Comparative Performance of Gradient Descent Based Optimization Algorithms

Saket Thavanani
Department of Materials Science and Engineering
University of Toronto

Gulsima Bilgin
Department of Electrical and Computer Engineering
University of Toronto

***Abstract*—In this paper, we will perform a comparative evaluation of five most commonly used gradient-based optimization algorithms for computing parameters in Neural Network. The investigated techniques are the Stochastic Gradient Descent with momentum (SGDm), Root Mean Square Propagation (RMSProp), Adaptive Moment Estimation (Adam), Adap-tive Gradient (AdaGrad). In the initial section, we have provided the detailed Numerical Analysis of these five optimizers with the help of simple quadratic funtion with two variables. These optimizers have been compared in terms of convergence with minimal iteration for this simple function.**

Then, the paper follows a comparison the performance of these algorithms for the image classification task using convolution neural networks. The dataset used here will be ones already exist in Keras libraries such as the Fashion MNIST datasets. The performance will be visualized using different sets of hyperparameters such as accuracy, number of epochs to converge, batch size, etc.

***Index Terms*—Machine Learning, Deep Learning, Optimization, Neural Networks, Gradient Descent**

I. Motivation and Challenges

All the real word applications are developed using state of Deep Learning. Gradient Based Optimization is the most used algorithm for optimization. Due to complexity of non-convex function the main motivation is to develop gradient descent based algorithms which can compute the Global optimum and local optimum. This papers reveals the new advancement in field of gradient based optimization which involves change of learning rate based upon gradients. We aimed to arrive at the best optimization algorithm and best Neural network hyper-parameters for computer vision tasks. The major challenges faced were to determine the best algorithm based upon their convergence and also the best batch size and architecture of neural network desired for image classification task.

II. Introduction

Deep Learning is named as the future of Artificial Intelligence. These days neural networks are called as Universal Function approximators, i.e. they have the power to represent any complicated function in this universe. The idea behind computing this universal function with millions of parameters comes from a basic mathematics of optimization. Optimization can be done in several ways, but for this paper focuses on gradient descent based optimization techniques.

One of the major area of focus is computer vision. This technique uses Convolution Neural Network for tasks such as image classification and object detection. Training of a deep Convnet is a problem of extreme challenge. This involves minimizing the non-convex loss function with high dimensional complexity. Training involves tuning the parameters of the model to a state which minimizes the losses function. Since this is a non-convex function, there are high chance of getting trapped in local minimum, saddle point, and plateaus [1], [2]. Different scientists have proposed different gradient based optimization algorithms which help us to overcome these drawbacks along with the hopes of finding the local optimum and may be global. Some of the famous algorithms include Momentum [3], Adagrad [4], Adam [5] and RMSProp [6]. The RMSprop algorithm was discovered by a famous University of Toronto Professor Geoffrey Everest Hinton.

This paper has been divided into two sections- In the first section, a theoretical background of optimization algorithms along with their numerical analysis has been demonstrated. It has been proved numerically by implementing these algorithms on a simple quadratic function. The second section of the paper involves comparative evaluation of convergence rate of these algorithms on convolution neural networks. Tho goal is to find the best optimization algorithm for image classification task. Once the best optimization algorithm is found the goal is to find the best batch size corresponding to that algorithm to obtain the highest accuracy on test and training dataset.

III. Theoretical Background and Numerical Analysis of Optimizers

In order to provide a better understanding of each optimization algorithm, the numerical analysis was performed on a simple quadratic function with two variables (1). Its 3D graph is given in Figure 1.

