AdaSelection: Accelerating Deep Learning Training through Adaptive Data Subsampling

Minghe Zhang*
minghe_zhang@gatech.edu
Georgia Institute of Technology
USA

Chaosheng Dong†
chaosd@amazon.com
Amazon
Seattle, WA, USA

Jinmiao Fu
jinmiaof@amazon.com
Amazon
Seattle, WA, USA

Tianchen Zhou
tiancz@amazon.com
Amazon
Seattle, WA, USA

Jia Liang*
Jialiang@stanford.edu
Stanford University
USA

Jia Liu
hliujia@amazon.com
Amazon
Seattle, WA, USA

Bo Liu
boliucs@amazon.com
Amazon
Seattle, WA, USA

Michinari Momma
michi@amazon.com
Amazon
Seattle, WA, USA

Bryan Wang
brywan@amazon.com
Amazon
Seattle, WA, USA

Yan Gao
yanngao@amazon.com
Amazon
Seattle, WA, USA

Yi Sun
yisun@amazon.com
Amazon
Seattle, WA, USA

Abstract

In this paper, we introduce *AdaSelection*, an adaptive sub-sampling method to identify the most informative sub-samples within each minibatch to speed up the training of large-scale deep learning models without sacrificing model performance. Our method is able to flexibly combines an arbitrary number of baseline sub-sampling methods incorporating the method-level importance and intra-method sample-level importance at each iteration. The standard practice of ad-hoc sampling often leads to continuous training with vast amounts of data from production environments. To improve the selection of data instances during forward and backward passes, we propose recording a constant amount of information per instance from these passes. We demonstrate the effectiveness of our method by testing it across various types of inputs and tasks, including the classification tasks on both image and language datasets, as well as regression tasks. Compared with industry-standard baselines, *AdaSelection* consistently displays superior performance.

Keywords

Data Subsampling, Deep Learning, Adaptive Selection

*Work done as an intern at Amazon
†Corresponding author

RelKDD, Aug 2024, Spain

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Minghe Zhang, Chaosheng Dong, Jinmiao Fu, Tianchen Zhou, Jia Liang*, Jia Liu, Bo Liu, Michinari Momma, Bryan Wang, Yan Gao, and Yi Sun. 2024. *AdaSelection*: Accelerating Deep Learning Training through Adaptive Data Subsampling. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Large deep learning models (DNNs) trained with data of massive volume continues to push the state-of-the-art across various domains. However, reducing the time and cost incurred remains a challenging problem. It can take days or weeks to train a single image classification model using conventional optimizers such as minibatch Stochastic Gradient Descent due to the high complexity of most model backbones. For example, the ImageNet contests have seen the parameter size of Convolutional Neural Networks (CNN) increase to over n^9. The gradient computation using back-propagation is the most time-consuming component during the training process. It has been shown that DNNs tend to learn simple patterns first [1], therefore training using these samples at later epochs brings little to no incremental value while being unnecessarily time-consuming. This motivates us to design a framework to identify the most informative sub-samples within each mini-batch during model training. We believe that this approach might become more promising in the era of large language models as shown in Table 1.

Existing sub-sampling works mainly formulate the problem with two approaches. One approach is to learn the data importance score per sample and sub-sample the data with probability proportional to the importance weights, such as Big Loss [2], Small Loss [3], RHO-loss [4], gradient norm [5], etc. The other approach identifies the best sub-samples in an optimization manner. For example, [6]

casts the sample selection as a Mixed Integer Programming (MIP) problem and solves it with an optimization solver. The optimization problem can also be approximated and solved more efficiently using the Bayesian approach [7] or Frank-Wolfe approach [8]. However, none of these methods provide a clear analysis on the trade-off among training time, sampling rate and accuracy. Additionally, these methods have a fixed policy for each iteration/epoch, which do not adapt to distribution shifts in the training batch. Furthermore, none of the existing methods outperform others consistently across all tasks (classification and regression), therefore it's still unclear to users which method to choose when facing a new task.

To address these drawbacks, we propose *AdaSelection*, an algorithm that adaptively combines different baseline methods to select the top $k\%$ most informative sub-samples at each iteration by leveraging intra-method sample-importance and the inter-method method-importance. Our approach can significantly reduces the training time with only a marginal drop in model performance. We demonstrate our algorithm's SOTA performance across different public datasets with various types of inputs and tasks, including three image classification tasks (Cifar10, Cifar100 and SVHN), two regression tasks (an artificial simple linear regression and bike, a public regression task), and one natural language processing task.

