Final Project of Applied Machine Learning (COMP 551)
Track 3: Reproducibility Challenge
"ON THE CONVERGENCE OF ADAM AND BEYOND"*

Barleen Kaur[1], Jacob Shnaidman[2] and Hamed Layeghi[3]

***Abstract*— In this project, we try to reproduce the results from a recent paper in ICLR 2018 which is titled "ON THE CONVERGENCE OF ADAM AND BEYOND". The paper introduces some convergence problems with existing stochastic optimization methods and in particular ADAM and proposes the concept of using "long term memory" in their new optimization method which is called AMSGRAD. While the majority of the paper focuses on the theoretical part of AMSGRAD with respect to ADAM, we focused on reproducing the experiments performed to illustrate the advantage of AMSGRAD over ADAM.**

I. INTRODUCTION

In this report, we will try to reproduce the results from [1]. The paper describes an important flaw in some of the prevailing stochastic optimization methods such as ADAM [2], ADAGRAD [3], RMSPROP [4], ADADELTA [5] and NADAM [6] that are based on using gradient updates scaled by the exponential moving average of the squared past gradients. It has been shown in application that the exponential moving average method used in these algorithms leads to divergence or convergence to a wrong minimum. This could happen especially for high dimensional networks with large output spaces.

A precise analysis is performed by the authors to illustrate the convergence issues of ADAM for any values of $\beta_1, \beta_2 \in [0, 1)$. In particular, a synthetic example is produced that contradicts the convergence theorem in the original paper of Adam by Kingma and Ba in 2014 [2]. Most of the paper is devoted to this analysis and the underlying proofs for non-convergence of ADAM in some special cases and convergence of AMSGRAD in the same setting.

Our main focus was on the reproduction of the experimental results of the paper which compares the performance of ADAM and AMSGRAD. These experiments included a synthetic example, classification of MNIST dataset using logistic regression and a single-layer feedforward neural network and also classification of CIFAR-10 using a convoluted neural network named CIFARNET.

[1]Barleen Kaur (Student ID: 260783838) is with Department of Computer Science, McGill University, Montreal, QC, Canada barleen.kaur@mail.mcgill.ca

[2]Jacob Shnaidman (Student ID: 260655643) with the Department of Electrical Engineering, McGill University, Montreal, QC, Canada jacob.shnaidman@mail.mcgill.ca

[3]Hamed Layeghi (Student ID: 260524499) with the Department of Electrical Engineering, McGill University, Montreal, QC, Canada hamedl@cim.mcgill.ca

Algorithm 1 Generic Adaptive Method Setup

Input: $x_1 \in \mathcal{F}$, step size $\{\alpha_t > 0\}_{t=1}^{T}$, sequence of functions $\{\phi_t, \psi_t\}_{t=1}^{T}$
for $t = 1$ **to** T **do**
  $g_t = \nabla f_t(x_t)$
  $m_t = \phi_t(g_1, \ldots, g_t)$ and $V_t = \psi_t(g_1, \ldots, g_t)$
  $\hat{x}_{t+1} = x_t - \alpha_t m_t / \sqrt{V_t}$
  $x_{t+1} = \Pi_{\mathcal{F}, \sqrt{V_t}}(\hat{x}_{t+1})$
end for

Algorithm 2 AMSGRAD

Input: $x_1 \in \mathcal{F}$, step size $\{\alpha_t\}_{t=1}^{T}$, $\{\beta_{1t}\}_{t=1}^{T}$, β_2
Set $m_0 = 0$, $v_0 = 0$ and $\hat{v}_0 = 0$
for $t = 1$ **to** T **do**
  $g_t = \nabla f_t(x_t)$
  $m_t = \beta_{1t} m_{t-1} + (1 - \beta_{1t}) g_t$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$ and $\hat{V}_t = \mathrm{diag}(\hat{v}_t)$
  $x_{t+1} = \Pi_{\mathcal{F}, \sqrt{\hat{V}_t}}(x_t - \alpha_t m_t / \sqrt{\hat{v}_t})$
end for

Fig. 1. Generic Adaptive Method Setup and AMSGRAD Algorithms [1]

II. OPTIMIZATION ALGORITHMS

A. Online Optimization Problem

The paper considers an online optimization setting with full information feedback. In this setting, optimization parameter $x_t \in \mathcal{F}$ where $\mathcal{F} \subset \mathbb{R}^d$ is the set of feasible points. There exists a loss function $f_t(x_t)$ for which the following regret function is introduced after a set of T steps:

$$R_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{F}} \sum_{t=1}^{T} f_t(x) \tag{1}$$

It is assumed that $\mathcal{F}$ has bounded diameter and $\|\nabla f_t(x)\|$ is bounded for all $t \in [T]$ and all $x \in \mathcal{F}$.

B. ADAM *vs* AMSGRAD

Many of the stochastic optimization methods such as Vanilla SGD, ADAGRAD, RMSPROP, ADAM could be expressed in terms of the generic adaptive setting introduce in Algorithm 1 in Figure 1. The functions ϕ_t and ψ_t are the "averaging" functions that define the algorithm.

For ADAM, we have

$$\phi_t(g_1, \ldots, g_t) = (1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}g_i$$
$$\psi_t(g_1, \ldots, g_t) = (1-\beta_2)\text{diag}\left(\sum_{i=1}^{t}\beta_2^{t-i}g_i^2\right) \tag{2}$$

It is shown in the paper, that the above exponential averaging over the past squared gradients could lead to divergence or convergence to non-optimal solutions.

AMSGRAD which is shown in Algorithm 2 of Figure 1 tries to resolve this issue by scaling with the maximum of averaged past square gradients. This gives the algorithm a "long term memory" which resolves the occasional convergence issues of other stochastic optimization methods and plus improves their performance and some of the common application.

C. Synthetic Example

Consider the following loss function

$$f_t(x) = \begin{cases} Cx, & \text{for } t \text{ mod } 3 = 1 \\ -x, & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{F} = [-1, 1]$.

