# Deep learning mini project

**Ankur Aggarwal (aa10336), Anupam Tiwari (ast9885)**

## Abstract

Residual Networks (ResNets) are widely employed in image classification tasks. In this project, we developed and trained a customized ResNet model for CIFAR-10 image classification, focusing on maximizing test accuracy while keeping the model size within a fixed budget of 5 million trainable parameters. Model size, often measured by the number of trainable parameters, is crucial when deploying models on devices with limited storage capacity, such as IoT or edge devices.

In this report, we introduce our innovative residual network design, which contains less than 5 million parameters. Our ResNet model achieves a remarkable test accuracy of 85.39% on the CIFAR-10 benchmark when combined with appropriate training strategies and hyperparameters. The models and code can be accessed at https://github.com/anupam-tiwari/dl-mini-project

## Overview of the project

ResNet is a highly regarded deep neural network widely used in computer vision tasks. The prevailing approach has been to create deeper and more complex networks to enhance accuracy. However, these improvements do not necessarily lead to more efficient networks in terms of size and speed. In various real-world applications, such as embedded systems, robotics, self-driving cars, and augmented reality, recognition tasks must be executed promptly on computationally constrained platforms. As a result, there is a growing demand for memory-efficient models that deliver competitive performance. Related works, such as MobileNet and WideNet, have sought to address this issue.

In this article, we discuss our efforts to devise a memory-efficient configuration for the ResNet-18 model. We present an overview of our approach, the parameters we adjusted, and the optimization methods we employed to ultimately achieve an optimal configuration. This configuration boasts less than 50 percent trainable parameters

## 2. Methodology: ResNet Hyperparameters

### 2.1 Residual Layers (N) and Residual blocks in Residual Layer i (Bi)

To adhere to the criterion of fewer than 5 million trainable parameters, we limited the maximum number of residual layers to 4. We experimented with various configurations, including different numbers of residual layers and blocks. By adjusting the number of channels and residual blocks in each module, we can create diverse ResNet models.

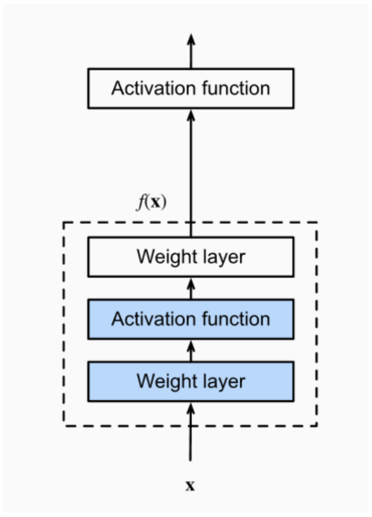


**Figure 1: ResNet block**

### 2.2 Number of Channels in Residual Layer i (Ci)

Let us observe the changes in the input shape on changing the number of channels in ResNet. The resolution decreases and the number of channels increase up until the point where a global average pooling layer aggregates all features.

| | |
|---|---|
| Sequential output shape | torch.Size([1, n, 32, 32]) |
| Sequential output shape | torch.Size([1, 2*n, 16, 16]) |
| Sequential output shape | torch.Size([1, 4*n, 8, 8]) |
| Sequential output shape | torch.Size([1, 8*n, 4, 4]) |
| AvgPool2d output shape | torch.Size([1, 8*n, 1, 1]) |

| | |
|---|---|
| Flatten output shape | torch.Size([1, 512]) |
| Linear output shape | torch.Size([1, 10]) |

Figure 2: Varying the number of channels in residual layers

### 2.3 Convolutional Kernel Sizes in Residual Layer i (Fi)

Convolutional neural networks extract low-level features from local viewpoints in images. The number of parameters increases quadratically with kernel size, making large convolution kernels inefficient, especially with numerous channels. We experimented with kernel sizes and found that a convolution kernel greater than 3 only satisfies the 5M parameters constraint for 3 or fewer layers.

