Multiple Importance Sampling for Stochastic Gradient Estimation

Corentin Salaün[1][a] Xingchang Huang[1][b] Iliyan Georgiev[2][c] Niloy Mitra[2,3][d] Gurprit Singh[1][e]
[1]*Max Planck Institute for Informatics, Saarland University, Saarbrücken, Germany*
[2]*Adobe Research, London, United Kingdom*
[3]*Department of Computer Science, University College London, London, United Kingdom*
{*csalaun,xhuang,gsingh*}*@mpi-inf.mpg.de, igeorgiev@adobe.com, n.mitra@cs.ucl.ac.uk*

Keywords: Optimization, Machine Learning, Gradient Estimation

Abstract: We introduce a theoretical and practical framework for efficient importance sampling of mini-batch samples for gradient estimation from single and multiple probability distributions. To handle noisy gradients, our framework dynamically evolves the importance distribution during training by utilizing a self-adaptive metric. Our framework combines multiple, diverse sampling distributions, each tailored to specific parameter gradients. This approach facilitates the importance sampling of *vector-valued* gradient estimation. Rather than naively combining multiple distributions, our framework involves optimally weighting data contribution across multiple distributions. This adapted combination of multiple importance yields superior gradient estimates, leading to faster training convergence. We demonstrate the effectiveness of our approach through empirical evaluations across a range of optimization tasks like classification and regression on both image and point cloud datasets.

1 Introduction

Stochastic gradient descent (SGD) is fundamental in optimizing complex neural networks. This iterative optimization process relies on the efficient estimation of gradients to update model parameters and minimize the optimization objective. A significant challenge in methods based on SGD lies in the influence of stochasticity on gradient estimation, impacting both the quality of the estimates and convergence speed. This stochasticity introduces errors in the form of noise, and addressing and minimizing such noise in gradient estimation continues to be an active area of research.

Various approaches have been introduced to reduce gradient estimation noise, including data diversification Zhang et al. (2019); Faghri et al. (2020); Ren et al. (2019), adaptive mini-batch sizes Balles et al. (2017); Alfarra et al. (2021), momentum-based estimation Rumelhart et al. (1986); Kingma and Ba (2014), and adaptive sampling strategies Santiago et al. (2021). These methods collectively expedite the optimization by improving the gradient-estimation accuracy.

Another well-established technique for noise reduction in estimation is importance sampling (IS) Loshchilov and Hutter (2015); Katharopoulos and Fleuret (2017, 2018), which involves the non-uniform selection of data samples for mini-batch construction. Data samples that contribute more significantly to gradient estimation are selected more often. This allows computational resources to focus on the most critical data for the optimization task. However, these algorithms are quite inefficient and add significant overhead to the training process. Another limitation of importance sampling, in general, lies in determining the best sampling distribution to achieve maximal improvement, often necessitating a quality tradeoff due to the simultaneous estimation of numerous parameters.

We propose an efficient importance sampling algorithm that does *not* require resampling, in contrast to Katharopoulos and Fleuret (2018). Our importance function dynamically evolves during training, utilizing a self-adaptive metric to effectively manage initial noisy gradients. Further, unlike existing IS methods in machine learning where importance distributions assume scalar-valued gradients, we propose a

[a] https://orcid.org/0000-0002-5112-7488
[b] https://orcid.org/0000-0002-2769-8408
[c] https://orcid.org/0000-0002-9655-2138
[d] https://orcid.org/0000-0002-2597-0914
[e] https://orcid.org/0000-0003-0970-5835

multiple importance sampling (MIS) strategy to manage *vector-valued* gradient estimation (i.e., multiple parameters). We propose the simultaneous use of multiple sampling strategies combined with a weighting approach following the principles of MIS theory, well studied in the rendering literature in computer graphics Veach (1997). Rather than naively combining multiple distributions, our proposal involves estimating importance weights w.r.t. data samples across multiple distributions by leveraging the theory of optimal MIS (OMIS) Kondapaneni et al. (2019). This optimization process yields superior gradient estimates, leading to faster training convergence. In summary, we make the following contributions:

- An efficient IS algorithm with a self-adaptive metric for importance sampling is developed.
- An MIS estimator for gradient estimation is introduced to improve gradients estimation.
- A practical approach to computing the OMIS weights is presented to maximize the quality of vector-valued gradient estimation.
- The effectiveness of the approach is demonstrated on various machine learning tasks.

2 Related work

Importance sampling for gradient estimation. Importance sampling (IS) Kahn (1950); Kahn and Marshall (1953); Owen and Zhou (2000) has emerged as a powerful technique in high energy physics, Bayesian inference, rare event simulation for finance and insurance, and rendering in computer graphics. In the past few years, IS has also been applied in machine learning to improve the accuracy of gradient estimation and enhance the overall performance of learning algorithms Zhao and Zhang (2015).

By strategically sampling data points from a non-uniform distribution, IS effectively focuses training resources on the most informative and impactful data, leading to more accurate gradient estimates. Bordes et al. (2005) developed an online algorithm (LASVM) that uses importance sampling to train kernelized support vector machines. Loshchilov and Hutter (2015) suggested employing data rankings based on their respective loss values. This ranking is then employed to create an importance sampling strategy that assigns greater importance to data with higher loss values. Katharopoulos and Fleuret (2017) proposed importance sampling the loss function. Subsequently, Katharopoulos and Fleuret (2018) introduced an upper bound to the gradient norm that can be employed as an importance function. Their algorithm involves resampling and computing gradients with respect to the final layer. Despite the importance function demonstrating improvement over uniform sampling, their algorithm exhibits significant inefficiency.

Multiple importance sampling. The concept of Multiple Importance Sampling (MIS) emerged as a robust and efficient technique for integrating multiple sampling strategies Owen and Zhou (2000). Its core principle lies in assigning weights to multiple importance sampling estimator, each using a different sampling distribution, allowing each data sample to utilize the most appropriate strategy. Veach (1997) introduced this concept of MIS to rendering in computer graphics and proposed the widely adopted *balance heuristic* for importance (weight) allocation. The balance heuristic determines weights based on a data sample's relative importance across all sampling approaches, effectively mitigating the influence of outliers with low probability densities. While MIS is straightforward to implement and independent of the specific function, Variance-Aware MIS Grittmann et al. (2019) advanced the concept by using variance estimates from each sampling technique for further error reduction. Moreover, Optimal MIS Kondapaneni et al. (2019) derived optimal sampling weights that minimize MIS estimator variance. Notably, these weights depend not only on probability density but also on the function values of the samples. Appendix B summarizes the theory behind (multiple) importance sampling. It also states the optimal MIS estimator and how to compute it.

