Verification of Super Convergence in Convolutional Neural Networks

Shahmeer Mohsin
shahmeer@kth.se

Yunpeng Ma
ypma@kth.se

Chu-Cheng Fu
ccfu@kth.se

Abstract

The report uses various techniques to improve the performance of diverse convolutional neural networks on sample images. The main aim of the report is to prove that a relatively new technique called super convergence can drastically improve the performance of a neural network on test image data as stated in [Smith and Topin, 2017]. Meanwhile, the results of applying traditional cyclical learning rate and constant learning will be compared with the outcomes that deploy super-convergence.

1 Introduction

In a nutshell, super-convergence is the use of large learning rates with one learning rate cycle to achieve improved test data accuracy along with efficient computation. According to [Smith and Topin, 2017], the RESNET-56 network achieves a greater accuracy (92.4%) with super-convergence than a constant learning rate (91.2%) in a fewer number of iterations (1000 for super-convergence and 8000 for the constant learning rate) for the CIFAR-10 dataset. For complexity purpose, we consider image data sets such as TinyImageNet[1], Oxflowers[2] and CIFAR-10[3]. For comparison purposes, we illustrate the results attained after experiments on different networks including MobileNetV2[Sandler et al., 2018], VGGNet[Simonyan and Zisserman, 2014] and a simple convolutional network. On the other hand, as deep neural networks are getting deeper and more complex these days, fine tuning suitable hyper-parameters for training is a necessity. Finding ways to balance regularization is also an important aspect for generating good training results. We gather ideas and theories in [Smith, 2018] to train the networks without any pre-trained parameters and evaluate the performances with a constant learning rate, cyclical learning rate and using super-convergence. After this, a comparison is carried out to prove that the test performance of the network is improved by the application of super-convergence to complex networks like MobileNet.

2 Related Work

Without pre-trained weights, it is quite challenging to pick satisfactory hyper-parameters. A related paper, [Smith, 2018] that describes the process of selecting network hyper-parameters is inspiring for novices in neural network training tasks. Learning rate tuning in particular is always an annoying task that bothers many engineers working with deep neural networks. The paper, [Smith, 2017] brings us a new concept that can eliminate this obstacle more or less. Finally, super-convergence is a relatively new technique and there is very little help regarding the technique online. The source of this project is the paper, [Smith and Topin, 2017]. The paper carries out a comparison of the performances of networks with and without super-convergence for different data sets. Most of the methodology carried out for the project is based on the paper.

[1]https://github.com/seshuad/IMagenet
[2]http://www.robots.ox.ac.uk/ vgg/data/flowers/17/
[3]https://www.cs.toronto.edu/ kriz/cifar.html

3 Data

3.1 CiFAR 10

We import CIFAR-10 from Keras directly. It contains 60,000 color images, each image has 32x32 pixels. There are totally 10 different classes and each class includes 6000 images. We selected 50,000 images as training data and 10,000 as test data.
CIFAR-10 is a very standard data-set. Basically, we use it as a reference for different neural networks. During the experiment, in order to overcome over-fitting caused by small size of the data-set, we used real-time data augmentation such as width shift and horizontal flip. In the experiment, this method dramatically reduced over-fitting as expected.

3.2 Oxflower17

Oxflower17 has a higher number of pixels per image compared to CIFAR-10. The dimension of Oxflower17 images is 224x224. However, it has 17 categories with only 80 images for each class. The data set presents a higher complexity but low labels situation.
As the labels of this data sets are less, we also do real-time data augmentation including random horizontal flip and random degree rotation.
We import this data-set from tflearn package and split it in $80/20$ for training and testing sets respectively.

3.3 TinyImageNet

In order to prove our ability to train complex data-sets, a lager data set is necessary to test the performance of different neural networks. However, taking the computation time into consideration, we take TinyImageNet as our option. TinyImageNet has 200 classes and each class has 500 training images, 50 validation images and 50 test images. The dimension of each image is 64x64. The TinyImageNet dataset is a compulsory dataset to be trained in the Stanford CNN course. We downloaded it from Tiny ImageNet Visual Recognition Challenge[4].

