CIFAR-10 Image Classifier

Justin Sciortino 40247931

COMP 472 Sect F

Introduction

This project focuses on developing and evaluating an image classifier for the CIFAR-10 dataset which consists of images of ten classes. The project involved training and testing four models: Naïve Bayes, Decision Tree, Multiplayer Perceptron (MLP), and Convolutional Neural Network (CNN): VGG11.

For the Naïve Bayes, Decision Tree and the MLP models, pre-processing involved normalizing the dataset images and resizing them from 32x32x3 to 224x224x3. Additionally, only a subset of 500 training images and 100 testing images per class in the dataset were selected for training and evaluation. For the three mentioned models above, feature extraction was conducted using a pre-trained ResNet-18 Convolutional Neural Network (CNN) by removing its last layer, resulting in 512 features per image. The Naïve Bayes and Decision Tree models were evaluated both on the full 512 features extracted and a reduced set of 50 features obtained from Principal Component Analysis (PCA), while the MLP model was only evaluated on the full 512 features extracted. On the other hand, the Convolutional Neural Network (CNN) model was trained and evaluated on the original dataset itself without any pre-processing besides normalization.

The report is structured into five sections: the first four sections detail the architecture, training process and performance of each model with comparisons to Scikit-learn implementations where applicable. The final section provides an evaluation of all the four models, comparing the performance between the models.

Naïve Bayes

The Naïve Bayes model which is a probabilistic classifier based on Bayes theorem, was used to predict the class label given a sample from the dataset. During the model's fit phase, the mean and variance for each feature within each class were calculated. To avoid division by zero, a regularization term of $1e^{-9}$ was added to the variance. Additionally, the class priors were derived based on the proportion of samples in the class. For each sample in the dataset subset, the posterior probabilities were calculated for each class, and the class with the highest posterior probability was selected as the predicted label.

As was outlined in the introduction, the model was evaluated with and without PCA reduction, using 50 and 512 features respectively. Table 1 shows the confusion matrix that was generated to visualize the classification performance of the custom model implementation with PCA reduction. Table 2 shows the confusion matrix that was generated for the custom model without PCA reduction. Similarly, Tables 3 and 4, show the generated confusion matrices for the Scikit-learn implementation of the Gaussian Naïve Bayes models with and without PCA reduction.

Table 1: Confusion matrix for the custom implementation of the Naive Bayes model without PCA reduction

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	77	2	2	2	0	0	1	0	11	5
Automobile	2	91	0	2	0	0	0	0	1	4
Bird	5	0	64	4	9	6	11	1	0	0
Cat	1	0	4	74	1	9	9	2	0	0
Deer	2	0	2	6	81	1	1	7	0	0
Dog	0	0	4	16	2	74	2	1	1	0
Frog	1	0	2	6	6	1	84	0	0	0
Horse	2	1	1	8	6	3	0	77	0	2
Ship	6	1	0	0	0	0	1	0	89	3
Truck	3	3	0	1	0	0	0	1	1	91

Table 2: Confusion matrix for the custom implementation of the Naive Bayes model with PCA reduction

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	23	0	23	0	36	1	2	13	2	0
Automobile	0	0	21	40	11	9	14	4	1	0
Bird	31	21	9	4	3	7	1	10	12	2
Cat	10	12	3	20	1	19	4	4	9	18
Deer	21	6	3	8	2	3	0	15	34	8
Dog	5	11	12	24	2	29	0	4	10	3
Frog	39	16	15	2	0	1	3	0	10	14
Horse	15	2	5	4	6	15	0	40	4	9
Ship	8	0	17	1	50	3	16	3	2	0
Truck	1	0	12	17	24	4	33	9	0	0

Table 3: Confusion matrix for the Scikit-learn implementation of the Naive Bayes model without PCA reduction

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	77	2	2	2	0	0	1	0	11	5
Automobile	2	91	0	2	0	0	0	0	1	4
Bird	5	0	64	4	9	6	11	1	0	0
Cat	1	0	4	74	1	9	9	2	0	0
Deer	2	0	2	6	81	1	1	7	0	0
Dog	0	0	4	16	2	74	2	1	1	0
Frog	1	0	2	6	6	1	84	0	0	0
Horse	2	1	1	8	6	3	0	77	0	2
Ship	6	1	0	0	0	0	1	0	89	3
Truck	3	3	0	1	0	0	0	1	1	91

