RESNET MODEL FOR CIFAR-10

ECE-GY-7123 Teaching Team
ECE, NYU Tandon
Spring 2022

Navneeth Krishna M. nm3932@nyu.edu
Anshika Jain aj3437@nyu.edu
Sharvari Gote ssg9784@nyu.edu

Abstract

Image Processing is a growing domain among research topics in Deep Learning with competitive innovations in Convolutional Neural Networks, Artificial Neural Networks, & related domains. Atop the underlying principle of establishing patterns in pixel representations of images, researchers have placed significant confidence in a technique called ResNets (Residual Neural Networks). We present one of such ResNet models with optimal accuracy on a standard CIFAR-10 dataset under a constraint on parameter count while presenting milestones in augmenting the approach & describing behaviours in model performance.

1 Introduction

As researchers strive for stronger computational accuracy by furthering the depth of neural networks, they do not make an effective attempt to balance the tradeoff between computational complexity versus model accuracy–as the model trains through deeper layers, it drains out millions of more computing power. Residual Neural Networks were first introduced to tackle this specific training problem in deeper neural networks as put forth by He et al. [1]. In order to not compromise on the accuracy derived from considerably increased depth, ResNets reduce such computational overhead. Another prevalent 'degradation' problem [2]-[3] raises questions about optimization as accuracy saturates while training in larger depth. Residual Neural Networks work on an 'identity mapping' paradigm–one that effectively tackles the above issue by reintroducing the identity of the shallower (erstwhile) model at effective intervals.

Fundamentally, ResNets are 'residual' models that introduce skipping connections to jump over layers to reintroduce identity. They are augmented with standard techniques to improve the overall outcome in accuracy and computational overhead. In this setting, we are expected to adhere to limitations in the count of parameters to optimize the model. Hence, we will assume that the proposed best-case outcome is under limitations of the number of Residual Layers, Residual blocks in Residual Layer i, number of channels in Residual Layer i, Convolution kernel size in Residual Layer i, Skip connection kernel size in Residual Layer i, and Average pool kernel size.

The most basic 'Convolution' consists of a Convolution operation, a ReLu unit, and a Batch Normalization operation as defined in the breakthrough research by He et al. [1]. Within multiple layers of our ResNet model, there are basic blocks that contain a standard set of 2 Convolutions and a skip connection. A skip connection is a reintroduction of an erstwhile output of the model to reinforce identity integrity in the model.

2 Methodology

In our research work, we formulate our evaluation metrics based on the parameters that we were allowed to tune. By studying related work, conducting evaluations, & adjusting augmentations, parameters, optimizers, & learning rates, profound results were obtained. We further describe the dataset in consideration, the primary factors of accuracy, the importance of each changeable parameter, and the findings that inarguably finalized our model.

2.1 Dataset: CIFAR-10

The CIFAR-10 dataset [4] consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. As standard practice, the dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

From this dataset, a model understands to predict 10 classes that are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. As we utilized PyTorch [5] for conducting our experimentation, we used the classes of DataLoader & Dataset (torch.utils.data library) for dealing with the loading, augmentation, & iteration of the dataset. For the sake of stability of the model, we use standard values for normalizing the images in order to be compatible with mini-batches of 3-channel RGB images of shape (3 x H x W). For Data Augmentation, we perform Random Horizontal Flip, a technique popular for Image Classification, to randomly flip the image on the horizontal axis. We also perform Random Crop with an input size of 32 and padding of 4.

While we confidently measured an augmented accuracy in test sets as a result of Random Horizontal Flip & Random Crop, we wanted to study the impact of applying a Grayscale augmentation on the CIFAR-10 dataset. In comparison, after repeated runs of Stochastic Gradient Descent as the Optimizer & the final model defined further as the chosen model to run, we observed a difference of 1.0-1.2% in test-accuracy with and without performing the Grayscale augmentation. Although this range is insignificant, it is herculean in the context of assessing deep learning performance. As a result, we discontinued the usage of Grayscale augmentation and relied on the Normalization, Horizontal Flip, & Random Crop as the Data Augmentation techniques for our Methodology. As the original model utilizes Batch Normalization after each ReLu operation, it can also be considered as a Data Augmentation technique.

2.2 Primary factors of accuracy

Among Residual Networks, there are confident factors that determine the accuracy of the model in assessment. Each of these factors are integral to the ideal performance of the ResNet model.

