CIFAR-10 Classification by Utilizing Support Vector Machines

Christos Samaras
Graduate Student of Electrical and Computer Engineering Department,
Aristotle University of Thessaloniki

***Abstract*—CIFAR-10 classification problem is approached by utilizing Support Vector Machines along with Principal Components Analysis (PCA) and statistics about skewness, kurtosis, entropy and bispectrum magnitude. The best achieved test accuracy with SVM is 51%. Furthermore, SVM performance is compared with the performance of MLP with just one hidden layer, K-Nearest Neighbor (KNN) with one and three neighbors, and Nearest Centroid (NC). The overall highest test accuracy of 53.82 is achieved by the MLP model.**

I. Introduction

The CIFAR-10 dataset is composed of 10 object classes with 6000 32x32 RGB images per class. 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.

For CIFAR-10 classification, SVMs are used along with PCA and various statistics. Principal Compononent Analysis is one of the most famous techniques for dimensionality reduction. Support Vector Machines are strictly binary classifiers. In this study, three types of SVMs are utilized: linear SVM, polynomial SVM, and SVM with radial basis function kernel (RBF). A Linear Support Vector Classifier tries to wide the margin between the classes. A polynomial SVM creates more complex decision boundaries in order to separate the classes. A SVM using RBF kernel tries to correlate any instance to support vectors based on their similarities and is preferable for non-linear separable problems.

II. Method

A. Mathematical Background

1) Skewness: For a random variable x following a distribution, skewness denotes the asymmetry of this distribution relative to normal distribution (Fig.1). 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:

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

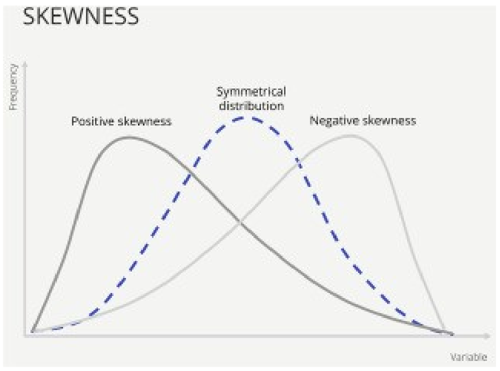


Fig. 1: Skewness

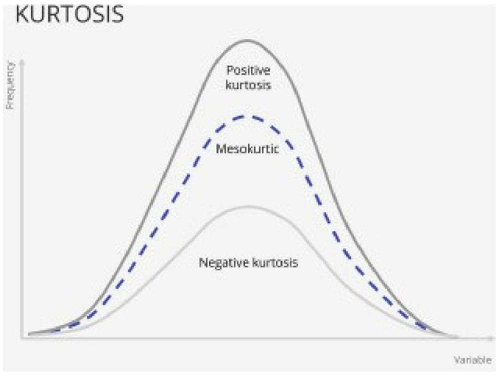


Fig. 2: Kurtosis

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

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

where m_4 is the fourth central moment:

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

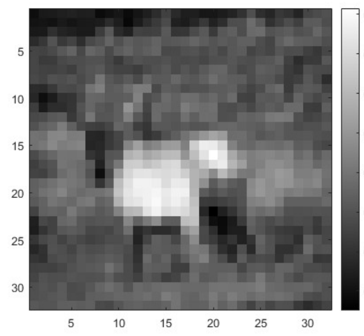


Fig. 3: Gray-scale image

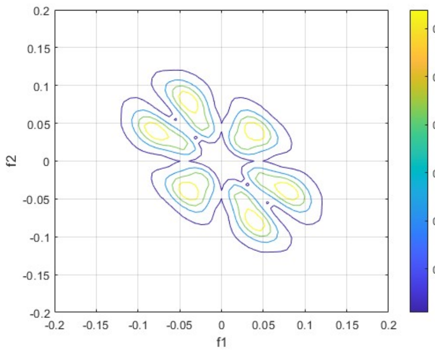


Fig. 4: Bispectrum Visualization

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 $f_c = f_a + f_b$ and $\phi_c = \phi_a + \phi_b$, 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 the Direct Method and *bispecd* function from HOSA Toolbox for MATLAB. Mean and max value and number of peaks of bispectrum magnitude are extracted. In Fig.4, bispectrum of a random sample (Fig.3) is plotted.

