Deep Learning Assignment: Utilizing CNN for image classification of CIFAR 10 data

Sidharth Serjy

November 10, 2024

1 Introduction

The CIFAR-10 dataset consists of 60,000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images.In the context of the CIFAR-10 dataset, "classes" referred to the 10 different categories that the model was trained to classify. Each class represents a different category in the CIFAR-10 with labelled images in the following categories: Class 0: Airplane, Class 1: Automobile, Class 2: Bird, Class 3: Cat, Class 4: Deer, Class 5: Dog, Class 6: Frog, Class 7: Horse, Class 8: Ship, Class 9: Truck.

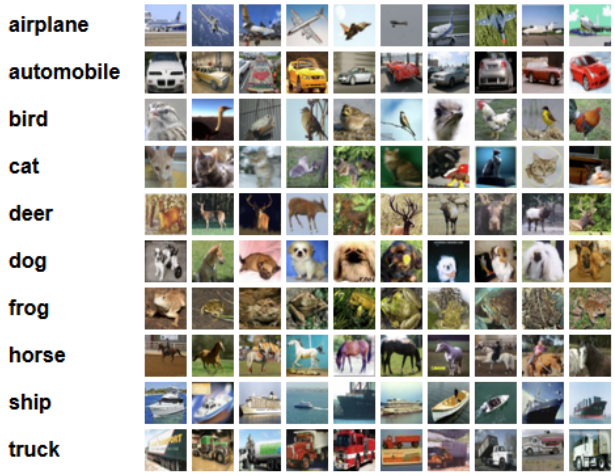

Figure 1: *CIFAR 10 image classes*

My task is to train and test several machine learning algorithms to determine which of the methods to use to train on the CIFAR dataset. By trying and testing with different formats of architecture and also tweaking the values of hyperparameters, I can arrive at the right model which correctly analyzes the features of an image in CIFAR for a pass and comprehensive classification.

2 Building the Custom Baseline CNN Model

In this report, I have developed a CNN model for the CIFAR-10 dataset, which includes 60,000 32x32 colour images in 10 classes. The dataset was loaded using TensorFlow's CIFAR-10 API. The images were then normalised in the pixel values in a range from 0 to 1, and divided them by 255. It would help the model converge faster and generally improve stability during training. I then transform the labels into one-hot encoded vectors which creates a 10-dimensional output vector for each image, with each element representing the probability of the image belonging to a particular class.

2.1 One-hot Encoding

One-hot encoding is a technique of converting the categorical labels to numerical. In this task, each CIFAR-10 label class is represented as a vector. The vector representing every image label contains a 1 in the position corresponding to its class and 0s elsewhere. For example, for an image whose class is 3, its one-hot encoded vector shall look something like this: [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].

The reason this transformation is necessary is because a later step in training will be calculating the categorical cross-entropy loss. The categorical cross-entropy is a measure of the difference between the true label's one-hot encoded probability distribution and the predicted probability distribution created by the softmax output layer of the model. The aim of training, in that respect, is to minimize this difference, which means the model will learn to predict high probabilities for the correct classes. The one-hot encoding here ensures each class gets an equal say in the calculation of loss, which is important for effective training in multi-class classification.

For the model validation and generalization, I split the training data till the formation of the training and the validation data set. Therefore, the total of the training data will be split for the validation sets by 20%. It will give a chance to better adjust the model and decide what happens to the performance of the model, when it is tested on new data, and therefore, make the early detection of over or under fitting more efficient.

2.2 CNN Model Layers

I define the CNN model architecture using a Sequential model, which lets me stack layers in a linear fashion. The network starts with three convolutional lay-

ers, each with an increasing number of filters (32, 64, and 128). Each convolutional layer applies a 3x3 kernel with ReLU activation and 'same' padding to maintain the input dimensions. After each convolutional layer, I add a MaxPooling layer to reduce spatial dimensions by half, decreasing computation and emphasizing the key features in the input images.

2.2.1 ReLU activation

The ReLU (Rectified Linear Unit) activation function is used to introduce non-linearity. ReLU is defined mathematically as:

$$\text{ReLU}(z) = \max(0, z)$$

where z is the output of the convolution operation. This function outputs the value z if it is positive; otherwise, it outputs zero. ReLU accelerates training by making the network sparse (i.e., many neurons become inactive or zeroed out), which reduces computation and helps avoid vanishing gradients—a common issue in deep networks where gradients become too small to propagate back through layers, hindering learning. This was one reason other activation functions were avoided.

