CS60050 MACHINE LEARNING
ASSIGNMENT 03 — MULTI-LAYER PERCEPTRON CLASSIFIER

Group 23
Amrta Chaurasia (19EE10004)
Nakul Aggarwal (19CS10044)

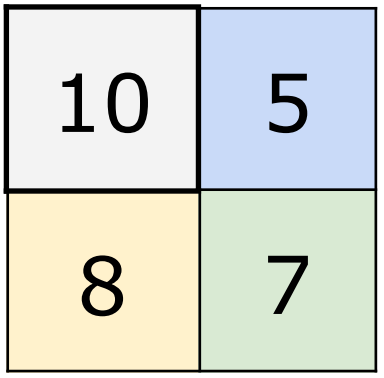
PREFACE

➢ **Dataset used: Optical Recognition of Handwritten Digits (ORHD)**

PRE-PROCESSING ORHD DATASET (STAGE I)

The dataset originally had binary-coloured (0 for white and 1 for black) images of the digits, which were available as 32x32 bitmaps. In this case, each picture could be mapped to a 1024 dimensional feature vector where each attribute is categorical and binary in nature. We selected the pre-processed version of the dataset in which the 32x32 bitmaps are divided into non-overlapping blocks of 4x4 and the number of black pixels (with value 1) in each block is assigned as the attribute value of the block. This generates a bitmap of 8x8 size where each cell is a categorical attribute that can have value from 0 to 16. This **reduces dimensionality from 1024 to 64** and gives invariance to small distortions.

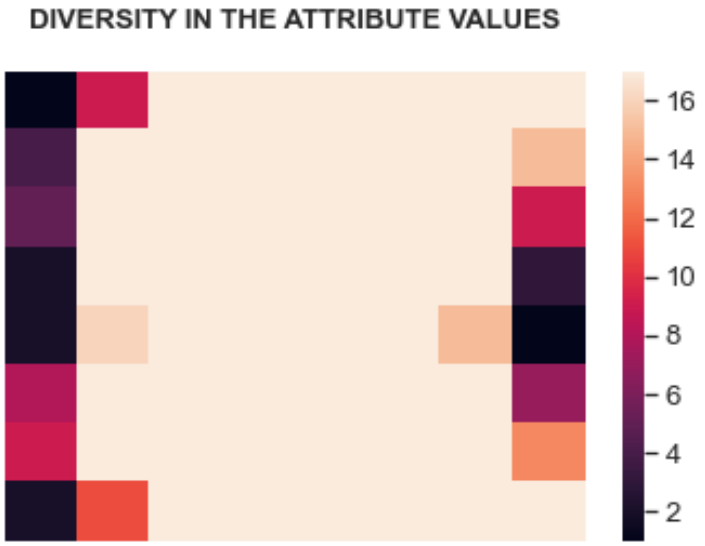
0	0	1	1	1	1	0	0
1	0	1	1	1	0	0	0
1	0	1	0	0	1	0	0
1	1	1	0	0	0	0	1
1	0	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	1	1	1	0
0	0	0	0	0	1	1	0



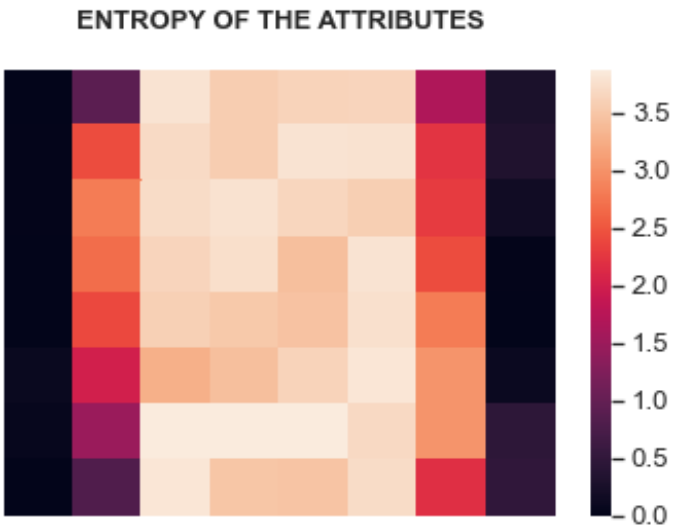
PRE-PROCESSING ORHD DATASET (STAGE II)

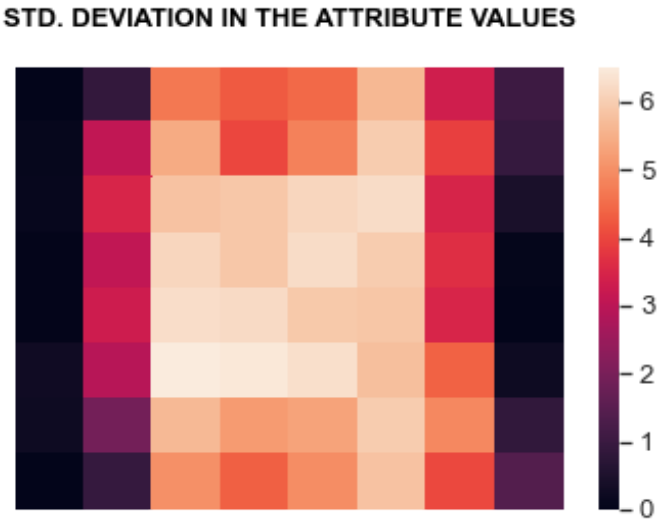
After the first stage of pre-processing, each sample is a vector of 64 categorical attributes, each one of which can assume one of the 17 values (from 0 to 16). In this stage we aim at omitting even more attributes based on the following heuristics. Note that the aim to omit some attributes is not to solely reduce the dimensionality of the instances but is also to focus more on the other attributes that might contribute much more information and semantics to the underlying relatedness between samples of the same classes.

Diversity of an attribute is defined as the number of distinct values that are observed for that particular attribute in the entire dataset. Refer to this 8x8 heatmap in which brighter the color of the cell , more diverse is the corresponding attribute. Naturally, the cells on the left and right edges of a picture hardly have any black pixels hence making the blocks on the edges of their corresponding 8x8 bitmaps much less diverse than the ones around the center. The attributes that are very less diverse, that means the ones that observe very less (say no more than 2) values out of a set of total 17 possible values, can be omitted because their behaviour is more or less constant across the samples of different classes, especially when compared to the other important attributes that may observe all the 17 possible values.



Diversity of an attribute may be a little misleading heuristic in some cases. Say an attribute observes 15 (out of 17) values in a dataset of 1000 images. But in 950 of the images its value is 0 and in the other ones it observes the remaining 14 values. Though this attribute has a very good diversity, yet it can still be omitted due to its almost constant behaviour. In these cases the *standard deviation* of the values can be used. This metric is better than diversity but is more suited for continuous-valued attributes. In the case of categorical attributes, *entropy* is more relevant.





Entropy measures the amount of information that is present in the attribute. Attributes with less entropy (below a threshold) can be omitted because of their less contribution in determining the class to which a sample belongs to. As is clear from the heatmap above, some attributes can have entropy as high as 3.5. In this case, we removed the attributes with entropy less than 0.1 in the second pre-processing stage. It turns out that there are 10 attributes with such low entropy. This reduces **the dimension of the feature vectors further from 64 to 54**.

NOTE

A total of 50 multi-layer perceptron classification models were trained in this project, across all the parts. For each model, we generated a detailed analysis report which includes all the information related to the model, training, evaluation, graphs and confusion matrices. Though we have included most of the information, related to the accuracies, losses, training times etc for all the models and also confusion matrices for some models, it was not logical to include the entire

analysis reports of all the models in this document itself because of the constraint on its length. You can find the detailed analysis report of every model in GRP23_A03_IMPLEMENTATION.pdf.

