PADAM: CLOSING THE GENERALIZATION GAP OF ADAPTIVE GRADIENT METHODS IN TRAINING DEEP NEURAL NETWORKS

ICLR REPRODUCIBILTY CHALLENGE 2019

Harshal Mittal *
Department of Electronics and Communication Engineering
Indian Institute of Technology, Roorkee
hmittal@ec.iitr.ac.in

Kartikey Pandey *
Department of Electronics and Communication Engineering
Indian Institute of Technology, Roorkee
kpandey@ec.iitr.ac.in

Yash Kant *
Department of Electrical Engineering
Indian Institute of Technology, Roorkee
ysh.kant@gmail.com

ABSTRACT

This work is a part of ICLR Reproducibility Challenge 2019, we try to reproduce the results in the conference submission *PADAM: Closing The Generalization Gap of Adaptive Gradient Methods In Training Deep Neural Networks*. Adaptive gradient methods proposed in past demonstrate a slightly degraded performance than the stochastic gradient descent (SGD) with momentum. The authors try to address this problem by designing a new optimization algorithm that bridges the gap between the space of Adaptive Gradient algorithms and SGD with momentum. With this method a new tunable hyperparameter called partially adaptive parameter p is introduced that varies between [0, 0.5]. We build the proposed optimizer and use it to mirror the experiments performed by the authors. We review and comment on the empirical analysis performed by the authors. Finally, we also propose a future direction for further study of Padam. Our code is available at: https://github.com/yashkant/Padam-Tensorflow

1 INTRODUCTION

Adaptive gradient methods such as Adam Kingma & Ba (2014), Adagrad, Adadelta Kingma & Ba (2014), RMSProp, Nadam, Adamw Loshchilov & Hutter (2017), were proposed over SGD with momentum for solving optimization of stochastic objectives in high-dimensions. Amsgrad was recently proposed as an improvement to Adam to fix convergence issues in the latter. These methods provide benefits such as faster convergence and insensitivity towards hyperparameter selection i.e. they are demonstrated to work with little tuning. On the downside, these adaptive methods have shown poor empirical performance and lesser generalization as compared to SGD-momentum. The authors of Padam attribute this phenomenon to the "over-adaptiveness" of the adaptive methods.

The key contributions of Padam as mentioned by the authors are:

- The authors put forward that Padam unifies Adam/Amsgrad and SGD with momentum by a partially adaptive parameter and that Adam/Amsgrad can be seen as a special fully adaptive instance of Padam. They futher claim that Padam resolves the "small learning rate dilemma" for adaptive gradient methods and allows for faster convergence, hence closing the gap of generalization.

*All the authors contributed equally to the reproducibilty challenge. The names are sorted in alphabetical order.

- The authors claim that Padam generalizes equally good as SGD-momentum and achieves fastest convergence.

We address and comment on each of the above claims from an empirical point of view.

Next, we would like to discuss the convergence speed of the algorithm.

We built the Amsgrad and Padam optimizers from the the base code of Adam in tensorflow[1].

2 BACKGROUND

The two most recent adaptive methods proposed that inspires Padam are Adam and Amsgrad. Adam made use of bias-corrected first order and second order moments along with the gradients to obtain the final weight update rule.

$\theta_{t+1} = \theta_t - \alpha_t \frac{m_t}{\sqrt{v_t}}$ where $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ (Adam)

A convergence issue in Adam was recently uncovered and addresssed by Amsgrad, where they suggest tweaking the update rule slightly to fix it. Padam makes use of this updated algorithm.

$\theta_{t+1} = \theta_t - \alpha_t \frac{m_t}{\sqrt{\hat{v}_t}}$ where $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$ (Amsgrad)

2.1 PADAM ALGORITHM

Padam introduces a new partially adaptive parameter p that takes value within the range [0, 0.5] and on the extremities of which it takes the form of SGD with momentum or AMSGrad. From Algorithm 1, when p is set to 0 Padam reduces to SGD with momentum whereas setting it to 0.5 leaves us with AMSGrad optimizer.

