DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING, IIT MADRAS

CS6910: Programming Assignment 2

Team 37

Sheth Dev Yashpal
CS17B106

Harshit Kedia
CS17B103

1 TASK 1 - DIMENSION REDUCTION

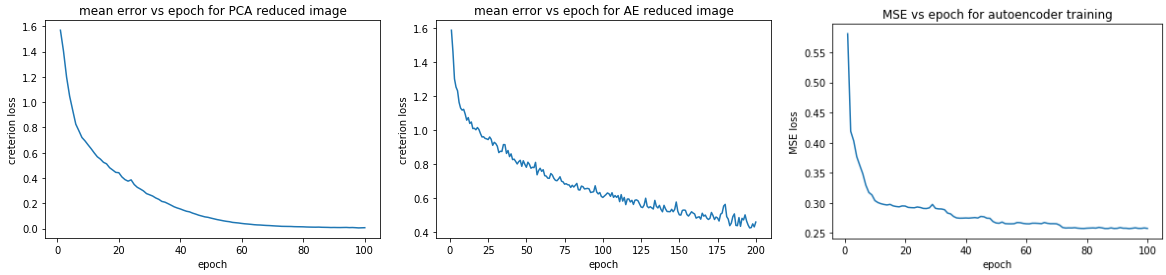

Figure 1.1: Training error curves vs epoch number

	0	1	2	3	4
0	291	0	0	0	0
1	0	246	0	0	1
2	0	0	230	0	0
3	0	0	0	324	0
4	0	0	0	0	288

	0	1	2	3	4
0	34	9	0	20	6
1	4	37	7	4	9
2	1	16	38	4	3
3	15	5	0	61	5
4	4	10	7	4	43

Figure 1.2: Confusion Matrices for the model with dimensionality reduction using PCA followed by MLFFNN for training and validation data respectively. Accuracy's are **99.93%** and **61.56%** respectively.

	0	1	2	3	4
0	224	4	1	64	2
1	3	221	4	6	7
2	1	19	201	1	12
3	40	7	1	274	5
4	4	13	9	11	246

	0	1	2	3	4
0	43	7	1	11	3
1	2	43	7	2	13
2	0	9	44	0	5
3	15	2	1	60	5
4	7	14	4	2	46

Figure 1.3: Confusion Matrices for the model with dimensionality reduction using AANN followed by MLFFNN for training and validation data respectively. Accuracy's are **84.49%** and **68.2%** respectively.

Configuration and Observations:

- From 828 dimension vector, we brought it down to 64 dimensions using PCA and AANN. Then we feed this vector as input to a MLFFNN to classify Data-set 1 with hidden layer sizes 32 and 16 using ReLU as activation function for the neurons, followed by output layer of size 5 (number of classes). Moreover, we split the data randomly in ratio 80:20 for training and validation.

- Used learning rate of 0.001 for classifier network training in case of PCA and 0.003 for the same after AANN, both having batch size of 32. Also, trained the dimension reducing Autoencoder with learning rate of 0.004 and batch size 32. For all cases we have used the Adam optimizer with default hyper-parameters.
- Despite having lower train accuracy for AANN, results on validation data are better for AANN than PCA (both having similar classifier network structure), this reflects that Auto-encoder learns better features than the transformation done by PCA.
- Even after training the second network for 200 epochs, we see the overall error going down slowly, unlike first case, where error almost saturates around 80 epochs. If we train the second network for more epochs, we observed the training accuracy to increase, but the validation accuracy took a hit in that case. Also, increasing the learning rate causes more jumps in the tail of the error v epoch graph, which is already very unstable.

2 Task 2 - Stacked Auto-encoder Dataset 1

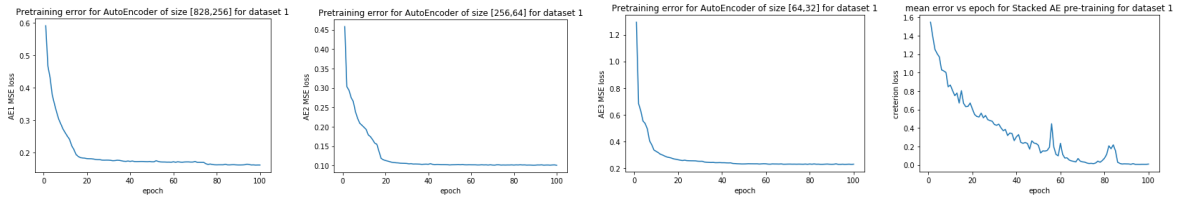

Figure 2.1: Loss v Epoch plots for training three autoencoders and fine-tuning the pre-trained network over Dataset 1.

	0	1	2	3	4
0	290	0	0	0	0
1	0	239	0	0	0
2	0	0	222	0	0
3	0	0	0	337	0
4	0	1	0	0	291

	0	1	2	3	4
0	46	7	6	7	4
1	7	40	8	2	12
2	2	8	55	1	4
3	9	2	8	49	5
4	8	13	5	5	33

Figure 2.2: Confusion Matrices for the Stacked Autoencoder model on Dataset 1. Accuracy's are **99.93%** and **64.45%** respectively.

