Classification Performance Using KNN and NC Algorithms on CIFAR-10

Christos Samaras
Graduate Student of Electrical and Computer Engineering Department,
Aristotle University of Thessaloniki
Email: samarasc@ece.auth.gr
AEM: 10486

***Abstract*—The CIFAR-10 [1] dataset is used for the classification of 10 object categories. In this study, the Nearest Neighbor (KNN) algorithm with neighbors K set to 1 and 3 and the Nearest Centroid (NC) algorithm are implemented along with pixels' value normalization and Principal Component Analysis (PCA). The best accuracy achieved on the test set with these methods is 39%. Furthermore, the classification task is approached using neural network leading to the development of SAM12 and its three variations - SAM12a, SAM12b, and SAM12c - as a combination of CNN and MLP subnets. Utilizing normalized pixel values and statistical features of skewness, kurtosis, entropy, and bispectrum, these networks achieve accuracy of approximately 75% on the test samples.**

I. Introduction

The CIFAR-10 dataset is composed of 10 object classes with 6000 32x32 RGB images per class (Fig.1). An RGB image consists of three separate color channels: one for red, one for green and one for blue. Each channel stores intensity values for the respective color. These three channels can get 256 different values that range from 0 to 255. In RGB, R stands for red, G for green, and B for blue which correspond to (255, 0, 0), (0, 255, 0) and (0, 0, 255) respectively.

In an image, pixels' values change across its spatial dimensions. The rate of these changes introduces the concept of frequency into images.

November 24, 2024

II. Method

A. Mathematical Background

1) Skewness: For a random variable x following a distribution, skewness denotes the asymmetry of this distribution relative to a normal distribution. Skewness which derives from the Fisher-Pearson coefficient of skewness is defined as:

$$g_1 = \frac{m_3}{\sqrt{m_2^3}} \tag{1}$$

where m_3 is the third central moment:

$$m_3 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^3 \tag{2}$$

and m_2 is the second central moment:

$$m_2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{3}$$

Fig. 1: CIFAR-10 Classes Visualization

2) Kurtosis: For a random variable x following a distribution, kurtosis measures the sharpness or flatness of this distribution relative to a normal distribution.

$$K = \frac{m_4}{m_2^2} \tag{4}$$

where m_4 is the fourth central moment:

$$m_4 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^4 \tag{5}$$

3) Entropy: Entropy describes the uncertainty and randomness of a distribution.

$$H(X) = -\sum_{i=1}^{N} p(x_i) log_2(p(x_i)) \tag{6}$$

where $p(x_i)$ is the occurrence probability of x_i

4) Bispectrum: In High-Order-Statistics (HOS), n^{th}-order spectrum is defined as the $(n-1)$-DFT (Discrete Fourier Transform) of the n^{th}-order cumulants. In reality, finite size of data hardens the calculation of cumulants. For this reason, Bispectrum $C_3^x(f_1, f_2)$ is calculated as:

$$C_3^x(f_1, f_2) = X(f_1)XK(f_2)X^*(f_1 + f_2) \tag{7}$$

where $X(f)$ is the Fourier Transform of $x(t)$ and $X^*(f)$ denotes the complex conjugate of X. Bispectrum reveals Quadratic Phase Coupling (QPC) phenomena. Two frequency components a and b are quadratically phase coupled when there is a component c for which $f_a + f_b = f_c$ and $\phi_a + \phi_b = \phi_c$, where f and ϕ stand for frequency and phase, respectively.

B. Features Extraction

First of all, pixels are normalized by dividing them by 255. Let's suppose that pixels of each RGB channel follow a random distribution. Skewness, kurtosis and entropy are calculated for these distributions. Besides, RGB images are converted to grayscale using this formula:

$$0.2989 * R + 0.5870 * G + 0.1140 * B \tag{8}$$

For the grayscale images, bispectrum is calculated using direct-method and $bispecd$ function from HOSA Toolbox for MATLAB. Mean value, max value, and number of peaks of bispectrum magnitude are extracted.

III. Classification Algorithms

Two different classification algorithms are used for this study: Nearest-Neighbor (KNN) and Nearest Centroid (NC). These algorithms are implemented with the help of Scikit-Learn open-source machine learning library in Python.

