ACML Report

Sayfullah Jumoorty (2430888)

Pratham Tejas Berawala (2441515)

Muhammed Muaaz Dawood (2425639)

Mujammil Mohsin Gulam Sakhidas (2436109)

May 2024

1 Dataset

1.1 Data Source

We utilized the CIFAR-10 dataset for this project. CIFAR-10 is an established computer-vision dataset used for object recognition tasks. Instead of downloading the full dataset from Kaggle, we imported it directly in our code using Keras and TensorFlow, leveraging it for ease of use and integration. The Python Keras dataset is a subset of the 80 million tiny images from Kaggle and consists of 60,000 32x32 color images, each belonging to one of 10 object classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class contains 6,000 images. The dataset was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton and has been widely used in the machine learning community for benchmarking algorithms.

1.2 Objectives

The primary objective of utilizing the CIFAR-10 dataset in this project is to develop, train, and evaluate a machine learning model capable of accurately classifying images into one of the 10 predefined object categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

Specific objectives include:

- Preprocessing the dataset to ensure it is in an optimal format for model training.
- Designing and implementing a convolutional neural network (CNN) to effectively learn from the CIFAR-10 images and generalize to unseen data.
- Training the model using the training dataset and fine-tuning hyperparameters to achieve the best performance.
- Evaluating the model's performance on the test dataset using metrics such as accuracy, precision, recall, and F1-score.

2 Dataset Pre-Processing

In our implementation, the dataset preprocessing includes the following steps:

1. **Loading the Dataset:**

   The CIFAR-10 dataset consists of 60,000 32×32 color images, with the dataset already split into 50,000 training images and 10,000 testing images. After loading and preprocessing this data, scikit-learn's `train_test_split` function is used to further split the 50,000 training images into a training set $(x_{\text{train}}, y_{\text{train}})$ comprising 80% of the data, and a validation set $(x_{\text{val}}, y_{\text{val}})$ comprising the remaining 20%, with a fixed `random_state` for reproducibility. The original 10,000 test images $(x_{\text{test}}, y_{\text{test}})$ remain as the test set. During model training, the validation set is used for hyperparameter tuning and model selection, while the test set evaluates the final trained model's performance.

2. **Normalization:** To ensure the pixel values are on a comparable scale, we normalize the image data by scaling the pixel values to the range [0, 1]. This is done by dividing each pixel value by 255.0.

3. **One-Hot Encoding:** The class labels provided in the dataset are integers ranging from 0 to 9. For training the neural network, we convert these labels to one-hot encoded vectors using TensorFlow's **Categorical** function which converts a class vector (integers) into a binary class matrix.

4. **Data Augmentation:** To improve the generalization of the model and prevent over-fitting, we apply data augmentation techniques to the training images. This includes:

    - **Rotation:** Randomly rotating the images within a range of 15 degrees.
    - **Width Shift:** Randomly shifting the images horizontally by 10
    - **Height Shift:** Randomly shifting the images vertically by 10
    - **Horizontal Flip:** Randomly flipping the images horizontally.

    Data augmentation is implemented using TensorFlow's **ImageDataGenerator** class, which generates batches of tensor image data with real-time data augmentation.

By performing these preprocessing steps, we ensure that the data fed into the neural network model is in an optimal format, which helps improve the accuracy and robustness of the model.

3 The Model

In this section, we describe the choice and implementation of the model used for image classification on the CIFAR-10 dataset.

3.1 Model Choice: Convolutional Neural Network (CNN)

The model chosen for this implementation is a Convolutional Neural Network (CNN). CNNs are particularly well-suited for image classification tasks due to their ability to capture spatial hierarchies in images. The specific architecture of the CNN we developed in this implementation is designed to handle the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class.

3.2 Machine Learning Algorithm: Supervised Learning

The learning algorithm employed is supervised learning, specifically using the categorical cross-entropy loss function. This is appropriate for classification tasks where the model is trained to predict the probability distribution over multiple classes.