In order to compare the convergence rate, each algorithm was ruined for a fixed number of iterations = 15.We studied the path of the convergence for same number of iterations to analyze the pros and cons of each algorithm. The sub-level sets of the function is displayed in Figure 2.

$$f(x) = 0.2x_1^2 + 2x_2^2 \tag{1}$$

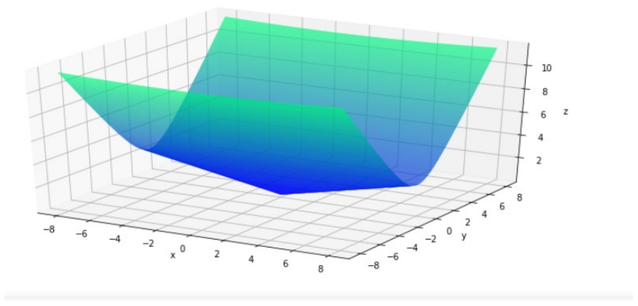

Fig. 1. 3D graph of Equation 1

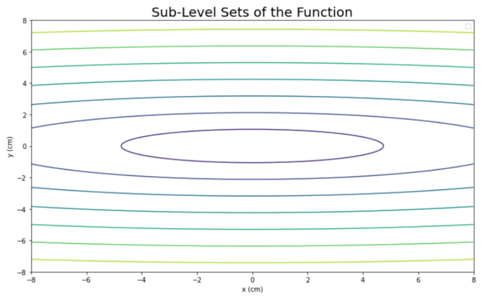

Fig. 2. Level Sets of Function 1

A. Momentum

Gradient descent with momentum is a quite commonly used optimizer which eliminates oscillation resulting from standard gradient descent and accelerates the converging optimum point. While it speeds up the horizontal direction, it slows down the vertical direction. With the help of that outstanding behavior, it enables to be taken a large step in the learning rate. In addition to this, momentum is much stabler than standard gradient descent.

The following equations give better intuition for the updated algorithm of momentum [7].

$$v_t = \rho v_{t-1} + \alpha \cdot g_t \tag{2}$$

$$w_{new} = w_{old} - v_t \tag{3}$$

where ρ and v are decay term and locally accumulated gradients, respectively, and α is learning rate.

To be more clear, the first three iterations are written as follows.

$$w_2 = w_1 - \alpha g_1 \tag{4}$$

$$w_3 = w_2 - \alpha\left(\rho g_1 + g_2\right) \tag{5}$$

$$w_4 = w_3 - \alpha\left(\rho^2 g_1 + \rho g_2 + g_3\right) \tag{6}$$

Based on the above equation, numerical comparison of standard gradient descent and gradient descent with momentum is given in Figure.8. For this analysis, learning rate, decay term, and number epoch are chosen as 0.4, 0.5 and 20, respectively.

From Figure 8, it is easily observed that momentum technique decrease oscillation and reaches the optimum point faster compared to standard gradient descent.

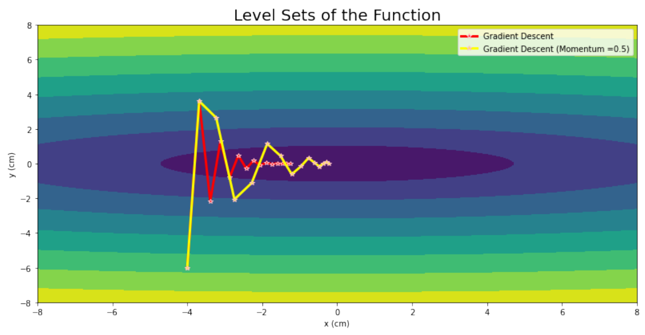

Fig. 3. Numerical comparison of standard gradient descent and gradient descent with momentum

B. Adagrad

The adaptive gradient descent algorithm is a kind of scholastic gradient descent. Its main difference is that adagrad utilizes different learning rates for each weight according to the importance of the parameters in the network. In other words, while unnecessary parameters are trained with a higher learning rate, important parameters are trained by the smaller learning rate to converge more stably. It leads to accelerate the algorithm without allowing the distortion. Updating formula is [4]

$$G_t = G_{t-1} - dw^2 \tag{7}$$

$$w_{new} = w_{old} - \frac{\alpha}{\sqrt{G_t + \epsilon}} \odot dw \tag{8}$$

where G_t is a diagonal matrix consisting of the sum of the squares of the past gradients and ϵ is a smoothing term.In addition, $\odot$ indicates matrix-vector product operation.