For KNN, a number of 1 and 3 neighbors is selected. In addition, these 2 algorithms are combined with Principal Component Analysis (PCA) in order to reduce dimensionality from 3072 to 100. In RGB images, the pixel values of each color channel range from 0 to 255. For this reason, pixels' values are also normalized by dividing them by 255.

Of the total $60\,000$ images, $50\,000$ of them are used for model training and the rest $10\,000$ for model testing. Classification accuracy is firstly measured for the Nearest Neighbor (for 1 and 3 neighbors) and Nearest Centroid models only, then for KNN and NC with normalization and finally for KNN and NC combined with PCA. In addition, the response time of each model is measured.

IV. Neural Network

This classification task is also approached with the idea of a feedforward Neural Network (NN). Again, of the total 60 000 images, 50 000 of them are used for model training and the rest 10 000 for model testing.

A. Pathway to SAM12

Neural Networks are developed with the help of TensorFlow, an end-to-end open source machine learning platform. Various ideas were implemented until the final model reached. MLP with 1 to 3 layers, with various neurons, combined or not with CNN with various numbers of convolutional layers and different activation functions were implemented. Furthermore, various inputs have been used. For example, considering that the object is displayed in the center of each image, the central pixels (8:24) of each image were used as input. However, the accuracy of these variations was low and the error was high for train and test procedures. This error was often above 2 even during training. Through this process, the final model SAM12 was reached.

B. SAM12

SAM12 final model is a combination of one Convolutional Neural Network (CNN1) and four Multilayer Perceptrons, MLP1, MLP2, MLP3, and MLP4 (Fig.2). The CNN subnet receives as input the normalized pixels' values of RGB images. Skewness, kurtosis, entropy and bispectrum stats are inserted into MLP1, MLP2, MLP3 and MLP4 respectively. The outputs of these five subnets are combined and passed into the final MLP (MLP5). The last layer of MLP5 has $softmax$ as activation function and 10 neurons, as many as the classes of the dataset. This tactic ensures that all probabilities are between 0 and 1 and sum up to 1.

For SAM12, batch size is set to 128 and epochs to 60. Model uses zero padding if necessary. Learning rate is set to 0.01. The AdaMax optimizer and categorical cross-entropy loss function are chosen. In order to use this specific loss function, labels are converted to one-hot encoded vectors.

C. SAM12a, SAM12b, SAM12c

Three variations of SAM12 are created by adjusting some of its parameters, such as the learning rate or the optimizer, as well as the number of hidden layers, and the following three NN emerged: SAM12a, SAM12b, and SAM12c. Again, epochs and batch size are set to 60 and 128 respectively, categorical cross entropy loss function and zero padding are used. The differences between these three models and the original model are described below.

1) SAM12a uses Stochastic Gradient Descent (SGD) optimizer instead of AdaMax with 0.01 learning-rate.
2) SAM12b has 2 more convolutional layers with 128 and 256 filters and 2 more dropouts with $p = 0.4$ in CNN. It also has 2 more dense layers with 32 and 256 neurons and dropout with $p = 0.2$ in MLP5 (Fig.3). It also uses AdaMax with 0.01 learning-rate.
3) SAM12c has one more dropout with $p = 0.4$ in CNN and dropout with $p = 0.2$ in MLP5. This NN network uses SGD optimizer with 0.001 learning rate instead of 0.01 which is used by all other models (Fig.4).