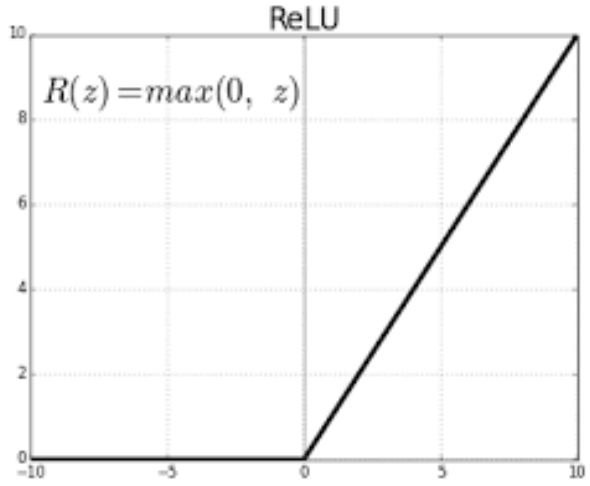

Figure 2: *ReLU activation function*

2.2.2 Padding

In this model, the padding is set to "same," which means that the input and output feature maps have the same spatial dimensions. Mathematically, to achieve this, we add a padding of p pixels on each side, where:

$$p = \frac{k-1}{2}$$

for a kernel of size k. For example, a 3×3 filter requires a padding of 1 pixel on each side. This padding ensures that edge features are retained in the convolutional layers, giving the model full access to all regions of the image.

2.2.3 Max-Pooling Layer

After the activation function, each convolutional layer is followed by a MaxPooling layer, which reduces the spatial dimensions of the feature map. This layer computes the maximum value within each 2×2 region. Mathematically, for a region R of size 2×2, the output of MaxPooling at position (x, y) is:

$$\text{MaxPooling}(x, y) = \max(R_{11}, R_{12}, R_{21}, R_{22})$$

This operation effectively downsamples the feature map, reducing the model's computational load and capturing the most prominent features, which enhances translational invariance (the ability to recognize objects regardless of small shifts in their position).

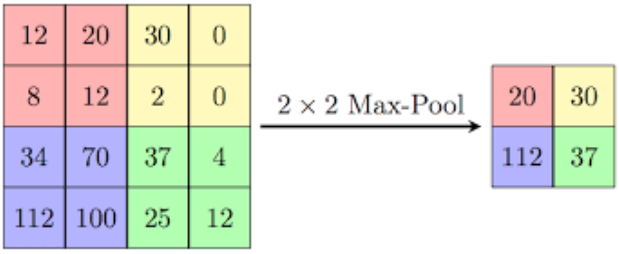

Figure 3: *Example of Maxpooling*

2.2.4 Softmax Activation Function

The final layer uses the softmax activation function to produce probabilities for each of the 10 CIFAR-10 classes. Softmax converts the outputs into a probability distribution by calculating each class probability as:

$$\text{Softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{10} e^{z_k}}$$

where z_j is the output of the j-th neuron in the final layer. This function ensures that all class probabilities sum up to 1, allowing the model to make a confident prediction by selecting the class with the highest probability.

Together, these mathematical operations form the basis of the CNN, enabling it to progressively extract features and make predictions on complex image data, like the CIFAR-10 classes. Each layer's operations contribute to building a hierarchical understanding of the image, from simple edges to more complex textures and shapes, culminating in class probabilities in the softmax layer.

2.3 Compilation of the CNN Model

After the layers are decided the model preparation was done by compiling it where the optimizer of choice, loss function and the evaluation metric was set, and the model was trained on the chosen dataset while testing the model on the validation set.

```
# Define optimizers
optimizers = {
    "SGD": SGD(learning_rate=0.01),
    "Nesterov Momentum": SGD(learning_rate=0.01, momentum=0.9, nesterov=True),
    "Adagrad": Adagrad(learning_rate=0.01),
    "Adadelta": Adadelta(learning_rate=1.0),
    "RMSProp": RMSprop(learning_rate=0.001),
    "Adam": Adam(learning_rate=0.001)
}
```

Figure 4: *Optimizers for Compilation*

Adam is a better choice as it dynamically scales the learning rate for each parameter of the function, which makes it fuse the operations of Adagrad and RMSProp. Due to this adaptableness of the model, the model will converge even faster and perform well on various tasks.

Categorical cross-entropy has been a perfect match for the loss function, especially for multi-class classification where it measures the deviation of the true and predicted probabilities in one hot format. This directs the model to reduce this difference as can be seen below.

3 Hyper-parameter Tuning the CNN model for better model performance

In order to look for ways of improving the accuracy of the baseline CNN model, I have performed grid search and random search procedures. This has led to selection of appropriate hyperparameters in this model through enhancing its performance on CIFAR-10 dataset.

