# Deep Learning Lab 3 Report

Shusrith S

PES1UG22AM155

February 16, 2025

## 1 CIFAR-10

CIFAR-10 is a widely used dataset for image classification, consisting of 60,000 color images across 10 classes, including airplanes, cars, cats, and dogs. Each image is 32x32 pixels, with 50,000 for training and 10,000 for testing. It serves as a benchmark for evaluating machine learning models, particularly in deep learning and computer vision research.

## 2 MobileNetV2

MobileNetV2 is a lightweight neural network designed for mobile and edge devices, improving efficiency with inverted residuals and linear bottlenecks. It uses depthwise separable convolutions and a bottleneck structure to reduce computations while maintaining accuracy, making it ideal for tasks like image classification and object detection.

Unlike its predecessor, MobileNetV2 enhances feature reuse with a novel residual connection and linear projection, preventing information loss. This design improves accuracy with fewer parameters, making it widely used in AI applications for mobile and embedded systems.

## 3 Finetuning the model

The PyTorch implementation of MobileNetV2 is used for its modular class based design and ease of use in Python. The MobileNetV2 class consists of three basic members, the features layer, the average pooling layer and the classifier layer. The features layer consists of 17 inverted residual layers and one convolutional layer each at the start and the end of the features layer. The classifier layer is trained over the ImageNet-1K dataset, hence it has a dropout layer and a Dense layer with 1000 output neurons.

For finetuning this model on the CIFAR-10 dataset, the classifier layer is removed and replaced with one average pooling layer, a dense layer of size 128 and an output Dense layer of size 10, with dropout layers in between for regularization. The ImageNet-1K weights are retained, and the model is initially trained by freezing and unfreezing certain layers, along with training the newly added classifier layers.. Freezing and unfreezing is done by setting the requires_grad parameter of the tensor to true or false. Experiments have been conducted to tune parameters like learning rate, batch size and number of trainable layers.

# 4 Experiments

## 4.1 Initial training and finetuning

The custom model with the added layers is first trained by freezing all feature layers and training only the classifier. This is done for five epochs, using Cross Entropy loss, a learning rate of 0.001 and the Adam optimizer. After this, the first ten inverted residual layers are set to require gradients and included in the training process for five epochs.