For large enough T, the total loss is given by

$$\begin{aligned}\sum_{t=1}^{T} f_t(x) &= f_1(x) + f_2(x) + \ldots + f_T(x) \\ &= Cx - x - x + Cx - x - x + \ldots \\ &\simeq \frac{T}{3}(C-2)x\end{aligned} \tag{4}$$

Hence, for $C > 2$, $x = -1$ is a minimizer for the total loss. The regret is now given by

$$R_T(x) \simeq \frac{T}{3}(C-2)(x+1), \quad x \in [-1, 1] \tag{5}$$

which has a minimum at $x = -1$. This example illustrates an underlying problem with ADAM'S convergence theorem in [2]. ADAM, for this problem converges to $x = 1$ which creates the maximum regret and by equation (5), the average regret, R_T/T, does not go to zero. Figure 2 shows the results obtained in the main paper [1].

III. IMPLEMENTATION GOALS

The primary goal of our project is to reproduce the AMSGRAD algorithm and the performance comparisons between ADAM and AMSGRAD. To do this, we set out to reproduce the performance of AMSGRAD and ADAM on the deterministic and stochastic synthetic experiments, logistic regression example, feed forward neural network example, and the convolutional neural network example.

To ensure that the central claims of the paper are reproducible, we reproduced both synthetic experiments. These experiments should show that AMSGRAD converges to -1 and asymptotically approaches zero average regret, while ADAM fails to converge to -1 and does not approach zero average regret. Because both the deterministic and stochastic synthetic experiments are exemplary of different theorems produced within the paper, they were both important to reproduce.

We also aim to reproduce the other graphs shown in figure 2 of the paper, since they make substantial claims about the performance of AMSGRAD in comparison to ADAM in many different contexts. They claim that it has better performance than ADAM not only in classification using logistic regression, but also using feed-forward neural networks and convolutional neural networks. It's important to verify that the outcomes of these experiments are valid in order to see that the relative performance of AMSGRAD in comparison to ADAM can be generalized.

Although the authors provide proofs that there are cases where ADAM does not converge to zero average regret and does not converge to the optimal solution, the proof for AMSGRAD only says that there is an the upper bound on its average regret. The proofs alone are insufficient to claim that AMSGRAD will perform better than ADAM in practice. Therefore, it is important to verify in the different experiments presented in the paper that AMSGRAD does have improved performance compared to ADAM. Since there is already some theoretical proof of the synthetic experiments, it is more important to be able to reproduce the set of experiments that operate on real data with common methods like neural networks, logistic regression, etc. For completeness, we have reproduced all the graphs shown in the report for all experiments.

The paper also mentions another alternative extension to ADAM algorithm called ADAMNC [1]. However, we chose to focus only on the main proposed AMSGRAD algorithm and save the implementation of the extension algorithm for future work. For similar reasons, we did not implement any bias correction in our implementation.

IV. DISCUSSION

It should be noted that the authors did not mention the loss function that they used in any of their experiments. We assume that they used categorical cross-entropy loss for all experiments on real datasets as this is a common loss function for classification.

Also, all the following reproductions were done with an implementation of AMSGRAD that does not have any bias correction.

A. Deterministic Synthetic Experiment

The first synthetic experiment was replicated with

$$f_t(x) = \begin{cases} 1010x, & \text{for } t \text{ mod } 101 = 1 \\ -10x, & \text{otherwise} \end{cases} \tag{6}$$

as seen in the experiments section of the paper. We clamped the output of the x variable between -1 and 1 as prescribed.

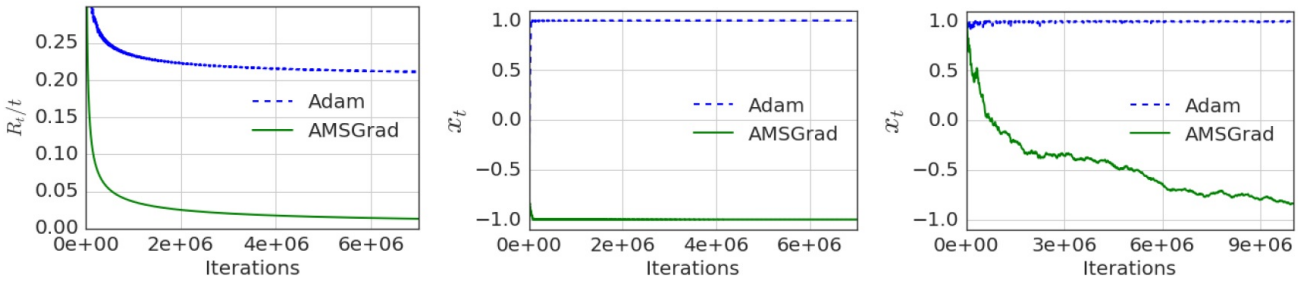

Fig. 2. Original Results for Synthetic Example. The left plot is showing the average regret of both optimizers in the online setting. The center plot shows the convergence of x_t in the online setting. The right plot shows the convergence of x_t in the stochastic setting. [1]