3.1 Grid-Search on Baseline CNN Model

To fine-tune the baseline model created, the option of grid-search was used at first to identify the best parameters of the custom-made CNN model. Before each run the TensorFlow backend is cleaned with the purpose to prevent that the previous models and weights affect the current model. The very type of stopping is used in the training process to avoid over-training.

Each option in hyperparameters is specified with the help of ParameterGrid from scikit-learn to consider all the possible combinations of the dictionary of hyperparameters. The combinations of four parameters are two optimizers, including SGD with Nesterov momentum and Adam; two dropout rates 0.3 and 0.5; two batch sizes 32 and 64; the learning rate is set to 0.001. However, as we are fixing it at 20 epochs, early stopping allows training to stop if no improvement has been made in validation loss even though we mention 20 epochs.

The best parameters identified were a batch size of 32, a dropout rate of 0.5, 20 epochs, a learning rate of 0.001, and the Adam optimizer, achieving a validation accuracy score of 0.7407.

3.2 Random Search on Baseline CNN Model

This code randomly searches over the spectrum of the hyperparameters of a CNN model training CIFAR-10 dataset. A model architecture that includes convolutional, pooling, dense and dropout layers of neural networks. With the help of a given array of possible hyperparameters, the code then selects ten random sets of parameters: the optimizer (SGD or Adam), the dropout rate, the batch size, epochs, and learning rate. For the training of each of the sampled combinations, a set of iterations with early stopping is used to reduce overfitting. The validation accuracy is then noted and the best parameter set according to validation accuracy is printed as the final output. This approach is useful in that it enables the processing of hyperparameter space without a strict need to complete grid search.

The best parameters found were an optimizer of Adam, a dropout rate of 0.5, a batch size of 128, 20 epochs, and a learning rate of 0.001, achieving a validation accuracy score of 0.7509.

3.3 Grid-Search on Baseline CNN Model with Image Augmentation

In this work, I have done grid search on CNN model for the CIFAR-10 data set along with the data augmentation to make the model more robust. Normalization of the pixel values and one hot coding of labels are performed to the dataset.

Images go through a horizontal flip using ImageDataGenerator where a larger quantity of images are used during training and this will increase generality. This is the additional step apart from the steps already explained earlier in grid search.

The optimal parameters were found to be a batch size of 64, a dropout rate of 0.3, 20 epochs, a learning rate of 0.001, and the Adam optimizer, achieving a validation accuracy score of 0.7350.

4 Other Models used for Comparison for Performance

Apart from the custom-made CNN model, I have evaluated the performance of commonly used models, including ResNet-18, AlexNet, and MobileNet, to compare their effectiveness.

4.1 ResNet 18 Model

ResNet-18 is a CNN model with 18 layers, designed to address the vanishing gradient problem in deep networks. It introduces the concept of residual learning, where shortcut connections allow the model to bypass certain layers, enabling gradients to flow more easily during backpropagation. ResNet-18 is often used for tasks involving image classification and can provide strong baseline performance due to its efficient design and balanced depth.

4.1.1 Architecture and Modifications

Currently, the ResNet-18 contains 18 layers with residual connectivities that help to bring information from the earlier layers and make the learning process effective with a low risk of overfitting. For this project, the final layer of fully connected layer was modified to predict the three classes of interests. Using weights from the pre-trained ImageNet database, this model begins from general image features which is useful for medical image analysis.

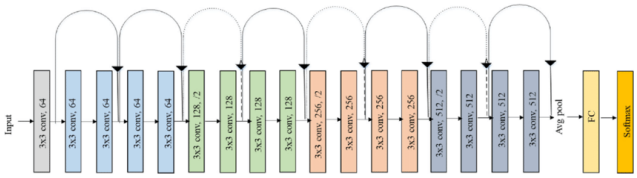

Figure 5: *ResNet 18 Architecture*

4.1.2 Key Pointers

- New layers incorporate a global average pooling layer to decrease dimensions, a dropout layer to regulate overfitting and a dense output layer with softmax activation for network 10 CIFAR-10 classes categorization. The model is compiled with a particular optimizer and categorical cross-entropy loss as the model often needs to perform multi-class classification.
- The model has two optimizers: SGD with Nesterov momentum and Adam; the difference in optimizer strategies allow to compare their performance. Technique used to stop training after the validation loss has not started to improve in 3 consecutive epochs. Furthermore, the model goes through an augmented data generator, providing horizontally flipped images, rotated and shifted images which makes the model more robust. This will help to train a model with the best parameters and so get the final model using these best parameters.
- Lastly, the grid search approach is utilized with a parameter grid to obtain various hyperparameter setting options. For each value of each parameter, the model is trained, and the validation accuracy of the model is noted. Finally, the best parameters of the configurations are selected based on the maximum validation accuracy achieved and shown as the suitable configuration for ResNet-18 inspired model on CIFAR-10 dataset.
- The best parameters identified were a batch size of 32, a dropout rate of 0.5, 20 epochs, a learning rate of 0.001, and the SGD optimizer, achieving a validation accuracy score of 0.4367.

