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

ICLR REPRODUCIBILTY CHALLENGE 2019

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

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

Yash Kant *
Dept. 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 degraded generalization 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 (Zeiler (2012)), RMSProp (Geoffrey Hinton & Swersky (2012)), Nadam (Dozat (2016)), 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 further claim that Padam resolves the "small learning rate

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

dilemma" for adaptive gradient methods and allows for faster convergence, hence closing the gap of generalization.

- 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. We run additional experiments to study the effect of learning rate (and its schedule) on the optimal value of partially adaptive parameter *p*. From our analysis we propose the use of a suitable schedule to vary *p* as the training proceeds in order to actually enjoy the best from both the worlds.

2 BACKGROUND

Padam is inspired from two recent adaptive techniques introduced in Adam and Amsgrad, we discuss them briefly here. Adam made use of bias-corrected first and second order moments along with the gradients for weight update.

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

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

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

2.1 PADAM ALGORITHM

Padam introduces a new partially adaptive parameter *p* that takes value within the range [0, 0.5]. On the extremities of this range it takes the form of SGD with momentum or AMSGrad. From Algorithm 1, when *p* is set to 0.0, 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 $\theta_\mathbf{1} \in \mathcal{X}$; step sizes $\{\alpha_t\}$; momentum parameters $\{\beta_{1t}\}$, β_2 ; partially adaptive parameter $p \in (0, 1/2]$
set $\mathbf{m_0} = \mathbf{0}, \mathbf{v_0} = \mathbf{0}, \hat{\mathbf{v}}_\mathbf{0} = \mathbf{0}$
for t = 1,...,T **do**
  $\mathbf{g_t} = \nabla f_t(\theta_\mathbf{t})$
  $\mathbf{m_t} = \beta_{1t}\mathbf{m_{t-1}} + (1-\beta_{1t})\mathbf{g_t}$
  $\mathbf{m_t} = \mathbf{m_t}/(1-\beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\mathbf{v_t} = \beta_2 \mathbf{v_{t-1}} + (1-\beta_2)\mathbf{g_t^2}$
  $\mathbf{v_t} = \mathbf{v_t}/(1-\beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\hat{\mathbf{v}}_\mathbf{t} = \max(\hat{\mathbf{v}}_\mathbf{t-1}, \mathbf{v_t})$
  $\theta_{\mathbf{t+1}} = \Pi_{\mathcal{X}, diag(\hat{\mathbf{v}}_\mathbf{t}^\mathbf{P})}(\theta_\mathbf{t} - \alpha_t \cdot \mathbf{m_t}/\hat{\mathbf{v}}_\mathbf{t}^\mathbf{P})$
end for

3 EXPERIMENTS

In this section we describe our experiment settings for evaluating Padam. We have tried to keep our implementation faithful to the authors code [1]. We build three different CNN architectures as proposed in the paper, and compare Padam's performance against the other baseline algorithms. We built the Amsgrad and Padam optimizers on top of the base code of Adam in tensorflow[2].

[1] Authors code is available at: https://github.com/uclaml/Padam/

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

3.1 Environmental Setup

We have built the experiments using Tensorflow version 1.13.0 and graph-free Eager Execution mode within Python 3.5.2. We ran the experiments on 4 Tesla Xp GPU cards (with 12Gb RAM per GPU).

3.2 Datasets

The experiments were conducted on two popular datasets for image classification: CIFAR-10 and CIFAR-100 (Krizhevsky (2009)). The performance of various optimizers on the aforementioned datasets was evaluated with three different CNN architecture: VGGNet (Simonyan & Zisserman (2014)), ResNet (He et al. (2016)) and Wide ResNet(Zagoruyko & Komodakis (2016)). We run CIFAR-10 and and CIFAR-100 task for 200 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 time constraints and limited availability of computing resources.

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 later stage and the details about it were not completely disclosed in the updated version of the paper or code, owing to this delay we have not been able to carry out experiments with AdamW.

Table 1: Baseline Algorithms and their corresponding hyperparameters

Optimizer	**Padam** [3]	**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

We have built architectures faithful to the code released by the authors, and are shown in Figure 1.

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.