PART 01

TASK SUMMARY

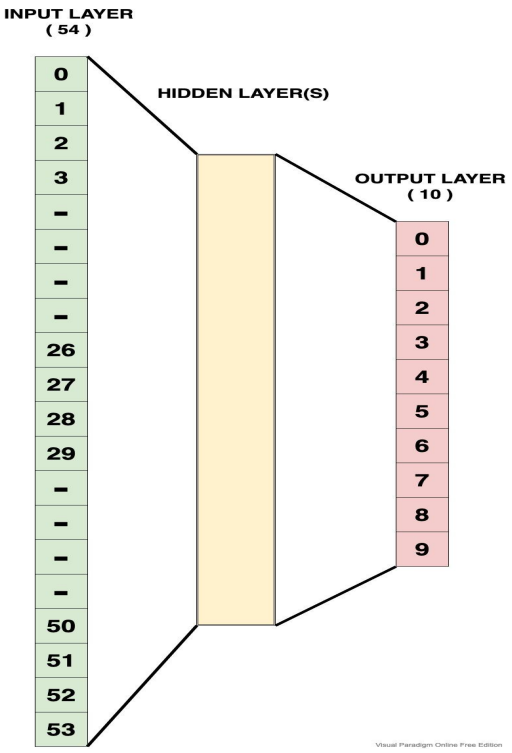
This part involves specifying the various assumptions and design considerations for the MLP classifier that will be trained in the subsequent parts. The number of nodes in the input and output layers of the Multi-Layer Perceptron and the various hyperparameters along with their significance are specified and explained. Some other algorithmic details are also specified.

SPECIFICATIONS

INPUT & OUTPUT LAYERS

After the two stages of pre-processing the data samples, the dimensionality of each sample was reduced from 1024 to 54. The input layer of the MLP model should accept a vector of these 54 attributes and hence the **input layer must have 54 nodes**.

The 54-dimensional instances have to be mapped to a unique label that belongs to one of the 10 classes (labelled from 0 to 9). Though at first it might be intuitive to have a single node in the output layer that gives the label of the class to which the instance belongs to, it is not possible because these labels are categorical and there is no such constraint on the values that the nodes in the MLP can assume. Therefore, it is better to have **10 nodes in the output layer** such that the value of the i^{th} node is *positively correlated* with the confidence that the instance belongs to that i^{th} class. The class predicted by the model is the one with the highest confidence.



The above diagram shows the basic schematic of every MLP classifier that will be trained in this project. It has an input layer of fixed size (will be changed later when the dimensionality is reduced) and an output layer of fixed size. The internal layers are hidden layers that, as you will see later, will be treated as a hyperparameter.

HYPERPARAMETERS

The various hyperparameters that are assumed while training the MLP Classifier are as follows.

• Architecture – The basic prototype of the MLP classifier will be the same as the one shown above for all the models that will be trained in the subsequent parts. But, the number and the dimensionality of the hidden layers can vary. So the architecture of the model can be varied by varying dimensionality or the number of hidden layers.

• Learning Rate – Large training rates can result in unstable training and small learning rates fail to train or converge very slowly. Therefore, the optimal learning rate, that may be dependent on the architecture of the model, must be discovered by adjusting it and observing the results.

• Momentum – Adaptive learning rates can accelerate training and alleviate some pressure of choosing the most optimal learning rate, which itself may require a lot of experiments. Momentum accelerates training and attempts to make the model converge faster to an optimal configuration.

• Batch Size – Using a batch size (or mini-batch size) greater than 1 can significantly improve the training time and also the quality of results by processing multiple training examples in one iteration or step of an epoch. This is particularly suited for large datasets where the training time with a batch size of 1 (one example per iteration) can be undesirably high. Small batch sizes can exacerbate the training time and so can large batch sizes, along with varying degrees of impact on the quality of the resultant model. Hence, the batch size must be adjusted accordingly.

• Early Stopping Tolerance – Early stopping is a regularization that avoids overfitting and deviation from the minima by providing a tolerance (bound) on the number of iterations that can be run before the model begins to over-fit on the training data and before the training loss begins to increase monotonically. Whereas a high tolerance might be of no use at all, a very low tolerance can lead to premature termination in most of the cases by being too stringent. Hence, the early stopping tolerance must be adjusted accordingly.

• Number of Epochs – If the training doesn't end prematurely due to overfitting or monotonic increase in loss, the model would train for a certain predetermined number of epochs. The maximum number of epochs for which a model is allowed to train must be adjusted so that a near-optimal configuration is reached, without overfitting.

PART 02

TASK SUMMARY

The following MLP architectures are trained, on the preprocessed dataset, 5 times with the learning rates as 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4} and 10^{-5}. The length of the list denotes the number of hidden layers and the list itself consists of the number of nodes in each hidden layer (in the respective order from the input layer to the output layer).

(A) $Hidden\ Layers\ -\ [\ ]$
(B) $Hidden\ Layers\ -\ [2]$
(C) $Hidden\ Layers\ -\ [6]$
(D) $Hidden\ Layers\ -\ [2, 3]$
(E) $Hidden\ Layers\ -\ [3, 2]$

PROCEDURE

Train 5 MLP models for each of the above architectures, with each one of the above 5 learning rates (25 models in total). Report the various details related to training like *training time* and *number of epochs*. Finally evaluate each of the models post-training on the training set and test set and report their respective accuracies and losses.

RESULTS

• ARCHITECTURE A

Learning Rate	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Training Epochs	Training Time
10^{-1}	95.706	93.665	3.2424	4.94064	25	0.131
10^{-2}	98.173	96.228	0.18075	0.46853	140	0.715
10^{-3}	99.051	96.940	0.04211	0.12488	258	1.310
10^{-4}	98.648	**97.011**	0.05868	0.09799	400	2.038
10^{-5}	96.631	95.587	0.11937	0.14934	400	2.060

• ARCHITECTURE B

Learning Rate	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Training Epochs	Training Time
10^{-1}	25.552	24.911	2.38130	2.39130	148	1.006
10^{-2}	25.386	24.911	1.85413	1.87943	63	0.417
10^{-3}	31.056	30.391	1.94052	1.96083	219	1.435
10^{-4}	33.665	**35.445**	1.85955	1.86181	400	2.638
10^{-5}	12.527	13.310	2.52232	2.52341	19	0.124

• ARCHITECTURE C

Learning Rate	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Training Epochs	Training Time
10^{-1}	51.696	53.381	1.76245	1.74457	28	0.183
10^{-2}	62.800	60.071	1.14713	1.21000	55	0.358
10^{-3}	60.285	59.288	1.14123	1.18146	259	1.694
10^{-4}	67.046	**66.192**	1.04393	1.02238	400	2.615
10^{-5}	59.359	60.498	1.29493	1.27350	400	2.611