4.2 AlexNet Model

The AlexNet model comprises several convolutional and pooling layers designed to capture hierarchical features in images, followed by fully connected layers for classification. Its architecture incorporates ReLU activations and dropout layers, helping to reduce overfitting and improve learning efficiency.

4.2.1 Architecture and Modifications

Max-pooling layers are used following specific convolution layers that therefore decreases spatial dimension while emphasizing features. The networks then flatten the feature maps before passing them through two fully connected layers with 4096 neurons. The last layer with softmax activation gives the class probability for the data classes.

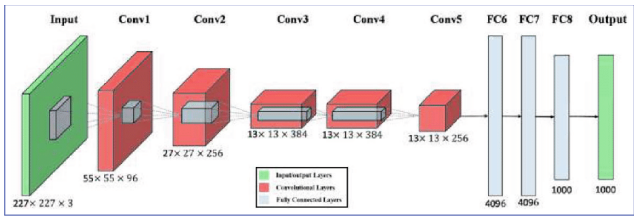

Figure 6: *AlexNet Architecture*

4.2.2 Key Pointers

- The AlexNet model is modified to work with CIFAR-10, using convolutional and pooling layers and dense layers. Batch normalization layers make the objective function smooth, and they help avoid depending on the first layer's scale while dropout layers randomly set neurons to be 0 during the training phase.
- Hyperparameter Tuning via Grid Search by choosing the best hyperparameters concerning an optimizer (SGD or Adam), dropout ration, batch size, learning rate, and number of epochs. For each of the parameter combinations, the average validation accuracy is compared and overfitting is avoided by using early stopping.
- The model applies data augmentation that allows to improve the model performance on different unseen data.
- The best parameters identified were a batch size of 32, a dropout rate of 0.3, 20 epochs, a learning rate of 0.001, and the SGD optimizer, achieving a validation accuracy score of 0.7331.

4.3 MobileNet Model

The MobileNet model consists of depthwise separable convolutions, which efficiently capture spatial and channel-wise features while significantly reducing computational cost. This architecture includes ReLU activations and dropout layers, enhancing learning efficiency and minimizing overfitting. For this project, the final fully connected layer was modified to classify the three target classes.

4.3.1 Architecture and Modifications

Taking advantage of using depthwise separable convolution which can detect both depth and its separation MobileNet can capture accurate features within simple layers. This design has less computational complexity and improves the learning process with less chances of over-fitting.

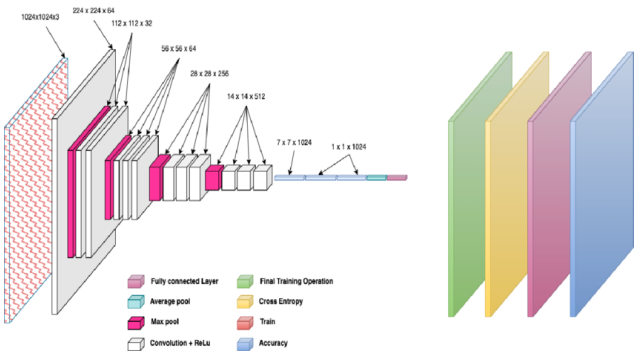

Figure 7: *MobileNet Architecture*

4.3.2 Key Pointers

- The MobileNet model is used without its top layers, and custom layers, including a global average pooling layer and dropout, are added to adapt it for the CIFAR-10 dataset. A dense output layer with softmax activation is included for multi-class classification.
- The code uses grid search to optimize key hyperparameters, including optimizer type (SGD or Adam), dropout rate, batch size, epochs, and learning rate. Each combination is tested, and the one yielding the highest validation accuracy is chosen as the best configuration.
- The model applies horizontal flip augmentation to increase image diversity during training, while early stopping prevents overfitting by halting training if validation loss does not improve for three epochs.
- The best parameters identified were a batch size of 32, a dropout rate of 0.3, 20 epochs, a learning rate of 0.001, and the SGD optimizer, achieving a validation accuracy score of 0.5859.

5 Evaluation Metrics

Using the best parameters found by the hyperparameter tuning, I have trained and tested different models which include a new CNN model along with ResNet-18, AlexNet and MobileNet. Using the better configurations for batch size and learning rate, dropout rate, and the optimizer, I hoped to achieve the maximum of every model's performance and properly evaluate their efficiency when working with CIFAR data. With this comparison, I can see how various architectures behave on the same data and which of them has the highest ability to generalize on the test dataset.