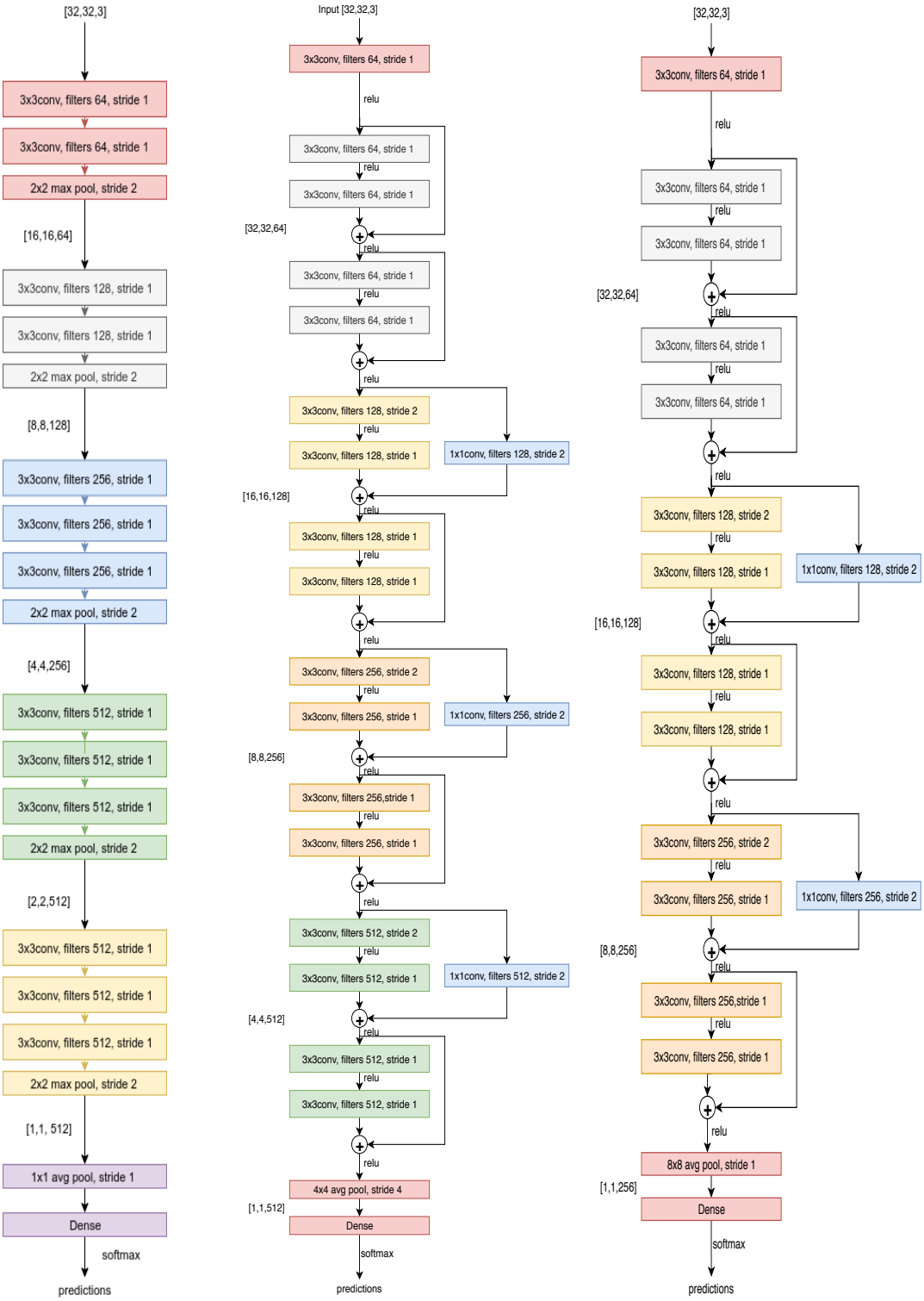
3.4.2 ResNet

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.

[3] 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 brevity.