C. Support Vector Machine (SVM)

Support Vector Classifiers (SVC) are strictly binary classifiers. Since CIFAR-10 is composed of 10 classes, the one-versus-all (OvA) strategy is utilized to perform multiclass classification. According to OvA, 10 different binary classifiers-one for each class-are trained. For a specific sample, the decision score of each classifier is computed, and the sample is sorted as the class whose classifier has the highest score.

For these machines, there is a regularization parameter C. While C is reduced, the margin expands. Thus, reducing C leads to more instances defining the margin, resulting in a lower risk of overfitting. For polynomial kernels, while the degree is increased, the number of features is also increased, resulting in model deceleration. For RBF kernel, γ acts similarly to C as a regularization parameter.

When it comes to large datasets, Support Vector Classifiers are highly time-consuming machines. Just out of curiosity, RBF SVC was tried to be trained with the whole train set. The model was running for 12 hours without converging. For this reason, $10\,000$ train samples are used for model training and the whole test set, namely $10\,000$ test samples, for model testing. For these samples, PCA is performed so that 90% of features remain. These principal components, concatenated with statistics for skewness, kurtosis, entropy, and bispectrum, are used as features for training and testing. The models are trained for $0.1, 1,$ and 10 C values. Polynomial models are also trained for 2, 3, and 4 polynomial degrees and RBF models for $0.001, 0.01, 0.1$ γ values. The model, which is performing better, is finally trained with all train samples.

D. Other Classifiers

Classification is also approached by a Multilayer Perceptron, by k-Nearest Neighbor and by Nearest Centroid classifiers.

MLP has one hidden layer consisted of 256 neurons. This layer has $Relu$ activation function. The output layer of MLP has $softmax$ for 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 loss function, the Hinge loss is selected, just like SVM. This model is trained with learning rate equal to 0.01, for 30 epochs, and with batch size equal to 128.

KNN is trained for 1 and 3 neighbors. The input of all these models is the same (PCA and statistics) with the input of the SVM model that is finally trained with the whole train set.

III. Conclusion

A. Experimentation with parameters and models of SVM

As mentioned, $10\,000$ train samples are firstly used for training. In the following plots, training time, train accuracy, and test accuracy are displayed for the 3 SVC types and all combinations of parameters.