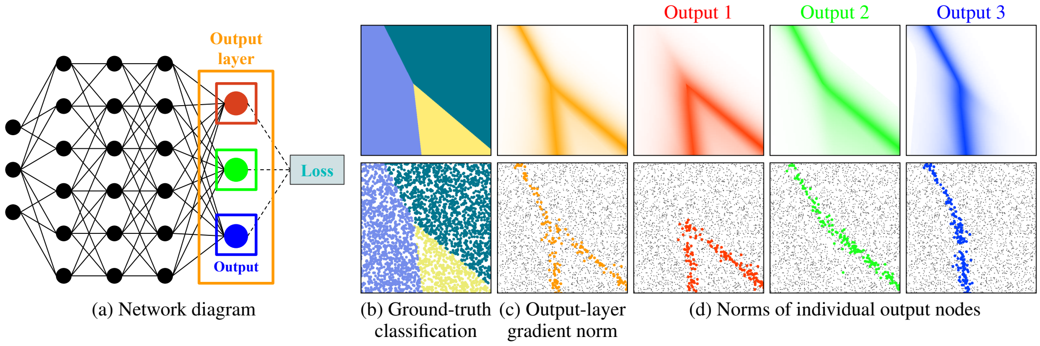
3 Problem statement

The primary goal of machine-learning optimization is to find the optimal parameters θ for a given model function $m(x,\theta)$ by minimizing a loss function $\mathcal{L}$ over a dataset Ω:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \underbrace{\int_\Omega \mathcal{L}(m(x_i,\theta),y)\,\mathrm{d}x}_{L_\theta}. \tag{1}$$

The loss function $\mathcal{L}$ quantifies the dissimilarity between the model predictions $m(x,\theta)$ and observed data y. In the common case of a discrete dataset, the integral becomes a sum.

In practice, the total loss is minimized via iterative gradient descent. In each iteration t, the gradient ∇L_{θ_t} of the loss with respect to the current model parameters θ_t is computed, and the parameters are updated



(a) Network diagram (b) Ground-truth classification (c) Output-layer gradient norm (d) Norms of individual output nodes

Figure 1: We visualize different importance sampling distributions for a simple classification task. We propose to use the output layer gradients for importance sampling, as shown in the network diagram (a). For a given ground-truth classification (top) and training dataset (bottom) shown in (b), it is possible to importance sample from the L_2 norm of the output-layer gradients (c) or from three different sampling distributions derived from the gradient norms of individual output nodes (d). The bottom row shows sample weights from each distribution.

as

$$\theta_{t+1} = \theta_t - \lambda \underbrace{\int_\Omega \nabla \mathcal{L}(m(x,\theta_t),y)\,\mathrm{d}x}_{\nabla L_{\theta_t}}, \tag{2}$$

where $\lambda > 0$ is the learning rate. It is also possible to use an adaptive learning rate instead of a constant.

Monte Carlo gradient estimator. In practice, the parameter gradient is estimated from a small batch $\{x_i\}_{i=1}^B$ of randomly selected data points:

$$\langle \nabla L_\theta \rangle = \sum_{i=1}^{B} \frac{\nabla \mathcal{L}(m(x_i,\theta),y_i)}{Bp(x_i)} \approx \nabla L_\theta, \quad x_i \sim p. \tag{3}$$

The data points are sampled from a probability density function (pdf) p or probability mass function in discrete cases. The mini-batch gradient descent substitutes the true gradient ∇L_{θ_t} with an estimate $\langle \nabla L_{\theta_t} \rangle$ in Eq. (2) to update the model parameters in each iteration.

We want to estimate ∇L_{θ_t} accurately and also efficiently, since the gradient-descent iteration (2) may require many thousands of iterations until the parameters converge. These goals can be achieved by performing the optimization in small batches whose samples are chosen according to a carefully designed distribution p. For a simple classification problem, Fig. 1c shows an example importance sampling distribution derived from the output layer of the model. In Fig. 1d we derive multiple distributions from the individual output nodes. Below we develop theory and practical algorithms for importance sampling using a single distribution (Section 4) and for combining multiple distributions to further improve gradient estimation (Section 5).

4 Mini-batch importance sampling

Mini-batch gradient estimation (3) notoriously suffers from Monte Carlo noise, which can make the parameter-optimization trajectory erratic and convergence slow. That noise comes from the often vastly different contributions of different samples x_i to that estimate.

Typically, the selection of the multiple samples constructing a mini-batch is done with uniform probability $p(x_i) = 1/|\Omega|$. Each data of the mini-batch is sampled with replacement following this distribution. Importance sampling is a technique for using a non-uniform pdf to strategically pick samples proportionally on their contribution to the gradient, to reduce estimation variance.

Practical algorithm. We propose an importance sampling algorithm for mini-batch gradient descent, outlined in Algorithm 1. Similarly to Schaul et al. (2015), we use an importance function that relies on readily available quantities for each data point, introducing only negligible memory and computational overhead over classical uniform mini-batching. We store a set of persistent *un-normalized importance* scalars $q = \{q_i\}_{i=1}^{|\Omega|}$ that are updated continuously during the optimization.

The first epoch is a standard SGD one, during which we additionally compute the initial importance of each data point (line 3). In each subsequent epoch, at each mini-batch optimization step t we normalize the importance values to a valid distribution p (line 6). We then choose B data samples (with replacement) according to p (line 7). The loss $\mathcal{L}$ is evaluated for each selected data sample (line 8), and backpropagated to compute the loss gradient (line 9). The per-

Algorithm 1 Mini-batch importance sampling for SGD.

```
1:  $\theta \leftarrow$ random parameter initialization
2:  $B \leftarrow$ mini-batch size, $N = |\Omega|$                      $\leftarrow$ Dataset size
3:  $q, \theta \leftarrow \text{Initialize}(x, y, \Omega, \theta, B)$          $\leftarrow$ Algorithm 4
4:  **until** convergence **do**                               $\leftarrow$ Loop over epochs
5:      **for** $t \leftarrow 1$ **to** $N/B$                        $\leftarrow$ Loop over mini-batches
6:          $p \leftarrow q/\text{sum}(q)$                  $\leftarrow$ Normalize importance to pdf
7:          $x, y \leftarrow B$ data samples $\{x_i, y_i\}_{i=1}^{B} \propto p$
8:          $\mathcal{L}(x) \leftarrow \mathcal{L}(m(x,\theta), y)$
9:          $\nabla \mathcal{L}(x) \leftarrow \text{Backpropagate}(\mathcal{L}(x))$
10:         $\langle \nabla L_\theta \rangle \leftarrow (\nabla \mathcal{L}(x) \cdot (1/p(x))^T)/B$       $\leftarrow$ Eq. (3)
11:         $\theta \leftarrow \theta - \lambda \langle \nabla L_\theta \rangle$                    $\leftarrow$ SGD step
12:         $q(x) \leftarrow \alpha \cdot q(x) + (1-\alpha) \cdot \left\| \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)} \right\|$
13:
14:     $q \leftarrow q + \varepsilon$                          ↼ Accumulate importance
15: **return** $\theta$
```

sample importance is used in the gradient estimation (line 10) to normalize the contribution. In practice lines 9-10 can be done simultaneously by backpropagating a weighted loss $\mathcal{L}(x) \cdot (1/(p(x) \cdot B))^T$. Finally, the network parameters are updated using the estimated gradient (line 11). On line 12, we update the importance of the samples in the mini-batch; we describe our choice of importance function below. The blending parameter α ensures stability of the persistent importance as discussed in Appendix E. At the end of each epoch (line 14), we add a small value to the unnormalized weights of all data to ensure that every data point will be eventually evaluated, even if its importance is deemed low by the importance metric.