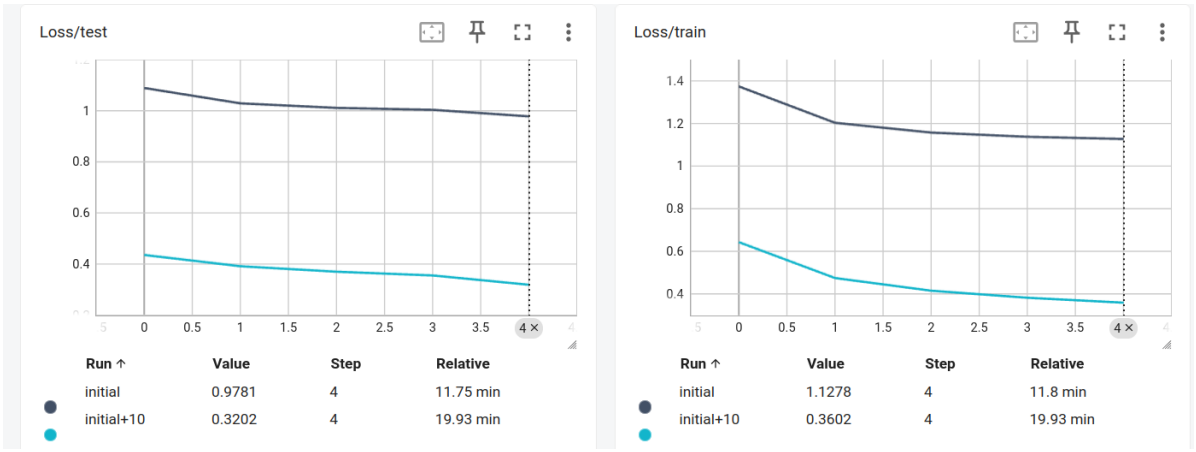


Figure 1: Initial training and 10 layer unfreezed loss


Figure 2: Initial training and 10 layer unfreezed accuracy

The initial training ends with a train loss of 0.978 and test loss of 1.27 and finetuning by training ten layers ends with a train loss of 0.3202 and test loss of 0.3602. The final test accuracy is 87.9% showing that this model benefitted from the initial training followed by the finetuning. It allows the completely untrained classifier layers to train a little and then allows the already trained feature layers to retain their existing knowledge and also learn information from the new data, resulting in a high accuracy despite training for only five epochs.

## 4.2 Hyperparameter tuning

The hyperparameters learning rate, batch size and number of trainable layers were tuned and experimented with to choose the optimum set of hyperparameters for training the model. The experimentation involved trying out a set of values for each hyperparameter and the value with the best accuracy and lowest loss was chosen. A grid search like method is adopted, by training the model exhaustively through a number of parameters to choose the ones in which it performs the best.

## 4.3 Learning rate

The model was trained with learning rates of 0.01, 0.05, 0.001 and 0.005. It was observed that the model performed best for a learning rate of 0.0005, having the lowest loss and highest accuracy and is chosen as the final learning rate to be used.