Table 4: Confusion matrix for the Scikit-learn implementation of the Naive Bayes model with PCA reduction

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	23	0	23	0	36	1	2	13	2	0
Automobile	0	0	21	40	11	9	14	4	1	0
Bird	31	21	9	4	3	7	1	10	12	2
Cat	10	12	3	20	1	19	4	4	9	18
Deer	21	6	3	8	2	3	0	15	34	8
Dog	5	11	12	24	2	29	0	4	10	3
Frog	39	16	15	2	0	1	3	0	10	14
Horse	15	2	5	4	6	15	0	40	4	9
Ship	8	0	17	1	50	3	16	3	2	0
Truck	1	0	12	17	24	4	33	9	0	0

Table 5: Naïve Bayes model metrics

Model	Accuracy	Precision	Recall	F1 Measure
Custom Naïve Bayes with PCA	12.80%	11.83%	12.80%	12.19%
Custom Naïve Bayes without PCA	80.20%	80.64%	80.20%	80.20%
Scikit Gaussian Naïve Bayes with PCA	12.80%	11.83%	12.80%	12.19%
Scikit Gaussian Naïve Bayes without PCA	80.20%	80.64%	80.20%	80.20%

The results shown in tables 1-5, reveal that the custom implementation of the Naïve Bayes algorithm yielded the same results as the Scikit-learn's Gaussian Naïve Bayes classifier. Furthermore, the implementations performed better when using 512 features compared to the reduced 50 features as is shown in Table 5. This difference in performance could be attributed to the PCA reduction discarding information leading to poorer separation between classes.

Analysis of the confusion matrices show that the Bird class was the most frequently misclassified. The Bird class was often misclassified as Deer or Frog, and this could be attributed due to similarity in features extracted for these classes, such as sharing the same background environments (i.e., trees), making these classes harder to distinguish. In addition, the Dog and Cat classes also showed misclassification with one another which can also be speculated to be because of their similar appearances and overlapping features.

On the other hand, the Truck and Automobile classes were the most recognized and it could be speculated that they were the most recognized due to their distinct features such as having edges, shapes and textures making them easier to differentiate among the other classes. The Ship class was also well recognized, and this could also be attributed to the same distinct features of the Truck and Automobile classes in addition of its unique background context (i.e., water and ocean) that sets it apart from the other classes.

Decision Tree

The Decision Tree model is a non-parametric supervised learning model and can grow based on the training data it receives. During pre-processing, the dataset was normalized to ensure consistent scaling of features, similar to the approach used for the Naïve Bayes model. This step helped to standardize feature ranges, ensuring fair comparisons when evaluating split criteria.

In the implementation, the dataset was recursively split into subsets based on the feature threshold. This aimed to minimize the impurity at each split and the Gini impurity was employed to measure the quality of splits. The tree was built by selecting the feature threshold that resulted in the lowest Gini impurity at each node. The data continued to split until either two conditions occurred: either the maximum depth of the tree was reached or there were no further splits that could improve the classification.

As was outlined in the introduction, the model was evaluated with and without PCA reduction. Tables 6-9 show the generated confusion matrices for the model without PCA reduction where the maximum depth of the decision tree differed between 50, 15, 10. Table 10, shows the generated confusion matrix for the model with PCA reduction and a maximum depth of 50. Tables 11 and 12 show the corresponding confusion matrices for the Scikit-learn Decision Tree implementation, evaluated under identical conditions as the custom implementation (i.e., with and without PCA reduction).

It should be noted that although some of the models had a maximum depth of 50, following the project guidelines and the amount of data to be used to train the models, the depth was never reached in both the custom implementation of the Decision Tree and the Scikit-learn implementation of the Decision Tree. Therefore, the custom implementation was not evaluated with maximum depths above 50 as it would give identical metrics as the model with a maximum depth of 50.