5.1 Final CNN Model

Class	Precision	Recall	F1-Score	Support
0	0.74	0.83	0.79	1000
1	0.87	0.87	0.87	1000
2	0.68	0.58	0.57	1000
3	0.60	0.60	0.60	1000
4	0.66	0.71	0.68	1000
5	0.66	0.65	0.65	1000
6	0.66	0.65	0.66	1000
7	0.78	0.80	0.79	1000
8	0.80	0.82	0.85	1000
9	0.83	0.84	0.83	1000

Table 1: Classification Report for Each Class

Metric	Score
Accuracy	0.7505
Precision	0.7489731708123746
Recall	0.7505
F1 Score	0.748456089099856
ROC AUC Score	0.9660061222222222

Table 2: Overall Metrics

- Classes 0, 1, and 9 have the highest precision and recall values, indicating that the model can identify these classes with relatively high confidence and consistency.
- Class 2 and Class 5 show relatively lower F1-scores (0.57 and 0.65, respectively), with Class 2 having the lowest recall (0.54). This suggests that the model struggles to correctly identify samples of Class 2, leading to more false negatives.

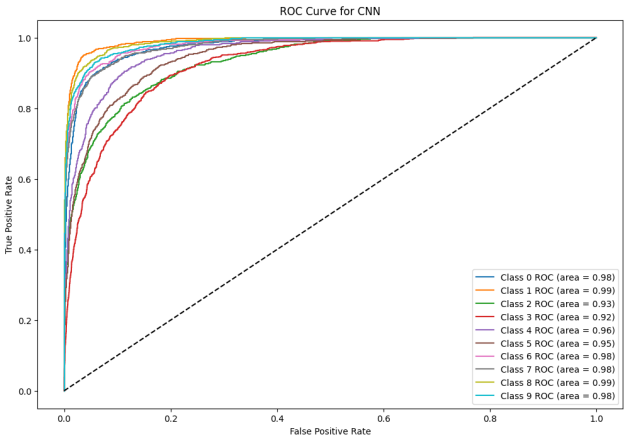

Figure 8: *ROC for Final CNN Model*

This CNN model performs very well across all classes, with AUC values above 0.9 for each class. Minor improvements could be considered for classes with slightly lower AUC values, such as Class 2 and Class 3.

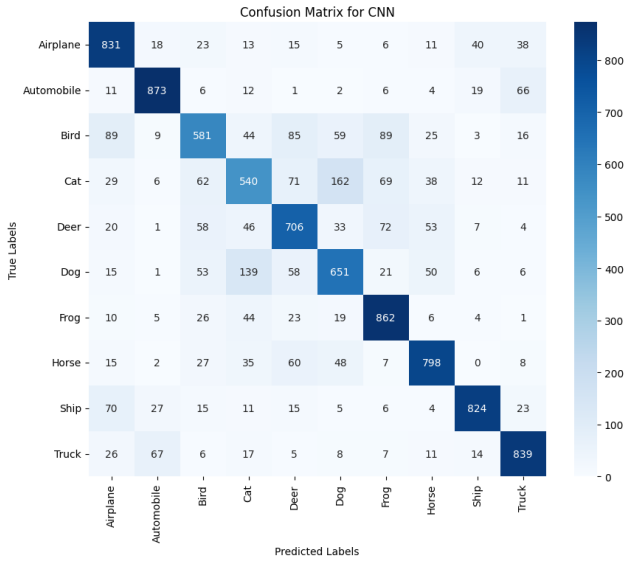

Figure 9: *Confusion Matrix for Final CNN Model*

Classification accuracy of the CIFAR-10 CNN model is quite high and the model recognizes the different classes with high true positives, especially for class "Automobile", "Frog", "Horse" and class "Truck". Nevertheless, there are still misclassifications retained, particularly if classes are visually similar. For example, some clients refer to it as "Bird," wrongly combined with the term "Airplane" while others, term it "Deer" confusing it with "Cat" and "Dog." Furthermore, there is barely some confusion between "Truck" and "Automobile" probably because they bear some similarities in their visual feature. In general, the CNN model is good enough but the issue in feature extraction may be improved make distinction of similar classes better.

