One Backward from Ten Forward, Subsampling for Large-Scale Deep Learning

Chaosheng Dong*†
chaosd@amazon.com
Amazon

Xiaojie Jin‡
jinxiaojie@bytedance.com
ByteDance Inc.

Weihao Gao
weihao.gao@bytedance.com
ByteDance Inc.

Yijia Wang
yiw94@pitt.edu
University of Pittsburgh

Hongyi Zhang
hongyiz@mit.edu
ByteDance Inc.

Xiang Wu
Xwu3@snapchat.com
Snap Inc.

Jianchao Yang
yangjianchao@bytedance.com
ByteDance Inc.

Xiaobing Liu
will.liu@bytedance.com
ByteDance Inc.

ABSTRACT

Deep learning models in large-scale machine learning systems are often continuously trained with enormous data from production environments. The sheer volume of streaming training data poses a significant challenge to real-time training subsystems and ad-hoc sampling is the standard practice. Our key insight is that these deployed ML systems continuously perform forward passes on data instances during inference, but ad-hoc sampling does not take advantage of this substantial computational effort. Therefore, we propose to record a constant amount of information per instance from these forward passes. The extra information measurably improves the selection of which data instances should participate in forward and backward passes. A novel optimization framework is proposed to analyze this problem and we provide an efficient approximation algorithm under the framework of minibatch SGD as a practical solution. We also demonstrate the effectiveness of our framework and algorithm on several large-scale classification and regression tasks, when compared with competitive baselines widely used in industry.

KEYWORDS

deep learning, data subsampling, large-scale

ACM Reference Format:
Chaosheng Dong, Xiaojie Jin, Weihao Gao, Yijia Wang, Hongyi Zhang, Xiang Wu, Jianchao Yang, and Xiaobing Liu. 2018. One Backward from Ten Forward, Subsampling for Large-Scale Deep Learning. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

*Work done prior to joining Amazon
†Equal contribution.
‡Equal contribution.

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Deep neural networks (DNNs) have achieved unprecedented success in many machine learning tasks, for example, in computer vision [1, 2], speech recognition [3], natural language processing [4], recommender systems [5], and game playing [6, 7]. As these DNNs typically have a huge number of learnable parameters, they require millions of data for training. For example, in computer vision, state-of-art DNN models (e.g., [1, 2]) use the ImageNet [8] that contains more than 1.4M images. In natural language processing, BERT [9] uses the BooksCorpus (800M words) and English Wikipedia (2, 500M words) for the pre-training corpus. In recommendation systems, RecVAE [10] uses the Netflix dataset that contains more than 100M movie ratings performed by anonymous Netflix customers [11]. YouTube product-DNN [5] uses the dataset that has a vocabulary of 1M videos and 1M search tokens. Moreover, TDM product-DNN [12, 13] uses datasets that consist of more than 8M user-book reviews from Amazon and more than 100M records of Taobao user behavior data. With the advent of such large scale datasets, training large DNNs has become exceptionally challenging. For instance, training BERT takes 3 days on 16 TPUv3 [9] and training PlaNet [14] even takes 2.5 months on 200 CPU cores using the DistBelief framework [15]. Thus, there is a growing interest in developing subsampling algorithms to downsize the data volume and accelerate training large DNNs.

Many approaches have been proposed for data reduction from a wide range of perspectives while preserving the performance as much as possible. Most of these existing approaches fall into one of the two broad categories: Randomized methods and Non-randomized methods. Although both categories construct the samples in a non-uniform data-dependent fasion [16], there is a key difference in the data they operate on. The randomized methods construct the samples by directly operating on a randomized sketch of the input covariate matrix [16–18]. In contrast, the non-randomized methods operate on a randomized sketch of both the input covariate matrix and the responses [19, 20]. These subsampling approaches

have been applied to matrix-related problems in large scale machine learning tasks, e.g., linear regression [18, 19, 21–25], logistic regression [20, 26, 27], and low-rank matrix approximation [28, 29]. We summarize the literature in Table 1.