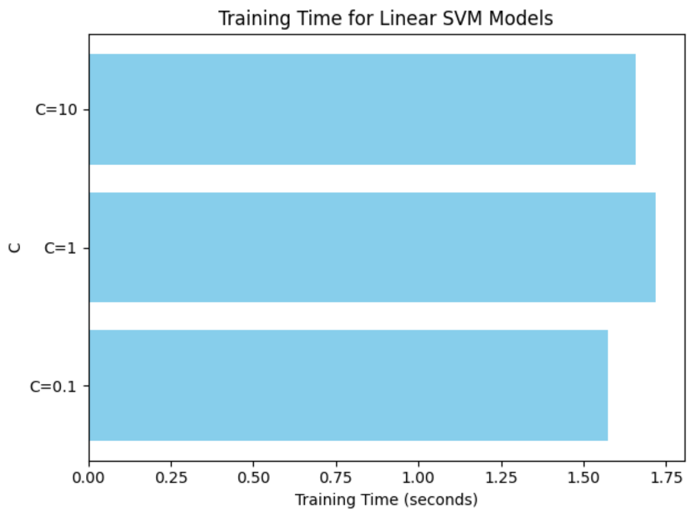


Fig. 5: Training Time of Linear SVCs

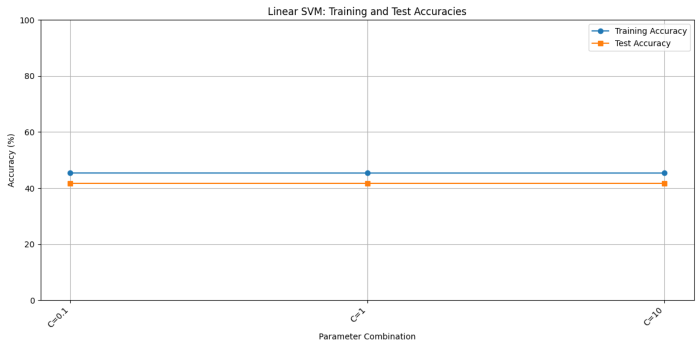


Fig. 8: Accuracy of Linear SVCs

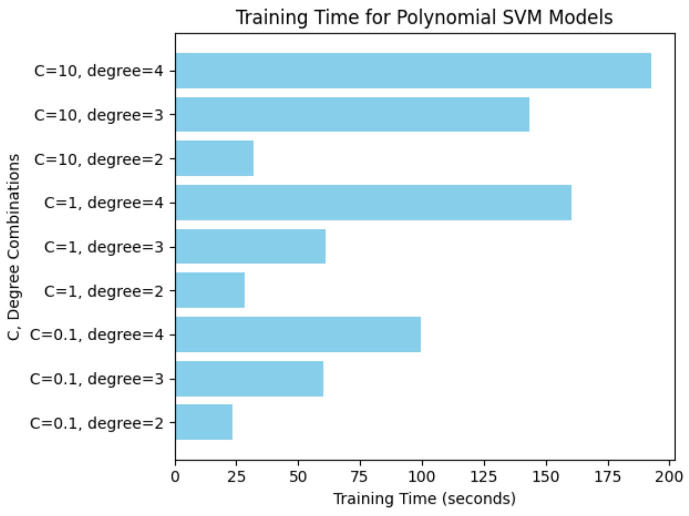


Fig. 6: Training Time of Polynomial SVCs

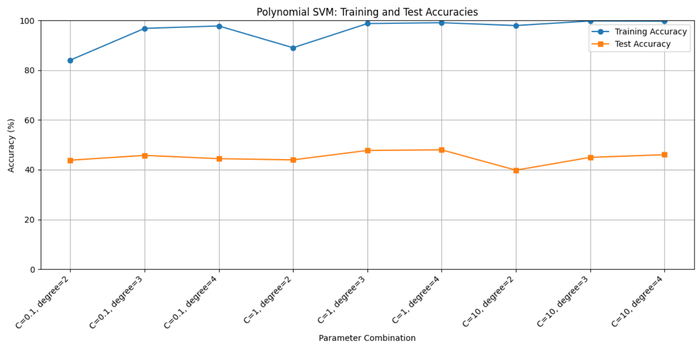


Fig. 9: Accuracy of Polynomial SVCs

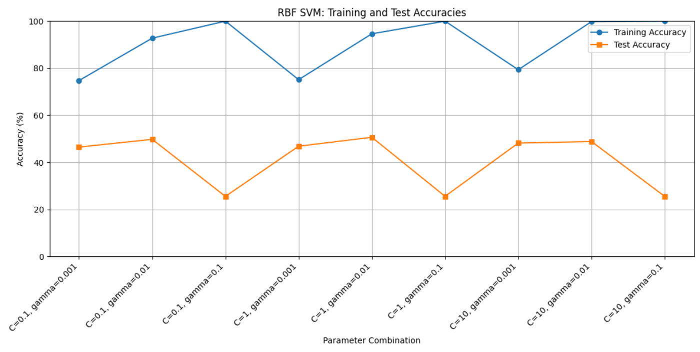


Fig. 10: Accuracy of RBF SVCs

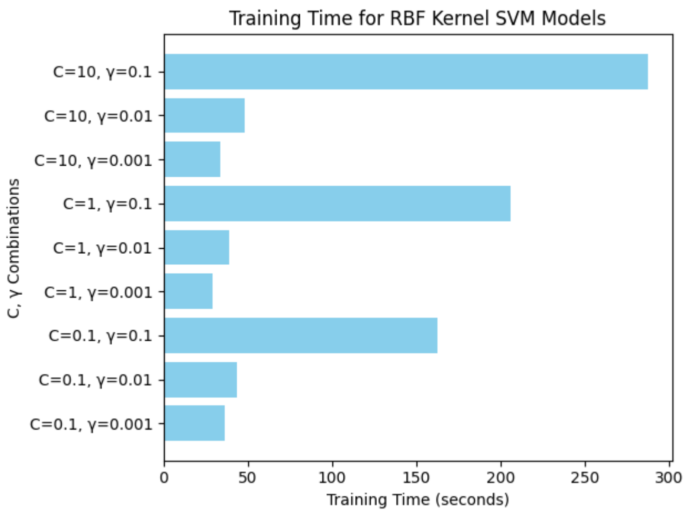


Fig. 7: Training Time of RBF SVCs

As observed, linear SVMs have low accuracies for both training and testing, revealing that the problem is not linearly separable. Although these models are trained up to 100 times faster than the others, their performance is suboptimal. For polynomial kernels with degrees higher than 2, training accuracy exceeds 96%. However, the highest test accuracy is 48% (for $C = 1$ and $degree = 4$), indicating that these models overfit, even when C is set at 0.1. Finally, for the RBF kernel, train accuracy reaches 100% when γ is equal to 0.1. In contrast, the highest test accuracy is 51%, highlighting the strength of RBF machines and the high risk they have to overfit.

TABLE I: Train and Test Accuracy for Different SVMs

SVC	Parameters	Train Accuracy(%)	Test Accuracy(%)
Linear	$C = 0.1$	45.39	41.68
Linear	$C = 1$	45.37	41.68
Linear	$C = 10$	45.37	41.68
Polynomial	C=0.1, degree=2	84.07	43.82
Polynomial	$C = 0.1$ $degree = 3$	96.83	45.76
Polynomial	$C = 0.1$ $degree = 4$	97.82	44.44
Polynomial	$C = 1$ $degree = 2$	89.05	43.95