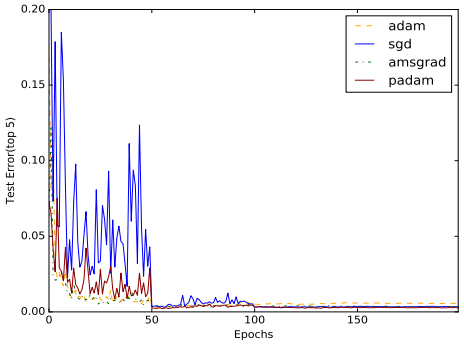
4 EVALUATION AND RESULTS

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

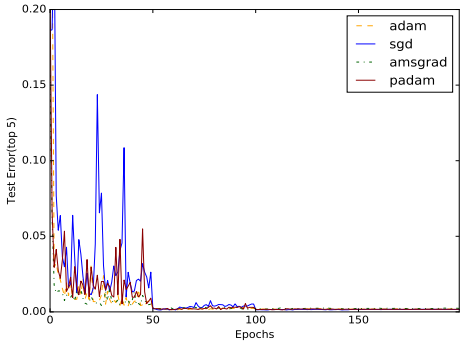
4.1 TRAIN EXPERIMENTS

Padam is compared with other proposed baselines. Figure 2 demonstrates the Top-5 Test Error and Figure 3 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 and maintains a rate of convergence between Adam/AMSgrad and SGD. 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 this partially adaptive parameter p.

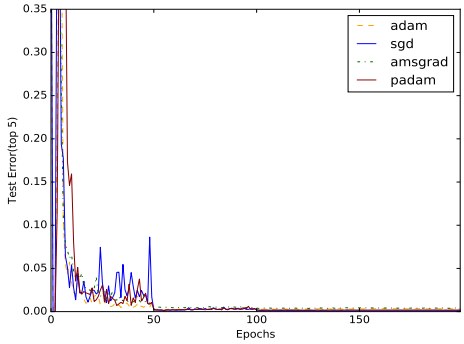
The results on CIFAR-100 can be found in the appendix.



(a) Top-5 Error for VGGNet

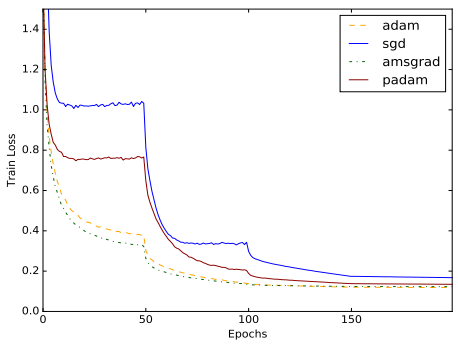


(b) Top-5 Error for ResNet

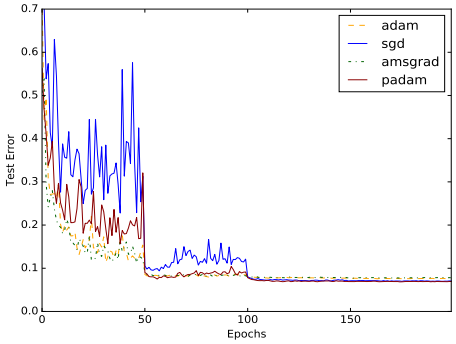


(c) Top-5 Error for Wide ResNet

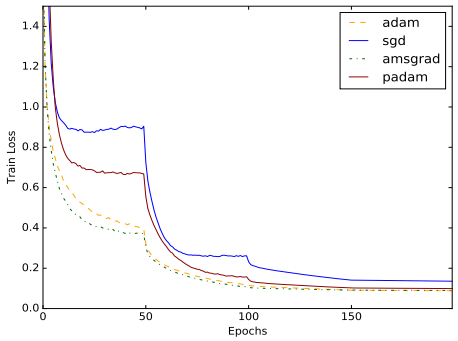
Figure 2: Top-5 error for three CNN architectures on CIFAR-10.



