REPRODUCIBILITY REPORT FOR A RESIZABLE MINI-BATCH GRADIENT DESCENT BASED ON A MULTI-ARMED BANDIT

Kumar Kshitij Patel, Anay Mehrotra & Tim Lebailly
Department of Computer Science
EPFL
{kumar.patel,anay.mehrotra,tim.lebailly}@epfl.ch

ABSTRACT

We reproduce the results for the 2019 ICLR submission, *A Resizable Mini-batch Gradient Descent based on a Multi-Armed Bandit*, under the ICLR reproducibility challenge 2019. We also review the claims and findings of the paper carefully. We study the probability distributions which the algorithm converges to during different runs, and try to explain the behaviour. We point out some limitations in their work, provide a discussion of their results. We also try to verify the extent of their claims, and actual effect of their algorithm. Precisely, we try to find how much tuning does it require and how exactly is it benefiting the test accuracy. This underscores some limitations in their work. Finally, we offer some suggestions to improve their research.

1 INTRODUCTION

The choice of learning rate and batch size, is intimately linked to convergence in non-convex optimization. It is well known in literature that these choices affect the bias, variance and generalization in different ways. In recent years there has been an increasing trend (Keskar et al. (2016)) to push for larger batch sizes, in order to fully leverage the computation-communication trade-offs. However, it is common to keep the batch size constant once a suitable setting is obtained after grid search over the learning rate and batch combinations. It is contrary to some observations that different batch sizes or learning rates could be better (Goodfellow et al. (2016)). While adaptive learning rate schedules are pretty widely used, it is also interesting to see the effect of varying the batch size. There exists some work trying to do that. However, such schemes monotonically increase or decrease the batch size, based on the evidence from the bias-variance trade-offs (Byrd et al. (2012); Friedlander & Schmidt (2012)). These intuitions are often misplaced in the complicated non-convex setting, where algorithm's neighbourhood keeps changing as the training proceeds. This paper proposed an adaptive batch size scheme that consider batches in the multi-armed bandit framework. EXP3 algorithm is used to find the best batch size for every epoch as the training proceeds, based on reduction of the loss. This gives the algorithm some flexibility, as well as maximizes the accuracy given the history of using a batch size and loss. There has also been some work trying to tune the hyper-parameters using bandits and Bayesian optimization (for e.g., Li et al. (2016)).

We observe that their algorithm is indeed capable of performing similarly to mini batch SGD, without the need to grid search on the optimal parameters. It also outperforms it in some cases. However, it is not clear if the need to tune is actually reduced. This is because the bandit algorithm has its own hyper-parameter. Moreover, in our experimentation with their loss, some negative results are obtained, which brings in question the need for the algorithm. Regardless, we reproduce the results in the paper and also discuss some suggestions below to improve it.

Algorithm 1 Resizable Mini-batch Gradient Descent

Input:
$\mathcal{B} = \{b_k\}_{k=1}^{K}$: Set of batch sizes
$\pi_0 = \{\frac{1}{K}, \ldots, \frac{1}{K}\}$: Prior probability distribution

Output:
$\hat{w}$: Final weights of the network

Procedure:
Initialize model parameters w_0
for epoch τ = 0,1, ... **do**
  Select batch size $b_{k_\tau} \in \mathcal{B}$ from π_τ
  Set temporal parameters $\tilde{w}_0 = w_\tau$
  for t = 0,1,...T - 1 where $T = \lceil m/b_{k_\tau} \rceil$ **do**
    Compute gradient $g_t = \nabla J(\tilde{w}_t)$
    Update $\tilde{w}_{\tau+1} = \tilde{w}_t - \eta_\tau g_t$
  end for
  Update $w_{\tau+1} = \tilde{w}_T$
  Observe validation loss $\ell(w_{\tau+1})$
  if $\ell(w_{\tau+1}) < \ell(w_\tau)$ **then**
    Get cost $y^{k_\tau} = 0$
  else
    Get cost $y^{k_\tau} = 1$
  end if
  Update probability bucket $\tilde{\pi}^{k_\tau} = \pi_\tau^{k_\tau} e^{-\beta y^{k_\tau}/\pi_\tau^{k_\tau}}$
  Normalize probability distribution $\forall i \in [K], \pi_{\tau+1}^i = \tilde{\pi}^i / \sum_j \tilde{\pi}^j$
end for
$\hat{w} = w_{\tau+1}$
return $\hat{w}$