Algorithm 1 Partially adaptive momentum estimation method (Padam)

input: initial point $\theta_1 \in \mathcal{X}$; step sizes $\{\alpha_t\}$; momentum parameters $\{\beta_{1t}\}$, β_2 ; partially adaptive parameter $p \in (0, 1/2]$
set $m_0 = 0, v_0 = 0, \hat{v}_0 = 0$
for t = 1,...,T **do**
  $g_t = \nabla f_t(\theta_t)$
  $m_t = \beta_{1t} m_{t-1} + (1-\beta_{1t}) g_t$
  $m_t = m_t / (1-\beta_1^t)$ (Compute bias-corrected first moment estimate)
  $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  $v_t = v_t / (1-\beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$
  $\theta_{t+1} = \Pi_{\mathcal{X}, diag(\hat{v}_t^P)}(\theta_t - \alpha_t \cdot m_t / \hat{v}_t^P)$
end for

3 EXPERIMENTS

In this section we describe our experiment settings for evaluation of Padam on three different CNN architectures as proposed in the paper, comparing its performance against the standard baseline optimization algorithms.

3.1 ENVIRONMENTAL SETUP

All our experiments are conducted on 4 Tesla Xp GPUs (12Gb RAM per GPU). We have implemented all the experiments using Tensorflow version 1.13.0 graph free eager execution mode within Python 3.5.2.

[1] Implementation of Adam at: https://github.com/tensorflow/tensorflow/blob/r1.12/tensorflow/python/training/adam.py

3.2 DATASETS

The experiments were conduted on three of the popular datasets for image classification: CIFAR-10 Krizhevsky (2009), CIFAR-100 Krizhevsky (2009). The perfomance of various optimizers on the aforementioned datasets was evaluated on three different CNN architechture: VGGNetSimonyan & Zisserman (2014), ResNetHe et al. (2016) and Wide ResNetZagoruyko & Komodakis (2016). We test CIFAR-10 task for 200 epochs and CIFAR-100 task for 100 epochs. The experiments for CIFAR datasets were performed with a learning rate decay at every 50th epoch ie. 50,100,150. We were unable to perform the experiments on the ImageNet dataset because of the computational capacity of our machines and the time-constraints.

3.3 BASELINE ALGORITHMS

We compare Padam against the most popular adaptive gradient optimizers and SGD-momentum. Note, that the evaluation against AdamW was added by the authors at a delayed stage and the details about it were not completely released in the updated version of the paper, owing to this delay we have not been able to carry out experiments with AdamW, but we would immediately address this as soon as possible.

Table 1: Baseline Algorithms and their corresponding hyperparameters

Optimizer	**Padam** [2]	**SGD+Momentum**	**Adam**	**Amsgrad**
Initial Learning rate	0.1	0.1	0.001	0.001
Beta1	0.9	-	0.9	0.9
Beta2	0.999	-	0.99	0.99
Weight decay	0.0005	0.0005	0.0001	0.0001
Momentum	-	0.9	-	-

3.4 ARCHITECTURES

3.4.1 VGGNET

The VGG-16 network uses only 3 x 3 convolutional layers stacked on top of each other for increasing depth and adopts max pooling to reduce volume size. Finally, two fully-connected layers are followed by a softmax classifier.