It is important to note that the first epoch is done without importance sampling to initialize each sample importance. This does not add overhead as it is equivalent to a classical epoch running over all data samples. While similar schemes have been proposed in the past Loshchilov and Hutter (2015), they often rely on a multitude of hyperparameters, making their practical implementation challenging. This has led to the development of alternative methods like re-sampling Katharopoulos and Fleuret (2018); Dong et al. (2021); Zhang et al. (2023). Tracking importance across batches and epochs minimizes the computational overhead, further enhancing the efficiency and practicality of the approach.

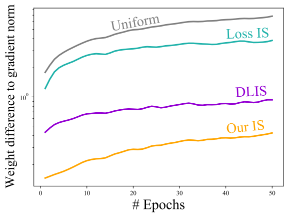
Importance function. In combination with the presented algorithm, we propose an importance function that is efficient to evaluate. While the gradient L_2 norm has been shown to be optimal Zhao and Zhang (2015); Needell et al. (2014); Wang et al. (2017); Alain et al. (2015), calculating it can be computationally expensive as it requires full backpropagation for every data point. To this end, we compute the gradient norm only for a subset of the parameters, specifically the output nodes of the network: $q(x) = \left\| \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)} \right\|$. This choice is based on an upper bound of the gradient norm, using the chain rule and the Cauchy–Schwarz inequality Katharopoulos and Fleuret (2018):

$$\left\| \frac{\partial \mathcal{L}(x_i)}{\partial \theta} \right\| = \left\| \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)} \cdot \frac{\partial m(x,\theta)}{\partial \theta} \right\| \leq \qquad (4)$$

$$\left\| \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)} \right\| \cdot \left\| \frac{\partial m(x,\theta)}{\partial \theta} \right\| \leq \underbrace{\left\| \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)} \right\|}_{q(x)} \cdot C,$$

where C is the Lipschitz constant of the parameters gradient. That is, our importance function is a bound of the gradient magnitude based on the output-layer gradient norm.

We tested the relationship between four different importance distributions: uniform, our proposed importance function, the loss function as importance Katharopoulos and Fleuret (2017), and the work by Katharopoulos and Fleuret (2018) using an other gradient norm bound. The inline figure plots the L_2 difference between these importance distributions and the ground-truth gradient-norm distribution across epochs for an MNIST classification task. It shows that Our IS distribution has the smallest difference, i.e., it achieves high accuracy while requiring only a small part of the gradient.



For some specific task when the output layer has predictable shape, it is possible to derive a closed form definition of the proposed importance metric. Appendix D derives the close form importance for classification task using cross entropy loss.

Note that any importance heuristic can be used on line 12 of Algorithm 1, such as the gradient norm Zhao and Zhang (2015); Needell et al. (2014); Wang et al. (2017); Alain et al. (2015), the loss Loshchilov and Hutter (2015); Katharopoulos and Fleuret (2017); Dong et al. (2021), or more advanced importance Katharopoulos and Fleuret (2018). For efficiency, our importance function reuses the forward-pass computations from line 8, updating q only for the current mini-batch samples.

5 Multiple importance sampling

The parameter gradient ∇L_θ is vector with dimension equal to the number of model parameters. The individual parameter derivatives vary uniquely across the data points, and estimation using a single distribution (Section 4) inevitably requires making a trade-off, e.g., only importance sampling the overall gradient magnitude. Truly minimizing the estimation error requires estimating each derivative using a separate importance sampling distribution tailored to its variation. However, there are two practical issues with this approach: First, it would necessitate sampling from all of these distributions, requiring "mini-batches" of size equal at least to the number of parameters. Second, it would lead to significant computation waste, since backpropagation computes all parameter derivatives but only one of them would be used per data sample. To address this issue, we propose using a small number of distributions, each tailored to the variation of a parameter subset, and combining *all* computed derivatives into a low-variance estimator, using multiple importance sampling theory. As an example, Fig. 1d shows three sampling distributions for a simple classification task, based on the derivatives of the network's output nodes, following the boundary of each class.

MIS gradient estimator. Combining multiple sampling distributions into a single robust estimator has been well studied in the Monte Carlo rendering literature. The best known method is *multiple importance sampling* (MIS) Veach (1997). In our case of gradient estimation, the MIS estimator takes for form

$$\langle \nabla L_\theta \rangle_{\mathrm{MIS}} = \sum_{j=1}^{J} \sum_{i=1}^{n_j} w_j(x_{ij}) \frac{\nabla \mathcal{L}(m(x_{ij}, \theta), y_{ij})}{n_j p_j(x_{ij})}, \quad (5)$$

where J is the number of sampling distributions, n_j the number of samples from distribution j, and x_{ij} the i^{th} sample from the j^{th} distribution. Each sample is modulated by a weight $w_j(x_{ij})$; the estimator is unbiased as long as $\sum_{j=1}^{J} w_j(x) = 1$ for every data point x in the dataset.

Optimal weighting. Various MIS weighting functions w_j have been proposed in literature, the most universally used one being the balance heuristic Veach (1997). In this work we use the recently derived optimal weighting scheme Kondapaneni et al. (2019) which minimizes the estimation variance for a given set of sampling distributions p_j:

$$w_j(x) = \alpha_j \frac{p_j(x)}{\nabla \mathcal{L}(m(x,\theta), y)} + \frac{n_j p_j(x)}{\sum_{k=1}^{J} n_k p_k(x)} \left(1 - \frac{\sum_{k=1}^{J} \alpha_k p_k(x)}{\nabla \mathcal{L}(m(x,\theta), y)} \right). \quad (6)$$

Here, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_J]$ is the solution to the linear system

$$\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{b}, \text{ with } \begin{cases} a_{j,k} = \int_\Omega \frac{p_j(x) p_k(x)}{\sum_i^J n_i p_i(x)} d(x,y), \\ b_j = \int_\Omega \frac{p_j(x) \nabla \mathcal{L}(m(x,\theta), y)}{\sum_i^J n_i p_i(x)} d(x,y), \end{cases} \quad (7)$$

where $a_{j,k}$ and b_j are the elements of the matrix $\boldsymbol{A} \in \mathbb{R}^{J \times J}$ and vector $\boldsymbol{b} \in \mathbb{R}^J$ respectively.

Instead of explicitly computing the optimal weights in Eq. (6) using Eq. (7) and plugging them into the MIS estimator (5), we can use a shortcut evaluation that yields the same result Kondapaneni et al. (2019):

$$\langle \nabla L_\theta \rangle_{\mathrm{OMIS}} = \sum_{j=1}^{J} \alpha_j. \quad (8)$$

In Appendix B we provide an overview of MIS and the aforementioned weighting schemes. Importantly for our case, the widely adopted balance heuristic does not bring practical advantage over single-distribution importance sampling (Section 4) as it is equivalent to sampling from a mixture of the given distributions; we can easily sample from this mixture by explicitly averaging the distributions into a single one. In contrast, the optimal weights are different for each gradient dimension as they depend on the gradient value.

Practical algorithm. Implementing the optimal-MIS estimator (8) amounts to drawing n_j samples from each distribution, computing $\boldsymbol{\alpha}$ for each dimension of the gradient and summing its elements. The integrals in $\boldsymbol{A}$ and $\boldsymbol{b}$ (sums in the discrete-dataset case) can be estimated as $\langle \boldsymbol{A} \rangle$ and $\langle \boldsymbol{b} \rangle$ from the drawn samples, yielding the estimate $\langle \boldsymbol{\alpha} \rangle = \langle \boldsymbol{A} \rangle^{-1} \langle \boldsymbol{b} \rangle$.