Despite of these impressive algorithmic achievements in tradition machine learning tasks, none of these work on leveraging or leverage-based sampling demonstrates the capabilities to handle large-scale deep learning tasks. The main reason is that most of these methods are inflexible as they are derived only for specific linear, logistic regression models or low-rank approximations. In this paper, we bridge this gap by proposing a novel subsampling methods on approximating the full data empirical risk under the framework the minibatch stochastic gradient descent (Mini-SGD). Our method is motivated by minimizing the discrepancy between the true empirical risk on the entire dataset and the empirical risk on the sampled data.

Our contributions: In this paper, we have two major contributions for methodological developments in subsampling for solving large-scale deep learning. First, we propose the general optimization framework for data subsampling in any machine learning tasks. This is achieved by minimizing the discrepancy between the true empirical risk when training the model on the entire dataset and the empirical risk when training the model on the sampled data. Second, we develop approximation algorithms through two-step relaxations of the previous optimization problem under the framework of Mini-SGD. We conduct experiments on the synthetic linear regression experiments that provide insight, as well as on the MNIST and ImageNet datasets and show that our method can substantially improve the performance across different tasks given a fixed budget.

2 RELATED WORK

Importance sampling These methods are most closely related to our proposed techniques. The key idea behind these methods is to replace the uniform distribution used for sampling with a non-uniform distribution instead. Importance sampling has been used to accelerate the training of DNNs in various applications, such as image classification [37, 40], face recognition [41], and object detection [42, 43]. Specifically, we consider the Stochastic Gradient Descent (SGD) with importance sampling [32, 36, 39] in this paper.

We summarize the literature in Table 2. Among these paper, [38, 39] are most related to our work. Similar to ours, both approaches use the loss to construct the sampling distribution. [38] prioritizes samples with high loss at each iteration while [39] chooses the sample with lowest loss. However, approaches prioritizing samples with high loss are not robust to outliers while approaches using the samples with low loss often leads to low convergence rate and worse testing performance in practical applications. Although shown to be robust against outliers, the test accuracy of [39] are often inferior to other approaches in practical applications. In contrast, we develop algorithms choosing the subset of samples, the average loss of which best approximates that of the whole batch. We show that our approach achieves better balance in terms of robustness and convergence speed than the existing approaches.

Coresets selection Also related to our work is the problem of coresets selection since our algorithm consists of solving the coresets selection problem. This problem aims to select a subset of the full dataset such that the model trained on the selected subset will perform as closely as possible to the model trained on the entire dataset. Originating from computational geometry [46], the idea of coresets selection has been successfully employed to design various machine learning algorithms for, e.g., k-Means and k-Medians clustering [47–49], SVM [50], SVR [51], and logistic regression [27]. Most recently, algorithms based on coresets selection have also been proposed for CNN [52] and GAN [53].

Among those approaches that use coresets selection, most similar to ours are the batch active learning (AL) in [52, 54] and the bayesian coresets in [27, 55, 56] . [52, 54] formulate AL as a coresets selection problem. They choose a subset of unlabeled points to label such that a model learned over the selected subset is expected to yield competitive result over the whole dataset. Our algorithms consider a different setting where all the data is labeled. This is often the case in many real world applications, e.g., recommerder systems. [27, 55, 56] consider constructing Bayesian coresets for the Bayesian statistical models, attempting to select a small subset of the data to approximate the log-likelihood of the full data. Differently, we consider the general loss function under any empirical risk minimization framework. We consider our work to be complementary to the Bayesian coresets literature.

3 SUBSAMPLING FOR LARGE-SCALE DEEP LEARNING

3.1 Background

Let $\mathbf{x}_i, y_i$ be the i-th input-output pair from the training set, $y = g_\theta(\mathbf{x})$ be a Deep Learning model parameterized by the vector θ, and $\mathcal{L}(\mathbf{x}, y)$ be the loss function to be minimized during training. The goal of training is to find

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(g_\theta(\mathbf{x}_i), y_i\right), \tag{1}$$

where N corresponds to the number of examples in the training set.

Stochastic gradient descent (SGD) is a common procedure applied to solve (1). Specifically, SGD proceeds in a number of iterations, at each step selecting a single example i and updating the weights by subtracting the gradient of the loss multiplied by a the learning rate η.