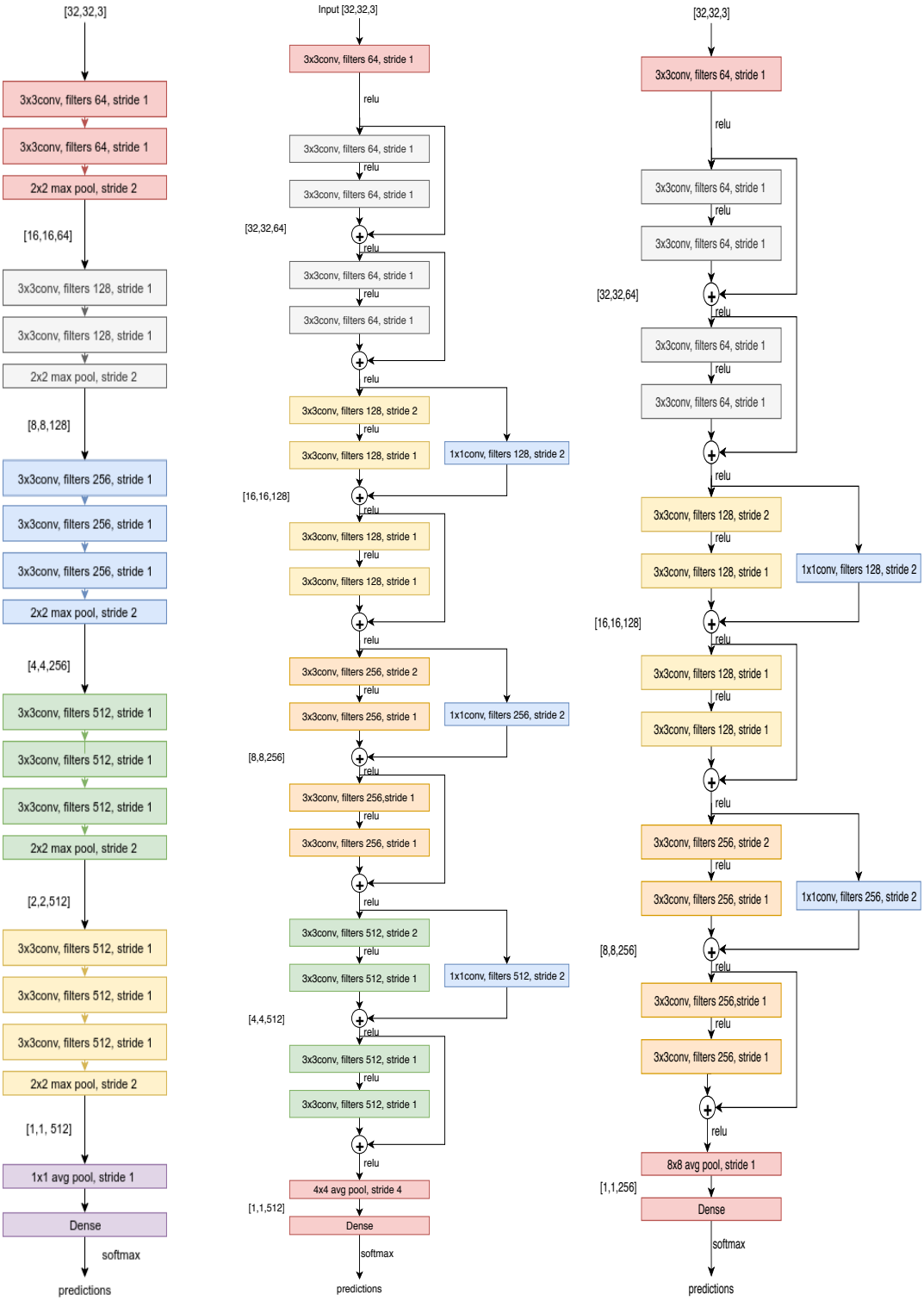
3.4.2 RESNET

Residual Neural Network (ResNet) He et al. (2016) introduces a novel architecture with "skip connections" and features a heavy use of batch normalization. As per authors, we use ResNet-18 for this experiment, which contains 4 blocks each comprising of 2 basic building blocks.