• ARCHITECTURE D

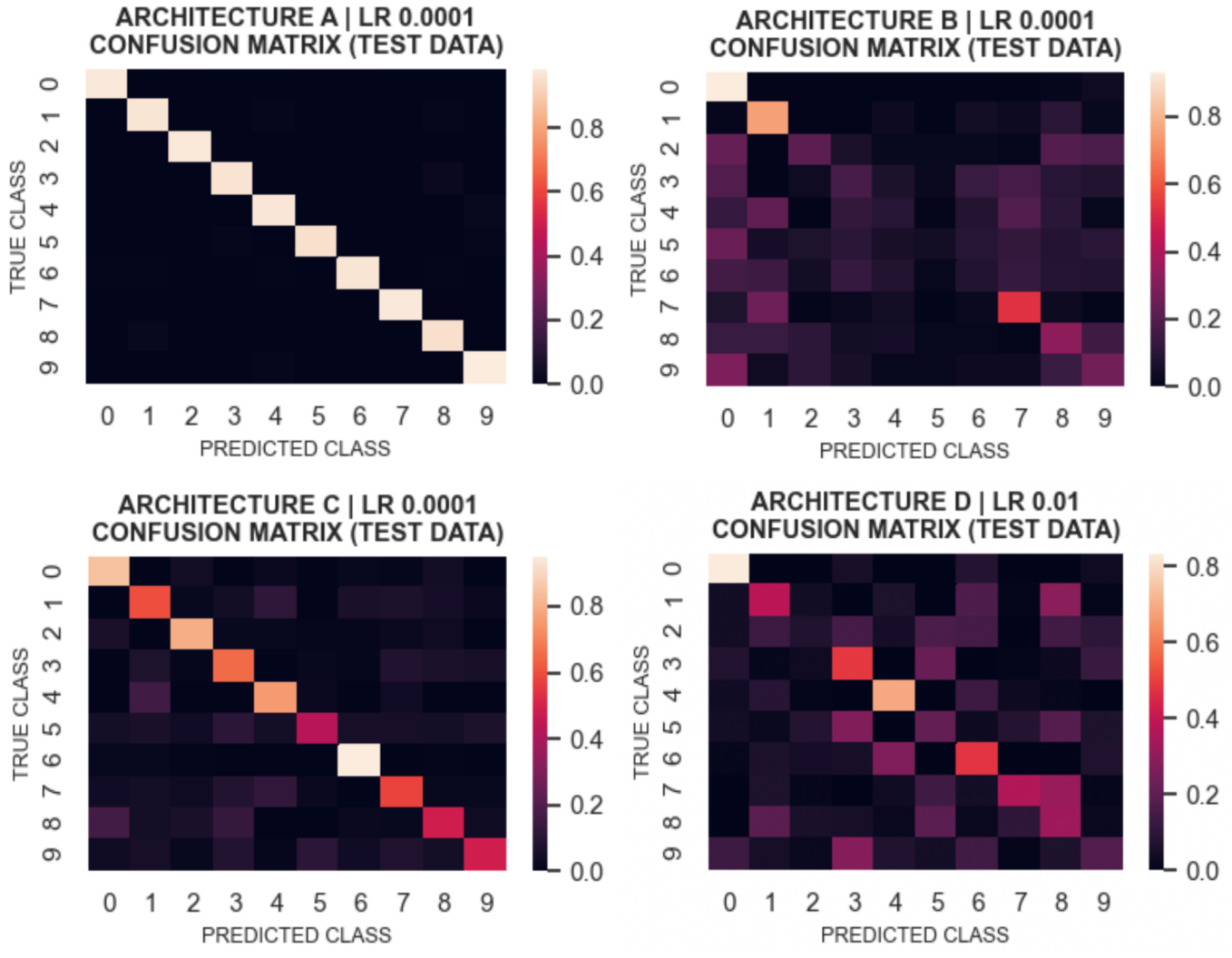
Learning Rate	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Training Epochs	Training Time
10^{-1}	24.057	23.772	2.04321	2.04290	89	0.712
10^{-2}	39.786	**40.498**	1.61620	1.61429	182	1.474

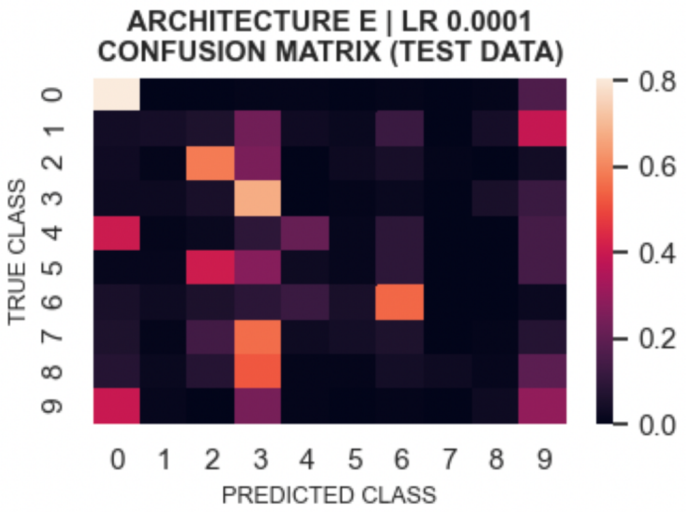
10^{-3}	31.412	32.242	1.89160	1.90526	195	1.576
10^{-4}	25.125	24.128	1.96389	1.97660	400	3.226
10^{-5}	16.702	15.231	2.39823	2.41872	36	0.286

• ARCHITECTURE E

Learning Rate	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Training Epochs	Training Time
10^{-1}	21.756	21.708	2.00306	1.99366	205	1.631
10^{-2}	29.253	29.680	1.84567	1.83232	120	0.958
10^{-3}	30.083	30.463	1.90318	1.89050	270	2.146
10^{-4}	33.476	**32.598**	1.87046	1.87454	400	3.181
10^{-5}	23.986	22.989	2.03159	2.06213	400	3.213

• BEST MODELS OBTAINED FROM VARIOUS ARCHITECTURES





OBSERVATIONS

The quality of the resultant model is extremely sensitive to the underlying architecture of the model, as governed by the number and sizes of the hidden layer(s), and also to the learning rate.

COMMENTS

The entire dataset is split in 3:1 ratio for *training* and *testing*. While training, the training set is split in 4:1 ratio for training and *validation* respectively.

This part deals with only two of the hyperparameters mentioned in the first part – *architecture* and *learning rate*. The remaining hyperparameters are used in their default values of 0.8 for *momentum*, 4 for *early stopping tolerance*, 16 *batch size* and 400 as the maximum allowed *number of training epochs*. These hyperparameters will be tuned and analyzed in the fourth part. Otherwise unless mentioned explicitly, these default values will be used.

The training time wherever mentioned is in $minutes$, unless explicitly specified.

PART 03

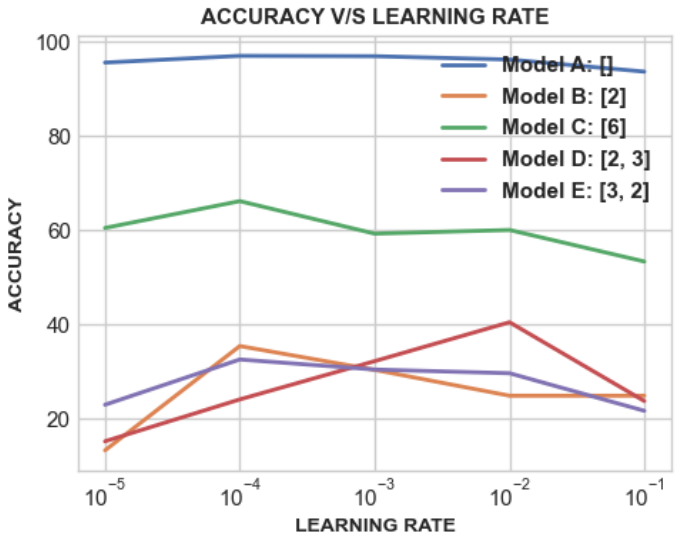
TASK SUMMARY

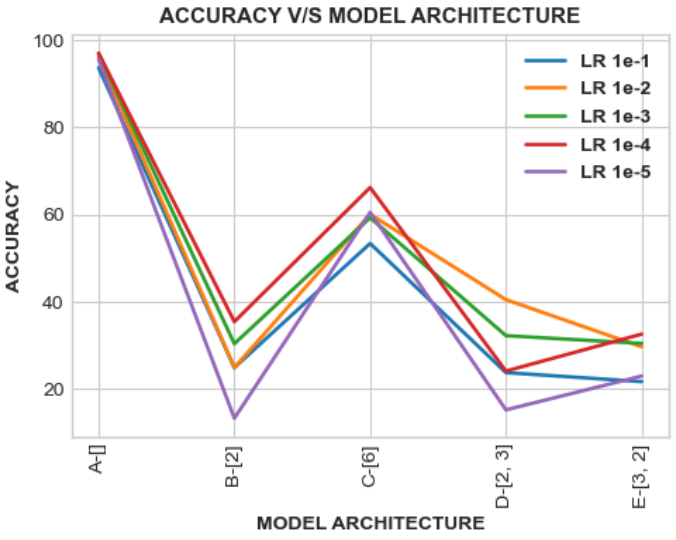
For the results obtained in the second part, graphs of accuracy of each model architecture v/s learning rate and accuracy for each learning rate v/s architecture are plotted.