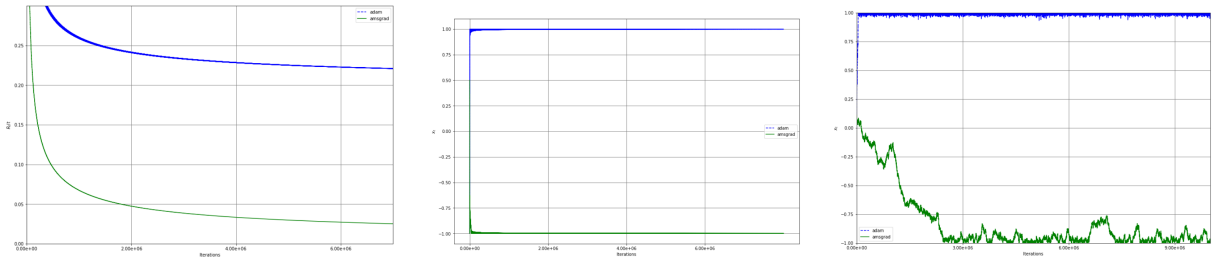

Fig. 3. Reproduction of Synthetic Experiments. The first

We used the described beta values of β_1 = 0.9 and β_2 = 0.99. For epsilon, the value was not mentioned in the paper, so a value of 10^{-8} was used. We varied the learning rate between 10^{-5} and 1 and tried with and without a decaying learning rate of $\frac{\alpha}{\sqrt{t}}$ for each iteration t. After hyper-parameter tuning, the learning rate used to reproduce the graphical results shown in the two left-hand plots of figure 3 was $\alpha_t = \frac{1}{\sqrt{t}}$

B. Stochastic Synthetic Experiment

The second synthetic experiment was replicated with

$$f_t(x) = \begin{cases} 1010x, & \text{with probability } 0.01 \\ -10x, & \text{otherwise} \end{cases} \tag{7}$$

For this, we used the same betas as in the first synthetic example (β_1 = 0.9 and β_2 = 0.99). Epsilon remained at 10^{-8}. The methodology to obtain the results of the stochastic synthetic experiment were very similar to that which was described above for the deterministic experiment. Although the theorems imply a learning rate of $\alpha_t = \frac{\alpha}{t}$, the learning rate was not specified. Since the learning rate seemed sensitive, both alternating and non alternating learning rates were attempted. A small learning rate was important to be able to reproduce the graph, since higher learning rates may diverge in the wrong direction. Our best results were with the static learning rate, which is shown in figure 3 on the right-hand side. This was done with $\alpha_t = 5 \times 10^{-4}$.

C. Logistic Regression

Logistic regression is used to investigate the performance of AMSGRAD versus ADAM in a convex optimization classification problem for the MNIST dataset. The dataset includes handwritten numbers in 28×28 pixel gray-scale images resulting in a 784 dimensional image vector which are mapped to the corresponding 10 class labels. The step size was decaying with $\alpha_t = \alpha/\sqrt{t}$. The weighting parameters were selected as $\beta_1 = 0.9$ and $\beta_2 \in [0.9, 0.999]$. The best value of hyper-parameters α and β_2 were found using a grid search. The original results from the authors are shown in Figure 4. As can be seen, only 5000 iterations have been used in both Logistic Regression and Feed-Forward Neural Network. This suggests that the used learning rates could be large.

In our experiment, we used a batch size of 128 with 391 batches in the training set and 79 batches in our validation set. Because they did not specify the sizes of their training and validation sets, we made use of the entire MNIST dataset that was given by the PyTorch Torchvision library. For AMSGRAD, a grid search was performed over hyper-parameter pairs (α, β_2) which spanned logarithmically spaced step sizes with $\alpha \in [0.0001, 1]$ and linearly spaced values for β_2 with $\beta_2 \in [0.99, 0.999]$. The final validation loss for each pair was found and the pair which minimized this value was selected as the optimal hyper-parameter.The selected hyper-parameters are $\alpha_t = \frac{0.005}{\sqrt{t}}$ and $\beta_2 = 0.999$ for AMSGRAD and for ADAM.

Figures 5 and 6 show the reproduced results for the train and test loss of logistic regression using both AMSGRAD

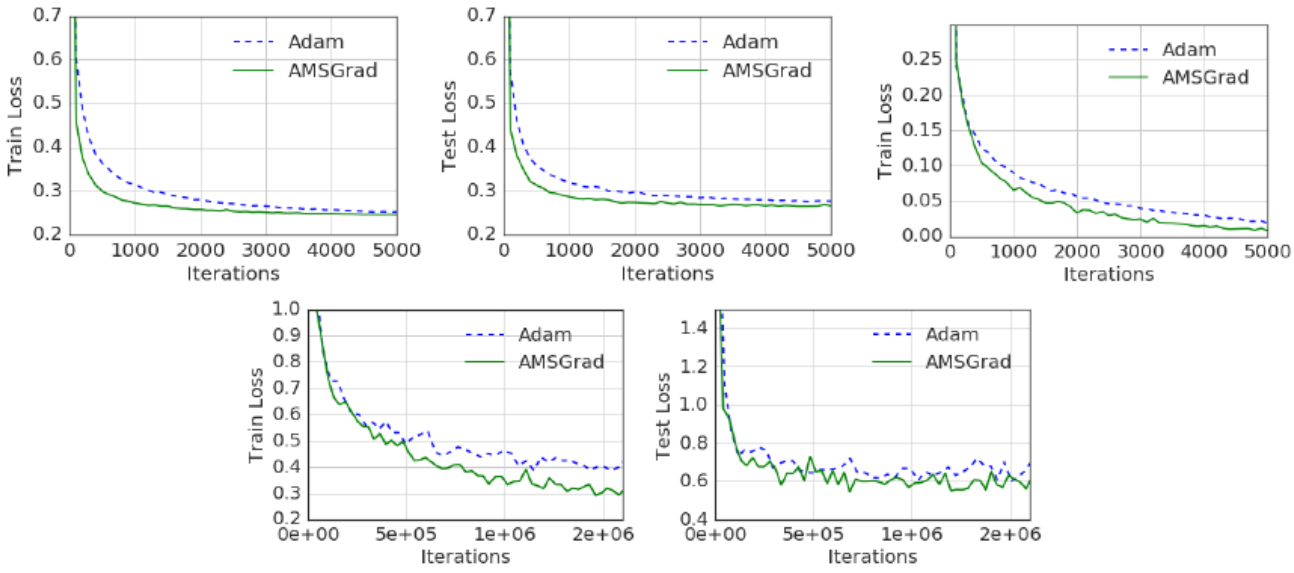

Fig. 4. Performance comparison of ADAM and AMSGRAD for logistic regression, feedforward neural network and CIFARNET. The top row shows results for logistic regression (left and center) and 1-hidden layer feedforward neural network (right) on MNIST. In the bottom row, the two plots compare the training and test loss for CIFARNET [1].