3.4.3 WIDE RESNET

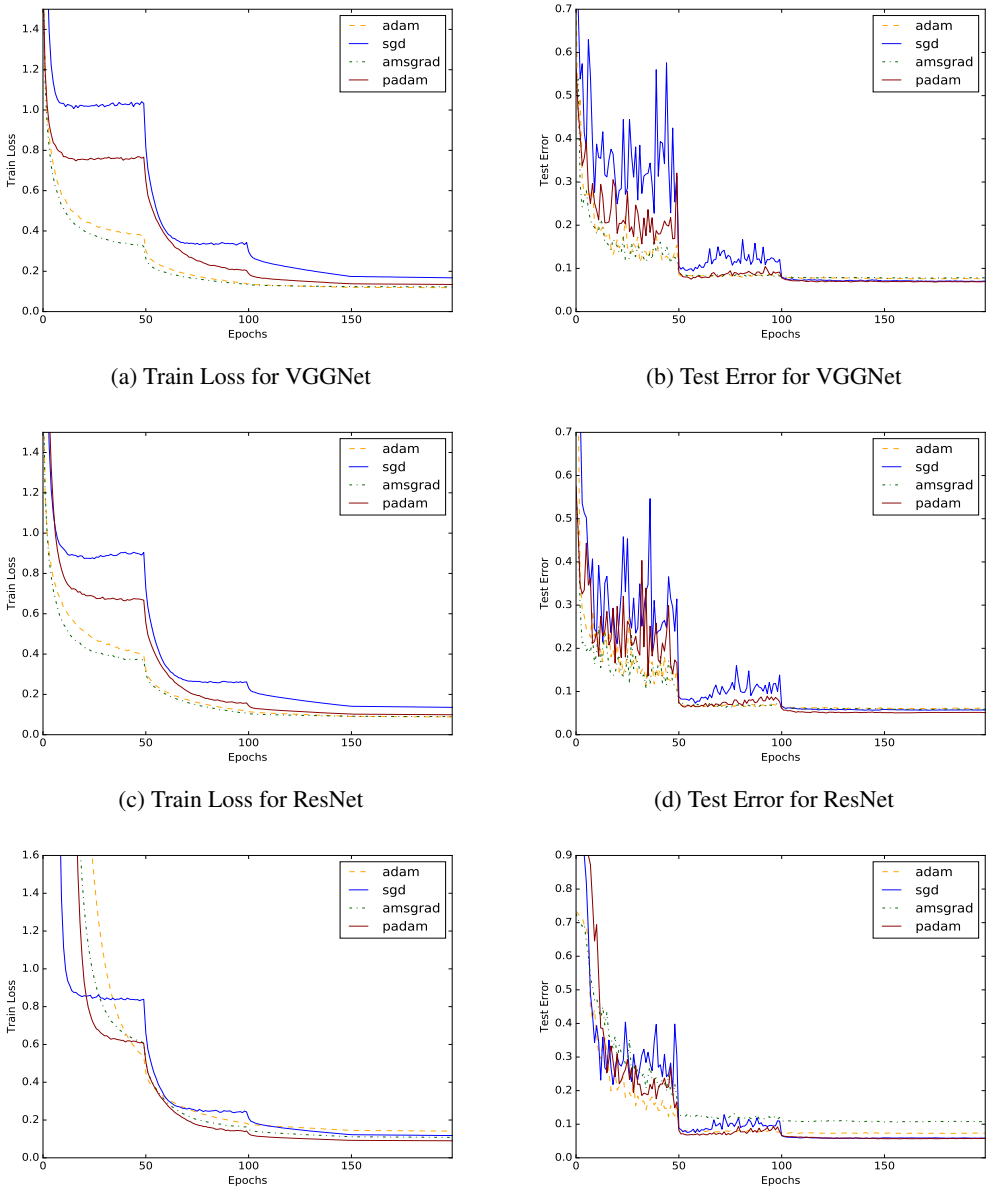
Wide Residual Network Zagoruyko & Komodakis (2016) further exploits the "skip connections" used in ResNet and in the meanwhile increases the width of residual networks. In detail, we use the 16 layer Wide ResNet with 4 multipliers (WRN-16-4) in the experiments.

[2]We have used p=0.125 as the hyperparameter for the experiments unless specifically mentioned



(a) VGG-Net (b) ResNet (c) WideResNet

Figure 1: Architectures used in experiments. Note that we do not explicitly show batch normalization layer after each convolution operation for brewity.