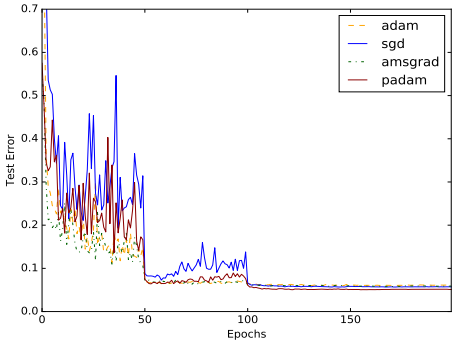
(a) Train Loss for VGGNet



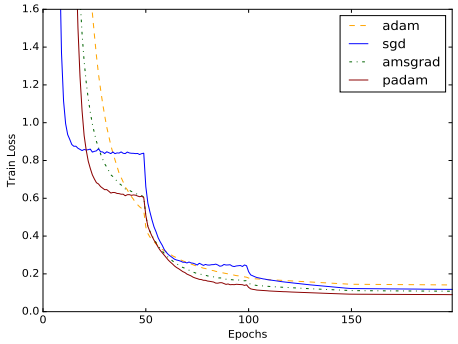
(b) Test Error for VGGNet



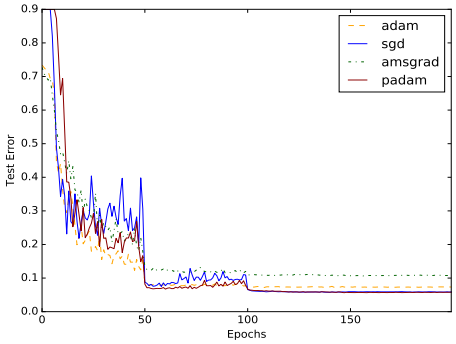
(c) Train Loss for ResNet



(d) Test Error for ResNet



(e) Train Loss for Wide ResNet



(f) Test Error for Wide ResNet

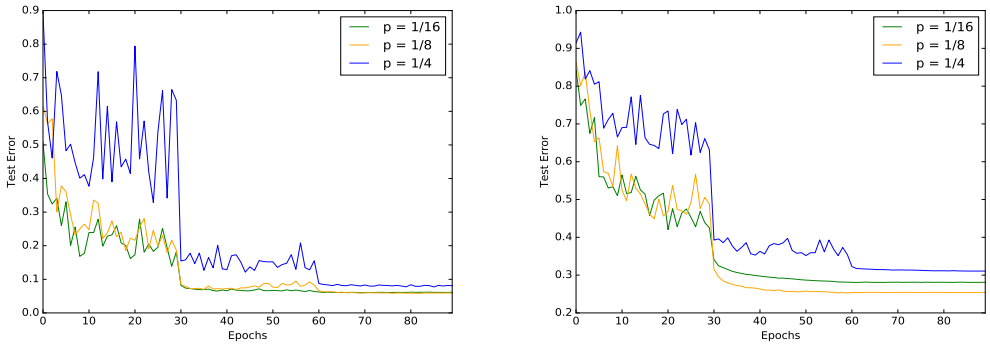
Figure 3: Train loss and test error (top-1 error) of three CNN architectures on CIFAR-10.

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**

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 of p is sub-optimal and may turn out to be sensitive to the learning rate's base value. To analyze this we perform sensitivity experiments of p against various learning rates and it turns out that p is indeed sensitive to it.



(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.

4.3 SENSITIVITY EXPERIMENTS

To evaluate the possibility that optimal value of partial adaptive parameter p is entangled with learning rate we run the sensitivity experiments. We perform experiments with three fixed values of p from 0.25, 0.125, 0.0625. For each fixed value of p we vary the base learning rate over $\{0.1, 0.01, 0.001\}$. We run each evaluation for 30 epochs on CIFAR-10 and CIFAR-100 with ResNet. We expect that this would uncover the dependence of p on base learning rate.

The results for CIFAR-10 are plotted in Figure 5. From Figure 5(b) we observe that with p = 0.25 base learning rate of 0.01 or 0.001 seems to be a more appropriate choice as compared to 0.1 owing to its better Test Error performance. As we decrease the value of p, we find that higher base learning rates start performing better as evident from Figure 5(d) and 5(f). This observation favors the argument that p is indeed sensitive to the base learning rate.

Results of sensitivity experiments on CIFAR-100 are moved to Appendix.

