Assignment 1
Pattern Recognition and Neural Network

Alokendu Mazumder (20134)* , Manan Tayal (20133)**, Piyush Tiwary (19897)†, and Saurabh Shrivastava (19881) ‡

Email: {*alokendum, **manantayal, †piyushtiwary, ‡saurabhshriv}@iisc.ac.in

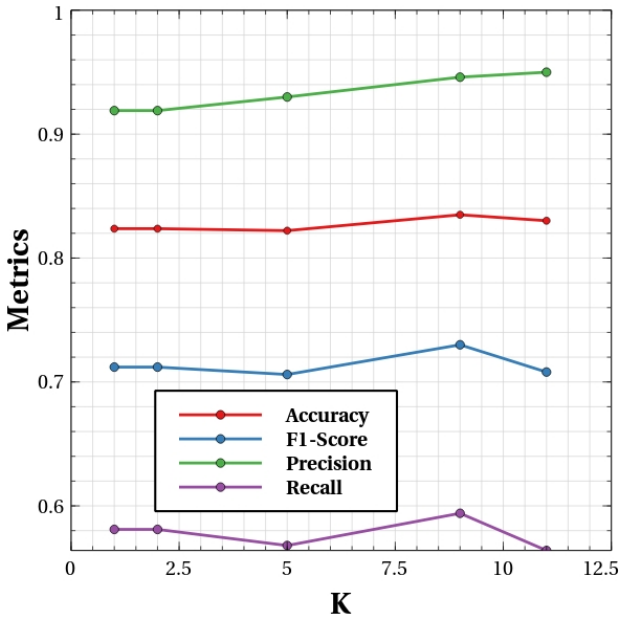


Fig. 1: Variation in accuracy for different value of K in kNN

***Abstract*—In this assignment we perform various classification, regression and generative tasks.**

I. Introduction

This assignment is about implementation of primitive ML algorithms for 5 different tasks.

II. Binary and Multi-Class Classification Problem

Binary Classification Problem In this problem, the Chest X-Ray images have been given and they are to be classified either as Normal or as Pneumonia.The PneumoniaMNIST dataset was divided into training, testing and validation datasets in the proportion 0.7:0.2:0.1. Data pre-processing step involved converting the 28x28 sized images into 784 dimensional vectors followed by mean centering and scaling to unit variance. We implemented Gaussian MLE , KNN, Naive Bayes and Parzen Window methods and achieved decent results in terms of variegated metrics such as accuracy, recall, precision, F1-score, AUC to name a few. Gaussian MLE performed the best in terms of accuracy with the same reaching to 0.884, followed by Parzen Window. For Parzen window, we used both Gaussian Window and the Box Window in which the latter performed slightly better. For Gaussian MLE, the ML estimate parameters of the Multivariate Gaussian were calculated, representing densities of each class. The consequent class conditional densities were used to compute the posteriors of the test data, thereby predicting the class labels.

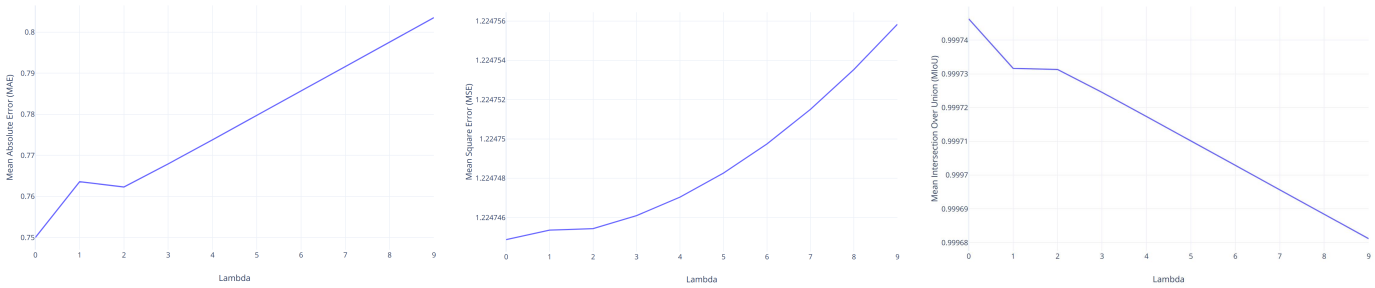
MultiClass Classification Problem This problem accords us with the Blood Cell Microscope images and demands them to be classified as one out of the 8 presumable categories. The BloodMNIST dataset was divided into training, testing and validation datasets in the proportion 0.7:0.2:0.1. The images were of 28x 28 size which we converted into a 784-dimensional vector format.We converted the data into a 0-mean and unit-variance form. We implemented Naïve Bayes, MAP, Gaussian MLE, KNN and Parzen Window. We used various metrics for performance measurements, including but not limited to accuracy, precision, recall, F1 score and AUC. We achieved mediocre results for almost all of them except for KNN, where we achieved an accuracy of 0.76. KNN was performed with hyperparameter tuning using validation dataset by varying K from 1 to 30 and K=9 gave optimal results in terms of accuracy. For Gaussian MLE, the ML estimate parameters of the Multivariate Gaussian were calculated representing densities of each class. The consequent class conditional densities were used to compute the posteriors of the test data, thereby predicting the class labels.