In summary, our algorithm possesses the following advantages: (1) tuning-free - it requires no hyper-parameter tuning as the optimal sample-importance and method-importance are adjusted automatically at each iteration, allowing the algorithm to handle cases with sluggish convergence; (2) efficient - the computation overhead is marginal as the importance weights are computed within each forward pass and back-propagation; (3) generic - it has shown SOTA performance across various ML tasks and datasets; (4) *AdaSelection* helps us better understand the proposed model and get to know how well the model is trained to prevent under/overfitting problem.

2 Related work

In recent years, there has been a growing interest in improving the training efficiency of DNNs by identifying the most informative subsamples within each minibatch [2–4, 12]. Training with such subsamples can reduce the amount of data required in each iteration while maintaining the quality of the resulting model. This section gives a comprehensive review of the existing methods, analyzes their strength and weakness, and explains the technical advantage of our proposed model.

2.1 Importance sampling

Existing works [13–17] have proved that reducing the variance of gradient is able to speed up the convergence of SGD and hence shorten the training process. Importance sampling is a common method utilized for this purpose, where the losses of certain samples are prioritized over others. The relationship between the variance of gradient estimates in SGD and the optimal sampling distribution has been established in [18]. The authors determine that the sampling probability should be proportional to the gradient norm, which in simple linear classification, is proportional to the Lipschitz constant of the per-sample loss function. Another paper, [5], uses the gradient norm as a measure of sample importance and derives an upper bound for the gradient norm for optimization.

Table 1: Model sizes and batch sizes in token of the popular large language models.

Model Name	n_{params}	Batch Size
GPT-3 [9]	125M	0.5M
	350M	0.5M
	760M	0.5M
	1.3 B	1M
	2.7 B	1M
	6.7 B	2M
	13.0 B	2M
	175.0 B	3.2M
LLaMA 1 [10]	7B	4M
	13B	4M
	33B	4M
	65 B	4M
LLaMA 2 [11]	7B	4M
	13B	4M
	34B	4M
	70 B	4M

The aforementioned methods improves the convergence speed by assigning probability to each sample when computing gradients, but still leverages all of the training samples for back-propagation. In comparison, Selective-Backprop [2], also known as the Big Loss method, further speeds up the training process by only leveraging the sub-samples with k largest losses in each minibatch. In [3], a noiseless setting with outliers is considered and a variant of simple SGD called Small Loss is proposed. This method selects k samples, then the sample with the smallest current loss, and performs and SGD-like update. The paper presents a theoretical analysis of the robustness of this approach for ML problems represented as sums of convex losses. Selective-Backprop and Small Loss employ divergent methodologies for utilizing losses, with each representing an extreme point within the spectrum of data subsampling techniques. Consequently, their efficacy is contingent upon specific situational factors, limiting their performance to particular contexts. While Another algorithm, Reducible holdout loss selection (RHO-LOSS) [4] calculates data-wise importance based on the irreducible loss difference between the hold-out dataset model and the current model. [12] provides a self-supervised pruning metric with comparable

performance to supervised metrics. Our work differs in that all these methods can be integrated into our policymaker to select the best candidates.

2.2 Coresets selection

Instead of identifying the most informative sub-samples based on sample level importances, coresets selection aims to directly select the whole sub-sample set in an optimization manner. In [7], batch active learning is defined as a coresets selection problem and batch construction is re-casted as optimizing a sparse subset approximation. Meanwhile, [6] uses an approximation algorithm under the framework of Mini-batch GD, which translates the subsampling step into a combinatorial optimization problem, making it cost-efficient. However, this optimization problem must be solved with a convex optimizer, which can be time-consuming with huge variable sets, unlike the most efficient importance sampling method such as big-loss. To mitigate this problem, a Frank-Wolfe method [19] is proposed in [20] to approximate the complete data posterior of the model parameters.

2.3 Meta Learning

Meta Learning is a powerful technique that allows models to fine-tune themselves based on the task at hand, offering a significant advantage over traditional machine learning methods that rely on fixed hyperparameters. This is achieved by using a bilevel optimization method, where the model's parameters are trained in such a way that a small number of gradient steps with a limited amount of training data will result in good generalization performance on the new task. One of the popular approaches to Meta Learning is the Model-Agnostic Meta-Learning (MAML) method [21], which explicitly trains the model parameters to achieve this. In our problem setting, we are similarly related to meta-learning as we aim to adaptively choose the best subsampling method to find the optimal subset for training. A related work [22] proposes a two-level optimization method that assigns weights on tasks and data samples, which could be useful in our setting as well. Another recent work [23] proposes a new weight update rule that significantly improves the fast adaptation process by using a small meta-network to generate per-step hyperparameters such as learning rate and weight decay coefficients. This meta-network could be treated as an inner loop for hyperparameter exploration in a bilevel optimization scenario, and could also be used to choose the subset in our problem setting. Overall, Meta Learning offers a promising approach for our problem as it considers the adaptive tuning of methods to fit the current task, similar to our goal of adaptively choosing the best subsampling method for training.