3.3 Model Iterations

To find a suitable model and fine-tune it for the CIFAR-10 dataset, several experiments were conducted with different architectures and hyperparameters. The goal was to improve the model's performance and generalization ability. The following experiments were performed:

1. **Baseline Model:**

    - A simple CNN architecture was used as a starting point, consisting of one convolutional layer, max pooling, and dense layers.
    - This model achieved a test accuracy of 0.6065, serving as a baseline for further improvements.

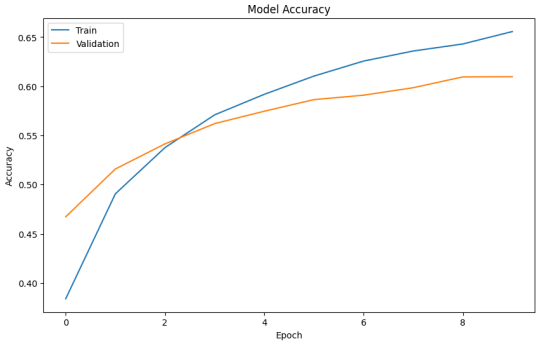

Figure 1: Baseline Model Accuracy

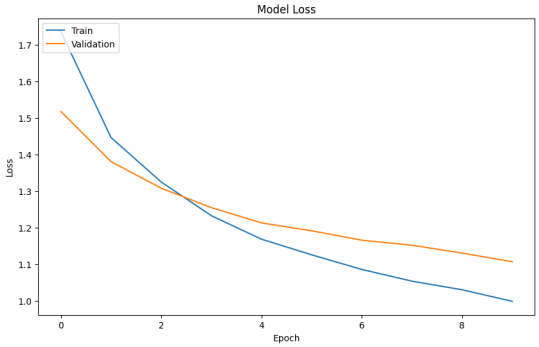

Figure 2: Baseline Model Loss

2. **Increasing Model Depth:**

   - The model architecture was deepened by adding more convolutional layers.
   - The final architecture consisted of three sets of two convolutional layers, each followed by max pooling.
   - Increasing the depth allowed the model to learn more complex features, resulting in an improved test accuracy of 0.7137.

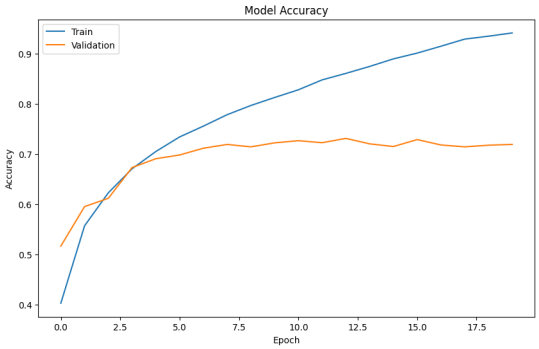

Figure 3: Depth Model Accuracy

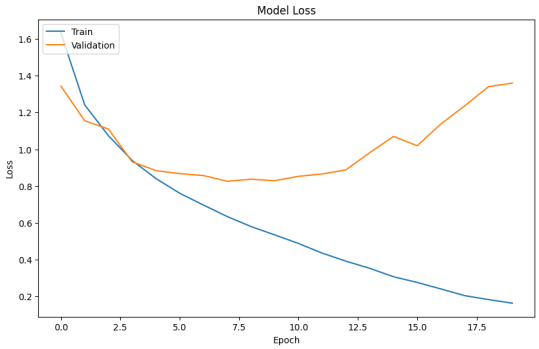

Figure 4: Depth Model Loss

3. **Batch Normalization:**

   - Batch normalization layers were added after each convolutional layer.
   - Batch normalization helps stabilize the training process by normalizing the activations and reducing internal covariate shift.
   - The model achieved a higher training accuracy of 0.9858, but the test accuracy was 0.7227 suggesting potential over-fitting however it still gave a better accuracy than the previous model.

