009). Learning multiple layers of features from tiny images.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). Pytorch: An imperative style, high-performance deep learning library. arXiv Preprint arXiv:1912.01703.

Portes, J., Blalock, D., Stephenson, C., & Frankle, J. (2022). Fast Benchmarking of Accuracy vs. Training Time with Cyclic Learning Rates (arXiv:2206.00832). arXiv. https://doi.org/10.48550/arXiv.2206.00832

Safayenikoo, P., & Akturk, I. (2020). Weight Update Skipping: Reducing Training Time for Artificial Neural Networks (arXiv:2012.02792).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4510–4520.

Shen, L., Sun, Y., Yu, Z., Ding, L., Tian, X., & Tao, D. (2023). On efficient training of large-scale deep learning models: A literature review.

Sun, X., Ren, X., Ma, S., & Wang, H. (2019). meProp: Sparsified Back Propagation for Accelerated Deep Learning with Reduced Overfitting (arXiv:1706.06197). arXiv. https://doi.org/10.48550/arXiv.1706.06197

Thompson, N. C., Greenewald, K., Lee, K., & Manso, G. F. (2020). The computational limits of deep learning.

Wei, B., Sun, X., Ren, X., & Xu, J. (2017). Minimal effort back propagation for convolutional neural networks. arXiv Preprint arXiv:1709.05804.

Yarally, T., Cruz, L., Feitosa, D., Sallou, J., & van Deursen, A. (2023). Uncovering energy-efficient practices in deep learning training: Preliminary steps towards green AI. 2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN), 25–36. https://doi.org/10.1109/CAIN58948.2023.00012

Zhu, Y., & Newsam, S. (2017). Densenet for dense flow. 2017 IEEE International Conference on Image Processing (ICIP), 790–794.