ADAPTIVE GRADIENT METHODS WITH DYNAMIC BOUND OF LEARNING RATE - REPRODUCIBILITY CHALLENGE

Michelle Abela, Sandeep Mistry, Jesus Adrian Rodriguez
School of Electronics and Computer Science
University of Southampton
Southampton, UK
{mna1u18, sbm1g18, jarr1g16}@soton.ac.uk

ABSTRACT

Adaptive optimisation methods such as AdaGrad, RMSProp and Adam are commonly employed in deep learning training phases, however have been found to exhibit poor generalisation and even convergence failure. AMSGrad was introduced to combat this, however it failed to achieve significant improvement over current methods. New variants of Adam and AMSGrad, called AdaBound and AMSBound, were introduced by Luo et al. (2019) and employed dynamic bounds on learning rates to achieve a smooth and gradual transition from adaptive methods to SGD. We attempted to reproduce their experimental results on various popular tasks and models, however only managed partial reproduction, specifically using feedforward and convolutional neural network models. We were unable to reproduce their results on recurrent neural network models. Our implementation can be found at https://github.com/COMP6248-Reproducability-Challenge/Adaptive-Gradient-Methods

1 INTRODUCTION

Adaptive optimisation methods are used to achieve rapid training processes. The proposed optimisation methods in this paper are the Stochastic Gradient Descent with Momentum (SGDM), Adam, AdaGrad, AmsGrad, AdaBound and AMSBound. The aim is to implement these different optimisers on multiple datasets, with various network types and architectures, to be able to evaluate the performance of each optimiser. We aim to study the effects of extreme learning rates on generalisation performance, and whether AdaBound and AMSBound suffer from the same flaws as Adam and AMSGrad. We conduct an empirical study of these methods on various popular tasks and models in computer vision and natural language processing. Further to this, an analysis between the original paper by Luo et al. (2019) and our results will be undertaken to assess the reproducibility of their work.

2 IMPLEMENTATION

A comparison of different optimisers, including SGDM, AdaGrad, Adam and AMSGrad was conducted. An extension to these were the AdaBound and AMSBound optimisers introduced in the conference paper. We experimented on four different datasets: MNIST and FashionMNIST grey scale images, CIFAR-10 RGB images and the text dataset Penn Treebank. The table below outlines these datasets, network types and architectures used when implementing the optimisers:

Our implementation used PyTorch to create the architectures and we added our own variants to improve the scope of our work, e.g., our simple MLP architecture also included a variant with dropout (0.2). Due to the extensive experimentation in the original paper, our implementation was heavily influenced by the authors'. We used their implementation of AdaBound and AMSBound on the CIFAR-10 dataset, along with their visualisation method. We introduced our own MLP and LSTM models and any other sources used for code are clearly referenced in our GitHub repository.

Table 1: Summary of models

Dataset	Network Type	Architecture
MNIST	Feedforward	1-Layer Perceptron
FashionMNIST	Feedforward	Simple MLP
CIFAR-10	Deep Convolutional	DenseNet121
CIFAR-10	Deep Convolutional	ResNet34
Penn Treebank	Recurrent	1-Layer LSTM
Penn Treebank	Recurrent	2-Layer LSTM
Penn Treebank	Recurrent	3-Layer LSTM

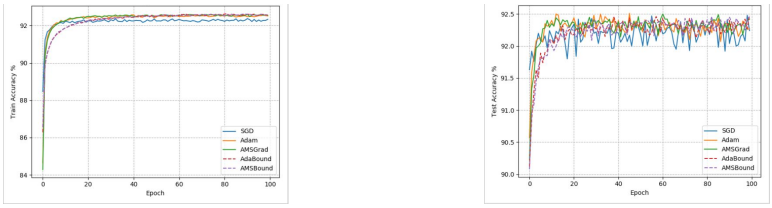
2.1 Hyperparameter Tuning

For all optimisation algorithms discussed, the hyperparameters stated in the paper were used as a starting point. We then finetuned them in an attempt to achieve the highest possible accuracy. The hyperparameters used were learning rates, betas, gamma, weight decay and batch size.