III. Bounding Box Regression Problem

In this problem, we are given images of traffic signs and the task is to find the bounding boxes that encompass the sign in the image. So, basically it will be a regression problem where we are approximating a function $f : \mathbf{R}^d \rightarrow \mathbf{R}^4$. Here for each image we have to predict the set of (x, y) tuples which represents the co-ordinates of the bounding box in a specific way. Basically we have to regress the images over maximum of x co-ordinates and maximum of y co-ordinates of the boxes, hence mapping the feature vector to a 4 dimensional subspace. We haven't done any specific feature engineering for the images, we took the images and first resized each training images into $(30, 45)$ and scaled the coordinates of the bounding boxes according to the resizing of images. We used a linear

	Parameters	True Positives	False Positives	True Negatives	False Negatives	Precision	Recall	Accuracy
MLE	Gaussian Class Conditional	181	53	371	19	0.85	0.774	0.884
kNN	k = 11	137	97	381	9	0.938	0.585	0.830
Naive Bayes	-	234	0	0	390	0.375	1.0	0.375
Parzen Window	Gaussian Window	132	102	383	7	0.95	0.564	0.825
	Box Window	123	112	367	14	0.93	0.61	0.839

TABLE I: Performance of different algorithms on Binary Classification



(a) (b) (c)

Fig. 2: Variation of Accuracy metrics for different values of regularization constant :(a) Mean Absolute Error (MAE), (b) Mean Squared Error (MSE), (c) Mean Intersection Over Union (MIoU)

hypothesis for regression and the optimal hyper-plane is given by:

$$w^* = (X_{train}^t X_{train}) X_{train} y_{train} \tag{1}$$

Where X_{train} is the training data matrix where each column is a flatten version of resized image and it ends with a constant 1, denoting that bias term is included in the w. Hence we have **d** = (1350 + 1) for this problem. **t** denotes the transpose operator and y_{train} is the target vector containing the min and max coordinates (x, y) of bounding boxes. The last coordinate entry of w^* is the optimal bias. The results are shown below.

The same regression problem is solved using $L_2 regression$ also. In that case we have our optimal hyper-plane as:

$$w^* = (X_{train}^t X_{train} + \lambda I) X_{train} y_{train} \tag{2}$$

Here, λ is the regularization constant(a hyper-parameter). We tuned our optimal solution for different λ and noted the results for the metrics asked.

The results could have been better if we would have used non-linear hypothesis functions!

	λ = 0	λ = 1	λ = 5	λ = 9
MSE	1.225	1.225	1.225	1.225
MAE	0.750	0.763	0.779	0.803
mIoU	0.999	0.999	0.999	0.999

TABLE II: Accuracy metrics for different values of regularization constant

IV. Frame Classification on Audio Data

For vowel prediction task, we extracted Mel filter bank features for each speech signal given. We threshold(ed) the dimension of extracted features to 156 to maintain same size for all the input data-point. All the models were trained on speaker – FCJF0, FDAW0, FDML0 and FECD0, whereas for testing, data corresponding to speakers FETB0, FJSP0 and FKFB0 were taken.

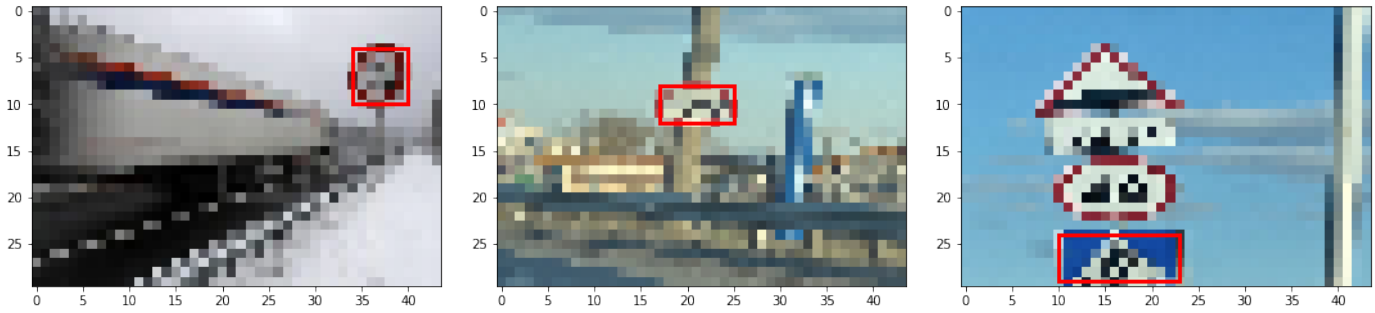
Different classification/generative algorithms with proper metrics are implemented as asked in the question and their results are given in appropriate table.