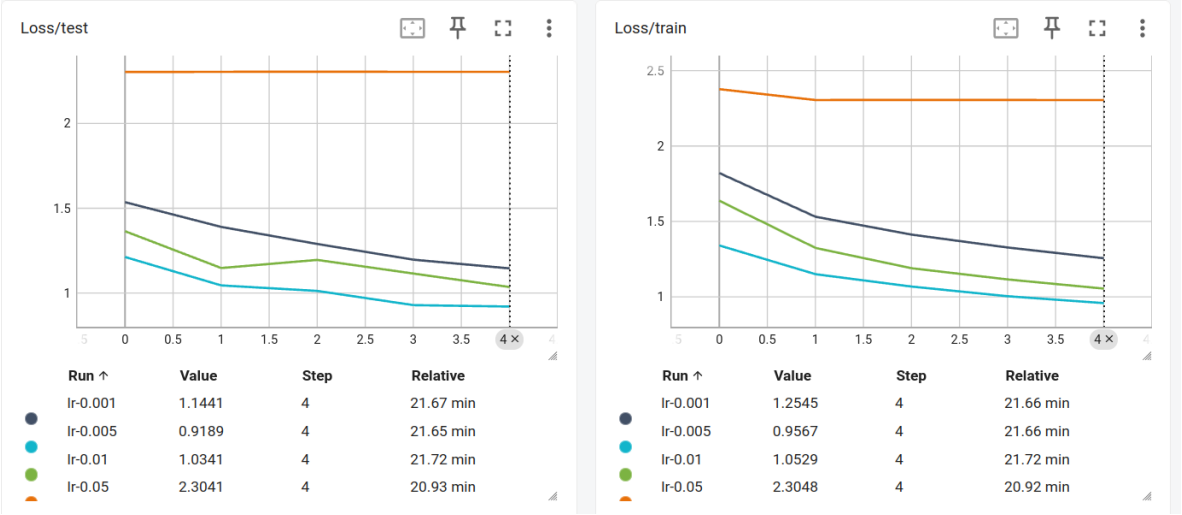


Figure 3: Train and test loss for various learning rates

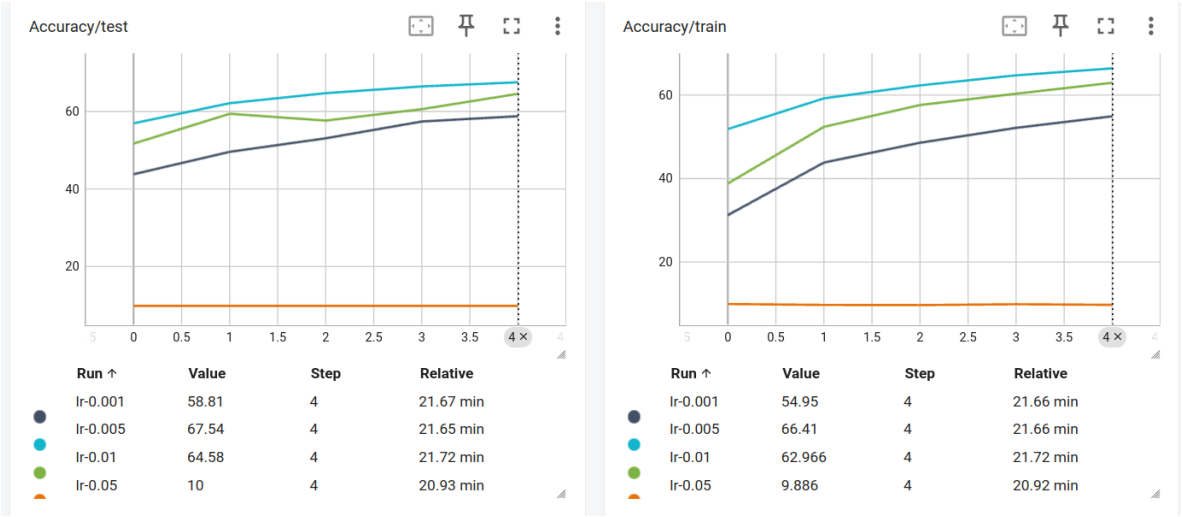


Figure 4: Train and test accuracy for various learning rates

| Learning rate | Train Loss | Test Loss | Train Accuracy | Test Accuracy |
|---|---|---|---|---|
| 0.05 | 2.3041 | 2.3048 | 10 | 9.86 |
| 0.01 | 1.0341 | 1.0529 | 64.58 | 62.96 |
| 0.005 | 0.9189 | 0.9567 | 67.54 | 66.41 |
| 0.0001 | 1.1441 | 1.2545 | 58.51 | 54.95 |

Table 1: Loss and accuracy of various learning rates

## 4.4 Batch size

The model was trained initially with a batch size of 192 and then with sizes of 32, 64, 128 and 192. 192 has the highest overall accuracy and lowest loss, attributed to the smoother gradients that come with higher batch sizes leading to more uniform updates and better training.