4 Methodology

4.1 Building Neural Networks

For the following experiments, we consider Keras along with Tensorflow as front-end and back-end. We generate a simple convolutional neural network, a VGG16-like neural network with batch normalization and MobilenetV2[5]. All of the networks above are trained without pre-trained weights.

4.2 Tuning Hyper-Parameter

This subsection explains the hyperparameter tuning carried out for the MobileNet convolutional neural network applied to the OxFlowers dataset.The hyperparameter setting for the other networks we trained on other datasets is stated in the Experiments and Analysis section. The hyperparameter tuning process is as follows:

4.2.1 Learning Rate

For finding the best learning rate, we consider a method [6] that analyzes the training loss vs learning rate plot.

[4]https://tiny-imagenet.herokuapp.com/
[5]https://github.com/keras-team/keras-applications.git
[6]https://github.com/titu1994/keras-one-cycle

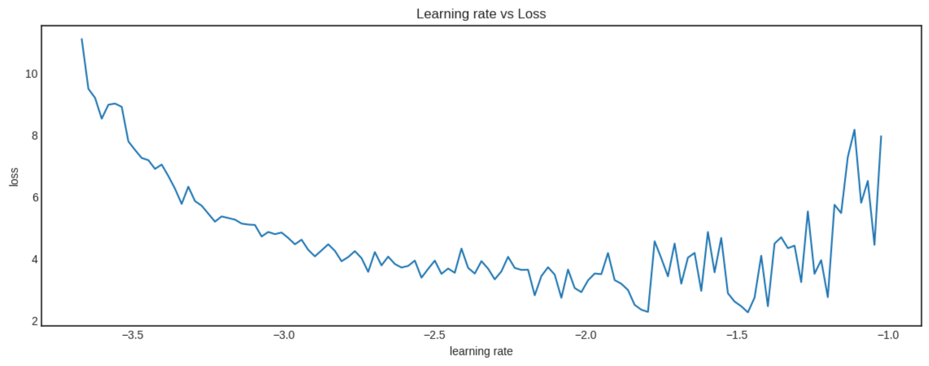

Figure 1: Learning Rate vs Loss Plot

The x-axis of the plot represents logarithmic learning rates and the y-axis represents the corresponding loss achieved. As seen in the figure, the loss has a very large value initially. It decreases to its minimum value before starting to rise again to a large value due to overfitting and divergence. For this plot, a learning rate of $10^{-2.12}$ was selected as the maximum learning rate. This is because, the point at which overfitting and divergence occur is around 10^{-2}. For the maximum learning rate, we select a point which is to the left of this overfitting/divergence point but close to it, and thus select $10^{-2.12}$.

4.2.2 Momentum Selection

For picking a good momentum, we observe that momentum range test is not useful when for looking for momentum[Smith, 2018]. However, when we conduct tests on different networks and data sets, all results show that a momentum of 0.9 has a best overall loss performance. Therefore, we pick it as our default value for training.

