Machine Learning Assignment-2

Hetvi Jethwani
Maths and Computing
Indian Institute of Technology, Delhi
mt1180754@iitd.ac.in

Shreyansh Choudhary
Maths and Computing
Indian Institute of Technology, Delhi
mt6180794@iitd.ac.in

***Abstract*—This report comprises the results of the various experiments we did on the provided data-sets to obtain best fit models. We have used various techniques to predict the pattern in data, explained the results.**

***Index Terms*—SVM,Kernel Machines,**

I. Question1

A. Problem Statement

Given Medical Health Data-set. We have to train a binary SVM classifier, explore different kernels and find the best model among them. We have used LibSVM[1] library as instructed for our work.

B. Data Pre-processing

We observed that the different measures lies in different ranges. So to avoid problems like underflow/overflow/floating point Errors, we have normalised the data to lie in the same range. Data is transformed according to LibSVM format. Data is shuffled randomly and split into test and training sets(300-400). For curves depicting decision boundaries we have first Projected the data to a 2D PCA space and then find the classifier.

C. Metrics Used

- Accuracy

For all the plots:

– yellow x denotes positive samples
– green x denotes positive support vectors
– blue o denotes negative samples
– pink o denotes negative support vectors

D. Different Kernels

- Linear SVM
  For different values of C we have recorded the train and test accuracy to find the best Linear SVC.

Model Performance for 3d Data		
C	Train Accuracy	Test Accuracy
0.001	0.565	0.58
0.03	0.8275	0.8533
0.6	0.8525	86.33
0.1	0.85	0.8667
1	0.8575	0.8767
30	0.86	0.8533
600	0.855	0.86
1000	0.855	0.86

  As we can see, the lower the C value, the lower is the train and test accuracy. Also, if we increase C too much, The pattern in the data is lost and errors in training data is reduced significantly and model performs comparatively ill on the test data.

Model Performance for 2d Data		
C	Train Accuracy	Test Accuracy
0.001	0.565	0.58
0.03	0.825	0.8533
0.6	0.8525	86.33
0.1	0.85	0.8633
1	0.8575	0.8533
30	0.86	0.8533
600	0.86	0.8533
1000	0.86	0.8533

  There is an important observation here, that is, the PCA approximates the 3d data into 2d data so well, we got nearly the same figures in 2d data as well. We present 3 plots here, the best test accuracy SVC and 2 extreme case classifiers. Hence, from now on we will present Accuracy values for 2d PCA data only

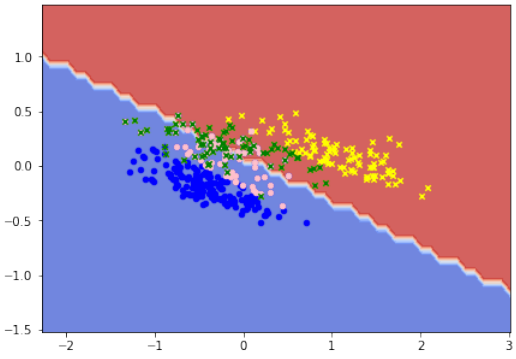

Fig. 1. Linear SVC with C=0.6, best Case

No. of Support Vectors vs C	
C	No. of Support Vectors
0.001	348
0.6	156
1000	120

  As we can see, the lower the C is, more misclassifications are allowed and performance is poor.
  Plot of best Linear SVC without support vectors:
- Polynomial SVC.
  For different Degrees and different values of C, we have recorded the train accuracy and test accuracy.

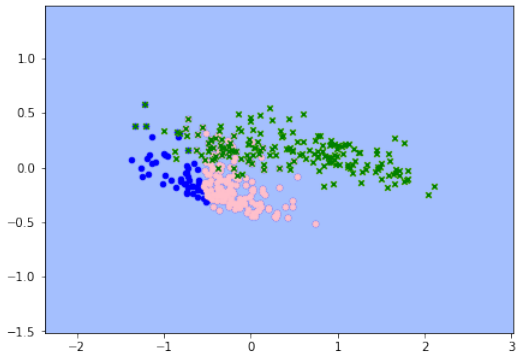

Fig. 2. Linear SVC for C=0.001, here all the vectors of class 1 are wrongly classified.

