Not All Samples Are Created Equal: Deep Learning with Importance Sampling

Angelos Katharopoulos and François Fleuret

Machine Learning, Data Loading
https://doi.org/10.48550/arXiv.1803.00942
http://proceedings.mlr.press/v80/katharopoulos18a.html
https://github.com/idiap/importance-sampling

Abstract

Deep Neural Network training spends most of the computation on examples that are properly handled, and could be ignored. We propose to mitigate this phenomenon with a principled importance sampling scheme that focuses computation on "informative" examples, and reduces the variance of the stochastic gradients during training. Our contribution is twofold: first, we derive a tractable upper bound to the per-sample gradient norm, and second we derive an estimator of the variance reduction achieved with importance sampling, which enables us to switch it on when it will result in an actual speedup. The resulting scheme can be used by changing a few lines of code in a standard SGD procedure, and we demonstrate experimentally on image classification, CNN fine-tuning, and RNN training, that for a fixed wall-clock time budget, it provides a reduction of the train losses of up to an order of magnitude and a relative improvement of test errors between 5% and 17%.

Related Study

Importance sampling aims at increasing the convergence speed of SGD by focusing computation on samples that actually induce a change in the model parameters. This formally translates into a reduced variance of the gradient estimates for a fixed computational cost.

- Importance Sampling for Convex Problems[1]
  - For convex optimization problems, many works have taken advantage of the **difference in importance among the samples to improve the convergence speed of stochastic optimization methods**.
- Importance Sampling for Deep Learning
  - For deep neural networks, sample selection methods were mainly employed to generate hard negative samples[2] for embedding learning problems or to tackle the class imbalance problem.
  - More closely related to our work, Schaul et al. (2015) and Loshchilov & Hutter (2015) **use the loss to create the sampling distribution**.
    - Both approaches **keep a history of losses for previously seen samples**, and sample either proportionally to the loss or based on the loss ranking.
    - One of the main limitations of history based sampling, is the need for tuning a large number of hyperparameters that control the effects of “stale” importance scores;
- Importance sampling to improve and accelerate the training of neural networks
  - Those works, employ either **the gradient norm** or **the loss** to compute each sample’s importance.
  - However, the former is prohibitively expensive to compute and the latter is not a particularly good approximation of the gradient norm.
- Other Sample Selection Methods
  - Design a distribution (suitable only for the distance based losses) that maximizes the diversity of the losses in a single batch.
  - Use reinforcement learning to train a neural network that selects samples for another neural network in order to optimize the convergence speed.
  - Although their preliminary results are promising, the overhead of training two networks makes the wall-clock speedup unlikely and their proposal not as appealing.

Proposed Method

1. Compared to the aforementioned works, we derive ***an upper bound* to the per sample gradient norm** that can be computed in a single forward pass.
2. Furthermore, we **quantify the variance reduction** achieved with the proposed importance sampling scheme and associate it with ***the batch size increment* required to achieve an equivalent variance reduction**.

Algorithm 1 Deep Learning with Importance Sampling

Inputs $B, b, \tau_{th}, a_\tau, \theta_0$
$t \leftarrow 1$
$\tau \leftarrow 0$
repeat
  if $\tau > \tau_{th}$ **then**
    $\mathcal{U} \leftarrow B$ uniformly sampled datapoints
    $g_i \propto \hat{G}_i \quad \forall i \in \mathcal{U}$ according to eq 20
    $\mathcal{G} \leftarrow b$ datapoints sampled with g_i from $\mathcal{U}$
    $w_i \leftarrow \frac{1}{B g_i} \quad \forall i \in \mathcal{G}$
    $\theta_t \leftarrow \text{sgd\_step}(w_i, \mathcal{G}, \theta_{t-1})$
  else
    $\mathcal{U} \leftarrow b$ uniformly sampled datapoints
    $w_i \leftarrow 1 \quad \forall i \in \mathcal{U}$
    $\theta_t \leftarrow \text{sgd\_step}(w_i, \mathcal{U}, \theta_{t-1})$
    $g_i \propto \hat{G}_i \quad \forall i \in \mathcal{U}$
  end if
  $\tau \leftarrow a_\tau \tau + (1 - a_\tau)\left(1 - \frac{1}{\sum_i g_i^2}\left\|g - \frac{1}{|\mathcal{U}|}\right\|_2^2\right)^{-1}$
until convergence

- In order to solve the problem of computing the importance for the whole dataset, we pre-sample a large batch of data points, compute the sampling distribution for that batch and re-sample a smaller batch with replacement.
- The inputs to the algorithm are the pre-sampling size *B*, the batch size *b*, the equivalent batch size increment after which we start importance sampling τ_{th} and the exponential moving average[3] parameter a_τ used to compute a smooth estimate[4] of $\tau \cdot \theta_0$ denotes the initial parameters of our deep network.
- Due to the large cost of computing the importance per sample, we only perform importance sampling when we know that the variance of the gradients can be reduced.
- Our implementation is generic and can be employed by adding a single line of code in a standard Keras model training.