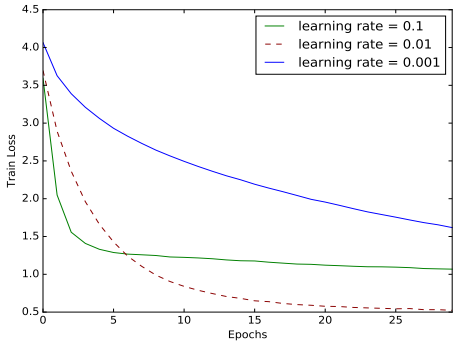
4.4 PROPOSED FURTHER STUDY

From the sensitivity experiments we can infer that while using higher values of p Padam behaves more adaptive-like (performs better with lower learning rates) and with smaller values of p Padam demonstrates a behavior closer to SGD (performs better with higher learning rates).

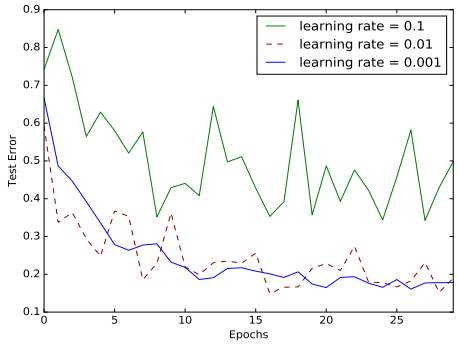
Our primary objective behind designing Padam was to achieve two things: good convergence (initially) and better generalization (finally). In order to do so we would like Padam to behave adaptive-like initially and SGD-like finally. In this way Padam would be able to exploit both worlds to their fullest within the training life-cycle.

To do so we propose to initialize Padam with high p and low base learning rate and then decay p during the training life-cycle. Correspondingly the learning rate can be mildly decreased in the middle or towards the end of the training cycle in order to generate conditions for the SGD-like Padam to converge.

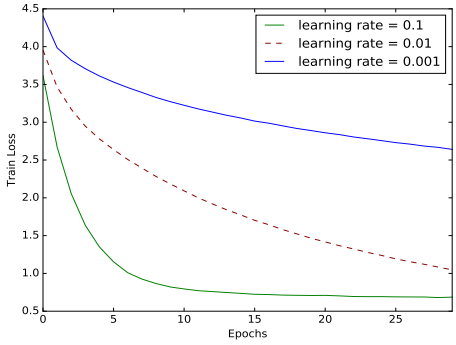
Recently, AdamW has demonstrated better generalization by decoupling the mechanism of weight decay from the update rule, this method can also further compliment Padam's result. We haven't been able to finish running these proposed experiments due to time and resource constraints.



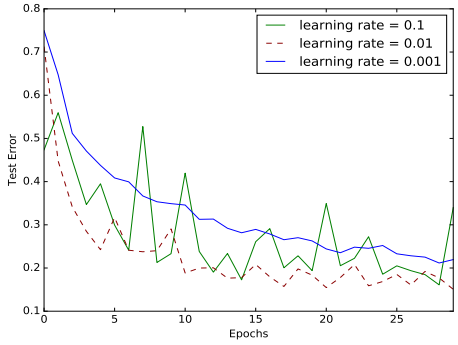
(a) Train Loss with p = 0.25



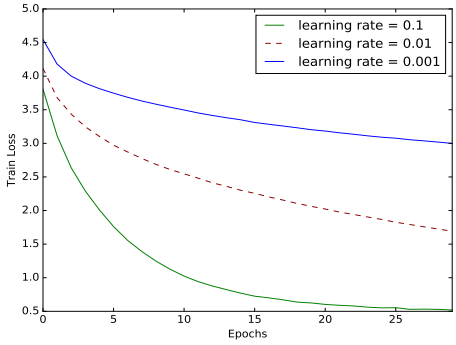
(b) Test Error with p = 0.25



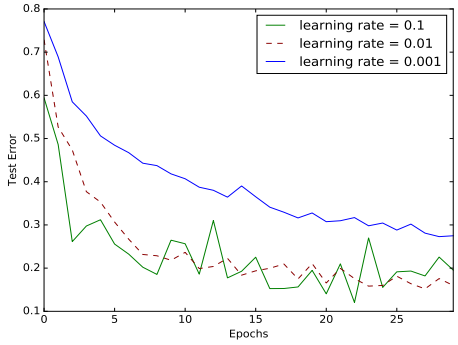
(c) Train Loss with p = 0.125



(d) Test Error with p = 0.125



(e) Train Loss with p = 0.0625



(f) Test Error with p = 0.0625

Figure 5: Performance comparison of Padam with different choices of learning rate for three different fixed values of p (0.25, 0.125, 0.0625) for ResNet on CIFAR-10 dataset .

