Traffic Sign Classification using State of the Art Machine Learning Model

Srinivas Rao Daru
Electrical and Computer Engineering
University of Califorina, San Diego
sdaru@ucsd.edu
A59003596

Sri Harsha Pamidi
Electrical and Computer Engineering
University of Califorina, San Diego
spamidi@ucsd.edu
A59005361

Venkata Sai Abhiram Medisetti
Electrical and Computer Engineering
University of Califorina, San Diego
vmediset@ucsd.edu
A59005405

Abstract

To classify the types of traffic signals, we implemented Logistic regression and Softmax regression classifiers. The dataset includes 43 traffic signs with various speed limit signs, no passing , yield etc. To enhance the precision of the model we implemented Principal Component Analysis followed by Normalization techniques in order to achieve better input features. Logistic Regression Classifier is used to classify two speed limit signs namely "Speed limit 100km/h" and "Speed limit 120km/h". We achieved about **76.1%** accuracy using Batch gradient descent for logistic regression. We implemented Softmax Regression Classifier to classify all the 43 traffic signals present in the dataset. With PCA enabled, we achieved about **71.2%** accuracy using batch gradient descent and **92.1%** using stochastic gradient descent on aligned data.

1 Introduction

In modern day world, automation is everything. The current trend in the self driving cars need a lot of information to be processed and decisions need to be taken very fast. One such processing task is to classify the traffic sign. We have taken this chance to understand how to classify various traffic signs with the best accuracy possible. Classification plays a key role in many ways like controlling the speed of the car based the Speed Limit traffic sign, taking a U turn, yield at a signal, slow down at school zone, etc. These can be done only when the model is good enough to understand the picture taken by the camera at high speed. It is not always the case that one can go back to get a good picture. This is where Machine Learning Classification comes into picture. By using a machine learning model, one can train the model and deploy to model into the system like a self driving car. This model helps in understanding the image captured at high speed and classifies it into the best possible traffic sign and accordingly decisions can be taken.

2 Loading the Dataset for the Model

The dataset consisted of images(each image - 32x32) along with its labels(each label value - 0 to 42). We have flattened the 2 Dimension images into a single array of 1x1024. Since we wanted to

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

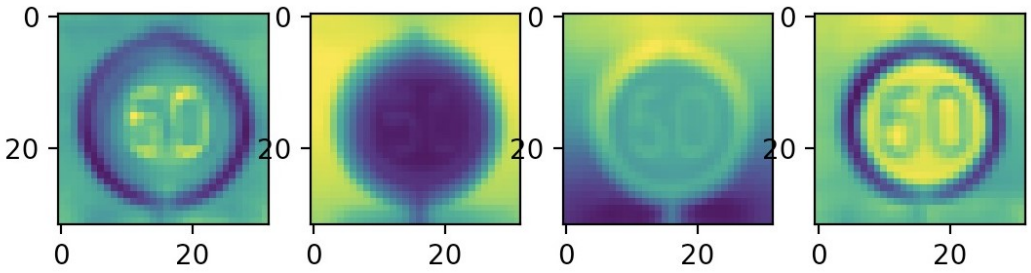


Figure 1: PCA visual on Aligned data Class 2-3 , leftmost is 1st PC, rightmost in 4th PC

extract better features and reduce the dimension , we have resorted to using Principal Component Analysis followed by Normalization.

In this project, our dataset has 43 different traffic signals. Any model works best when the input data is aligned. To understand the importance of aligned data we have worked on 2 types of data set, one is aligned dataset and the other is unaligned dataset. Aligned data set has the center of the traffic sign at the center of the image and Unaligned dataset has the center of the traffic sign elsewhere and not uniform across the images.

We also Normalized the input features obtained from PCA. This is done because the cost function we defined is applicable when each feature is uni-variant gaussion distribution. So to do that, we normalized along each dimension.

We have used 2 types of data splitting – We used 80% of the dataset to train the model, 10% of the data set to validate the model, and the final 10% of the data set to test the model. This 80/ 10/ 10 ratio is decided to make sure that model has enought data to be trained and at the same time, not to overfit by considering validation loss. This process of stopping the training of the model when validation loss increases is called Early Stopping and is used in order to make sure that the model doesn't overfit.

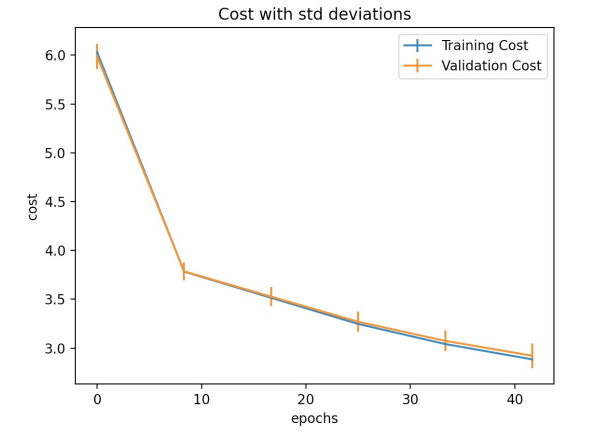
3 Design of Principal Component Analysis

Figure(1) gives an example of applying PCA to a 50KM/h and 60KM/h, which lists the top 4 components. Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimension of large data sets. The idea of PCA is to reduce the number of variables of a data set, while preserving as much information as possible. When we perform PCS we find those eigen vectors related to the highest eigen value which are the vectors that makes the data to be with high variance when we project different input vectors onto them. For calculating them we first center the input data to be equal mean at each feature. Then we find the co variance matrix using numpy.covar. The eigen values and eigen vectors of this matrix gives us the principle components. We order the vectors by the eigen values and select the top once. The number of the components can be decided depending on the task at hand. We generally choose higher dimensions when we use large data like in softmax. But note that higher components always doesn't mean better, it decreases the computation speed due to more calculations per descent and also adds noise from lower order features, which need more data to give valuable information. For tranform we subtract each vector from the mean vector calculated above then project onto the principal components.