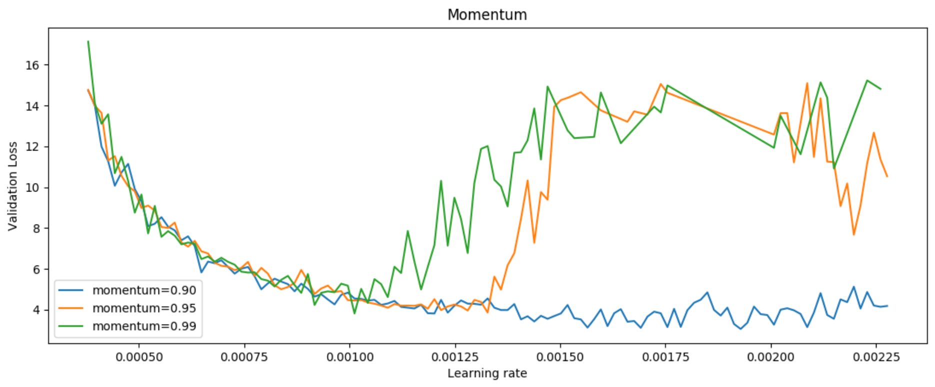

Figure 2: Momentum, Learning Rate vs Loss Plot

As the plot illustrates, the momentum of 0.9 gives the lowest validation loss for higher learning rates and does not undergo overfitting/divergence for higher learning rates, therefore the optimal momentum is selected as 0.9. This optimal momentum is then used when applying MobileNet on the Oxflowers17 dataset for a constant learning rate and cyclical learning rate. But, in the case of super-convergence, an inverted triangular momentum with one cycle is used. The peak of this inverted triangular momentum is at 0.9.

4.2.3 Weight Decay Selection

For finding suitable weight decay, a coarse search was first conducted within a range of learning rates to examine the validation loss.

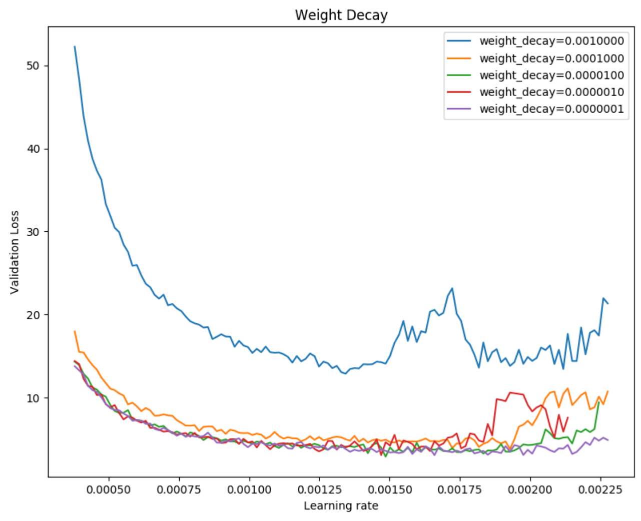

Figure 3: Weight Decay Coarse Search

As the plot illustrates, lowering the weight decay during the coarse search lowers the loss. The lowest loss is thus attained for a weight decay of 10^{-7}, which is represented in the above plot in purple. Then a fine search was implemented for finding the optimal weight decay as illustrated below:

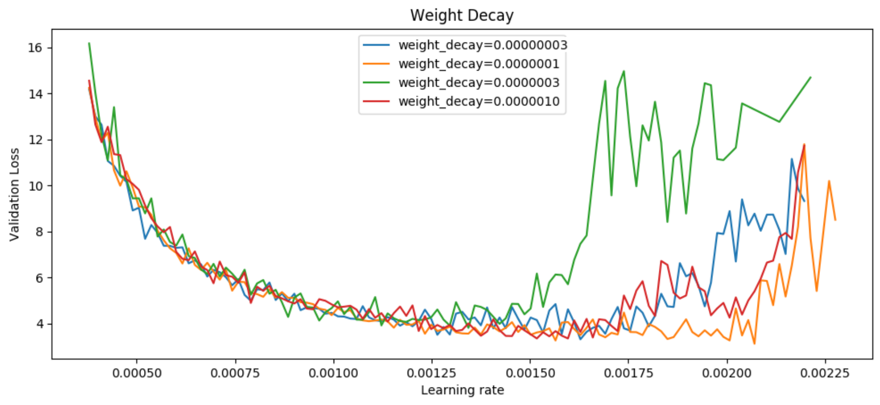

Figure 4: Weight Decay Fine Search

As Figure 4 illustra tes, there is a limit until which we can decrease the weight decay and get a lower loss curve. According to Figure 3 the blue curve in Figure 4 should have the lowest loss as it has the lowest weight decay, but we get a lower loss curve for a higher weight decay of 10^{-6}, which is represented in the figure in orange. Thus, decreasing the weight decay further beyond this range does not give a lower overall loss curve. So, we select 10^{-6} as the optimal weight decay.

4.3 Cyclical Learning rate

[Smith, 2017] explains a technique for setting the maximum and minimum learning rates for cyclical learning rate training. Minimum learning rate = $1/3$ or $1/4$ of the optimal constant learning rate and maximum learning rate = the optimal constant learning. Also we pick the step size as 2 to 10 times the iterations per epoch.