V. Generative Models

The task here is to fit a GMM over TinyImagenet data set. TinyImagenet is a miniature version of the mammoth Imagenet data set. It has 100 classes. Each class has 1,000 training images, 100 validation images. The test set contains 10,000 images in total. So, we fit a GMM over it and estimate it's parameter with famous EM algorithm.

As this algorithm maximizes only the likelihood, it will not bias the means towards zero, or bias the cluster sizes to have specific structures that might or might not apply. While working on it, we encountered several difficulties, one of which is when one has insufficiently many points per mixture, estimating the covariance matrices becomes difficult, and the algorithm is known to diverge and find solutions with infinite likelihood unless one regularizes the covariances artificially.

The main difficulty in learning Gaussian mixture models from unlabeled data is that it is one usually doesn't know which points came from which latent component (if one has access

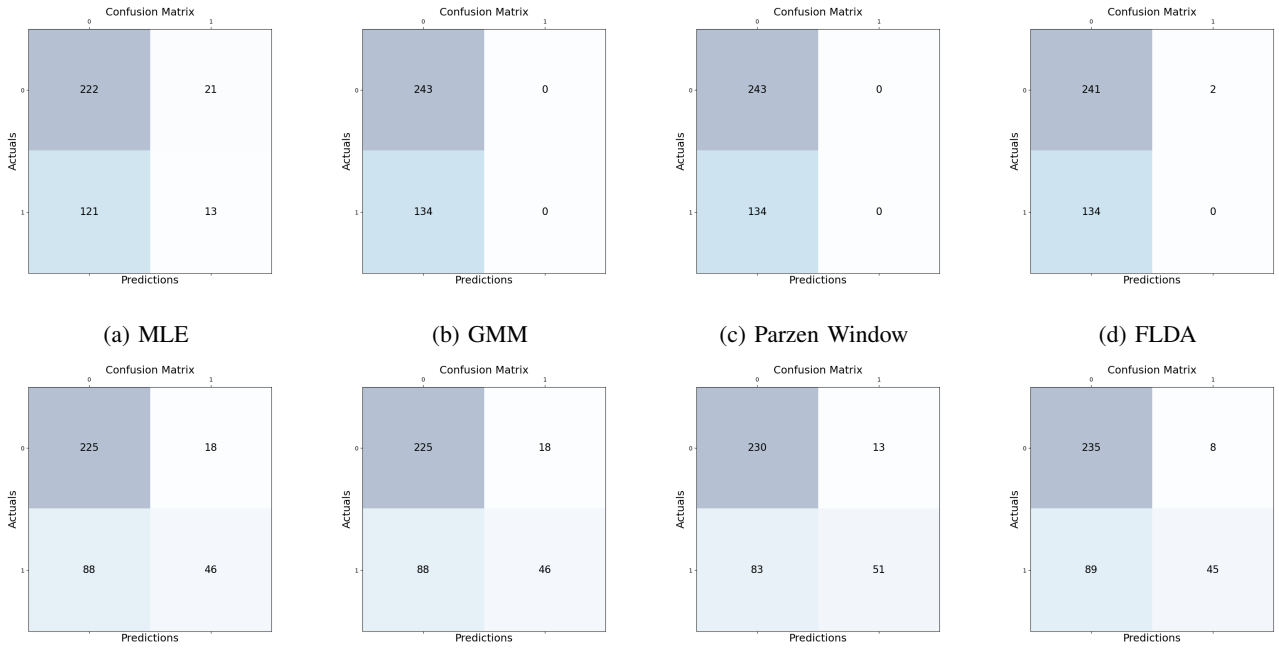


(a) (b) (c)

