ResNet18 Parameter Reduction and Training

Rakshana BS[1] (rb5118), Abirami S[1] (as16288), Nagharjun M[1] (nm4074)

[1]New York University - Tandon School of Engineering
rb5118@nyu.edu, as16288@nyu.edu, nm4074@nyu.edu
https://github.com/Nagharjun17/ResNet18MiniProjectDL

Abstract

In this project, we explore the impact of tuning hyperparameters, using data augmentations, and changing optimizers on ResNet-18 training for achieving a 90% test accuracy on the CIFAR-10 dataset. The ResNet-18 has a reduced number of parameters by reducing block size. We experimented with different combinations of hyperparameters and data augmentations, along with various optimizers, such as SGD and Adam, to find the optimal configuration that yields the highest accuracy. Our results demonstrate that tuning hyperparameters and using data augmentations significantly improve the model's accuracy. Additionally, changing optimizers can also have a significant impact on the performance of ResNet-18 on CIFAR-10. We provide insights into best practices for ResNet-18 training on CIFAR-10 and achieve our goal of 90% test accuracy.

Overview

Residual networks, or ResNets, are a type of deep neural network architecture that utilizes shortcut connections to address the vanishing gradient problem in deep neural networks. When a neural network has many layers, the gradients during backpropagation can become very small, resulting in slower learning or even degradation in performance. The ResNet architecture aims to mitigate this issue by introducing residual blocks that allow the network to learn residual functions instead of mapping directly from input to output.

ResNets [1] are characterized by their use of skip connections, which are also known as identity mappings or shortcut connections[3][4]. These connections bypass one or more layers of the network, allowing the activations of earlier layers to be reused in later layers. By doing so, these connections help to preserve the gradient signal during backpropagation, which enables the network to learn more effectively.

The ResNet architecture has several variations, each with a different number of layers. The number of layers directly affects the depth of the network, with deeper networks typically resulting in better performance, albeit at the cost of increased computation and memory requirements. Common ResNet variants include ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152.

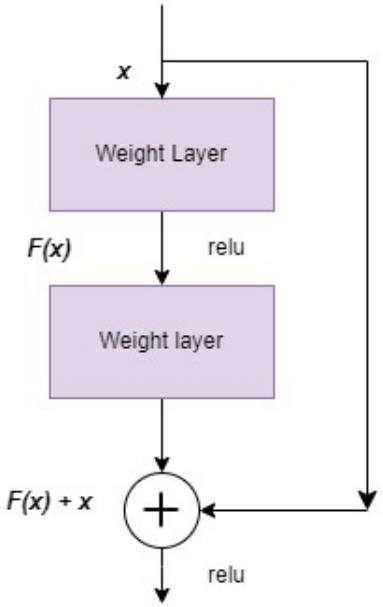

Figure 1: Residual Block

In a ResNet, a residual block typically consists of two or more convolutional layers and a skip connection that bypasses one or more of the convolutional layers. The residual block output is then added to the input of the block, creating the residual connection(Figure 1). This approach allows the network to learn residual functions that are easier to optimize than the original mapping and helps to prevent the vanishing gradient problem.

To reduce loss and increase the accuracy of our findings, we employed a variety of approaches, including data augmentation and optimizers. After multiple efforts, we discovered that combining the above strategies with a reduction in the number of blocks and the implementation of a learning rate scheduler produced the desired outcomes.