To observe the difference of adagrad from other optimizers, Equation 1 is implemented and the obtained comparison figure is provided in (4).

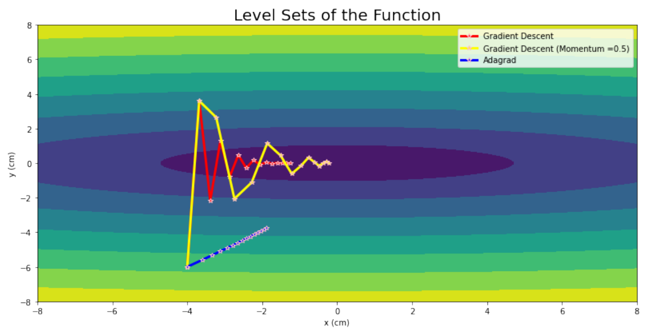

Fig. 4. Numerical comparison of gradient descent, momentum and, adagrad algorithms

As seen in Figure 4, although adagrad presents smooth movement without oscillations, its convergent ability is insufficient. To tackle this problem, a more efficient optimizer called RMSprop is introduced by Geoffrey Hinton.

C. RMSprop

RMSprop is another very efficient optimization algorithm using a moving average of squared gradients to normalize the gradient itself. Similar to the momentum technique, one of the superior features is to speed up gradient descent; it restricts the oscillations in the vertical direction and it, in turn, allows us to increase the learning rate and ensure to convergent the minimum point stably.

Regarding the mathematical background of RMSprop, a similar formulation to gradient descent with momentum is implemented as follows [8].

$$v_t = \rho v_{t-1} + (1-\rho)\cdot dw^2 \tag{9}$$

$$w_{new} = w_{old} - \frac{\alpha}{\sqrt{v_t + \epsilon}} \cdot dw \tag{10}$$

In the Equation 9, an exponential average of the square of gradient is calculated. The reason for using exponential average is to weight the more recent gradient updates more than the less recent ones.

By using the same Equation 1, the efficiency of the RMSprop is compared with gradient descent and momentum and level sets of this function are plotted in Figure 6. For this analysis, the decay rate is chosen as 0.9 and the rest of the parameters are the same with momentum analysis.

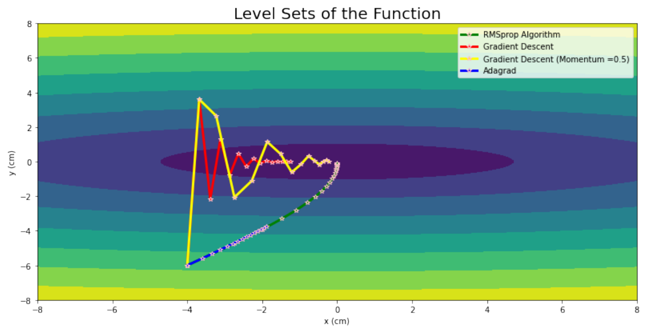

Fig. 5. Numerical comparison of gradient descent, momentum, adagrad, and, RMSprop algorithms

As obviously seen, when converging optimal point, while the gradient descent algorithm creates huge oscillations in the vertical direction, RMSprop restricts the vertical movement and speeds up the horizontal direction. Besides, although momentum decreases the oscillation as seen, RMSprop provides more reliable and faster convergence.

D. Adam

Adaptive Moment Estimation is another type of optimizer calculating adaptive learning rates for each parameter. It performs more robust and reliably compared to other optimizers because it basically combines momentum and RMSprop (i.e. moving average of gradient like momentum and using squared gradients to tune the learning rate like RMSprop [5]). To be more precise, adam algorithm performs as follows [8].

$$v_t = \rho_1 v_{t-1} + (1-\rho_1)\cdot dw \tag{11}$$

$$s_t = \rho_2 s_{t-1} + (1-\rho_2)\cdot dw^2 \tag{12}$$

$$\hat{v}_t = \frac{v_t}{1-\rho_1^t} \tag{13}$$

$$\hat{s}_t = \frac{s_t}{1-\rho_2^t} \tag{14}$$

$$w_{new} = w_{old} - \frac{\alpha \hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon} \tag{15}$$

where ρ_1 and ρ_2 are weighting parameters; $\hat{v}_t$ and $\hat{s}_t$ are normalized state variables.