Table 6: Confusion matrix for the custom implementation of the Decision Tree model without PCA reduction and a maximum depth of 50

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	50	3	9	3	5	1	6	2	11	10
Automobile	5	68	1	1	2	0	3	1	8	11
Bird	9	2	45	6	11	6	12	7	1	1
Cat	4	1	8	32	6	22	12	10	3	2
Deer	4	0	15	7	48	2	7	14	3	0
Dog	1	3	5	17	5	54	4	5	4	2
Frog	4	2	8	11	7	7	56	2	2	1
Horse	5	1	7	8	14	4	2	56	1	2
Ship	11	2	5	2	1	2	2	3	68	4
Truck	5	14	0	1	0	1	2	2	6	69

Table 7: Confusion matrix for the custom implementation of the Decision Tree model without PCA reduction and a maximum depth of 15

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	51	4	9	2	4	1	4	5	10	10
Automobile	6	68	1	2	2	0	2	2	8	9
Bird	13	2	44	5	11	5	11	7	1	1
Cat	6	0	9	37	6	19	9	10	2	2
Deer	7	0	15	6	48	1	5	16	2	0
Dog	2	4	5	19	5	53	4	6	1	1
Frog	4	1	8	11	7	7	56	3	2	1
Horse	6	2	6	8	11	5	2	57	1	2
Ship	10	3	5	2	1	2	2	5	67	3
Truck	5	13	0	1	1	1	2	2	6	69

Table 8: Confusion matrix for the custom implementation of the Decision Tree model without PCA reduction and a maximum depth of 10

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	57	5	4	5	1	0	3	3	15	7
Automobile	10	71	0	3	1	0	0	1	7	7
Bird	11	1	46	5	14	11	9	2	0	1
Cat	2	4	11	47	3	16	9	5	2	1
Deer	9	1	15	5	50	3	5	11	1	0
Dog	2	1	4	21	5	55	3	9	0	0
Frog	8	1	11	9	7	4	55	2	3	0
Horse	8	4	7	11	8	3	2	56	1	0
Ship	12	4	4	2	0	1	1	2	72	2
Truck	7	14	0	1	0	1	1	0	4	72

Table 9: Confusion matrix for the custom implementation of the Decision Tree model without PCA reduction and a maximum depth of 5

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	37	4	3	6	5	0	4	0	39	2
Automobile	1	59	0	4	0	0	2	0	33	1
Bird	5	1	25	17	14	1	32	3	2	0
Cat	0	0	0	54	12	3	14	2	15	0
Deer	1	0	3	20	52	0	12	7	5	0
Dog	0	0	2	43	12	27	7	3	6	0
Frog	0	2	1	15	6	1	65	2	7	1
Horse	0	1	2	15	21	0	2	41	18	0
Ship	2	4	0	6	1	1	1	0	83	2
Truck	2	28	0	4	1	0	0	0	16	49

Table 10: Confusion matrix for the custom implementation of the Decision Tree model with PCA reduction and a maximum depth of 50

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	15	1	34	0	24	2	1	12	10	1
Automobile	5	0	14	31	8	28	7	6	0	1
Bird	20	28	2	2	4	0	1	8	20	15
Cat	2	21	5	12	3	15	3	3	11	25
Deer	17	5	3	8	1	3	2	15	32	14
Dog	5	26	5	12	3	20	4	7	9	9
Frog	18	17	2	3	2	0	4	3	30	21
Horse	7	5	7	6	6	4	1	45	7	12
Ship	10	0	17	3	47	4	9	6	3	1
Truck	6	0	18	25	6	14	24	5	2	0

Table 11: Confusion matrix for the Scikit-learn implementation of the Decision Tree model without PCA reduction and a maximum depth of 50

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	56	3	5	3	3	1	2	1	15	11
Automobile	5	73	2	3	0	0	3	3	4	7
Bird	7	2	50	3	7	9	11	6	5	0
Cat	0	4	6	47	7	14	10	8	2	2
Deer	2	1	9	6	46	10	4	17	1	4
Dog	3	3	11	16	3	47	4	7	5	2
Frog	3	3	7	9	4	3	65	4	2	0
Horse	2	3	7	7	14	9	1	53	1	3
Ship	5	3	5	4	1	1	1	0	71	9
Truck	5	25	1	3	2	1	0	3	3	57

Table 12: Confusion matrix for the Scikit-learn implementation of the Decision Tree model with PCA reduction and a maximum depth of 50

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	5	4	28	6	25	5	3	14	5	5
Automobile	4	0	21	21	8	23	11	10	1	1
Bird	21	32	1	4	4	1	1	6	20	10
Cat	3	24	3	13	4	11	2	2	11	27
Deer	22	11	4	3	4	5	1	5	27	18
Dog	6	25	2	24	2	13	7	6	9	6
Frog	18	24	2	4	1	1	6	2	30	12
Horse	15	8	5	8	6	11	0	20	4	23
Ship	2	2	16	6	50	4	9	10	1	0
Truck	3	0	20	21	7	14	24	8	2	1