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \mathcal{L}\left(g_{\theta_t}\left(\mathbf{x}_i\right), y_i\right). \tag{2}$$

In minibatch stochastic gradient descent (Mini-SGD), at each step, one selects a subset of examples $\left\{\left(\mathbf{x}_i^t, y_i^t\right)\right\}_{i=1}^{n_t}$, often by sampling from $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ at random without replacement, traversing the full training set once per epoch, applying the update

$$\theta_{t+1} = \theta_t - \eta_t \sum_{i=1}^{n_t} \nabla_\theta \mathcal{L}\left(g_{\theta_t}\left(\mathbf{x}_i^t\right), y_i^t\right). \tag{3}$$

Table 1: Classification of the research in data subsampling

	linear regression	logistic regression	low-rank approximation	DNN
Randomized	[16, 18, 22–24, 29]		[16, 28, 29]	
Non-Randomized	[19, 21], ours	[20, 26, 27], ours	ours	ours

Table 2: Gradien-based vs Loss-based importance sampling

	Convex Program	Deep Learning
Gradient Norm	[30–33]	[34–37]
Loss	[38, 39], ours	[38–45], ours

In the current Mini-SGD framework, all of $\left\{\left(\mathbf{x}_i^t, y_i^t\right)\right\}_{i=1}^{n_t}$ is used in the gradient procedure: the DNN model parameter θ is being updated using the entire batch data $\left\{\left(\mathbf{x}_i^t, y_i^t\right)\right\}_{i=1}^{n_t}$ iteration.

This batch contains n_t data points, but potentially, many of them will not contribute much to improving the model because the deep learning model begins to classify these examples accurately, especially redundant examples that are wellrepresented in the dataset. The main question is:

Can we only keep a (small) subset of $\left\{\left(\mathbf{x}_i^t, y_i^t\right)\right\}_{i=1}^{n_t}$ for training and throw away the rest? And how should we implement this scheme so that previous performance stays close to intact?

3.2 A general framework for data subsampling

Before discussing about selecting a subset of $\left\{\left(\mathbf{x}_i^t, y_i^t\right)\right\}_{i=1}^{n_t}$ for training in Mini-SGD, we propose a general framework for data subsampling in any machine learning tasks. Formally, denote C the training dataset. Denote $\mathcal{T}$ the testing dataset. Then, one way to formulate the data subsampling problem is as follows

$$
\begin{aligned}
\min_{z_i, \hat{\theta}_C, \theta_C^*} \quad & \| \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}} \mathcal{L}\left(g_{\theta_C^*}(\mathbf{x}_i), y_i\right) - \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}} \mathcal{L}\left(g_{\hat{\theta}_C}(\mathbf{x}_i), y_i\right) \|_2 \\
\text{s.t.} \quad & \theta_C^* \in \arg\min_{\theta} \sum_{(\mathbf{x}_i, y_i) \in C} \mathcal{L}\left(g_\theta\left(\mathbf{x}_i\right), y_i\right), \\
& \hat{\theta}_C \in \arg\min_{\theta} \sum_{(\mathbf{x}_i, y_i) \in C} z_i \cdot \mathcal{L}\left(g_\theta\left(\mathbf{x}_i\right), y_i\right), \\
& \sum_{i=1}^{|C|} z_i \leq K, \\
& z_i \in \{0, 1\}, \quad i = 1, \ldots, |C|,
\end{aligned}
\tag{4}
$$

where K is the sample size of the subset we hope to construct. The objective function seeks to measure the discrepancy between the true empirical risk when training the model on the entire dataset and the empirical risk when training the model on the sampled data. The first constraint restricts that θ_C^* is the optimal estimator when training on the entire dataset. The second constraint restricts that $\hat{\theta}_C$ is the optimal estimator when training on the sampled data. The third constraint restricts that at most K data points would be selected during the subsampling process. The last constraint restricts that each data point would be either selected or not selected, where $z_i = 1$ means selected, and not, otherwise.

Solving the formulation (4) yields the optimal sampling strategy regarding to achieve the best empirical risk obtainable. However, note that (4) involves many indicator variables and has non-convex objective function and constraints, making it a very complex combinatorial optimization problem and thus NP-hard to solve. Although state-of-art mixed integer non-convex algorithms might solve (4) to optimal, it is extremely time-consuming to achieve this and might take much more efforts than solving (1) itself. Hence, the general framework is not directly applicable in subsampling the data for large-scale deep learning.