5.2 Final ResNet-18 Model

Class	Precision	Recall	F1-Score	Support
0	0.45	0.23	0.30	1000
1	0.32	0.78	0.45	1000
2	0.51	0.26	0.34	1000
3	0.27	0.11	0.16	1000
4	0.21	0.37	0.27	1000
5	0.41	0.34	0.37	1000
6	0.43	0.45	0.44	1000
7	0.36	0.28	0.33	1000
8	0.36	0.64	0.46	1000
9	0.44	0.16	0.23	1000

Table 3: ResNet-18 Classification Report

The selected ResNet-18 is not very effective on this classification task with the efficiency of only 36.38%. Precision, recall, and the F1 score values lie roughly between 0.39 and 0.36 for class 1 and class 0, respectively, which means that the accuracy of the model is also low and it has low interclass discrimination capacity. Such being the case, what the low balanced precision and recall indicate is a relative inability of the model to navigate between the classes, but the ROC

Metric	Value
Accuracy	0.3638
Precision	0.3895
Recall	0.3640
F1 Score	0.3221
ROC AUC Score	0.8293

Table 4: Overall Metrics for ResNet-18

AUC of 0.8293 means on a large scale, it still has a fairly ability of class discriminant capability.

In fact, the classification report analysis shows a rather high fluctuation in values depending on the class chosen. Recall of 0.78 and the F1 score of 0.45 above the average also show that the model has a higher reliability of identifying Class 1. On the other hand, classes 3, 4, and 9 receive noticeably low Recall and F1 scores, which signifies the difficulties for correctly classifying these classes based on the used features and model complexity to work for these classes.

Comparing the average results to the weighted average results gives us an understanding of the distribution of the data among the classes and again, low accuracy is observed. Given from this analysis it can be inferred that possibly more complex feature extraction, better tunable hyperparameters or maybe a different model is more appropriate for stylometric analysis to be incorporated in ResNet-18 for better outcomes to be realized.

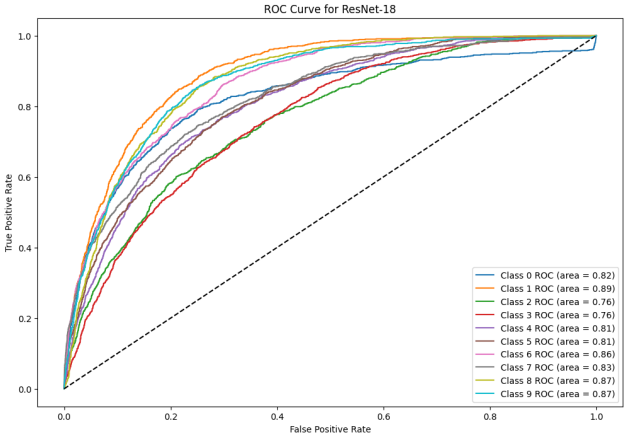

Figure 10: *ROC for Final ResNet18 Model*

Overall, the AUC scores show that while the ResNet-18 model achieves reasonable separability for several classes, particularly Class 1, it struggles with Classes 2 and 3. This might point to a need for additional feature engineering or adjustments in model architecture to improve class separability for challenging classes.

5.3 Final AlexNet Model

The CIFAR-10 data has a fair classification utilizing an accuracy and F1 score of approximately 63%, meaning AlexNet will properly classify about 63% of the available samples. Even though the ROC AUC of the

Metric	Value
Accuracy	0.6279
Precision	0.6290
Recall	0.6279
F1 Score	0.6218
ROC AUC Score	0.9389

Table 5: AlexNet Overall Metrics

Class	Precision	Recall	F1-Score	Support
0	0.74	0.62	0.67	1000
1	0.73	0.76	0.74	1000
2	0.57	0.47	0.51	1000
3	0.47	0.38	0.42	1000
4	0.63	0.64	0.64	1000
5	0.51	0.60	0.55	1000
6	0.65	0.65	0.65	1000
7	0.57	0.78	0.66	1000
8	0.64	0.77	0.70	1000
9	0.64	0.79	0.71	1000

Table 6: AlexNet Classification Report

high 93.89%, this tells the tale that the model has general ability to separate the different classes though still needs to improve on the precision or recall to give accurate classification.

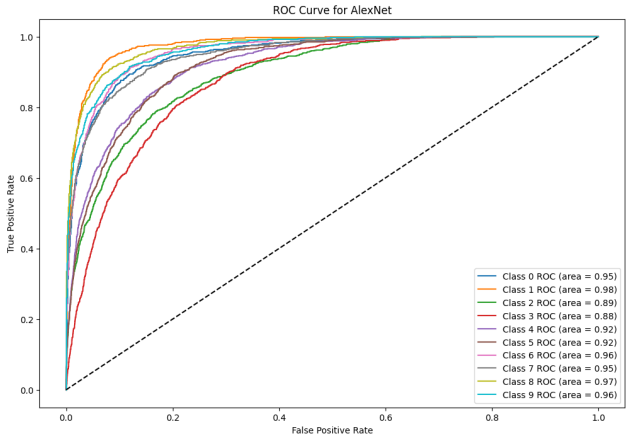

Figure 11: *ROC for Final AlexNet Model*

The ROC analysis for AlexNet model on CIFAR-10 dataset discovered that model has good level of classification on class separation with AUC score between 0.88 to 0.98. Similarly to the previous evaluation CoLa has high AUC values for Classes 1 and 7 (0.98 and 0,97 respectively) which shows that the model is the most accurate at the differentiation of these classes. Classes 2 and 3 are classified with the lowest AUCs 0.89 and 0.88 meaning that the classifier has slight difficulty in giving them some degree of separability from other belonging classes. In general, the high average AUC values, which are above 0.90 in majority of cases, evidence decent exercising of the classification ability of AlexNet even though accuracy was still potential to enhance.

