SwiftLearn: A Data-Efficient Training Method of Deep Learning Models using Importance Sampling

Habib Hajimolahoseini, Omar Mohamed Awad, Walid Ahmed,
Austin Wen, Saina Asani, Mohammad Hassanpour, Farnoosh Javadi,
Mehdi Ahmadi, Foozhan Ataiefard, Kangling Liu, Yang Liu
*Ascend Team, Toronto Research Center, Huawei Technologies
`habib.hajimolahoseini@huawei.com`

Abstract

In this paper, we present *SwiftLearn*, a data-efficient approach to accelerate training of deep learning models using a subset of data samples selected during the warm-up stages of training. This subset is selected based on an importance criteria measured over the entire dataset during warm-up stages, aiming to preserve the model performance with fewer examples during the rest of training. The importance measure we propose could be updated during training every once in a while, to make sure that all of the data samples have a chance to return to the training loop if they show a higher importance. The model architecture is unchanged but since the number of data samples controls the number of forward and backward passes during training, we can reduce the training time by reducing the number of training samples used in each epoch of training. Experimental results on a variety of CV and NLP models during both pretraining and finetuning show that the model performance could be preserved while achieving a significant speed-up during training. More specifically, BERT finetuning on GLUE benchmark shows that almost 90% of the data can be dropped achieving an end-to-end average speedup of 3.36$\times$ while keeping the average accuracy drop less than 0.92% .

1 Introduction

Over the past few years, deep learning models have undergone a significant increase in size, boasting millions, or even billions, of trainable parameters (Li et al., 2021; Radford et al., 2018). The training of such immense models requires substantial memory resources and computational power (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023). Furthermore, they demand extensive training on large datasets, resulting in prolonged training durations. These challenges impose restrictions on deploying these models on edge devices with limited computational capabilities, thereby limiting their applicability across numerous scenarios.

To tackle these issues, three distinct categories of methods have been introduced to accelerate the training either by Structural Efficiency, Data Efficiency, or Hardware Efficiency. Structural efficiency improvements involve modifications to the model's architecture to enhance computational efficiency, essentially reducing its computational or memory demands. Some notable examples of such methods encompass network pruning (Sun et al., 2023), low-rank decomposition (Hu et al., 2021), and weight quantization (Dettmers et al., 2023), among others. Hardware-efficient training, on the other hand, focuses on achieving high model performance while minimizing the hardware requirements (Dao et al., 2022), which can be crucial for practical deployment in resource-constrained environments, such as edge devices, mobile phones, and embedded systems.

In contrast, data-efficient training approaches do not change the model's architecture. Instead, they focus on expediting the training process by excluding less crucial data samples from the training dataset (Okanovic et al., 2023). In fact, since the number of data samples directly influences the number of forward and backward passes during training, reducing the number of training samples used in each training epoch can substantially reduce the overall training time. This paper primarily concentrates on this category of techniques. The key advantage of this approach lies in its independence from the model's structure or the specific task at hand. Consequently, any successful data-efficient method can be employed concurrently with other techniques aimed at expediting the training process, providing a synergistic boost to overall training speed.

Data-efficient training methods can be classified into three main categories: Dataset Condensation, Dataset Pruning, and Curriculum Learning. Each of these categories aims to optimize the use of training data in deep learning models. Dataset Condensation involves the synthesis of a new, smaller dataset that captures the essential distributional characteristics of the original dataset. This new dataset may not necessarily contain the same data samples as the original one, but its distribution closely aligns with that of the complete dataset. Techniques used for dataset condensation include optimizing Kernel ridge-regression with respect to input data samples (Xu et al., 2023), employing generator functions (Zhao et al., 2021), utilizing auxiliary small unlabeled samples (Sattler et al., 2021), and learning soft labels on a reduced dataset (Vyas et al., 2020).

Dataset Pruning on the other hand focuses on reducing the size of the original dataset by removing less important and less representative data samples. Unlike dataset condensation, the resulting dataset in this case is a subset of the original data. Recent approaches have introduced novel metrics, such as gradient norms, to determine the importance of each data sample (Ahn et al., 2023). Alternatively, a scaled-down model can be trained to serve as a proxy for selecting important data samples based on predefined metrics (Coleman et al., 2020). Other metrics include the number of times a model misclassifies a sample and the high variance of gradients, indicating example difficulty (Dadalto et al., 2023). Prediction depth, the layer at which a k-NN classifier can successfully classify an example, can also serve as a measure of computational difficulty (Ma and Karaman, 2018).

