Deep Learning Mini Project Report
Running Modified ResNet Model on CIFAR-10 Dataset

Sheena Garg (sg7394), Shruti Garg (sg7395), Tiyas Dey (td2355)

NYU Tandon School of Engineering

Code Link: https://github.com/shruti-garg30/ResNet-on-CIFAR-10/blob/main/main.ipynb

Abstract

This report presents a project aimed at designing a modified residual network (ResNet) architecture for image classification on the CIFAR10 dataset with a constraint of no more than 5 million parameters. The project investigates various modifications to the ResNet architecture, including changes to the number of layers, kernel sizes, and channel sizes. The models are trained using ADAM optimizer and cross-entropy loss. The performance of the modified architectures is evaluated based on the test accuracy achieved on the CIFAR10 dataset. The report presents the results of the experiments and analyzes the impact of different modifications on the accuracy and the number of parameters. The report concludes by presenting the optimal modified ResNet architecture with the highest test accuracy on CIFAR10 within the parameter constraint.

Methodology

Background

ResNet, short for Residual Neural Network, is a type of deep neural network architecture that has been widely used in computer vision tasks such as image classification, object detection, and segmentation. ResNet introduces a new type of connection between the layers, called residual connections, which allow the network to learn residual mappings instead of the actual mappings. This makes it possible to train very deep networks, up to hundreds of layers, without facing the vanishing or exploding gradient problem. The residual connections also help in improving the accuracy of the network by allowing the network to learn the identity mapping and minimizing the degradation problem that can occur when adding more layers to a network.

The CIFAR10 dataset is a popular image classification dataset consisting of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The dataset is split into a training set of 50,000 images and a test set of 10,000 images. The classes include common objects such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The CIFAR10 dataset is widely used in the computer vision community as a benchmark dataset for evaluating image classification algorithms and deep learning models due to its relatively small size and the diversity of the objects it contains. [Refer Figure 1.]

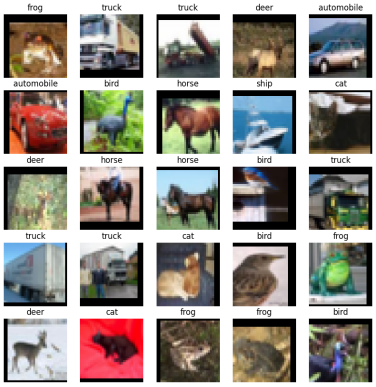

Figure 1: Examples from the CIFAR10 dataset

Experimentation

Our main objective was to strike a balance between the model's performance and the constraint imposed by the number of trainable parameters. To achieve this, we conducted a series of experiments involving various aspects of the network architecture such as the number of layers, channel sizes, filter size, optimizer function, learning rate, and the number of epochs.

Data Augmentation: Since the CIFAR-10 dataset is very small, we employed data augmentation techniques by incorporating various transformations into our dataset. These transformations, such as Random Horizontal Flip, Random Crop, and Normalization, were applied to augment the training data, aiming to improve the model's performance on unseen data.

Loss and Optimizer: Given that the task at hand involved classification, we utilized cross-entropy loss as the evaluation metric for our model. To enhance the model's performance in terms of both convergence speed and generalization capacity, we opted for the ADAM optimizer with weight decay instead of regular Stochastic Gradient Descent.

Regularization: In order to enhance feature extraction in our residual layers, we experimented with increasing the

number of channels. However, we also recognized that this led to an increase in the number of trainable parameters, which could result in overfitting. Hence, we carefully balanced the number of channels and layers to achieve optimal performance within the given constraints. Given the relatively small size of the CIFAR10 dataset, we observed that increasing the number of convolutional layers beyond 60 resulted in overfitting after approximately 22 epochs. To handle overfitting and in order to aim at achieving higher performance, we introduced regularisation and a learning rate scheduler to the model. We used Dropout and the "One Cycle Learning Rate Policy" respectively. The One Cycle policy helps to reduce overfitting by gradually decreasing the learning rate towards the end of training. This allows the model to generalize better to new data. It is also relatively robust to the choice of initialization for the model's weights. Dropout helps to reduce overfitting by randomly dropping out (i.e., setting to zero) some of the neurons in the neural network during training. This prevents the neurons from co-adapting and forces the network to learn more robust features. The modified model took a much longer time to converge. In our research study, the model was trained for 100 epochs. During this training process, the model achieved a validation accuracy of 92.24%. Importantly, the model did not show signs of overfitting to the training data, and its performance consistently improved over the subsequent epochs. After completing 100 epochs, the model's test accuracy was measured at 92.62%.

Hyperparameter Tuning: To strike a balance between capturing fine details and adhering to parameter constraints, we opted for a 3x3 filter in each of our convolutional layers. Additionally, considering the small channel size of our model, we employed an average pool layer with a pool size of 2. This choice was made to avoid excessive reduction of spatial features, which could result in the loss of important information compared to larger pool sizes. The number of layers was selected by a trade-off method between feature extraction, channels in each layer and the number of trainable parameters.

Results

The final implemented model can be found here: https://github.com/shruti-garg30/ResNet-on-CIFAR-10/blob/main/main.ipynb.