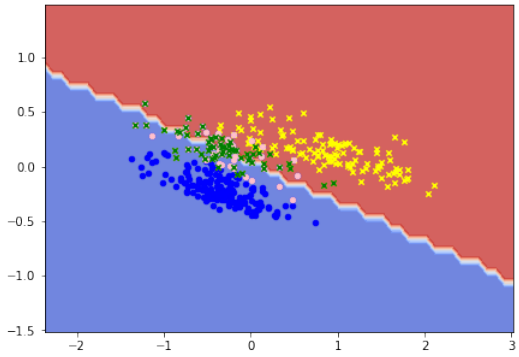

Fig. 3. Linear SVC for C=1000

Model performance for various Values of C and degrees.			
C ↓, degree →	2	3	4
0.001	0.565, 0.58	0.565, 0.58	0.565, 0.58
0.03	0.6275, 0.6433	0.6325, 0.6433	0.6275, 0.6167
0.6	0.67, 0.6833	0.7275, 0.7233	0.6525, 0.68
0.1	0.6425,0.6833	0.66, 0.69	0.635, 0.65
1	0.675,0.68	0.7375,0.7367	0.67,0.655
30	0.6775, 0.6733	0.8,0.82	0.6766, 0.6775
600	0.685,0.6733	0.82, 0.8233	0.6766 , 0.6875
1000	0.685,0.6733	0.83, 0.8433	0.6766 , 0.6875

Here, we can notice that degree 3 polynomial is able to generalise the data better than both degree 2 and degree 4 polynomial for every C. Also, that the higher the value of C, the better is both the train and test accuracy, this implies that the data tends to be linearly separable with this kernel. Here, we present the plots of decision boundaries for best C for each degree with support Vectors.

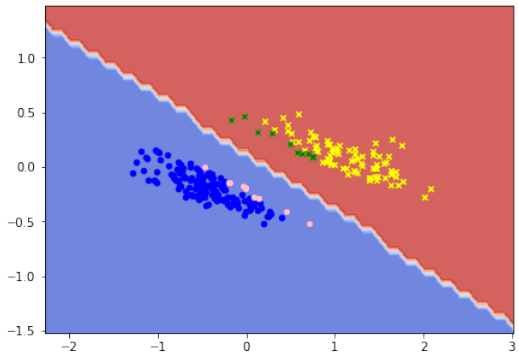

Fig. 4. Linear SVC for C=0.6,without support vectors, No. Of SV=20

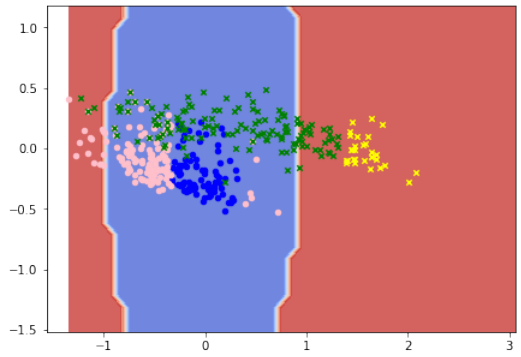

Fig. 5. Polynomial SVC with degree=2, C=0.6. Here No. of SV= 291

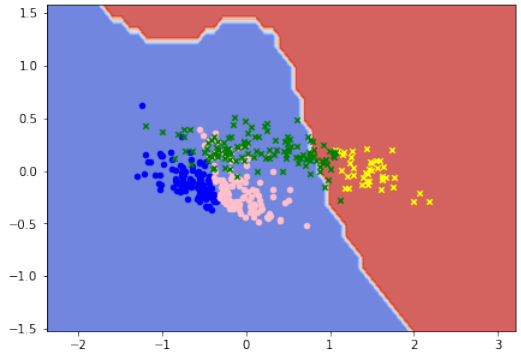

Fig. 6. Polynomial SVC with degree=3, C=0.6. Here No. of SV= 240

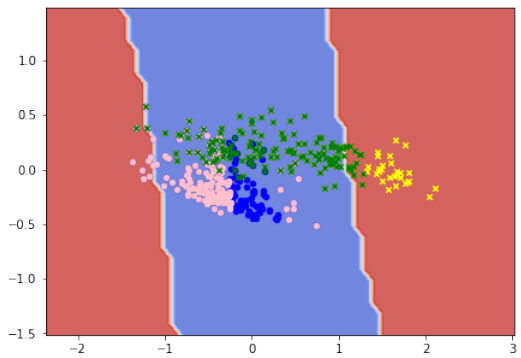

Fig. 7. Polynomial SVC with degree=4, C=0.6. Here No. of SV= 292

Respective Plots without Support Vectors are in Fig8, Fig9 and Fig10

- RBF Kernel SVC
  For different γ and C values, we have followed a grid search procedure to find the best pair.

