Residual Network Design
Mini-Project Report

Mehul Sudrik, Rajesh Nagula, Utsav Oza

ECE-GY 7123 Deep Learning

Abstract

In convolution neural networks, ResNet-18 architecture has been widely used in deep learning for image classification tasks. In this project, we propose a modified residual network architecture that reduces the number of trainable parameters to less than 5 million while maintaining high accuracy on CIFAR-10 image classification dataset. Our approach involves applying basic residual blocks with a reduced number of channels and residual blocks to further reduce the number of parameters. Our results demonstrate the effectiveness of our approach, with the modified architecture achieving competitive performance compared to the original ResNet-18 architecture while requiring significantly fewer trainable parameters.

Introduction

A residual network (ResNet) is a deep neural network architecture that was designed to address the problem of vanishing gradients in very deep neural networks, which can cause the accuracy of the network to decrease as the depth of the network increases.

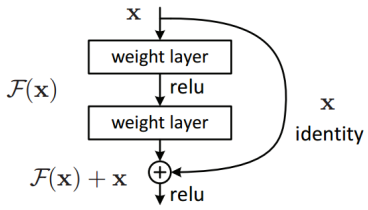

Figure 1: Residual Block (He et al. 2015)

The core idea behind ResNet is the use of residual blocks, which are a type of building block for the neural network. A residual block consists of two or more convolutional layers followed by a shortcut connection that skips one or more layers. The output of the convolutional layers is added to the input of the shortcut connection, which allows the network to learn the residual mapping between the input and output of the block. This residual mapping is then added to the output of the block, allowing the network to learn more complex representations without sacrificing accuracy.

ResNet architectures typically consist of multiple residual blocks stacked on top of each other, with increasing numbers of filters and decreasing spatial dimensions as the depth of the network increases. The final layer of the network is typically a global average pooling layer followed by a fully connected layer for classification or regression.

In this project, we were able to design and experiment with modified residual network architectures with the primary constraint that the total number of trainable parameters of the network do not exceed 5 million, while also maintaining high accuracy on CIFAR-10 image classification dataset. We achieved the best test accuracy of 94.12% with ResNet-22 architecture with total number of trainable parameters equal to 4,922,826. This model performance was achieved using the cross-entropy loss function, with the learning rate of 0.1 and Adadelta optimizer. Code accompanying our findings is hosted on this GitHub repository: https://github.com/utsavoza/aperture.

Methodology

A residual block is a key building block in designing a residual network. The skip connection in the residual block allows the input to flow through the block unchanged and is added to the output of the block. By doing this, the block can learn to model the residual or the difference between the input and the output. The most common form of the residual block in ResNet is the "identity" block, which consists of two or three convolutional layers with batch normalization and ReLU activation, followed by a skip connection that adds the input to the output. A deep ResNet architecture is designed by stacking such residual blocks together. Hyperparameters in such residual networks include:

- N, the number of residual layers.
- B_i, the number of residual blocks in i^{th} residual layer.
- C_i, the number of channels in i^{th} residual layer.
- F_i, convolution kernel size in i^{th} residual layer.
- K_i, skip connection kernel size in i^{th} residual layer.
- P, average pool kernel size

Our general approach for the project has been to first figure out the best network architecture by trying a fixed set

Model	N	B	C	P	Parameters	Accuracy
ResNet-10	4	1, 1, 1, 1	64, 128, 256, 512	4	4,903,242	81.610
ResNet-18	4	2, 2, 2, 2	64, 157, 211, 270	4	4,996,254	83.350
ResNet-20	4	2, 2, 2, 3	64, 128, 185, 245	4	4,998,382	83.630
ResNet-22	4	3, 2, 3, 2	64, 128, 192, 256	4	4,922,826	84.530
ResNet-24	4	3, 3, 2, 3	64, 128, 128, 256	4	4,935,242	83.950
ResNet-32	4	4, 4, 4, 3	32, 64, 128, 256	4	4,754,218	82.800

Table 1: Test accuracies (%) on different network architectures

of configurations on different network architectures. The architecture with the best test accuracy is then chosen to further improve the performance by trying out different learning rates, optimizers, batch size, epochs, and data augmentation strategies.

Network Architecture

We started with a baseline ResNet-18 model and standard hyperparameters, which has around 11 million trainable parameters. To reduce the number of trainable parameters, we tried reducing the number of residual layers (N). However, for any configuration of residual blocks (B), architecture with 3 residual layers always performed worse than using 4 residual layers, which also confirms the findings presented in this paper (Naranjo-Alcazar et al. 2019).