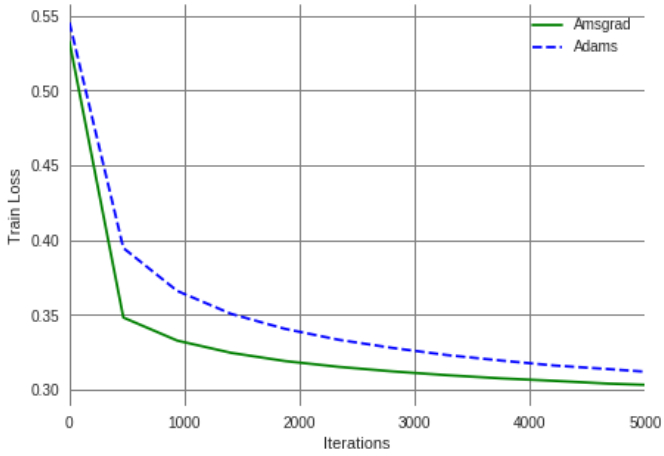

Fig. 5. Train loss comparison between AMSGRAD and ADAM for Logistic Regression on MNIST

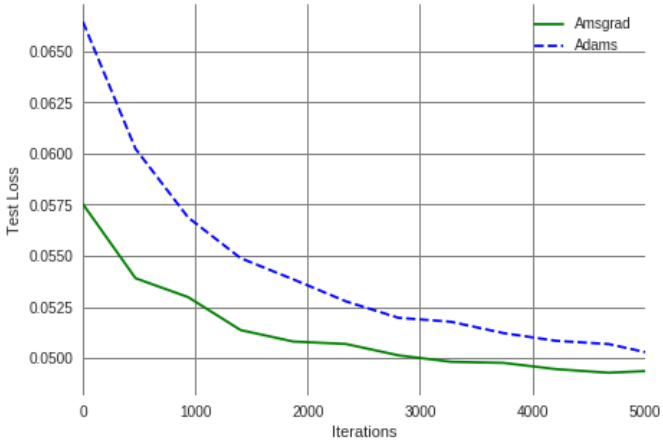

Fig. 6. Test loss comparison between AMSGRAD and ADAM for Logistic Regression on MNIST

and ADAM.

D. Feed-Forward Neural Network

The feed forward neural network experiment aimed to classify the MNIST dataset. As was done in the logistic regression experiment, the entire MNIST dataset available through the PyTorch Torchvision library was used. The neural network was described quite specifically; it uses a single fully connected hidden layer with 100 rectified linear units (ReLU). This implies an input layer of 784 units and an output layer of 10 units. The β_1 used in the paper was fixed to 0.9 while β_2 was found using a grid search between 0.99 to 0.999. The learning rate was specified to be some constant $\alpha_t = \alpha$. The batch size was left unspecified, however it was implied that the hyper-parameters were similar to the logistic regression experiment which used a batch size of 128. We inferred this to mean that they used the same batch size of 128.

We found that the best validation performance was with a value of $\alpha_t = \alpha = 10^{-3}$ on both ADAM and AMSGRAD and with $\beta_2 = 0.995$ for AMSGRAD and $\beta_2 = 0.994$ for ADAM as shown in Figure 7. This was found by doing a grid search of different learning rates logarithmically spaced between 10^{-1} to 10^{-5} and 10 different $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$. Although we computed a grid search of the hyper parameters as recommended, the grid search proved computationally

expensive, and we were unable to replicate the performance of either optimizer completely. Although the training loss was not as low as in the paper, we were able to reproduce the relative performance of AMSGRAD compared to ADAM.

A bar graph of our hyper-parameters compared to other hyper-parameters found during a grid-search can be seen in figures 22 and 23 in the appendix.

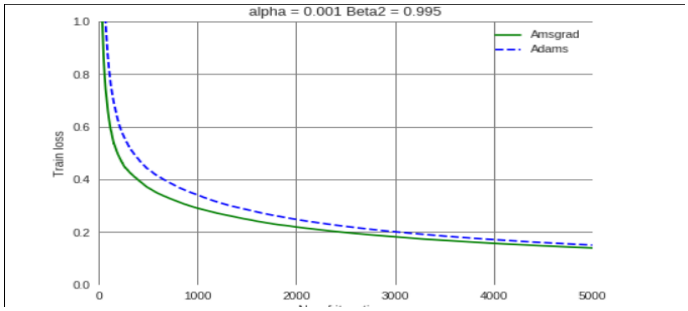

Fig. 7. Results for Feed-Forward Neural Network

E. CIFARNET

The performance of AMSGRAD and ADAM algorithms were compared in a multiclass image classification problem on the standard CIFAR-10 dataset which consists of 50,000 training examples and 10,000 test examples, each of size 32×32 pixel colored images. Since the exact architecture of CIFARNET, a Convolutional Neural Network (CNN), was not clearly defined in the paper (like stride, padding size, number of max-pool layers, activation function etc), we experimented with two architectures of CIFARNET which are shown in Figure 8. Here, the CIFARNET architecture at the top refers to the one based on our understanding from the paper and the second is the one used in TensorFlow[1]. For simplicity, we address the first CIFARNET architecture as CIFARNET-1 and TensorFlow's one as CIFARNET-2. Both the architectures used two convolutional layers with 64 channels, 2×2 max pooling layers with batch normalization between the convolutional layers, which was followed by two fully connected layers of size 384 and 192. Dropout layer with p=0.5 was also kept in between the fully connected layers as prescribed in the paper. For both CIFARNET architectures, we set the strides of 1, padding of 2 and used ReLU activation as it solves the vanishing gradient problem as shown in literature [7]. However, there are few differences in both the architectures which are shown in Figure 9. The implementation of both CIFARNET architectures was done using PyTorch [2]. We assumed that the step size was kept constant $\alpha_t = \alpha$ for CIFARNET as the authors mentioned to keep constant step size throughout all experiments for training neural networks.