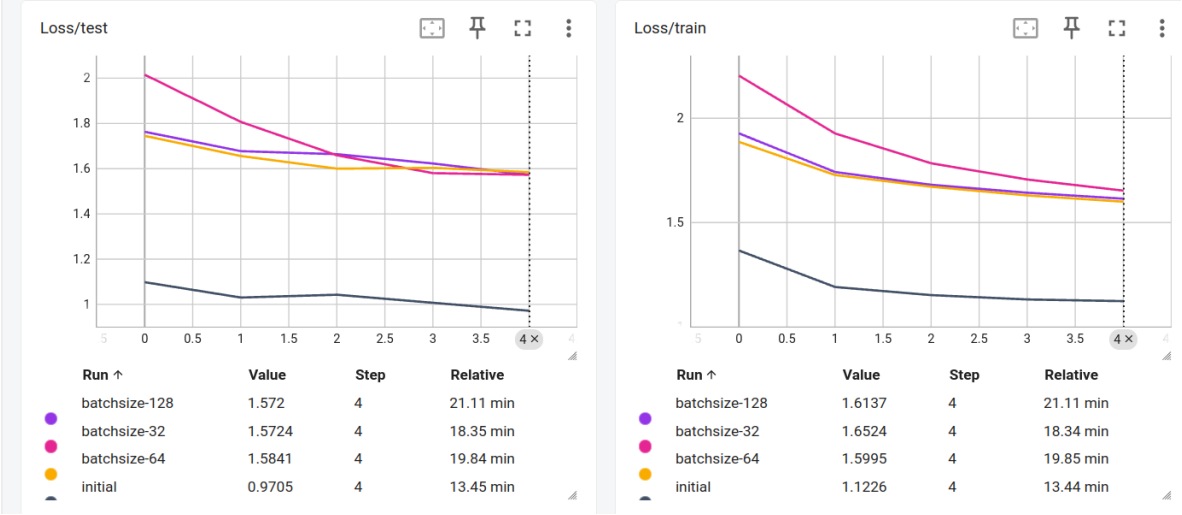


Figure 5: Train and test loss for various batch sizes

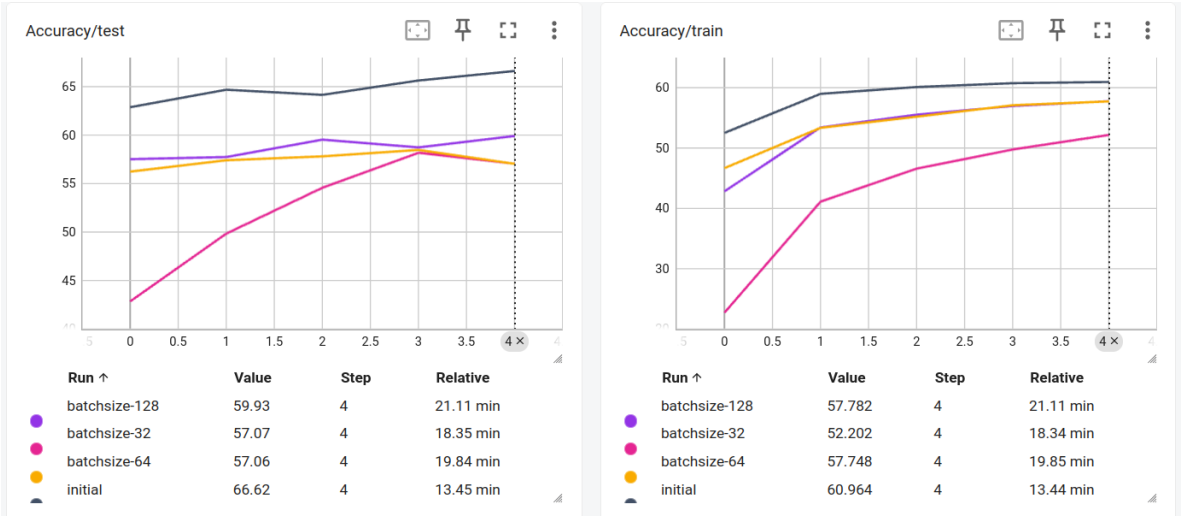


Figure 6: Train and test accuracy for various batch sizes

| Batch Size | Train Loss | Test Loss | Train Accuracy | Test Accuracy |
|---|---|---|---|---|
| 32 | 1.5724 | 1.6524 | 57.07 | 52.202 |
| 64 | 1.5841 | 1.5994 | 57.06 | 57.748 |
| 128 | 1.572 | 1.6137 | 59.93 | 57.782 |
| 192 | 0.9705 | 1.1226 | 66.62 | 60.964 |

Table 2: Loss and accuracy of various learning rates

## 4.5 Number of trainable layers

The model was trained with 2, 5 and 10 trainable layers to determine the optimal number of layers to be trained. 5 and 10 layers have nearly the same train loss and accuracy but 10 layers has a higher test accuracy and lower test loss so it is chosen final training.