Polynomial	$C = 1$ $degree = 3$	98.78	47.75
Polynomial	$C = 1$ $degree = 4$	99.13	47.99
Polynomial	$C = 10$ $degree = 2$	97.98	39.81
Polynomial	$C = 10$ $degree = 3$	99.83	44.96
Polynomial	$C = 10$ $degree = 4$	99.74	46.03
RBF	$C = 0.1$ $\gamma = 0.001$	74.75	46.50
RBF	$C = 0.1$ $\gamma = 0.01$	92.76	49.78
RBF	$C = 0.1$ $\gamma = 0.1$	100	25.54
RBF	$C = 1$ $\gamma = 0.001$	75.13	46.86
RBF	$C = 1$ $\gamma = 0.01$	94.62	50.65
RBF	$C = 1$ $\gamma = 0.1$	100	25.57
RBF	$C = 10$ $\gamma = 0.001$	79.37	48.19
RBF	$C = 10$ $\gamma = 0.01$	99.78	48.86
RBF	$C = 10$ $\gamma = 0.1$	100	25.54

B. RBF SVC, MLP, KNN and NC results

Based on the analysis above, the RBF SVC model with $C = 0.1$ and $\gamma = 0.001$ is selected to be trained with the whole train set (PCA and statistics) because the lower the C and γ, the lower the risk of overfitting. The performance of this SVM is compared with the performance of MLP, KNN and NC (TABLE II). As seen, the MLP with just one hidden layer has slightly higher test accuracy than SVM and much lower train time. The confusion matrices (Fig.11,12,13,14,15) provide a comprehensive representation of correct and incorrect classifications for each model.

The train accuracy and loss of MLP over each epoch can be seen in Fig.16.

TABLE II: Comparison of RBF SVM with MLP, KNN and NC

Model	Train Accuracy (%)	Test Accuracy (%)	Training Time (sec)
RBF SVM $C = 0.1$ $\gamma = 0.001$	61.54	51.00	1172.75
MLP	71.08	53.82	19.54
KNN 1 neighbor	100.00	35.15	0.03
KNN 3 neighbors	61.41	34.61	0.01
NC	24.56	25.44	0.05