Algorithm 2 shows a complete gradient-descent algorithm. The main differences with Algorithm 1 are the use of multiple importance distributions $\boldsymbol{q} = \{q_j\}_{j=1}^{J}$ (line 5) and the linear system used to compute the OMIS estimator (line 6). This linear system is updated (lines 15-18) using the mini-batch samples and solved to obtain the gradient estimation (line 22). Since the matrix $\langle \boldsymbol{A} \rangle$ is independent of the gradient estimation (see Eq. (7)), its inversion can be shared across all parameter estimates.

Algorithm 2 Optimal multiple importance sampling SGD.

```
1:  $\theta \leftarrow$ random parameter initialization
2:  $B \leftarrow$ mini-batch size, $J \leftarrow$ number of pdf
3:  $N = |\Omega| \leftarrow$ dataset size
4:  $n_j \leftarrow$ sample count per technique, for $j \in \{1,..J\}$
5:  $\boldsymbol{q}, \theta \leftarrow \text{InitializeMIS}(x, y, \Omega, \theta, B)$            $\leftarrow$ Algorithm 5
6:  $\langle \boldsymbol{A} \rangle \leftarrow 0^{J \times J}, \langle \boldsymbol{b} \rangle \leftarrow 0^J$            $\leftarrow$ OMIS linear system
7:  until convergence do            $\leftarrow$ Loop over epochs
8:      for $t \leftarrow 1$ to $N/B$            $\leftarrow$ Loop over mini-batches
9:          $\langle \boldsymbol{A} \rangle \leftarrow \beta \langle \boldsymbol{A} \rangle, \langle \boldsymbol{b} \rangle \leftarrow \beta \langle \boldsymbol{b} \rangle$
10:         for $j \leftarrow 1$ to $J$            $\leftarrow$ Loop over distributions
11:             $p_j \leftarrow q_j / \text{sum}(q_j)$
12:             $x, y \leftarrow B$ data samples $\{x_i, y_i\}_{i=1}^{n_j} \propto p_j$
13:             $\mathcal{L}(x) \leftarrow \mathcal{L}(m(x, \theta), y)$
14:             $\nabla \mathcal{L}(x) \leftarrow \text{Backpropagate}(\mathcal{L}(x))$
15:             $S(x) \leftarrow \sum_{k=1}^{J} n_k p_k(x)$
16:             $\boldsymbol{W} \leftarrow n_i p_i(x) / \sum_{k=1}^{J} n_k p_k(x)$            $\Lsh$ Momentum estim.
17:             $\langle \boldsymbol{A} \rangle \leftarrow \langle \boldsymbol{A} \rangle + (1-\beta) \sum_{i=1}^{n_j} \boldsymbol{W}_i \boldsymbol{W}_i^T$
18:             $\langle \boldsymbol{b} \rangle \leftarrow \langle \boldsymbol{b} \rangle + (1-\beta) \sum_{i=1}^{n_j} \nabla \mathcal{L}(x_i) \boldsymbol{W}_i / S(x_i)$
19:             $\boldsymbol{q}(x) \leftarrow \alpha \boldsymbol{q}(x) + (1-\alpha) \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)}$
20:
21:         $\langle \boldsymbol{\alpha} \rangle \leftarrow \langle \boldsymbol{A} \rangle^{-1} \langle \boldsymbol{b} \rangle$
22:         $\langle \nabla L_\theta \rangle_{\text{OMIS}} \leftarrow \sum_{j=1}^{J} \langle \alpha_j \rangle$
23:         $\theta \leftarrow \theta - \eta \, \langle \nabla L_\theta \rangle_{\text{OMIS}}$            $\leftarrow$ SGD step
24:
25: return $\theta$
```

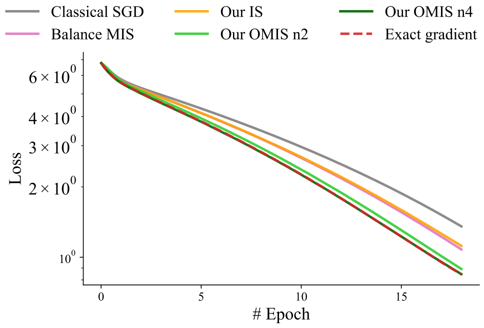


Figure 2: Convergence comparison of polynomial regression of order 6 using different method. Exact gradient show a gradient descent as baseline and classical SGD. For our method, we compare importance sampling and OMIS using $n = 2$ or 4 importance distributions. Balance heuristic MIS is also visible. Our method using OMIS achieve same convergence as exact gradient.

Momentum-based linear-system estimation. If the matrix estimate $\langle \boldsymbol{A} \rangle$ is inaccurate, its inversion can be unstable and yield a poor gradient estimate. The simplest way to tackle this problem is to use a large number of samples per distribution, which produces a accurate estimates of both $\boldsymbol{A}$ and $\boldsymbol{b}$ and thus a stable solution to the linear system. However, this approach is computationally expensive. Instead, we keep the sample counts low and reuse the estimates from previous mini-batches via momentum-based accumulation, shown in lines 17–18, where β is the parameter controlling the momentum; we use $\beta = 0.7$. This accumulation provides stability, yields an estimate of the momentum gradient Rumelhart et al. (1986), and allows us to use 1–4 samples per distribution in a mini-batch.

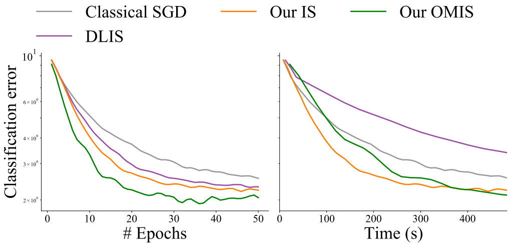


Figure 3: Classification error convergence for MNIST classification for various methods. Both Katharopoulos and Fleuret (2018) (DLIS) and resampling SGD approach. In comparison, our two method use the presented algorithm without resampling. It is visible that while DLIS perform similarly to our IS at equal epoch, the overhead of the method makes ours noticeably better at equal time for our IS and OMIS.

Importance functions. To define our importance distributions, we expand on the approach from Section 4. Instead of taking the norm of the entire output layer of the model, we take the different gradients separately as $\boldsymbol{q}(x) = \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)}$ (see Fig. 1d). Similarly to Algorithm 1, we apply momentum-based accumulation of the per-data importance (line 19 in Algorithm 2). If the output layer has more nodes than the desired number J of distributions, we select a subset of the nodes. Many other ways exist to derive the distributions, e.g., clustering the nodes into J groups and taking the norm of each; we leave such exploration for future work.