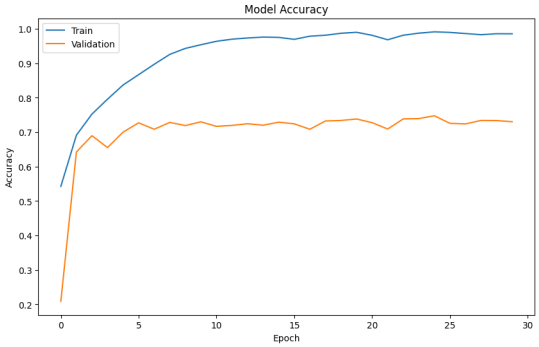

Figure 5: Batch Normalization Model Accuracy

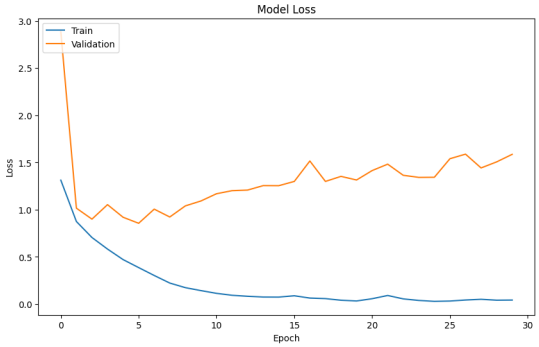

Figure 6: Batch Normalization Model Loss

4. **Data Augmentation:**

   - Image data augmentation techniques were applied to expand the training dataset and improve the model's ability to generalize.
   - Techniques such as rotation, width and height shifts, and horizontal flipping were used.
   - Data augmentation helped the model learn invariance to these transformations, resulting in an improved test accuracy of 0.7593.

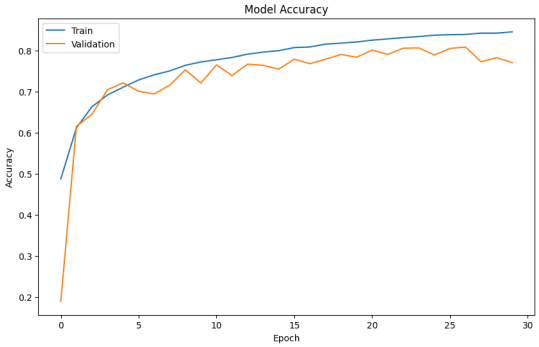

Figure 7: Data Argumentation Model Accuracy

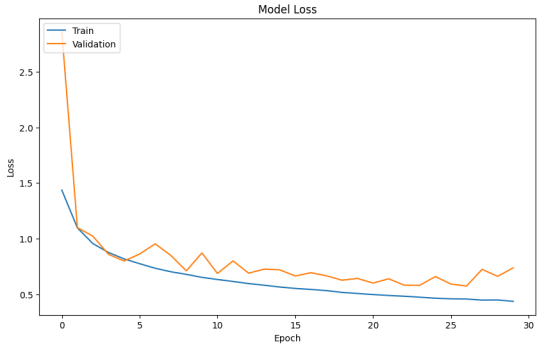

Figure 8: Data Argumentation Model Loss

5. **Learning Rate and Optimizer:**

   - Experiments were conducted with different learning rates (0.001, 0.01, 0.1) and optimizers (Adam, SGD).
   - The Adam optimizer with a learning rate of 0.01 yielded the best test accuracy of 0.7889.
   - Our findings are supported by a comparative study on the performance of different optimizers on deep learning tasks using the CIFAR-10 dataset, as seen in 9. The figure shows that while the Adam optimizer quickly reaches high accuracy, other optimizers like RMSprop, Adadelta, and Adagrad also achieve similar accuracy levels but with varying rates of convergence. In contrast, SGD converges much more slowly and requires more epochs to reach comparable accuracy.

