A Closer Look At The Convergence of Adam and AMSGrad: A Reproduction Study

Tamir Bennatan
tamir.bennatan@mail.mcgill.ca
260614526

Lea Collin
lea.collin@mail.mcgill.ca
260618407

Emmanuel Ng Cheng Hin
emmanuel.ngchenghin@mail.mcgill.ca
260615964

I. INTRODUCTION

The authors of the paper present issues with the popular stochastic gradient descent optimizers: RMSProp and ADAM, focusing mainly on ADAM. ADAM uses exponential moving averages of squared past gradients, which limits the reliance of parameter updates to only the last few gradients. Though ADAM has been proven to be very useful in many settings, it has also been shown to fail to converge to optimal solutions in certain cases. The usual problem in these other cases is that large, informative gradients during updates occur infrequently. Because ADAM limits the reliance of parameter updates to only the past few gradients, the influence of these informative gradients quickly die out due to the use of exponential moving averages, leading to poor convergence.

The paper introduces a toy example where it is obvious that ADAM does not converge to the optimal solution. The authors go on to discuss an incorrect assumption in the proof of convergence of ADAM that illuminates why ADAM does not converge in certain cases. Finally, the authors present a solution to the issue: AMSGrad. Their solution takes into consideration the "long-term" past history of gradients, so as not to lose the informative yet rare gradients that occur. The authors ran several experiments and compared the performance on training and test loss of both ADAM and AMSGrad. In this reproduction study, we aim to recreate these results as best as possible, while taking into consideration the cost of doing so. Additionally, we seek to identify any shortcomings or oversights of the authors.

II. THE NON-CONVERGENCE OF ADAM

The authors of this paper discuss in detail the issue with ADAM and point out an error in ADAM's proof of convergence. The quantity of interest in the proof is:

$$\Gamma_{t+1} = \left(\frac{\sqrt{V_{t+1}}}{\alpha_{t+1}} - \frac{\sqrt{V_t}}{\alpha_t} \right)$$

This quantity essentially measures the change in the inverse of the learning rate of the adaptive method with respect to time. The update rules of stochastic gradient descent lead to "non-increasing" learning rates. What the authors point out, however, is that ADAM and RMSProp can have potentially indefinite Γ_t, which leads to a lack of convergence. The authors claim that this problem arises because algorithms like ADAM use exponential moving averages.

The authors present a toy, adversarial example to illustrate their point on why ADAM fails to converge. The example is as follows, for $\mathcal{F} = [-1, 1]$:

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

where $C > 2$. It is easy to see that the value of x that leads to the minimum regret is -1, however, the authors show that ADAM converges to the highly suboptimal solution of $x = +1$. This elucidates the intuition that the influence of the large gradient C disappears too quickly to counteract the gradient of -1, which moves the algorithm in the wrong direction.

As a solution to ADAM's shortcomings, the authors use a smaller learning rate in comparison to ADAM that still incorporates the intuition of slowly decaying the effect of past gradients on the learning rate, as long as Γ_t is positive semidefinite. The solution they present is AMSGrad.

III. METHODOLOGY

The authors ran several experiments to compare the performance of ADAM and AMSGrad. We wanted to recreate every experiment the authors ran, as closely as possible, in order to assess the reproducibility and validity of their results.

The first two experiments, which they refer to as synthetic experiments, are very similar to the toy example used to show the non-convergence of ADAM. The first synthetic experiment is with an online setting:

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

while the second is with a stochastic setting:

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

with both having the constraint set $\mathcal{F} = [-1, 1]$. Very similar to the general toy example, the optimal solutions are both clearly $x = -1$ and so the algorithms are expected to converge to this optimal value.

For these experiments, we aimed to recreate both settings with the goal of reproducing the figures the authors included. The authors' results are shown in Figure 1.

