CLOSING THE GENERALIZATION GAP OF ADAPTIVE GRADIENT METHODS IN TRAINING DEEP NEURAL NETWORK – REPRODUCE CHALLENGE

Keyan Shi & **Tong Ye** & **Chizhou Zhang**
School of Electronies and Computer Science
University of Southampton
Southampton,UK
{ks1g19, ty1u19, cz4u19}@soton.ac.uk

ABSTRACT

This paper aims to reproduce an algorithm called Padam which stands for Partially adaptive momentum estimation method. According to the original paper, this is an algorithm that combines the advantages of both Adam/Amsgrad and stochastic gradient descent (SGD): fast convergece of Adam/Amsgrad and good generalization of SGD. The reproducability of the algorithm is confirmed and further work is suggested.

1 INTRODUCTION

SGD is the most dominant method in training the deep learning network (Reddi et al., 2018). This method updates one of the parameters only in each training iteration by moving it alone the direction of the negative-gradient of the lost function on a mini-batch of training examples thus makes it slow to converge. The first successful attempt of SGD is called Adagrad (Duchi et al. (2011); McMahan & Streeter (2010)). However, the drawback of this method is that, it is not good enough when dealing with the cases that the loss functions are nonconvex or the gradient is dense (Bottou (1991)). An adaptive gradient method called Adam (Kingma & Ba (2014)) on the other hand has less computational cost. But, In the large gradients scenario, when these large gradients are quite informative and do importance, they will also be decreased rapidly due to the exponential averaging which leads to a bad convergence. So, people come up with a method benefits from both Adam and RMSprop (Reddi et al. (2019)). This method had a long-term memory to the method to ensure the convergence of the optimization algorithm. However, the author achieved to combine the advantage of the SGD and the adaptive gradient method together called Padam, which we will illsuatraite in the next part.

2 BACKGROUND

Padam aims to combine both the fast convergence of Adam/Amsgrad and good generalization of SGD with momentum. The paper unifies the Adam/Amsgrad and SGD with momentum by using a partially adaptive parameter p. This algorithm is based on the formula: $\theta_{t+1} = \theta_t - \alpha_t \frac{m_t}{\hat{v}_t^p}$ in which $\hat{V}_t = \max(\hat{v}_{t-1}, v_t)$. When $p \to 0$, this algorithm will become SGD, and when we $p \to 1/2$, we will have an Amsgrad optimizer. In the Padam algorithm, the learning rate is not as small as other adaptive gradient methods to avoid the exploding of the gradients updates.(Chen & Gu (2018)). **Algorithm 1** shows the pseudo code of the Padam algorithm.

3 EXPIREMENT

3.1 EXPERIMENTAL SETUP

All experiments are conducted on Southampton ECS GPU compute platform with Intel i7 8700 CPU and Nvidia RTX 2070 GPU (8 GB VRAM). The original code of Padam implementation (i.e. the

Algorithm 1 Partiall adaprive momentum estimation method (Padam)

Input: initial point $\theta_1 \in X$; step sizesα_t; momentum parametersβ_{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**
    $g_t = \nabla f_t(\theta_t)$;
    $m_t = \beta_{1t} m_{t-t} + (1 - \beta_{1t}) g_t$;
    $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$;
    $\hat{v}_t = max(\hat{v_{t-1}}, v_t)$;
    $\theta_{t+1} = \sum_{X, diag(\hat{v}_t{}^p)} (\theta_t - \alpha_t \cdot m_t / \hat{v}_t{}^p)$
end for

optimizer module Padam.py) is maintained and we have written our own code to run on Windows and Jupyter notebook (Python 3.7.3) and utilise torchbearer 0.5.2 and Pytorch 1.2.0 to train the model.

3.2 Dataset and Architectures

The original work run experiments on CIFAR-10 and CIFAR-100. For simplicity we will conduct all experiments on CIFAR-10 dataset. We will also train the three CNN architectures in the original paper: VGGNet, ResNet and WideResNet. The training task will run 100 epochs since we have found that the model converges before 100th epoch.