Table 13: Decision Tree model metrics

Model	Accuracy	Precision	Recall	F1 Measure
Custom Decision Tree with PCA and max depth 50	10.20%	10.16%	10.20%	10.11%
Custom Decision Tree without PCA and max depth 50	54.60%	54.39%	54.60%	54.46%
Custom Decision Tree without PCA and max depth 15	55.00%	55.14%	55.00%	55.02%
Custom Decision Tree without PCA and max depth 10	58.10%	58.77%	58.10%	58.27%
Custom Decision Tree without PCA and max depth 5	49.20%	60.28%	49.20%	49.19%
Scikit Decision Tree with PCA and max depth 50	6.40%	7.16%	6.40%	6.71%
Scikit Decision Tree without PCA and max depth 50	56.50%	56.39%	56.50%	56.32%
Scikit Decision Tree without PCA and max depth 10	57.40%	58.86%	57.40%	57.68%

The results shown in the tables 6-13 above, reveal that the custom implementation of the Decision Tree yielded different results compared to the Scikit-learn's Decision Tree implementation. Overall, the custom implementation with a maximum depth of ten, without PCA reduction, performed the best among the five configurations of the custom implementation and among the three configurations of the Scikit-learn's implementations. Furthermore, both

implementations performed better when using 512 features compared to the reduced 50 features as is shown in Table 13.

Examining Table 8, which shows the confusion matrix generated from the custom implementation with a maximum depth of ten, the Bird and Cat classes were the most frequently misclassified. The Bird class was often misclassified as Airplane, Deer, or Dog. Misclassification as Airplane could be attributed to similarities in features extracted, such as sharing the same background environments (i.e., skies), while confusion with Dog and Deer might result from overlapping features or similarities in appearance. Similarly, the Cat class was often misclassified as Dog, likely due to these classes having overlapping features and similarities in appearances. On the other hand, the Automobile, Ship and Truck classes were the least frequently misclassified. As noted in the Naïve Bayes analysis, this could be attributed to the classes having distinct features such as having edges, shapes and textures making them easier to differentiate among the other classes.

As is shown in Table 13 of the metrics and the confusion matrices in Tables 6-9, decreasing the maximum depth improved the learning, generalization, and overall performance up to a certain point. At shallower depths such as maximum depth of five, the custom decision tree had lower metrics compared to the other configurations indicating underfitting. With a moderate maximum depth of ten, the custom tree performed the best which indicated a balance between complexity and generalization. In addition, at this depth, the model captured detailed features without overfitting or underfitting. However, with a deeper maximum depth of 50 or 25, performance declined when compared with a moderate maximum depth of ten, suggesting overfitting.

Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron (MLP), a parametric model, is a feedforward neural network that relies on multiple layers of neurons to map input features to the output classes. The model utilized the fully connected layers with non-linear activations functions to capture the relationships in the data.

The implemented MLP architecture consisted of three, two, four or six layers depending on the configuration of the model and the experiment being conducted. The layers in the middle consisted of a combination of Linear transformations, Batch Normalization and ReLU activation functions. The cross-entropy loss function was used for optimization, and the model was trained

using the Stochastic Gradient Descent (SGD) optimizer with momentum of 0.9 to accelerate convergence. During the training process, multiple epochs were used to update the weights of the network to minimize the loss on the training set.

As was specified in the introduction, all trained MLP models were implemented without PCA reduction to maximize the performance. In addition, all trained models were trained with 30 epochs, 512 features, a learning rate of 0.001, a SGD momentum of 0.9, and a batch size of 128. These values were chosen after experiment with different values. The trained models vary in the number of layers as well as the size of the hidden layers which were either 256, 512 or 1024. Tables 14-16 show the generated confusion matrices with a three-layer MLP and varying hidden layer sizes. Table 17 shows the evaluation metrics for each trained MLP model.

