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

***Abstract*— In this reproduction study, we assess the reproducibility and validity of the results reported in the paper *On the Convergence of Adam and Beyond*. The authors of this paper, Sashank, Kale and Kumar, discuss issues with the popular stochastic gradient descent optimization algorithm, *Adam*, and present a solution, *AMSGrad*. We reproduced quite similar results to those found in the paper, yet not without costs. (Finish up in some way).**

I. INTRODUCTION

Amongst the deep learning community, Stochastic Gradient Descent (SGD) is the predominant method of training deep neural networks. As researchers and practitioners continue to experiment with large networks and parameter spaces reaching hundreds of millions of dimensions, it has become increasingly important in recent decades to optimize SGD, so that it may train large networks faster and more effectively.

One such optimization to SGD is *Adam* (Kingma, Ba; 2015). Inspired by previous variants of SGD which apply time varying learning rates and momentum terms such as AdaGrad (Duchi et al., 2011) and RMSProp (Tieleman & Hinton, 2012), Adam adjusts the learning rates and momentum terms associated with each parameter of a network using exponential moving averages. In recent years, Adam has been shown to be effective in many deep learning settings, and is considered a state of the art technique.

In the paper *On the Convergence of Adam and Beyond*, authors Sashank, Kale and Kumar (2018) present a defect in the Adam method. Since Adam uses exponential moving averages of squared past gradients, the influence of these gradients on past parameter updates drops off quickly - effectively limiting the reliance of parameter updates to a small set of recent gradients. The authors show that because of this property, Adam will fail to converge in convex optimization problems where large, informative gradients occur infrequently.

The authors propose a new optimization scheme - *AMSGrad*, which aims to remedy this issue with Adam. To demonstrate that AMSGrad performs better than Adam, they devise a syntethic convex optimization problem, and show that AMSGrad converges to the optimal solution, while Adam does not (in fact it converges to a highly suboptimal solution). They then train modest neural networks on the widely used MNIST and CIFAR-10 datasets using the Adam and AMSGrad optimizers, and show that AMSGrad converges to a lower loss than Adam.

In this paper, we recreate the experiments described in the paper[1] and aim to reproduce the authors' results. We compare our results with those of the authors, and discuss the challenges in recreating the experiments described in this paper. Finally, we present a brief sensitivity analysis, to gauge the variability in our findings.

II. AMSGRAD: LONG TERM MEMORY OF GRADIENTS

Adaptive optimization procedures such as Adam yield different effecive learning rates for each of the different parameters of a model. Adam computes the effective learning rate using an exponential moving average of past squared gradients.

If we consider a model with a single parameter, θ, then Adam uses the following update step to optimize the function $f_t(x)$ at time t:

Algorithm 1 Adam Update Rule

1: $g_t = \nabla_\theta f_t(x_t)$
2: $m_t = \beta_{1t} m_{t-1} + (1-\beta_{1t}) g_t$ ▷ Adaptive Momentum
3: $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ ▷ Adaptive learning rate
4: $\theta_{t+1} := \theta_t - \alpha v_t$

Where α is the learning rate, and (β_1, β_2) are hyperparameters chosen in the range $(0, 1)$.

The exponential moving average terms on lines 2 and 3 of Alg. 1 reduce the reliance of each update on past gradients geometrically. The authors show that because of this property, the effective learning rate of each parameter is not guarenteed to be non-decreasing, which causes convergence issues in particular settings.

To account for this, the authors modify Adam to "remember" large gradients from further in the past:

Algorithm 2 AMSGrad Update Rule

1: $g_t = \nabla_\theta f_t(x_t)$
2: $m_t = \beta_{1t} m_{t-1} + (1-\beta_{1t}) g_t$
3: $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
4: $\hat{v}_t = max(\hat{v}_{t-1}, v_t)$ ▷ Propogate large updates
5: $\theta_{t+1} := \theta_t - \alpha \hat{v}_t$

[1] We refer to *On the Convergence of Adam and Beyond* as simply *"the paper"*, and *"the authors"* as the authors of this paper throughout.

Line 4 of Alg. 2 enables AMSGrad to propogate the large gradients into the future, which increases their effect on future updates. The authors show that after making this small adjustment, AMSGrad guarentees non-increasing effective learning rates, and favorable convergence behaviour.