(a) Train Loss for VGGNet

(b) Test Error for VGGNet

(c) Train Loss for ResNet

(d) Test Error for ResNet

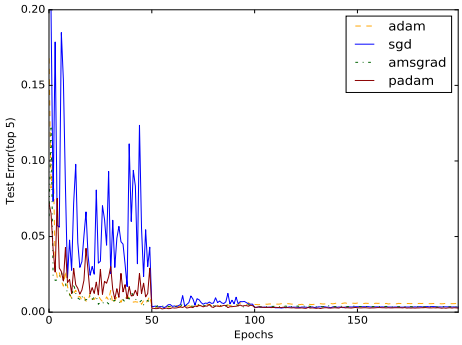
(e) Train Loss for Wide ResNet

(f) Test Error for Wide ResNet

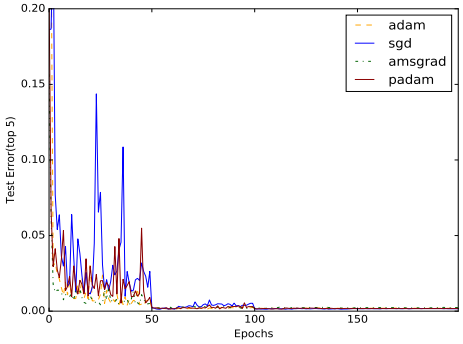
Figure 2: Train loss and test error (top-1 error) of three CNN architechtures on CIFAR-10.

Table 2: Test Accuracy of VGGNet on CIFAR-10

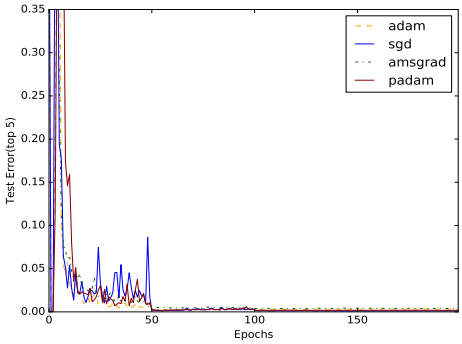
Methods	**50th epoch**	**100th epoch**	**150th epoch**	**200th epoch**
SGD Momentum	68.71	87.88	92.94	92.95
Adam	84.62	91.54	92.34	92.39
Amsgrad	**87.89**	**91.75**	92.26	92.19
Padam	67.92	90.86	**93.08**	**93.06**



(a) Top-5 Error for VGGNet



(b) Top-5 Error for ResNet



(c) Top-5 Error for Wide ResNet

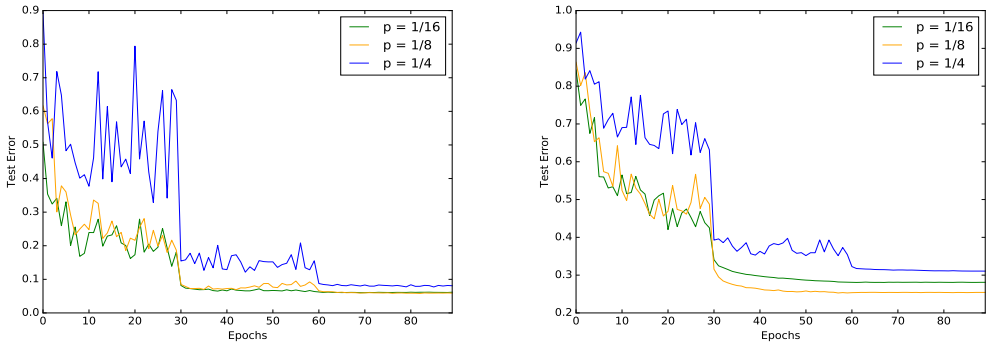
Figure 3: Top-5 error for three CNN architechtures on CIFAR-10.

4 Evaluation and Results

In this section we comment on the results we obtained with this reproducibility effort. We divide this section into three parts.