5 Discrepancies, Suggestions and Conclusion

The authors argue that adaptive gradient methods when used with larger base learning rate gives rise to the gradient explosion problem because of the presence of the second order moment term v 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 studying the newly introduced partially adaptive *p* parameter seems to be a good direction to further this work along.

References

Timothy Dozat. Incorporating nesterov momentum into adam, 2016.

Nitish Srivastava Geoffrey Hinton and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, 2012.

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.

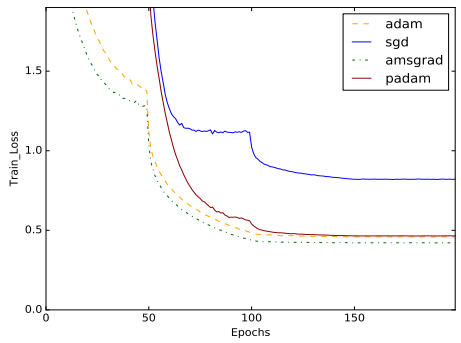
Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam, 2017.

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

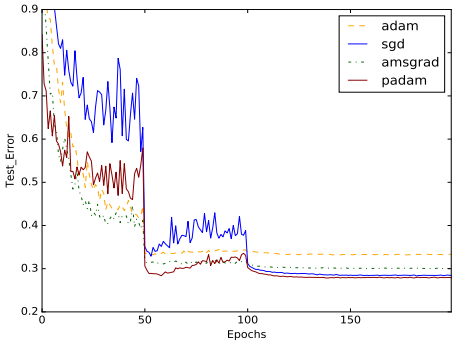
Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016.

Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012.

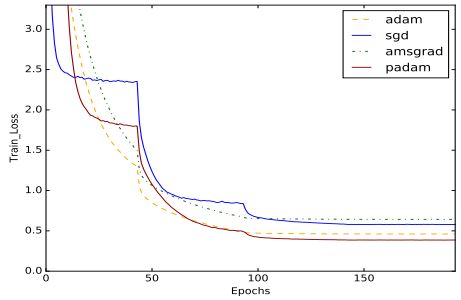
A EXPERIMENTS ON CIFAR-100



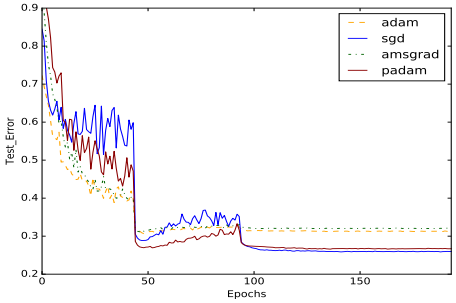
(a) Train Loss for VGGNet



(b) Test Error for VGGNet



(c) Train Loss for Wide ResNet

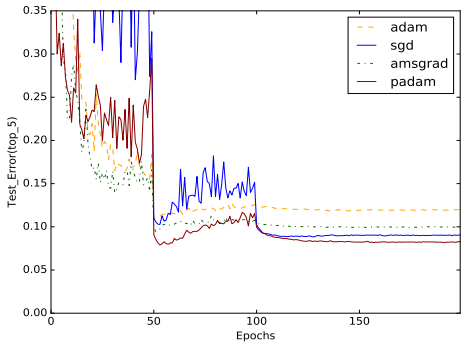


(d) Test Error for Wide ResNet

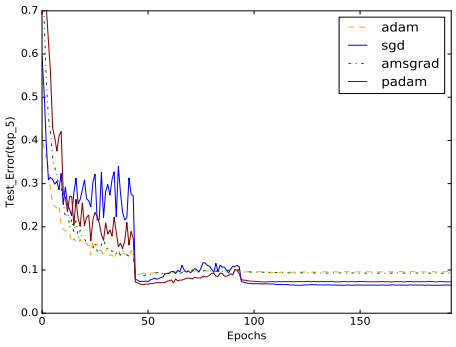
Figure 6: Train loss and test error (top-1 error) of two CNN architectures on CIFAR-100.