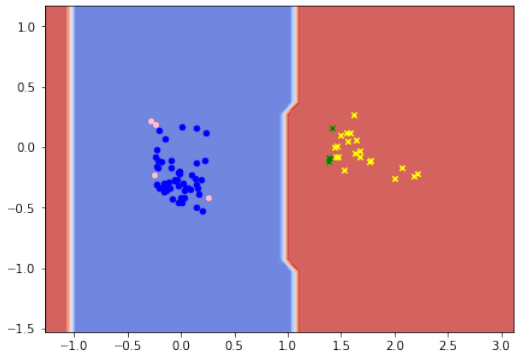

Fig. 8. Polynomial SVC with degree=2, C=0.6

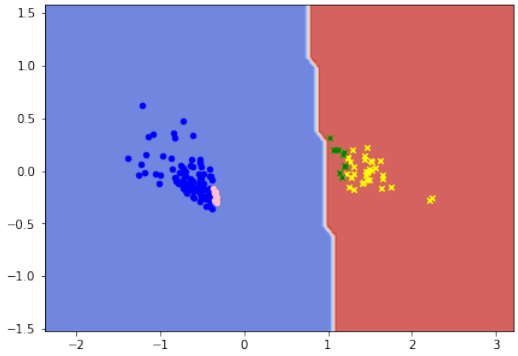

Fig. 9. Polynomial SVC with degree=3, C=0.6.

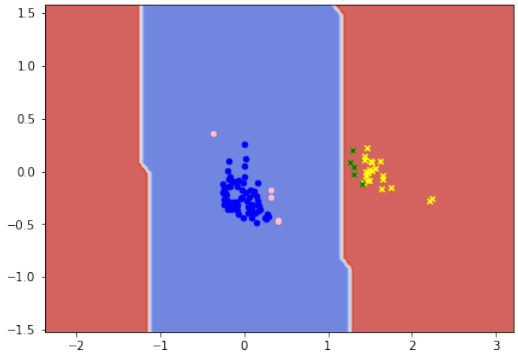

Fig. 10. Polynomial SVC with degree=4, C=0.6.

Grid Search over C and Gamma.(train and Test accuracy								
$\gamma \downarrow$, C$\rightarrow$	0.001	0.03	0.1	0.6	1	30	600	1000
0.001	56.50 58.00	56.50 58.00	56.50 58.00	56.50 58.00	56.50 58.00	84.25 86.67	86.00 85.33	86.25 85.33
0.05	56.50 58.00	56.50 58.00	78.25 82.00	83.75 86.00	84.75 85.67	86.00 85.33	86.50 85.67	86.50 85.67
0.5	56.50 58.00	81.00 84.33	84.75 86.33	85.50 86.33	85.50 85.33	86.0 85.33	85.5 85.33	85.5 85.33
1	56.50 58.00	82.25 85.33	84.75 86.33	85.50 86.33	85.50 86.00	86.50 85.67	85.75 85.33	86.00 85.67
50	56.50 58.00	56.50 58.00	86.25 85.67	87.25 84.67	87.75 82.67	88.50 81.67	89.75 82.67	90.00 82.00
500	56.50 58.00	56.50 58.00	56.50 58.00	90.00 80.00	90.00 79.67	90.50 80.00	90.50 80.00	90.50 80.00
1000	56.50 58.00	56.50 58.00	56.50 58.00	90.50 72.67	90.25 78.67	90.50 79.00	90.50 79.00	90.50 79.00

The higher the value of γ , the more complex the kernel function gets, Hence we will get an irregular decision boundary, which leads to over-fitting. This can be seen from the Values recorded, as γ increases, the train accuracy always increases but the test accuracy decreases after a certain point. (bias variance trade-off).
Also, another thing to note, this kernel is capable of separating points to a good extent. This is supported by the fact that as C increases both train and test accuracy increases for a particular γ.
Hence the better option to choose is from smaller values of γ. For our results, we have chosen C=1 and γ=1. Plot with and without the support vectors:

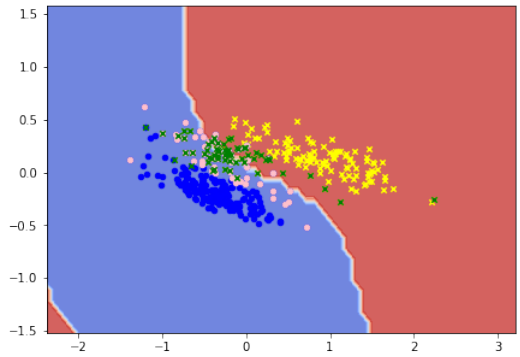

Fig. 11. RBF Kernel C=1 and γ=1

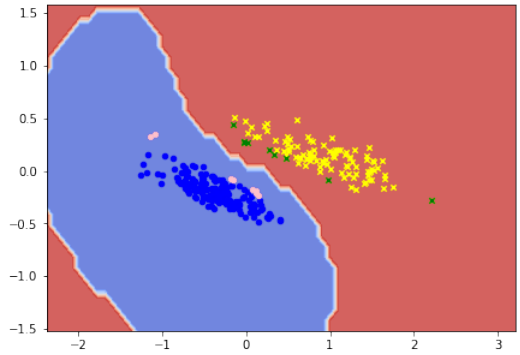

Fig. 12. RBF Kernel C=1 and γ=1

We have shown a case of over-fitting,with highly irregular decision boundary in Fig13.