4.4 Super Convergence

For super-convergence, the optimal constant learning rate is selected as the minimum learning rate and the maximum learning rate is set as 10xOptimal Learning Rate as suggested in [Smith, 2018]. In addition, the learning rate will be set to lower than the starting learning rate in the last 10% of the iteration to assist the training results converge better. The other thing we applied is the cyclical momentum (one cycle) that is mentioned in [Smith, 2018]. We simply set the momentum we pick in section 4.2.2 as the maximum momentum and the minimum one as 0.85. It runs through iteration with one cycle policy mentioned above as well but starts at its maximum point.

5 Experiments and Analysis

5.1 Initial Settings

According to the methodologies in section 4.2, the hyper-parameters we determine are shown in Table 1.

Table 1: Initial Hyper-Parameter

Network	Dataset	Learning rate(constant)	Weight decay	Momentum
Simple	Cifar-10	0.04	$3 * 10^{-7}$	0.9
Simple	Oxflower17	0.0084	10^{-6}	0.9
VGG16	Cifar-10	0.01288	10^{-6}	0.9
VGG16	Oxflower17	0.01023	10^{-6}	0.9
MobileNetV2	Cifar-10	0.002	$3 * 10^{-6}$	0.9
MobileNetV2	Oxflower17	0.00758	10^{-6}	0.9
MobileNetV2	TinyImageNet	0.16	10^{-7}	0.9

These parameters are used for training neural networks under constant learning rate policy.

5.2 Simple Convolutional Neural Network

For the simple convolutional neural network, we can observe that in Figure 5, the cyclical learning rate methodology assists the training process a lot in the beginning, and the final accuracy for a constant learning rate and cyclical learning does not have a large difference. On the other hand, it is surprising that super convergence completely fails for this network and dataset.
However, if we look into Figure 6, a different data set is now applied on the same network. We still follow the same methodology and get quite fair results. Cyclical learning method tends to converge faster than super convergence in the early epochs. But, if we compare the accuracy after some epochs, super convergence tends to have a higher potential to reach more ideal accuracy.