### 2.4 Ki: Skip Connection Kernel Size in Residual Layer i

We tested varying skip connection kernel sizes for a 3-layer network. Within the 5M parameter constraint, we could experiment with a kernel size as large as 9, which yielded the best accuracy.

### 2.5 Average Pool Kernel Size (P)

Average Pooling calculates the average value for feature map patches, creating a downsampled feature map with smoother extraction than Max Pooling. The kernel size depends on the number of ResNet architecture layers, given fixed strides, unchanged image resolutions, and the 5M parameter limit. The formula to determine the value for the average pool kernel (P) is: 32 P = 2N−1, where N represents the number of residual layers.

## 3. Methodology: Training Strategies

### 3.1 Data Preparation Strategies

We made key changes while creating PyTorch datasets, such as using the test set for validation, channel-wise data normalization, and randomized data augmentations. These strategies improve training performance and help the model generalize better.

### 3.2 Optimizers, Learning Rates (LR), and LR Schedulers

We experimented with Stochastic Gradient Descent (SGD) and Adam optimizers on CIFAR-10, finding SGD performed better. We analyzed the impact of varying learning rates, momentum, and batch sizes, identifying optimal combinations. We tested 12 learning rate schedulers provided in PyTorch, with CosineAnnealingLR giving the best performance.

### 3.3 Gradient Clipping

To address exploding gradients, we used gradient clipping, which rescales large gradients to keep them small. This technique bounds the gradient vector and ensures parameters stay in a good region.

### 3.5 Network Weight Initialization

We compared He, Xavier, and Normal weight initialization methods available in the PyTorch framework, as the choice of initial weights can impact convergence rate and network quality.

### 3.6 Regularization using Batch Normalization and Dropout

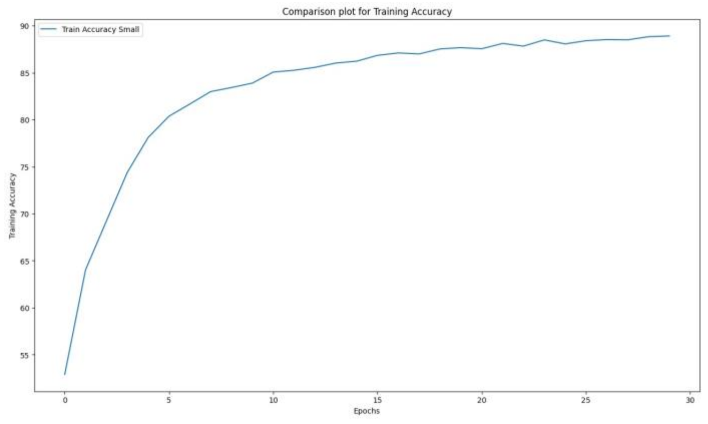
Batch normalization handles differences in training data batches, speeding up the training process, while dropout helps prevent overfitting. Our results show using dropout alongside batch normalization increases test loss compared to using only batch normalization.