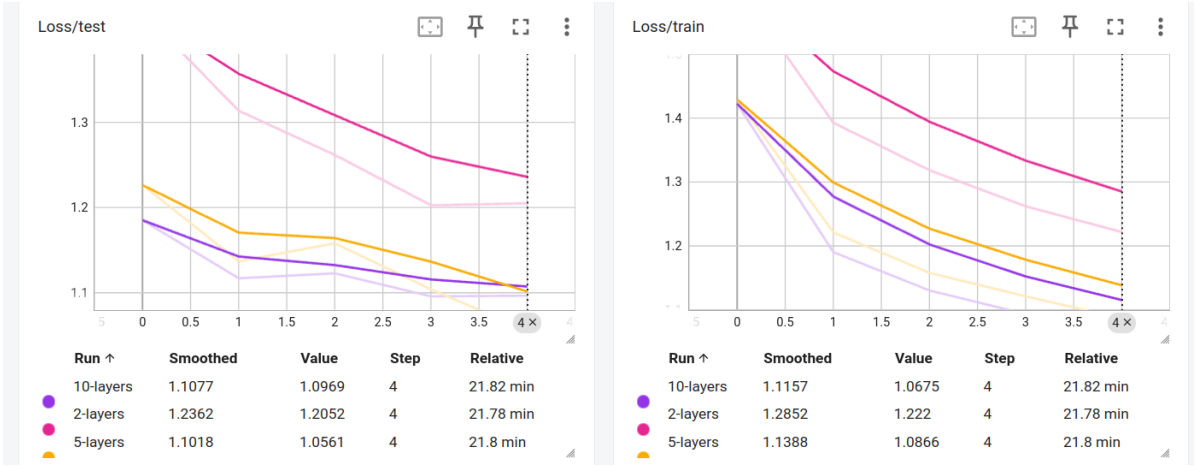


Figure 7: Train and test loss for various learnable layers

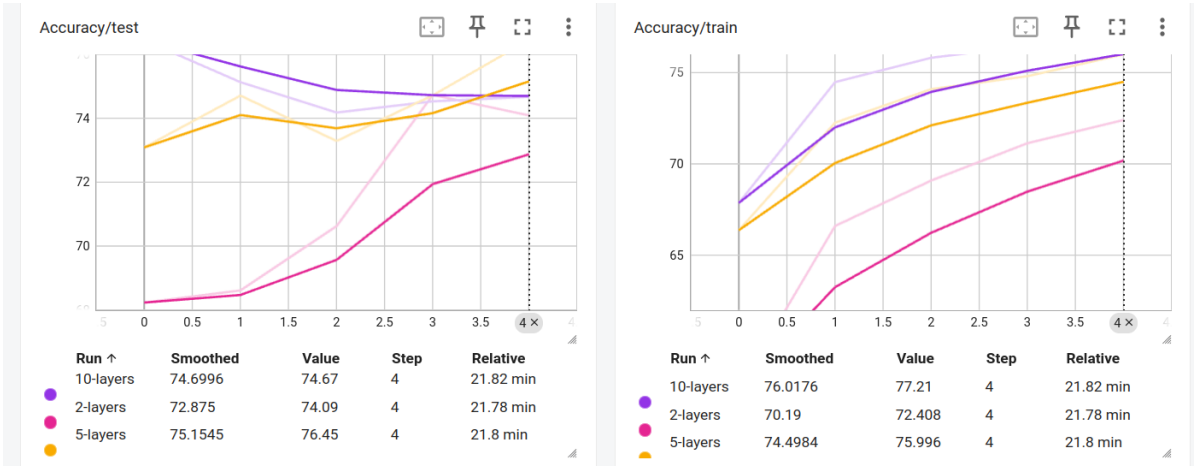


Figure 8: Train and test accuracy for various learnable layers

| No of learnable layers | Train Loss | Test Loss | Train Accuracy | Test Accuracy |
|---|---|---|---|---|
| 2 | 1.2052 | 1.222 | 74.09 | 72.308 |
| 5 | 1.0561 | 1.0866 | 76.45 | 75.996 |
| 10 | 1.1096 | 1.067 | 74.67 | 77.21 |