PROCEDURE

Plot the accuracy of the trained model on the test set v/s the model architecture (number of hidden layers and their sizes), for each learning rate on the same graph. Plot the accuracy of the trained model on the test set v/s the learning rate, for each model architecture on the same graph. Infer from the results the dependence of the quality of the MLP models on the model architecture and try to explain it.

RESULTS





OBSERVATIONS

– Performance of the model architecture A is the least sensitive to the learning rate. Also, its performance is the highest for all learning rates.
– Performance of models B and D is relatively more sensitive to the learning rate and therefore the choice of optimal learning rate for these architectures is more critical.
– Performance of the models B, D and E is consistently bad, with test accuracy at most 40% for all the learning rates. There is a significant improvement in the performance of the classifier when model architecture C is used instead.

INFERENCES

Model A performs exceedingly better than the rest of the models and the performances of the models B, D and E are excruciatingly bad. A precise explanation for these bad performances is the **bottleneck** that is created by the low-dimensional hidden layers in the architecture.
The size of the input layer is as high as 54 and in the models B/D/E, the very next layer in the pipeline has 2 (or 3) nodes. The last layer, that is the output layer, has 10 nodes. So, the size of the layers first decreases drastically, compressing the information that is spread across 54 dimensions into just 2 dimensions, and again then the size increases significantly to disperse this information packed in 2 dimensions among 10 dimensions. This narrow $54 \rightarrow 2 \rightarrow 10$ ($54 \rightarrow 2 \rightarrow 3 \rightarrow 10$ or $54 \rightarrow 3 \rightarrow 2 \rightarrow 10$) passage in the pipeline created by these low-size hidden layers can impact the propagation of information from the input to the output layer and hence slow the training of the model or effect its convergence to an optimal configuration.
There is no such bottleneck in model architecture A. In architecture C too, the bottleneck is not that bad (6 nodes in the hidden layer). Therefore, the performance of architecture A is consistently better than that of C that is consistently better than the performance of architectures B, D and E.

CONCLUSIONS

– Tuning learning rate may not produce drastic improvements in the quality of the model. For example, a poor model with performance less than 20% cannot magically cross 90% even for the most optimal learning rate.
– Though an optimal learning rate can improve convergence and quality of model to some extent, it cannot convert a model with an intrinsically bad architecture into a high performance model.
– Impact of changing learning rate on the accuracy of a model may depend on its architecture.

PART 04

TASK SUMMARY

The best architecture and its optimal learning rate as found from the results of the second part are reported. For an MLP classifier with this best architecture and optimal learning rate, the most suitable values of the other hyperparameters (as specified in the first part) are found and reported.

PROCEDURE

Train MLP classification models with the best architecture found in the second part and its optimal learning rate by varying the following hyperparameters with the corresponding values and keeping the other hyperparameters constant at their default values.

$BATCH\ SIZE\ -\ [2, 4, 8, 16, 32, 64, 128]$

$MOMENTUM\ -\ [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$

$EARLY\ STOPPING\ TOL\ -\ [1, 2, 3, 4, 5, 6]$

$TRAINING\ EPOCHS\ -\ [1\ to\ 800]$

Evaluate each of the above models (20 in total) on the train and test set and deduce the most suitable hyperparameters for the best architecture of the MLP classifier and its corresponding best learning rate. Justify the trends in the performances of the models with varying hyperparameters.

RESULTS

• TUNING BATCH SIZE

Batch Size	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Training Epochs	Training Time
2	98.505	96.370	0.05754	0.11632	92	0.768
4	98.909	96.655	0.04744	0.10786	289	1.874
8	98.671	**97.367**	0.05279	0.09610	400	2.219
16	98.458	96.868	0.05846	0.10282	400	2.014
32	98.078	96.797	0.07052	0.10147	400	1.926
64	97.699	96.726	0.08890	0.12018	400	1.877
128	96.750	95.801	0.11568	0.13644	400	1.853

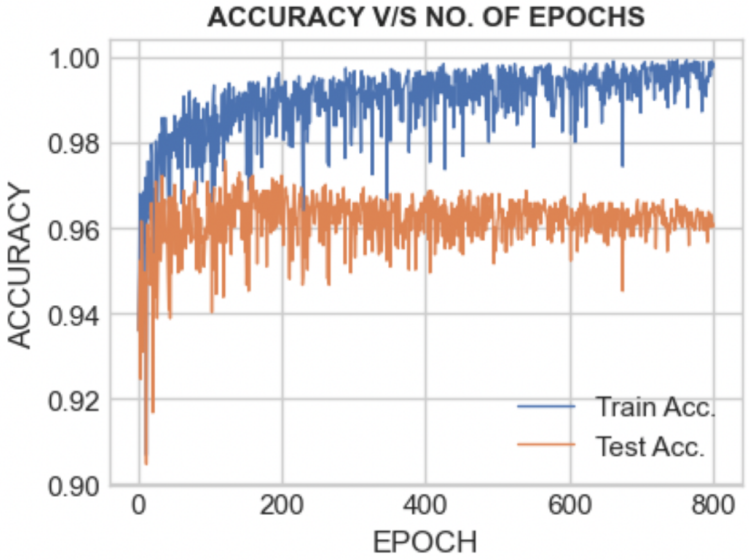
• TUNING MOMENTUM

Momentum	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Training Epochs	Training Time
0.5	98.102	96.441	0.07404	0.11812	400	1.991
0.6	98.292	96.228	0.06828	0.12797	400	2.009
0.7	98.458	96.726	0.06172	0.11840	400	2.006
0.8	98.577	**96.940**	0.05301	0.11243	400	2.025
0.9	98.268	96.370	0.06498	0.12044	127	0.641
1.0	95.231	93.737	1.82476	2.77624	6	0.030

• TUNING EARLY STOPPING TOLERANCE

Early Stopping	Train Accuracy	Test Accuracy	Train Loss	Test Loss	Training Epochs	Training Time
1	94.235	93.523	0.19930	0.21725	14	0.070
2	95.397	95.018	0.15647	0.16910	21	0.108
3	97.438	96.299	0.09294	0.12856	89	0.449
4	98.600	96.584	0.05321	0.11774	400	1.996
5	98.600	96.868	0.05345	0.11275	400	2.022
6	98.505	**96.940**	0.05062	0.11652	400	2.049

• TUNING TRAINING EPOCHS



OBSERVATIONS

• TUNING BATCH SIZE

– As the batch size is increased from 2 to 128, the training time is first observed to increase and then it decreases after reaching a maxima.