Curriculum learning methods revolve around identifying the optimal order of data samples to accelerate model convergence. Unlike dataset pruning, these methods do not remove data samples but emphasize the importance of the order in which data samples are presented during training. Curriculum learning methods define metrics to quantify the importance of each data sample, and the order of learning is determined based on these scores. These metrics can be crafted manually or generated automatically (Zhang and Pfister, 2021; de Mathelin et al., 2022). While handcrafted metrics may be task-specific, they often outperform automated metrics. Nevertheless, both types of metrics remain valuable for various tasks. It is worth noting that most dataset reduction techniques have primarily been explored in image classification tasks, with limited exploration in text-based tasks.

In this paper, we propose *SwiftLearn*, a technique that could be considered as a combination of dataset pruning and curiculum learning. The proposed algorithm is described in the following sections.

2 Proposed Algorithm

The proposed method is applied in two steps: Warm-up and Sampling:

2.1 Warm-up

In the warm-up step, which happens during the first 2 epochs of each training phase, all data samples in the dataset are fed to the model. During this step, the model logits are recorded in order to be used later for calculating the importance metric. The Mean Squared Error (MSE) between the logits of the model in two consecutive epochs is calculated for each data sample x_i as follows:

$$\text{MSE}(x_i) = ||P_i^s - P_i^{s-1}||_2 \tag{1}$$

where P_i^s is the logits of the model for data sample x_i at epoch s and P_i^{s-1} is the logits of the model for the same sample in the previous epoch $s-1$. This metric shows how the logits of the model are changing between consecutive epochs of training. This measure is then normalized using a softmax function in order to make it a probability distribution:

$$\text{Pr}(x_i) = \frac{e^{(\sigma \times \text{MSE}(x_i))}}{\sum_j e^{(\sigma \times \text{MSE}(x_j))}} \tag{2}$$

where σ is a controllable temperature factor. This probability is used in the next stage as a measure of importance, based on which the data samples are chosen as a subset for training. Note that the warm-up stage is applied during the training of the model (not before that) and no data sample is removed in this stage. We give all the samples a chance to contribute to the training but with different probability of being chosen.

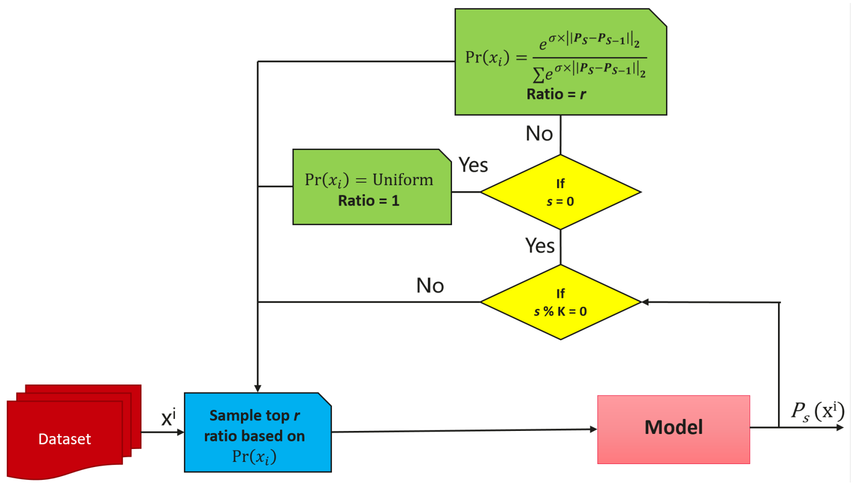
$\Pr(x_i) = \frac{e^{\sigma \times \|P_S - P_{S-1}\|_2}}{\sum e^{\sigma \times \|P_S - P_{S-1}\|_2}}$
Ratio = r
No
Yes
$\Pr(x_i)$ = Uniform
Ratio = 1
If $s = 0$
Yes
No
If s % K = 0
Dataset
x^i
Sample top r ratio based on $\Pr(x_i)$
Model
P_s (x^i)

Figure 1: The proposed data efficient sampling technique

2.2 Sampling

After the sampling metric is calculated/updated in the previous step, this measure is used to sample from the dataset using a ratio r defined by the user. This speed-up is controlled by two factors: ratio r and Metric Re-evaluation Interval. The ratio r controls the number of samples fed through the network(s). The lower the ratio, the less number of data samples are chosen from dataset. If N is the total number of data samples in the dataset, the number of data samples that will be fed during each epoch will be $N \times r$ where $0 \leq r \leq 1$. By feeding only a portion of data-set, the number of forward and backward passes decreases which leads to faster training.