Since, the range of hyper-parameters wasn't clearly mentioned in the paper, we performed hyper-parameter tuning of

[2]TensorFlow's CIFARNET model
[2]http://pytorch.org/

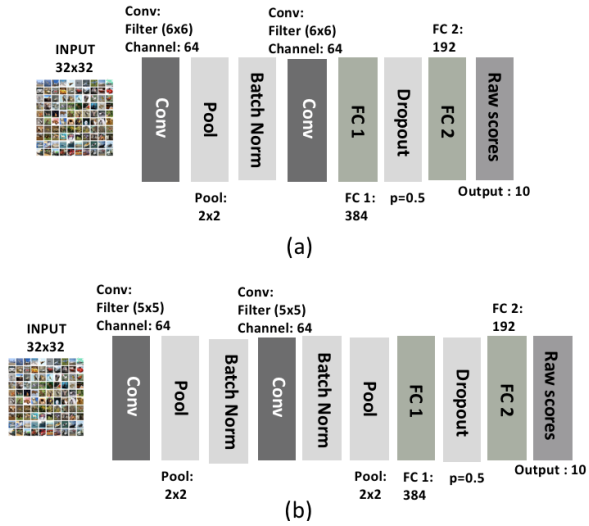

Fig. 8. The two CIFARNET architectures used. Here (a) represents the one based on our understanding of the base paper and (b) is TensorFlow's CIFARNET architecture

Architecture	Kernel size	(Number of pool and BN layers)	Order of BN and Pool layer	Stride, padding, Activation function
CIFARNET-1	(6,6)	(1,1)	Conv1-Pool-BN-Conv2	1,2,ReLU*
CIFARNET-2	(5,5)	(2,2)	Conv1-Pool-BN-Conv2-BN-Pool	1,2,ReLU

Fig. 9. Differences in CIFARNET architectures. Here, BN refers to Batch normalization. Items marked (*) are our assumptions as they weren't specified in the paper

α and β_2 using grid search on a wider range and then narrowed the range to $\alpha \in \{\text{np.geomspace}(1e{-}4, 1e{-}6, \text{num} = 7)\}$ and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ which resulted in a total of 49 hyper-parameter combination pairs (α, β_2) to explore. For this, we randomly sampled 5000 training examples and 1000 validation examples from our original training set of 50,000 examples ensuring that no training example was present in the validation set. For every hyper-parameter combination, we ran our models for 15 epochs only with a batch size of 128 due to time and computational constraints. Minimum validation loss was calculated for each combination and the pair which resulted in least validation loss with a non-increasing trajectory of validation loss with epochs was chosen as the best value for the hyper-parameters (α, β_2). We experimented with the first best value for hyper-parameters with $\alpha = 1e - 4$. However, it was observed that the models were over-fitting the training data in a lesser number of epochs. This indicated to us that we needed to use an even smaller learning rate. Also, since we were considering only 15 epochs for every combination pair in hyper-parameter tuning, it was obvious that the biggest α's in our selected range would produce the least validation loss compared to smaller α's if we train the model for just 15 epochs. Considering all these above factors, the best hyper-parameter values found were $\alpha = 2.1e - 05$ and $\beta_2 = 0.999$ for AMSGRAD and $\alpha = 2.1e - 05$ and $\beta_2 = 0.998$ for

ADAM for both CIFARNET architectures. The results of hyper-parameter tuning are shown in the appendix section.

With the best hyper-parameter values obtained, we trained our CIFARNET models using the original 50,000 training examples for 70 epochs with batch size of 128 and computed loss on training set and test set as shown in Figures 10, 11, 12 and 13. It can be seen from the results for both architectures that AMSGRAD performs better in terms of train loss than ADAM. However, the test loss increases with the number of iterations due to the models over-fitting. Although the graphs are not exact replications of the results of CIFARNET as in the paper, still they strongly indicate the success of AMSGRAD optimizer over ADAM.

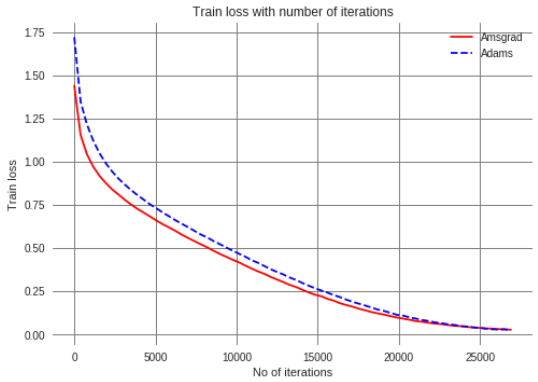

Fig. 10. CIFARNET-1: Train Loss with number of iterations

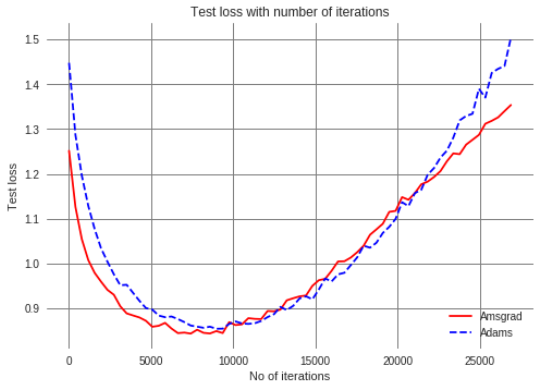

Fig. 11. CIFARNET-1: Test Loss with number of iterations

V. CHALLENGES

A. Implementation

The implementation of the AMSGRAD algorithm was fairly straightforward since it is not very different from the ADAM algorithm. Initially, we had made the mistake of not storing the maximum moving average of the squared gradients properly. This led to an implementation that was very similar to ADAM. At first glance, it seemed like the performance ran fine with logistic regression and our neural network example, so we assumed we had a bug in our