– An almost similar trend is observed in the accuracies and the losses of the model on the training and test set. The model seems to become less and less good if the batch sizes become very high (like near 128).

• TUNING MOMENTUM

– The performance of the model is observed to be sensitive to the momentum. The test accuracy of the model first gradually increases with increasing momentum and then starts to decrease.

• TUNING EARLY STOPPING TOLERANCE

– A low early-stopping tolerance naturally causes the model to terminate training prematurely. Hence, the models with lower early stopping tolerance are observed to run for fewer epochs than the models with high tolerance.

– The models with tolerance more than 3 run for the maximum number of epochs (400) and also exhibit better performances than the ones with low tolerances that terminated early.

Note that since the last three models ran for the maximum number of epochs, the experiment can be repeated for them with a higher bound on the *maximum number of training epochs*. We have chosen the default value of early stopping tolerance as 4 because a relatively high tolerance can still cause over-fitting, though it might also depend on the dataset.

• **TUNING NUMBER OF EPOCHS**

The graph shows the variation in the accuracy of the model on train and test datasets as the number of training epochs increase from 1 to 800.

– There is a clear divergence observed between the plots of accuracy on train-set (blue) and accuracy on test-set (orange). The two plots start almost together and as the number of epochs increase, they diverge. This is the evidence for over-fitting on the train set.

– Due to over-fitting caused by the large number of training epochs, the accuracy on test-set starts to decrease (irregularly) after reaching a maximum whereas the accuracy on the train-set continues to increase.

Note that there is a lot of oscillation in the plots but if you observe the average line for each of the two plots, the trends will be clear. Here when we say increase or decrease, we do not mean any monotonic trends in the plots but the approximate trends of their respective average values.

CONCLUSIONS

The architecture and the hyperparameters for the best MLP classifier are as follows.

Model Architecture	A
Learning rate	10^{-4}
Batch Size	8
Momentum	0.8
Early Stopping Tolerance	6
Number of Epochs	121

Architecture A is the one with no hidden layers.

JUSTIFICATIONS

• Architecture – The excruciatingly bad performances of particularly the models B/D/E is because of the bottleneck that is created by the low-dimensional hidden layers in the architecture. The narrow $54 \rightarrow 2 \rightarrow 10$ or $54 \rightarrow 2 \rightarrow 3 \rightarrow 10$ or $54 \rightarrow 3 \rightarrow 2 \rightarrow 10$ passages in the neural network impact the propagation of information from the input to the output layer and hence affect the training and the convergence of the model. Architecture A performs well because of lack of any bottleneck. In architecture C too, the bottleneck is not that bad (6 nodes in the hidden layer).

• Learning Rate – Large training rates result in unstable training and small learning rates converge very slowly. Practically too a similar trend was observed. Refer to the plot of ACCURACY V/S LEARNING RATE in the third part. The performance of the model is first observed to increase with learning rate but then decreases (sometimes sharply) when learning rates become high. So an optimum learning rate is somewhere in-between, that is 10^{-4} for architecture A.

• Batch Size – Small batch sizes have large training time because each training sample is processed separately, one at a time. So if multiple samples are processed in mini-batches, the performance of the model and so as the training efficiency increases. But if the batch size

becomes too large, the time to process each batch becomes very high, hence increasing the training time. So for this too, the optimum value lies somewhere in-between.

• Momentum – Low values of momentum do not deal with the abrupt changes in the gradient in consecutive iterations. On the other hand, high values of momentum increase the contribution of the Δw of previous step, in the Δw of the current step, hence over-shadowing the gradient term. So the optimum value should be neither too high nor too low.

• Early Stopping Tolerance – A high tolerance serves no purpose at all in controlling the over-fitting of the model with increasing number of epochs. A very low tolerance leads to premature termination, even without overfitting, by being too stringent; hence restraining the model from getting trained properly. So the optimum value is neither too high nor too low.

• Number of Epochs – If a model is trained for a very large number of epochs, it can cause overfitting, hence reducing the performance on test (or validation) set(s). Therefore, the optimum number of epochs for which the model is trained should not be very high.

PART 05

TASK SUMMARY

The dimension of the feature vectors is reduced to 2 using *Principal Component Analysis*. The reduced dimensional data is plotted on a Cartesian plane with all data points of the single class colored the same and data points of different classes differently colored.

PROCEDURE

Build the covariance matrix on the 54 attributes of the samples, using only the training samples and from that derive the projection matrix (or weights) to reduce the dimension of the features from 54 to 2. Apply these weights on all the samples in the dataset (both training and testing) and transform them into a new dataset with 2-dimensional feature vectors, and the same class labels. Assign a unique color to each of the 10 classes (without repetition) and plot the data points from each class in the following ways.
– Separate plot for data points of each class.
– Plots for data points of multiple classes partitioned in *proper* groups.
– Plot for data points from all the classes.