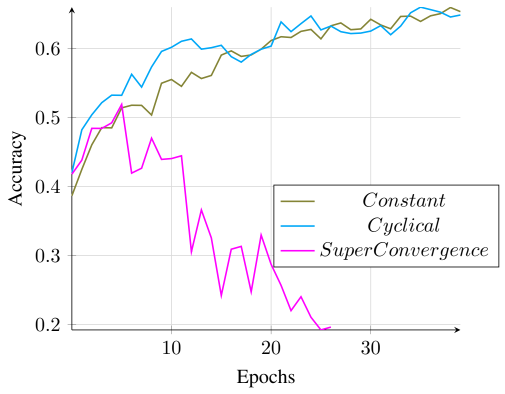

Figure 5: Accracy vs Epochs plots, Cifar-10 on Simple CNN

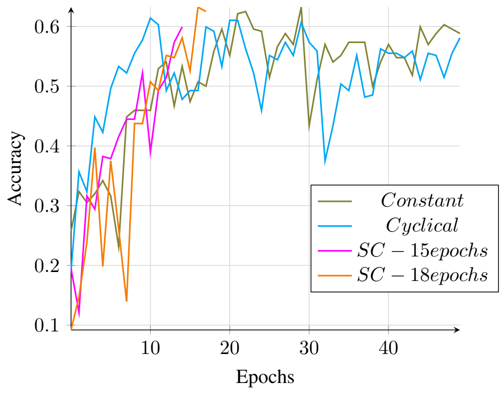

Figure 6: Accracy vs Epochs plots, Oxflower17 on Simple CNN

5.3 VGG16 Network

For the VGG-16 neural network, the same methodology is followed. First, when we observe the result of Figure 7, we can see that super convergence has large improvement compared to Figure 5. However, the accuracy it reaches is not higher than constant learning rate approach too much. Anyways, the cyclical learning rate method wins in the very beginning and can be selected as an optimal training approach in this network data combination. However, super convergence can still be an option. We can take the step size of cyclical learning rate into consideration when picking the final epochs for super convergence. Since there might be accuracy peaks and valleys existing in cyclical learning rate, choosing an epoch number that stops at the valley will be a potential strategy for super convergence to get better accuracy than cyclical learning rate.
In figure 8, we can see a breakthrough of super convergence. If we select 30 epochs as a standard, super convergence wins over the other two learning approaches. Also, as we discuss in Figure 7, here, we luckily picked a super convergence epoch which stops at an accuracy valley of cyclical learning rate approach. This results in highest accuracy if only 20 epochs of training time is available.

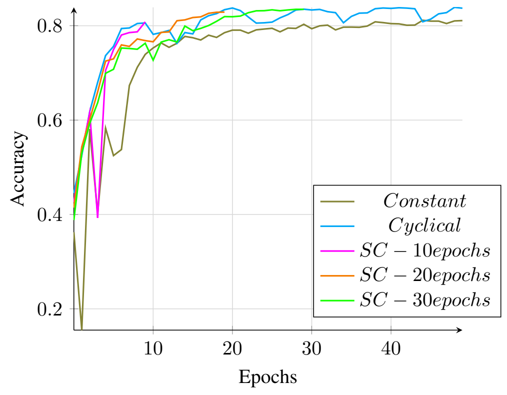

Figure 7: Accracy vs Epochs plots, Cifar-10 on VGG16

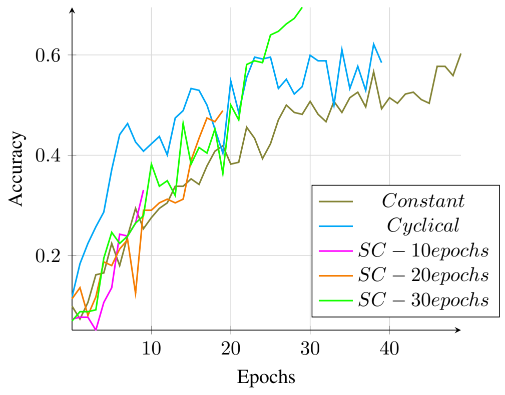

Figure 8: Accracy vs Epochs plots, Oxflower17 on VGG16

5.4 MobileNetV2

On MobileNetV2 network, Figure 9 shows that super convergence can perform well in only 12 epochs. Also, as the epochs get larger, the accuracy can still reach a little bit higher. The cyclical learning rate, however, has a bad result. This implies that sometimes even when we cannot get a good result from cyclical learning rate due to some bad tuned parameters, we can still reach a better accuracy via super convergence.Figure 10 gets a similar result as Figure 9. As for figure 11, we trained the MobilenetV2 with TinyImageNet, in order to see if the theory is still applicable on super complex data sets. As we can observe, super-convergence is still very effective and can even reach an accuracy which is 20% higher than than the accuracy attained by applying a constant learning rate with only half the number of the iterations.