3.3 One backward from ten forward for deep learning

Due to challenges of solving (4) directly, we seek to propose approximation algorithms through two-step relaxations under the framework of Mini-SGD.

First, given the data $\left\{\left(\mathbf{x}_i^t, y_i^t\right)\right\}_{i=1}^{n_t}$ at batch t, we approximate the true empirical risk on the whole dataset in (4) by the empirical risk on the batch data:

$$
\frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}} \mathcal{L}\left(g_{\theta_C^*}(\mathbf{x}_i), y_i\right) \approx \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{L}\left(g_{\theta_{t-1}}(\mathbf{x}_i^t), y_i^t\right).
$$

Second, we approximate the empirical risk on the selected data in (4) by the empirical risk on the selected data in batch t:

$$
\frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}} \mathcal{L}\left(g_{\hat{\theta}_C}(\mathbf{x}_i), y_i\right) \approx \frac{1}{b} \sum_{i=1}^{n_t} z_i^t \cdot \mathcal{L}\left(g_{\theta_{t-1}}(\mathbf{x}_i^t), y_i^t\right).
$$

Here, b is the number of data points we are allowed to sample within a batch.

Let $\bar{l}^t = \frac{1}{n_t} \sum_{i=1}^{n_t} l_{\theta_{t-1}}(\mathbf{x}_i, y_i)$ be the average loss for the t-batch data using θ_{t-1}. Then, we convert the problem of subsampling data points from the entire dataset into the problem of subsampling data points from each batch. Now, one key step of subsampling the training data in the current batch can be formulated as

$$
\begin{aligned}
\min_{z_i^t} \quad & \| \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{L}\left(g_{\theta_{t-1}}(\mathbf{x}_i^t), y_i^t\right) - \frac{1}{b} \sum_{i=1}^{n_t} z_i^t \cdot \mathcal{L}\left(g_{\theta_{t-1}}(\mathbf{x}_i^t), y_i^t\right) \|_2 \\
\text{s.t.} \quad & \sum_{i=1}^{n_t} z_i^t \leq b, \\
& z_i^t \in \{0, 1\}, \quad i = 1, \ldots, n_t
\end{aligned}
\tag{5}
$$

where b is the size of the data we hope to sample from the batch of training data $\left\{\left(x_i^t, y_i^t\right)\right\}_{i=1}^{n_t}$. (5) is a sparse subset approximation problem.

Our algorithm for solving large-scale deep learning problem with subsampling of the data points iterates as in Algorithm 1.

Algorithm 1 One Backward from Ten Forward for deep learning (OBFTF)

1: **Input:** batch data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n_t}$ for $t = 1, 2, \ldots$, and a budget b.
2: **Initialization:** θ_0 could be an arbitrary hypothesis of the parameter.
3: **for** $t = 1, 2, \ldots$ **do**
4: **Forward propagate** and compute $g_{\theta 4pt_{t-14pt}}(\mathbf{x}_i^t)$ for $i = 1, \ldots, n_t$
5: Compute loss $\mathcal{L}\left(g_{\theta_{t-1}}(\mathbf{x}_i^t), y_i^t\right)$ for $i = 1, \ldots, n_t$
6: Solve (5), get z_i^t for $i = 1, \ldots, n_t$
7: Keep $(\mathbf{x}_i, y_i)$ if $z_i^t = 1$
8: **Back propagate** and train the model using the selected data, get θ_t
9: **end for**

Although (5) is still a combinatorial optimization problem, it is much easier to solve than (4) and there exists efficient approximation algorithms, such as Frank-Wolfe. For the current paper, the combinatorial problem is solved to optimal using state-of-art solver to fully illustrate the performance of Algorithm 1. In future, we shall develop fast and accurate algorithms to solve the sparse subset approximation problem.

4 EXPERIMENTS

We test the objectives on the following datasets: (1) Synthetic dataset for linear regression, (2) MNIST, and (3) ImageNet. Detailed descriptions are in the supplementary material. Code will be open-sourced upon acceptance of the manuscript.