RESULTS

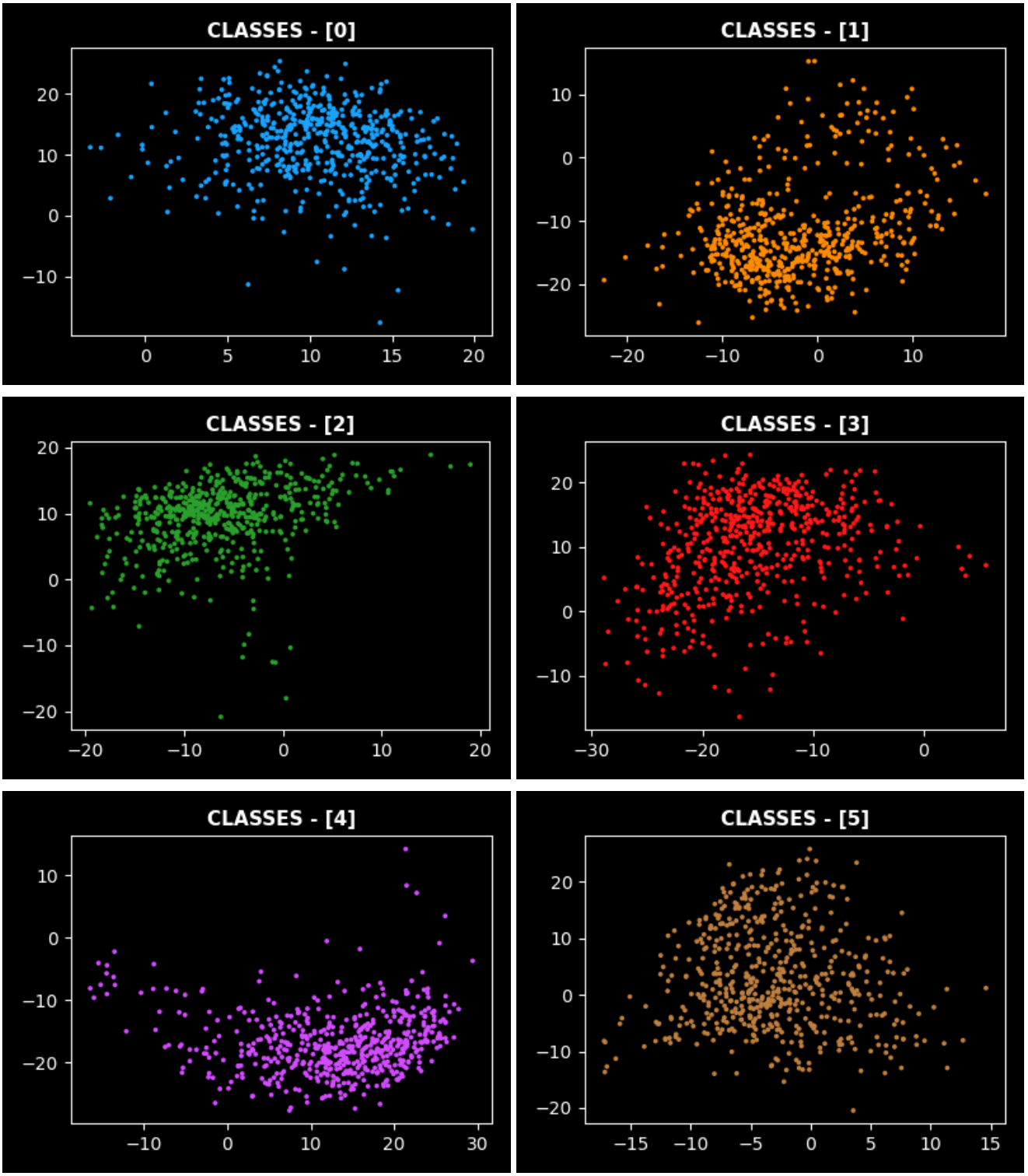
• EIGENVALUES OF THE COVARIANCE MATRIX

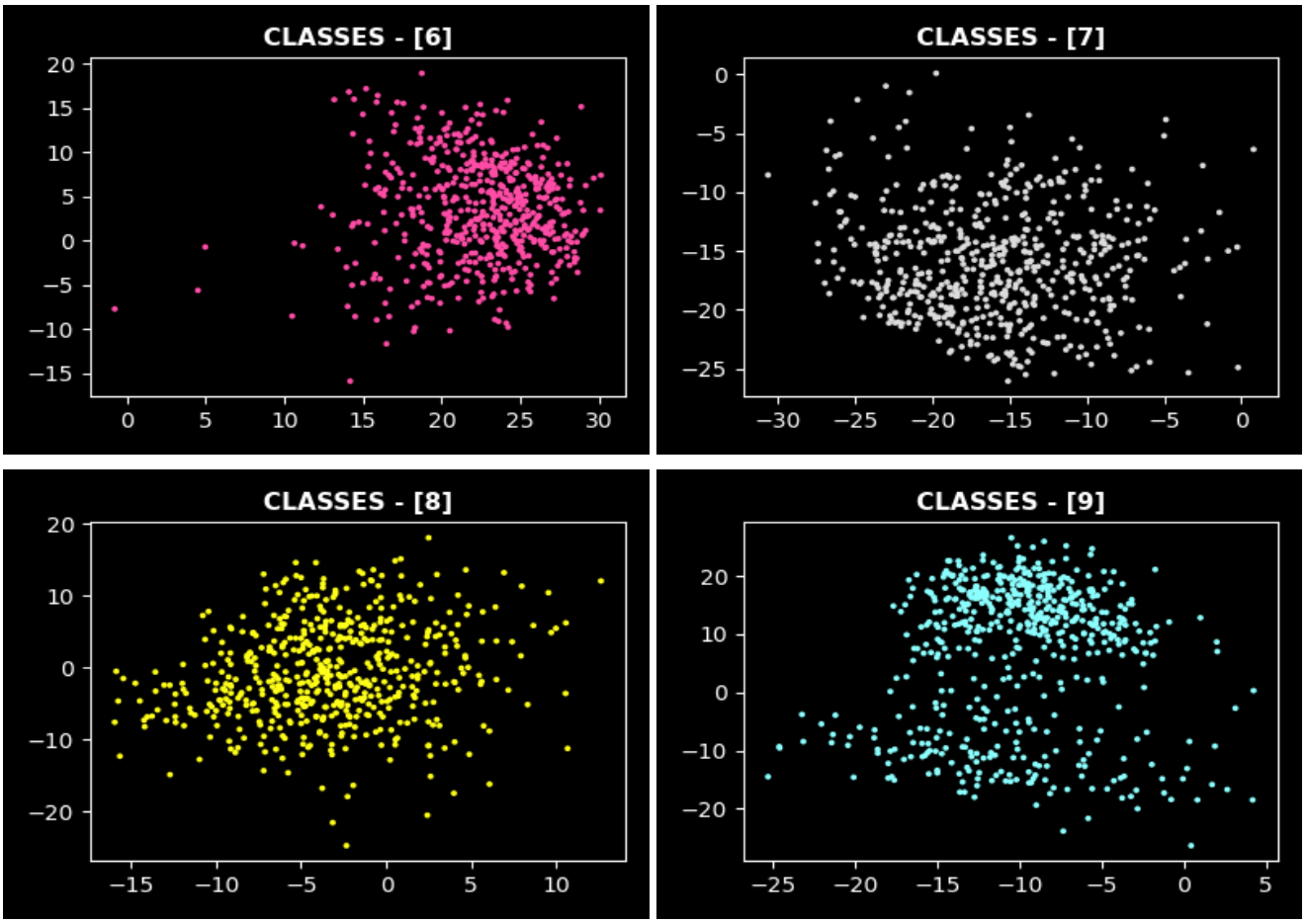
Following are the 54 eigenvalues of the covariance matrix of the 54 attributes of the features. They are ranked from 1 to 54 in descending order.

1	175.800	10	37.902	19	11.995	28	6.225	37	3.203	46	1.273
2	162.306	11	28.604	20	11.669	29	5.643	38	3.046	47	1.097
3	141.801	12	26.946	21	10.903	30	5.164	39	2.833	48	1.045
4	101.327	13	22.567	22	9.872	31	4.802	40	2.736	49	0.689
5	68.527	14	19.980	23	9.133	32	4.364	41	2.357	50	0.376
6	60.860	15	17.436	24	8.855	33	4.219	42	2.132	51	0.230
7	54.028	16	16.943	25	8.180	34	4.003	43	1.877	52	0.110