With 4 residual layers, we then experimented with different configurations of residual blocks (B) and the number of channels (C). Different network architectures were tested on the following fixed set of configuration:

- **Batch Size** = 32
- **Number of Epochs** = 20
- **Learning Rate** = 0.1.
- **Optimizer** = Adadelta
- **Data Augmentation** = Standard

Table 1 summarizes, the test accuracies achieved across different network architectures using the above configuration. Based on the test accuracies received on different residual network architectures using fixed learning parameters, we were able to conclude that **ResNet-22** architecture performed better than other network architectures.

Hyperparameters

In our study, we conducted several experiments to determine the optimal combination of optimizer and learning rate for our ResNet-22 architecture.

Optimizer: First, we aimed to find the most suitable optimizer, keeping the batch size fixed at 32 and using a learning rate of 0.1 without using any learning rate scheduler. To achieve this, we experimented with several optimizers, namely **SGD**, **SGD with Nesterov**, **Adam**, **Adagrad**, and **Adadelta** for 20 epochs. Results in Figure 3 shows that the Adadelta optimizer converged the fastest as compared to other optimizers. Adadelta is an adaptive learning rate

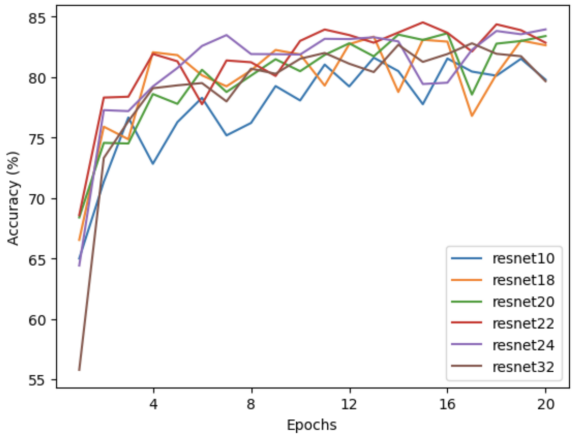

Figure 2: Training on different **ResNet** architectures

method that dynamically adjusts the learning rate based on the past gradient updates. It is a variant of the Adagrad optimizer that addresses the problem of learning rate becoming too small during the training process.

Learning Rate: Next, we conducted several experiments to determine the optimal learning rate in combination with our optimizer - we primarily tried three learning rates, **0.1** **0.01** and **0.001** along with two schedulers, 1 cycle learning rate policy and cosine annealing. We trained our model for 20 epochs with a batch size of 32, without any data augmentation. To evaluate the performance of each combination, we used both the test accuracy and convergence rate as our goal metrics. Based on our results, we observed that a constant learning rate of **0.1** with **Adadelta** converged fastest and outperformed other optimizers in terms of accuracy achieved in 20 epochs.

Batch Size: To determine the optimal batch size, we kept the other hyper parameters fixed. Our goal metrics for ideal batch size were faster convergence rate with highest accuracy, in fixed number of epochs. Table 2 summarizes the results obtained after training our model for 20 epochs.

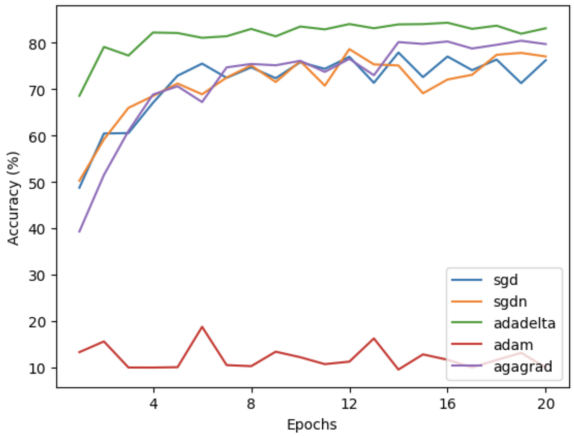

Figure 3: Performance of Optimizers

Batch Size	Accuracy	Time in secs (per epoch)
8	85.30	77.76
16	84.45	41.30
32	84.31	28.58
64	84.45	24.42
128	82.44	23.21

Table 2: Effect of Batch Sizes

Despite batch size 8 giving the best performance, we chose **64 as our ideal batch size** as the performance improvement with batch size 8 is not significant enough to justify more than 3x increase in training time taken per epoch.

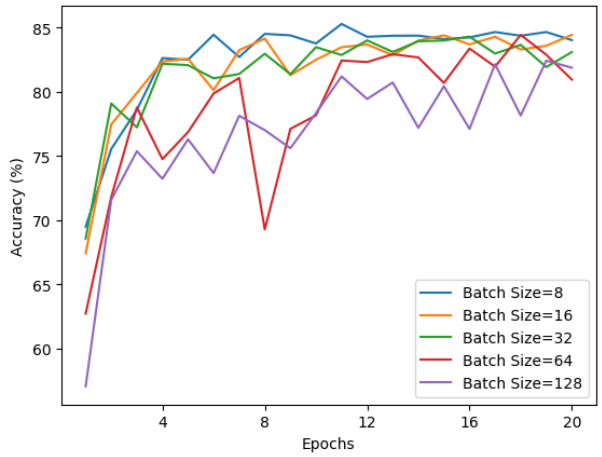

Figure 4: Batch Size vs. Test Accuracy

Data Augmentation: Finally, we experimented with two data augmentation strategies: Standard and Mixup (Zhang et al. 2017).