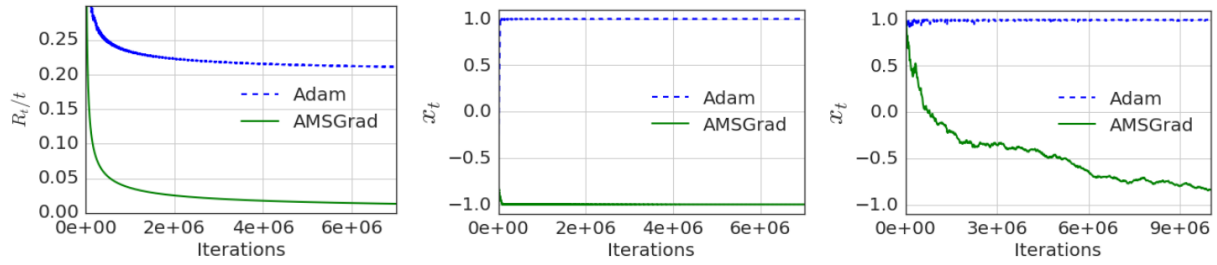

Fig. 1. The left two plots (left and center) are for the online setting and the last one (right) is for the stochastic setting. Note: these graphs were taken directly from the paper.

To implement these synthetic experiments, we used the same values for β_1 and β_2 as the authors, 0.9 and 0.99, respectively, for both ADAM and AMSGrad. As the authors did, we looked at the regret and the value of the iterate x_t for both algorithms. To find a "good" value for α we did a by-hand search through several values. (NEED TO ELABORATE ON THIS - ASK EMMANUEL).

The authors then investigated the performance of the algorithm on a logistic regression problem. They used the MNIST dataset, which contains 784 dimensional image vectors that each belong to one of 10 classes. The authors are clear in describing the step size parameter they chose ($\alpha/\sqrt{t}$), the minibatch size (128), and β_1 (0.9). They do not specify the values of β_2 and α that they tried, however, they do mention that they chose these values using a grid search and they give the range of values they tried for β_2 (0.99 - 0.999). We initially ran one experiment of logistic regression with set values for all of the hyperparameters to confirm that these algorithms behave similarly to the way that the authors described, before we started looking at many different values.

The authors then looked at the performance of ADAM and AMSGrad on different neural networks. They trained a 1-hidden layer, fully connected neural network on MNIST. Once again they specify that they used $\beta_1 = 0.9$ and the range of β_2 that they tried (0.99 - 0.999). They also specify that they used 100 ReLU nodes in the hidden layer for the experiment and that they used a constant $\alpha_t = \alpha$. They again say that they performed a grid search to look through ranges of α and β_2. Similar to what we did for logistic regression, we ran the experiment once before doing the gridsearch to confirm that the algorithms behaved in a way that was expected.

As a final experiment, the authors trained a convolutional neural network (CNN) on the standard CIFAR-10 dataset, which consists of 60,000 labeled 32 x 32 images. They specify the architecture that they used which has 2 convolutional layers with 64 channels and a kernel size of 6 x 6 followed by 2 fully connected layers of size 384 and 192. The network uses a 2 x 2 max pooling and layer response normalization between the convolutional layers. A dropout layer with keep probability of 0.5 was applied in between the fully connected layers. Once again, the minibatch size was set to 128, the same as the previous experiments. We followed the architecture almost exactly, though we used batch normalization rather than the layer response normalization the authors used. Once again, they state that they used a grid search to iterate through different values of α and β_2. The results from the paper can be seen in Figure 2.

Overall, we tried to replicate as much as possible what the authors did. Once we had found the best hyperparameters (namely, α and β_2) as a result of our grid search, we ran each algorithm 5 times and plotted the training and test losses for each "run". We thought this was important to do as each algorithm has random initializations and can sometimes randomly get "stuck" while running. Performing each experiment allows us to see if perhaps the authors hand-picked results that went along with their hypothesis or if the results truly were consistent, even after taking into consideration the randomness of the algorithms.

IV. RESULTS

The results for the synthetic experiments are shown in Figure 3 (haven't put it in yet).