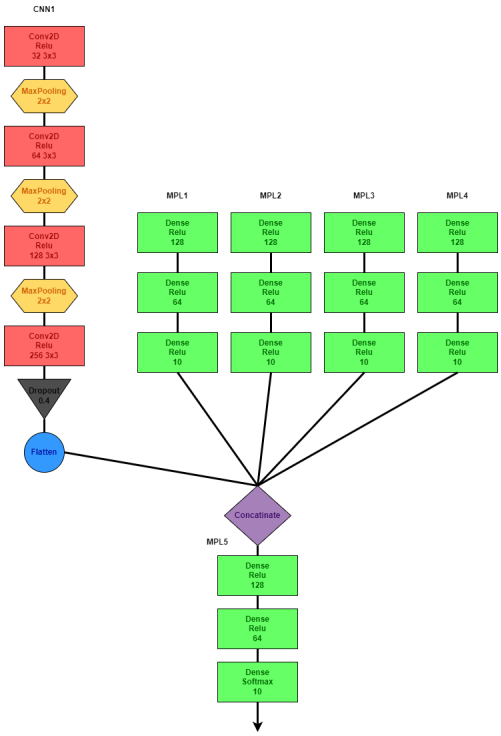

Fig. 2: SAM12 and SAM12a models

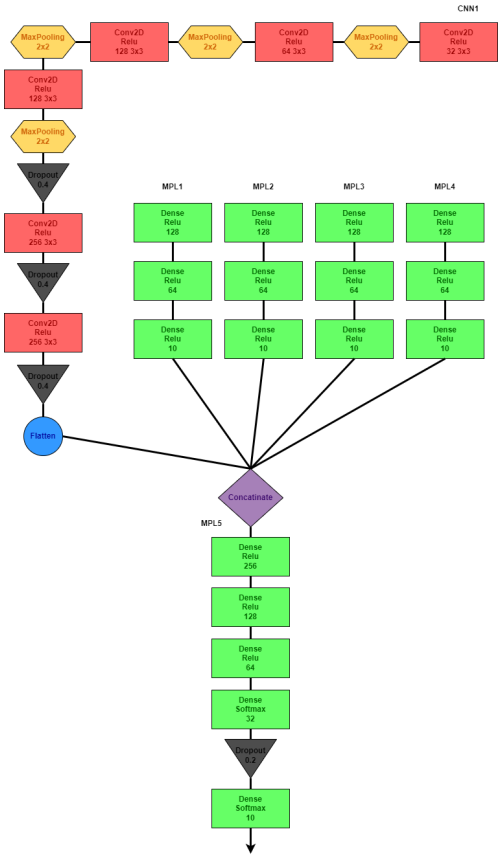

Fig. 3: SAM12b model

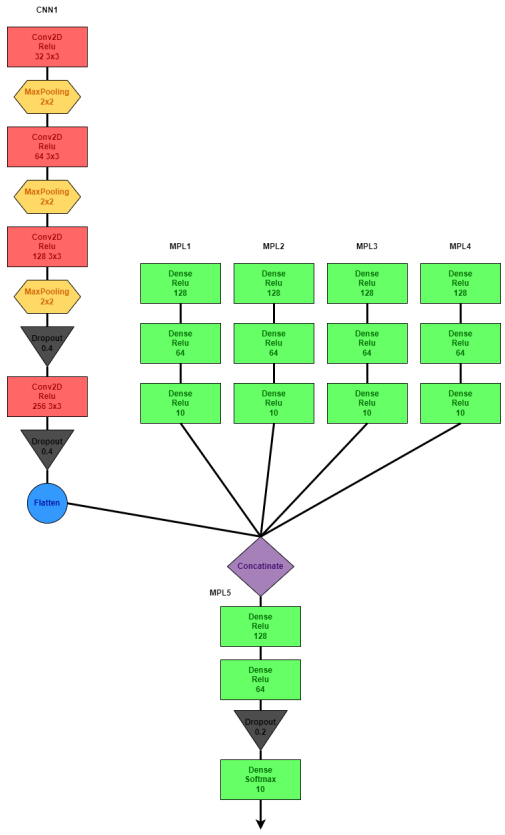

Fig. 4: SAM12c model

V. Conclusion

A. KNN and NC

The classification results are not particularly encouraging for both the KNN and NC models (TABLE I). It can be observed that the former achieves higher accuracy than the latter. With or without normalization, the classification accuracy remains identical for a specific model. This is consistent with the corresponding confusion matrices (Fig.5,6 and Fig.11,12). Using 3 neighbors results in accuracy reduction compared with 1 neighbor. The results are better for KNN when PCA is used (Fig.7,10) but slightly lower for NC (Fig.13).