2.2.1 Optimizer Choice

The choice of optimizer is ideal in the operations performed at the Dense Neural Network phase of the model. As concluded by Choi et al. [6], inclusion relationships between optimizers are meaningful in practice & although there is no statistically significant separation in the optimizer ranking for deep learning, it is recommended to tune all of the hyperparameters and obtain the ideal optimizer. Imbibing this finding in our research, we compared Stochastic Gradient Descent (SGD) [7] and Adam [8] based on a heuristic model and observed a satisfactory consistency in test accuracies as observed in Figures 1 & 2 and Table 1 with 94.4% test accuracy using SGD and 93.17% test accuracy using Adam. Even between Models 5 & 8 as seen in Figures 3 & 4. we notice a 1.03% better performance (in magnitude) in the former optimizer.

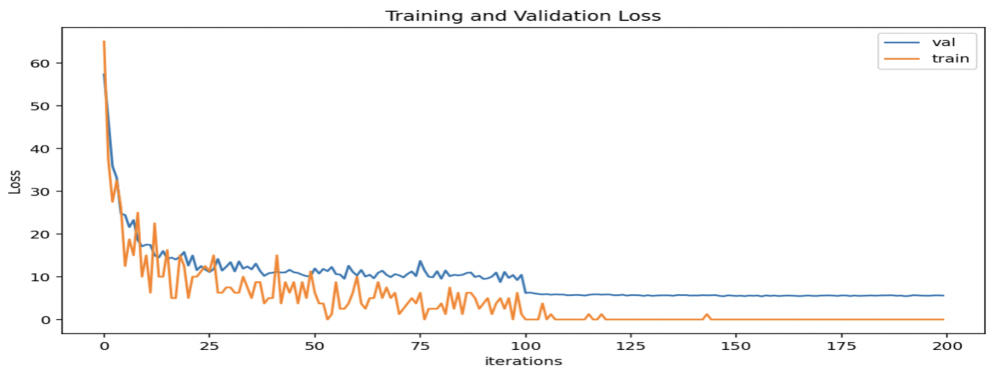


Figure 1: Training and Validation losses for Stochastic Gradient Descent on the final model architecture (Model 1 in Table 1)

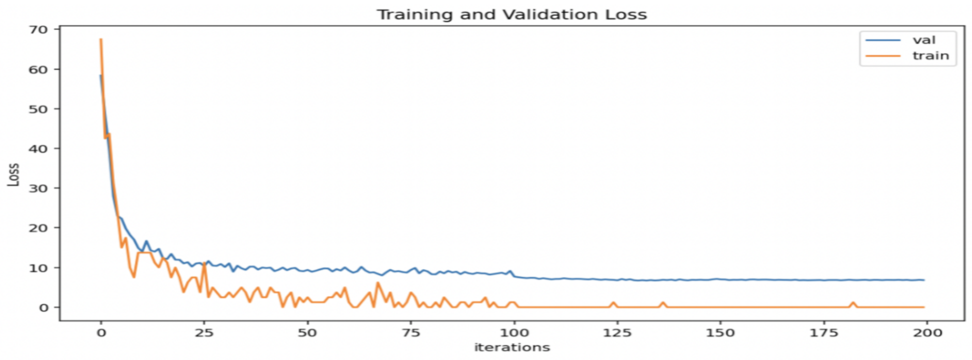


Figure 2: Training and Validation losses for Adam on the final model architecture (Model 3 in Table 1)

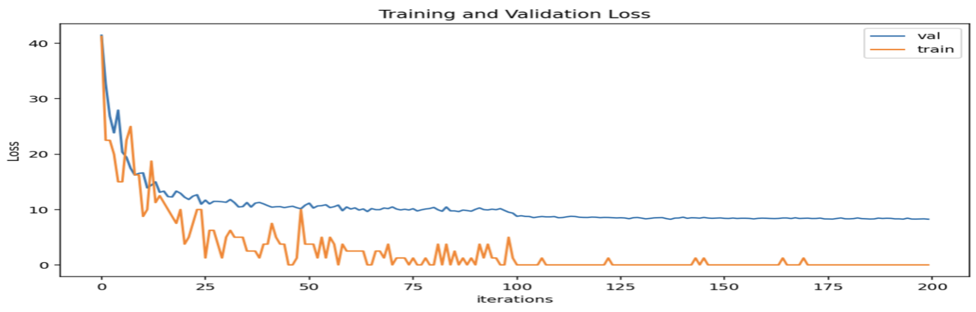


Figure 3: Training and Validation losses for Deeper Layers Model (Model 5 in Table 1)