2.4 AdaBoost

AdaBoost [24] is a highly innovative ensemble learning method that was created with the purpose of increasing the efficiency of binary classifiers. This method uses an iterative approach in which it learns from the mistakes of weak classifiers and transforms them into strong ones through the assignment of weights to the data points. The algorithm is based on the principle of assigning greater weight to those data points that were misclassified in the previous iteration and reducing the weight for those that were classified correctly.

Table 2: List of main notation.

Notation	Definition
i	Sample index
m	Candidate method index
t	Iteration index
$f_\theta(\cdot)$	A deep learning model
$l(\cdot)$	The loss function
$\alpha_{i,t}^m$	The importance of sample i at iteration t given method m
w_m^t	The importance of method m at iteration t
$r_t(x_i)$	The reward of sample i at iteration t
$s_{i,t}$	The final importance of sample i at iteration t

To calculate the weight of each data point, we first need to define ℓ_i as the data-wise loss for data point i. AdaBoost then uses this information to determine the weight of each data point through the following:

$$w_i = \frac{1}{2}\log(\frac{1+\ell_i}{1-\ell_i}), \tag{1}$$

which can then be used as the importance score for each data point to facilitate the subsampling process.

3 Proposed Method: *AdaSelection*

In this section, we will discuss our proposed method *AdaSelection*, which borrows the wisdom of selected baseline methods to find the optimal subset of the batch through a majority vote. We also summarize the main notations used in our paper at Table 2.

3.1 Selective Methods

In this section, we list several baseline and state-of-the-art (SoTA) subsampling methods that we will compare with to validate the efficacy of our method. These methods include uniform, Big Loss [2], Small Loss [3], Gradient Norm [5], AdaBoost [24] and Coresets selection [7]. Due to the complexity of solving the coresets selection problem, we develop two importance sampling methods as its approximations: (1) a mixture policy that constructs the subsampling subset with 50% of datapoints having the biggest loss and 50% having the smallest loss, and (2) finding datapoints closest to the mean loss of the whole batch. The intuition is, the coresets are expected to be representative enough for the whole batch, and the mean losses of the sub-samples selected using these two methods are both close to the mean loss of the whole batch.

Given a mini-batch of b data points and a sampling rate of γ, the key features for each baseline method are summarized as follows:

- **Uniform**: randomly choose $b\gamma$ data points in the whole batch;
- **Big Loss**: choose data samples with $b\gamma$ highest training loss;
- **Small Loss**: choose data samples with $b\gamma$ smallest training loss;
- **Gradient Norm**: choose data samples with $b\gamma$ highest norm of gradients;
- **AdaBoost**: choose data samples with $b\gamma$ highest weights from AdaBoost computed from (1);

Algorithm 1 General process for data subsampling baselines.

Input: dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{D}|}$, batchsize b, sampling rate γ, loss function $\ell(\cdot)$, learning rate η, selected list $C = \{\}$.
repeat
  for each epoch e **do**
    for batch B_k **do**
      A forward pass to get losses/gradients for each data point
      Select a subset $\mathcal{M}_k$ from B_k according to a certain subsampling strategy
      Add selected data points into $C = C \cup \mathcal{M}_k$
      if $|C| = |B_k|$ **then**
        do batch SGD update on C
        $C = \{\}$
      end if
    end for
  end for
until Converged

- **Coresets Approximation 1**: choose $b\gamma/2$ data points that have highest training loss and $b\gamma/2$ data points with smallest training loss;
- **Coresets Approximation 2**: choose $b\gamma$ data points closest to average batch training loss.

To summarize, we conclude the general framework for all our baseline methods in Algorithm 1.

Algorithm 2 *AdaSelection* for data subsampling.

Input: dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{D}|}$, batchsize b, sampling rate γ, loss function $\ell(\cdot)$, learning rate η, selected list $C = \{\}$, subsampling M candidates $\{g_m\}_{m=1}^{M}$.
repeat
  for each epoch e **do**
    for batch B_k **do**
      A forward pass to get data-wise losses
      Compute data-wise importance score $\boldsymbol{s_{i,t}}$ according to (5)
      Form a subset $\mathcal{M}_k$ from B_k by selecting top highest x_i with score $\boldsymbol{s_{i,t}}$
      Add selected data points into $C = C \cup \mathcal{M}_k$
      if $|C| = |B|_k$ **then**
        do batch SGD update on C
        $C = \{\}$
      end if
    end for
  end for
until Converged

3.2 *AdaSelection* for subsampling strategy

We now introduce our method and how it combines multiple baseline methods incorporating both the sample importance within each method and the overall importance across all the methods.