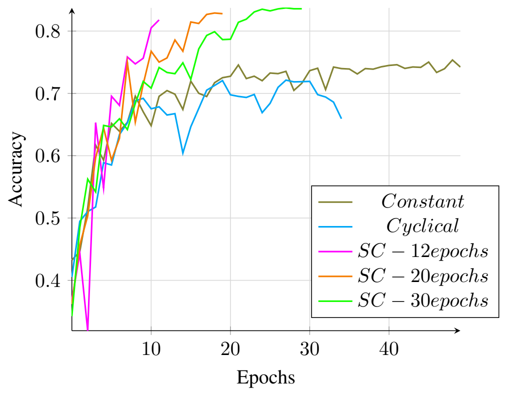

Figure 9: Accracy vs Epochs plots, Cifar-10 on MobilenetV2

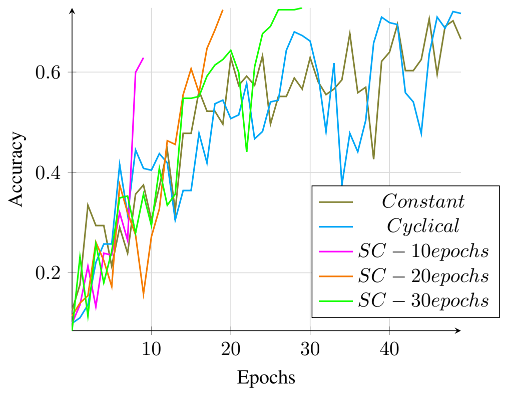

Figure 10: Accuracy vs Epochs plots, Oxflower17 on MobilenetV2

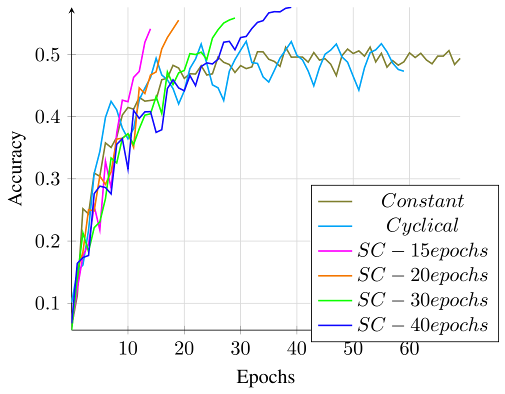

Figure 11: Top-1 Accracy vs Epochs plots, TinyImageNet on MonbilenetV2

6 Conclusion

The result we present shows that super convergence is not a panacea for all condition. One thing [Smith and Topin, 2017] fails to address is the cases where super-convergence does not perform well. Through experimentation, our team observed that super-convergence performs well for complex convolutional neural networks like MobileNet, but the performance for simpler networks is below par. However, we observe that super convergence becomes more effective when there is limited labeled data which matches what the paper mentioned. The simplest network we implement is not compatible for super convergence with CIFAR-10, but super convergence truly helps in improving accuracy when we triain a less labelled data set, Oxflower17 on the simple network.
To sum up, in spite of the fact that super convergence can help with effective computation and even higher accuracy, the mechanism behind it is still unknown. Further research should be done for a more insightful picture of its reason.

References

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017. URL `http://arxiv.org/abs/1708.07120`.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. URL `http://arxiv.org/abs/1801.04381`.

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

Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. URL `http://arxiv.org/abs/1803.09820`.

Leslie N. Smith. Cyclical learning rates for training neural networks. pages 464–472, 2017. doi: 10.1109/WACV.2017.58. URL `https://doi.org/10.1109/WACV.2017.58`.

7 Appendix

7.1 Learning Outcomes

Shahmeer Mohsin

The project has instilled the following skills in me:

- How to find the optimal hyper-parameters for various learning techniques like super-convergence, cyclical learning rate and constant learning rate.

- The strong points and shortcomings of super-convergence. I plan to present a research paper based on my findings at some conference and take this up as a thesis topic.

- Characteristics of various go-to image data-sets used in Deep Learning.

Yunpeng Ma

- From this project, I got to know the methodology of research work such as reading papers, implementing the methodologies from the paper into real neural network, which is beneficial for our further thesis research.

- I also learned how to build a module for convolutional neural network and practiced programming skills of python.

- Finally, I understood how convolutional neural network really works apart form the lectures and got some practical skills in training ConvNet.

Chu-Cheng Fu

- The first thing that I acquire is knowing how to build my own neural network by Keras. Though it is a relatively easy front end, I still discover many functionality that can be modified in in the bottom. For example, we have to implement different regularization parameters, and some of them are not changeable in the top layers. Thanks to the comprehensive documentation of Keras website, I am now able to revise networks in a bottom level.
- Secondly, I learn not only how to build environment for deep learning on different operating systems,Linux, MacOSX, but also have a chance to make use of Google Cloud Computing which is a great tool if I have enough money in the future.
- Thirdly, I think I made myself like deep learning a lot after the project, I was not able to be patient before in other subjects. But I find myself so concentrated when no matter reading papers about deep learning and also training models, writing codes. I think really learnt a lot about how deep neural networks should be tuned by considering different trade off.