Table 14: Confusion matrix for Three-Layer MLP with hidden layer size of 256

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	83	0	2	1	0	0	2	0	8	4
Automobile	3	92	0	1	0	0	0	0	2	2
Bird	6	0	73	6	4	3	6	2	0	0
Cat	0	0	2	79	2	10	3	3	1	0
Deer	1	0	3	4	83	1	1	6	1	0
Dog	0	0	6	14	4	69	2	4	1	0
Frog	1	0	2	4	2	1	90	0	0	0
Horse	0	0	0	3	7	2	0	88	0	0
Ship	4	0	1	0	0	0	0	0	94	1
Truck	1	3	0	0	0	0	0	0	1	95

Table 15: Confusion matrix for Three-Layer MLP with hidden layer size of 512

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	85	1	1	0	0	0	1	0	9	3
Automobile	2	94	0	1	0	0	0	0	0	3
Bird	5	1	76	4	3	6	3	2	0	0
Cat	0	0	4	79	2	11	4	0	0	0
Deer	2	0	4	6	81	1	1	5	0	0
Dog	0	0	5	13	0	79	2	0	1	0
Frog	1	0	2	3	2	4	88	0	0	0
Horse	0	0	0	6	6	2	0	86	0	0
Ship	4	0	1	1	0	0	0	0	94	0
Truck	3	3	0	2	0	0	0	0	0	92

Table 16: Confusion matrix for Three-Layer MLP with hidden layer size of 1024

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	83	0	2	1	0	0	1	0	9	4
Automobile	3	94	0	1	0	0	0	0	0	2
Bird	6	0	73	7	2	3	6	2	1	0
Cat	0	0	2	79	2	11	3	3	0	0
Deer	1	0	2	6	84	0	0	6	1	0
Dog	0	0	5	16	1	75	2	0	1	0
Frog	1	0	1	4	2	1	90	0	1	0
Horse	0	0	0	4	5	1	0	90	0	0
Ship	4	0	2	0	0	0	0	0	94	0
Truck	2	4	0	1	0	0	0	0	1	92

Table 17: Multi-Layer Perceptron evaluation metrics

Size of hidden layers	Number of layers	Accuracy	Precision	Recall	F1 Measure	Training time (sec)	Evaluation time (sec)	GPU memory allocated (MB)
256	3	84.60%	84.70%	84.60%	84.53%	2.92	0.007	51.14
	2	84.30%	84.47%	84.30%	84.30%	2.61	0.005	50.35
	4	84.30%	84.66%	84.30%	84.37%	3.87	0.008	51.78
	6	82.60%	83.28%	82.60%	82.42%	4.02	0.011	53.38
512	3	85.40%	85.72%	85.40%	85.49%	3.00	0.008	54.96
	2	85.10%	85.23%	85.10%	85.03%	2.85	0.006	51.95
	4	85.30%	85.65%	85.30%	85.32%	3.67	0.010	58.13
	6	84.70%	85.10%	84.70%	84.77%	4.77	0.016	64.71
1024	3	85.40%	85.74%	85.40%	85.44%	3.06	0.013	67.88
	2	85.00%	85.34%	85.00%	85.06%	3.15	0.008	55.17
	4	84.40%	84.66%	84.40%	84.42%	3.66	0.019	80.51
	6	82.80%	83.36%	82.80%	82.83%	4.75	0.029	106.15

The results in Tables 14-16 indicate that Bird, Cat, and Dog classes were the most frequently misclassified. The Bird class was most often misclassified as Airplane or Dog. Misclassification as Airplane is likely due to both appearing in sky backgrounds, resulting in overlapping features while misclassification as Dog is likely due both classes being in natural environments. The Cat and Dog class were most often misclassified with each other, likely due to their similar appearances and overlapping features. The Automobile, Truck and Ship classes the

least frequently misclassified. This could be attributed to distinct features, such as sharp edges, unique shapes, and textures, which made them easier to differentiate.

As is shown in the Table 17, the number of layers influenced the MLP's ability to balance feature learning and overfitting. The MLP with three layers showed the best metrics while avoiding overfitting, achieving a strong performance at a reasonable computational cost. In contrast, models with four and six layers exhibited slight decreases in metrics, likely due to overfitting.

The hidden layer size also impacted the model's performance and computational cost. As is demonstrated in Table 17, a size of 512 for the hidden layers demonstrated the best performance to cost ratio, achieving strong metrics with a moderate computational demand. When the hidden layer size was 256, the computational demands were lower, however this came at a cost of having slightly lower metrics. Finally, when the hidden layer size was 1024, there was an increase in computational demands (GPU memory, training time) with only marginal improvements in metrics compared to hidden layer size of 512. This model configuration would only be viable if the slight performance gains justified the additional computational costs.