Let f_θ be a deep learning model, l be the loss function, $B_t = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ be the mini-batch of k samples at iteration t, and $g_1(\cdot), \ldots, g_M(\cdot)$ be the candidate pool of M baseline methods. For a given method g_m, the importance weight of a sample (x_i, y_i) is defined as

$$\alpha_{i,t}^m = g_m(x_i, y_i, B_t), \tag{2}$$

, where $g_m(x_i, y_i, B_t)$ is derived based on the loss of (x_i, y_i). For example, if m is the Big Loss method, then $g_m(x_i, y_i, B_t)$ will be larger on the samples with larger losses (Softmax function can be applied to achieve the purpose).

To measure the overall importance of method m, inspired by AdaBoost, we adjust the importance of method m based on the variation of the average loss from the last iteration using:

$$w_t^m = w_{t-1}^m \times \exp\left(\frac{\beta(\ell_{t-1}^m - \ell_t^m)}{\ell_{t-1}^m}\right), \tag{3}$$

, where ℓ_t^m is the average loss across all the samples in the mini-batch of iteration t and $\beta \in [-1, 1]$.

At iteration t, given $\alpha_{i,t}^m$ (i.e., the importance of sample i based on method m), and w_t^m (i.e., the importance of method m), the overall importance of sample i considering all the M methods can be expressed as $\boldsymbol{s_{i,t}} = \sum_{m=1}^{M} w_t^m \boldsymbol{\alpha_{i,t}^m}$.

Interestingly, the design of *AdaSelection* can be viewed as a reinforcement learning (RL)/Bandit framework. In this framework, each iteration t corresponds to a state represented by the training batch B_t, and the action taken by *AdaSelection* is the selection of weights w_t^m associated with a subsampling method m. The objective is to learn the effectiveness of each strategy m, reflected by the weights w_t^m, with the aim of identifying the optimal subsampling method that maximizes the frequency of positive rewards.

There exist multiple reward functions within this context. One possible approach is to employ automated curriculum learning (CL), which rewards the selection of temporally rare samples. Incorporating CL offers several advantages. In RL/Bandit scenarios, the reward function often exhibits significant variance, leading to stochasticity in the sampling process and an unknown reward structure. By utilizing CL, it becomes possible to capture the intrinsic structure of the reward function, thus mitigating variance and enhancing stability in the learning process.

As explained in [25], CL focuses on easy samples at the beginning of the training process and gradually increases the difficulty, whereas in our case, an easier sample refers to a sample with a Small Loss. To capture the intrinsic structure of the reward function and reduce variance, we design a reward function for CL in the following:

$$r_t(x_i) \propto \exp\left(-t^\gamma \frac{\ell(i)}{\sum_i \ell^2(i)}\right). \tag{4}$$

At the start of the training process, $\frac{\ell(i)}{\sum_i \ell^2(i)}$ is the dominant term as t is small. Consequently, the subsampling strategy will focus on small, easily learnable losses. As the dominant term t increases, the reward function gradually becomes fair to all samples and has no effect.

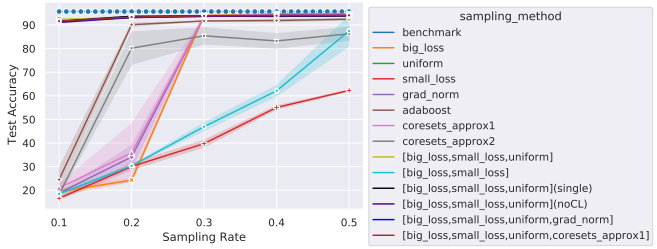


Figure 1: (SVHN) Testing accuracy for different sampling rate from 0.1 to 0.5.

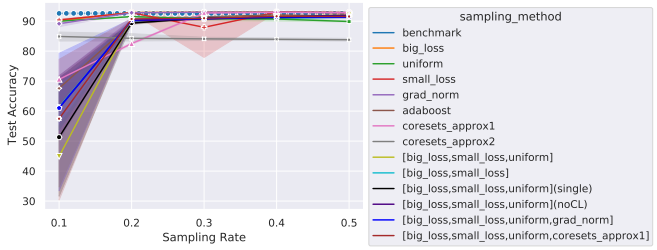


Figure 2: (CIFAR10) Testing accuracy for different sampling rate from 0.1 to 0.5.

The final importance score $s_{i,t}$ of sample i at iteration t is defined as:

$$s_{i,t} = r_t(x_i) \sum_{m=1}^{M} w_t^m \alpha_{i,t}^m \tag{5}$$

By setting the threshold for selecting the most significant scores, we are able to filter the selected data points from B and discard the non-selected ones. The select/not select decision is indicated by a binary variable z_t^i where

$$z_t^i = \begin{cases} 1 & s_{i,t} \text{ above threshold} \\ 0 & s_{i,t} \text{ below threshold.} \end{cases} \tag{6}$$

Our training process has been completed as described in Algorithm 2.