2 Problem Setting and Algorithm Description

The algorithm RMGD proposed in the paper is described in Algorithm 1. It is a simple combination of mini-batch SGD and EXP3. After every epoch, the loss of the model is used to update the sampling probabilities of different batch sizes. In the basic variant the cost of the bandit algorithm is binary and depends on whether the loss has increased or decreased in an epoch. However, the paper also proposes and tests some graduating losses. Besides this, the other relevant choice is the set of possible batch sizes which the selector algorithm inputs. Three different batch sets are considered. A variant $name$ is defined for some combination of batch set and loss. This naming convention will be used throughout the report and is defined in Table 1. The simple Algorithm 1 can be changed to be used with any other SGD-like algorithm, including Adam and AdaGrad which have been used in this paper.

RMGD variant naming	Batch Set	cost y^{κ_τ}
basic	[16, 32, 64, 128, 256, 512]	$\mathbb{1}_{\ell_\tau > \ell_{\tau-1}}$
sub	[16, 64, 256]	$\mathbb{1}_{\ell_\tau > \ell_{\tau-1}}$
super	[16, 24, 32, 48, 64, 96, 128, 192, 256, 384, 512]	$\mathbb{1}_{\ell_\tau > \ell_{\tau-1}}$
hinge	[16, 32, 64, 128, 256, 512]	$\max\{0, \ell_{\tau+1} - \ell_\tau\}$
ratio	[16, 32, 64, 128, 256, 512]	$\max\{0, \frac{\ell_{\tau+1}-\ell_\tau}{\ell_\tau}\}$

Table 1: Description of different RMGD variants

Layer	Layer description
input	• 28 x 28 grayscale image
conv1	• Conv2d: 1 to 32 channels, kernel 5, stride 1, padding 2 • ReLU • MaxPool2d: kernel 2, stride 2
conv2	• Conv2d: 32 to 64 channels, kernel 5, stride 1, padding 2 • ReLU • MaxPool2d: kernel 2, stride 2
linear1	• Linear 3136 to 1024 • ReLU
linear2	• Linear 1024 to 10 • Softmax

Table 2: Precise architecture of the simple CNN for MNIST

3 RESULTS

Results on three data-sets have been reported: *MNIST*, *CIFAR-10* and *CIFAR-100*. We chose to only reproduce the results using MNIST with Adam and AdaGrad optimizer. We made this decision in interest of computational resources and time. A detailed architecture of the used network is to be found in Table 2. Having done that, we realize that MNIST data-set is arguably too simple to be a proof of concept for real improvements. In our actual submission we plan to run experiments on CIFAR if we can get sufficient computational resources.

3.1 SAMPLING PROBABILITIES

We observe that as mentioned in the paper there is an affinity for a larger batch size on the MNIST data-set. Moreover, this happens toward the end of the run. This behaviour could be accredited to the fact that MNIST is a *"convex-like"* data-set and during the final phases of train the algorithm wants to reduce the stochastic variance. More experiments with convex data-sets are required to prove this hypothesis. We are not able to reproduce a result in our ten runs for the basic setting which illustrates that a smaller batch size could have been chosen. Some representative results for variant $basic$ are shown in Figure 1. It should be noted that there is some difference in what batch does the distribution converge to or if it even converges. We didn't obtain a strong convergence as reported in the paper. There are however, some cases, where the best batch changes during different training phases. It seems as if this depends strongly upon the run. There is no explanation provided in the paper. We believe that this effect can be attributed to the different geometry of different local minimas.