In conclusion, an MLP with three layers and hidden layers of size 512 achieved the best balance between performance and computational efficiency among all tested configurations of the MLP model. This model offered the optimal combination of accuracy, generalization, and manageable computational resources.

Convolutional Neural Network (CNN): VGG11

The VGG11 is a convolutional neural network containing five convolution-batch norm-pooling layers and three fully connected layers. Batch normalization is incorporated to stabilize training and accelerate convergence. Max-pooling layers are also used to down sample feature maps while preserving important features. Each convolutional layer is also followed up by a ReLU activation function to ensure non-linearity. During training of the VGG11 models, the cross-entropy loss, and Stochastic Gradient Descent (SGD) optimizer with momentum of 0.9 were used. Besides the dataset being normalized, there was no other pre-processing performed.

The implemented VGG11 model consisted of six, eight or ten convolutional layers and either a kernel size of three, five or seven, depending on the model configuration and experimental setup. For each model configuration, the learning rate was set to 0.01 and the number of epochs to 20. The model's performance was evaluated across different VGG11 configurations to analyze the

trade-offs between computational cost and classification metrics. Tables 18-20 show the generate confusion matrices for three of the configurations of the VGG11 model. Table 21 shows the evaluation metrics for each trained VGG11 model.

Table 18: Confusion matrix for VGG11 with a kernel size of 3x3 and six convolutional layers

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	855	7	20	14	7	4	8	10	17	28
Automobile	12	874	0	1	1	1	6	4	4	97
Bird	37	3	779	33	41	28	48	23	3	5
Cat	17	5	52	676	35	89	70	37	5	14
Deer	11	1	44	36	808	21	27	48	3	1
Dog	6	5	26	137	27	730	18	42	2	7
Frog	1	3	17	19	10	8	933	1	2	6
Horse	7	0	9	12	22	18	7	914	1	10
Ship	78	19	6	11	0	2	5	5	828	46
Truck	18	26	5	2	2	1	3	5	5	933

Table 19: Confusion matrix for VGG11 with a kernel size of 5x5 and ten convolutional layers

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	884	3	23	7	12	6	4	16	26	19
Automobile	13	880	2	5	1	3	9	0	29	58
Bird	46	0	763	53	35	31	44	23	3	2
Cat	13	3	42	703	38	96	45	48	7	5
Deer	11	1	45	46	821	12	20	41	3	0
Dog	7	2	29	143	22	732	16	48	1	0
Frog	3	0	23	36	14	10	908	2	1	3
Horse	3	2	9	22	25	19	2	912	2	4
Ship	36	0	9	10	4	7	5	6	916	7
Truck	20	24	6	15	1	3	6	11	17	897

Table 20: Confusion matrix for VGG11 with a kernel size of 7x7 and six convolutional layers

True Labels	Predicted Labels									
	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	889	10	28	17	8	4	2	6	28	8
Automobile	22	885	0	2	1	6	8	2	26	48
Bird	59	2	779	28	26	40	45	11	10	0
Cat	16	3	68	616	31	168	69	9	11	9
Deer	15	3	67	42	775	37	36	21	4	0
Dog	7	2	38	101	18	773	29	28	1	3
Frog	4	3	20	31	5	16	911	2	5	3
Horse	25	0	22	21	32	71	5	818	1	5
Ship	46	8	10	11	0	3	4	1	903	14
Truck	40	58	2	9	1	5	8	7	20	850

Table 21: Convolutional Neural Network: VGG11 evaluation metrics

Kernel Size	Number of conv layers	Accuracy	Precision	Recall	F1 Measure	Training time (sec)	Evaluation time (sec)	GPU memory allocated (MB)
3x3	6	83.60%	83.75%	83.60%	83.49%	505.07	4.05	256.65
	8	83.00%	83.22%	83.00%	83.04%	561.29	4.16	243.71
	10	82.51%	84.03%	82.51%	82.78%	646.14	4.55	2543.28
5x5	6	82.58%	83.35%	82.58%	82.72%	823.19	4.75	318.93
	8	80.82%	81.93%	80.82%	81.10%	1017.56	4.32	373.36
	10	84.16%	84.28%	84.16%	84.15%	1251.12	4.94	392.15
7x7	6	80.94%	81.20%	80.94%	80.94%	1364.48	6.07	415.67
	8	80.64%	80.63%	80.64%	80.45%	1809.85	7.41	570.33
	10	81.99%	82.26%	81.99%	81.96%	2368.54	9.86	605.47

