# An Improvement on the Bag of Tricks for Image Classification with Convolutional Neural Networks

1st Vivek-Koshy Thomas Poovathoor
*School of Electrical and Computer Engineering*
*Georgia Institute of Technology*
Atlanta, GA 30332
vpoovathoor3@gatech.edu

***Abstract*—The work done in [1] managed to increase the performance of the ResNet50 V1 model on ImageNet. The improvements were accomplished using efficient training methods and refinements in the training process. This work focuses on the latter. By applying and improving these methods on the CIFAR100 dataset with the CIFAR ResNet56 V1 model, it was shown that these training refinements can lead to a overall 7.73% performance increase in top-1 validation accuracy. Learning rate schedules, warm-up period, label smoothing, mixup and distillation training methods were studied. This work showed that the most effective way to further increase performance of the CNN model was through using spatial mixup techniques (stitch-mixup) rather than using the interpolation-mixup techniques used in [1]. These refinements are implemented without making changes to the actual architecture of the ResNet56 V1 model.**

***Index Terms*—CNNs, ResNet, image classification, training refinements, spatial mixup training, knowledge distillation, learning rate scheduling**

## I. Introduction

### A. Background

Rather than changing model architectures and designs to improve classification accuracy, changes in training procedures can yield significant improvements. Techniques such as adjusting learning rate schedules, using different optimizer functions, and data augmentation are explored in this work and compared to the reference literature, i.e. [1].

The work done in [1] showed that adding successive training techniques to a ResNet50 Convolutional Neural Network (CNN) model led to significant improvements in model accuracy. Although modifications to the CNN architectures were explored, the work done by the authors reveal that training refinements lead to better classification scores. The authors in [1] divided up their improvements between "efficient training" and "training refinements". Once they implemented linear scaling learning rate, learning rate warmup, zero-$\gamma$ initialization, no bias decay, and low-precision training for efficient training, the authors tweaked the ResNet architecture to the ResNet-D variant to achieve a top-1 and top-5 validation accuracy of 77.16% and 93.52%, respectively. Afterwards the authors implemented training refinements such as cosine learning rate decay, label smoothing, mixup-training and knowledge distillation. When stacked up together, these refinements led to a top-1 and top-5 score of 79.33% and 94.66%, respectively. These training refinements led to a 2.81% increase in top-1 validation accuracy on ImageNet compared to the ResNet-D model architecture with efficient training methods implemented.

This paper will focus on exploring and improving specifically the training refinements used in [1] on the CIFAR100 dataset. This work used the ResNet56 model, which is available via GluonCV's "model zoo" based on MXNet, because the architecture is compatible for the 32x32x3 images in CIFAR100. This decision was made in accordance with the hardware limitation (single Nvidia V100 GPU) and time-constraint for this work.

### B. Project Overview

Compared to the initial proposal for this study, this work shows that the changing the optimizer function for the baseline model or changing the learning rate schedule did not improve classification accuracy or training speed. Instead, tweaking the warmup period and using spatial-mixup instead of interpolation-mixup techniques led to improved classification accuracy.

This work explores and implements the following training refinements: warmup period, learning rate scheduling, label smoothing, (interpolation) mixup training, and knowledge distillation. Implementations of these refinements in the training-loop were taken from the GluonCV code in [1]. However, the baseline ResNet56 model and preparation of the CIFAR100 dataset were done independently in MXNet on a AWS "p3.2xlarge" instance. In addition, the implementation of spatial-mixup techniques, the creation and training of the teacher model used for distillation, and testing of step and polynomial learning rate schedules were implemented independently. The efficient training methods are not studied because CIFAR100 dataset is much smaller than ImageNet and thus faster to train on. Finally, this work avoids modifying ResNet56 architecture. The code can be found here.

### C. Paper Outline

A baseline model is created for the CIFAR100 dataset using the ResNet56 architecture and without any training refinements in Section II. In the same section the performance of the baseline model will be compared to implementations with different optimizers and then the training refinements used in [1] will be applied. In Section III improvements in the training refinements will be explored.