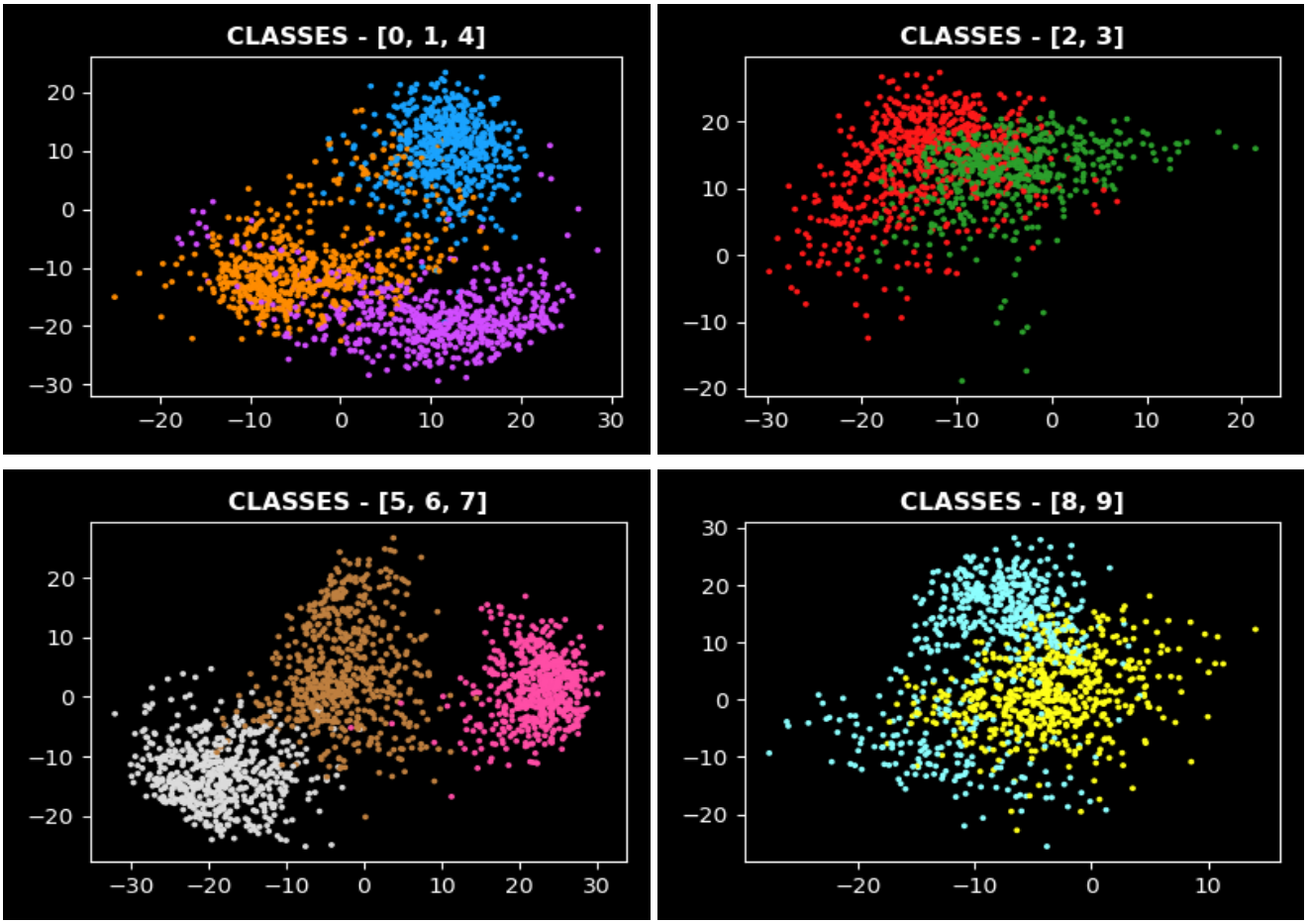
8	43.646	17	15.232	26	7.460	35	3.916	44	1.645	53	0.071
9	42.287	18	14.910	27	6.846	36	3.737	45	1.504	54	0.040

• SCATTER PLOTS FOR INDIVIDUAL CLASSES

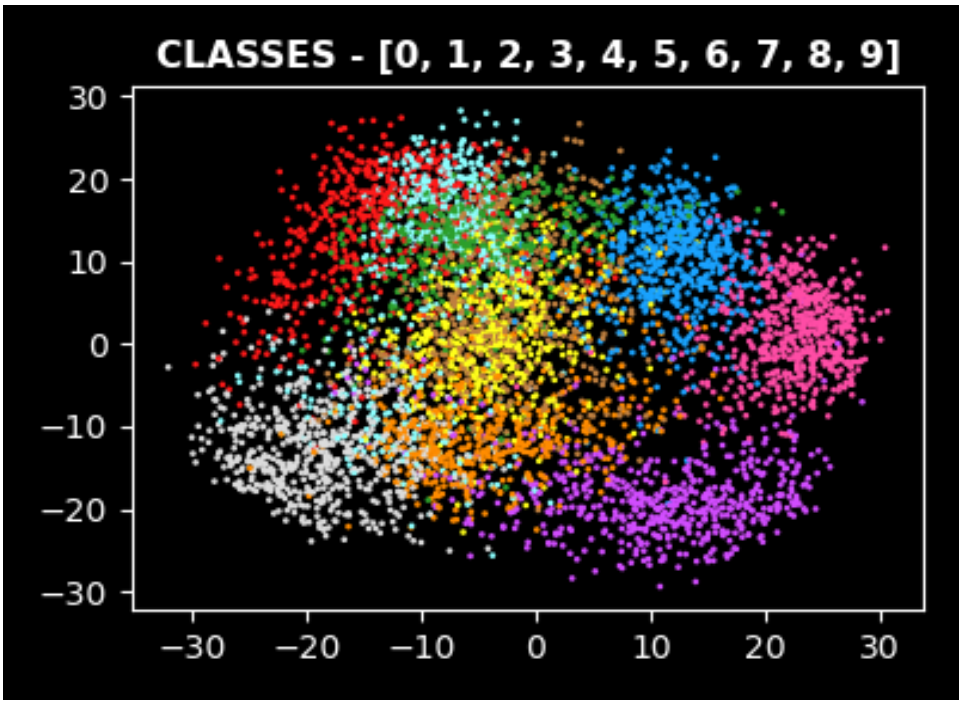




- **SCATTER PLOTS OF CLASSES IN GROUPS**



• **SCATTER PLOT OF ALL CLASSES**



OBSERVATIONS

– The density of data points of the same class is high for most of the classes and clusters can be observed quite distinctively. Even after reducing the dimensions to 2, the semantic similarity among data points of the same class is effectively captured. However, there are few to many outliers in the data samples for some of the classes.

– The clusters were observed to have mostly *spherical* shapes. But the outliers can distort the shape of the cluster, like for the samples of class 9 and class 1.

– Clusters of some pairs (or groups) of classes are very prominently separated, like [0, 4] and [5, 6, 7]. Some groups of classes witness a considerable amount of overlap among their classes, like [2, 3] and [8, 9].

– In the last scatter plot, where data points from all the classes are plotted, many of the 10 classes can be prominently differentiated from the others. Though there is some overlap among the classes, they can still be spotted quite distinctly, which is a good achievement considering that the dimensions were reduced from 54, straight to 2.

COMMENTS

Though in the assignment it was asked to plot only the last graph (with all the data points) but it was important to first plot data points from different classes separately and then in small groups to check the shape of the clusters, the density of points of the same class and the overlap between some classes. This analysis would have not been possible through the last plot.

PART 06

TASK SUMMARY

For each model architecture in the second part, an MLP classifier is trained on the dataset with reduced dimensionality of the features, using the best observed learning rate in the second part as the learning rate for the respective model architecture. The performance of the models obtained on the reduced dataset is compared with the earlier performance.

PROCEDURE

Train 5 MLP classifiers on the training dataset with reduced dimensions, each with a different model architecture and the best learning rate for that architecture. According to the results of the second part, the 5 models should have the following hyperparameters.