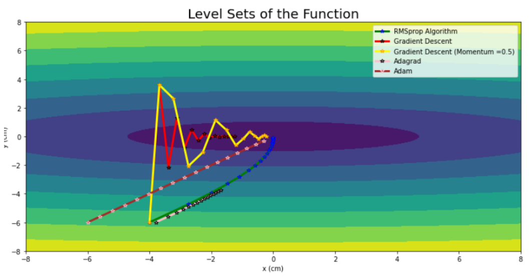

Fig. 6. Numerical comparison of gradient descent, momentum, adagrad, rmsprop and adam

As seen in the final figure the adam optimizers leads to the fastest convergence in a straight line path. In case of adam, it dampens oscillations by accumulated the sum over previous gradients (Equation 11) and it follows a straight line with due to the squared gradient term (Equation 12) analogous to rmsprop. This leads to a evident conclusion that adam is combined version of momentum and rmsprop.

IV. Neutral Network optimization and python implementation

Neural networks consists of several hidden layers where each layers is made up of several neurons superimposed by an activation function. The figure below represents the basic architecture of neural network. In our case, we will be using using these optimizers to minimize the loss for the image classification task. A set of input images along with their corresponding pixels as features will be feed into the neural network as input. Then, the loss will be computed between the actual labels and predicted labels. Further, this loss function will be optimized to compute the parameters. Convolution neural networks are special class of neural networks that deals with 3D images, they will be used here for this task.

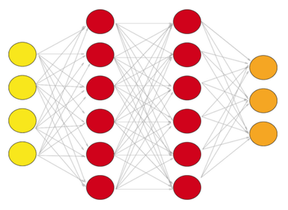

Fig. 7. Basic Neural Network

A. Dataset

In this paper, Fashion MNIST dataset available in Keras Library is used and different optimizers are compared experimentally based on their accuracy and loss. Fashion MNIST is a dataset consists of a training set of 60,000 image examples and a test set of 10,000 image examples. Each example is a 28x28 grayscale images. The first 20 training examples have been displayed below along with their labels [9].

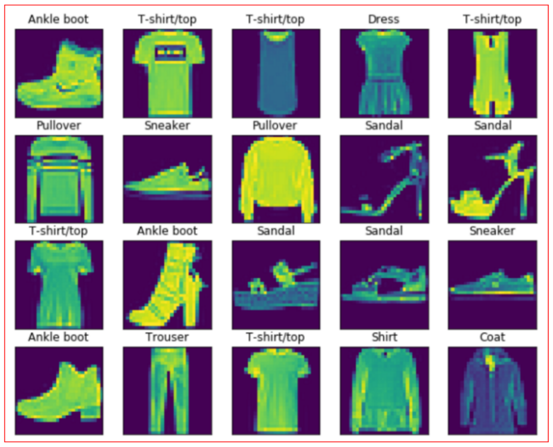

Fig. 8. Samples from fashion dataset

B. ConvNet Architecture

The ConvNet architecture used for our model has a 5x5 sized filter for the first convolution layer spread across 32 dimensions with stride size equal to 1, followed by another convolution layer with 3x3 filters spread across 64 dimensions.This layers was reduced down by a filter of size 3x3 into a max-pooling layer followed by a dense full connected layer of 32 neurons. Finally, the outputs was obtained across 10 neurons by passing logits through a softmax activation. The model was built with the keras deep earning library and trained using 12GB VRAM GPU. Rectifier Linear Unit (ReLU) which is nonlinearity activation function was also applied to each convolution layer [10]. A summary of the ConvNet, configuration applied in this study is outlined in Figure 9. In addition, the python used to develop the architecture is shown below.

```
model.add(Conv2D(32,(5,5), activation=”
    relu”,input_shape=(28,28,1)))
model.add(Conv2D(64,(3,3), activation=”
    relu”))
model.add(MaxPool2D(pool_size=(3,3)))
model.add(Dense(32,activation=”relu”))
model.add(Dense(10,activation=”softmax”))
```