4 Experiment

In our experiments, we vary the data sub-sampling methods applied to various datasets and summarize the configurations, including the learning rate, batch size, backbone model, and dataset size in Table 3. Our learning strategy follows the "biggest losers" approach, reducing the number of batches in each epoch. For the NN model training, we use the ResNet18 backbone for classification tasks and MLP for regression tasks, and optimize using general SGD with momentum weight decay of 0.9.

We report the results for all the datasets listed in Table 3 as follows:

SVHN: To evaluate the performance of our proposed *AdaSelection* method, we conduct experiments on SVHN with the benchmark, which is the original training process without subsampling. Seven methods are selected as our baselines: uniform, Big Loss, Small Loss, AdaBoost, gradient norm, coreset approximation 1, and

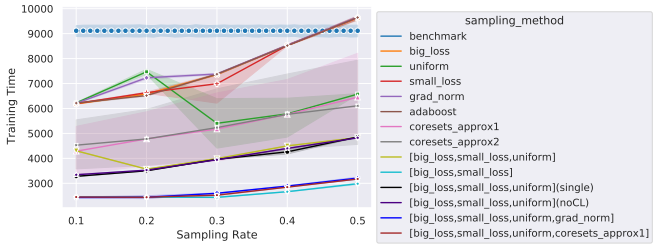


Figure 3: (CIFAR10) Training Time for different sampling rate from 0.1 to 0.5.

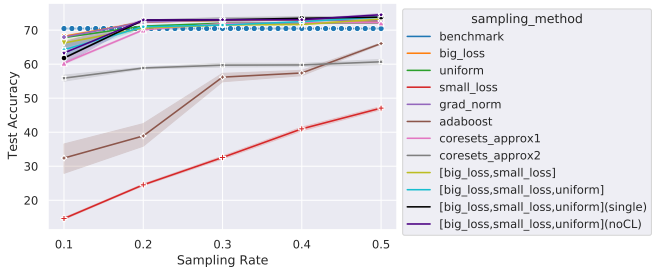


Figure 4: (CIFAR100) Testing accuracy for different sampling rate from 0.1 to 0.5.

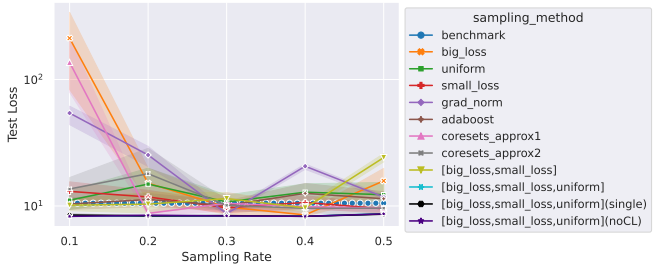


Figure 5: (Regression) Testing Loss for different sampling rate from 0.1 to 0.5.

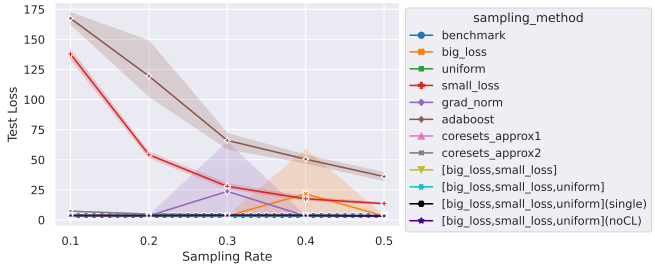
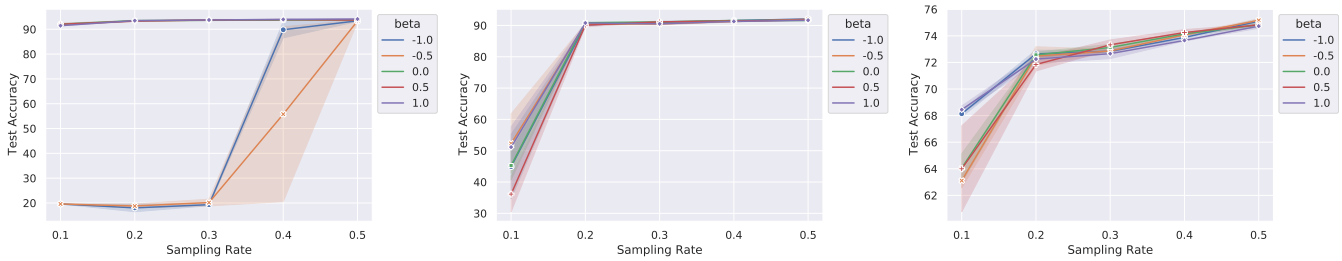