2.2 Feedforward Neural Network

Two fully connected neural networks were trained with a feedforward network type. The MNIST dataset was classified with a single-layer perceptron, while the Fashion MNIST dataset was classified with a simple Multi-Layer Perceptron (MLP) with no hidden layers, omitting the learning rate decay over 100 epochs.

Adam and AMSGrad tended to converge at a faster rate, whilst AdaBound and AMSbound outperformed all optimisers in the final accuracy metric. The MNIST average test accuracy from all the methods was between 92% and 92.5%. Results on the FashionMNIST dataset resemble the performance of the ones obtained with the MNIST dataset. However, in terms of test accuracy, FashionMNIST classification showed lower results.



(a) Train Accuracy

(b) Test Accuracy

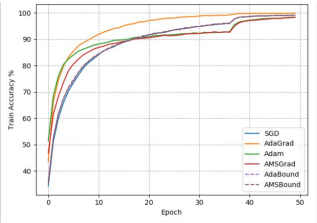
Figure 1: Train and Test Accuracies for Single-Layer Perceptron on MNIST dataset

2.3 Convolutional Neural Network

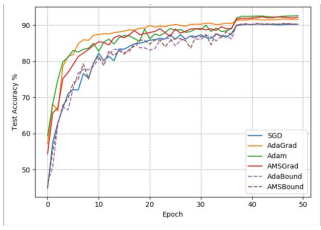
The CIFAR-10 dataset, which is made up of 60,000 RGB images and has 10 different classes, was classified using the ResNet-34 and DenseNet-121. The number of epochs was set to 50, with a reduction in the learning rate by 10 after 37 epochs, which is approximately 3/4 of the process, as in the paper.

DenseNet: We ran a DenseNet-121 model on CIFAR-10 and our results are shown in Figure 2. At epoch 37, we see a abrupt increase in train and test accuracies across all optimisers due to the learning rate reduction. AMSBound and AdaBound achieved higher accuracies than SGDM. AdaGrad, Adam and AMSGrad performed best overall.

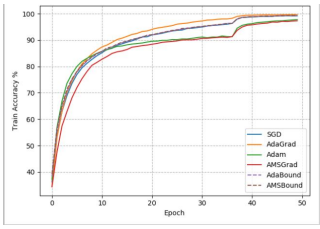
ResNet: We achieved similar results for the overall performance on ResNet-34 as we did on DenseNet-121.



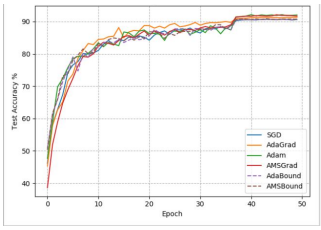
(a) DenseNet-121 Train Accuracy



(b) DenseNet-121 Test Accuracy



(c) ResNet-34 Train Accuracy

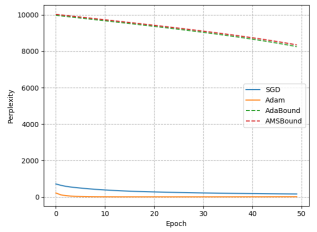


(d) ResNet-34 Test Accuracy

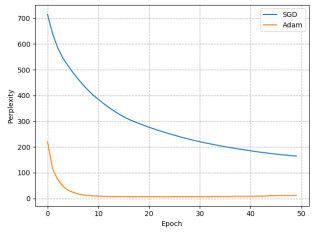
Figure 2: Train and Test Accuracies for DenseNet-121 and ResNet-34 on CIFAR-10

2.4 Recurrent Neural Network

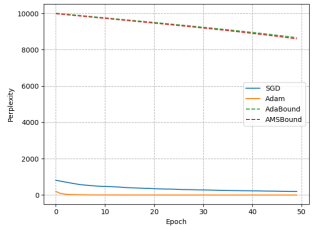
We conducted an experiment on the language modeling task with Long Short-Term Memory (LSTM) network. We ran three models with **(L1)** 1-layer, **(L2)** 2-layer and **(L3)** 3-layer LSTM respectively. We trained them on Penn Treebank, running for a fixed budget of 50 epochs. We used perplexity as the metric to evaluate the performance and report results in Figure 3.