- Bonus Part- SMO Algorithm
We have implemented SMO algorithm following CS229 Notes[2].
The running time is measured by breaking the complete Dataset into 20 parts and adding a part in each iteration.
Test Accuracy for Our Implemented SMO algorithm in case of linear SVC is 0.855.
Plot for LibSVM SMO algorithm vs Our SMO algorithm is in Fig14

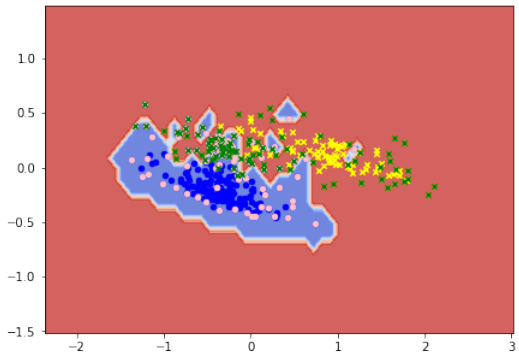

Fig. 13. RBF Kernel C=30 and γ=50, as seen the decision boundary is highly irregular

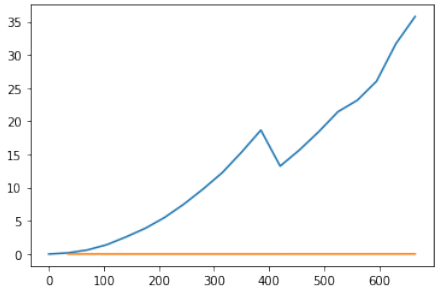

Fig. 14. LibSVM SMO vs Our SMO, the blue line represents running time of our SMO and the orange line LibSVM SMO

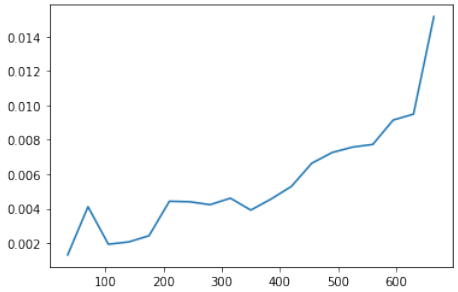

Fig. 15. LibSVM SMO running time

II. Question 2

A. Feedforward Neural Networks (FNNs)

We have hard-coded the error-back propagation algorithm for a FNN, along with implementations of various methods of regularization, namely: L1 regularization, L2 regularization, batch normalization, dropout, and early stopping. We have trained our FNN implementation on tiny-imagenet. The dataset was reduced to 100 training examples per each class (randomly sampled), the architecture has 4 hidden layers (800,500,100,50 neurons) and hyperbolic tan is first activation, rest are sigmoids. Clearly, the FNN doesn't perform well because we don't account for the grid-like topology of images. We're using mini-batch SGD for batches of size 100. Moreover, we were not able to overfit our data because our code kept timing out whenever we went beyond 100 epochs. Graphs show best scaled values.

Accuracies, FNN, mnist						
Regularizer	None	L1	L2	Dropout	Batch norm.	Early stop
Params.	None	$\lambda = 0.5$	$\lambda = 0.5$	0.2 (all layers)	None	None
Accuracy	0.104	0.114	0.094	0.1	0.099	0.098