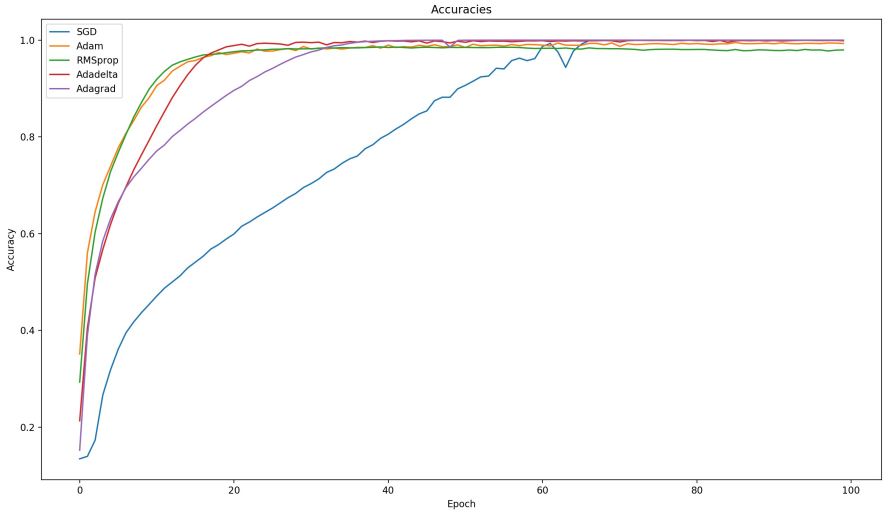

Figure 9: Different Optimizer Results

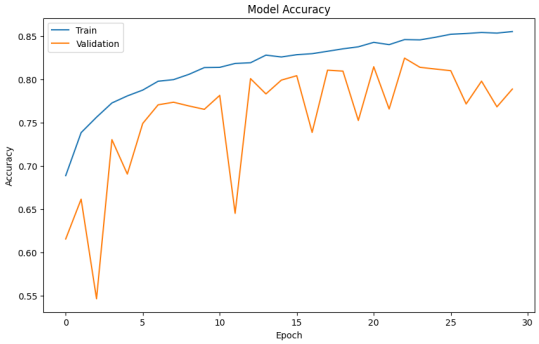

Figure 10: Best optimizer Model Accuracy

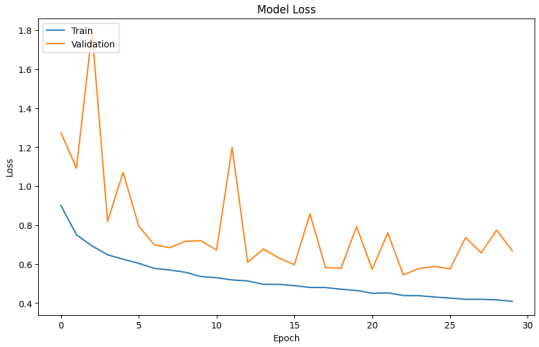

Figure 11: Best optimizer Model Loss

6. **Early Stopping:**

- Early stopping was implemented to prevent over-fitting and find the optimal number of training epochs.
- The model's performance on the validation set was monitored, and training was stopped if the validation loss did not improve for 10 epochs.
- Early stopping helped achieve a high test accuracy of 0.8313, demonstrating its effectiveness in preventing over-fitting.