NYU Tandon School of Engineering

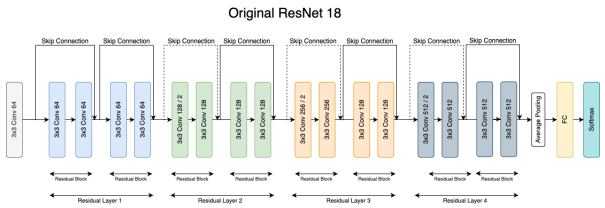

Figure 2: Original ResNet18 Architecture

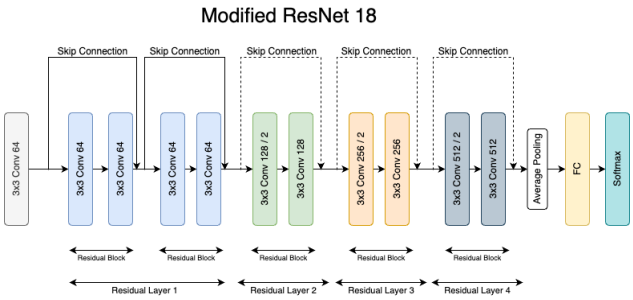

Figure 3: Modified ResNet18 Architecture

Methodology

This section discusses various techniques utilized to achieve the target accuracy metric.

Modified Model Architecture

Figure 1 represents the original resnet18 architecture which contains 4 residual layers with 2 residual blocks in each layer. The overall model has 11,173,962(11.2M) parameters in total.

Figure 2 represents the modified resnet18 architecture which contains 4 residual layers with 2 residual blocks in the first layer and 1 residual block in the other 3 layers. The blocks have been removed to reduce the number of parameters of the model. The overall model has 4,977,226(4.9M) parameters in total.

Dataset

For our experiments, we use the CIFAR-10 dataset [2] which is comprised of 60000 color images that are 32x32 pixels in size, divided into 10 distinct classes, with each class containing 6000 images. The dataset is partitioned into 50000 training images and 10000 test images. The training images are distributed across five batches, with each batch containing 10000 randomly ordered images. The test batch is also composed of 1000 randomly selected images from each class. Although the training batches are arranged randomly, some of them may contain more images from one class than another. However, collectively, the training batches contain precisely 5000 images from each class.

Attempts and Findings

- To bring the number of parameters under 5 million, block size has been reduced in the last 3 blocks.
- The number of parameters has been brought from 11,173,962 to 4,977,226.
- Tuning other parameters such as batch size does not have any effect on the number of parameters.
- The number of parameters depends on channel size, layer size, kernel size and block size.
- A range of optimizers were used with different learning rates.
- It was found that the Adam optimizer performs the best.
- The Adam optimizer was taken as the base optimizer and hyperparameter tuning was performed.
- Cosine Annealing Learning Rate scheduler was used for better convergence.
- A major finding is the tuning of learning rate: A higher learning rate does not let the model converge and a smaller learning rate slowly converges and often misses out on the global minima. So an optimal learning rate of 0.001 was found after experimentation.
- After training the model using different parameters, the best is a batch size of 64, a learning rate of 0.001, with Adam optimizer decay rate of $5e^{-5}$ and Cosine Annealing Scheduler with scheduling every 200 steps. The best testing accuracy attained is 93%.

Augmentations

Data augmentation is a process of artificially increasing the amount of data for machine learning models by generating new data points from existing data. This process involves manipulating datasets with techniques such as flipping, rotating, or adding noise to create new and diverse instances for training datasets.

Data Augmentation techniques used in this project are:

- Random Crop: When applied to an image, a random crop is taken from a randomly selected position in the image. The parameters passed are the output size and padding = 4 which means the image is padded after cropping on all 4 sides.
- Random Horizontal Flip: It randomly flips the input image horizontally with a probability of 0.5, which means that the left side of the image becomes the right side, and vice versa. This technique is useful because it can help the model learn to recognize objects in images from different angles.

Optimizers

The optimizers used while experimenting are:

Stochastic Gradient Descent (SGD): It is a basic optimization algorithm that updates the model parameters in the opposite direction of the gradient of the loss function with respect to the parameters.

RMSProp: It divides the learning rate by a running average of the root mean squared (RMS) of recent gradients, which helps to reduce the impact of noisy gradients and speed up convergence.

AdaGrad: It is an adaptive learning rate optimization algorithm that adapts the learning rate of each parameter based on historical gradient information. It performs smaller updates for frequently occurring features and larger updates for infrequent features.