TABLE I: Aggregate Classification Results

Classifier	combined with	Accuracy (%)
KNN using 1 neighbor	-	35.39
KNN using 1 neighbor	normalization	35.39
KNN using 1 neighbor	PCA	38.56
KNN using 3 neighbors	-	33.03
KNN using 3 neighbors	normalization	33.03
KNN using 3 neighbors	PCA	36.56
Nearest Centroid	-	27.74
Nearest Centroid	normalization	27.74
Nearest Centroid	PCA	27.66
SAM12	normalization, statistics	75.16
SAM12a	normalization, statistics	74.40
SAM12b	normalization, statistics	75.78
SAM12c	normalization, statistics	42.40

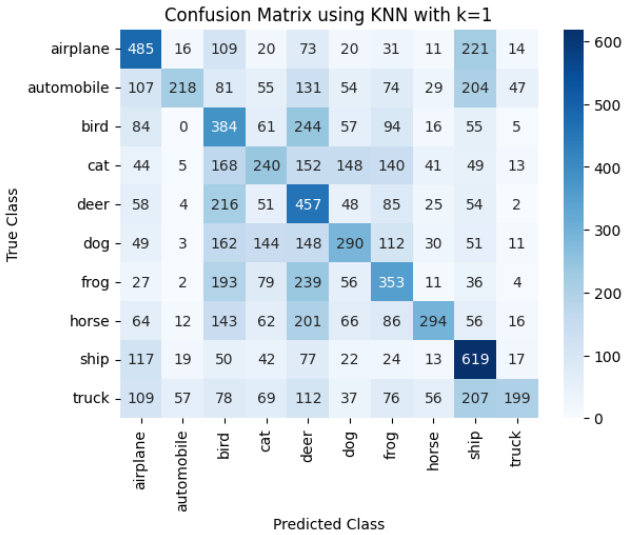

Fig. 5: KNN Confusion Matrix with 1 neighbor

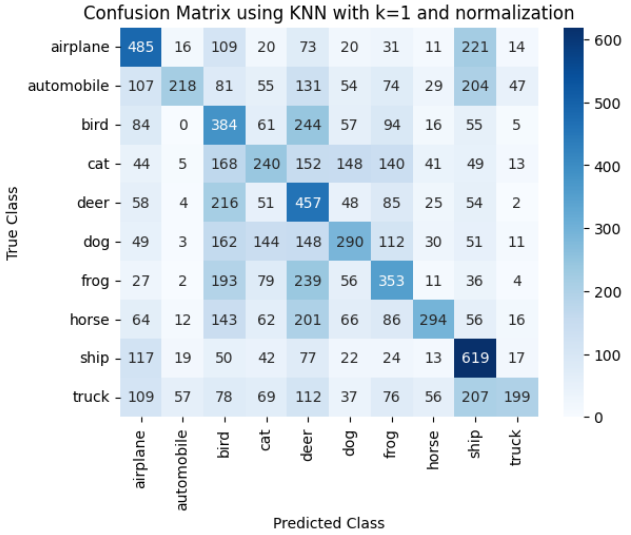

Fig. 6: KNN Confusion Matrix with 1 neighbor and normalization

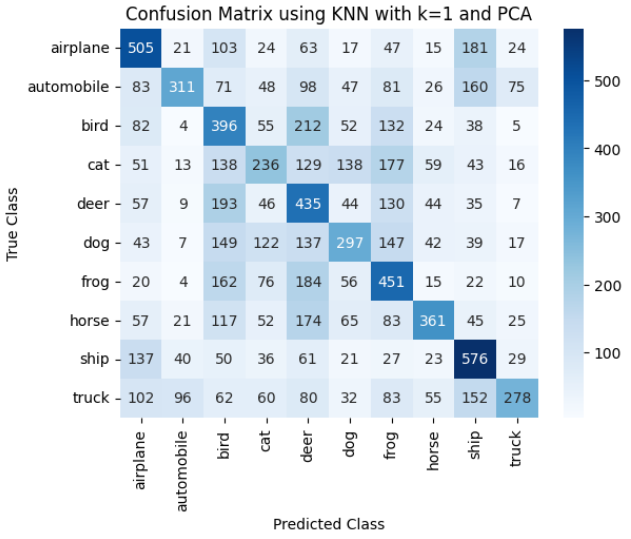

Fig. 7: KNN Confusion Matrix with 1 neighbor and PCA