6 Experiments

Implementation details. We evaluate our importance sampling (IS) and optimal multiple importance sampling (OMIS) methods on a set of classification and regression tasks with different data modalities (images, point clouds). We compare them to classical SGD (which draws mini-batch samples uniformly without replacement), DLIS Katharopoulos and Fleuret (2018), and LOW Santiago et al. (2021). DLIS uses a resampling scheme that samples an initial, larger mini-batch uniformly and then selects a fraction of them for backpropagation and a gradient step. This resampling is based on an importance sampling metric computed by running a forward pass for each initial sample. LOW applies adaptive weighting to uniformly selected mini-batch samples to give im-

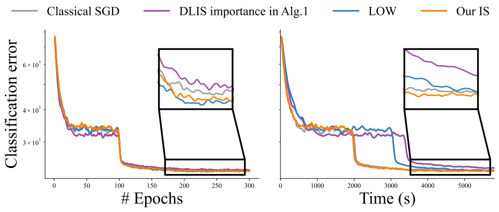


Figure 4: On CIFAR-100, we use the DLIS importance metric in our Algorithm 1 instead of the DLIS resampling algorithm. The zoom-in highlights show error drops when the learning rate decreases after epoch 100. Our method (Our IS) outperforms LOW Santiago et al. (2021) and DLIS weights at equal epochs (left). It also converges faster than LOW and DLIS weights at equal time (right).

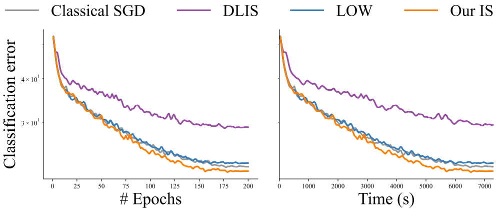


Figure 5: Comparisons on CIFAR-10 using Vision Transformer (ViT) Dosovitskiy et al. (2020). The results show our importance sampling scheme (Our IS) can improve over classical SGD, LOW Santiago et al. (2021) and DLIS Katharopoulos and Fleuret (2018) on modern transformer architecture.

portance to data with high loss. All reported metrics are computed on data unseen during training, with the exception of the regression tasks.

All experiments are conducted on a single NVIDIA Tesla A40 graphics card. Details about the optimization setup of each experiment can be found in Appendix A.

Convex problem. We performed a basic convergence analysis of IS and OMIS on a convex polynomial-regression problem. Figure 2 compares classical SGD, our IS, and three MIS techniques: balance heuristic Veach (1997) and our OMIS using two and four importance distributions. The exact gradient serves as a reference point for optimal convergence. Balance-heuristic MIS exhibits similar convergence to IS. This can be attributed to the weights depending solely on the relative importance distributions, disregarding differences in individual parameter derivatives. This underscores the unsuitability of the balance heuristic as a weighting method for vector-valued estimation. Both our OMIS variants achieve convergence similar to that of the exact gradient. The four-distribution variant achieves the same quality as the exact gradient using only 32 data samples per mini-batch. This shows the potential of OMIS to achieve low error in gradient estimation even at low mini-batch sizes.

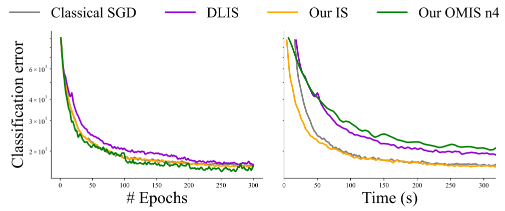


Figure 6: Comparison of our two methods (Our IS, Our OMIS) on point-cloud classification using PointNet Qi et al. (2017) architecture. Our OMIS achieves lower classification error at equal epochs, though it introduces computation overhead as shown at equal-time comparisons. At equal time, our method using importance sampling achieves the best performance.

Classification. In Fig. 3, we compare our algorithms to the DLIS resampling algorithm of Katharopoulos and Fleuret (2018) on MNIST classification. Our IS performs slightly better than DLIS, and our OMIS does best. The differences between our methods and the rest are more pronounced when comparing equal-time performance. DLIS has a higher computational cost as it involves running a forward pass on a large mini-batch to compute resampling probabilities. Our OMIS requires access to the gradient of each mini-batch sample; obtaining these gradients in our current implementation is inefficient due to technical limitations in the optimization framework we use (PyTorch). Nevertheless, the method manages to make up for this overhead with a higher-quality gradient estimate. In Fig. 3 we compare classification error; loss-convergence plots are shown in Appendix F (Fig. 8).

In Fig. 4, we compare our IS against using the DLIS importance function in Algorithm 1 and LOW Santiago et al. (2021) on CIFAR-100 classification. At equal number of epochs, the difference between the methods is small (see close-up view). Our IS achieves similar classification accuracy as LOW and outperforms the DLIS variant. At equal time the difference is more important as our method has lower computational cost. This experiment shows that our importance function achieves better performance than that of DLIS within the same optimization algorithm.

Figure 5 shows a similar experiment on CIFAR-10 using a vision transformer Dosovitskiy et al. (2020). Our IS method achieves consistent improvement over the state of the art. The worse convergence of (original, resampling-based) DLIS can be attributed to its resampling tending to exclude some training data with very low importance, which can cause overfitting.

Figure 6 shows point-cloud classification, where our IS is comparable to classical SGD and our OMIS

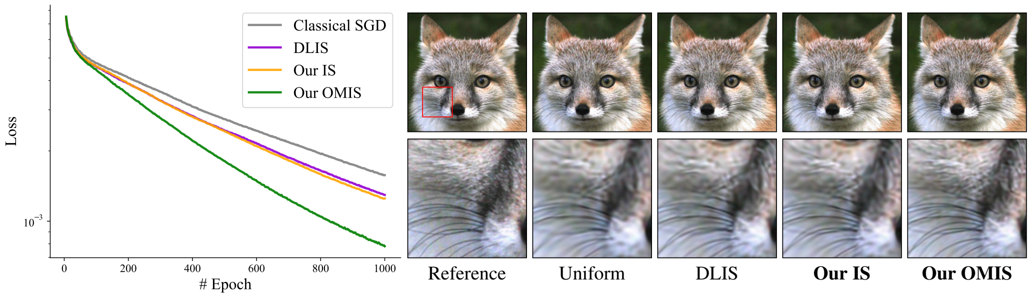


Figure 7: Comparison at equal step for image 2D regression. Left side show the convergence plot while the right display the result regression and a close-up view. Our method using MIS achieves the lower error on this problem while IS and DLIS perform similarly. On the images it is visible that our OMIS recover the finest details of the fur and whiskers.

outperforms other methods in terms of classification error at equal epochs. In complex cases where importance sampling cannot enhance convergence by providing a more accurate gradient estimator, our method is still as efficient as SGD due to minimal overhead. This means that even though importance sampling does not offer additional benefits in these scenarios, our implementation remains competitive with classical methods. In his case DLIS and our OMIS both suffer from computational overhead.

We also perform an ablation study for linear-system momentum in Algorithm 2. We apply same momentum on the gradient for classical SGD, DLIS and our IS. Appendix F (Fig. 9) shows this comparison. Our OMIS still outperforms other methods for this task at equal steps.

Regression. Figure 7 shows results on image regression, comparing classical SGD, DLIS, and our IS and OMIS. Classical SGD yields a blurry image, as seen in the zoom-ins. DLIS and our IS methods achieves similar results, with increased whisker sharpness but still blurry fur, though ours has slightly lower loss and is computationally faster, as discussed above. Our OMIS employs three sampling distributions based on the network's outputs which represent the red, green and blue image channels. This method achieves the lowest error and highest image fidelity, as seen in the zoom-in.

7 Limitations and future work

We have showcased the effectiveness of importance sampling and optimal multiple importance sampling (OMIS) in machine-learning optimization, leading to a reduction in gradient-estimation error. Our current OMIS implementation incurs some overhead as it requires access to individual mini-batch sample gradients. Modern optimization frameworks can efficiently compute those gradients in parallel but only return their average. This is the main computational bottleneck in the method. The overhead of the linear system computation is negligible; we have tested using up to 10 distributions.