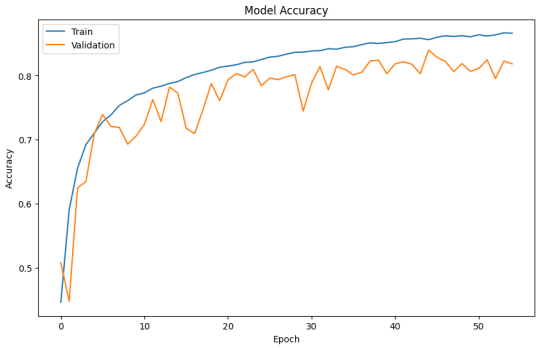

Figure 12: Early Stopping Model Accuracy

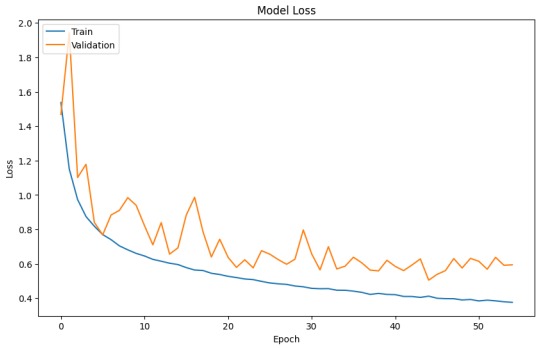

Figure 13: Early Stopping Model Loss

7. **Regularization Techniques:**

- Dropout layers were introduced after each max pooling layer and before the final dense layer.
- Dropout helps prevent over-fitting by randomly dropping out a fraction of the neurons during training.
- The test accuracy resulted to 0.7811, averaging below 0.8 almost all the time, indicating the ineffectiveness of regularization in reducing over-fitting in this specific case.

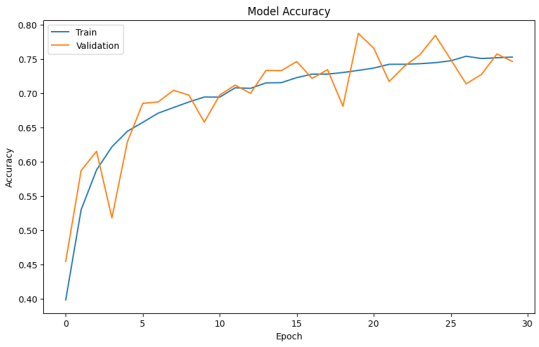

Figure 14: Dropout Model Accuracy

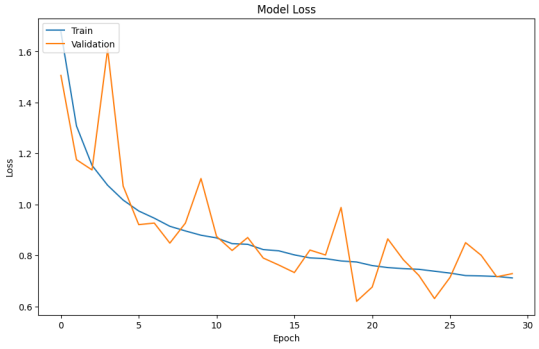

Figure 15: Dropout Model Loss

- The learning rate of 0.01 was selected because it offered a good balance between fast convergence and training stability, allowing the model to learn effectively without causing oscillations or divergence.
- The Adam optimizer was chosen due to its ability to achieve high accuracy quickly and efficiently. It often leads to better generalization in various deep learning tasks compared to other optimizers.
- The number of epochs was set to 100 with early stopping to provide ample training time, ensuring the model could learn sufficiently from the data while early stopping helped to mitigate the risk of over-fitting.
- The batch size of 128 was used as it strikes an optimal balance between computational efficiency and the stability of gradient updates, allowing for effective learning without exhausting memory resources.

In conclusion, the experiments conducted helped identify the most suitable model architecture and hyperparameters for the CIFAR-10 dataset. The final model, which incorporated data augmentation, batch normalization, and early stopping, achieved a test accuracy of 0.8313, demonstrating significant improvement over the baseline model. The choice of hyperparameters was guided by empirical results and common practices, leading to a well-performing and generalized model.

4 Final Model

The final model developed for image classification on the CIFAR-10 dataset represents a culmination of iterative experimentation and refinement. Through a series of model iterations, encompassing adjustments to architecture, hyperparameters, and various techniques, we aimed to enhance both accuracy and generalization. Our model, a Convolutional Neural Network (CNN) augmented with data augmentation, batch normalization, and early stopping, achieved a notable test accuracy of 0.84, selected from hundreds of runs done on the MSL Cluster.

The selection of optimal hyperparameters, informed by empirical evidence and best practices, facilitated robust learning while mitigating the risk of over-fitting. The training accuracy and loss curves for this final model are shown in Figure 16, respectively. Similarly, it illustrates the validation accuracy and loss curves, which were used for early stopping to prevent over-fitting.