## II. Establishing a Baseline Model

The baseline model uses the (CIFAR) ResNet56 V1 architecture from the GluonCV model zoo since it is compatible with the CIFAR dataset (input layers accept 32x32x3 images). This model was chosen since it has the closest number of layers compared to the ResNet50 model used in [1] and is based on ResNet implementation found in [3]. This ResNet56 model was built for the CIFAR10 dataset. However in this work the model is retrained on CIFAR100 using the following pre-processing pipeline:

- Randomly flip the images horizontally with 0.5 probability.
- Add random brightness, saturation and hue.
- Add random lighting.
- Pad the images such that the dimensions are 40x40x3 and then conduct a random crop to 32x32x3.
- Transpose the image such that the dimensions change from $(height, width, num\_channels)$ to $(num\_channels, height, width)$ and map the values from $[0, 255]$ to $[0, 1]$.
- Normalize the mean and standard deviations using $[0.485, 0.456, 0.406]$ and $[0.229, 0.224, 0.225]$, respectively.

For testing, the only operations that were done were to transpose the dimensions of the image, map the values to a range of $[0, 1]$, and normalize the mean and standard deviations using the same values as training. The optimizer function used was Nesterov Acclerated Gradient (NAG), initialized to a rate of 0.1 and then step-decayed at the 30th, 60th, and 90th epochs by 10. The training was conducted for 120 epochs with a batch-size of 256. Table I shows the comparison of the baseline model with the baseline model produced in [1].

### A. Varying Optimizer Functions

It was hypothesized (in the proposal) that using a different optimizer function would improve baseline performance, namely Adam. However as Table II indicates, changing the optimizer function did not lead to increased performance. Adam is an adaptive learning rate optimizer that builds upon benefits of RMSprop and Stochastic Gradient Descent with momentum. It computes the learning rate for each weight in the neural net using the first and second moments of the gradient [12]. However, despite its promises, Adam does not seem to converge to optimal solutions for complex deep learning tasks [12, 13]. Additionally, [13] shows that Adam and Adadelta do not generalize well and better results are obtained with SGD with momentum [14]. This is likely due to the fact that Adam's use of moving averages of squared gradients [12] does not converge to an optimal solution. Nadam is an improvement on Adam as it incorporates Nesterov Momentum [15]. However, even though the accuracy score slightly increased compared to Adam, the training time was five times longer than the baseline's average training time (per epoch).

Adam and the other optimizers showed no compelling reason to proceed further with. Therefore, the baseline optimizer function was kept the same and the emphasis was redirected to studying the training refinements to improve performance.

TABLE I
Comparison of baseline models.

| Model Name | Baseline Scores | |
|---|---|---|
| | ***Top-1*** | ***Top-5*** |
| ResNet50 [1] | 75.87 | 92.70 |
| ResNet56 | 74.92 | 95.66 |

TABLE II
Comparison of optimizer functions on ResNet56 model.

| Model Name | *Top-1* | *Top-5* | *Train Time per Epoch (sec)*[a] |
|---|---|---|---|
| NAG (Baseline) | 75.87 | 92.70 | 31.639 |
| Adam | 72.74 | 94.44 | 29.902 |
| Nadam | 73.55 | -[b] | 150.799 |
| Adadelta | 71.54 | 94.44 | 71.323 |
| RMSProp | 71.49 | 94.07 | 31.308 |

[a] Average training times in seconds per epoch.
[b] Top-5 accuracy not recorded at training for Nadam.

### B. Adding Training Refinements

With the optimal baseline model selected, the training refinements from [1] were sequentially added in order to increase model performance and recreate the results from the literature.

First, the learning rate schedule was changed from a step decay of 10% every 30 epochs to a cosine learning rate decay schedule. The authors found this cosine annealing recommendation in [2] but simplified it without using warmup restarts. This strategy utilizes the fact that the cosine schedule slowly decays the learning rate in the beginning unlike the step schedule [1]. This allows the learning rate to remain large in the beginning, which allows for faster convergence, and then almost decays linearly in the middle. Towards the end of training, the learning rate is slowly decayed again which allows for better accuracy/lower loss. With this new schedule, a linear warmup of 5 epochs [1, 4] is also introduced. In the linear portion, the learning rate is brought up to the initial learning rate (0.1) and then the cosine schedule is started.