Our current OMIS implementation is limited to sequential models; hence its absence from our ViT experiment in Fig. 5. However, there is no inherent limitation that would prevent its use with such more complex architectures. We anticipate that similar improvements could be achieved, but defer the exploration of this extension to future work.

In all our experiments we allocate the same sampling budget to each distribution. Non-uniform sample distribution could potentially further reduce estimation variance, especially if it can be dynamically adjusted during the optimization process.

Recent work from Santiago et al. (2021) has explored a variant of importance sampling that forgoes sample-contribution normalization, i.e., the division by the probability $p(x)$ in Eq. (3) (and on line 10 of Algorithm 1). This heuristic approach lacks proof of convergence but can achieve practical improvement over importance sampling in some cases. We include a such variant of our IS method in Appendix F.

8 Conclusion

This work proposes a novel approach to improve gradient-descent optimization through efficient data importance sampling. We present a method incorporates a gradient-based importance metric that evolves during training. It boasts minimal computational overhead while effectively exploiting the gradient of the network output. Furthermore, we introduce the use of (optimal) multiple importance sampling for vector-valued, gradient estimation. Empirical evaluation on typical machine learning tasks demonstrates the tangible benefits of combining several importance distributions in achieving faster convergence.

REFERENCES

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. (2015). Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*.

Alfarra, M., Hanzely, S., Albasyoni, A., Ghanem, B., and Richtarik, P. (2021). Adaptive learning of the optimal batch size of sgd.

Balles, L., Romero, J., and Hennig, P. (2017). Coupling adaptive batch sizes with learning rates. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, page ID 141.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(54):1579–1619.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142.

Dong, C., Jin, X., Gao, W., Wang, Y., Zhang, H., Wu, X., Yang, J., and Liu, X. (2021). One backward from ten forward, subsampling for large-scale deep learning. *arXiv preprint arXiv:2104.13114*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Faghri, F., Duvenaud, D., Fleet, D. J., and Ba, J. (2020). A study of gradient variance in deep learning. *arXiv preprint arXiv:2007.04532*.

Grittmann, P., Georgiev, I., Slusallek, P., and Křivánek, J. (2019). Variance-aware multiple importance sampling. *ACM Trans. Graph.*, 38(6).

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Kahn, H. (1950). Random sampling (monte carlo) techniques in neutron attenuation problems–i. *Nucleonics*, 6(5):27, passim.

Kahn, H. and Marshall, A. W. (1953). Methods of reducing sample size in monte carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278.

Katharopoulos, A. and Fleuret, F. (2017). Biased importance sampling for deep neural network training. *ArXiv*, abs/1706.00043.

Katharopoulos, A. and Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2525–2534. PMLR.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kondapaneni, I., Vévoda, P., Grittmann, P., Skřivan, T., Slusallek, P., and Křivánek, J. (2019). Optimal multiple importance sampling. *ACM Transactions on Graphics (TOG)*, 38(4):37.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, Toronto, ON, Canada.

Loshchilov, I. and Hutter, F. (2015). Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*.

Needell, D., Ward, R., and Srebro, N. (2014). Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Owen, A. and Zhou, Y. (2000). Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

Ren, H., Zhao, S., and Ermon, S. (2019). Adaptive antithetic sampling for variance reduction. In *International Conference on Machine Learning*, pages 5420–5428. PMLR.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Santiago, C., Barata, C., Sasdelli, M., Carneiro, G., and Nascimento, J. C. (2021). Low: Training deep neural networks by learning optimal sample weights. *Pattern Recognition*, 110:107585.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473.

Veach, E. (1997). *Robust Monte Carlo methods for light transport simulation*, volume 1610. Stanford University PhD thesis.

Wang, L., Yang, Y., Min, R., and Chakradhar, S. (2017). Accelerating deep neural network training with inconsistent stochastic gradient descent. *Neural Networks*, 93:219–229.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

Zhang, C., Öztireli, C., Mandt, S., and Salvi, G. (2019). Active mini-batch sampling using repulsive point processes. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 33, pages 5741–5748.

Zhang, M., Dong, C., Fu, J., Zhou, T., Liang, J., Liu, J., Liu, B., Momma, M., Wang, B., Gao, Y., et al. (2023). Adaselection: Accelerating deep learning training through data subsampling. *arXiv preprint arXiv:2306.10728*.

Zhao, P. and Zhang, T. (2015). Stochastic optimization with importance sampling for regularized loss minimization. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1–9, Lille, France. PMLR.

A Optimization details

Classification. The classification tasks include image classification (MNIST Deng (2012), CIFAR-10/100 Krizhevsky et al. (2009) and point-cloud (ModelNet40 Wu et al. (2015)) classification.

The MNIST database contains 60,000 training images and 10,000 testing images. We train a 3-layer fully-connected network (MLP) for MNIST over 50 epochs with an Adam optimizer Kingma and Ba (2014). CIFAR-10, introduced by Krizhevsky et al. (2009), is a dataset that consists of 60,000 color images of size 32x32. These images belong to 10 different object classes, each class having 6,000 images. On the other hand, CIFAR-100 Krizhevsky et al. (2009) contains 100 classes with 600 images each. For each class, there are 500 training images and 100 testing images. In our experiments, we train the ResNet-18 network He et al. (2016) on both datasets. We apply random horizontal flip and random crops to augment the data during training. ModelNet40 contains 9,843 point clouds for training and 2,468 for testing. Each point cloud has 1,024 points. We train a PointNet Qi et al. (2017) with 3 shared MLP layers and 2 fully-connected layers for 300 epochs on point-cloud classification. We use the Adam optimizer Kingma and Ba (2014), with batch size 64, weight decay 0.001, initial learning rate 0.00002 divided by 2 after 100, 200 epochs.

Regression. Polynomial regression consists of optimizing the coefficients of a 1D polynomial of a given order to fit randomly drawn data from a reference polynomial of the same order. The reference data are generated on the interval $[-2;2]$. Optimization is done using an Adam optimizer Kingma and Ba (2014) with a mini-batch size of 32 elements.

The image regression task consists in learning the mapping between a 2D coordinate input (pixel coordinate) and the 3-color output of the image for this pixel. We use a network with 5 fully-connected layers associated with positional encodings using SIREN activations Sitzmann et al. (2020). The training is done over 500 epoch using an Adam Kingma and Ba (2014) optimizer and each mini-batch is composed of 256 pixels for a 512^2 reference image.

B Multiple importance sampling in brief

Importance sampling. An Importance sampling Monte Carlo estimator $\langle F\rangle_{\mathrm{IS}}$ of a function f is define as :

$$\langle F\rangle_{\mathrm{IS}} = \sum_{i=1}^{n} \frac{f(x_i)}{np(x_i)}, \qquad x_i \propto p(x). \tag{9}$$

With x_i the i^{th} data sample drawn following the probability distribution function $p(x)$.

The effectiveness of this estimator depends on the relation between the functions $f(x)$ and $p(x)$. The variance of such estimator is :

$$\mathrm{Var}[\langle F\rangle_{\mathrm{IS}}] = \frac{1}{n}\mathrm{Var}[{}^{f}\!/\!_{p}]. \tag{10}$$

Reducing variance in the estimation depends on the proportionality between the function f and the probability density p.