3.3 Baseline Models

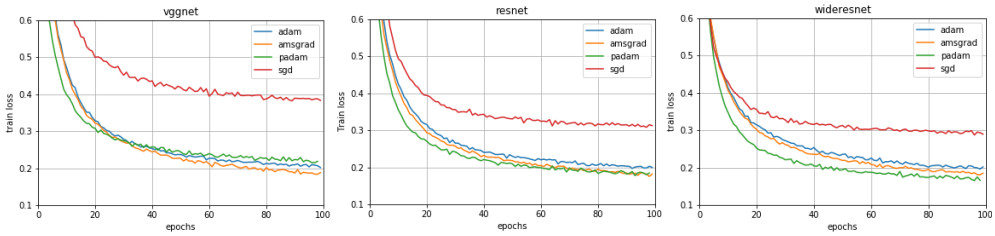
We will compare Padam against some most popular optimizers. Table below shows the hyperparameters of these optimizers and the hyperparameters are faithful to the original paper. The default p value is 1/8. However, we did not apply the fixed multi-stage learning rate decaying scheme as the original work. All experiments use cross-entropy as the loss function. Mini batch size of training set is 128 and 100 of test set.

Table 1: Baseline models and their hyperparameters

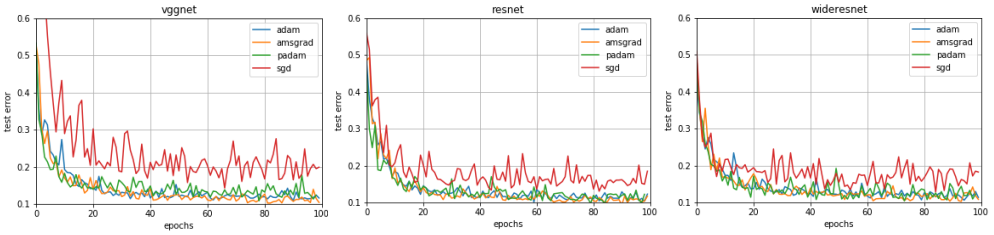
Optimizer	**Padam**	**SGDM**	**Adam**	**Amsgrad**
leanear rate	0.1	0.1	0.001	0.001
β_1	0.9	-	0.9	0.9
β_2	0.999	-	0.999	0.999
Weight decay	0.005	0.005	0.005	0.005
Momentum	-	0.9	-	-

3.4 Training Results

Figure 1 shows the train loss and test error (top-1 error) of 4 baseline models on 3 different architectures. Because we did not implement the multi-stage learning rate decaying scheme, there is no sharp loss or error decrease in our result. It can be observed that the Padam algorithm performs the best on Wide ResNet and achieves the lowest train loss during the whole training process and this result is consistent with the original work. The experiment on ResNet shows similar result. The train loss on VGGNet also has similar result with the original work: Padam has lower loss than Adam and Amsgrad before 30th epoch. Despite the multi-stage learning rate decaying scheme is not implemented, the performance is similar but we cannot observe the SGD with momentum outperforms the Adam and Amsgrad during the late stage of training. Still, we can conclude that the original work can be reproduced.



(a) The train loss of VGGNet (b) The train loss ofResNet (c) The train loss of Wide ResNnet

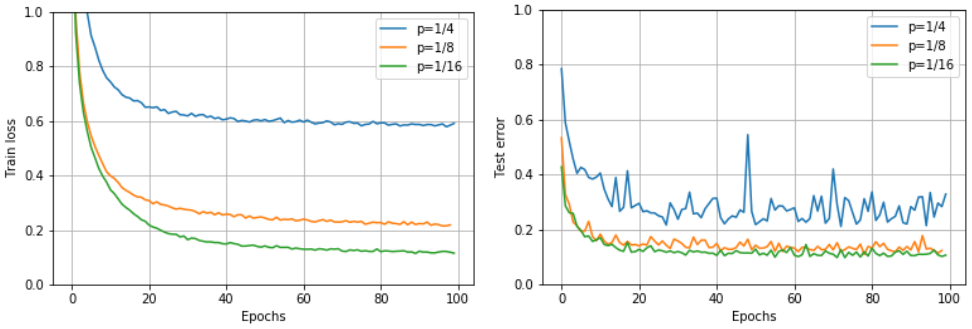


(d) The test Erroe of VGGNet (e) The test Erroe of ResNet (f) The test Erroe of Wide ResNet