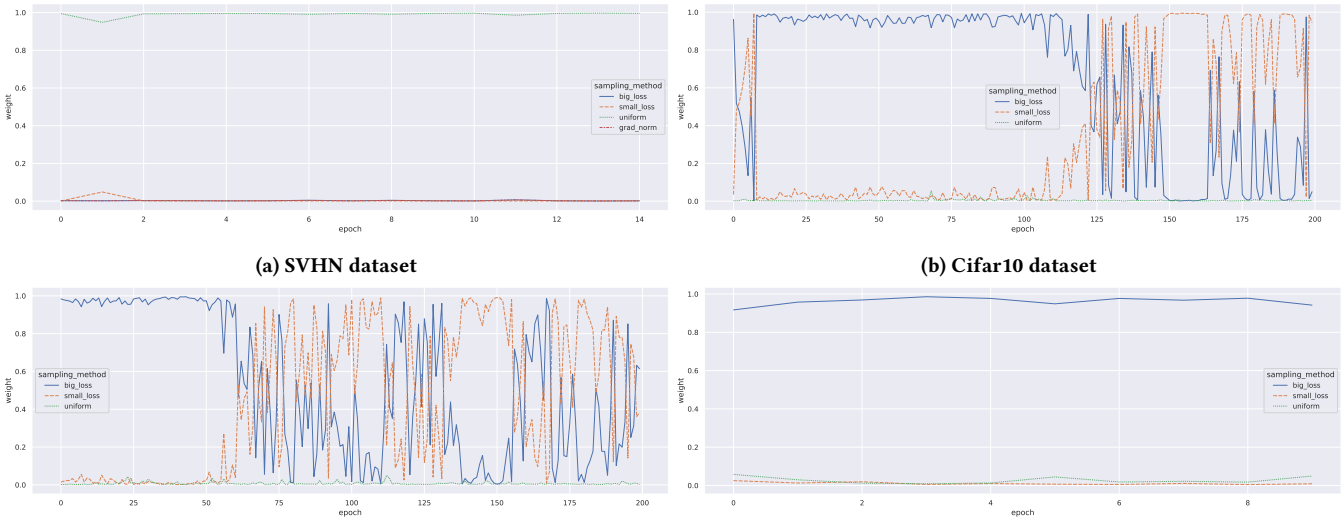
Figure 6: (Regression) Testing Loss for different sampling rate from 0.1 to 0.5.

coreset approximation 2. A detailed explanation of these methods can be found in Section 3.1. Our *AdaSelection* is then conducted by combining several baselines such as [Big Loss, Small Loss], [Big Loss, Small Loss, uniform], etc. The learning rate is then changed



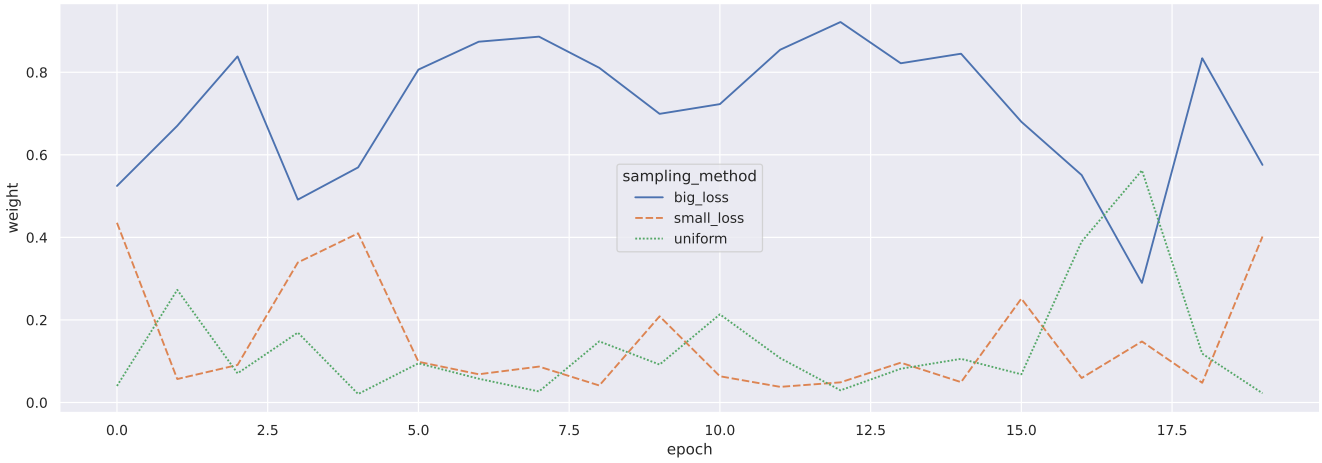
(a) SVHN dataset **(b) Cifar10 dataset** **(c) Cifar100 dataset**

Figure 7: β-selection on different datasets: SVHN, Cifar10 and Cifar100. The range of β is $[-1, -0.5, 0, 0.5, 1]$.