Table 3: Loss and accuracy of various learning rates

## 4.6 Final chosen hyperparameters

The final chosen hyperparameters are

- Learning rate - 0.005
- Batch size - 192
- Number of trainable layers - 10

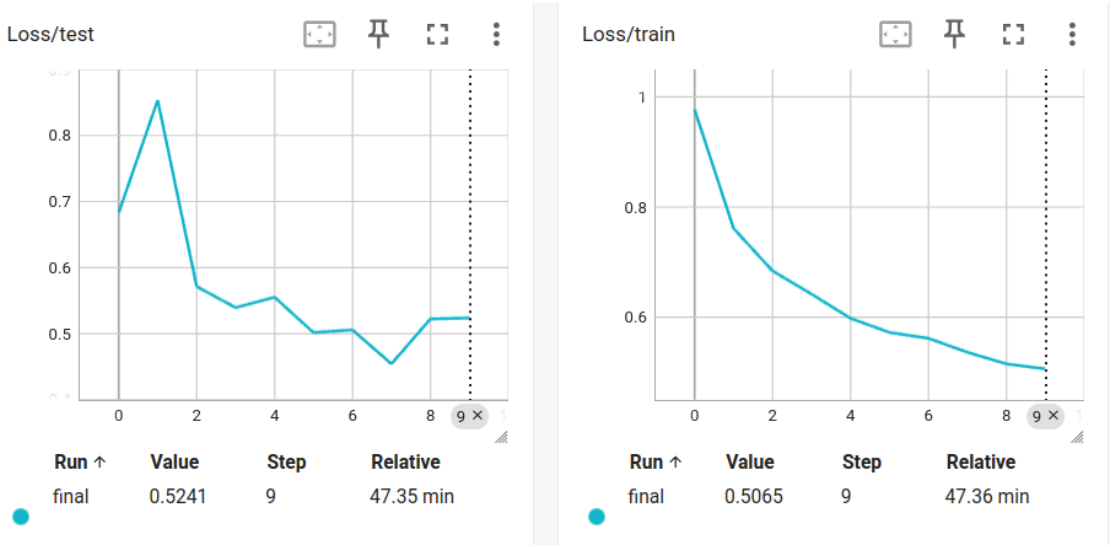


Figure 9: Train and test loss using tuned hyperparameters

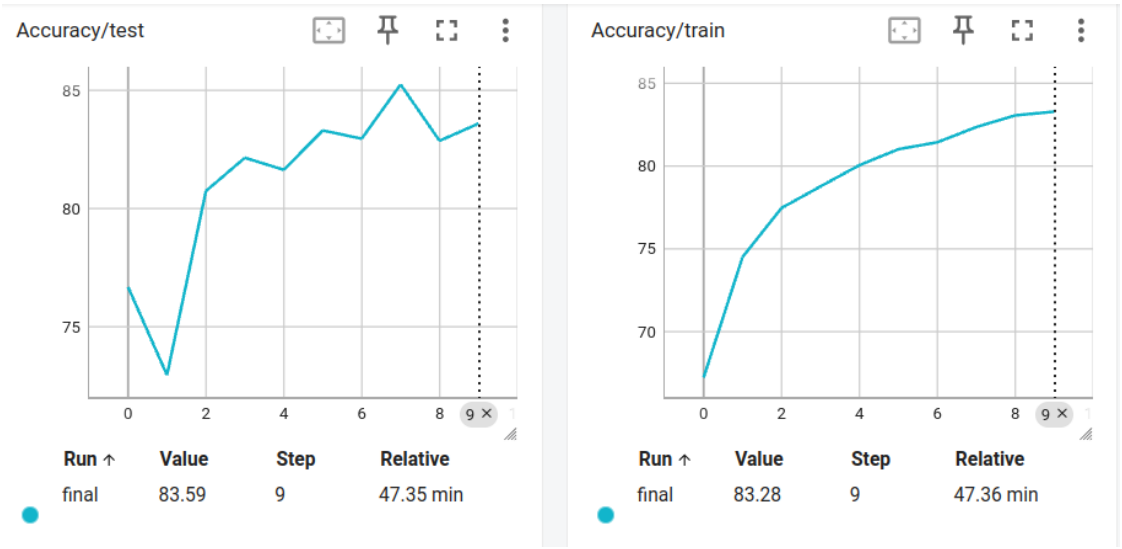


Figure 10: Train and test accuracy for tuned hyperparameters

With the finalized hyperparameters, we can see the model has an even and gradual testing curve in both loss and accuracy and shows balanced and stable performance. The model is able to generalize well and learns without overfitting.