The results for the MNIST and CIFAR-10 experiments are in Figure 4. Similar to what is seen in the results provided by the authors, we have that AMSGrad has a slightly lower loss than ADAM for these experiments. We also observe quite a small loss for our CIFAR-10 experiments, about 0.05 for both training and test losses. This is a much smaller loss than what was observed by the authors, who achieved about 0.3 training loss with AMSGrad and about 0.6 test loss with AMSGrad. We do not know what loss the authors used however, which could be the source of the discrepancy. Overall, the general trends we observe in our results coincide with the general trends reported by the authors.

V. DISCUSSION

The authors did quite a nice job of describing in detail what they did in each experiment they ran. For example, the full architectures of the two neural networks they trained were fully specified. Additionally, they stated the exact value of hyperparameters they did not tune (for example β_1) and gave a range for β_2 which they did tune. They also mentioned that they used a grid search to tune these parameters, so we knew to do that as well.

This is not to say that the authors' work was perfectly reproducible. Though they mentioned the range of β_2 values they tried, they did not give a range of α values for any

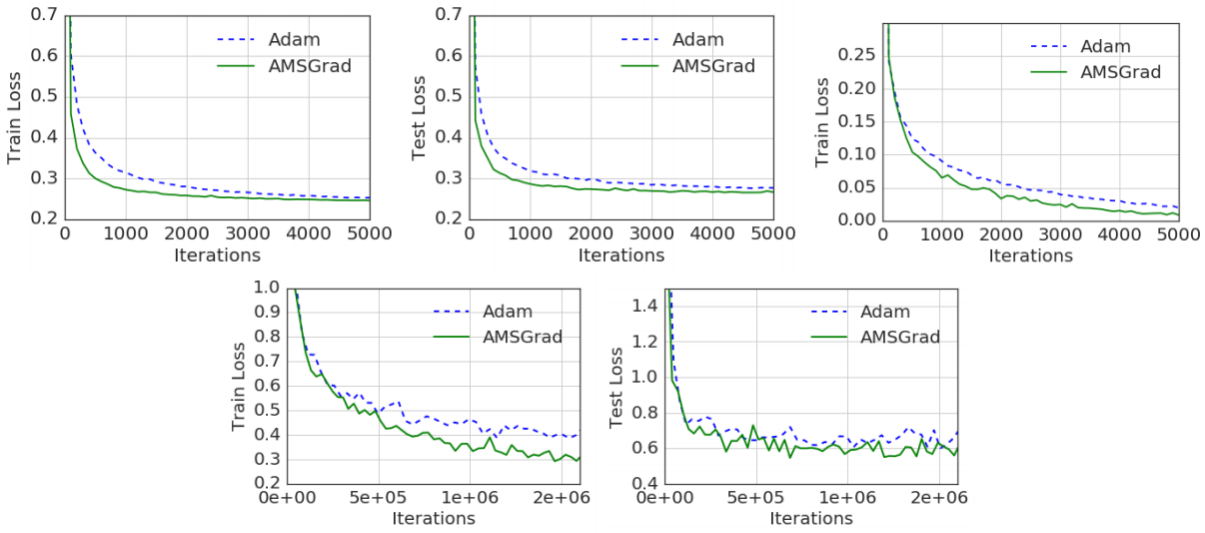

Fig. 2. The top row shows performance of ADAM and AMSGrad on logistic regression (left and center) 1-hidden layer feedforward neural network (right) on MNIST. In the bottom row, the two plots compare the training and test loss of ADAM and AMSGrad with respect to iterations for CIFARNET. Note: these graphs were taken directly from the paper.