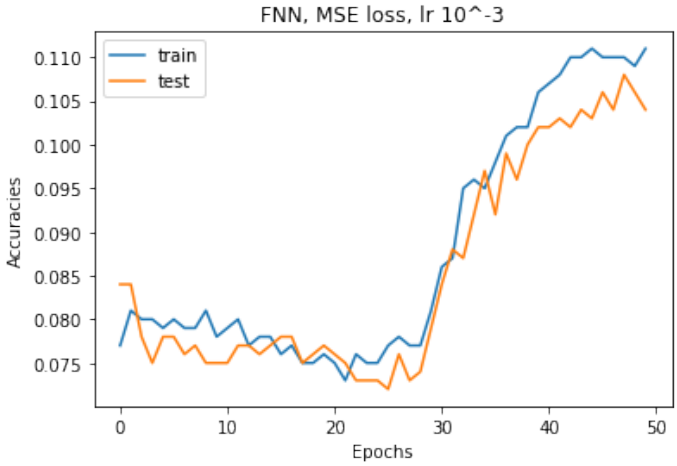

Fig. 16. No regularization, FNN on MNIST

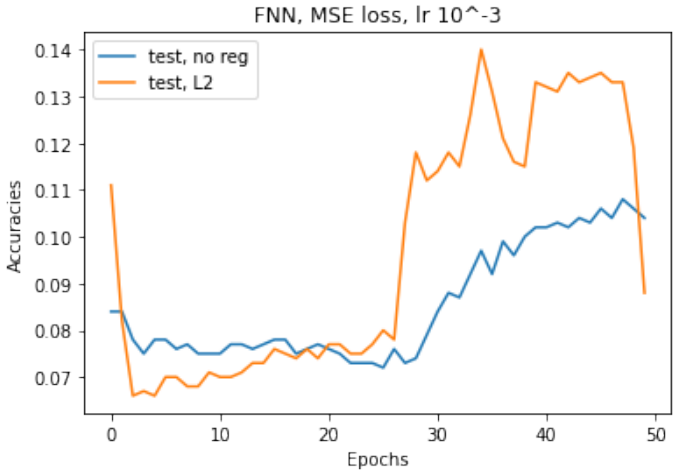

Fig. 17. Comparision, FNN on MNIST

B. Convolutional Neural Networks (CNNs)

We have implemented LeNet, AlexNet, and VGG-16 from scratch using PyTorch. We have trained all 3 CNNs on the MNIST dataset, and have compared the effect of varying optimizer for LeNet. LeNet performs the best with Adam optimizer. Adam learns the per-parameter learning rates itself, and its hyperparameters define the decay rate for this, whereas SGD has a global learning rate, and SGD with momentum is an algorithm to help us progress to the local optima faster (by adding a term which accounts for the last parametric updates in our weight update eqn).

Accuracies, CNN, MNIST			
Model	LeNet (SGD)	LeNet (SGD+Mom.)	LeNet (Adam)
Accuracy	0.986	0.987	0.988

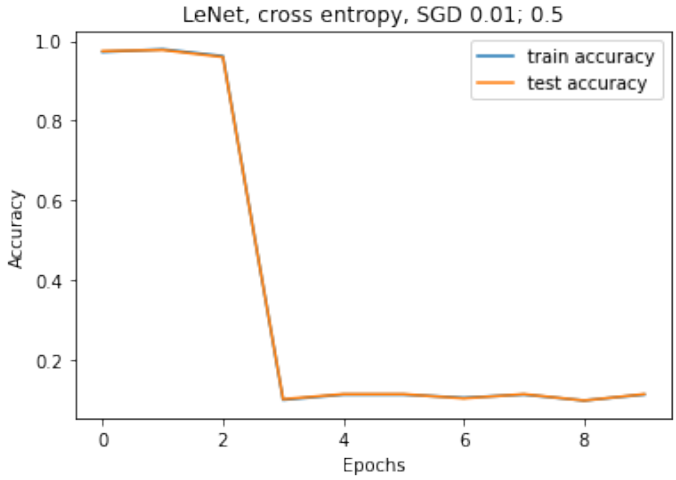

Fig. 18.

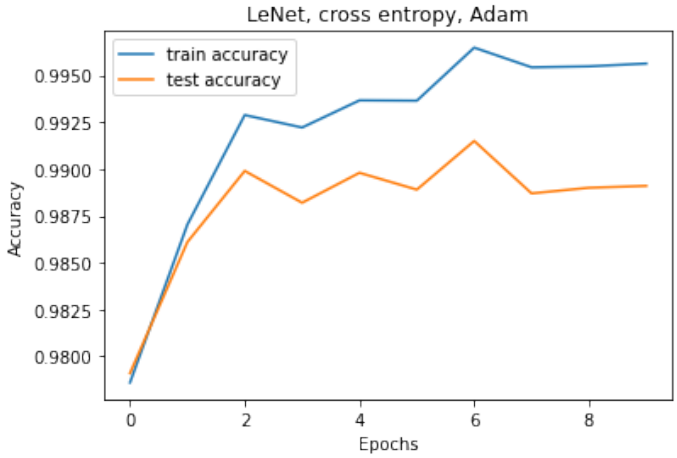

Fig. 19.

III. Question 3

A. QUESTION 3.1

1) Problem Statement : An Imbalanced Data Set is provided. Objective is to train a classifier modifying the loss function to improve the accuracy of the model.

2) Data Pre-processing: We observed that the different measures lies in different ranges. So to avoid problems like floating point Errors, we have normalised the data to lie in the same range. Selected 3 classes are 1,8,5. Data is downsampled to 5000 samples in total. Training Data and Test Data are obtained by random shuffling and 80-20 split.