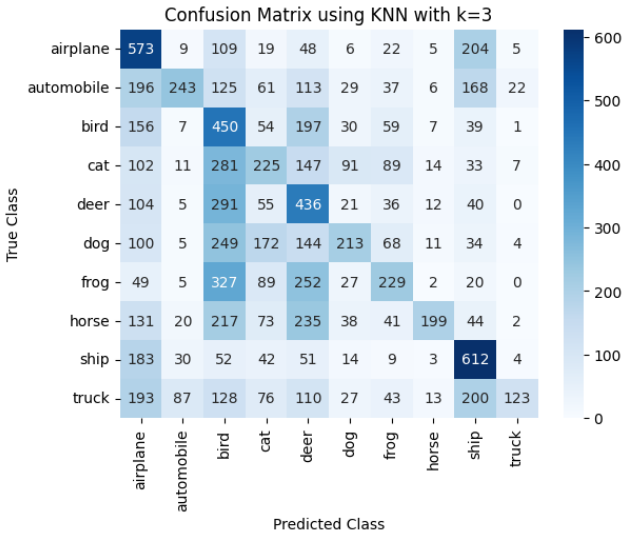

Fig. 8: KNN Confusion Matrix with 3 neighbors

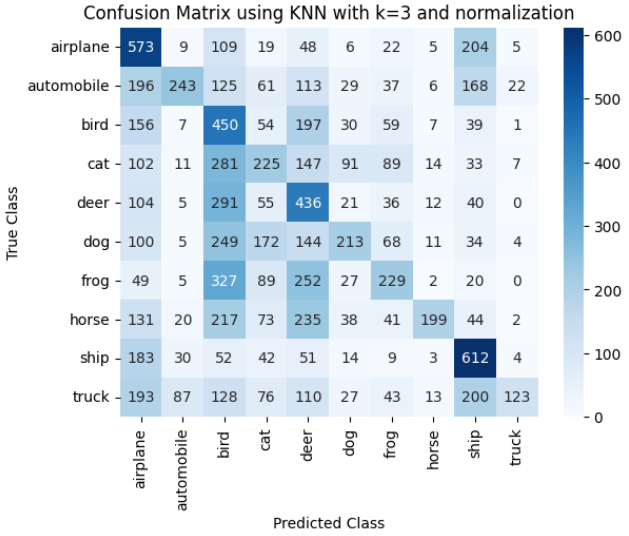

Fig. 9: KNN Confusion Matrix with 3 neighbors and normalization

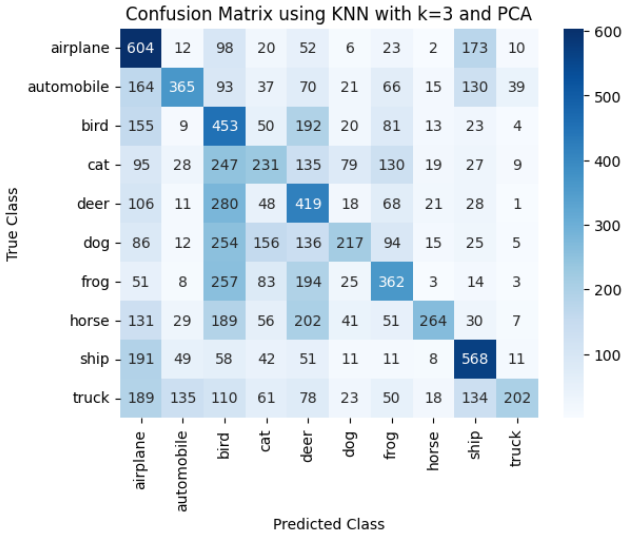

Fig. 10: KNN Confusion Matrix with 3 neighbors and PCA

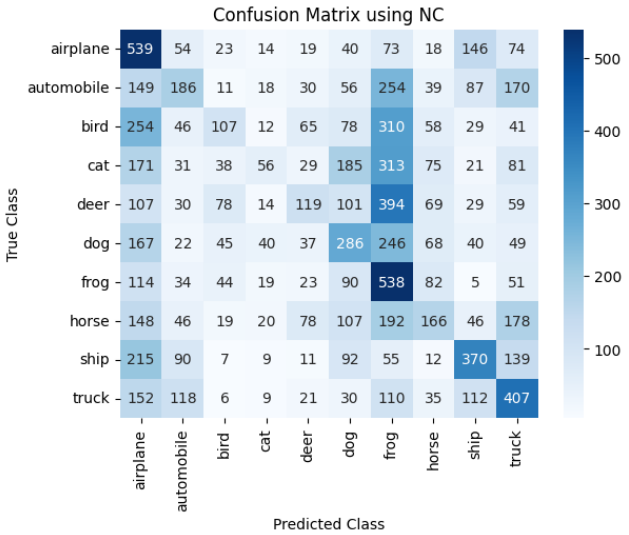

Fig. 11: NC Confusion Matrix

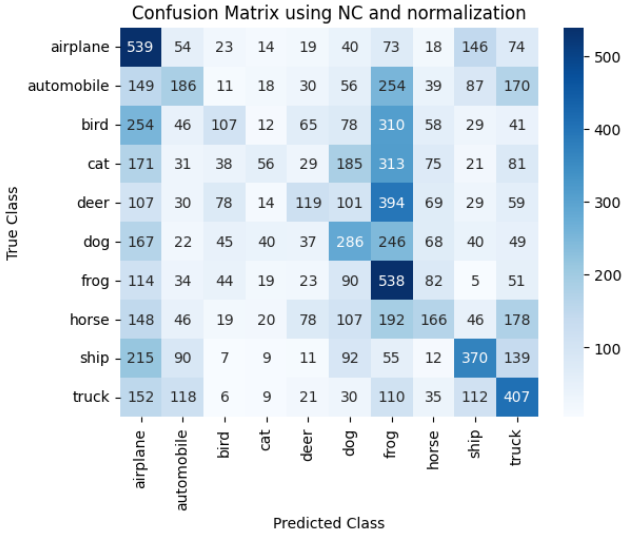

Fig. 12: NC Confusion Matrix with normalization

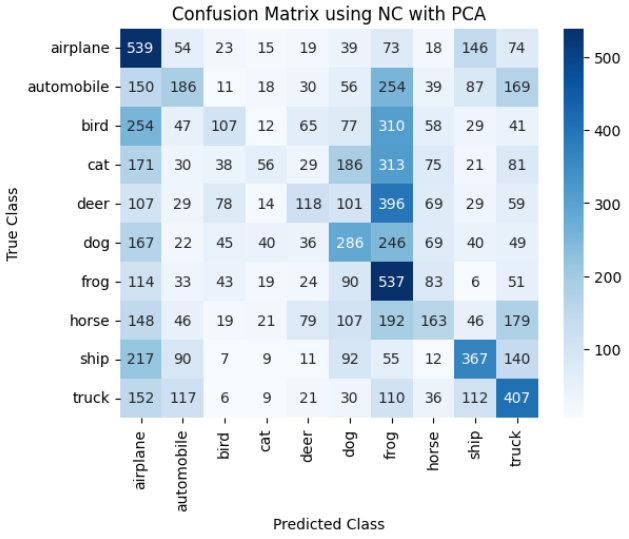

Fig. 13: NC Confusion Matrix with PCA

Observing confusion matrices, it is clear that airplane and ship are the first two most correct-predicted classes for the Nearest Neighbor classifier. When it comes to Nearest Centroid algorithm, airplane and frog are the two most correct-predicted classes.