In the graduated loss settings for hinge and ratio, we observe some more effects. The probability distribution doesn't change much in these cases. This is clear from Figure 2. This can be explained by the following. Recall that the probability update is the following $\tilde{\pi}^{k_\tau} = \pi_\tau^{k_\tau} e^{-\beta y^{k_\tau}/\pi_\tau^{k_\tau}}$. Using the $basic$ variant, the associate cost y^{κ_τ} at epoch τ is either 1 or 0. This means that when $\ell(w_{\tau+1}) > \ell(w_\tau)$, the probability update will be $\tilde{\pi}^{k_\tau} = \pi_\tau^{k_\tau} e^{-\beta/\pi_\tau^{k_\tau}}$. For the $hinge$ variant however, when

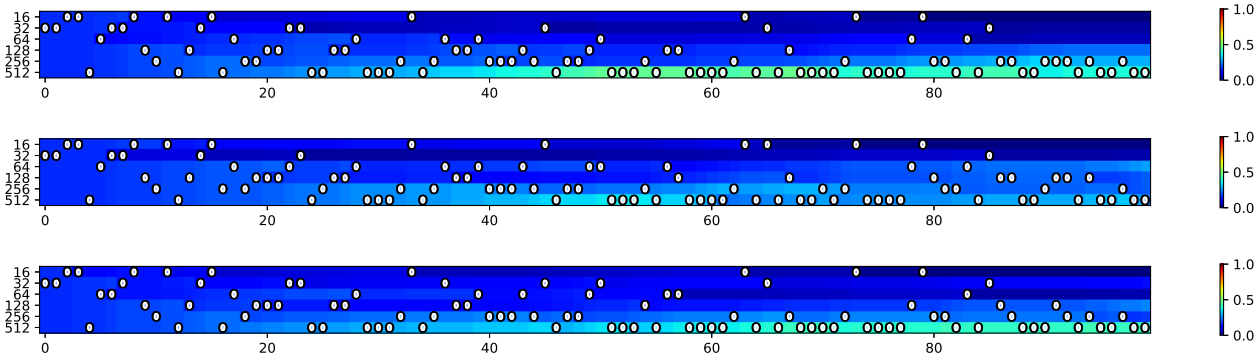

Figure 1: Sampling probabilities and selected batch sizes over different runs for the $basic$ variant

$\ell(w_{\tau+1}) > \ell(w_\tau)$, the associate cost y^{κ_τ} at epoch τ will be $\ell(w_{\tau+1}) - \ell(w_\tau)$. If the loss function is such that this value is extremely small, then the probability update will be marginal. An analog reasoning holds for $ratio$ although it is less drastic due to the normalization. The $ratio$ variant also has an additional flaw. If the loss function ℓ_τ happens to change sign during training, then the opposite of the desired probability update will happen. This can easily fixed by having a cost y^{κ_τ} to be $\max\{0, \frac{\ell_{\tau+1}-\ell_\tau}{|\ell_\tau|}\}$ instead of $\max\{0, \frac{\ell_{\tau+1}-\ell_\tau}{\ell_\tau}\}$.

A higher value of β could change this, but it hasn't been studied in the paper. Ideally, the value of beta should be obtained from the upper bound on regret. If it has to be tuned, it defeats the purpose of this paper.

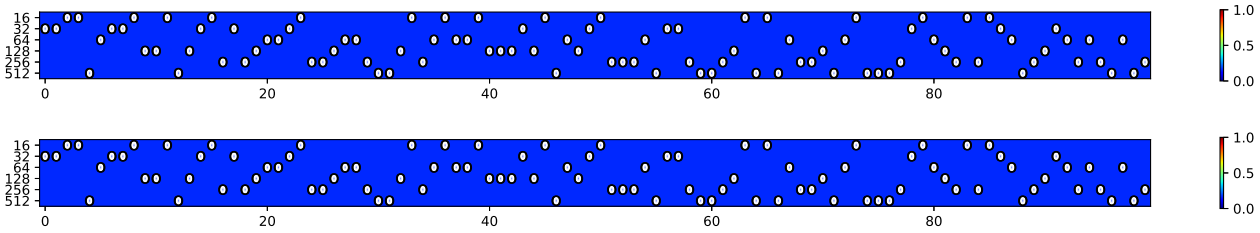

Figure 2: Sampling probabilities and selected batch sizes over different runs for the hinge (top) and ratio (bottom) variants

3.2 MNIST Results

We obtain slightly worse results compared to the ones presented in the paper, in 3. However, the relative difference between them is still the same. We expect the difference to have arisen from a non-random initialization of weights and kernels. We used PyTorch's random initialization. On contacting the authors, we find they use a different initialization. However, we didn't repeat the experiments, as the relative difference is similar and the story is unchanged. Moreover, we wanted to see how robust their results are to such random aberrations. It seems that the gains are limited but consistent. We haven't tuned Adam or β here, as the whole point of the algorithm is to do away with tuning.