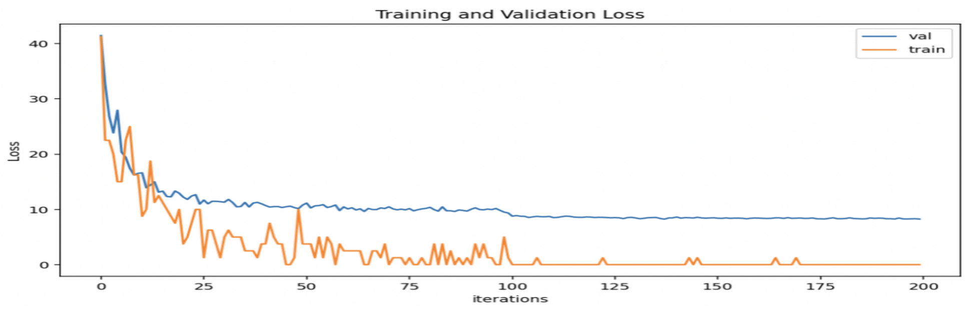


Figure 4: Training and Validation losses for Model 8 (Table 1)

Between them, SGD performed slightly better than Adam given the restrictions in our hyperparameter tuning.

2.2.2 Learning Rate

In our approach, we have experimented with values of learning rates, the dynamism of learning rates, and the correlation between the learning rate & the optimizer involved in training. Based on Wu et al. [9], the policies involved in formulating learning rates depend on multiple factors. However, the group's assimilation from ResNets on CIFAR-10, a decaying learning rate known as 'NSTEP' works better than its competitors. This technique lowers the learning rate after 'n' steps or milestones and adjusts the learning rate accordingly. Therefore, we incorporated a decaying learning rate with a Scheduler that observed the steps (epochs) and adjusted the learning rate. Through experimentation, we determined that for SGD, a powerful learning rate such as 0.1 decaying into 0.01 and 0.001 at the 100th and 150th epoch proved ideal in the learning process for this dataset. When comparing this dynamic learning approach with an approach involving a fixed learning rate, we observed a sharp contrast in the accuracies.

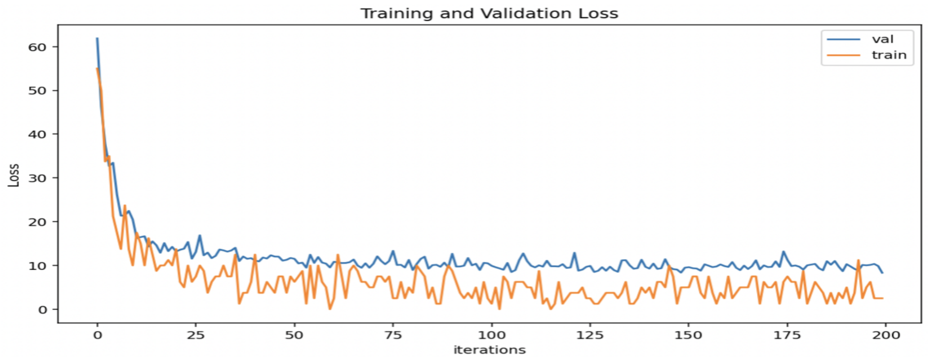


Figure 5: Training and Validation losses for Model 4 (Table 1)

Based on Table 1, we notice a 94.4% accuracy with a particular model that utilizes SGD and the proposed learning rate scheduling (Figure 1). Whereas when the same model undergoes a constant learning rate, the best performance was with a high learning rate of 0.1 at 91.67% test accuracy (Figure 5). Thus, we efficiently advocate the conclusive research from Wu et al. [9] and use the dynamic learning rate approach. Other factors such as the loss function in consideration become important but in this scenario, we default to Cross-Entropy loss.