By systematically exploring various configurations and techniques, we successfully identified a model capable of delivering improved performance compared to baseline iterations. This final model stands as a testament to the efficacy of meticulous experimentation in the pursuit of superior machine learning outcomes.

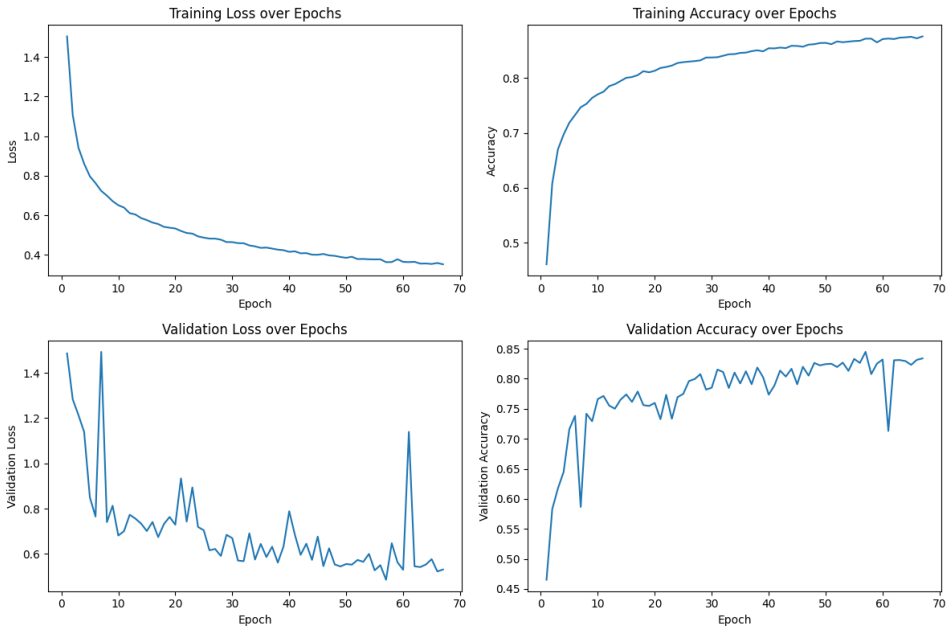

Figure 16: Final Model Graphs

5 Results

The latest results obtained from running the chosen model on the MSL cluster, show a test accuracy of 0.84. A detailed analysis of the classification report, shown in Figure 18, reveals promising performance across all classes, with precision, recall, and F1-score metrics consistently above 0.80 for most categories. Particularly noteworthy are the high scores for automobile, ship, and truck classes, indicating robust classification capabilities for transportation-related objects.

However, slight discrepancies in precision and recall metrics across classes, as illustrated in Figure 19, suggest potential areas for further optimization or fine-tuning. The confusion matrix in Figure 17 provides a visual representation of the model's performance, highlighting the classes with higher misclassification rates.

Overall, the heightened test accuracy and balanced performance metrics underscore the efficacy of our final model, affirming its readiness for real-world deployment and validation in diverse scenarios.