To illustrate this, the authors propose a simple convex function on the domain $x \in [-1, 1]$:

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

for $C > 2$. The minimum value of f is achieved at $x = -1$. However, the authors show that Adam converges to the suboptimal solution of $x = 1$, due to the fact that the large gradients with magnitude C are only observed once every three time steps. The influence of this large gradient C disappears too quickly to counteract the gradients of -1, which move the algorithm in the wrong direction. AMSGrad, on the other hand, is designed to account for these settings, and minimizes this function without difficulty.

III. Experiments

The authors ran several experiments to compare the performance of Adam and AMSGrad. In this section, we describe the experiments run by the authors, and their reported results.

A. Synthetic Experiments

The authors construct two convex functions, similar to the one described in Section II, on the domain $x \in [-1, 1]$, designed to highlight Adam's shortcomings:

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

where t is the time step at which the function is evaluated, and:

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

Both functions reach a global minimum at $x = -1$. The first is referred to as the "online setting", and the second as the "stochastic setting."

In the first experiment, the authors use Adam and AMSGrad to minimize both functions, and compare the convergence behaviour of each optimizer. The authors fix the values of β_1 and β_2 at 0.9 and 0.99, respectively, and perform a grid search to find a learning rate α which yields good convergence for both optimizers.

B. Logistic Regression on MNIST

The authors then investigated the performance of the algorithm on a logistic regression problem. They used the MNIST dataset, which contains 70,000 28x28 images of handwritten digits, labeled as one of 10 classes. The authors decrease the learning rate over time, where the effective learning rate α_t at time t is defined as $\alpha/\sqrt{t}$. The authors train using minibatches of size 128, and fix β_1 to be 0.9. They then perform a grid search to select a value for β_2 in the range $(0.99, 0.999)$, and to select a value for α, for which no range of values is provided. These hyperparameters were then chosen based on which value yielded the lowest validation loss for each optimizer.

C. Feedforward Neural Network on MNIST

The authors trained a feedforward neural network (FFNN) with one hidden layer on the MNIST dataset as well. The hidden layer consists of 100 neurons, and uses the ReLU nonlinearity. The authors fix $\beta_1 = 0.9$, and use a grid search to select β_2 from the range $(0.99, 0.999)$, and to select a value for α, for which no range of values is provided. These hyperparameters were then chosen based on which value yielded the lowest validation loss for each optimizer.

D. Convolutional Neural Network on CIFAR-10

Finally, the authors experiment with a larger convolutional neural network (CNN), designed to classify images in the CIFAR-10 dataset. CIFAR-10 consists of 60,000 32 x 32 images, labeled as one of 10 classes.

The authors specify the architecture that they used (named *Cifarnet*), which consists of two convolutional layers, max-pooling and batch-normilization layers, and two fully connected layers (see appendix 1 for a detailed specification of the architecture.)

The authors trained CifarNet using a minibath of size 128 using each of the two optimizers.The authors fix $\beta_1 = 0.9$, and perform a grid search to select β_2 from the range $(0.99, 0.999)$, and to select a value for α, for which no range of values is provided. These hyperparameters were then chosen based on which value yielded the lowest validation loss for each optimizer.

E. Results

The authors present their results by visualizing the losses of their models achieved during training using Adam and AMSGrad for each of the experiments discussed [Figure 1]. These plots show that in all of the experiments, models trained using AMSGrad achieved lower losses and converged faster than those trained with Adam. These results bolster the author's claims that AMSGrad's dependence on long term gradients indeed ameliorates its convergence behavior.

IV. METHODOLOGY

Here lies a brief description of what we did - tuning, training multiple runs.

A. Hyperparemter Tuning

1) Synthetic experiments: The synthetic experiments have three hyperparameters, the weights $\beta 1$ and $beta2$ and the initial learning rate, $alpha$. The two first being fixed by the paper (to 0.9 and 0.99 respectively), the hyperparameter tuning process focused only on the latest. The trade off is the following: choosing low values of alpha leads to a slow convergence, whereas large values allow the algorithm to converge faster, but with less stability (i.e. the algorithm "oscillates" around its convergence value). As no objective function for convergence speed and stability exists to our