3) Metrics Used:

- Accuracy and AUC (since the data-set is Skewed, accuracy isn't a useful measure)

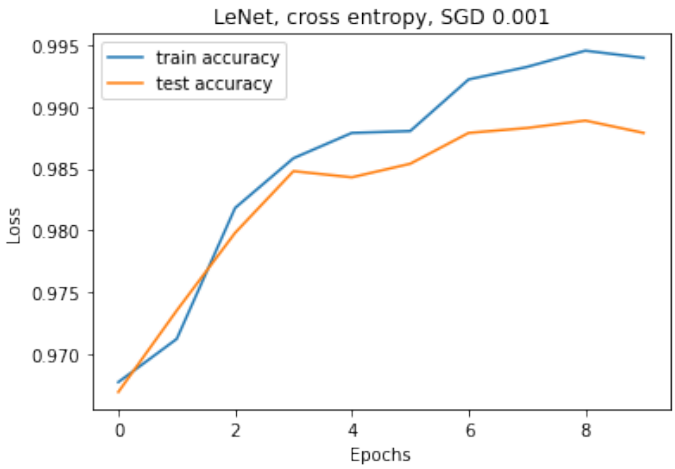

Fig. 20.

4) Modifications to the Loss: We have used a Neural Network as classifier with the following architecture:

Model Architecture for k clusters.			
Layer	Input Dimension	OutPut Dimension	Activation function
Layer 0	784	512	ReLu
DropOut Layer with rate=0.2			
Layer 1	512	512	ReLu
DropOut Layer with rate=0.2			
Layer 2	512	3	ReLu

Here we suggest weighing the loss in such a way that making an error on minority class is penalised more than making an error on majority class. This way the classifier learns to not neglect the minority class. We train the model 5 times and take average of the test accuracy and AUC to minimize any computational error or floating point error.

- Modification to Cross-Entropy Loss: For Standard loss: After 40 epochs with batch size of 64, we attained 0.5530 accuracy on test data with AUC=0.7214.
  We present our results with different set of class weights. For Class weights as 0: 1.5, 1:2, 2: 2.5 Test Accuracy

Model performance for different set of Class weights.		
Weights	Test Accuracy	Test AUC
0: 1.5, 1:2, 2: 2.5	0.5832	0.7312
0: 1, 1:2, 2: 3	0.6030	0.7402
0: 1, 1:2.5, 2: 5	0.5970	0.7321

  As we can note from above table, the average performance of the weighted loss model is better than the standard loss model.
- Modification to MSE loss: After 40 epochs with batch size of 64, we attained 0.5920 accuracy on test data with AUC=0.7374. We present our results with different set of class weights. For Class weights as 0: 1.5, 1:2, 2: 2.5 Test Accuracy

Model model performance for different set of Class weights.		
Weights	Test Accuracy	Test AUC
0: 1.5, 1:2, 2: 2.5	0.6162	0.7474
0: 1, 1:2, 2: 3	0.6212	0.7488
0: 1, 1:2.5, 2: 5	0.6056	0.7377

  As we can note from above table, the average perfor-

mance of the weighted loss model is better than the standard loss model.

- Focal Loss
  Following the work done by Facebook AI Research(FAIR)[3], we have implemented Focal Loss. For standard cross entropy loss, we found the test accuracy to be 0.5530 with AUC 0.7214. In the research paper, they have shown that we ca obtain best results for $\gamma = 2$ For γ =2 and α =0.25, Test accuracy with focal loss is 0.6048 with AUC=0.7368. Here we can see the improvement in accuracy.

B. Question3.2

1) Problem Statement : Given is SVHN datasets. We have performed KMeans clustering on the data for various k's and trained the classifier considering each cluster as class.

2) Data Pre-processing: Converted each image into grayscale. Applied PCA(principal component Analysis) to reduce the number of features from 1024 to to 20.
We observed that the different measures lies in different ranges. So to avoid problems like underflow/overflow/floating point Errors, we have standardize the data to lie in the same range.

3) Metrics Used:

- Accuracy

Kmeans Clustering was used to cluster the data into k=3,5,10,15,20. With each cluster as class, we have trained an SVM and a Neural networks and Record the results.

4) Classifiers Used:

- C-SVC
  PCA reduced the features from 1024 to 20. Using SVM for such kind of Data makes sense. The Data is classified on the basis of its distance from each of the cluster's centroids. Hence, if 2 centroids are there, the perpendicular bisector hyperplane of the line joining the 2 centroids is the classifier. And this classifier will have the highest margin possible. For multi-class case, we have used one vs one case. Hence, we have implemented a linear SVC, with one vs one approach, C=0.1. Below we note the model performance.

Model performance for C=0.1 and different values of k					
C ↓, k →	3	5	10	15	20
0.1	0.9115	0.98	0.9725	0.9565	0.9425

- Neural Network. PCA reduced the features from 1024 to 20 Model Architecture:

Model Architecture for k clusters.			
Layer	Input Dimension	OutPut Dimension	Activation function
Layer 0	20	32	ReLu
DropOut Layer with rate=0.2			
Layer 1	32	28	ReLu
DropOut Layer with rate=0.2			
Layer 2	28	25	ReLu
DropOut Layer with rate=0.2			
Layer 3	20	32	ReLu
DropOut Layer with rate=0.2			
Layer 4	28	25	ReLu
DropOut Layer with rate=0.2			
Layer 5	25	k	Sigmoid

For training, parameters are epochs=96, batch size=128 We have used 2 different Loss functions namely MSE and Categorical Cross Entropy. For each different cluster model, we have trained the NN 5 times and selected the one with highest test accuracy.(This is done to handle the non convexity of the Cost function.) Below we mention the test accuracy for different k.