Our code is based on the code used in (**?**) (**?**). Our model has 5 layers of 4 residual blocks each. Alternating residual blocks are connected through skip connections, as presented in the (**?**) paper. Thus the total number of convolutional layers in our network is 42 including the initial layer and the final connected layer. The final hyperparameters used in our model are:

- $C_1 = 32$, $C_2 = 64$, $C_3 = 96$, $C_4 = 128$ and $C_5 = 185$ where C_i represents the number of channels in the ith layer,
- $F_i = 3x3$ where F_i is filter size in the ith layer and is the same throughout our model
- $P = 2$, where P_i represents the pool size in the average pool layer

We use ADAM with weight decay of $1e-4$ as the optimizer and measure cross-entropy loss to evaluate the performance of the model. We also use the One Cycle Learning Rate Policy as the learning rate scheduler with a maximum learning rate of $1e-2$.

The total number of trainable parameters in our model is 4,907,445, which adheres to the constraint imposed (¡5,000,000). We ran our model for 150 epochs and reached an accuracy of **93.08**% on the test dataset. Fig 2 shows the graph for variation of training loss with epochs and Fig 3 shows the variation of validation accuracy with each epoch. Fig 4 shows the variation of learning rate with each epoch. The model architecture is displayed in Fig 5. Finally, Fig 6 shows the confusion matrix based on the test results.

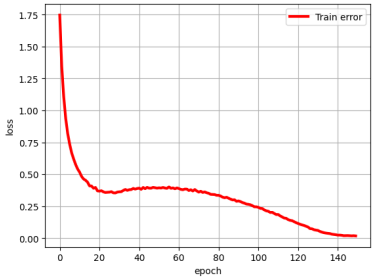

Figure 2: Train Loss vs Epochs

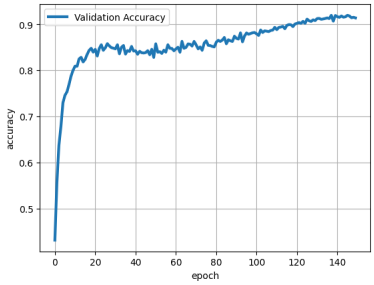

Figure 3: Validation Accuracy vs Epochs

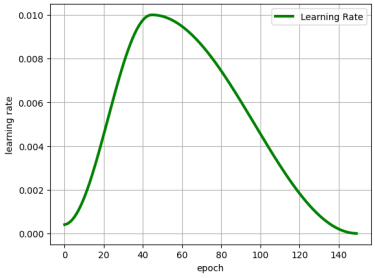

Figure 4: Learning Rate vs Epochs

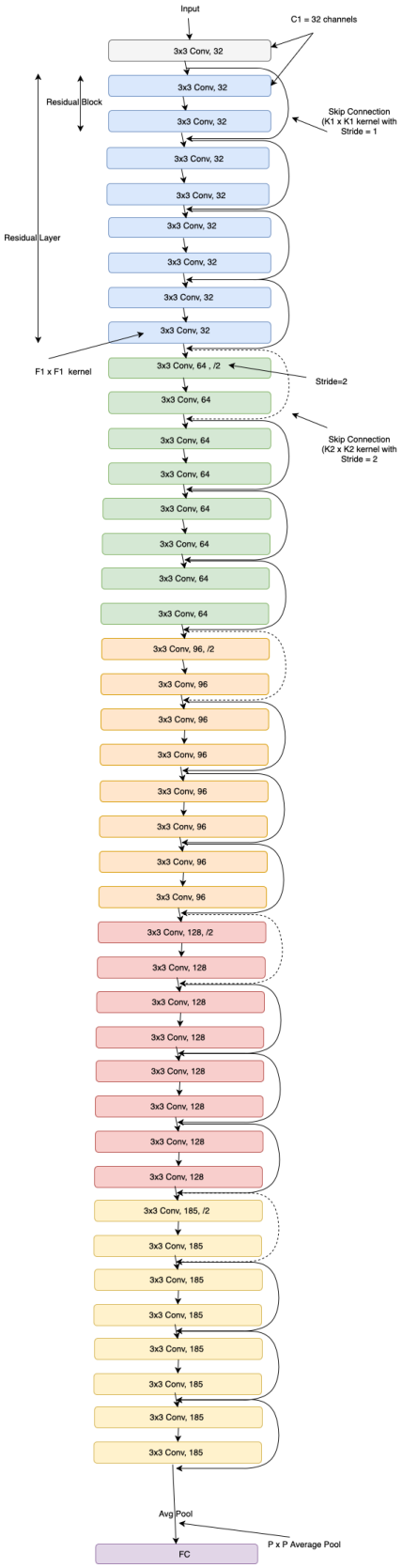

Figure 5: Model Architecture

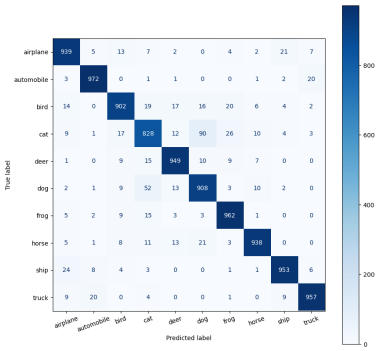

Figure 6: Confusion Matrix on Test Data

References

1. He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. 770–778.
2. https://pytorch-tutorial.readthedocs.io/en/latest/tutorial/chapter03_intermediate/3_2_2_cnn_resnet_cifar10
3. https://jovian.com/aakashns/05b-cifar10-resnet