Afterwards label smoothing was added. When smoothing is not used and 'hard labels' are used for classification, the difference between the activation functions of the correct class and the incorrect class is much greater - thus increasing the opportunity for overfitting. With label smoothing, the differences between the activations of the correct class and incorrect classes are reduced (made less drastic) and the variations between the incorrect classes are also reduced (made constant) [16].

Mixup training was then added as a form of data augmentation. In mixup, two random images and their labels are linearly interpolated together to produce a new sample. The new sample is generated according to Eq 1 and 2.

$$\hat{x} = \lambda x_i + (1 - \lambda) x_j \tag{1}$$

$$\hat{y} = \lambda y_i + (1 - \lambda) y_j \tag{2}$$

This form of mixup is known as interpolation-mixup and thus improves accuracy by reducing overfitting. Just like in [1], the newly formed sample, $(\hat{x}, \hat{y})$, is used for the duration of 200 epochs. The training time is extended for better convergence [1]. The parameter $\lambda \in [0, 1]$, which is drawn from Beta$(\alpha, \alpha)$, (where $\alpha = 0.2$ by default) controls the amount of mixup from both random images and their labels.

And finally, knowledge distillation was added. The CIFAR ResNext29 16x64d model [11] was used as the teacher model. This model was trained with the cosine decay schedule and label smoothing on CIFAR100 with top-1 and top-5 validation accuracies of 87.17% and 97.53%, respectively. The loss function was changed to a distillation cross-entropy loss with the temperature hyper-parameter set to $T = 20$. A slightly different architecture was chosen as the teacher because the ResNet110 V1 model (for CIFAR) had top-1 and top-5 scores of 79.36% and 95.02%, respectively, when trained with label smoothing and cosine decay. Therefore the model with greater accuracy scores was chosen as the teacher.

Table III shows the results of stacking one training refinement after the other. The overall performance increase from the baseline model to the final iteration (with all refinements) is 7.234% and 0.094% for top-1 and top-5 validation accuracies, respectively. The work done in [1] revealed that the training refinements alone lead to a 2.679% and 1.208%, respectively. Because the CIFAR100 dataset is less complex and similar in dimensions compared to ImageNet, the training refinements on the CIFAR ResNet56 model are more greater. Nonetheless the benefits of stacking these refinements together was successfully replicated.

TABLE III
TRAINING REFINEMENTS TO CIFAR RESNET56 MODEL.

| Refinement | *Top-1* | *Top-5* |
|---|---|---|
| Baseline | 74.92 | 95.66 |
| + cosine decay | 75.99 | 95.55 |
| + label smoothing | 78.38 | 94.70 |
| + mixup | 79.10 | 95.18 |
| + distillation | 80.34 | 95.75 |

TABLE IV
COMPARISON OF LEARNING RATE SCHEDULES WITH 5- AND 20-WARMUP EPOCHS AND DIFFERENT LEARNING RATE DECAY INTERVALS.

| Learning Schedule | *[30, 60, 90] Period* | | *[40, 80] Period* | |
|---|---|---|---|---|
| | **5-epoch** | **20-epoch** | **5-epoch** | **20-epoch** |
| step decay | -[a] | 76.00 | 76.18 | 76.77 |
| poly decay | 76.15 | 75.88 | 75.46 | -[b] |
| cosine decay | 75.99 | **76.99** | -[c] | 76.54 |

[a] This is baseline setting from [1] and thus not collected.
[b] Not collected as poly-decay did not improve results.
[c] Not collected as decay period did not improve 20-epoch result.

## III. IMPROVEMENTS TO TRAINING PROCEDURES

### A. Comparison of Learning Rate Schedules

The learning rate schedules and warmup periods were first explored to find improvements. It was first hypothesized that reducing the epoch-decay interval from [30, 60, 90] to [40, 80] would improve results because the decay will be more gradual - leading to better convergence. However, this only applied to step-decay and not cosine or polynomial as shown in Table [IV]. This is likely because cosine and polynomial schedules already decay the learning rate linearly in the middle of training and shortening the interval leads to slightly sub-optimal accuracy values. Step decay on the other hand drastically reduces the learning rate and thus spacing out the intervals seem to prevent the accuracy of the model from being prematurely reduced.