Model performance for C=0.1 and different values of k					
Loss ↓, k →	3	5	10	15	20
Categorical Cross Entropy	0.88	0.745	0.827	0.846	0.831
MSE	0.879	0.875	0.8249	0.730	0.7095

Here is an important observation to note: The Neural Network in spite of having 5 layers could not learn the data well. Neural Nets require a sophisticated hyper-parameter tuning process, for it to perform well. on the other hand, SVMs don't have such issues.

C. Part 3.3

First, we train a classifier on the MNIST dataset and test it on the SVHN. Then we do the opposite of this process, i.e. train a classifier on SVHN and test on MNIST. The results are as follows:

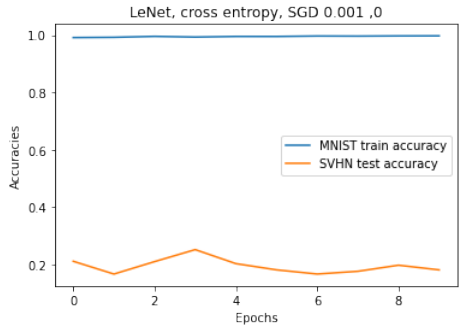

Fig. 21. Part 3.3, MNIST train, SVHN test

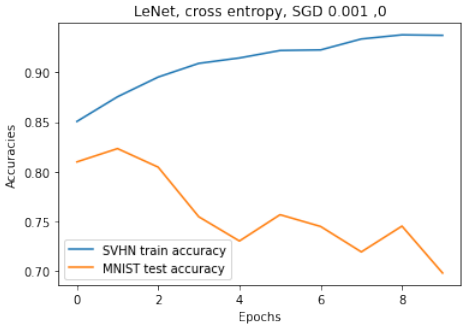

Fig. 22. Part 3.3 SVHN train, MNIST test

Observe that training on MNIST then testing using SVHN leads to terrible test accuracies for the SVHN dataset (20%), whereas if we first train on SVHN then test using MNIST - we get relatively much better accuracies (70%). An intuitive explanation for this is that in SVHN we learn "noisier" versions of numbers (in the sense that there is stuff in the

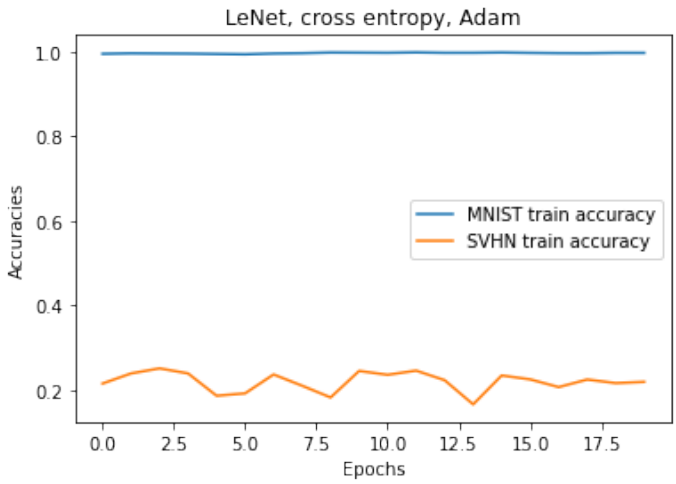

Fig. 23. Expt1- weighted loss

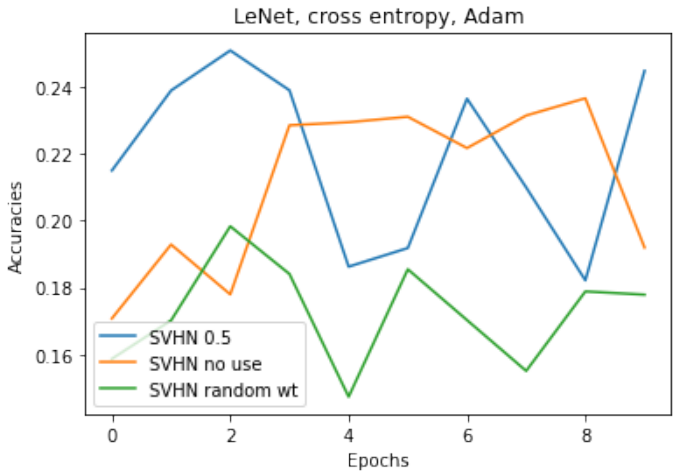

Fig. 24. Expt2- noisy MNIST

background) so the features our classifier learns when trained on SVHN are more robust in some sense.