4.1 Train Experiments

Padam is compared with other proposed baselines, Figure 2 shows the Train Loss and Test Error for the three architectures on CIFAR-10. We find that Padam performs comparably with SGD-momentum on all the three architectures in test error. Padam maintains a rate of convergence between Adam/AMSgrad and SGD. We find that Padam works as proposed by the authors to bridge the gap between adaptive methods and SGD at the cost of introducing a new hyperparameter p, which requires tuning. Although, we don't see a clear motivation behind the grid-search approach used by the authors to select the value of partially adaptive parameter p.



(a) CIFAR-10 (b) CIFAR-100

Figure 4: Performance comparison of Padam with different choices of p for training ResNet on CIFAR-10 / CIFAR-100 dataset.

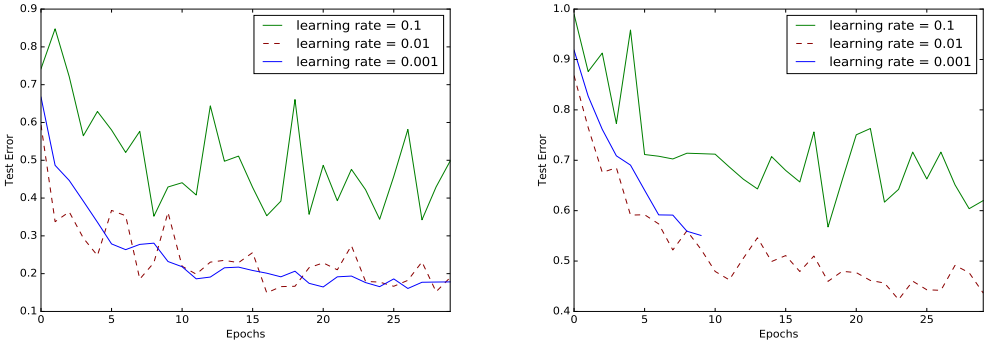
4.2 p-value Experiments

In order to find an optimal working value of p the authors perform grid search over three options [0.25, 0.125, 0.0625]. They do so by keeping the base learning rate fixed to 0.1 and decaying it by a factor of 0.1 per 30 epochs. We perform the same experiment on CIFAR-10 and CIFAR-100, the results are plotted in Figure 4. We observe similar results as the authors, and find the most optimal setting for p to be 0.125, out of the proposed three values. Nevertheless, we would like to press that this value p could be sub-optimal and may turn out to be sensitive to the learning rate schedule and base value. For this very reason we perform sensitivity experiments of p against various learning rates.

4.3 Sensitivity Experiments (presently incomplete and running)

To evaluate the possibility that optimal value of partial adaptive paramter p is entangled with learning rate we run the sensitivity experiments.In these, we propose to keep the p fixed and vary the base learning rate over [0.1, 0.01, 0.01] and run the each evaluation for 30 epochs on CIFAR-10 and CIFAR-100. We expect that this would uncover the dependence of p on base learning rate and its schedule. From the results plotted in Figure 5, we observe that for p = 0.25 base learning rate= 0.01/ 0.001 seems to be a more appropriate choice from the performance on test set than 0.1.

The experiments from this sub-section of the report are currently running and will be updated very soon with better insights from them. We plan to evaluate the above experiment with p value set to 0.125 and 0.0625 as well.



(a) CIFAR-10 (b) CIFAR-100

Figure 5: Performance comparison of Padam with different choices of learning rate for a fixed value of $p = 0.25$ ResNet on CIFAR-10 / CIFAR-100 dataset.

5 Discrepancies, Suggestions and Conclusion

The authors wrongly attribute the gradient explosion problem of adaptive methods with larger base learning rate to the second-order moment in the denominator. This proposition implicitly assumes v to be in between 0 and 1, which might not always be the case and hence the factor may cause the effective learning rate to either increase or decrease.

Overall, we conclude from the empirical evaluation that Padam is capable of mixing the benefits of adaptive gradient methods with those of SGD with momentum. Perhaps the optimal choice of newly introduced partially adaptive p parameter seems to be a good direction to further this work along.

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. doi: 10.1109/cvpr.2016.90. URL `http://dx.doi.org/10.1109/CVPR.2016.90`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016.