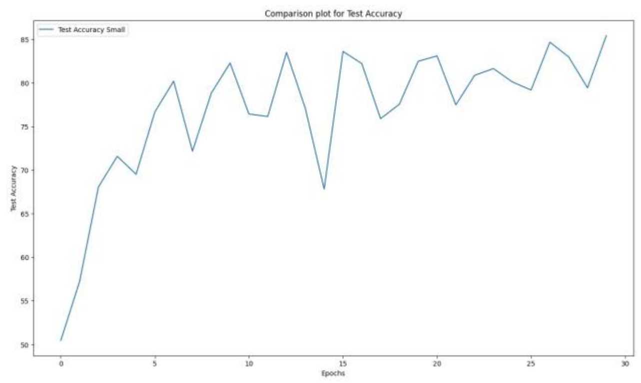
### 3.7 Squeezing and Excitation

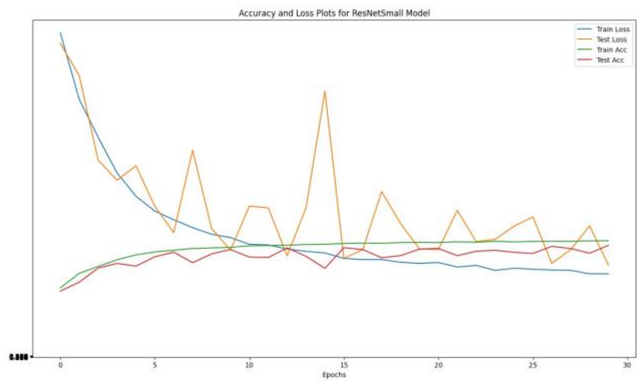
Squeeze-and-excitation blocks recalibrate channel-wise feature responses and model the interdependency between channels. Our experiments show that adding a squeeze-and-excitation block subtly improves test accuracy.

## 4 Results and Discussion

In this part, we evaluate the learning curves of our model, obtained by adjusting various hyperparameters while keeping the trainable parameters below 5 million. Figure 2 illustrates that our model achieves a final test accuracy close to 85.39%, trained without employing any of the earlier discussed training techniques (such as data augmentation and data normalization)







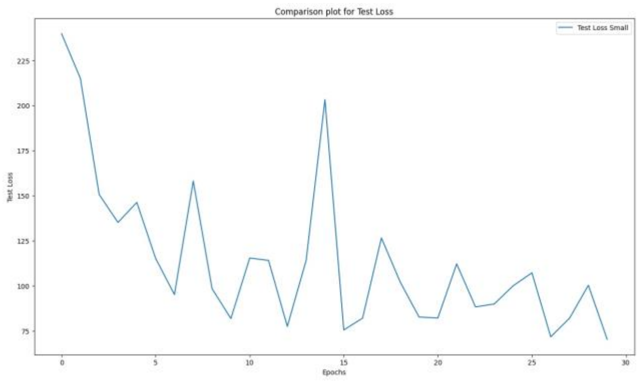


Table 4 presents the final configuration of hyperparameters used for our model and the ResNet18 model.

| Parameters | Our model | ResNet18 |
|---|---|---|
| Number of residual layers | 3 | 4 |
| Number of residual blocks | [4, 4, 3] | [2, 2, 2, 2] |
| Convolutional kernel sizes | [3, 3, 3] | [3, 3, 3, 3] |
| Shortcut kernel sizes | [1, 1, 1] | [1, 1, 1, 1] |
| Number of channels | [64, 128, 256] | [64, 128, 256, 512] |
| Average pool kernel size | 8 | 4 |
| Batch normalization | True | True |
| Dropout | 0 | 0 |
| Squeeze and excitation | True | False |
| Gradient clip | 0.1 | None |
| Data augmentation | True | False |
| Data normalization | True | False |
| Lookahead | True | False |
| Optimizer | SGD | SGD |
| Learning rate (lr) | 0.1 | 0.1 |
| lr scheduler | CosineAnnealing | CosineAnnealing |
| Weight decay | LR 0.0005 | LR 0.0005 |
| Batch size | 128 | 128 |
| Number of workers | 16 | 16 |
| **Total number of parameters** | **4,697,742** | **11,173,962** |

## Conclusion

Our model achieves a test accuracy of over 85.39%, using fewer than half the number of parameters compared to ResNet18

## Acknowledgments

The model we developed is based on ResNet-18 and includes modifications to its hyperparameters, along with the application of several new and effective training strategies.

## References

1. Liu, K. (n.d.). PyTorch CIFAR10. GitHub. Retrieved from https://github.com/kuangliu/pytorch-cifar
2. He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
3. Agarwal, A. (n.d.). Introduction to ResNets. Towards Data Science. Retrieved from https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4