2.3 Importance of hyperparameters

Among the hyperparameters that we are allowed to tune, we studied the effect of individual parameters on the test performance and evaluated their contributions. Cohesively, each hyperparameter played a pivotal role in optimizing the model amidst the limitations of 5M parameters and a 80% minimum test accuracy. They are as described below:

2.3.1 N, Number of Residual Layers

The number of residual layers is directly associated with the depth of the neural network architecture. Based on the work from Tan et. al [10], specifically for ResNets, depth scaling by increasing the number of layers while either keeping the other factors (width & height) constant or scaling them with a constant factor has proven to improve the test performance. Moreover, increasing the depth while keeping the width and height constant has proven to reduce the parameter size as well. In our experimentation, we arrived at an impasse in optimizing the number of layers while simultaneously trying to keep the total parameter count below 5 Million. At a stretch, one series of testing dealt with 1 basic block per layer and made the architecture as deep as 5 layers (Figure 3). The result was satisfactory at 92.78% test accuracy but did not challenge the ranking for the primary choice at less than 5M parameters. The potential for higher accuracy levels can be expected for more complex ResNet architectures.

2.3.2 B_i, Number of Residual Blocks in Residual Layer *i*

The task in optimizing the number of residual blocks in a given residual layer was challenging in the current context. Each residual block (also a basic block) operated at different strides and channel sizes in each layer. First, we experimented with a bulkier latter half model (Model 2 in Table 1) by placing 5 and 3 residual blocks in layers 3 & 4 respectively. The parameter count almost touched 5M and the 92.95% accuracy was no match to a bulkier former layer approach where we used 8 and 6 residual blocks in residual layers 1 & 2 at 94.4% test accuracy. The bulkier former layers stood better than having consistency in each layer in the architecture. Hence, under 5M, making the architecture bulkier at the earlier layers which deal with smaller channel sizes performed better on test datasets.

2.3.3 C_i, Number of Channels in Residual Layer *i*

The total parameter count was majorly affected by the choice of the number of channels in the residual layers. As the model had a fixed growth of '2x' per residual layer, the initial value C_1 proved to be highly important. For lower channel sizes (below 32), the accuracies dropped significantly while going ahead with larger channel sizes from 64 enforced smaller residual blocks and fewer residual layers. Model 6 works on similar grounds to reach a strong accuracy level of 93.87 with a larger channel size. We observe that keeping the channel size higher allows for quicker learning performance and thereby a mapping of more features. However, as tempting as its potential stays, it consumes parameters exponentially with a rise in depth and height.

2.3.4 F_i, Convolution Kernel Size in Residual Layer *i*

The value of choosing F_i performed at the Dense Neural Network phase of the model. Based on Aszemi et al. [11], for smaller CNN networks similar to our architecture, the kernel size proved insignificant in making a correlation with the accuracy and was exchanged with values between 3 and 5. In order to optimize the other parameters, we decided to go with the least count of 3.

2.3.5 k_i, Skip connection kernel size in Residual Layer *i*

The choice of optimizer is ideal in the operations performed at the Dense Neural Network phase of the model. From the studies by Li & He [12], only a dynamic skip connection methodology proves to be a strong influence on the accuracy and exclusively on deeper architectures. Thus, we believe that it is beyond the scope of this environment to optimize skip connection kernel size and default its mapping with the 3x3 kernel size.

2.3.6 P, Average pool kernel size

In our operations, we transform the model into a fully-connected layer right after performing the average pooling operation. In our testing results, we did not observe the average pool kernel size as an individual contributor as it relied upon the majority of the other hyperparameters and performed an important task in preparing the linearity of the model. Thus, we chose the default kernel size of 4.

2.4 Findings

Each of our findings on conclusive experiments have been presented with descriptions of each model. They are proving numerous hypotheses and determining the accuracies for respective configurations. The standard eight models are given below in Table 1:

MODEL NO	MODEL DESCRIPTION	TESTING	PARAMETERS

		ACCURACY (%)	(M)
Model 1	Optimal Model: 4 Layers, (8, 6, 4, 3) Block Sizes, SGD, & a Decaying Learning Rate	94.400	4.976
Model 2	Bulkier latter blocks Model: 4 Layers, (1, 1, 5, 3) Block Sizes, SGD, & a Decaying Learning Rate	92.950	4.772
Model 3	Model 1 with Adam instead of SGD	93.170	4.976
Model 4	Model 1 with the most effective Constant Learning Rate of 0.1	91.670	4.976
Model 5	Deeper layers Model: 5 layers, (1, 1, 1, 1, 1) Block Sizes, SGD, & a Decaying Learning Rate	92.780	4.905
Model 6	Larger C_i Model: 4 layers, (1, 1, 1, 1) Block Sizes, SGD, & a Decaying Learning Rate	93.870	4.903
Model 7	Smaller k_i Model: 4 layers, (7, 6, 2, 2) Block Sizes, SGD, & a Decaying Learning Rate	94.210	4.993
Model 8	Model 5 with Adam instead of SGD	91.750	4.905