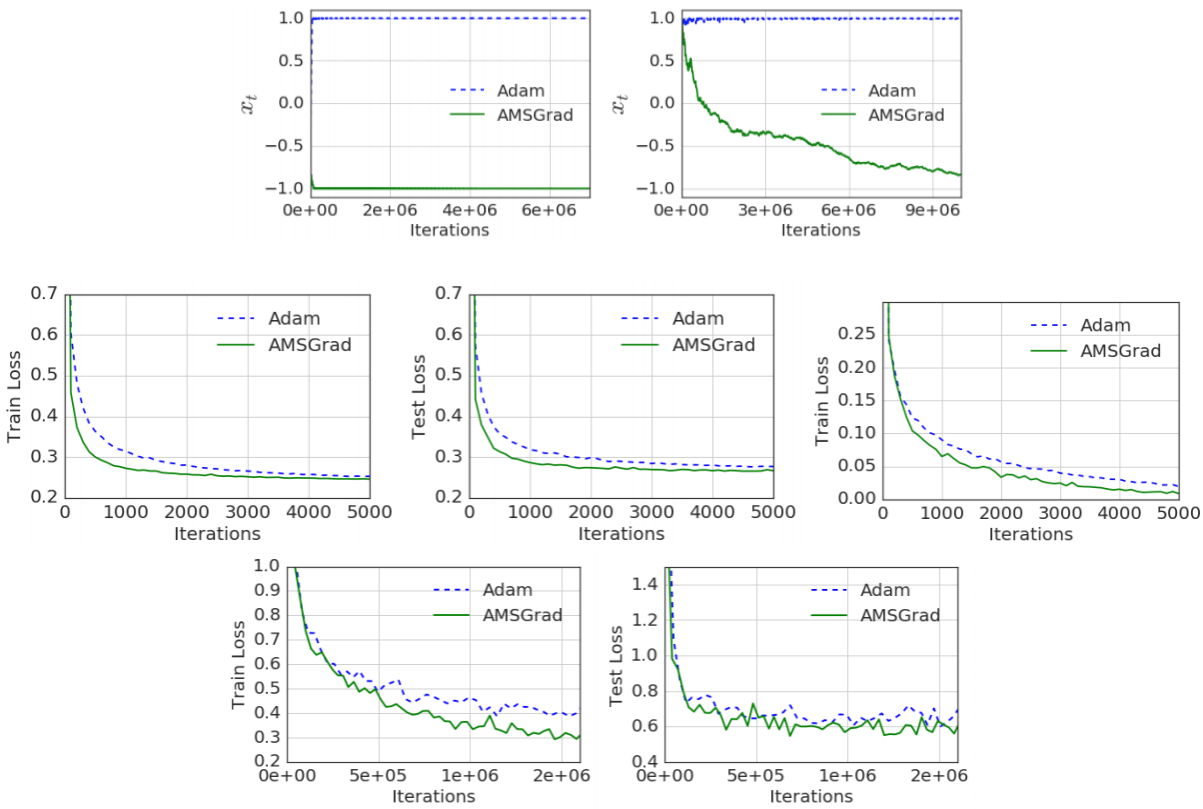

Fig. 1. Top: location of x_t in the online and stochastic syntetic experiments, respectively.
Middle: performance of ADAM and AMSGrad on logistic regression (left and center) 1-hidden layer feedforward neural network (right) on MNIST.
Bottom: training and test loss of ADAM and AMSGrad with respect to iterations for CifarNet.
These graphs were taken directly from the paper *On the Conergence of Adam and Beyond* (Sashank, Kale, Kumar; 2018).

knowledge, the tuning had to be done manually. It quickly turned out that no fixed value gives satisfactory results. Indeed, in the reference graphs, the algorithm converges fast at the beginning, which corresponds to a large alpha and is relatively stable (i.e. few oscillations), which typically happens when alpha is low. To address this issue, a learning rate decay mechanism was added to the algorithm. This worked particularly well for the stochastic setting, whereas no significant difference was observed in the online setting. The final values are α = 0.01 without decay for the online setting and α = 0.5 with decay for the stochastic setting. The two other hyperparameters, $\beta 1$ and $\beta 2$ were fixed at 0.9 and 0.99 respectively as described in the paper.

Here lies details on hyperparameter tuning

B. Reproduced Experiments

1) Synthetic experiments: To get more control on our experiments, we implemented the Adam and AMSgrad algorithms from scratch. They were designed as classes which keeps the value of a parameter and updates it with a gradient when the method "step" is called. This design allows to inject gradients in the algorithm. These gradients are generated with a mod operation in the online setting and a random generator in the stochastic setting. The first experiment was to reproduce the results described in the paper, to check their reproducibility. Further experiments were designed to check the significance of the gradients, as well as the influence of the learning rate in the convergence of both algorithm. To this effect, we tried various values for the mod, and different values of alpha.