To evaluate the performance of the proposed framework, we compare with the following methods:

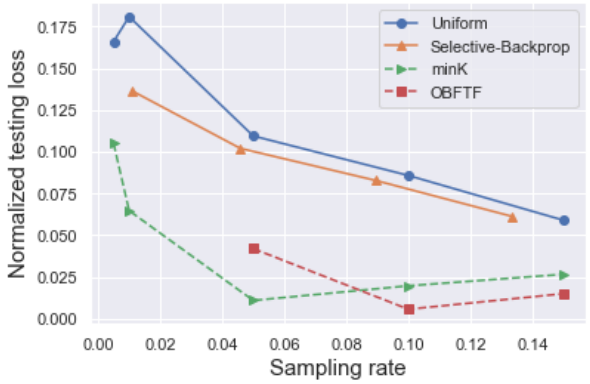
- **Uniform Sampling (Uniform).** Let $\pi_i = 1/n$, i.e., draw the subsample uniformly at random at random.
- **Selective-Backprop.** In each iteration, select the sample with the probability that is proportitial to the current loss [38].
- **Min-k Loss SGD (minK)** [39]. Choosing the subsample with lowest loss.

4.1 Synthetic data

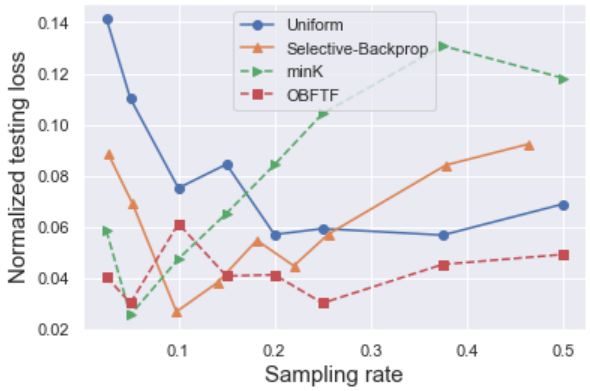
Simulated data: $y = 2x + 1 + U(-5, 5)$, 1000 training data, 10000 testing data.

Simulated data with outlier : $y = 2x + 1 + U(-5, 5)$ ($+U(-20, 20)$ for 20 data points), 1000 training data, 10000 testing data.

Results For data without outliers, we train the models with relatively small sampling rate (smaller than 0.15). minK is the most compatitive method in this case. It outperforms other methods with the sampling rate is smaller than 0.05. While OBFTF performs the best when the sampling rate is between 0.1 and 0.15. For data with outliers, we train the models with a variety of sampling rates between 0.01 and 0.5. When the sampling rate is smaller than 0.15, minK and selective-backprop are comparable to OBFTF. Their performance, however, is unstable that a slight increase or decrease in the sampling rate causes a significant drop off in the normalized testing loss. OBFTF performs stable within the sampling rate range, and outperforms other methods when the sampling rate is between 0.15 and 0.5.



(a) Data without outliers



(b) Data with outliers

Figure 1: Performance of the sampling algorithms for linear regression.

4.2 MNIST

We perform a classification task on the MNIST dataset [57], which contains 70,000 gray scale images of numerical digits from 0 to 9, divided as 60,000 training images and 10,000 test images. We do not apply any preprocessing to this dataset and only compare models without data augmentation.

Training settings All the models are trained for 500 epoches with the following settings: initial learning rate is 0.1, batch size is 128, two hidden layers and both of them have 256 neurons.

Results We compare OBFTF with the other methods under a variety of sampling ratios. As shown in Figure 2, OBFTF achieves higher accuracy than other methods when the sampling rate is small (0.1 to 0.25), indicating its benefits in speeding up the training phase of classification problems by using a small sampling rate. When the sampling rate increases (to 0.5), the difference of the performance of all these methods are insignificant. We note that the accuracy of

Table 3: Performance of the sampling algorithms on ImageNet 2012 `Val` set. ResNet50 and MobileNetV2 are used as baseline models.