Evaluation and Results

1. **uniform** : the usual training algorithm that samples points from a uniform distribution
2. **loss** : `Algorithm 1` but instead of sampling from a distribution proportional to our upper-bound to the gradient norm $\hat{G}_i$ (equations 8 and 20), we sample from a distribution proportional to the loss value
3. **upper-bound** : our proposed method.

- Ablation study[5]
  - The variance reduction achieved with every sampling scheme

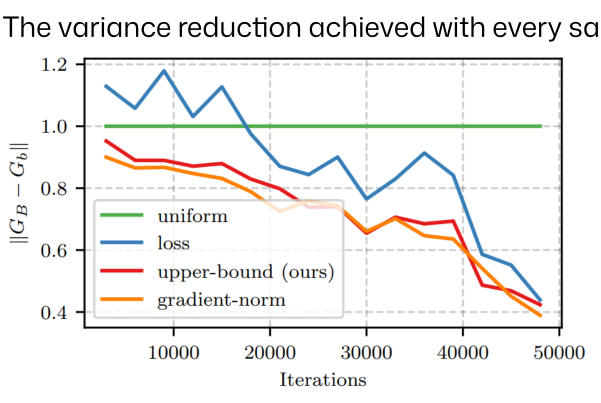


Figure 1: The y-axis denotes the L_2 distance of the average gradient of the large batch (G_B) and the average gradient of the small batch (G_b) normalized with the distance achieved by uniform sampling. The sampling of the small batch is done 10 times and the reported results are the average. The details of the experimental setup are given in § 4.1.

    - Subsequently, we sample 1,024 images uniformly at random from the dataset. Using the weights of the trained network, at intervals of 3,000 updates, we resample 128 images from the large batch of 1,024 images using **uniform** sampling or importance sampling.
  - The probabilities against the optimal ones (proportional to the *gradient-norm*)