Examining Table 19, which shows the confusion matrix generated from the VGG11 with an architecture of ten convolutional layers and a kernel size of 5x5, the Bird, Cat and Dog classes were the most frequently misclassified. The Bird class was often misclassified as Airplane, Cat, or Frog. Misclassification as Airplane and Frog could be attributed to similarities in features extracted, such as sharing the same background environments (i.e. skies, trees), while confusion with Cat might result from overlapping features or similarities in appearance. The Cat and Dog class were most often misclassified with each other, likely due to their similar appearances and overlapping features. On the other hand, the Ship, Horse, and Frog classes were the least frequently misclassified. The Ships class most likely showed minimal confusion due to its distinct features,

as was noted in the MLP analysis. The Frog and Horse classes most likely showed minimal confusion due to their unique textures and features.

The number of convolutional layers in the architecture impacted the performance of the VGG11 model. As is shown in Table 21, increasing the number of convolutional layers improved feature extraction. For instance, configurations with ten convolutional layers generally showed the best metrics in terms of accuracy, precision, recall and F1 score compared to the configurations with six or eight convolutional layers. However, this came at a trade-off of computational cost and training time compared to the configurations with less convolutional layers. For instance, the configuration with ten convolutional layers and a kernel size of 5x5 achieved the highest metrics of all models but required more computational resources such as GPU memory. In all configuration scenarios, the VGG11 model with eight convolutional layers, regardless of the kernel size, struggled to generalize and extract the features the most.

The choice of kernel size played a pivotal role in the performance of the model. A smaller kernel size of 3x3, performed well overall at capturing finer details in the images leading to a better overall performance regardless of the number of convolutional layers in the architecture. In one case, a larger kernel size of 5x5, performed the best out of configurations, achieving the best performance and metrics but at a cost of computational demands. In most cases with a larger kernel size of 5x5 or 7x7, the configurations performed worse than the same configurations with the same number of convolutional layers when the kernel size was 3x3.

Finally, the results shown in Table 21, reveal that in general increasing the number of convolutional layers and using larger kernel sizes such as 7x7 resulted in poorer metrics and may have led to overfitting. The optimal balance between performance and metrics and computational cost was the configuration of ten convolutional layers and a kernel size of 5x5.

Final evaluation of all four models

As is shown in the table below, Table 22, which summarizes the metrics of the best configurations of each of the four models, the Multi-Layer Perceptron with three layers and a hidden size of 512 outperformed the other models in terms of accuracy, precision, recall, F1 measure and GPU memory allocated. This could be attributed to the MLP's ability to effectively capture the relationships in the dataset through its fully connected layers and achieve a good balance between feature learning and overfitting.

The CNN model with ten convolutional layers and a kernel size of 5x5, closely followed behind the MLP model in terms of performance. The high performance of the CNN is indicative of its ability to extract features through its deeper architecture consisting of more convolutional layers and optimized kernel size. However, the MLP model could have performed better than the CNN due to its simpler structure which could have generalized better for the given CIFAR-10 dataset. In addition, although the deeper architecture of the VGG11 model allowed it to extract complex features, the model might have been more susceptible to overfitting or increased complexity due to the model being trained on the complete dataset whereas the MLP was only trained on a subset of the dataset. As well, in terms of cost-to-performance, the MLP achieved a better performance while having a lower computational cost (shown in Table 17) compared to the CNN (shown in Table 21). While the Naïve Bayes model did show respectable performance, this model and the Decision Tree, which showed the worst performance, were both outperformed by the CNN and MLP.

Table 22: Evaluation metrics of the four models without PCA reduction

Model	Accuracy	Precision	Recall	F1 Measure
Custom Naïve Bayes	80.20%	80.64%	80.20%	80.20%
Custom Decision Tree with maximum depth of ten	58.10%	58.77%	58.10%	58.27%
Multi-Layer Perceptron (MLP): Three layers, hidden size of 512	85.40%	85.72%	85.40%	85.49%
Convolutional Neural Network: VGG11 with ten convolutional layers and kernel size of 5x5	84.16%	84.28%	84.16%	84.15%