Fig. 3: Predicted Bounding Box

	Parameters	True Positives	False Positives	True Negatives	False Negatives	Precision	Recall	Accuracy
MLE	Gaussian Class Conditional	13	21	222	121	0.382	0.055	0.909
kNN	k = 1	46	18	225	88	0.718	0.169	0.718
	k = 2	46	18	225	88	0.718	0.169	0.718
	k = 5	51	13	230	83	0.796	0.181	0.745
	k = 9	45	8	235	89	0.849	0.160	0.742
Parzen Window	Box Window	0	0	243	134	-	0.0	0.644
	Gaussian Window	1	0	242	134	0.0	0.0	0.644
FLDA	-	0	2	241	134	0.0	0.0	0.639
MAP	Gaussian Class Conditional and Prior	11	25	218	123	0.305	0.048	0.789
GMM	n components = 2	0	0	243	134	-	0.0	0.644
	n components = 5	0	0	243	134	-	0.0	0.644

TABLE III: Performance of different algorithms on TIMIT dataset



(a) MLE (b) GMM (c) Parzen Window (d) FLDA

(e) kNN (n=1) (f) kNN (n=2) (g) kNN (n=5) (h) kNN (n=9)

Fig. 4: Confusion Matrix for different classification methods on TIMIT Dataset.

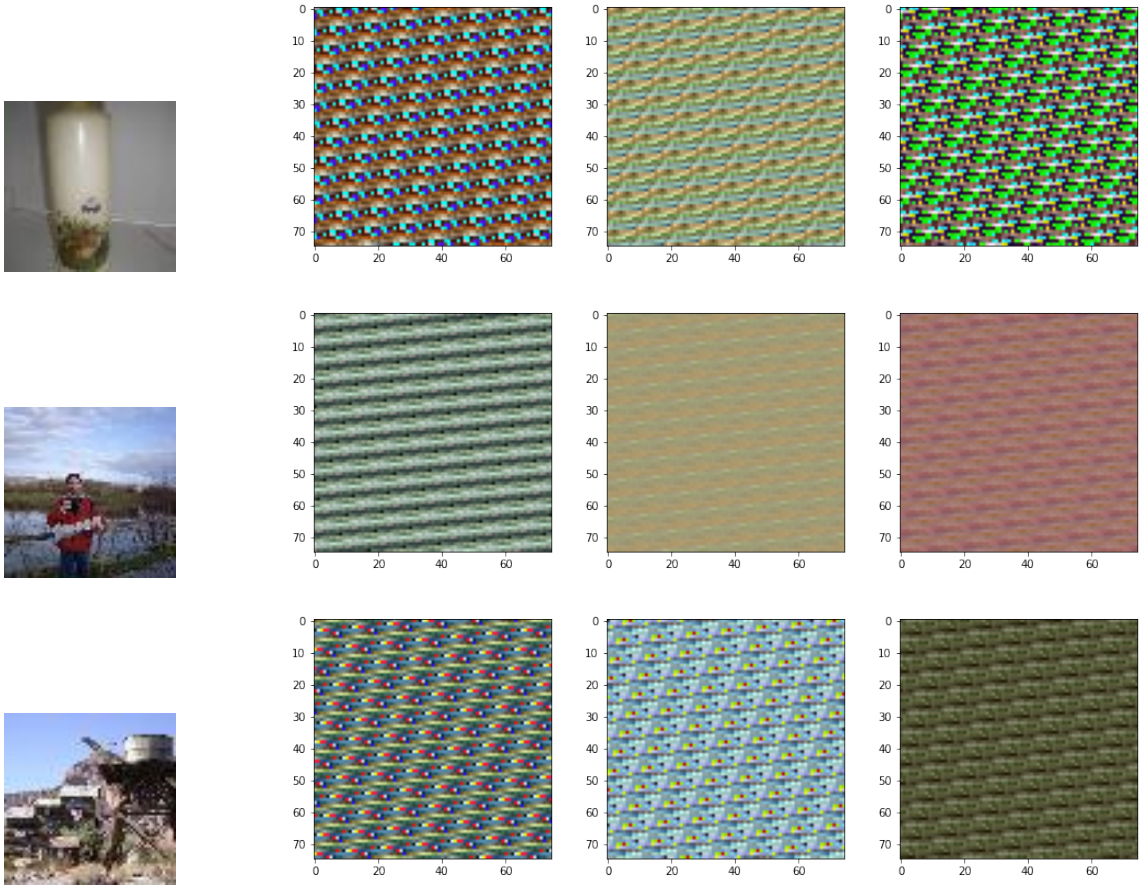


Fig. 5: Illustrative examples for images generated using GMM for different values of k. The left column shows the original images from the dataset, the second column shows images generated from GMM for $k = 2$, the third column shows images generated from GMM for $k = 3$ and the fourth column shows images generated from GMM for $k = 4$.

to this information it gets very easy to fit a separate Gaussian distribution to each set of points). Expectation-maximization is a well-founded statistical algorithm to get around this problem by an iterative process. First one assumes random components (randomly centered on data points, learned from k-means, or even just normally distributed around the origin) and computes for each point a probability of being generated by each component of the model.