On the other hand, the metric that we find during the warm-up stage may change during training, if the network's status is affected by that. To get the maximum speed up, we need to stick to the metric that we find or we can update that during each epoch for only those samples which have been chosen. This partial or lack of update causes degradation in performance. We can sacrifice the speed-up to perform better by re-evaluating the metric frequently for the whole data-set. By doing that, the forward pass needs to be done for all samples frequently. The length of this re-evaluation interval controls the save in time that our algorithm provides.

The entire data efficient sampling process is depicted in Fig. 1. In this figure, K represents the frequency by which we update the importance metric. At the 0^{th} epoch, all of the samples from the original model are used (ratio r = 1). Every K^{th} epoch, the importance function Pr is calculated using the softmax equation and MSE function. In our experiments, we set K to be the total number of training epochs so the importance metric is set only once for simplicity. Then, the top r number of samples from the dataset are chosen according to the importance metric. This process is repeated until the model is fully trained.

3 Experimental Results

Comparisons with baselines. Here, we evaluate *SwiftLearn* over pretraining and finetuning tasks on a variety of large deep learning models as shown in table 1. Across the models we studied, the SwinT (Liu et al., 2021), Conformer (Gulati et al., 2020), and RoBERTa (Liu et al., 2019) models showed the best speedup-accuracy tradeoff. We pretrained SwinT on the ImageNet dataset (Deng et al., 2009) for 300 epochs with a drop ratio of 0.2 and we achieved an end-to-end (E2E) speedup of 13.6% with no loss in the final validation accuracy. We also pretrained Conformer on the LibriSpeech (Panayotov et al., 2015) dataset for 50 epochs with a drop ratio of 0.3 achieving a 33.8% E2E speedup with only a

Model	Task	Dataset	Drop Ratio	Speed-up	Performance			
					Data-weighting	Baseline	Metric	Δ Acc
T5	Finetuning	wmt16/en-ro	0.3	1.08×	18.44	18.46	Bleu	-0.02
RoBERTa	Finetuning	SST-2	0.3	1.35×	94.00	94.00	Acc %	0.00
ViT	Pretraining	ImageNet	0.1	1.05×	69.89	70.01	Acc %	-0.12
SwinT	Pretraining	ImageNet	0.2	1.136×	81.00	81.00	Acc %	0.00
Conformer	Pretraining	LibriSpeech	0.3	1.338×	95.07	95.32	Acc %	-0.25
BERT-Large	Pretraining	Chinese-wiki	0.2	1.245×	70.68	71.43	Acc %	-0.75

Table 1: Comparison between the accuracy and speed-up of data-weighting and the vanilla training baseline across multiple tasks and models.

-0.25% drop in the validation accuracy. We finetuned RoBERTa on the SST-2 dataset for 10 epochs with a drop ratio of 0.3 achieving a 35% E2E speedup with no loss in the final validation accuracy. This confirms our hypothesis that not all training samples contribute equally to the model's learning and that with a careful selection of important samples, only a smaller subset will be sufficient.

Although finetuning BERT-Large (Devlin et al., 2019) on the Chinese-wiki (Xu and Lapata, 2019) dataset achieved a 24.5% E2E speedup with a drop ratio of 0.2, it suffered from a relatively higher drop in the validation accuracy of -0.75%. This is because Chinese-wiki is a small Pretraining dataset with only 1.2M samples and is trained using only 3 epochs Since the first two epochs are used during the warm-up stage, it may need more epochs during the sampling step to result in a higher accuracy.

Finetuning the T5 (Raffel et al., 2020) model on the WMT 2016 (Bojar et al., 2016) dataset for 3 epochs using a drop ratio of 0.3 achieved an 8% E2E speedup with a slight decrease in the Bleu score of 0.02.