The cosine annealing strategy with a 20-warmup epoch performs the best. The work in 1 uses a learning rate decay interval of $[30, 60, 90]$ epochs with a 5-epoch warmup period. A 20-warmup epoch may allow for faster convergence as the learning rate is increased to 0.1 over a longer period of time before being decayed. Compared to the ResNet56 baseline model, this led to a $2.7634\%$ increase in top-1 accuracy. With label smoothing added to the 20-warmup period, the top-1 accuracy increased to $78.44\%$ but the top-5 accuracy decreased to $94.06\%$. This decrease in the top-5 score may be due to the fact that label smoothing reduces drastic gaps between the incorrect-class predictions across the board.

Despite these improvements however, the 20-warmup period did not seem to help improve accuracy results when mixup was added. Mixup training seems to require longer training periods for better convergence and therefore 200-epoch training period was used [1]. With a 20-warmup period, the training of the model with mixup may not likely converge as well as the 5-epoch warmup, as the top-1 score was $78.77\%$ while top-5 improved to $95.57\%$. When mixup was turned off for the first 20 epochs, the training accuracies were lower with top-1 and top-5 scores of $79.04\%$ $95.05\%$, respectively. Turning off mixup for 20 epochs also led to poor convergence in the overall training. Given these lower scores during mixup training, the 20-epoch warmup refinement was abandoned and the scheduling strategy in the baseline model was used moving forward.

### B. Spatial Mixup Techniques

To further increase the accuracy of the model, a different variation of mixup was investigated. The technique used in the baseline model is interpolation-mixup. Here spatial mixup (stitching) is explored, namely horizontal and vertical stitching as recommended by [17]. The idea is to randomly stitch portions of two randomly sampled images together - thus forming a new image. The label of this corresponding image would be constructed using Eq 2. A Beta distribution is still used to choose the random value $\lambda$. The new image is formed by concatenating a portion $P = \lambda L$ of an image $x_i$ with the remaining $(1 - \lambda)L$ portion of the other image $x_j$, where $L$ is the dimension of choice of the randomly sampled images. Fig. 1 shows an example of two CIFAR100 images stitched together vertically and horizontally. The results of spatial mixup on the ResNet56 model are shown in Table V. Results

with and without knowledge distillation, using ResNext29 16x64d as the teacher, are shown.

These spatial mixup techniques (stitching) show improvements over interpolation mixup version because image stitching combines spatial information of two images rather than interpolating pixel values. The final sample contains two different sets of spatial information which lets the model capture information corresponding to two different spatial distributions [17].

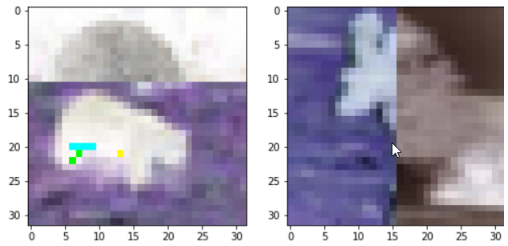

Fig. 1. Examples of horizontal and vertical stitching of two random pairs.

TABLE V
COMPARISON OF INTERPOLATION AND SPATIAL MIXUP TECHNIQUES ON CIFAR RESNET56 MODEL.

| **Refinement** | ***Top-1*** | ***Top-5*** |
|---|---|---|
| Interp. Mixup | 79.10 | 95.18 |
| Horiz. Spatial Mixup | 79.50 | 95.50 |
| Vert. Spatial Mixup | 79.50 | 95.64 |
| Interp. Mixup w/ Distill. | 80.34 | 95.75 |
| Horiz. Spatial Mixup w/ Distill. | 80.71 | **96.03** |
| Vert. Spatial Mixup w/ Distill. | **81.42** | 95.96 |

The spatial mixup, as a data augmentation technique, seemed to improve overall performance compared to the interpolation-mixup technique used in [1]. With all the training refinements stacked together, horizontal stitching improved top-5 accuracy the most while vertical stitching improved top-1 accuracy the most as shown in Table V.