Configuration and Observations:

- Stacked 3 Auto-Encoders, of size (828,256), (256,64), (63,32) in-order. Then recursively copied the individual AE parameters into the layers of a Multi-layer feed forward neural network, followed by the output layer of dimension 5. The input vector was of dimension 828.
- Used learning rate of 0.002 for Auto-encoder training, and 0.001 for fine tuning the neural network. Batch size is kept 32 everywhere. We can see that graphs of error vs epoch of all three auto-encoders are very smooth, signifying that learning rate is not too high, but high enough to steeply reduce the error within a few epochs.
- While fine-tuning the stacked Auto-encoder parameters, it is very important to maintain a very low learning rate otherwise the pre-trained weights will be changed drastically and the representation learned will be forgotten. Even with a slower learning rate, we can see that the Neural Network takes way more epochs than Auto-encoders for errors to saturate, yet having plenty of spikes in the plot.

- Using stacked Autoencoder resulted in very high training accuracy, and not so significant improvement on the validation accuracy. Data-set 1 having less number of examples, and also having to split that into train and validation sets, might have lead to over-fitting on train examples.
- Also experimented by training a simple MLFFNN (having same number of layers and nodes) for Dataset 1 to compare results, achieved training and validation accuracy of **99.23%** and **67.34%** respectively. Clearly, we have a case of over-fitting on train data on either case.

3 Task 3 - Stacked Autoencoder Dataset 2

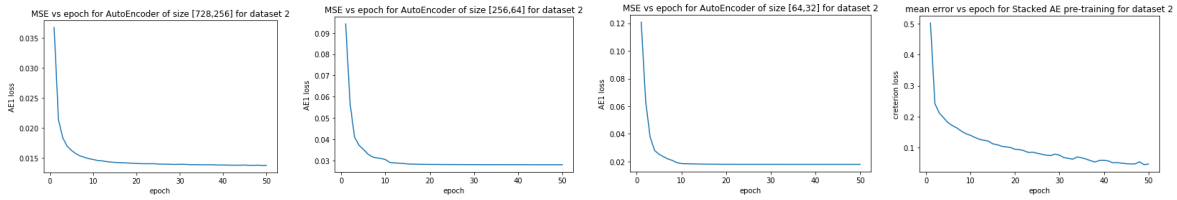

Figure 3.1: Loss v Epoch plots for training three autoencoders and fine-tuning the pre-trained network over Dataset 2.

	0	1	2	3	4
0	5002	1	1	0	0
1	0	4892	107	0	0
2	0	250	4749	0	0
3	0	0	0	4999	0
4	0	0	0	0	4999

	0	1	2	3	4
0	980	4	10	6	4
1	3	909	86	1	0
2	1	134	863	1	0
3	2	0	1	977	19
4	1	0	0	19	979

Figure 3.2: Confusion Matrices for the Stacked Autoencoder model on Dataset 2. Accuracy's are **98.56%** and **94.16%** respectively.

Configuration and Observations:

- Stacked 3 Auto-Encoders, of size (784,256), (256,64), (64,32) respectively. Then recursively copied the individual AE parameters into the layers of a Multi-layer feed forward neural network, followed by the output layer of dimension 5 (similar to task 2).
- Data-set 2 had exactly 6000 examples for each of the 5 classes, so we split classes individually into 5000 examples for training and 1000 validation examples for each class. The input vector is of dimension 784 (28x28). Used batch size of 100 for training both auto-encoders and final neural network.
- Used learning rate of 0.001 for Auto-encoder training, and 0.0005 for fine tuning the neural network. We can see that graphs of error vs epoch of all three autoencoders are very smooth. Also having so many examples leads to error minimization in very less number of epochs (in comparison to data-set 1).
- Similar to task 2, we keep learning rate lower during fine tuning of the final classifier network. Despite having a bigger batch size (100 vs 32), we see faster convergence for the network. As expected, from having abundant training examples, we get very high training and validation accuracy.
- Also experimented by training a simple MLFFNN (having same number of layers and nodes) for Dataset 2 to compare results, achieved training and validation accuracy of **98.21%** and **93.7%**

respectively. Clearly, we are getting a slight improvement when pre-training using autoencoders, for both training and validation.