Effect of drop ratio on speed up and accuracy. Table 2 shows how sweeping on the drop ratio affects the E2E speedup and accuracy of finetuning BERT on different tasks from the GLUE benchmark (Wang et al., 2019). All studied tasks showed significant improvement in the E2E speedup up to $3.52\times$. CoLA, SST-2, and QNLI tasks showed an increase in the validation accuracy with drop ratios of 0.9 and 0.7 relative to the baseline of 0.9-4%, and 1.4-4.1%, respectively. SST-2 with a drop ratio of 0.7 showed the highest accuracy improvement with 4.1% increase in the validation accuracy relative to the baseline. Only for MRPC that we saw a significant drop in the validation accuracy of 10.2% and 10.5% for a drop ratio of 0.9 and 0.7, respectively. This can be attributed to MRPC being a relatively more difficult task that requires a much lower drop ratio to maintain the baseline accuracy. However, in general, BERT finetuning on GLUE benchmark shows that almost 90% of the data can be dropped while keeping the average accuracy drop less than 0.92% .

Datasets	RTE			MRPC			CoLA			SST-2			QNLI		
Drop Ratio	0.00	0.70	0.90	0.00	0.70	0.90	0.00	0.70	0.90	0.00	0.70	0.90	0.00	0.70	0.90
Speed Up (×)	1.00	2.09	3.42	1.00	2.00	3.00	1.00	2.29	3.52	1.00	2.20	3.35	1.00	2.22	3.53
Accuracy (%)	65.7	65.9	65.3	99.9	89.4	89.7	80.7	82.9	81.6	87.5	91.6	91.5	88.5	89.9	89.4

Table 2: Studying the effect of drop ratio on BERT finetuning on the GLUE benchmark.

4 Conclusion

The computation and time requirements of pretraining and finetuning large deep learning models are ever-increasing. In this paper, we presented *SwiftLearn*, a dataset sampling technique that weighs samples dynamically during training based on how much they contribute to the model's learning enabling selection of only the important samples that matter. This results in significant computation and time savings. For example, BERT finetuning on GLUE benchmark can be done with less than 0.92% drop in average accuracy while almost 90% of the data is dropped. *SwiftLearn* opens a lot of doors for future data-efficient training. Instead of fixing the drop ratio throughout the whole training process, it can dynamically change based on the model's learning. In addition, more advanced techniques to weight the dataset samples can be used to further improve the model's training efficiency. Due to the time and space constraints, we leave those ideas for future work.

References

Sumyeong Ahn, Seongyoon Kim, and Se young Yun. 2023. Mitigating dataset bias by using per-sample gradient.

Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In Proceedings of the First Conference on Machine Translation, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2020. Selection via proxy: Efficient data selection for deep learning.

Eduardo Dadalto, Marco Romanelli, Georg Pichler, and Pablo Piantanida. 2023. A data-driven measure of relative uncertainty for misclassification detection.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness.

Antoine de Mathelin, Francois Deheeger, Mathilde Mougeot, and Nicolas Vayatis. 2022. Fast and accurate importance weighting for correcting sample bias.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognition.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Tianda Li, Yassir El Mesbahi, Ivan Kobyzev, Ahmad Rashid, Atif Mahmud, Nithin Anchuri, Habib Hajimolahoseini, Yang Liu, and Mehdi Rezagholizadeh. 2021. A short study on compressing decoder-based language models. arXiv preprint arXiv:2110.08460.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows.

Fangchang Ma and Sertac Karaman. 2018. Sparse-to-dense: Depth prediction from sparse depth samples and a single image.

Patrik Okanovic, Roger Waleffe, Vasilis Mageirakos, Konstantinos E. Nikolakakis, Amin Karbasi, Dionysis Kalogerias, Nezihe Merve Gürel, and Theodoros Rekatsinas. 2023. Repeated random sampling for minimizing the time-to-accuracy of learning.

OpenAI. 2023. Gpt-4 technical report.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, pages 5206–5210. IEEE.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67.

Felix Sattler, Tim Korjakow, Roman Rischke, and Wojciech Samek. 2021. Fedaux: Leveraging unlabeled auxiliary data in federated learning.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A simple and effective pruning approach for large language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Nidhi Vyas, Shreyas Saxena, and Thomas Voice. 2020. Learning soft labels via meta learning.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Yumo Xu and Mirella Lapata. 2019. Weakly supervised domain detection.

Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Kernel ridge regression-based graph dataset distillation. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '23, page 2850–2861, New York, NY, USA. Association for Computing Machinery.

Zizhao Zhang and Tomas Pfister. 2021. Learning fast sample re-weighting without reward data.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2021. Dataset condensation with gradient matching.