C. Results and Discussions

Based on the above code, the fashion MNIST dataset is trained by the most popular optimizers which are scholastic gradient descent, gradient descent with momentum, adagrad,

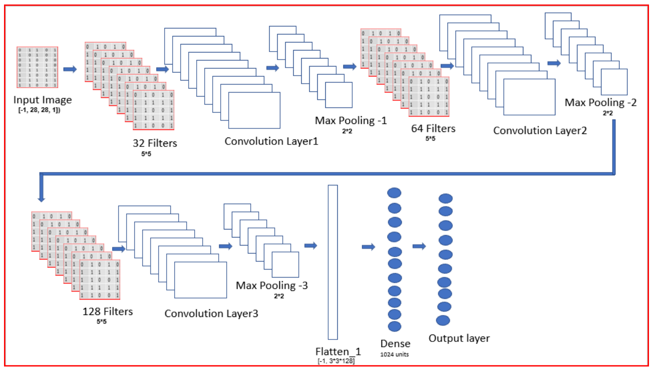

Fig. 9. Architecture of convolution neutral network

adadelta, RMSProp, adam. Their performance in terms of accuracy and loss is compared as given in Figure 10 and 11.

We trained each algorithm for fixed number of iterations (20) to study the convergence of these algorithms. We found that adam converges fastest followed by RMSprop and Momentum. On the other hand, SGD without momentum and adagrad showed a slow convergence. We know that gradient descent converges very slowly which is also very obvious in figures.

As seen in the figure, the adam optimizers leads to the fastest convergence in a straight line path. In case of adam, it dampens oscillations by accumulated the sum over previous gradients (Equation 11) and it follows a straight line with due to the squared gradient term (Equation 12) analogous to rmsprop. This leads to a evident conclusion that adam is combined version of momentum and rmsprop.

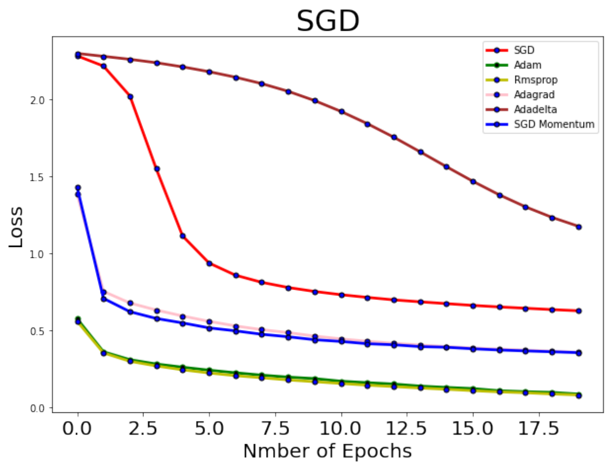

Fig. 10. Training and test accuracy of optimizers

From loss and accuracy graphs, it can be clearly observed that RMSprop and adam outperform with Fashion mnist dataset, but adam is a slightly better optimizer than RMSprop. In order to examine adam in detail, it is employed with different batch sizes.

Further, since we can not compute the gradient of all 60,000 training examples at once (it is computationally inefficient), it