Here lies details on experiments ran - number of runs, configs etc

V. RESULTS

The results for all the experiments can be seen in Figure 2. In the synthetic experiments, our results for both the online and stochastic settings are virtually identical to the results presented in the paper. The only difference is that our AMSGrad converged more quickly to the optimal x value than in the paper, otherwise the results are very comparable.

For the MNIST and CIFAR-10 experiments, we have that AMSGrad has a slightly lower loss than Adam. This is what we see in the original results, however, the original results have a slightly larger difference in loss between the two optimizers than what we observed. We also see quite a small loss for our CIFAR-10 experiments, about 0.05 for both training and test losses. This is quite a significantly 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. That said, the general trends

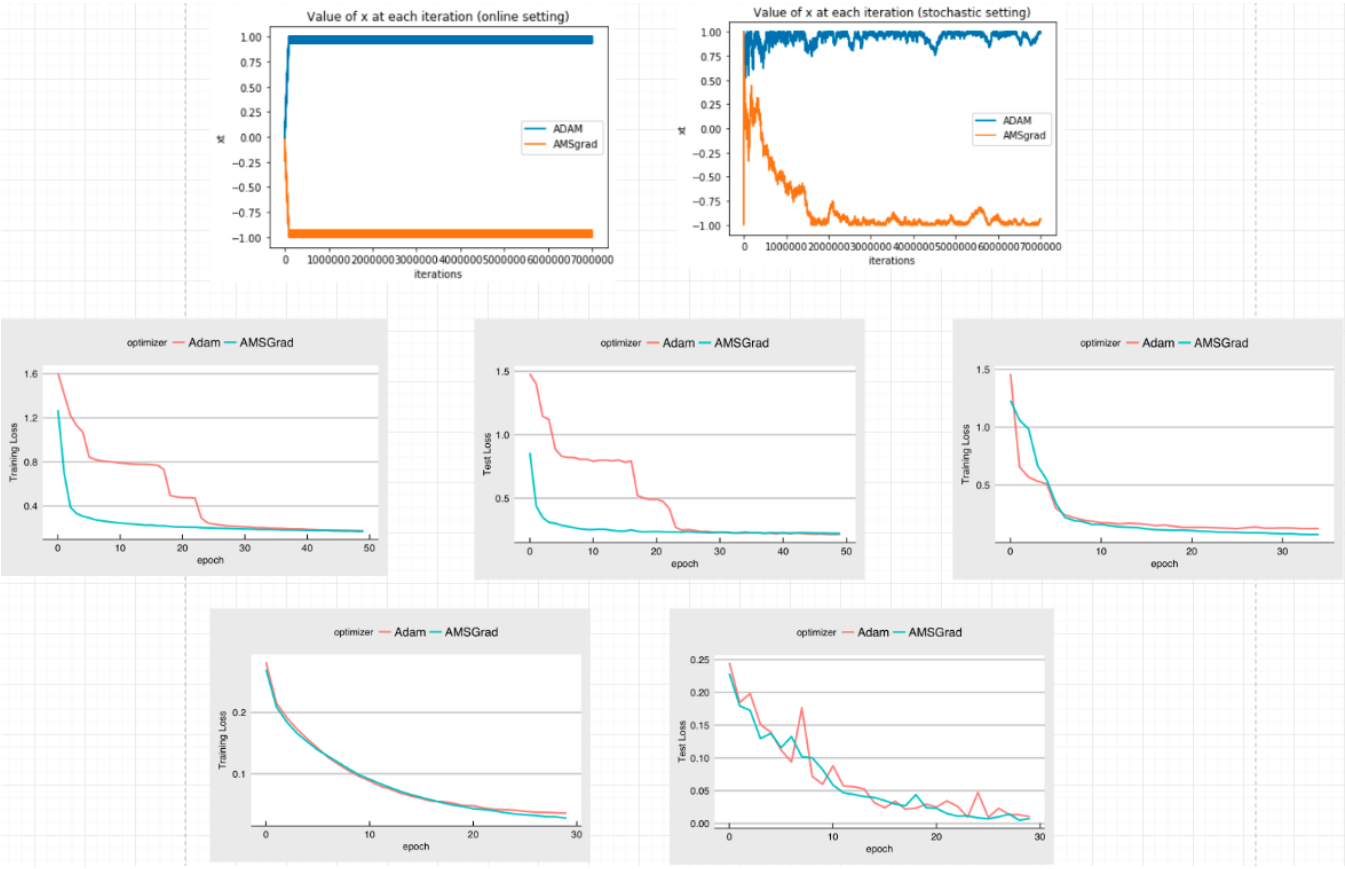

Fig. 2. Top: location of x_t in the online and stochastic synthetic experiments, respectively.
Middle: performance of ADAM and AMSGrad on logistic regression (left and center) 1-hidden layer feedforward neural network (right) on MNIST.
Bottom: training and test loss of ADAM and AMSGrad with respect to iterations for CifarNet.
These graphs were taken directly from the paper *On the Convergence of Adam and Beyond* (Sashank, Kale, Kumar; 2018).

we observe in our results coincide with the general trends reported by the authors.