3.3 Discussion

The biggest problem with the paper is that it is not clear how robust is the choice of β to different data-sets and loss functions. The authors have not specified what value has been used for graduating losses. As mentioned earlier, if the same value of β is used, uniform sampling probabilities are obtained. This amounts to the algorithm doing nothing! We suspect that the improved performance,

Variants	Mean $\pm$ SD	Max	Min
MGD (16)	99.226 ± 0.077	99.320	99.060
MGD (32)	99.242 ± 0.054	99.330	99.160
MGD (64)	99.282 ± 0.108	99.420	99.000
MGD (128)	99.270 ± 0.112	99.370	98.960
MGD (256)	99.272 ± 0.109	99.380	98.970
MGD (512)	99.167 ± 0.086	99.270	98.940
RMGD $(basic)$	99.331 ± 0.048	99.400	99.230
RMGD (sub)	99.309 ± 0.044	99.380	99.230
RMGD $(super)$	99.329 ± 0.066	99.460	99.250
RMGD $(hinge)$	99.305 ± 0.040	99.380	99.230
RMGD $(ratio)$	99.321 ± 0.021	99.350	99.280

Table 3: Test accuracies for different variants

despite this on test accuracy, is simply because this acts as an implicit regularization for neural-network training. Further, the notion of regret is in itself not appropriate to some extent. Minimizing regret over assumption of a stationary distribution, and adaptively choosing batch sizes as training proceeds are two opposite goals. Without evolving the probability distribution and learning a change point, the whole activity is futile. Overall, the paper comes off as a fortunate attempt at mixing two very different design goals. Below we test some of our hypothesis, in trying to explain our results.

4 Additional Experiments

In order to understand the actual utility of the bandit algorithm and also to study the effect of β as observed above, we carried out some of our own experiments. Precisely, the following:

- **Opposite loss:** We used the exact opposite loss for ratio loss. Surprisingly we observe that we could obtain an accuracy of 99.35 ± 0.046. This demonstrates, that even while maximizing the regret the algorithm performs much better than mini-batch. It underscores our hypothesis that their selection algorithm is doing implicit regularization, and doesn't necessarily optimize batch selection for local geometry. More evidence is required to prove the same. The selected probabilities for this case are largely uniform. Thus at this value of β, EXP3 is not updating the arm probabilities.
- **Varying β:** We want to study whether higher values of β (which we denote by $\beta_{experimental}$) are able to converge to a non-uniform probability distribution. As shown in Figure 3 it is in fact the case. This makes it clear, that tuning is still required in the algorithm to some extent. We also test whether the algorithm still performs well when using a high enough value of β with a negative objective. We observe that it is indeed the case. The performance is similar or better than mini-batch SGD, even with an opposing loss! This verifies the previous point.

The above two experiments make it clear that the gains from the algorithm can't fully be accredited to a smart selection of batch size. This brings into question the relevance of this paper. However, we realize that more extensive experiments on CIFAR are needed to conclude this hypothesis.

5 Suggested Changes

We propose the following changes and additions to the paper if it is accepted:

- **Writing:** The paper begins with a motivation to do away with using grid search. It makes it seem, as if the aim of the paper is to provide the optimal batch-size for the task. However, it doesn't do that, and is actually an adaptive algorithm on batch size. It should be rewritten