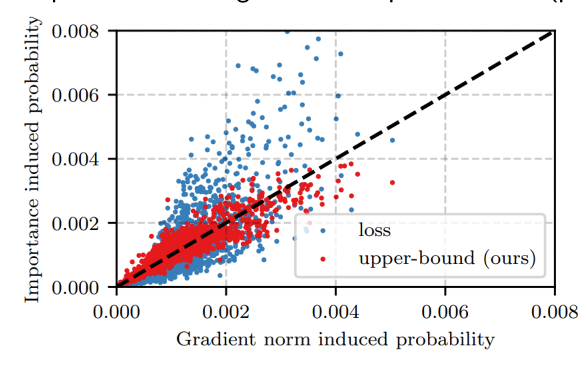
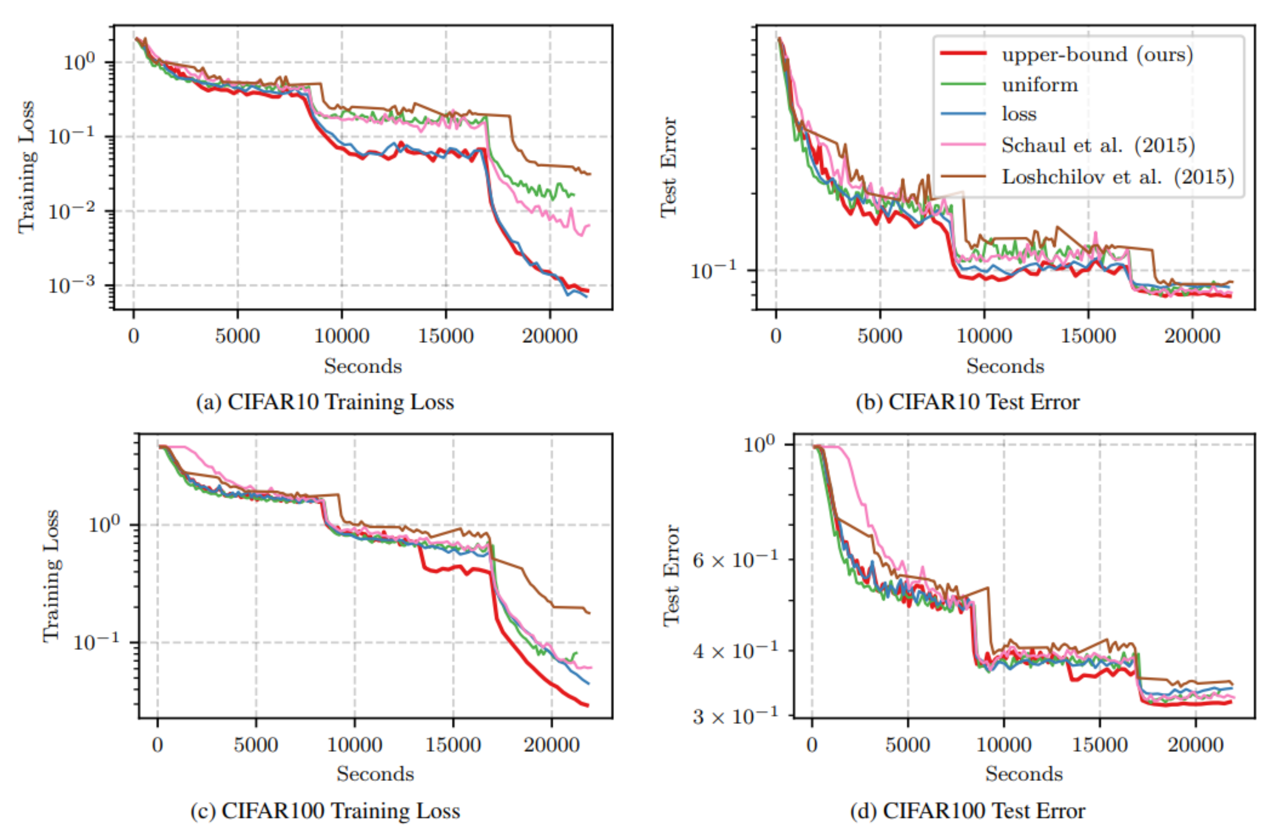


Figure 2: The probabilities generated with the *loss* and our *upper-bound* are plotted against the ideal probabilities produced by the *gradient-norm*. The black line denotes perfect correlation. The details of the experimental setup are given in § 4.1.

    - In figure 2, we observe that our **upper bound** is almost perfectly correlated with the gradient norm, in stark contrast to the **loss** which is only correlated at the regime of very small gradients.

- Image classification



(a) CIFAR10 Training Loss
(b) CIFAR10 Test Error
(c) CIFAR100 Training Loss
(d) CIFAR100 Test Error

Figure 3: Comparison of importance sampling using the *upper-bound* with *uniform* and loss based importance sampling. The details of the training procedure are given in § 4.2. Our proposed scheme is the only one achieving a speedup on CIFAR100 and results in 5% smaller test error. All presented results are averaged across 3 independent runs.

  - We follow the experimental setup of Zagoruyko & Komodakis (2016), specifically we train a wide resnet 28-2 with SGD with momentum. We use batch size 128, weight decay 0.0005, momentum 0.9, initial learning rate 0.1 divided by 5 after 20,000 and 40,000 parameter updates.
  - We train for a total of 50,000 iterations. In order for our history based baselines to be compatible with the data augmentation of the CIFAR images, we pre-augment both datasets to generate 1.5×10^6 images for each one.
  - For our method, we use a presampling size of 640. One of the goals of this experiment is to show that even a smaller reduction in variance can effectively stabilize training and provide wall-clock time speedup; thus we set $\tau_{th} = 1.5$. We perform 3 independent runs and report the average.

Notes

1. **Convex problem** : non-convex with local minimum problem↲
2. **Hard negative samples** : 어려운 negative 문제. 실제로는 negative이지만, positive(false positive)로 예측되기 쉬운 것. 이를 위해 false positive를 학습 데이터셋 안에 포함시키는 것을 hard negative mining이라고 부른다.↲
3. **Exponential moving average** : 과거의 모든 기간을 계산대상으로 하며 최근의 데이터에 더 높은 가중치를 두는 일종의 가중이동평균법이다.↲
4. **The smoothing problem** : the problem of estimating an unknown probability density function recursively over time using incremental incoming measurements.↲
5. **Ablation study** : investigates the performance of an AI system by removing certain components to understand the contribution of the component to the overall system.↲