VI. DISCUSSION AND CONCLUSION

Although the authors take care to justify their claims and experiments with theory, they omit several key details that make it difficult to recreate their experiments exactly. These details can be broadly categorized into those which concern the experimental setup, those which concern the model architectures, and those which concern the exact values of model hyperparameters.

A. Experimental Setup

In the paper, the models do not specify for how many epochs they trained each of the described classifiers.

Although the images they provide [Figure 1] show the number of *iterations* they trained for, it is difficult to decipher what these iterations actually represent.

For the models trained on MNIST, the authors show that they trained for 5,000 iterations. If an iteration is defined as one batch parameter update[2], then 5,000 iterations would correspond to 10-20 training epochs, which is reasonable. The authors show, however, that they trained CifarNet for over 2 million iterations. Using this same definition of an iteration, this would correspond to over $5,000$ training epochs. Given the size of CifarNet, we find it dubious that the authors trained for thousands of epochs[3].

To overcome this ambiguity, we determined a number of training epochs for each model that we found reasonable. We aimed to choose a number of training epochs for each number that gives each model ample time to converge, but that can train in a reasonable time frame. Thus, we trained our logistic regression and feedforwarnd neural network models for 50 epochs and our CifarNet models for 30 epochs.

B. Model Architectures

The authors do not specify several key details of their model architectures. Most severely, they do not specify the loss function they used to learn their parameters.

In a multi-class classification setting (such as MNIST and CIFAR-10 classification), a natural choice for a loss function is categorical cross-entropy. After running our experiments

[2]The authors use mini-batches of size 128 in all experiments. Thus, in the $50,000$ training examples of MNIST, one epoch would consist of $50000/128 \approx 390$ batch parameter updates.

[3]On an NVIDIA Kepler GK104 GRID GPU, one training epoch takes roughly 90 seconds.

using categorical-crossentropy, however, we noticed that the scale of our models' losses did not match those reported by the authors. For example, when running a feedforward neural network on MNIST, we observed training losses in roughly the range (1, 6), while the authors report losses in the range (.2, .7).

Based on these observations, we concluded that the authors must have used a loss function other than categorical-crossentropy. As an alternative, we tried using binary-crossentropy (also known as *multiclass log-loss* in the multiclass setting), which is defned for a k class classification problem as:

$$Loss(\hat{y}_i, y_i) = -\sum_{k=1}^{K} \big(y_{i,k} \log(\hat{y}_{i,k}) + (1-y_{i,k}) \log(1-\hat{y}_{i,k})\big)$$

When using this loss function we found that our observed losses aligned more closely with those reported by the authors in our experiments regarding the MNIST dataset. However, we observed losses much smaller than those reported by the authors when training CifarNet.

This could be because the authors used different loss functions between their experiments regarding MNIST and CIFAR-10. It could also be that the authors used a loss function other than categorical cross-entropy and multiclass log-loss.

As the paper's results focus on the superiority of AMSGrad over Adam, we decided that the magnitude of the losses observed is less important than the relative magnitude between those achieved when training with AMSGrad, and those achieved when training with Adam. Thus, we proceeded with our experiments using the multiclass log-loss, with knowledge that this loss may be different than the one used by the authors.

C. Model Hyperparameters

Finally, the authors do not go into full detail on the hyperparameters they chose for each model.

In the CifarNet CNN model, an important hyperparameter is the stride length to use in each convoltional layer. Given just the kernel size and number of filters, one cannot deduce the stride length used by the authors[4]. Unable to resolve this ambiguity, we proceeded with our experiments using a stride length of one. Any discrepancies between our results and those reported by the author may be partly due to the use of different stride lengths.

The authors mention that they tune the learning rate α and the hyperparameter β_2 using a grid search in each of their experiments. However, they do not specify the hyperparameters they ultimately used.