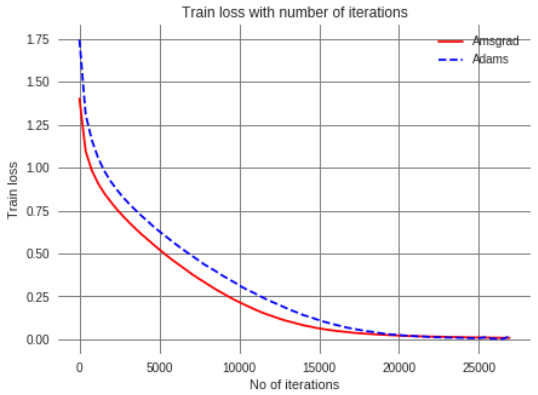

Fig. 12. CIFARNET-2: Train Loss with number of iterations

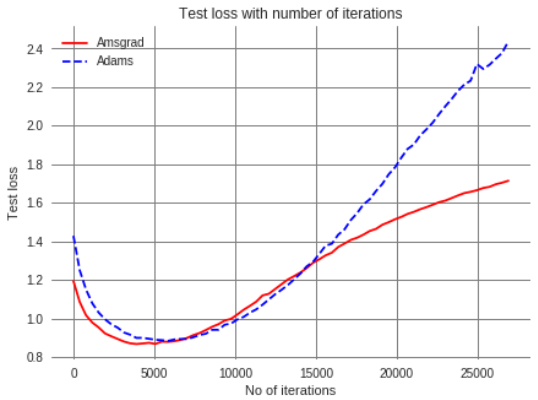

Fig. 13. CIFARNET-2: Test Loss with number of iterations

reproduction of the synthetic example. After some time, we noticed the bug, and started over. For future reproductions, the implementation should be checked first with the synthetic example to verify that it works.

B. Synthetic Experiments

The synthetic experiments were very sensitive to the adjusted learning rate. The proofs in the paper assumed an adjusted learning rate of $\alpha/\sqrt{t}$, however the synthetic experiments did not specify their learning rates. Many learning rates had to be tried and manually verified to see whether or not it could reproduce the graphs shown in the paper.

Another challenge that was faced in trying to reproduce the graphs for the synthetic experiments was that the horizontal axis of the graphs needed to be logarithmically scaled in the exact way as shown in the paper to appear non-noisy. This may be in part because the learning rate was not specified in the paper to reproduce these graphs, which may have resulted in a optimizer with higher variance than what the authors had when they produced their graphs originally. In larger part, this can be attributed to the frequent shifts in the magnitude and direction of the gradient of the loss function. Future reproductions of this paper should be aware of this detail.

C. Logistic Regression

For logistic regression, we had some trouble with hyper-parameter tuning because some of the settings were not clearly explained. In particular, as lower learning rates normally lead to a better solution, the primary challenge was to find the minimum value of the learning rates used in the experiment.

D. Feed-Forward Neural Network

The feed forward neural network was specified almost in full detail with instructions on how to attain the unspecified hyper-parameters. Unfortunately these hyper-parameters proved very costly to cross validate, and although much time was spent on cross validation, it seems that attaining a training loss as low as 0.01 after only 5000 iterations as seen in their original graph proved very difficult.

E. CIFARNET

The lack of clarity regarding the architecture of CIFARNET model used in the base paper led us to perform experimentation on two different architectures: CIFARNET-1 and CIFARNET-2. A lot of effort and time was invested in performing hyper-parameter tuning for both these models. From the paper, it seems that the authors used a very small learning rate for training the CIFARNET model as the number of iteration are too high, 2e+06 and still the model didn't converge fully. It was not possible for us to run our model for such high number of iterations due to time and computation constraints. It was also observed from experimentation that even using smaller learning rates in order of 1e-5 to 1e-6 resulted in over-fitting of the model and caused the test loss to increase after some iterations. Also, it is strange that the authors chose to keep alpha constant while training CIFARNET as it is reported in the literature [8] that annealing the learning rate while training provides better convergence.

VI. CONCLUSIONS

In this project, we were able to reproduce the results of the experiments in [1] to variable degrees.

For all experiments, the tuning of hyper-parameters for the optimizers played a key role in reproducing the results.

For the synthetic experiment, the results in our work match perfectly the ones in the paper and it clearly illustrates a basic flaw in ADAM with respect to AMSGRAD.

The results of the logistic regression experiment were successfully reproduced. AMSGRAD successfully outperformed ADAM on the MNIST dataset.

The results of the neural network supported the claims the authors made about the relative performance of AMSGRAD compared to ADAM; however, it proved computationally unfeasible to reproduce the absolute performance as seen in the graphs in figure 2 from the paper on the MNIST dataset.

Since the number of iterations for which the authors ran their CIFARNET model is too high, of the order 2e+06, it was unfeasible for us to exactly replicate their results due to time and computation constraints. However, our results clearly indicate the success of AMSGRAD optimizer over ADAM. It was also observed that using a smaller learning rate on the order of 10^{-5} to 10^{-6} resulted in the over-fitting of the models and caused the test loss to increase after some iterations. Moreover, we observed from experimentation that the authors might have used a slightly smaller learning rate for ADAM than AMSGRAD.

VII. SUMMARY OF CONTRIBUTIONS

Barleen Kaur (260783838) worked on the implementation of AMSGRAD algorithm, created the structure of codes for all simulations on real datasets, worked on reproducing CIFARNET results, helped with implementation of Logistic Regression and helped with hyper-parameter tuning for all the experiments.