(a) SVHN dataset **(b) Cifar10 dataset**

(c) Cifar100 dataset **(d) Simple regression dataset**



(e) Bike regression dataset

Figure 8: weights on candidates for *AdaSelection* on different datasets: SVHN, Cifar10, Cifar100, simple regression and Bike regression tasks. The sampling rate is set to be 0.2.

Dataset Name	# of classes	# of instances (train+test)	Configurations
CIFAR10 [26]	10	50, 000 + 10, 000	lr=0.01, batch=128, ResNet18
CIFAR100 [26]	100	50, 000 + 10, 000	lr=0.01, batch=128, ResNet18
SVHN [27]	10	73, 257 + 26, 032	lr=0.01, batch=128, ResNet18
Simple regression ($y = 2x + 1$)	NA	10000 + 5000	lr=0.01, batch=100, simple MLP
Bike regression [1]	NA	730 in total	lr=0.01, batch=100, 2-layer MLP
Wikitext-2 [28]	NA	2, 088, 628 + 245, 569	lr=0.01, batch=100, Transformer

Table 3: Dataset descriptions and summary

	Benchmark (no sampling)	*AdaSelection*	Uniform	Big Loss	Small Loss	AdaBoost	Grad_norm	Coresets1	Coresets2
Cifar10	3.5	8.25	8.25	**1.75**	4.75	7.5	3.0	7.0	11.75
Cifar100	8.75	**1.5**	6.25	4.25	12.0	10.75	3.5	7.0	10.25
SVHN	1.0	**4.4**	5.8	6.0	13.8	9.8	7.2	6.8	11.8
Regression	7.4	**1.8**	9.6	8.6	6.8	7.4	9.4	7.4	7.6
Bike	5.6	**3.8**	4.0	4.2	10.8	12.0	5.6	7.0	9.6
Wikitext-2	1.6	**2.0**	2.4	4.4	7.8	8.2	-	5.2	5.4

Table 4: Average ranking for testing accuracy with different sampling rate from 0.1 to 0.5. For *AdaSelection*, we choose the best ranking over several choices of *AdaSelection*: single choice, no CL setting, three candidates (Big Loss, Small Loss, Uniform), two candidates (Big Loss, Small Loss). Coresets1 represents Coresets approximation 1 with 50% Big Loss and 50% Small Loss while Coresets2 represents Coresets approximation 2 that aims to find best subset that contain datapoints that has loss that is closest to the average training loss. The bold number represents the best overall average ranking except the benchmark.

Table 5: Performance accuracy(classification)/loss(regression) by taking average for different sampling rate from 0.1 to 0.5.

	Benchmark (no sampling)	*AdaSelection*	Uniform	Big Loss	Small Loss	AdaBoost	Grad_norm	Coresets1	Coresets2
Cifar10	92.57%	89.39%	90.74%	**92.25%**	91.04%	85.13%	91.97%	84.67%	84.29%
Cifar100	70.47%	**71.38%**	71.05%	70.83%	31.96%	50.19%	71.29%	69.33%	59.98%
SVHN	95.71%	**93.38%**	93.32%	65.38%	40.74%	78.14%	67.14%	67.89%	70.66%
Regression	10.51	**8.42**	12.37	52.25	10.94	11.14	24.13	35.05	12.18
Bike	3.61	**3.45**	3.48	7.08	50.19	87.90	7.53	3.75	7.54
Wikitext-2	5.45	**5.52**	5.53	5.58	5.85	5.85	-	5.61	5.60

from 0.1 to 0.5 for complete training with ResNet on all tasks. Finally, we compare our proposed *AdaSelection* with the top-performed baseline methods by plotting the accuracy and training time versus sampling rate to provide a complete view.

To ensure the superiority of our proposed method, we rank its training accuracy against various methods and present the average ranking and training accuracy in Tables 4 and 5, respectively. The average is calculated by taking the mean of the target metric under sampling rates of 0.1, 0.2, 0.3, 0.4, and 0.5. Figure 1 shows the testing accuracy of the SVHN method, demonstrating that with four baseline methods (Big Loss, Small Loss, uniform, grad norm/coreset1) as candidates, the training accuracy can surpass the benchmark with a 95% accuracy rate even when the sampling rate is only 0.1.

Cifar10 and Cifar100: Our analysis on two other classification tasks on Cifar10 (Figure 2) and Cifar 100 (Figure 4) shows that our *AdaSelection* method can achieve better performance by assigning higher weights to the best performer when properly chosen candidates are used. As an example, Figure 3 demonstrates that for the Cifar10 dataset, almost all subsampling methods reduce the training time by a minimum of 20% for all sampling rates.

