DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING, IIT MADRAS

CS6910: Programming Assignment 2

Team 37

Sheth Dev Yashpal
CS17B106

Harshit Kedia
CS17B103

1 TASK 4 - STACKED RBM (BINARY-BINARY) DATASET 2

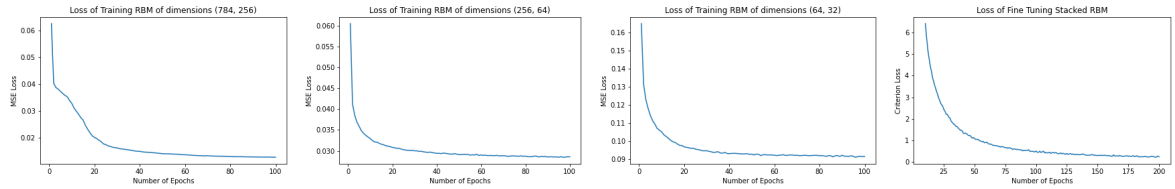

Figure 1.1: Loss v Epochs plots for training of pre-training three RBMs and fine-tuning the Stacked RBM over Dataset 2

	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 1.2: 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.

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

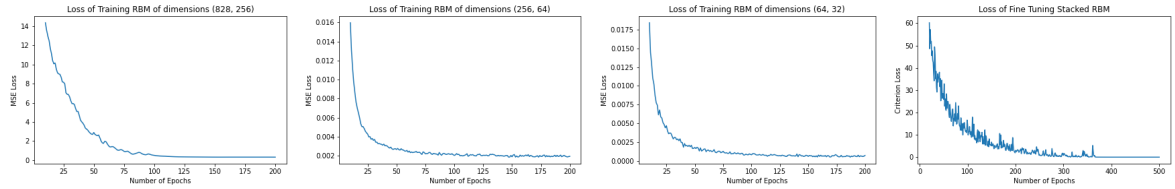

Figure 2.1: Loss v Epochs plots for training of pre-training three RBMs and fine-tuning the Stacked RBM over Dataset 1

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

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 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.