Next, we try to improve the accuracy of a classifier trained on MNIST and tested on SVHN assuming the unlabelled SVHN dataset is accessible during training. Going by our earlier observations, we propose 2 different algorithms to improve the same - one is based on penalizing the loss function using unlabelled SVHN data, the other is based on constructing a "noisy" version of the MNIST input by overlap with a SVHN dataset.

D. Part 3.4

In this part, we are given only 5 classes to train on and asked to improve accuracy considering all the 10 classes during testing.

Our work is inspired from- An embarrassingly simple approach to Zero-Shot learning[11]

This is a classic example for Zero Shot learning. In simpler words, a shot is the number of samples of a class one needs for the machine to learn about the class. In zero shot learning the challenge is to make the model learn about the class which it has not seen. For this problem, we have addressed this in terms of some classical problems like document classification. The model uses 2 layers, which tries to create relationship between features, attributes and classes with the help of a linear model. The first layer helps in defining the relationship between features and attributes with the help of weights in that layer. The second layer deals with modelling the relationship between attributes and classes where the prescribed attribute signatures is fixed. then they find the weight matrix by Coefficient matrix and a signature matrix in an unsupervised manner.

Implementation details: Instead of using the MNIST Dataset, we have used a smaller MNIST with each image being 8x8 instead of 28x28.

Model performance, Part 3.4		
Model	Logistic Regression	Zero-shot model
Accuracy on given	0.97	0.237
Accuracy on new	0.03	0.249
Net accuracy	0.19	0.247

E. Part 3.5

In this part, we compare the PCA and tSNE algorithms for the SVHN dataset. Observe that the idea of tSNE is that it preserves local similarity, and PCA focuses on maximizing the variance. There might be overlap between the clusters by virtue of nature of the dataset (it contains images of differing labels which highly overlap with each other). These figures help us visualize how the dataset lies on a lower dimensional manifold. However, the main difference between these 2 algorithms is that PCA helps us with analysing Global structure (effectively construct a mapping from a higher dimensional space to a lower dimensional space) but tSNE just plots similar items close to each other (it doesn't explicitly learn a mapping from the higher dimensional to lower dimensional space).

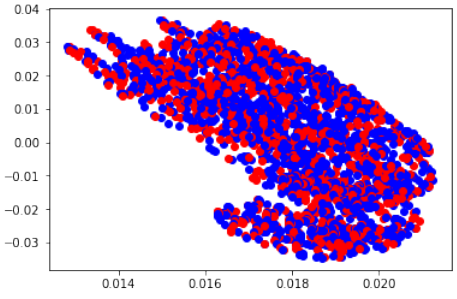

Fig. 25. Part 3.5, PCA components, features of SVHN with labels 0 and 5

ACKNOWLEDGMENT

Special Thanks to all my mates on Piazza and our Dear Professors.

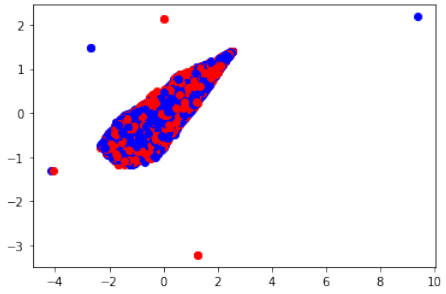

Fig. 26. Part 3.5, tSNE components, features of SVHN with labels 0 and 5

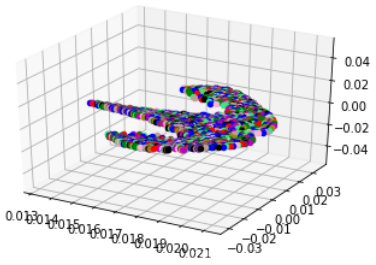

Fig. 27. Part 3.5, PCA 3 components, all features

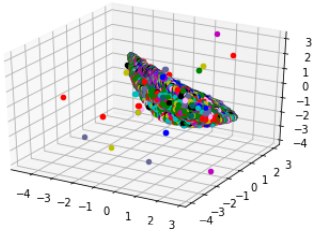

Fig. 28. Part 3.5, tSNE 3 components, all features

REFERENCES

[1] LIBSVM.
[2] SMO Algorithm
[3] Focal Loss for Dense Object Detection
[4] Kmeans Clustering
[5] Backpropagation
[6] Dropout regularization
[7] For CNN architectures
[8] Comparing tSNE and PCA
[9] Pytorch documentation
[10] Scikit learn documentation
[11] An embarrassingly simple approach to zero-shot learning