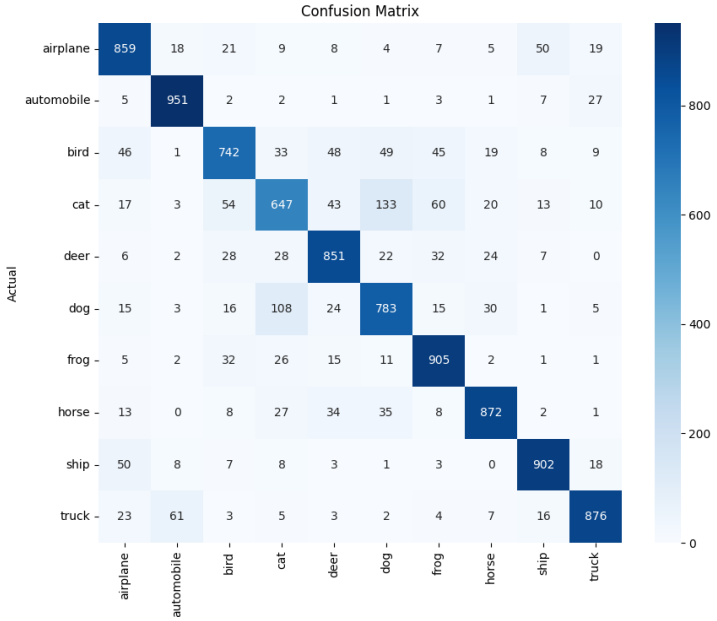

Figure 17: Confusion Matrix

Class	Precision	Recall	F1-Score
airplane	0.83	0.86	0.84
automobile	0.91	0.95	0.93
bird	0.81	0.74	0.78
cat	0.72	0.65	0.68
deer	0.83	0.85	0.84
dog	0.75	0.78	0.77
frog	0.84	0.91	0.87
horse	0.89	0.87	0.88
ship	0.9	0.9	0.9
truck	0.91	0.88	0.89

Figure 18: Class Scores

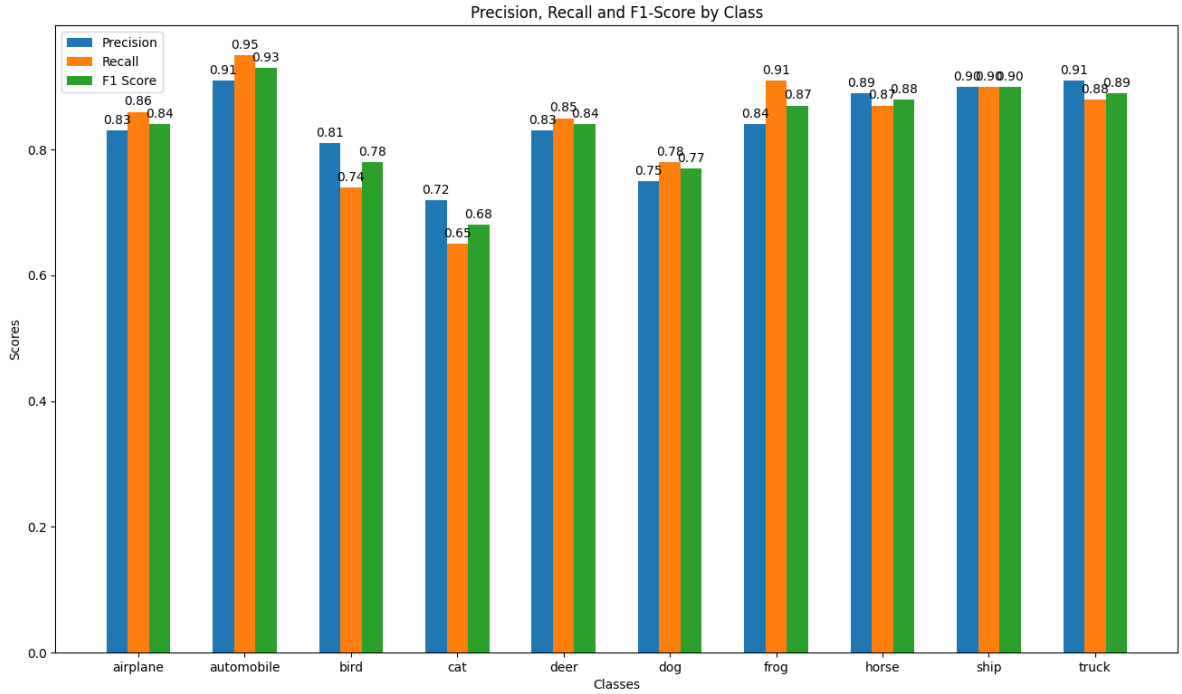

Figure 19: Plot of Class Scores