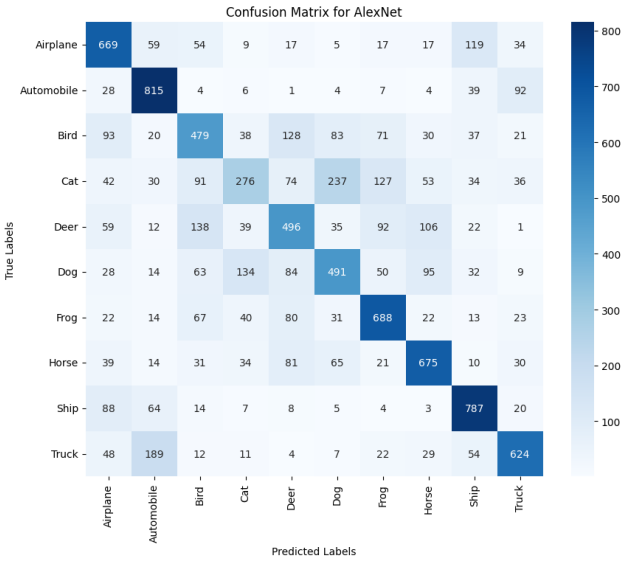

Figure 12: *Confusion Matrix for Final AlexNet Model*

The performance of the AlexNet confusion matrix on CIFAR-10 was relatively good for "Automobile, "Frog, " and "Ship, " and the model performed poorly on the visually similar classes of objects. Some of the confusion with classification is fairly amusing, such as "Bird" being misidentified as an "Airplane" and "Deer" being misidentified as a "Car;" "Cat" being misidentified as a "Dog, and "Truck" being misidentified as an "Automobile". In general, AlexNet is good at recognizing different classes of objects with good confidence but has low confidence distinguishing such classes and items in the same group.

5.4 Final MobileNet Model

Metric	Value
Accuracy	0.6
Precision	0.5915
Recall	0.6
F1 Score	0.5916
ROC AUC Score	0.9236

Table 7: MobileNet Overall Metrics

Class	Precision	Recall	F1-Score	Support
0	0.60	0.67	0.63	1000
1	0.66	0.81	0.73	1000
2	0.48	0.48	0.48	1000
3	0.45	0.28	0.35	1000
4	0.56	0.58	0.57	1000
5	0.51	0.49	0.50	1000
6	0.55	0.60	0.57	1000
7	0.55	0.68	0.61	1000
8	0.59	0.79	0.68	1000
9	0.70	0.62	0.66	1000

Table 8: MobileNet Classification Report

It remains at 60 % on CIFAR-10, similarly for bal-

anced precision/Recall 59/60 % depicting moderate classification accuracy. Parity of AUC ROC is 0.9236 which represents that there is good measure of separable between the two classes. Class 1 can be identified most accurately while class 3 has many problems which has low recall. In general, MobileNet appears to work fairly well at classifying between the presented classes, but it may be necessary to train it more intensively in the model in order to enhance its efficiency.

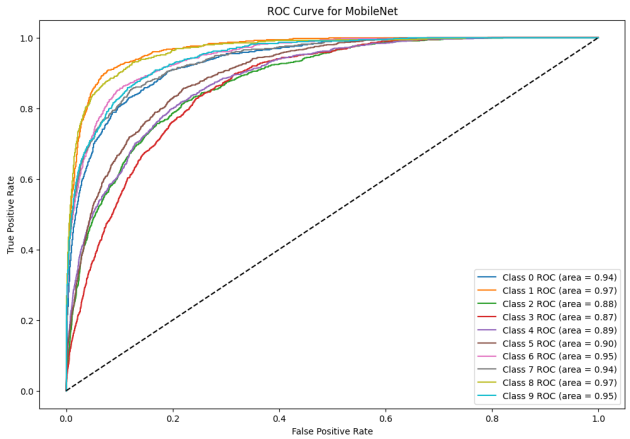

Figure 13: *Confusion Matrix for Final MobileNet Model*

ROC curve of MobileNet on CIFAR-10 is presented in the figure below, reveals a high class discrimination capability by having an AUC value more than 0.90 for most classes. The receivers show the highest AUC scores of 0.97 which indicates that classes 1 and 8 have good discrimination from other classes. It is found out that for Classes 2 and 3 has the lower AUC values of 0.88 and 0.87 respectively, hence are more complex for the model to distinguish. In general, high the value of AUC substantiate the fact that MobileNet has a sufficient ability to separate classes even though the accuracy could be better.