Figure 1: Experiment results of different architectures and algorithms

3.5 Experiments on p-value

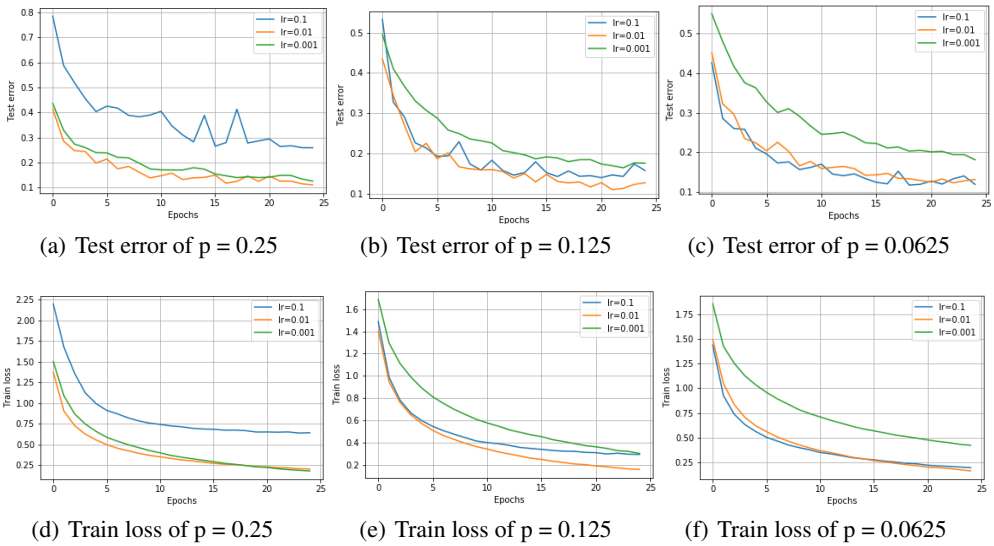
We have performed a grid search to find the optimal p for the Padam algorithm. The test is performed on CIFAR-10 and VGGNet with learning rate set to 0.1. Unlike the proposed/default value $p = 1/8$ in the original paper, it turns out that $p = 1/16$ is the optimal choice when learning rate is 0.1.



(a) The train loss (b) The test error

Figure 2: Train loss and test error (top-1 error) comparison of Padam with different choices of p for training ResNet on CIFAR-10 dataset

To find out more about the optimal choice of p and learning rate, we have tested the performance of p= {0.25,0.125,0.0625} under different learning rate= {0.1,0.01,0.001}. We have observed that when the value of p is close to zero, a larger learning rate has lower test error and loss. On the other hand, when p is close to 0.5, a smaller learning rate performs better. This is due to the Padam algorithm mechanism: it behaves like SGD when p is close to zero and behaves like Adam/Amsgrad when p is close to 0.5. This might suggest a new p value and learning rate scheme to take advantage of fast convergence of Adam/Amsgrad at the start and good generalization at the end. We can adopt a large p and small learning rate at the beginning and gradually decrease p and increase the learning rate.



(a) Test error of p = 0.25 (b) Test error of p = 0.125 (c) Test error of p = 0.0625

(d) Train loss of p = 0.25 (e) Train loss of p = 0.125 (f) Train loss of p = 0.0625

Figure 3: Train loss and test error of Padam with different p and learning rate on ResNet with CIFAR-10 dataset

4 Conclutions and Future Work

Overall, we can conclude that Padam can take advantage of fast convergency like adam/amsgrad as well as generalization like SGD with momentum. We have also found that a varying p and learning rate scheme may improve the performance of Padam and this can be explored in the future. The test is performed on the same dataset and CNN architectures. The performance of Padam on other CNN architectures or recurrent neural networks (RNNs) and generative adversarial networks (GANs) is yet to be explored.

References

Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8): 12, 1991.

Jinghui Chen and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

H Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.