This forced us to experiment with large parameter spaces when we recreated their grid searches, as we did not have an approximate neighborhood of values which we knew worked well in the proposed experiments. As training deep networks such as CifarNet is very slow, exhaustive searches are very costly. Thus, omitting the final hyperparameters introduces computational barriers to reproducing the authors' results well. The grid search for CifarNet took about 28 hours to complete, which is quite a significant amount of time that could have been greatly reduced had the authors either given us the exact values of the best hyperparameters they found, or at the very least given a range for the α values they looked through.

NOT SURE HOW TO ORGANIZE - bc below is unrelated to hindrances of reproducibility; ps feel free to reword

Lastly, we assess the validity of the author's claims based on their results. The authors present the issue with Adam as though it is an incredibly pervasive and serious problem when performing deep learning. As we have seen in this reproducibility study, though it is true that AMSGrad performs better than Adam (in the experiments presented), it is not as significant an improvement as the authors made it seem. Because we ran each experiment several times, as opposed to the authors who report only one run of each experiment, we saw that some runthroughs matched the authors' results very closely, while in other runthroughs, the difference between Adam and AMSGrad seemed quite insignificant. The synthetic experiment is the only experiment where it is clear that AMSGrad performs much better. However, it should be noted that this experiment was devised as the perfect example where Adam will perform poorly and it is not clear how many of these 'perfect' examples arise in practice.

[4]For the layer to be specified fully, either the stride length or zero-padding length should also be specified.

REFERENCES

[1] R. Sashank, S. Kale, S. Kumar, (2018). On the Convergence of Adam and Beyond. In Proceedings of *International Conference on Learning Representations*, http://https://openreview.net/forum?id=ryQu7f-RZ (link is external)

[2] Diederik P. Kingma and Jimmy Ba (2015). Adam: A method for stochastic optimization. In Proceedings of *3rd International Conference on Learning Representations, 2015.*

[3] Chollet, François and others, 2015. Keras: `https://keras.io`

[4] T. Tieleman and G. Hinton. RmsProp: Divide the gradient by a running average of its recent magnitude.

[5] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12:2121–2159, 2011.

BLOG POSTS

[6] Brownlee, J. (2016, August 9). Tuning optimization parameters using a gridsearch. Retrieved from https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

[7] Korzeniowski, F. (2017, December 22). Experiments with AMSGrad. Retrieved from https://fdlm.github.io/post/amsgrad/

VII. APPENDIX

A. Appendix 1: Model Architectures

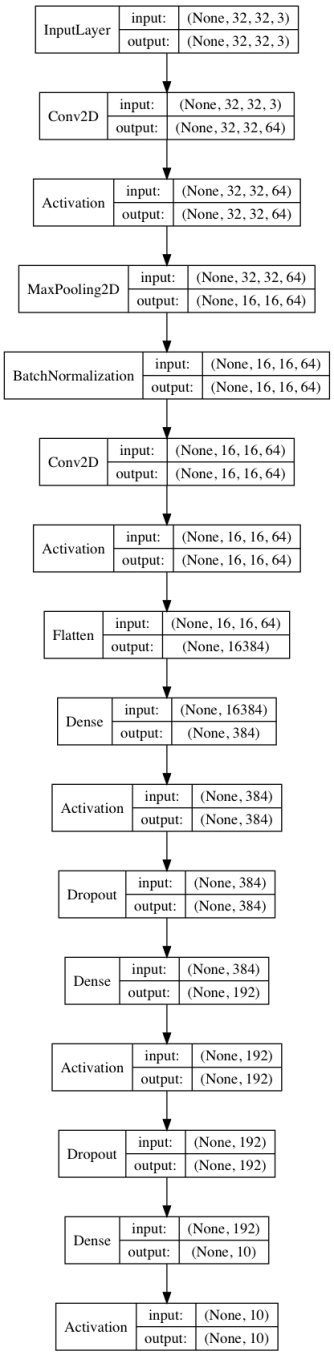

Fig. 3. CifarNet architecture; a deep Convolutional Neural Network designed to classify images in the Cifar-10 dataset.

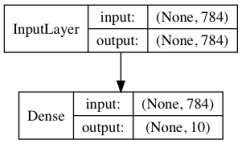

Fig. 4. Logistic Regression model, implemented as a feedforward neural network with no hidden layers.

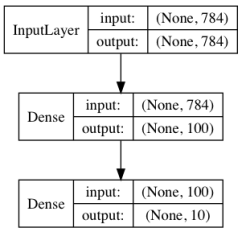

Fig. 5. Feedforward Neural Network Architecture, trained on MNIST 28x28 greyscale images.