Jacob Shnaidman (260655643) - helped with implementation of algorithm, created visualizations for presentation, reproduced synthetic experiments, hypertuned and reproduced the graphs for the feed-forward Neural Network.

Hamed Layeghi (260524499) helped with the theoretical part of the paper on AMSGRAD, derived the analytic equations for the synthetic problem, helped with coding and simulation of Logistic Regression, helped with hyper-parameter tuning of all experiments, wrote most parts of Introduction and Conclusion.

All three members contributed in writing the report.

"We hereby state that all the work presented in this report is that of the authors."

REFERENCES

[1] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

[2] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[3] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[4] T. Tieleman and G. Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. 2012. [Online]. Available: https://www.coursera.org/learn/neural-networks/lecture/YQHki/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude

[5] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[6] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. Equation 2.7.

VIII. APPENDIX

A. CIFARNET

Some of the additional graphs that are generated during the experiments are reported here.

The results of hyper-parameter tuning on a wider set for CIFARNET-1 architecture for $\alpha \in \{1e{-}2, 1e{-}3, \ldots, 1e{-}6\}$ and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for AMSGRAD and ADAM algorithms are shown in Figures 14 and 15 respectively. Figures 16 and 17 describe the results for hyper-parameter tuning on a wider set for CIFARNET-2 architecture for $\alpha \in \{1e-2, 1e-3, \ldots, 1e-6\}$ and

$\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for AMSGRAD and ADAM algorithms respectively.

The results of hyper-parameter tuning for CIFARNET-1 architecture for $\alpha \in \{np.geomspace(1e-4, 1e-6, num = 7)\}$ and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for AMSGRAD and ADAM algorithms can be seen in Figures 18 and 19 respectively. Figures 20 and 21 represent the results of hyper-parameter tuning for CIFARNET-2 architecture for $\alpha \in \{np.geomspace(1e-4, 1e-6, num = 7)\}$ and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for AMSGrad and ADAM algorithms respectively.

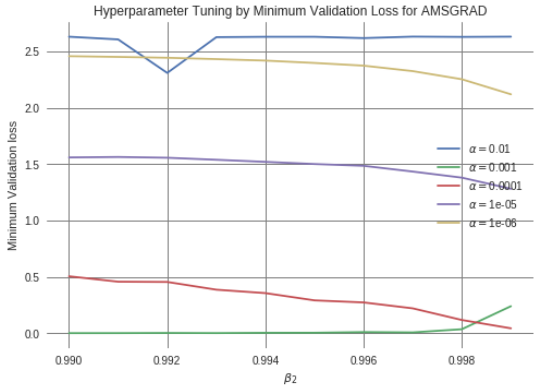

Fig. 14. CIFARNET-1: Hyper-parameter tuning for $\alpha \in \{1e-2, 1e-3, \ldots, 1e-6\}$ (wider set) and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for AMSGrad algorithm

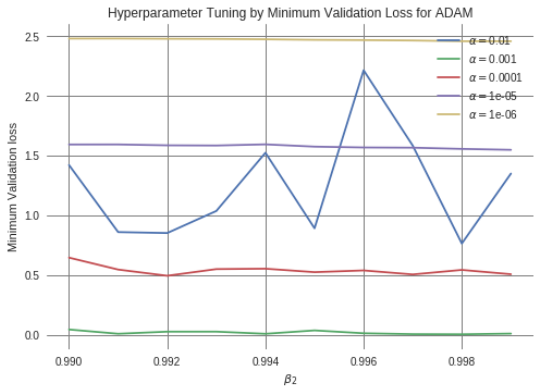

Fig. 15. CIFARNET-1: Hyper-parameter tuning for $\alpha \in \{1e-2, 1e-3, \ldots, 1e-6\}$ (wider set) and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for ADAM algorithm

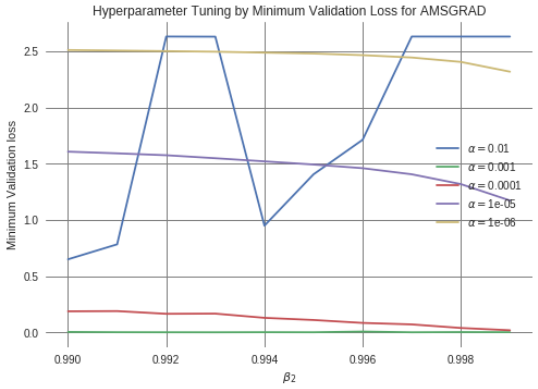

Fig. 16. CIFARNET-2: Hyper-parameter tuning for $\alpha \in \{1e-2, 1e-3, \ldots, 1e-6\}$ (wider set) and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for AMSGRAD algorithm

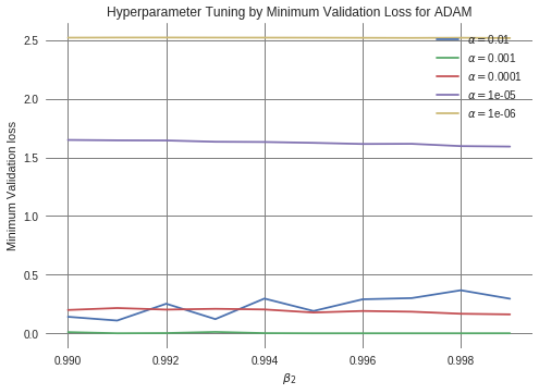

Fig. 17. CIFARNET-2: Hyper-parameter tuning for $\alpha \in \{1e-2, 1e-3, \ldots, 1e-6\}$ (wider set) and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for ADAM algorithm