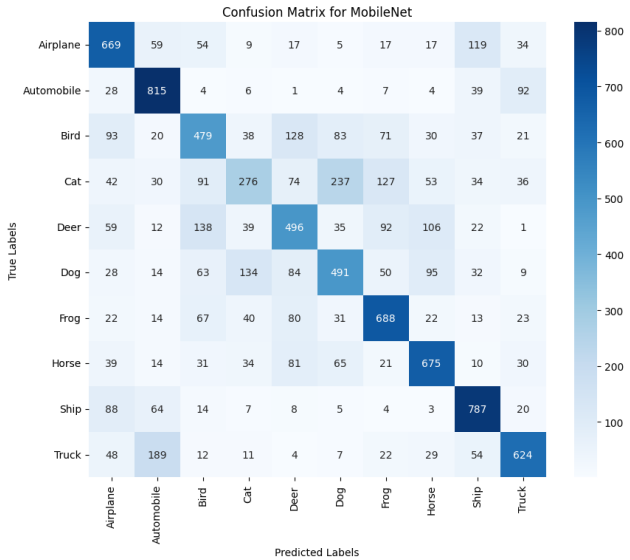

Figure 14: *Confusion Matrix for Final MobileNet Model*

The following confusion matrix we derived from the test of MobileNet on CIFAR-10 is shown in the table below: As you can see in the confusion matrix, the model has high true positives for classes "Automobile", "Frog", and "Ship" implying a good representation and recognition of these unique classes. However, the evaluated algorithms demonstrate better identification performance in visually similar classes, for instance, "Bird" is often mistaken for "Airplane," and "Deer," as "Cat," and "Cat" as "Dog." Also, "Truck" tends to be confused with "Automobile" which has similar appearance. By and large, MobileNet has a good success rate when it comes to classifying established classes but it fails to provide precise distinction between classes that lie close to each other which may imply that MobileNet still requires improved feature extraction for high performance in difficult categories.

6 Final Model After Evaluation

The proposed CNN model has great efficiency for CIFAR-10 data set and thus appropriate for this classification task. Identifying 75.05% of the images, it performs rather well for a difficult data set, CIFAR-10, classifying three-quarters of the images. Precision, recall, and F1 score of about 0 75 also reveals balanced classification on classes where there is equally good ability to find instances and the ability to accurately identify all actual positives without too much compromise.

The higher ROC AUC of 0.9660 for the present model also strengthens the discriminant accuracy of the model, as the values closer to the 1 are better discriminatory power of classes. By using AUC as the parameter, the ROC curve for each class shows the figures more than 0.90, while the "Automobile" and "Frog" classes have the highest figure of 0.99, meaning that the two classes are separated from the rest with better form.

Pointing to a high true positive coefficient, confusion matrix shows that the model offers high recognition of classes like "Automobile", "Frog", "Horse" , "Truck". Although some distinctions between similar classes are often misjudged, these errors are relatively small: for instance, "Bird" is sometimes confused with "Airplane" and "Cat" with "Dog".

A comparison of metrics indicates that the proposed CNN model exemplifies high accuracy, recall, and precision, while maintaining high AUC scores and low misclassification rates for most classes and similar images.

7 GitHub Link for access to Python Code

GitHub Repository for CIFAR-10 Analysis

8 References

- Giuste, F. O., Vizcarra, J. C. (2020). CIFAR-10 Image Classification Using Feature Ensembles.

Retrieved from https://arxiv.org/abs/2002.03846

- Abouelnaga, Y., Mao, Y. (2016). CIFAR-10: KNN-based Ensemble of Classifiers. Retrieved from https://arxiv.org/abs/1611.04905
- Sikdar, A., Sharma, A., Nath, S. (2022). Fully Complex-Valued Deep Learning Model for Visual Perception. Retrieved from https://arxiv.org/abs/2212.07146
- Gowdra, N., Mathew, S. (2021). Examining and Mitigating Kernel Saturation in Convolutional Neural Networks Using Negative Images. Retrieved from https://arxiv.org/abs/2105.04128
- Bird, J. J., Lotfi, A. (2023). CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images. Retrieved from https://arxiv.org/abs/2303.14126