When dealing with multivariate functions, finding a probability density proportional to every parameters is often impractical. A trade-off is required to obtain a single probability distribution maximizing the proportionality with all the parameters of the function simultaneously. Several studies, such as Zhao and Zhang (2015); Needell et al. (2014); Wang et al. (2017); Alain et al. (2015), have shown that the optimal choice of sampling strategy is the L_2 norm of the function f.

Multiple importance sampling. Multiple Importance Sampling (MIS) is a technique that combines multiple sampling strategies with associated weightings, unlike importance sampling which relies on a single strategy. This approach allows for a more versatile gradient estimation. The MIS Monte Carlo estimator, denoted as $\langle F\rangle_{\mathrm{MIS}}$, is calculated by summing over all samples drawn independently for each strategy, and then using a weighted estimator. The equation for $\langle F \rangle_{\text{MIS}}$ is given by:

$$\langle F \rangle_{\text{MIS}} = \sum_{j=1}^{J} \sum_{i=1}^{n_j} w_j(x_{ij}) \frac{f(x_{ij})}{n_j p_j(x_{ij})} \tag{11}$$

Here, x_{ij} represents the i^{th} sample from the j^{th} technique, $w_j(x)$ is a weighting function such that $f(x) \neq 0 \Rightarrow \sum_{j=1}^{J} w_j(x) = 1$, and $p_j(x) = 0 \Rightarrow w_j(x) = 0$. J is the number of sampling techniques, and n_j is the number of samples generated by the j^{th} technique. The variance of a Monte Carlo estimator using MIS, denoted as $\text{Var}[\langle F \rangle_{\text{MIS}}]$, can be expressed as:

$$\text{Var}[\langle F \rangle_{\text{MIS}}] = \sum_{j=1}^{J} \int_D \frac{w_j(x)^2 f(x)^2}{n_j p_j(x)} dx - \sum_{j=1}^{J} \frac{1}{n_j} \langle w_j, f \rangle^2 \tag{12}$$

The balance heuristic Veach (1997) is the most commonly used MIS heuristic. It sets the weight of the samples from each technique according to the following equation:

$$w_j(x_i) = \frac{n_j p_j(x_i)}{\sum_{k=1}^{J} n_k p_k(x_i)} \tag{13}$$

This weighting strategy effectively mitigates the impact of events with low probability when samples are drawn from a low-probability distribution. It prevents a large increase in the contribution of such events in the Monte Carlo estimator (11) where the function value would be divided by a very low value. The balance heuristic compensates for this and avoids extreme cases. Overall, this weighting strategy increases the robustness of the importance sampling estimator, but it is limited by its independence from the function value.

Optimal weighting. Following the discussion in Section 5, it can also be deduced from Eqs. (6) and (11) that $\langle F \rangle_{\text{OMIS}} = \sum_{j=1}^{J} \alpha_j$. Given a set of probability distribution functions $p_1, \ldots, p_J$, we can formulate the optimal MIS solver as Algorithm 3. $\boldsymbol{W}_{ij}$ represents the vector containing the balance weight (13) w.r.t. the J sampling techniques and the normalization factor $S(x_{ij}) = \sum_{k=1}^{J} n_k p_k(x_{ij})$.

The algorithm proceeds through three key stages. The first stage involves initializing the linear system defined in Eq. (7) (line 1). The second stage iteratively updates the system for each drawn data sample (lines 5-6). Upon completion of this process, the matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ provide Monte Carlo approximations of the quantities specified in Eq. (7). The third and final stage involve solving the linear system to obtain the vector $\boldsymbol{\alpha}$ (line 7). The estimated value of $\langle F \rangle^{o}_{\text{MIS}}$ is then returned.

Algorithm 3 Optimal multiple importance sampling solver.

```
 1: $\langle \boldsymbol{A} \rangle \leftarrow 0^{J \times J}, \langle \boldsymbol{b} \rangle \leftarrow 0^{J}$
 2: for $t \leftarrow 1$ to $T$
 3:     for $j \leftarrow 1$ to $J$
 4:         $\{x_{ij}\}_{i=1}^{n_j} \leftarrow$ draw $n_j$ samples from technique $p_j$
 5:
 6:     $\langle \boldsymbol{A} \rangle \leftarrow \langle \boldsymbol{A} \rangle + \sum_{j=1}^{J} \sum_{i=1}^{n_j} \boldsymbol{W}_{ij} \boldsymbol{W}_{ij}^{T}$
 7:     $\langle \boldsymbol{b} \rangle \leftarrow \langle \boldsymbol{b} \rangle + \sum_{j=1}^{J} \sum_{i=1}^{n_j} f(x_{ij}) \boldsymbol{W}_{ij} / S(x_{ij})$
 8:
 9: $\langle \boldsymbol{\alpha} \rangle \leftarrow$ solve linear system $\langle \boldsymbol{A} \rangle \langle \boldsymbol{\alpha} \rangle = \langle \boldsymbol{b} \rangle$
10: return $\sum_{j=1}^{N} \langle \boldsymbol{\alpha_j} \rangle$
```

It can be noted that the linear system size scales with the number of sampling techniques. More importantly each sampling technique needs to be sampled in order create a linear system possible to solve. The number a sample of each technique does not have to be the same but requires to be fixed at the start of the algorithm. Also the presented algorithm works for a scalar value function. In the case of multivariate function, multiple contribution vector $\boldsymbol{b}$ need to be constructed (one per parameter) and the linear system needs to be solved for each.

C Algorithm details

This section presents the two initialization subroutine for Algorithm 1 and Algorithm 2. The role of the methods is to run a first epoch in a classical SGD loop in order to process every data once. For each data the importance metric is reported into the memory q and returned with the current model parameters. This approach avoids computing the importance for all data without benefiting from the required forward step computed.

D Cross-entropy loss gradient

Machine learning frameworks take data x as input, perform matrix multiplication with weights and biases added. The output layer is then fed to the softmax function to obtain values s that are fed to the loss function. y represents the target values. We focus on the categorical cross-entropy loss function for the classi-

Algorithm 4 SGD-based initialization of persistent per-data importance q in Algorithm 1.

```
function INITIALIZE($x$,$y$,$\Omega$,$\theta$,$B$)
    for $t \leftarrow 1$ to $|\Omega|/B$
        $x,y \leftarrow \{x_i,y_i\}_{i=(t-1)\cdot B+1}^{t\cdot B+1}$
        $l(x) \leftarrow \mathcal{L}(m(x,\theta),y)$
        $\nabla l(x) \leftarrow \text{Backpropagate}(l(x))$
        $\langle \nabla L_\theta \rangle(x) \leftarrow \nabla l(x)/B$            ← Eq. (3)
        $\theta \leftarrow \theta - \eta \langle \nabla L_\theta \rangle(x)$            ← Eq. (2)
        $q(x) \leftarrow \left\| \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)} \right\|$

    return $q$,$\theta$
```