## IV. CONCLUSION AND LESSONS LEARNED

### A. Conclusion

The training refinements introduced in [1] were successfully reproduced on the CIFAR100 dataset using the ResNet56 model. Ultimately, without changing the model architecture, these refinements alone led to a 7.234% and 0.094% in top-1 and top-5 scores. This work showed that these scores can be improved even more by swapping the interpolation-mixup technique with spatial mixup (stitching) techniques in the pipeline. Horizontal stitching can improve both top-1 and top-5 scores by 7.728% and 0.387% while vertical stitching improves them by 8.676% and 0.314%, respectively.

If more time allowed, additional refinements in the training steps could be made if spatial-tilling could be implemented, possibly with more than two randomly chosen samples. Spatial and interpolation mixup techniques could also be combined in order to further improve the performance increases observed in this work. Additionally, due to time-constraints, the teacher model chosen for knowledge distillation was trained only once. However, as the [16] reveals, the teacher model's accuracy and performance could be further improved without using label smoothing. It appears that label smoothing causes worse performance in the student model when conducting knowledge distillation.

### B. Lessons Learned

This work was successfully able to reproduce the training refinements in the reference literature. Additionally, the technical core of each of these refinements were studied in depth which gave further insight into the actual training-process of the CNN. Great insights and fundamental knowledge were earned through this work as much of the depth and understanding came through debugging various MXNet components, functions and objects. Little knowledge of such training refinements were known previously prior to conducting this work. The experience with MXNet and AWS was also appreciated.

Unfortunately harsh lessons were handed out while working with the ImageNet dataset. Firstly, setting up AWS for training a neural net took more time than expected and presented a non-trivial learning curve. After working through bugs in the ImageNet preprocessing and preparation pipeline, it was discovered that training on ImageNet was not feasible which is why CIFAR100 was used instead. Realizing this sooner would have saved time to explore additional methods and experiments to improve the results in this work. Also moving forward, a more capable AWS Deep Learning instance might be chosen to speed up experiments and to handle more complex models. Lastly, the AWS instance would be used on a much more stricter schedule in order to save costs.

## REFERENCES


[1] T. He., Z. Zhang, H. Zhang, Z. Zhang, J. Xie and M. Li, "Bag of Tricks for Image Classification with Convolutional Neural Networks," 2018 *arXiv*, 1812.01187

[2] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. CoRR, abs/1608.03983, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[4] P. Goyal, P. Doll´ar, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: training imagenet in 1 hour. CoRR, abs/1706.02677, 2017.

[5] X. Jia, S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, L. Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. arXiv preprint arXiv:1807.11205, 2018.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[7] Pawar, Dipti. "Improving Performance of Convolutional Neural Network!" Medium, Medium, 14 Aug. 2018, medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7.

[8] Dwivedi, Priya. "Understanding and Coding a ResNet in Keras." Medium, Towards Data Science, 27 Mar. 2019, towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33.

[9] Fung, Vincent. "An Overview of ResNet and Its Variants." Medium, Towards Data Science, 17 July 2017, towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035.

[10] Anwar, Aqeel. “Difference between AlexNet, VGGNet, ResNet and Inception.” Medium, Towards Data Science, 13 June 2020, towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96.
[11] Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. “Aggregated residual transformations for deep neural networks.” In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pp. 5987-5995. IEEE, 2017.
[12] Bushaev, Vitaly. “Adam-Latest Trends in Deep Learning Optimization.” Medium, Towards Data Science, 24 Oct. 2018, towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.
[13] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, Benjamin Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. 2017. arXiv:1705.08292v2
[14] Nitish Shirish Keskar, Richard Socher. Improving Generalization Performance by Switching from Adam to SGD. 2017 arXiv:1712.07628v1
[15] Timothy Dozat. Incorporating Nesterov momentum into Adam. 2016.
[16] Rafael Muller, Simon Kornblith, Geoffrey Hinton. When Does Label Smoothing Help. In 33rd Conference on Neural Information Processing Systems. 2019.
[17] D. Liang, F. Yang, T. Zhang and P. Yang, ”Understanding Mixup Training Methods,” in IEEE Access, vol. 6, pp. 58774-58783, 2018, doi: 10.1109/ACCESS.2018.2872698.