4 Task 4 - Stacked RBM (binary-binary) Dataset 2

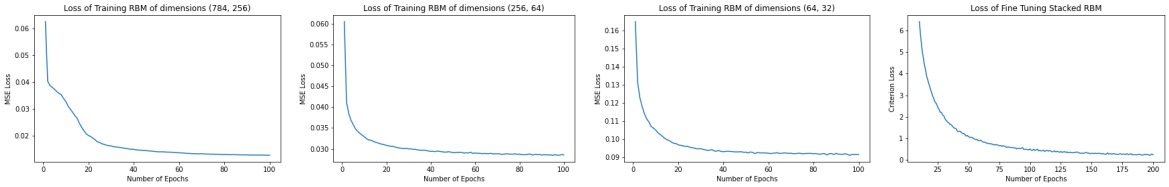

Figure 4.1: Loss v Epochs plots for training of pre-training three RBMs and fine-tuning the Stacked RBM over Dataset 2

	0	1	2	3	4
0	4717	27	34	4	0
1	2	4821	0	0	0
2	296	25	4259	245	0
3	19	53	246	4457	0
4	0	0	0	0	4795

	0	1	2	3	4
0	1168	11	21	17	1
1	7	1166	2	2	0
2	74	10	965	126	0
3	20	24	147	1034	0
4	0	2	0	1	1202

Figure 4.2: Confusion Matrices for the Stacked RBM model on Dataset 2. Accuracy's are **96.04%** and **92.25%** respectively.

Configuration and Observations:

- The configuration for training RBMs is kept same as that of Autoencoders so that we have a direct comparison across both the learning representations. Therefore we train three RBMs of size (784, 256), (256, 64), (64, 32) for this task. The first RBM is real-binary and others are binary-binary. The batch size is kept at 32 while the learning rate is kept at 0.01 and the **k** value of contrastive divergence step is 2.
- Increasing batch size leads to requirement of more number of epochs while decreasing leads to unstable training. In case of learning rate, increasing it decreases stability of training and leads to a sub-optimal solution while decreasing means very high number of epochs for convergence. All three RBMs are trained for 100 epochs. The parameter **k** was chosen as 2 as it provided some stability to learning over the value 1 and higher values of k were ruled out due to higher computation requirement.
- For the final classifier, again like autoencoders all weights are loaded in the MLFFNN and that is fine tuned for 200 epochs with batch size 32 and learning rate 0.001. We observe that the training and validation accuracies obtained are comparable to those obtained for autoencoder and MLFFNN. As there is already so much data available, we expected all the three approaches to give similar results. For RBMs we had 6000x5 examples which were randomly split into training validation sets of 80 and 20 percent respectively.

5 Task 5 - Stacked RBM (gaussian-binary) Dataset 1

Configuration and Observations:

- Again, the configuration for training RBMs is kept same as that of Autoencoders. Therefore we train three RBMs of size (828, 256), (256, 64), (64, 32) for this task. The first RBM is gaussian-binary and others are binary-binary. The batch size is kept at 32 while the learning rate is kept

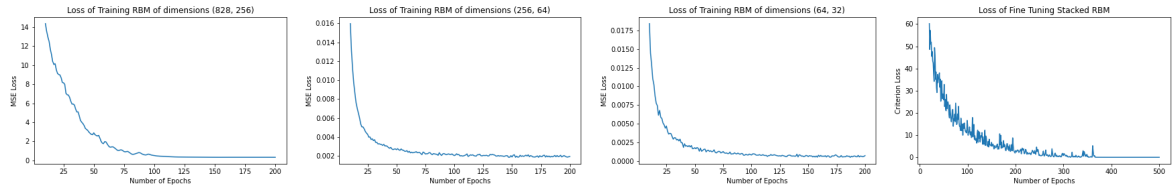

Figure 5.1: Loss v Epochs plots for training of pre-training three RBMs and fine-tuning the Stacked RBM over Dataset 1

	0	1	2	3	4
0	285	0	0	0	0
1	0	255	0	0	0
2	0	0	318	0	0
3	0	0	0	233	0
4	0	0	0	0	289

	0	1	2	3	4
0	45	1	20	4	5
1	4	30	3	8	8
2	12	4	66	1	9
3	3	8	2	42	4
4	7	6	7	2	45

Figure 5.2: Confusion Matrices for the Stacked RBM model on Dataset 1. Accuracy's are **100.0%** and **65.90%** respectively.

at 0.001 and the **k** value of contrastive divergence step is 2. All three RBMs are trained for 200 epochs and the learning curved and confusion matrices along with final accuracies are shown above.

- It is particularly difficult to train the gaussian binary RBMs as you have to account for the variance in the data. At some pixels, the variance is drastically low and as it occurs in the denominator of updates, this leads to highly unstable training process. Therefore, we resorted to clamping the variance of the data in the range (0.01, 1) in order to stabilize our training process.

- As you can see again our final accuracies are similar to that of autoencoders or simple MLFFNN or even the PCA or AANN based dimensionality reductions. But since the data available is already less, which was further split into a 80-20 percent training-validation sets, we tend to overfit our final classifier achieving 100% train accuracy but only 65% validation accuracy.

- The observations regarding the learning rate and batch sizes hold here as well i.e. decreasing the batch size or increasing the learning rate will lead to unstable training. Our classifier was trained for a whopping amount of 500 epochs which is part of the reason it over-fits perfectly. But having lesser number of epochs meant that we lose out on both the training and the validation accuracies. In all of our experiments for the assignment we used the Adam optimizer with default parameters and ReLU activation function wherever required.