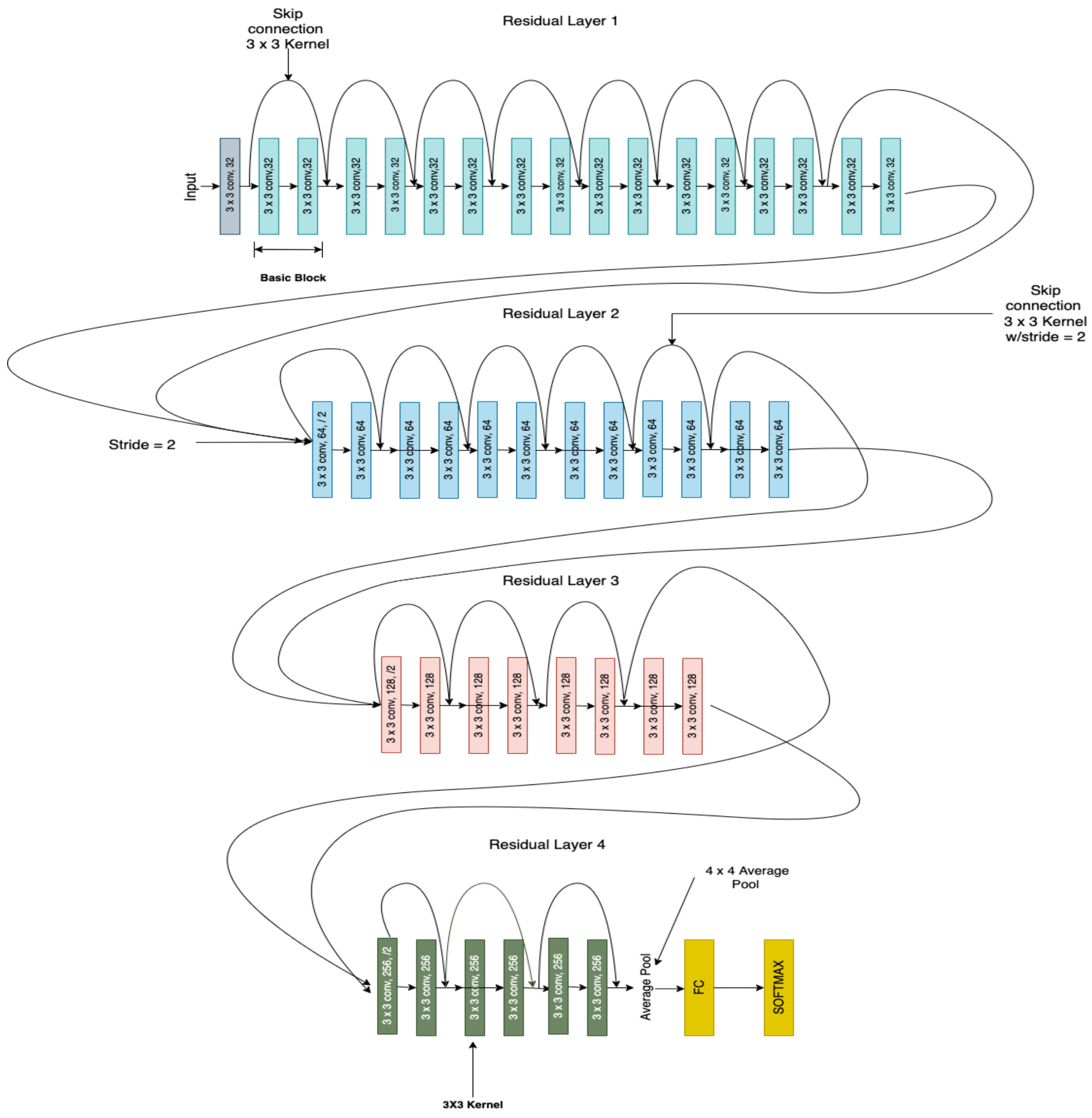
Table 1: Findings based on 8 conclusive experiments with adjustments to hyperparameters and other factors and the resulting test accuracies and parameter counts.