Table 3: Test Accuracy of VGGNet on CIFAR-100

Methods	**50th epoch**	**100th epoch**	**150th epoch**	**200th epoch**
SGD Momentum	37.29	61.18	71.55	71.54
Adam	55.44	65.67	66.70	66.65
Amsgrad	**58.85**	**68.21**	69.94	69.95
Padam	42.05	66.92	**72.04**	**72.08**

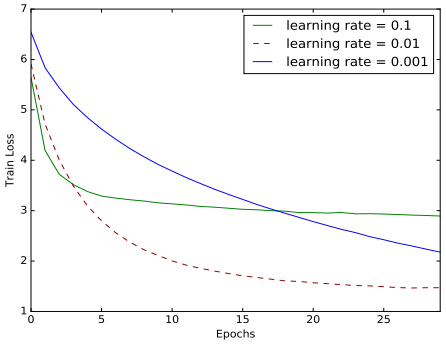


(a) Top-5 Error for VGGNet

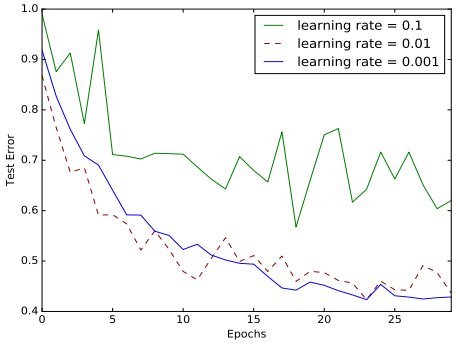


(b) Top-5 Error for Wide ResNet

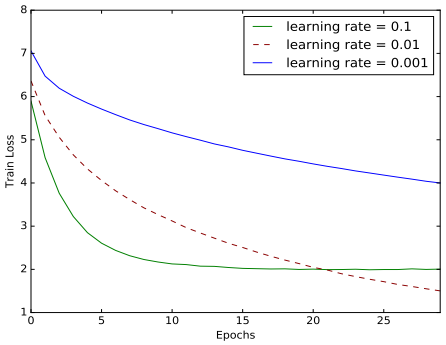
Figure 7: Top-5 error for two CNN architectures on CIFAR-100.



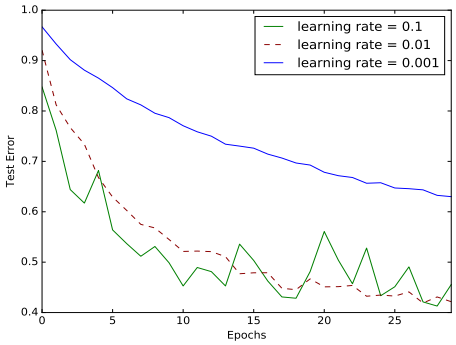
(a) Train Loss with p = 0.25



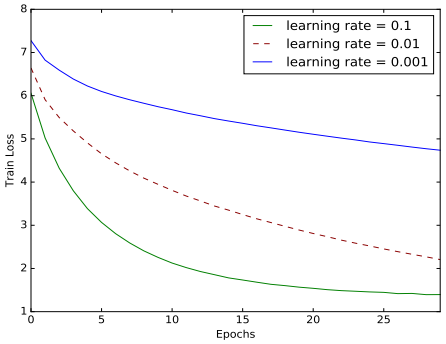
(b) Test Error with p = 0.25



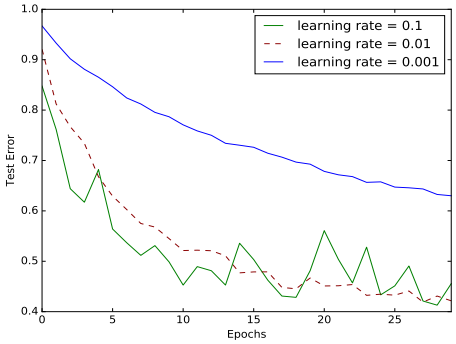
(c) Train Loss with p = 0.125



(d) Test Error with p = 0.125



(e) Train Loss with p = 0.0625



(f) Test Error with p = 0.0625

Figure 8: Performance comparison of Padam with different choices of learning rate for three different fixed values of p (0.25, 0.125, 0.0625) for ResNet on CIFAR-100 dataset.