Regression: The results of the simple regression task (Figure 5) and Bike regression task (Figure 6) show that the top performer in baseline methods for regression tasks is different than for classification tasks. The Big Loss method does not perform better than the Small Loss method in regression tasks, whereas in classification tasks, Small Loss is not performing well compared to the Big Loss method. However, the *AdaSelection* method consistently selects the best performer among its candidates, resulting in the best performance among all other baselines in regression tasks.

Transformer: For the sequence-to-sequence language model with the Transformer module, our results show that *AdaSelection* even with default parameters, outperforms other baseline methods[2] as demonstrated by the results in Figure 9.

The β selection for *AdaSelection* for classification tasks is shown in Figure 7 to illustrate the importance of β choice under different tasks. We can see SVHN dataset is more sensitive with a choice of β than Cifar10/100. From Figure 8, we see the evolution of the weights for different tasks is also quite different. This suggests the

[2]Gradient Norm failed to work in this transformer task because, for the NLP tasks, we cannot directly achieve gradients from the bag-of-words. Hence, we do not report its results.

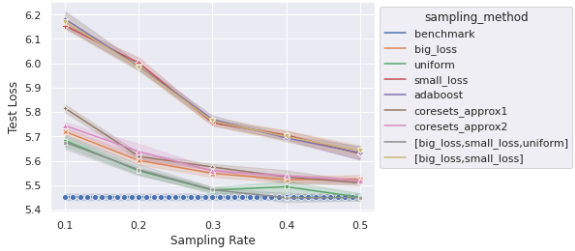


Figure 9: (Transformer) Testing Loss for different sampling rate from 0.1 to 0.5.

necessity of our proposed methods that adaptively select the best sampling strategies during the training process.

The results of the experiment demonstrate that the *AdaSelection* method outperforms all the candidate baselines and is the best selection among them. Additionally, the *AdaSelection* method is more robust and does not require tuning for different tasks, making it a guaranteed superior choice compared to the baseline methods.

5 Conclusion and Future work

In this paper, we introduce *AdaSelection*, an adaptive subsampling method that determines the optimal policy during specific training periods. *AdaSelection* combines baseline methods and assigns weights to each approach to achieve enhanced efficiency and accuracy. Experiments in image classification, text classification, linear regression, and natural language processing tasks have shown *AdaSelection*'s superiority, demonstrating its effectiveness in diverse tasks.

The future of *AdaSelection* holds potential for further exploration. One possible extension is using it as an indicator for stopping the learning process, which is straightforward. *AdaSelection* currently requires a forward pass for each batch to determine the training loss, which serves as the importance for feeding to baseline methods such as Big Loss or Small Loss. However, this process takes time and is not used for backward propagation, making it unnecessary. A forward pass approximation can be used instead to determine data-wise importance training, similar to a teacher-student model.

References

[1] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, 2017.

[2] 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.

[3] Vatsal Shah, Xiaoxia Wu, and Sujay Sanghavi. Choosing the sample with lowest loss makes sgd robust. In *International Conference on Artificial Intelligence and Statistics*, pages 2120–2130. PMLR, 2020.

[4] Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pages 15630–15649. PMLR, 2022.

[5] 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.

[6] Chaosheng Dong, Xiaojie Jin, Weihao Gao, Yijia Wang, Hongyi Zhang, Xiang Wu, Jianchao Yang, and Xiaobing Liu. One backward from ten forward, subsampling for large-scale deep learning. *arXiv preprint arXiv:2104.13114*, 2021.

[7] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[8] Trevor Campbell and Tamara Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. In *International Conference on Machine Learning*, pages 698–706. PMLR, 2018.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[11] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[12] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *arXiv preprint arXiv:2206.14486*, 2022.

[13] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(1), 2013.

[14] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

[15] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

[16] Thomas Hofmann, Aurelien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. *Advances in Neural Information Processing Systems*, 28, 2015.

[17] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

[18] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9. PMLR, 2015.

[19] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435. PMLR, 2013.

[20] Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. *Advances in neural information processing systems*, 32, 2019.

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[22] Hae Beom Lee, Hayeon Lee, Donghyun Na, Saehoon Kim, Minseop Park, Eunho Yang, and Sung Ju Hwang. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. *arXiv preprint arXiv:1905.12917*, 2019.

[23] Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. *Advances in Neural Information Processing Systems*, 33:20755–20765, 2020.

[24] Robert M Freund, Paul Grigas, and Rahul Mazumder. Adaboost and forward stagewise regression are first-order convex optimization methods. *arXiv preprint arXiv:1307.1192*, 2013.

[25] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[27] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[28] Merity Stephen, Xiong Caiming, Bradbury James, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017.