For testing, we sampled 1000 points(images) randomly from the GMM after fitting it over TinyImagenet and 1000 points(images) randomly from the data set itself and computed the FID for various values of **k** (no. of Gaussian's to be fit). The images are first down sampled into $(8, 8, 3)$ to make the algorithm converge faster in sense of computational time. We then fit the GMM's over the entire data set (10,00,00 flattened images). To compute the FID, we must up-sample the generated and original samples to $(75, 75, 3)$ dimension as inception takes samples of size greater than or equal to the stated above.

We restricted our search for Gaussian's to **k** = $(2, 3, 4)$ only, because if we fit it for higher values of **k**, it takes eternity to converge! The FID scores are not at all promising because the number of Gaussian's are very less as compared to actual number of classes. Also the FID score depends of the chunk of sampled points, if bad points gets sampled, it will give worse FID scores. Hence, FID score is sensitive towards **k** and the type of samples.

FID score is a neural network based distance measure which uses the inception module widely used in GoogleNet.

	k = 2	**k = 3**	**k = 4**
FID	172285	76172	109565

TABLE IV: FID Score for different value of number of components for GMM

Method	Metric	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
MLE	TP	60	5	0	183	0	0	470	7
	TN	2680	2795	3110	2212	3178	3137	1188	2951
	FP	184	619	311	396	243	284	196	463
	FN	497	2	0	630	0	0	1567	0
	Precision	0.108	0.714	1	0.225	1	1	0.231	1
	Recall	0.246	0.008	0	0.316	0	0	0.706	0.015
	F1-Score	0.150	0.016	0	0.263	0	0	0.348	0.029
	AUC	0.121	0.500	0.13	0.290	0.17	0.26	0.3	0.29
	Accuracy	**0.211**							
kNN	TP	121	539	242	413	195	191	458	469
	TN	3127	2673	3033	2552	3119	2989	2732	2929
	FP	123	85	69	166	48	93	208	1
	FN	50	124	77	299	59	148	23	22
	Precision	0.708	0.813	0.759	0.587	0.768	0.563	0.952	0.955
	Recall	0.496	0.864	0.778	0.713	0.802	0.673	0.688	0.998
	F1-Score	0.583	0.838	0.768	0.644	0.785	0.613	0.799	0.976
	AUC	0.2420	0.4347	0.3143	0.1424	0.3305	0.1291	0.19913	0.21318
	Accuracy	**0.768**							
Naive Bayes	TP	104	0	0	137	138	155	556	3
	TN	2407	2797	3110	2669	2890	3027	1768	2951
	FP	140	624	311	442	105	129	110	467
	FN	770	0	0	173	288	110	987	0
	Precision	0.119	1	1	0.442	0.324	0.585	0.360	1
	Recall	0.426	0	0	0.237	0.568	0.546	0.835	0.006
	F1-Score	0.186	0	0	0.308	0.413	0.565	0.503	0.013
	AUC	0.135	0.1	0.1	0.792	0.279	0.1409	0.0563	0.3
	Accuracy	**0.319**							
Parzen Window	TP	125	3	6	178	136	146	654	5
	TN	2354	2589	3015	2456	2922	3125	1456	2900
	FP	145	625	301	452	107	135	126	478
	FN	756	6	7	179	279	115	999	1
	Precision	0.22	0.9	0.98	0.442	0.334	0.655	0.370	1
	Recall	0.526	0.03	0.015	0.137	0.468	0.446	0.955	0.016
	F1-Score	0.206	0.02	0.02	0.318	0.393	0.435	0.483	0.023
	AUC	0.135	1	1	0.682	0.459	0.86	0.563	0.86
	Accuracy	**0.215**							
MAP	TP	55	10	0	173	0	1	480	6
	TN	2570	2895	3120	2112	3278	3237	1088	2951
	FP	174	629	301	406	233	294	186	453
	FN	487	3	2	650	10	5	1547	0
	Precision	0.158	0.624	1	0.235	1	1	0.291	1
	Recall	0.356	0.018	0	0.216	0	0	0.606	0.025
	F1-Score	0.250	0.116	0	0.283	0	0	0.448	0.139
	AUC	0.321	0.900	1	0.290	1	1	0.3	0.8
	Accuracy	**0.261**							

TABLE V: Performance of different Algorithms on Multi-Class Classification Task