Model	Method	0.10	0.15	0.20	0.25	0.30	0.45
ResNet50	Uniform sampling	0.7074	0.7086	0.7316	0.7391	0.7313	0.7430
	Selective-Backprop.	0.2551	0.2986	0.3699	0.3939	0.4431	0.4770
	Ours	0.7096	0.7113	0.7355	0.7439	0.7303	0.7452
MobileNetV2	Uniform sampling	0.6922	0.7065	0.7143	0.7196	0.7242	0.7279
	Selective-Backprop.	0.6164	0.6572	0.6654	0.6700	0.6795	0.6916
	Ours	0.6956	0.7102	0.7167	0.7198	0.7242	0.7283

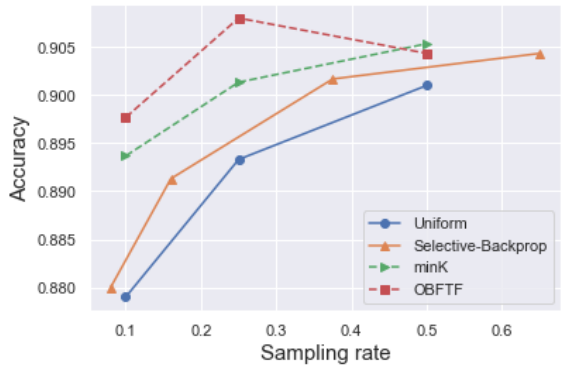


Figure 2: Performance of the sampling algorithms for MNIST.

OBFTF with 0.25 sample rate is higher than the accuracies of all of the methods with 0.5 sample rate! It demonstrates the effectiness of our method in classification tasks, and the fact that a small sampling rate may achieve the same or even better accuracy than a big sampling rate.

4.3 ImageNet

To further evaluate our method on large-scale datasets, we perform a much more challenging image classification task on 1000-class ImageNet dataset [8], which contains about 1.2 million training images, 50,000 validation images and 100,000 test images. To verify the effectiveness of our method on different types of neural networks, we choose the popular ResNet50 [2] and MobileNetV2 [58] as baseline models, both of which achieves state-of-the-art results on ImageNet. Compared to ResNet50 which has a higher accuracy, MobileNetV2 is advantageous with higher computational efficiency, thus more friendly to mobile devices, e.g. cell phones.

Training settings Following the training schedule in MNasNet [59], we train the baseline models using the synchronous training setup on 32 Tesla-V100-SXM2-16GB GPUs. The initial learning rate is set to be 0.016, and the overall batch size is 4096 (128 images per GPU). The learning rate linearly increases to 0.256 in the first 5 epochs and then is decayed by 0.97 every 2.4 epochs. We use a dropout of 0.2, a weight decay of 1e−5 and Inception image preprocessing [60] of size 224×224. We also use exponential moving average on model weights with a momentum of 0.9999. All batch normalization layers use a momentum of 0.99. Using above settings, we train ResNet50/MobileNetV2 for 150/350 epoches respectively.

Results On both ResNet50 and MobileNetV2, we compare our method with the uniform sampling and Selective-Backprop under a variety of sampling ratios, i.e. $[0.1, 0.15, 0.20, 0.25, 0.30, 0.45]$. We do not report the results of minK as it does not yield comparable results at all. As can be seen from Table 3, on both baseline models, our method achieves higher accuracy than the uniform sampling and Selective-Backprop in this challenging task. Particually when the sampling rate is small e.g. ranging from 0.10 to 0.25, our method has more obvious advantage over the counterpart. This result suggests our method can remarkably benefit training on large-scale datasets by using a small sampling rate to speedup the training phase. When the sampling rate increases, the margin shrinks which may be due to the fact that as sampled out data becomes more representative of the full-sized dataset, the weights of models trained using compared sampling methods receive more accurate gradient updates in each iteration. Above results demonstrates the effectiness of our method in large-scale machine learning tasks.

5 CONCLUSION

We consider accelerating training deep learning models in large-scale ML systems. We leverage the key insight that these deployed ML systems continuously perform forward passes on data instances during inference, but ad-hoc sampling does not take advantage of this substantial computational effort. Therefore, we propose to record a constant amount of information per instance from these forward passes. The extra information measurably improves the selection of which data instances should participate in forward and backward passes. A novel optimization framework is proposed to analyze this problem and we provide an efficient approximation algorithm under the framework of minibatch SGD as a practical solution. We demonstrate the effectiveness of our framework and algorithm on several large-scale classification and regression tasks.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *ICCV*, 2016.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, 2016.