RMSProp: It is another adaptive learning rate optimization algorithm that uses a moving average of the squared gradients to normalize the gradient updates. It performs well in non-convex optimization problems.

AdaDelta: It is a variant of AdaGrad that solves its drawback of monotonic learning rates by adapting the learning rate based on the moving average of the gradients and updates.

Adam: It is a combination of RMSProp and SGD with Momentum that uses moving averages of both the first and second moments of the gradient to adapt the learning rates of each parameter.

Schedulers

We have used the Cosine Annealing Learning Rate Scheduler which is used for adjusting the learning rate of a neural network during training. It is based on the idea of annealing, where the learning rate is gradually decreased over time to help the model converge to a good solution. The Cosine Annealing Scheduler works by following a cosine curve that gradually decreases the learning rate from an initial value to a minimum value, and then back up to the initial value again. This pattern is repeated over multiple epochs or iterations of the training process.

Results

This section presents the results of experimentation by tuning hyperparameters and optimizers.

Optimizer	Test Accuracy
SGD	87
RMSProp	89.5
AdaDelta	77.7
AdaGrad	86.5
Adam	93

Table 1: Testing accuracy on different optimizers with a fixed batch size of 64 and 200 epochs.

In Table 1, we have experimented with the modified resnet18 architecture on 5 different optimizers: Stochastic Gradient Descent, RMSProp, AdaDelta, AdaGrad, and Adam. All the variations were trained on a fixed batch size of 64, a fixed number of 200 epochs, a fixed learning rate of 0.001, and Cosine Annealing learning rate scheduler.

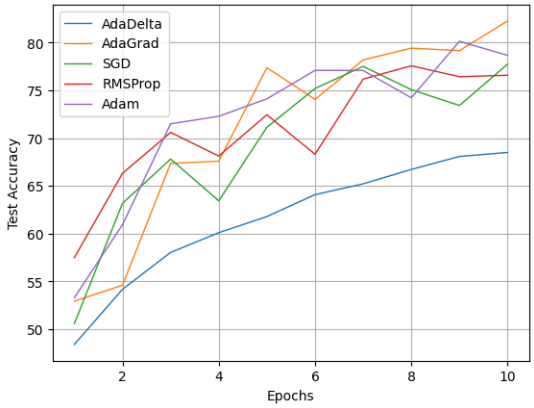

Figure 4: Performance of Optimizers in the first 10 epochs

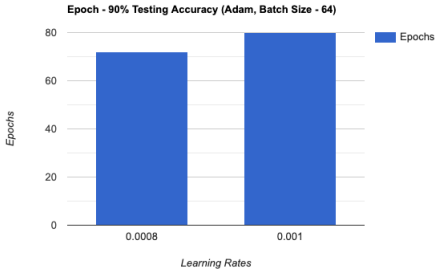

Figure 5: Epochs vs 90% Accuracy

From Table 1, it can be clearly seen that Adam optimizer performs better than other optimizers. Now, we will experiment with Adam using different learning rates.

Learning Rate (Adam)	Test Accuracy
0.0001	68.6
0.0008	90.7
0.001	93
0.0012	84.6
0.01	70.7

Table 2: Testing accuracy on Adam optimizers with varying learning rates

In Table 2, after experimenting with varying learning rates, we found that the model performs the best on a learning rate of 0.001. We can observe that models with higher learning rates does not converge well. Also, models with very low learning rates converges very slow and it will not find the global minima. So, an optimal learning rate is required.

Figure 5 represents the number of epochs taken by models with different learning rates to achieve a test accuracy of 90%

References

[1] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385.

[2] Krizhevsky, A.; Nair, V.; and Hinton, G. 2015. CIFAR-10 (Canadian Institute for Advanced Research).

[3] Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge University Press.

[4] Venables, W. N.; and Ripley, B. D. 2002. *Modern Applied Statistics with S*. New York: Springer, fourth edition. ISBN 0-387-95457-0.