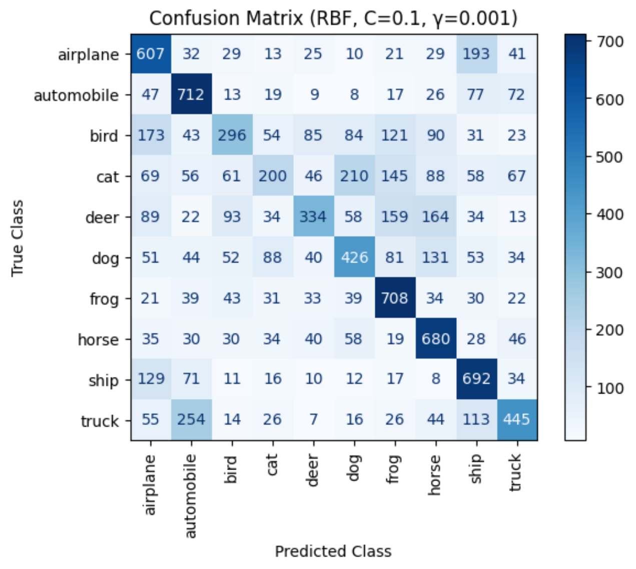


Fig. 11: Confusion Matrix of RBF SVC

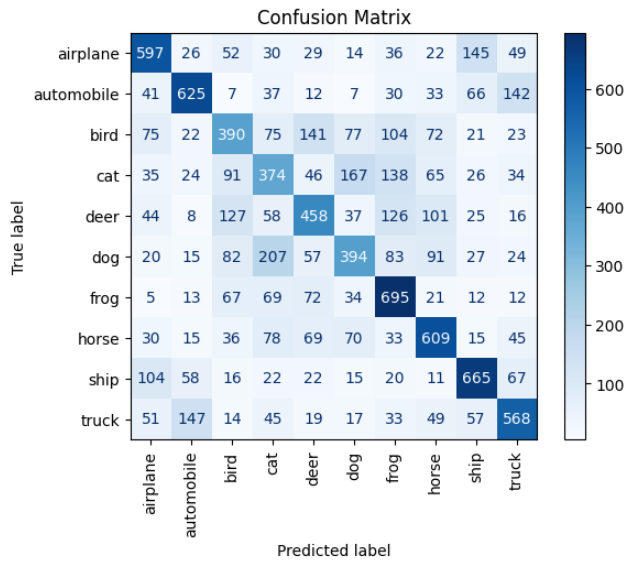


Fig. 12: Confusion Matrix of MLP

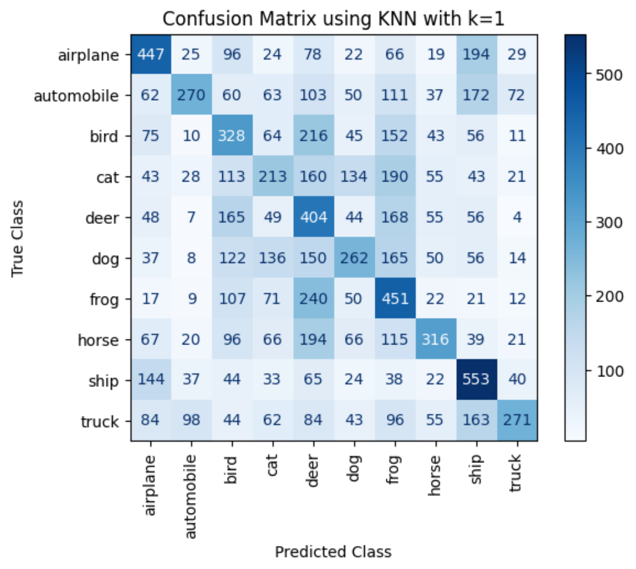
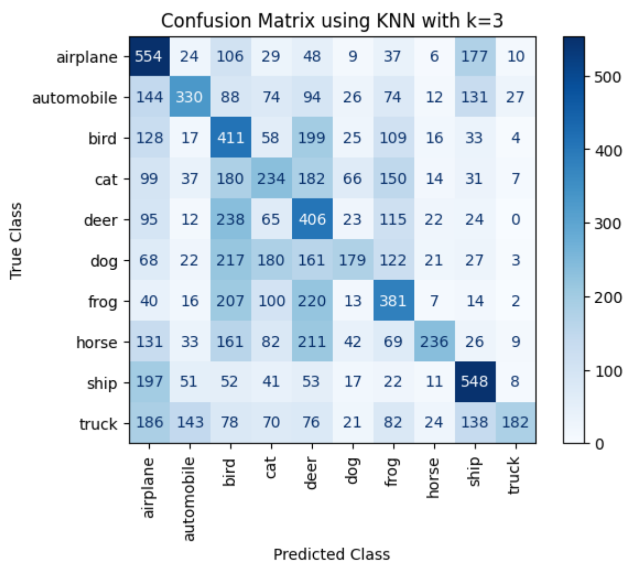
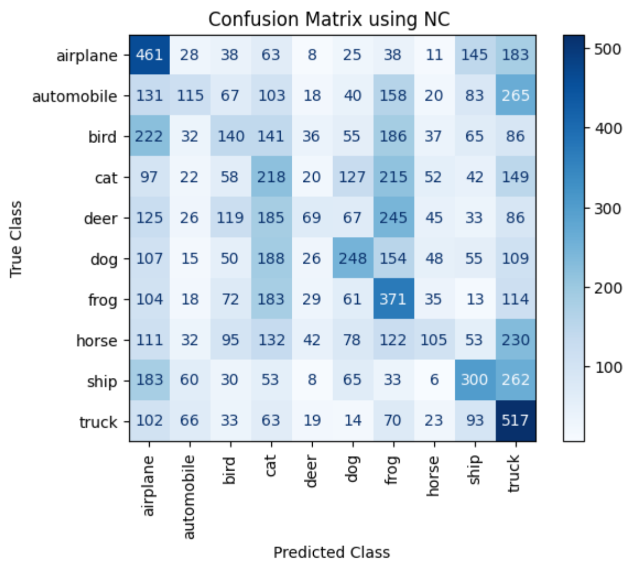