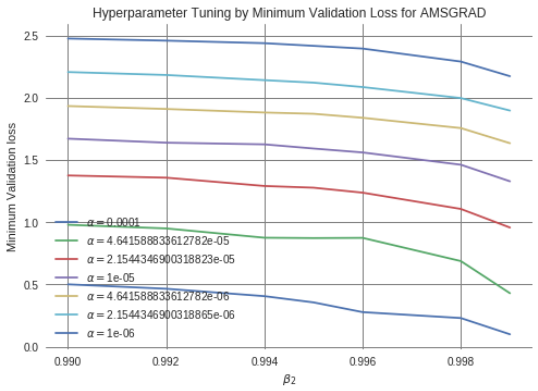

Fig. 18. CIFARNET-1: Hyper-parameter tuning for $\alpha \in \{np.geomspace(1e-4, 1e-6, num = 7)\}$ and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for AMSGRAD algorithm

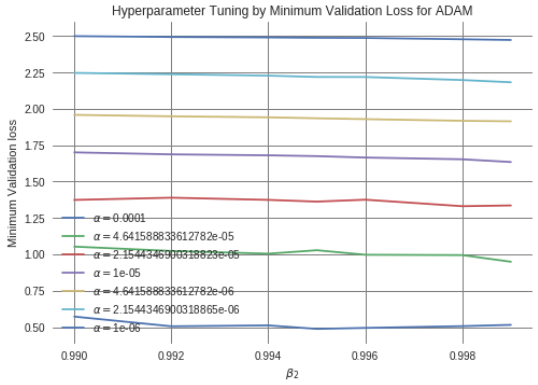

Fig. 19. CIFARNET-1: Hyper-parameter tuning for $\alpha \in \{np.geomspace(1e-4, 1e-6, num = 7)\}$ and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for ADAM algorithm

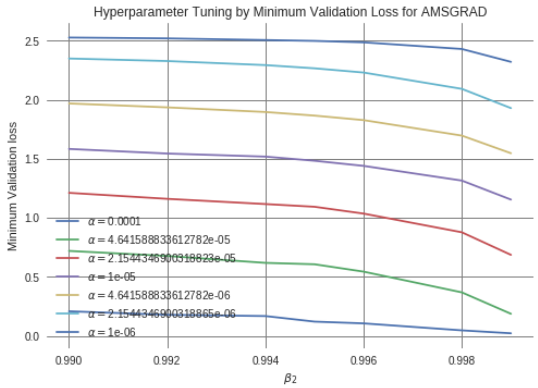

Fig. 20. CIFARNET-2: Hyper-parameter tuning for $\alpha \in \{np.geomspace(1e-4, 1e-6, num = 7)\}$ and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for AMSGRAD algorithm

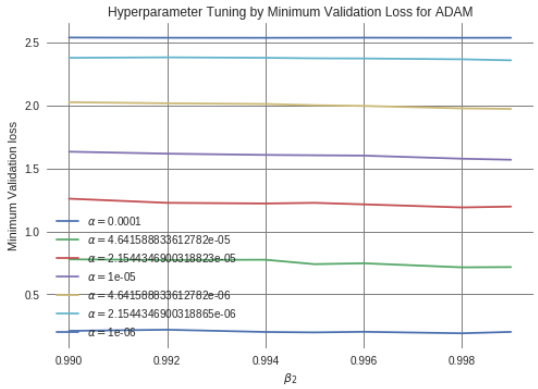

Fig. 21. CIFARNET-2: Hyper-parameter tuning for $\alpha \in \{np.geomspace(1e-4, 1e-6, num = 7)\}$ and $\beta_2 \in \{0.99, 0.991, 0, 992, \ldots, 0.999\}$ for ADAM algorithm

B. *Feed-Forward Neural Network*

Graphs were created that show a sample of the validation losses that were achieved with different hyper parameters fixed. Figure 22 shows with β_2 fixed to 0.995. Figure 23 shows with α fixed with 0.001.

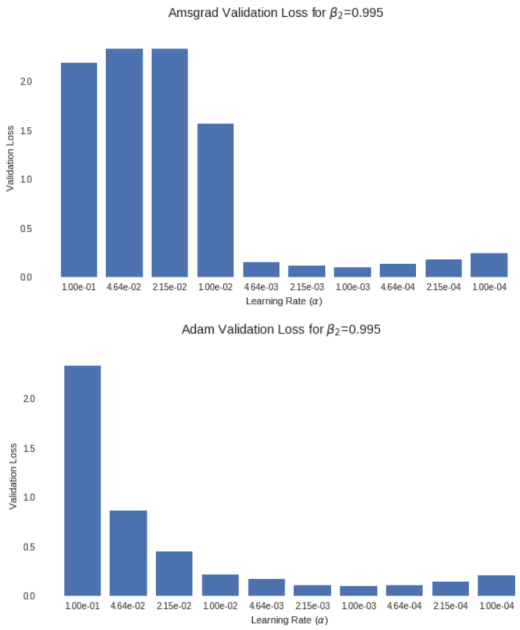

Fig. 22. Different lossess on the validation set for different alphas with β_2 fixed

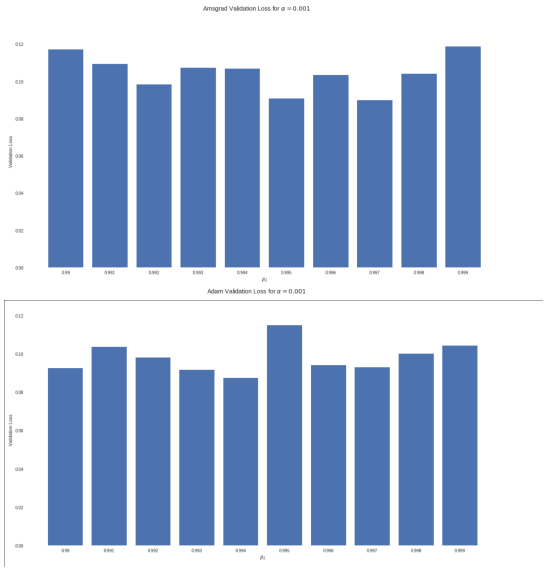

Fig. 23. Different losses on the validation set for AMSGRAD with α fixed