When it comes to models' response time, simple NC is faster than simple KNN (Fig.14). Normalization has slightly made models faster. Finally, response time plummet when PCA is used. Accurate response time values may differ for each code run but the pattern among them will be the same.

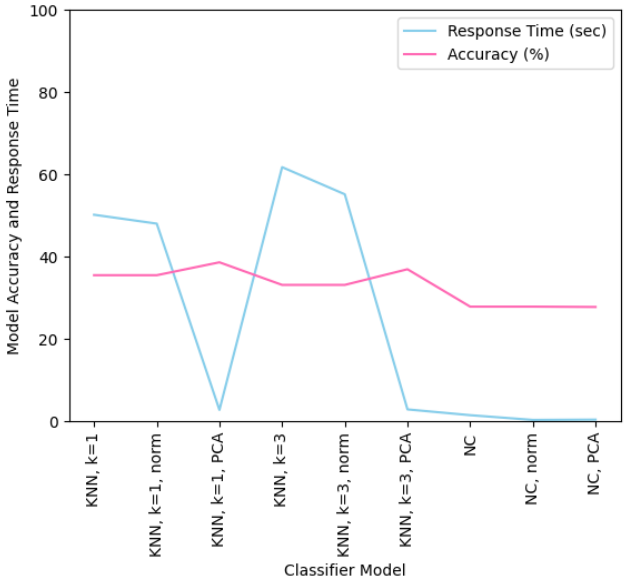

Fig. 14: Models accuracy and response time

B. SAM12 and its variations

TABLE II: Characteristics of SAM12 Models

Model	Optimizer	Training Time(sec)	Test Accuracy(%)	Test Loss
SAM12	AdaMax	2054.05	75.16	1.32
SAM12a	SGD	2060.16	74.40	0.79
SAM12b	AdaMax	2402.93	75.78	0.80
SAM12c	SGD	2102.45	42.40	1.58

Compared to KNN and NC, results are much more encouraging for neural networks. Classification accuracy on test set is 40% higher, expect for SAM12c.