3 Results

On concluding with the methodology in our architecture search, we now present the detailed breakdown of our proposed architecture.

3.1 Architecture

The final ResNet architecture (Model 1) is as follows:



3.2 Model Description

After working on various architectures and models, we finalized this architecture in which we have a default set of hyperparameters with epochs as 200, batch size as 128, learning rate as 0.1, weight decay as le-4, and momentum as 0.4. Our programming convention is inspired by that of Idelbayev [14] in classes & method definitions. We perform data augmentation techniques before proceeding to the convolution. The most fundamental convolution operation assumes a subsequent batch normalization and a ReLU operation. A basic block contains two convolution operations and a skip connection (shortcut with identity mapping) of size 3x3 and a fixed stride value of 2. There are four residual layers consisting of 8, 6, 4, and 3 blocks respectively. The output from the residual layers is transformed into a fully-connected dense neural network right after an Average Pool Kernel of size 4x4. With the constant stride, the initial value of C_i evolved to 256 after the 4th layer & the choice of F_i was 3x3. As previously established, Stochastic Gradient Descent was selected as the optimizer along with Cross-Entropy loss as the criterion to optimize the weights. A decaying learning rate of 0.1 for epochs 1-100, 0.01 for epochs 101-150, and 0.001 for epochs 151-200 was chosen and the test accuracy peaked at 94.4%.

4 Conclusion

Although the limitations confined the scope of hyperparameter tuning, this research tested the endurance of ResNet architectures at minute accuracy level differences. The potential in ResNets with Image Classification is immensely growing with upcoming research. As the scope of this research aligns with modifying hyperparameters both manually and with software that optimizes related tuning, it is motivating to dwell on more ResNet-related studies.

In a narrow margin of competitive accuracies, statistical significance in accuracies cannot be the ultimate decider on model performance. Outcomes based on applying the ideal techniques, raising the right training-based questions, and answering common challenges in parameter counts prove more valuable in this sphere. As it stands, we attempt to answer such heuristics with Stochastic Gradient Descent and sustain a mindset to deal with more statistical anomalies.

Acknowledgements

We extend our sincere gratitude to Professors Garg & Mosenia, TAs, and the NYU HPC team for facilitating resources, insights, & for guiding us through this research process. To access the code in the public domain and to reproduce it, we have created a GitHub repository [13]. For cloning the repository and for other requirements, we have provided a readme file with the necessary instructions.

References

[1] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian, 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

[2] K. He and J. Sun. Convolutional neural networks at constrained time cost. In CVPR, 2015.

[3] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. arXiv:1505.00387, 2015.

[4] Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

[5] Paszke, A. et al., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035.

[6] Choi, Dami & Shallue, Christopher J. & Nado, Zachary & Lee, Jaehoon & Maddison, Chris J. & Dahl, George E., 2019. On Empirical Comparisons of Optimizers for Deep Learning. arXiv:1910.05446.

[7] Ruder, S., 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[8] Kingma, Diederik P. & Ba, Jimmy, 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

[9] Wu, Yanzhao and Liu, Ling and Bae, Juhyun and Chow, Ka-Ho and Iyengar, Arun and Pu, Calton and Wei, Wenqi and Yu, Lei and Zhang, Qi, 2019. arXiv:1908.06477.

[10] Tan, Mingxing & Le, Quoc V., 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946.

[11] Mohd Aszemi, Nurshazlyn & Panneer Selvam, Dhanapal Durai Dominic. (2019). Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms. International Journal of Advanced Computer Science and Applications. 10. 269 - 278. 10.14569/IJACSA.2019.0100638.

[12] B. Li & Y. He, "An Improved ResNet Based on the Adjustable Shortcut Connections," in IEEE Access, vol. 6, pp. 18967-18974, 2018, doi: 10.1109/ACCESS.2018.2814605.

[13] K. M. Navneeth & Jain, Anshika & Gote, Sharvari. ResNet for CIFAR-10 with a restriction on parameter count (2022), GitHub repository, https://github.com/navneeeth/resnet-for-cifar-10-dl-project-1

[14] Yerlan Idelbayev. Proper ResNet Implementation for CIFAR10/CIFAR100 in Pytorch (2021), GitHub repository, https://github.com/akamaster/pytorch_resnet_cifar10