Standard Augmentation is simply applying random crop and random horizontal flipping on samples in the training dataset. In contrast, Mixup applies convex combinations of pairs of examples and their labels. By doing so, mixup regularizes the neural network to favor simple linear behavior in-between training examples. When appling standard data augmentation, we observed that test accuracy more or less plateaued around 91% after 100 epochs, whereas with mixup, we were able to observe that our test accuracy improved from approximately 91% to 93% when trained over 100 epochs, despite training accuracy plateauing at around 60%. Our experiments show that mixup improves the generalization of our ResNet-22 architecture.

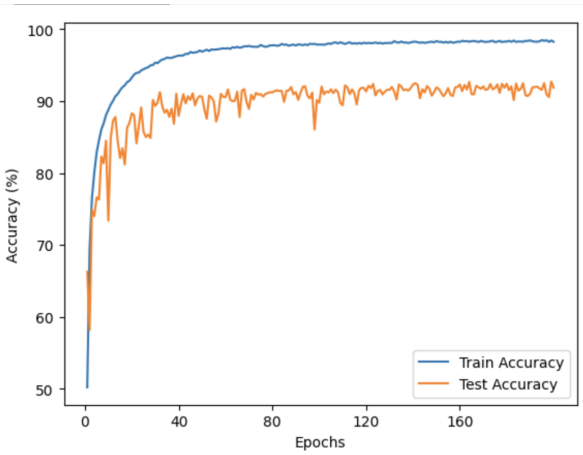

Figure 5: Standard Data Augmentation

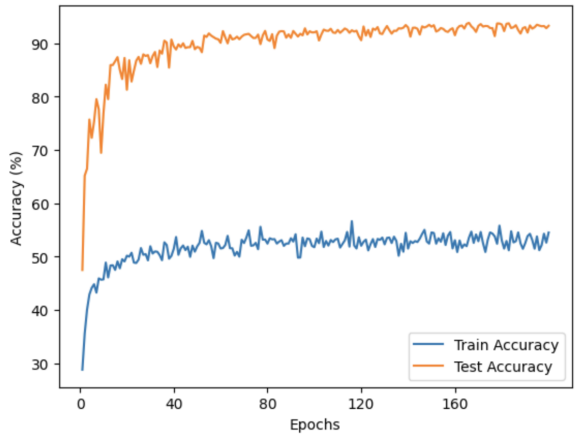

Figure 6: Mixup Data Augmentation

Results

In this project, we were able to achieve a **test accuracy of 94.12**% on CIFAR-10 image classification dataset. We experimented with different residual architectures, and applied hyperparameter tuning techniques on the best model architecture, i.e. **ResNet-22** to figure out the best combination of

hyperparameters to achieve the highest possible test accuracy on the dataset.

Our ResNet-22 model architecture, contains a total of **4,922,826 trainable parameters**. Our residual network contains **4 residual layers** with **3, 2, 3 and 2** residual blocks in each layers respectively, and the number of channels in each residual layers are **64, 128, 192 and 256** respectively. Convolution kernel sizes in all the residual layers are 3 and the skip connection kernel size in reach residual layers is 1. ResNet-22 achieves best performance when trained over 300 epochs using **0.1 learning rate with Adadelta** optimizer and **batch size equal to 64**. To further improve the test accuracy of our model, we applied Mixup data augmentation strategy. Table 3 summarizes the different hyperparameter values used to achieve the highest test accuracy of 94.12% on our ResNet-22 model architecture.

Configuration	**Values**
N	4
B	3, 2, 3, 2
C	64, 128, 192, 256
F	3, 3, 3, 3
K	1, 1, 1, 1
P	4
Learning Rate	0.1
Optimizer	Adadelta
Batch Size	64
Data Aug. Strategy	Mixup
# Parameters	4,922,826

Table 3: ResNet-22 with optimal hyperparameters

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Naranjo-Alcazar, J.; Perez-Castanos, S.; Martin-Morato, I.; Zuccarello, P.; and Cobos, M. 2019. On the performance of residual block design alternatives in convolutional neural networks for end-to-end audio classification. arXiv:1906.10891.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond Empirical Risk Minimization. arXiv:1710.09412.