Fig. 13: Confusion Matrix of KNN with 1 neighbor



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	554	24	106	29	48	9	37	6	177	10
automobile	144	330	88	74	94	26	74	12	131	27
bird	128	17	411	58	199	25	109	16	33	4
cat	99	37	180	234	182	66	150	14	31	7
deer	95	12	238	65	406	23	115	22	24	0
dog	68	22	217	180	161	179	122	21	27	3
frog	40	16	207	100	220	13	381	7	14	2
horse	131	33	161	82	211	42	69	236	26	9
ship	197	51	52	41	53	17	22	11	548	8
truck	186	143	78	70	76	21	82	24	138	182

Fig. 14: Confusion Matrix of KNN with 3 neighbors



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	461	28	38	63	8	25	38	11	145	183
automobile	131	115	67	103	18	40	158	20	83	265
bird	222	32	140	141	36	55	186	37	65	86
cat	97	22	58	218	20	127	215	52	42	149
deer	125	26	119	185	69	67	245	45	33	86
dog	107	15	50	188	26	248	154	48	55	109
frog	104	18	72	183	29	61	371	35	13	114
horse	111	32	95	132	42	78	122	105	53	230
ship	183	60	30	53	8	65	33	6	300	262
truck	102	66	33	63	19	14	70	23	93	517

Fig. 15: Confusion Matrix of Nearest Centroid

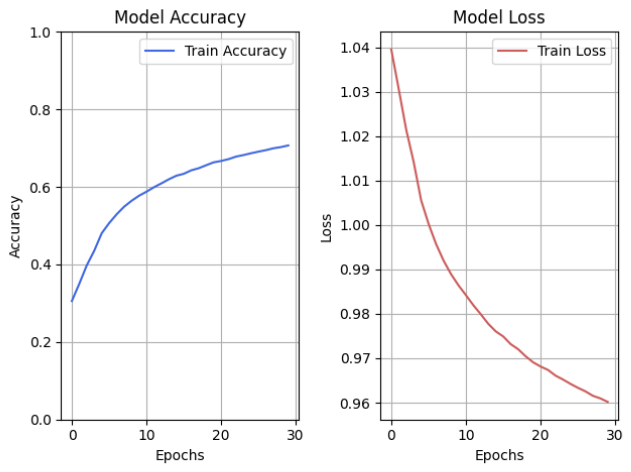


Fig. 16: Train Accuracy and Loss of MLP

Note: All models are implemented in python with the help of TensorFlow, an open source machine learning library.