Algorithm 5 Subroutine for initialization in Algorithm 2

```
function INITIALIZEMIS($x$,$y$,$\Omega$,$\theta$,$B$)
                                        Initialize $\boldsymbol{q}$ in a classical SGD loop
    for $t \leftarrow 1$ to $|\Omega|/B$
        $x,y \leftarrow \{x_i,y_i\}_{i=(t-1)\cdot B+1}^{t\cdot B+1}$
                                        See all samples in the first epoch
        $l(x) \leftarrow \mathcal{L}(m(x,\theta),y)$
        $\nabla l(x) \leftarrow \text{Backpropagate}(l(x))$
        $\langle \nabla L_\theta \rangle(x) \leftarrow \nabla l(x)/B$            ← Eq. (3)
        $\theta \leftarrow \theta - \eta \langle \nabla L_\theta \rangle(x)$            ← Eq. (2)
        $\boldsymbol{q}(x) \leftarrow \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)}$

    return $\boldsymbol{q}$,$\theta$
```

fication problem (with J categories) given by:

$$\mathcal{L}_{\text{cross-ent}} = -\sum_i y_i \log s_i, \quad s_i = \frac{\exp(m(x_i,\theta)_l)}{\sum_l^J \exp(m(x_i,\theta)_l)}. \tag{14}$$

For backpropagation, we need to calculate the derivative of the $\log s$ term w.r.t. the weighted input z of the output layer. We can easily derive the derivative of the loss from first principles as shown below:

Algorithm 6 Subroutine for cross entropy loss importance metric

```
$x_i$ = data sample, $y_i$ = class index of $x_i$
function IMPORTANCE($x_i$,$y_i$)
    $s \leftarrow \exp(m(x_i,\theta))/\sum_{k=1}^{J}\exp(m(x_i,\theta)_k)$        ← Eq.14
    $q \leftarrow \sum_{j=1}^{J} s_j - \mathbf{1}_{j=y_i}$        ← Eq.16
    return $q$
```

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\text{cross-ent}}}{\partial m(x_i,\theta)_j} &= -\frac{\partial}{\partial m(x_i,\theta)_j}\left(\sum_i^J y_i \log s_i\right) \\
&= -\sum_i^J y_i \frac{\partial}{\partial m(x_i,\theta)_j} \log s_i \\
&= -\sum_i^J \frac{y_i}{s_i} \frac{\partial s_i}{\partial m(x_i,\theta)_j} \\
&= -\sum_i^J \frac{y_i}{s_i} s_i \cdot (\mathbf{1}\{i == j\} - s_j) \\
&= \sum_i^J y_i \cdot s_j - \sum_i^J y_i \cdot (\mathbf{1}\{i == j\}) \\
&= s_j \sum_i^J y_i - y_j = s_j - y_j
\end{aligned} \tag{15}$$

The partial derivative of the cross-entropy loss function w.r.t. output layer parameters has the form:

$$\frac{\partial \mathcal{L}_{\text{cross-ent}}}{\partial m(x_i,\theta)_j} = s_j - y_j \tag{16}$$

For classification tasks, we directly use this analytic form of the derivative and compute it's norm as weights for adaptive and importance sampling.

E Importance momentum

Updating the persistent per-sample importance q directly sometime leads to a sudden decrease of accuracy during training. To make the training process more stable, we update q by linearly interpolating the importance at the previous and current steps:

$$q(x) = \alpha \cdot q_{prev}(x) + (1-\alpha) \cdot q(x) \tag{17}$$

where α is a constant for all data samples. In practice, we use $\alpha \in \{0.0, 0.1, 0.2, 0.3\}$ as it gives the best trade-off between importance update and stability. This can be seen as a momentum evolution of the per-sample importance to avoid high variation. Utilizing an exponential moving average to update the importance metric prevents the incorporation of outlier values. This is particularly beneficial in noisy setups, like situations with a high number of class or a low total number of data.

F Additional results

This section provides additional results, including an ablation study as shown in Fig. 9 for linear-system momentum used in Algorithm 2 and results of our adaptive sampling method. Figs. 6 and 9 demonstrate that classical SGD, DLIS and Our IS work similarly with and without momentum. Our OMIS outperforms other methods in both cases.

Figures 8, 10 and 11 show that our adaptive sampling variant (our AS) can achieve better results than our IS or our OMIS in practice. Our AS is a heuristic and we leave its theoretical formulation as future work.

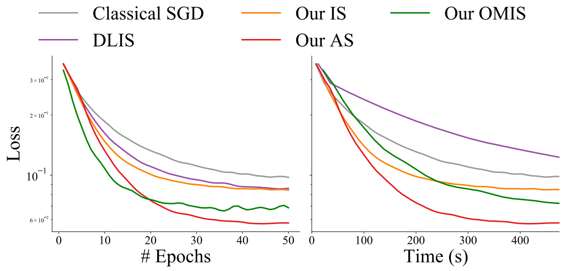


Figure 8: We compare loss for the MNIST dataset between the resampling algorithm by Katharopoulos and Fleuret (2018) (DLIS) and our algorithm. At equal epochs, DLIS works better than both classical and resampling SGD. However, at equal time, the resampling cost is too high, making DLIS even slower than standard SGD.

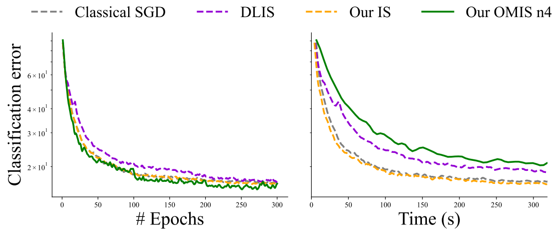


Figure 9: Ablation study on point-cloud classification using linear-system momentum as described in Algorithm 2 for baselines represented as dashed lines. Our OMIS still outperforms other baselines at equal epochs, similar to the results shown in Fig. 6.

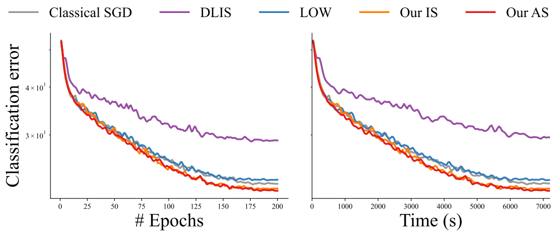


Figure 10: Comparisons on CIFAR-10 using Vision Transformer (ViT) Dosovitskiy et al. (2020). The results show our importance sampling scheme (Our IS) and the adaptive sampling variant (Our AS) can improve over classical SGD, LOW Santiago et al. (2021) and DLIS Katharopoulos and Fleuret (2018) on modern transformer architecture.

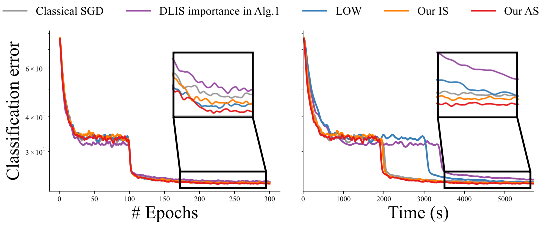


Figure 11: On CIFAR-100 classification dataset, instead of comparing the DLIS resampling algorithm, we use DLIS importance in our Algorithm 1. We display zoom-in of the end of the curves to highlight the differences. At equal epochs (left), our methods (Our IS & Our AS) show improvements compared to LOW Santiago et al. (2021) and DLIS weights. At equal time (right), LOW and the DLIS weights takes longer to converge. Overall our approach shows faster convergence with lower importance computation.