Max-pooling and dropout technique are enlisted against over-fitting. Dropout probability is 20% for dense layers and 40% for convolutional layers. Although, SAM12 accuracy reaches a plato near 95% after 60 epochs of training (Fig.15). Nevertheless, classification accuracy on test samples drops to 75%. This gap indicates that model learns by heart the train-set and is having difficulty deciding on new samples. When it comes to train loss, it drops below 0.2 after 60 epochs (Fig.16). In contrast, this GAP for SAM12a and SAM12b is much smaller. Probably for this reason SAM12 test loss is 1.6 times higher than SAM12a and SAM12b test losses (TABLE II). When it comes to SAM12c, train accuracy is much lower and train loss much higher after 60 epochs.

Images of cats continue to be among the classes with the worst ranking and images of ships continue to be a class with high classification accuracy (Fig.17). Confusion Matrices (Fig.17,18,19,20) provide unambiguous information on how the samples are classified using each model.

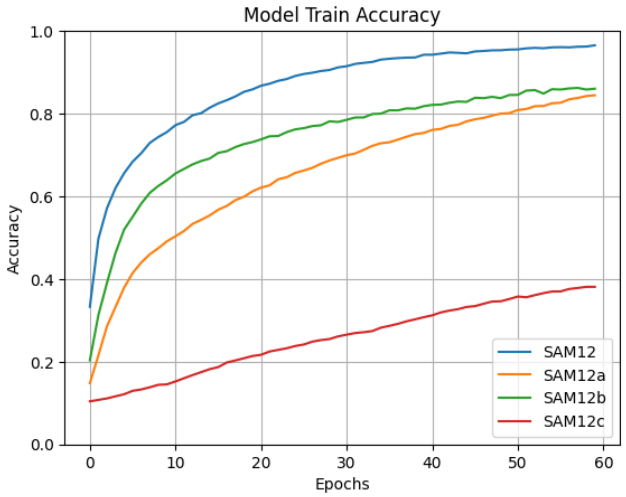

Fig. 15: Models train accuracy

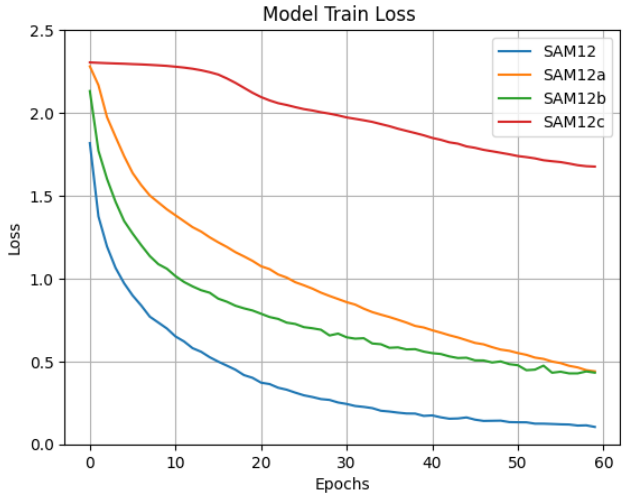

Fig. 16: Models train loss

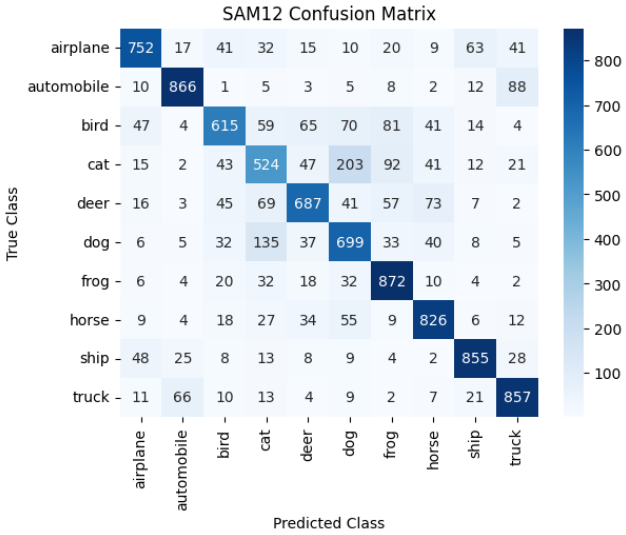

Fig. 17: SAM12 Confusion Matrix

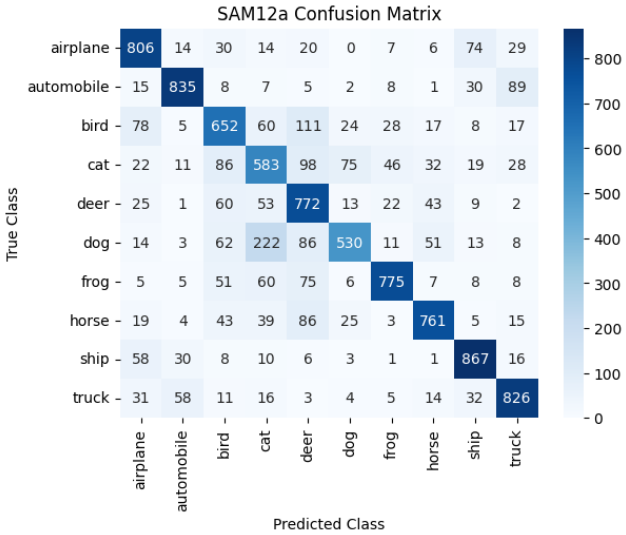

Fig. 18: SAM12a Confusion Matrix

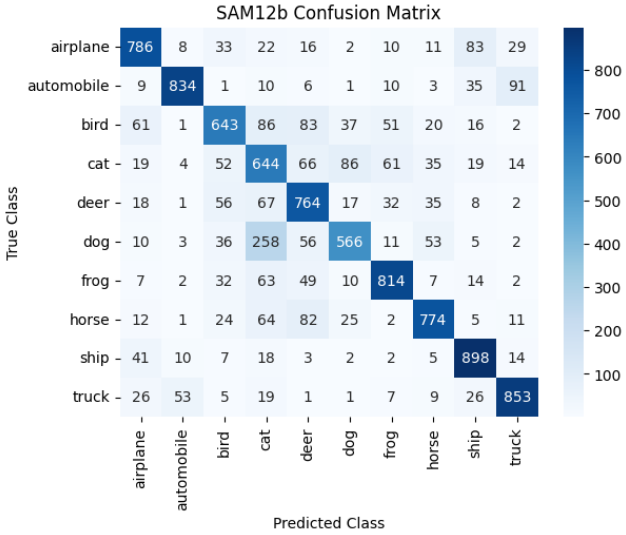

Fig. 19: SAM12b Confusion Matrix

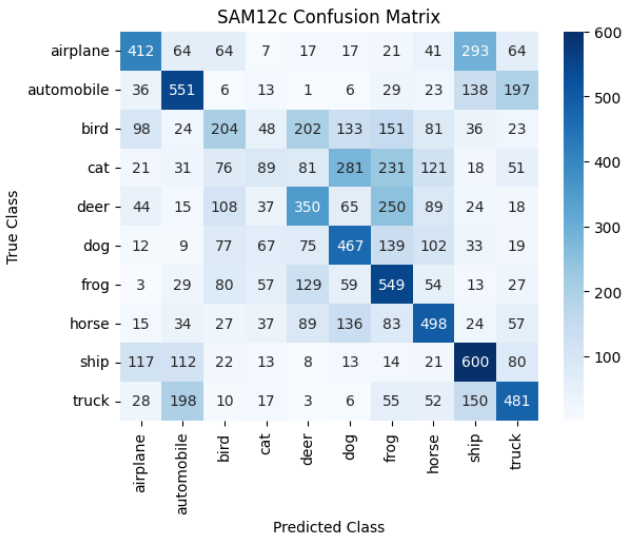

Fig. 20: SAM12c Confusion Matrix

REFERENCES

[1] Alex Krizhevsky. (2009). CIFAR-10 and CIFAR-100 datasets. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.