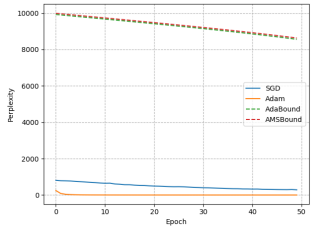
(a) L1: 1-Layer LSTM



(b) L1: 1-Layer LSTM (SGD and Adam)



(c) L2: 2-Layer LSTM



(d) L2: 3-Layer LSTM

Figure 3: Perplexity curves comparing SGD, Adam, AdaBound and AMSBound for the LSTM with different layers on Penn Treebank.

We observe that Adam performs the best in all models, however overfitting was most likely taking place. SGD has a slower initial progress and achieves a higher perplexity in our given epoch range. Both AdaBound and AMSBound perform very poorly, displaying extremely slow progress. In an attempt to improve our perplexity scores, we altered the hyperparameters for each model, however we did not notice any improvement. Comparing L1, L2 and L3, we note that performance of all models, but Adam, appear to noticeably degrade as more LSTM layers are added.

2.5 Analysis

We can confirm that we were able to partially reproduce the results of Luo et al. (2019) for feedforward neural networks on the MNIST dataset. This was extended to the FashionMNIST dataset, using a simple MLP, yielding a similar performance to that of MNIST, but with lower accuracies. For both datasets, using the authors' learning rates for AdaBound and AMSBound, we achieved poor performance and so we modified the learning rate, setting it at 0.1. We note that out final train and test accuracies were lower than the authors'. A possible reason for this discrepancy is that our Single-Layer Perceptron does not include an activation function.

Our train accuracies for the CIFAR-10 dataset for both DenseNet-121 and ResNet-34 generally agreed with the author's, however there were slight discrepancies in the test accuracies, resulting from the decreased number of epochs. The authors' original learning rates caused the two optimisers introduced in the original paper to perform poorly and so we implemented the same learning rate used in our feedforward neural networks. As expected, AdaBound and AMSBound outperformed SGDM, however surprisingly the adaptive methods, Adam, AMSGrad and AdaGrad outperformed the former.

We confirm that we could not reproduce the authors' results for the Penn Treebank dataset, using their hyperparameters. We achieved very low perplexity values using Adam, however for SGD they were higher than reported. Both AdaBound and AMSBound performed very poorly, achieving very high perplexity scores, which could not be explained by our reduced number of epochs. We also note that using the authors' recommended learning rate value of 10 for SGDM gave us extremely poor performance.

Our experimental results lead us to believe that the results of this paper are not easily reproducible. Indeed, the performances of AdaBound and AMSBound in our RNN implementation were far different to those reported in the paper and so it is possible that our implementation may need to be refined. The average training time expense increases with the complexity of the network. The Feedforward network had the lowest time expense, whilst the three-layer LSTM had the highest.

3 Future Work

To further analyse adaptive gradient methods, future work could look to implement other optimisation methods, including Nesterov momentum, RMSProp and AdaMax. Different architectures could be used with deeper neural networks and an increased number of epochs, although this will be more computationally expensive. A larger range of datasets could be experimented with, particularly for language modelling tasks. A wider range of hyperparameters and loss functions could be experimented with.

4 Conclusion

We investigate the reproducibility of Luo et al. (2019). Based on our experimentation, we find that the results from the author's feedforward and convolutional neural network models are partially reproducible, specifically that AdaBound and AMSBound outperform SGDM. We were unable to reproduce their results for recurrent neural network models, and obtained wildly conflicting results for all LSTM models. We recommend that the authors release the entire code of their implementation, to allow for easier reproducibility.

Implementation was computationally expensive, therefore we required the use of a GPU server, even though we used a reduced number of epochs. The feedforward and convolutional neural network models were easier to reproduce, whilst the recurrent neural networks where more challenging, as evidenced by our conflicting results.

References

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.