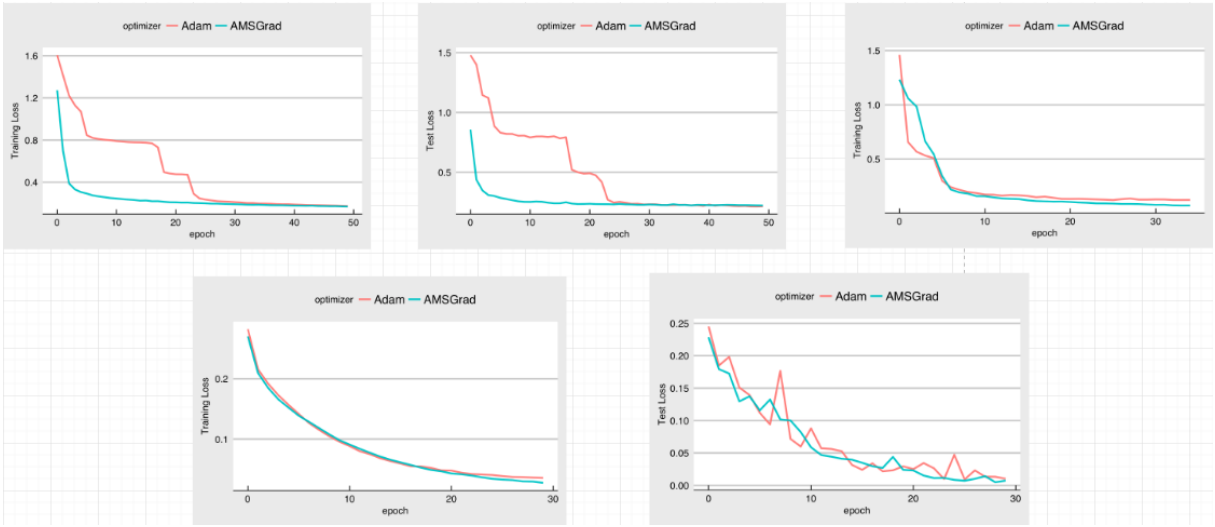

Fig. 3. The top row shows performance of ADAM and AMSGrad on logistic regression (left and center) 1-hidden layer feedforward neural network (right) on MNIST. In the bottom row, the two plots compare the training and test loss of ADAM and AMSGrad with respect to iterations for CIFARNET. These graphs are meant to coincide with the graphs in Figure 2, which were graphs provided by the authors.

of the experiments that they ran. Furthermore, though it is likely that they used categorical cross-entropy loss in the classification experiments, they do not explicitly state so. In their results, the authors often plotted the training or test loss against the number of iterations, but they do not define what they consider to be an iteration. We are fairly confident that one iteration means one run through one mini-batch, however, without the authors stating explicitly, we cannot be certain. Because of this, we are unsure of exactly how long the authors ran each experiment. To work around this, we ran each experiment until there was very little change in loss from one epoch to the next, as is usually done and this seems to be what the authors did, since we achieved similar results. However, in terms of absolute reproducibility of their results, this impeded our ability to reproduce the results exactly.

The biggest shortcoming of this paper is likely that the authors never state what they got to be the best hyperparameters after performing a grid search. Because we were able to get very similar results to the, we believe that the hyperparameters must be the same, yet we have no way to confirm.

Though we were able to produce extremely similar results to those provided in the original paper, these results did not come at a low cost. Because the authors did not provide a range of α values they tried for any of their experiments, we had to use a wide range in our grid search to ensure that we likely explored all of the same possible values the

authors had explored. Grid searches take a long time to run (INSERT LENGTH OF TIME FOR EACH EXPERIMENT HERE). This problem was exacerbated for the CIFAR-10 experiments as CNN's already take a long time to train (about 2 hours) and so to also do a grid search took (about 28 hours?). Thought it was useful that the authors stated they performed a grid search to do all hyperparameter tuning, it would have saved a lot of time and effort if they had simply stated what the best hyperparameters were that they found. This also would have made it much easier to decide if the results were reproducible or not as we would have been able to follow more exactly what the authors did.

Furthermore, CNN's also require a GPU to significantly speed up the amount of time required to train. This required us to use Amazon Web Services which required much more setup than simply running all of the code within our own machines. We chose to use Amazon Web Services rather than Google Collab because AWS was something that we had worked with before. Additionally, Google Collab would terminate if we let the screen of the computer turn off and considering we were running the CNNs overnight, this was not ideal.