[6] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[7] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I. Nikolenko. Recvae: a new variational autoencoder for top-n recommendations with implicit feedback. In *WSDM*, 2020.

[11] James Bennett, Stan Lanning, et al. The netflix prize. In *KDD Cup and Workshop*, 2007.

[12] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. Learning tree-based deep model for recommender systems. In *KDD*, 2018.

[13] Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. Joint optimization of tree-based index and deep model for recommender systems. In *NeurIPS*, 2019.

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[15] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. In *NIPS*, 2012.

[16] Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[17] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012.

[18] Ping Ma, Michael Mahoney, and Bin Yu. A statistical perspective on algorithmic leveraging. In *ICML*, 2014.

[19] Brian McWilliams, Gabriel Krummenacher, Mario Lucic, and Joachim M Buhmann. Fast and robust least squares estimation in corrupted linear models. In *NIPS*, 2014.

[20] HaiYing Wang, Rong Zhu, and Ping Ma. Optimal subsampling for large sample logistic regression. *Journal of the American Statistical Association*, 113(522):829–844, 2018.

[21] Petros Drineas, Michael W Mahoney, and Shan Muthukrishnan. Sampling algorithms for l 2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136. Society for Industrial and Applied Mathematics, 2006.

[22] Petros Drineas, Michael W Mahoney, Shan Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische mathematik*, 117(2):219–249, 2011.

[23] Ping Ma, Michael W Mahoney, and Bin Yu. A statistical perspective on algorithmic leveraging. *The Journal of Machine Learning Research*, 16(1):861–911, 2015.

[24] Ping Ma and Xiaoxiao Sun. Leveraging for big data regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(1):70–76, 2015.

[25] Rong Zhu, Ping Ma, Michael W Mahoney, and Bin Yu. Optimal subsampling approaches for large sample linear regression. *arXiv preprint arXiv:1509.05111*, 2015.

[26] William Fithian and Trevor Hastie. Local case-control sampling: Efficient subsampling in imbalanced data sets. *Annals of statistics*, 42(5):1693, 2014.

[27] Jonathan Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression. In *NIPS*, 2016.

[28] Michael W Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[29] Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

[30] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 2005.

[31] Siddharth Gopal. Adaptive sampling for sgd by exploiting side information. In *ICML*, 2016.

[32] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, 2015.

[33] Beidi Chen, Yingchen Xu, and Anshumali Shrivastava. Lsh-sampling breaks the computation chicken-and-egg loop in adaptive stochastic gradient estimation. In *NeurIPS*, 2019.

[34] 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.

[35] Kailas Vodrahalli, Ke Li, and Jitendra Malik. Are all training examples created equal? an empirical study. *arXiv preprint arXiv:1811.12569*, 2018.

[36] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *ICML*, 2018.

[37] Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In *NeurIPS*, 2018.

[38] 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.

[39] Vatsal Shah, Xiaoxia Wu, and Sujay Sanghavi. Choosing the sample with lowest loss makes sgd robust. In *AISTATS*, 2020.

[40] Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[41] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *ICCV*, 2015.

[42] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[43] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *ICCV*, 2016.

[44] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015.

[45] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2016.

[46] Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

[47] Mihai Bādoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 250–257, 2002.

[48] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM, 2004.

[49] Vladimir Braverman, Lingxiao Huang, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in graphs of bounded treewidths. In *ICML*, 2020.

[50] Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392, 2005.

[51] Ivor W Tsang, James T Kwok, and Kimo T Lai. Core vector regression for very large regression problems. In *ICML*, 2005.

[52] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018.

[53] Samarth Sinha, Han Zhang, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, and Augustus Odena. Small-gan: Speeding up gan training using core-sets. In *ICML*, 2020.

[54] Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. In *NeurIPS*, 2019.

[55] Trevor Campbell and Tamara Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. In *ICML*, 2018.

[56] Trevor Campbell and Tamara Broderick. Automated scalable bayesian inference via hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588, 2019.

[57] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

[58] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[59] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

[60] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.