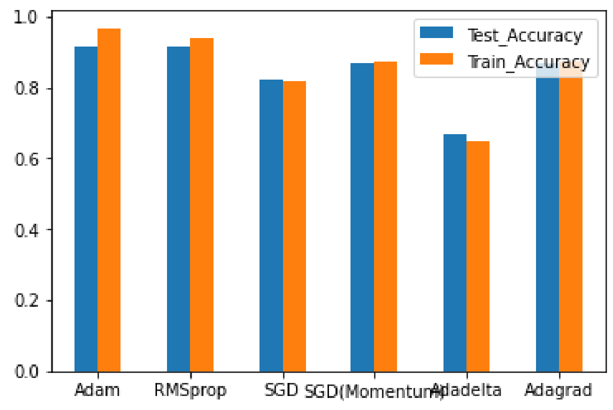

Fig. 11. Training and test accuracy of optimizers

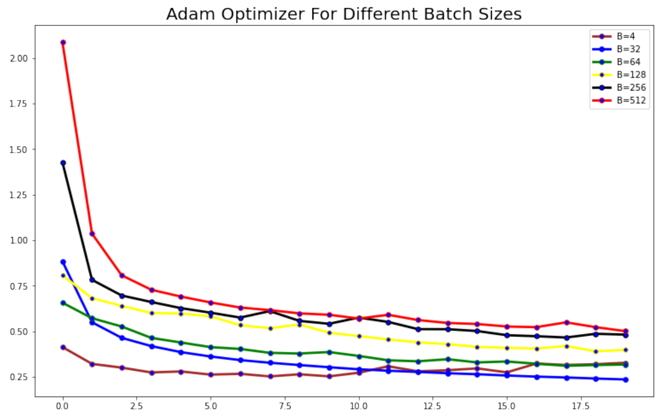

Fig. 13. Test loss of adam with for different batch size

is also necessary to find the effective batch in order to further speed up the convergence. Training and test Set losses both are plotted by sweeping batch size in power of two. However, for selection of best hyperparameter, we should look at the test set. Hence, we will be focusing on test set loss for selecting the optimum batch size.

We can see that B=4 converges fastest and the updates are very fast but noisy. They do not converge to the extreme minimum. Whereas, for very high Batch size (B=512), convergence is very slow [11], [12], but for B=32 it reaches to the extreme minimum as well with least number of update also being less noisy. As a result, the best size selected for this task would be B=16.

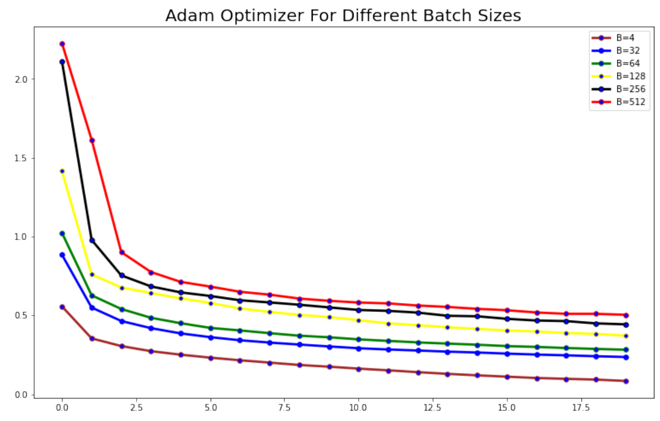

Fig. 12. Training loss of adam with for different batch size

D. Conclusions

In this paper, a comparative effect of five different optimization algorithms has been carried out for two tasks. First, these algorithms were used to find the optimal solution of a two variables quadratic function. Our findings revealed that in terms of number of iterations, Adam and RMSProp converged fastest. In the second part, these five algorithms were used to train MNIST dataset from keras libary. Our findings concluded that best optimization algorithm was Adam with the best batch size of 32.

E. Acknowledgement

We would like to thank Dr. Foad Sohrabi and Erfan Meskar for giving us great guidance throughout the project and course. The idea of providing numerical analysis of optimization algorithms given by Erfan was excellent. We were able dig deeper into the algorithms and were able to make this project more meaningful.

References

[1] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2933–2941, Curran Associates, Inc., 2014.
[2] R. Pascanu, Y. N. Dauphin, S. Ganguli, and Y. Bengio, "On the saddle point problem for non-convex optimization," *CoRR*, vol. abs/1405.4604, 2014.
[3] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," *arXiv preprint arXiv:1905.03817*, 2019.
[4] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 07 2011.
[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.
[6] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.
[7] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145 – 151, 1999.
[8] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.
[9] N. K. Manaswi, *Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras*. USA: Apress, 1st ed., 2018.
[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.
[11] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *CoRR*, vol. abs/1804.07612, 2018.
[12] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *CoRR*, vol. abs/1803.09820, 2018.