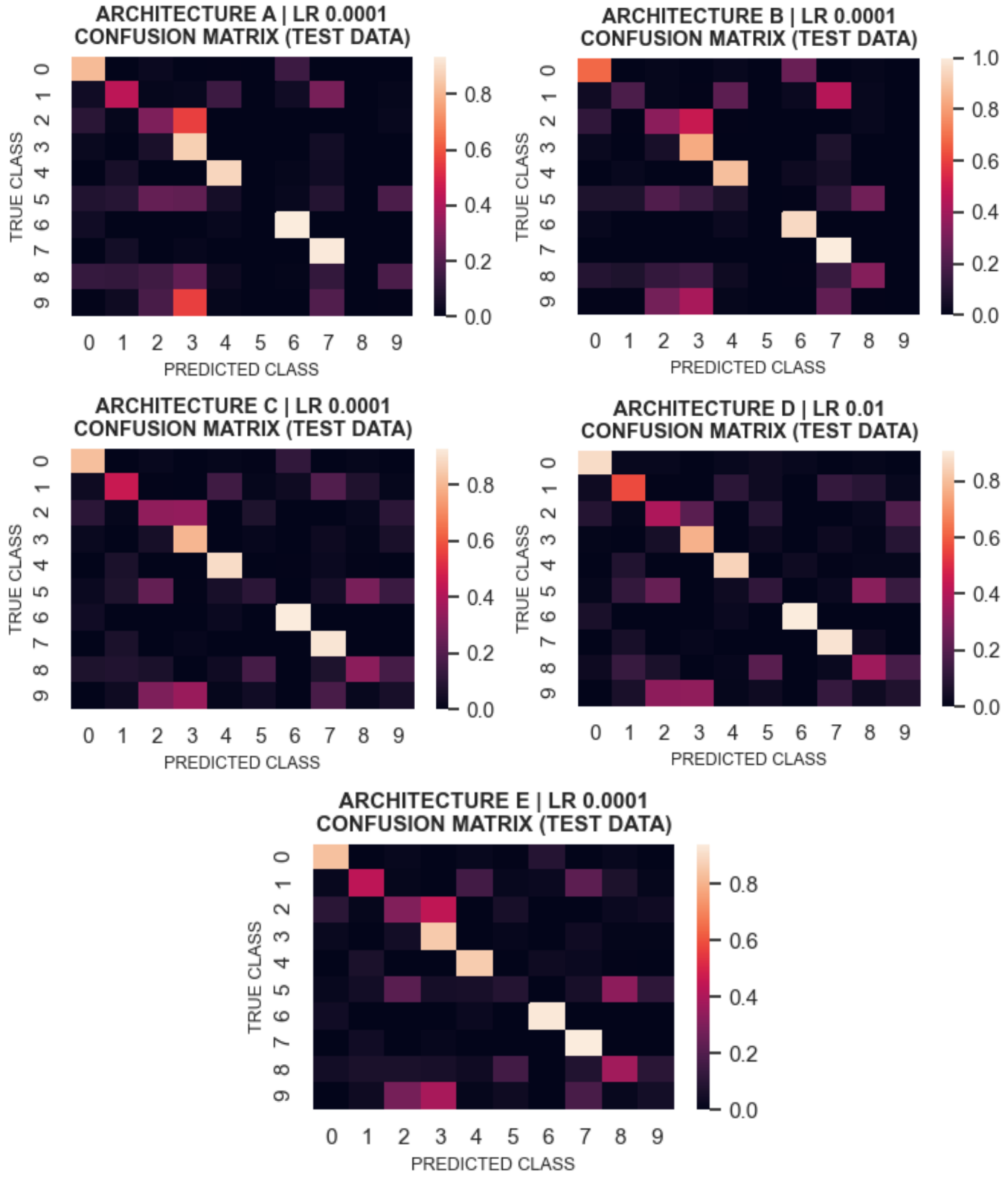
Model	Model Architecture	Learning Rate
01	A	10^{-4}
02	B	10^{-4}
03	C	10^{-4}
04	D	10^{-2}
05	E	10^{-4}

Compute the losses and accuracies of the models on training and testing datasets and also their training times and number of training epochs. Compare the results obtained for each model architecture with the results obtained on the original dataset with the same hyperparameters.

RESULTS

Arc.	**LR**	**Train Accuracy**		**Test Accuracy**		**Train Loss**	
		54D	2D	54D	2D	54D	2D
A	10^{-4}	98.648	51.673	97.011	50.819	0.05868	1.29258
B	10^{-4}	33.665	51.720	35.445	51.174	1.85955	1.32960
C	10^{-4}	67.046	58.102	66.192	55.730	1.04393	1.13433
D	10^{-2}	39.786	59.644	40.498	57.224	1.61620	1.05428
E	10^{-4}	33.476	56.987	32.598	55.943	1.87046	1.14874
Arc.	**LR**	**Test Loss**		**Training Epochs**		**Training Time**	
		54D	2D	54D	2D	54D	2D
A	10^{-4}	0.09799	1.31385	400	97	2.038	0.462
B	10^{-4}	1.86181	1.34624	400	160	2.638	0.989
C	10^{-4}	1.02238	1.15921	400	400	2.615	2.496
D	10^{-2}	1.61429	1.08120	182	125	1.474	0.962
E	10^{-4}	1.87454	1.17094	400	400	3.181	3.066

• MODELS TRAINED ON 2D (ORHD) DATASET



OBSERVATIONS

– The performances of the bad-performing models with the architectures B, D and E are significantly improved when trained on the reduced dataset.

– One of the least performing model architectures, architecture D, now performs the best among all other architectures on the reduced dataset.

– Though the performances of the architectures B, D and E have improved, the performances of the architectures A and C have reduced by almost 47% and 10% respectively.

– The accuracies on the training dataset have also significantly improved for the architectures B, D and E, signifying a better training.
– Unlike in the second part, the performance of the MLP classifier is more uniform across different model architectures, with less variation. All the model architectures have much closer performance than they did for the original dataset. The confusion matrices on the test set for the trained models are also very similar for all the architectures signifying that the performance has become less sensitive to the model architecture.

JUSTIFICATIONS

Refer to the table in the RESULTS section of the fifth part. There the 54 eigenvalues of the covariance matrix of the 54 attributes are ranked in descending order. When we reduce the dimension from 54 to 2, only the first and the second principal components are selected, along which the variances are 176 and 162 respectively. The variance of the data points along the third principal component is the third eigenvalue, i.e, 142, that is of the same order as the first two eigenvalues. The fourth eigenvalue is around 101 which is also of almost the same order.
Reducing 54-dimensional features to 2 dimensions by selecting only the first and second principal components neglects the other principal components along which the variance of data points is lower but still considerably high. As a result of this, a considerable amount of information in the original dataset is lost. This naturally can hamper the performance of the models trained on the reduced dataset.
This effect is seen in the models with architectures A and C, where the performance is significantly reduced. Though the other three model architectures perform better than before, the overall best accuracy observed (57%) is far less than the best accuracy observed earlier (97%). Nevertheless, the model architectures B, D and E now have an input layer of size 2 and hence there is no bottleneck in their neural networks.

SOLUTIONS

– The hyperparameters can be tuned again for this new dataset (repeat fourth part on this dataset). An optimal choice of the hyperparameters may increase the performance.
– Two dimensions may be less considering the high values of the subsequent (third and fourth) eigenvalues. The dimensions should be reduced to at least 4 (select first four principal components) and that might increase the performance.
– The dimension of the reduced dataset can itself be treated as a hyperparameter. Increase the dimensions from 2 (towards 54) till a particular threshold on the test accuracy is not crossed.

APPENDIX

ABBREVIATIONS

%AGE	Percentage
ACC	Accuracy
ARC	Architecture
LR	Learning Rate
ML	Machine Learning
MLP	Multi Layer Perceptron
ORHD	Optical Recognition of Handwritten Digits
STDDEV	Standard Deviation

CODE EXECUTION

- **requirements.txt**
Specifies the python packages and modules required to run the code.
Ensure all the necessary dependencies of required versions and latest version of Python3 are available with the following command

```
>>> pip3 install -r requirements.txt
```

- **optdigits.tes**
Contains the testing examples of the *Optical Recognition of Handwritten Digits* dataset.

- **optdigits.tra**
Contains the training examples of the *Optical Recognition of Handwritten Digits* dataset.

- **GRP23_A03_IMPLEMENTATION.ipynb**
Jupyter Notebook written in Python containing all the code relevant to the assignment.

Launch jupyter notebook through terminal using the command: `jupyter notebook`
Then open this notebook to view the code written and run to see the output results.

Note: Please make sure before running the notebook, the concerned files for *Optical Recognition of Handwritten Digits* dataset are present by the names of *optdigits.tes* and *optdigits.tra* in the same directory as *GRP23_A03_IMPLEMENTATION.ipynb.*

(other details are present in README.txt)

REFERENCES

- Heatmaps on Seaborn Python
- Matplotlib Python Documentation
- Numpy Python Documentation
- Optical Recognition of Handwritten Digits
- PyTorch Documentation
- Visual Paradigm Online