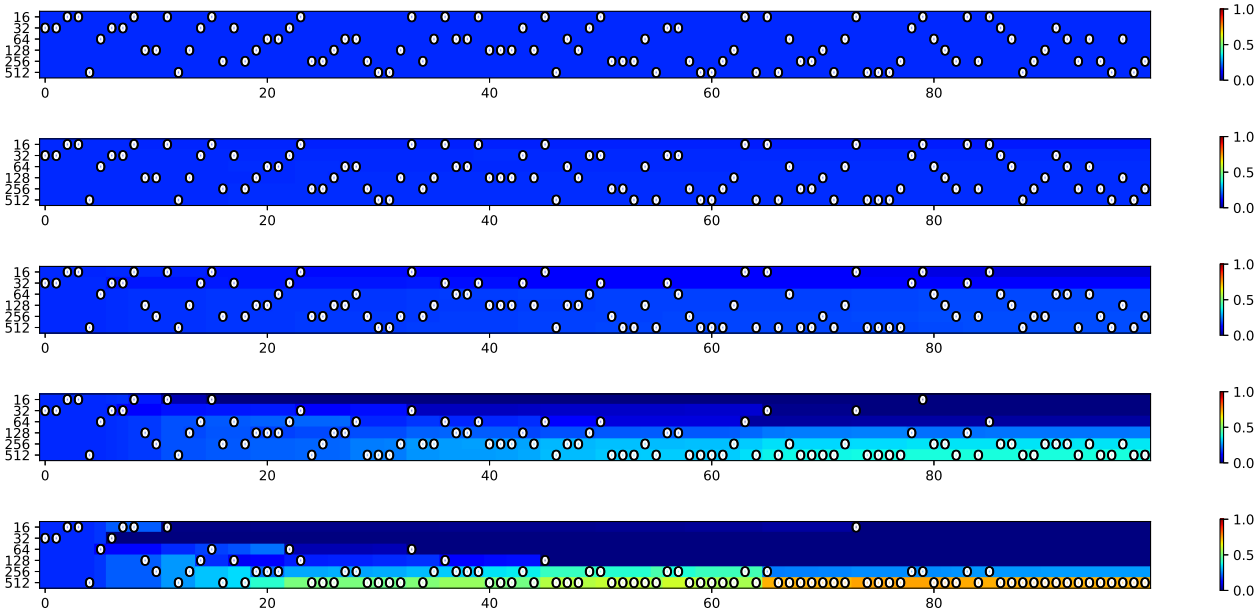

Figure 3: Sampling probabilities and selected batch sizes for the $hinge$ variant using (from top to bottom) $\beta_{experimental}$ = β, 10β, 100β, 1000β, 10000β

almost entirely with that motivation. Moreover, many of the intuitions and interpretations that we provide in this report are not available in the report.

- **Reproducibility:** The paper misses some key details about reproducing the results. We had to contact the authors for the same. These details must ideally be included in a revised version of the paper. Some of these details include the simple CNN architecture, the value of β, the initialization for neural network, etc.

- **Non-adaptive optimization algorithms:** The improvements offered by the paper are indeed surprising. However, they are marginal compared to accuracy gap with mini-batch SGD. It provides experiments with modified Adam and AdaGrad but not SGD. SGD is more sensitive to learning rate than adaptive algorithms, and changing batch sizes could worsen it. We ran some initial experiments with SGD and observe that this indeed happens in practice. However, it will be a much stronger contribution to make it work. Also, a more complex problem needs to be looked into to actually see any benefits over using tested strategies with SGD.

- **Visualizing the need for adaptivity:** It is not clear why the algorithm actually works! Mini-batch SGD, is already a great algorithm. This underscores that a more difficult setting needs to be considered to justify the utility of the algorithm. Also, experiments need to be performed to demonstrate the utility of using a different batch size at different times. We suggest studying the loss surfaces to study these phase transitions and different geometries. Links should also be made to existing non-convex literature, to better guide them. Otherwise, the strategy comes off as ad-hoc and ill-understood.

- **Proof of robustness:** Our initial results with MNIST suggest that regret minimization, the way it is defined is not detrimental to the performance gains. This brings the contribution of this paper into question. An extensive rebuttal, with more experiments need to be performed, to prove that the improvements are indeed due to better convergence, and not merely due to implicit regularization (say).

In our opinion, this paper should not be accepted to ICLR without resolving the above issues.

REFERENCES

Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

Michael P Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016. URL http://arxiv.org/abs/1609.04836.

Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560, 2016. URL http://arxiv.org/abs/1603.06560.