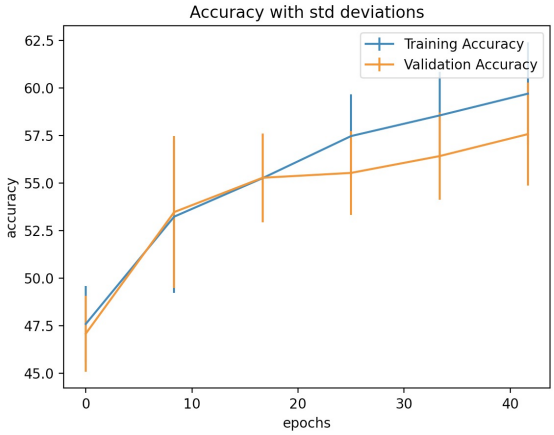
We also regularize by dividing with the variance along this dimension which can be calculated from the square root of eigen value. The reason this has been done is that the loss function is determined by the ML estimates with an assumption of the same variance variables. Also, it makes sure that the weights to be in same range, we need them to be as we use same alpha for all of weights.

4 Classification of Two Traffic Signals

In this problem statement, the goal is to distinguish 2 traffic signs each indicating a different speed limit. We have chosen Speed Limit 100km/h and 120km/h for this particular task. Among the 43 traffic signs, the labels for 100 km/h Speed limit sign is Class 7 and 120 km/h is Class 8. We



(a)Cost vs Epochs for Logistic



(b) Accuracy vs Epochs for Logistic

Figure 2: Plots for Question5b , Logistic Classification on Class 7-8 for Unaligned Dataset with PCA using lr=0.07, epochs = 50, dim = 60

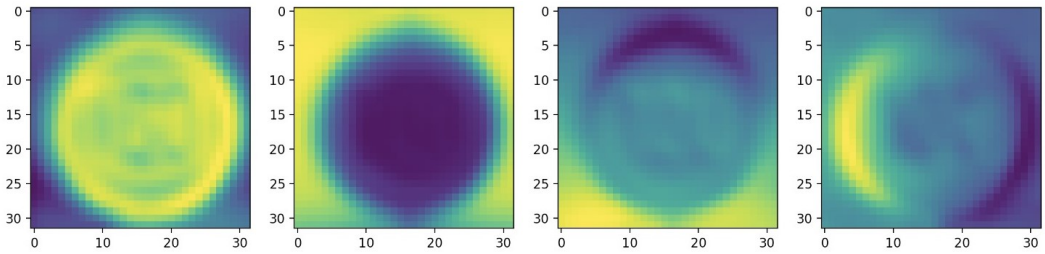


Figure 3: PCA visual on Unaligned data Class 7-8 , leftmost is 1st PC, rightmost in 4th PC

designed the logistic regression classifier to predict 1 in case of 100 km/h Speed limit(class 7) sign and predict 0 in case of 120 km/h Speed limit(class 8) sign. Similarly we classified Dangerous Curve to the Left (Class 19) and Dangerous Curve to the right (Class 20) using logistic regression classifier. We designed the model to predict 1 in case of Right dangerous curve and predict 0 in case of left dangerous curve

4.1 Q-5a Design of Logistic Regression Classifier

We have designed a logistic regression classifier using the Binary Cross Entropy Cost function and batch gradient descent principle to update the weights. Although we need to classify the input into either of the 2 classes, we only needed one output feature. This is because the Logistic regression classifier output indicates the probability of a certain class.

4.2 Q-5b Logistic Regression Classifier on Unaligned Dataset, 100km/h (class 7) vs Speed limit 120km/h

The plots for the **question (5b)** can be seen in **Figure(2)**. The test accuracy is **58.2%**. The principal components for unaligned dataset is in **Figure(3)** When we first trained the model on unaligned data set, the results were not very good looking on this data set. This is because the PCA tries to maximize variance among each dimension that is used in representing the input. Since the input images are not aligned, they dont have a fixed center. So when PCA is done on such input data, the output data doesn't necessarily extract good input features. So applying PCA on an unaligned dataset would be like raining on a buffalo. It doesn't make sense like, just like the previous statement! We have attached the first 4 PCA outputs to get a better visualization. Like mentioned, since the input features doesn't have any meaningful features, all we can see is some noise.

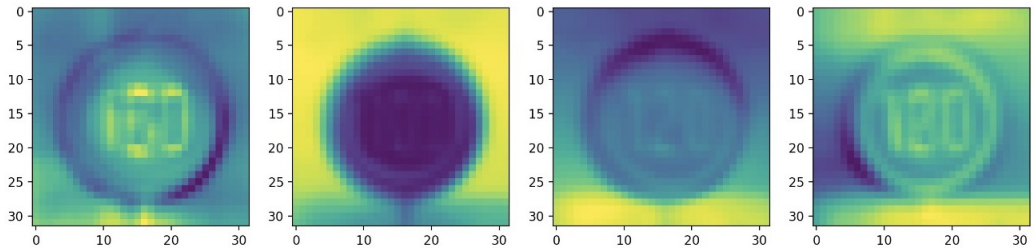
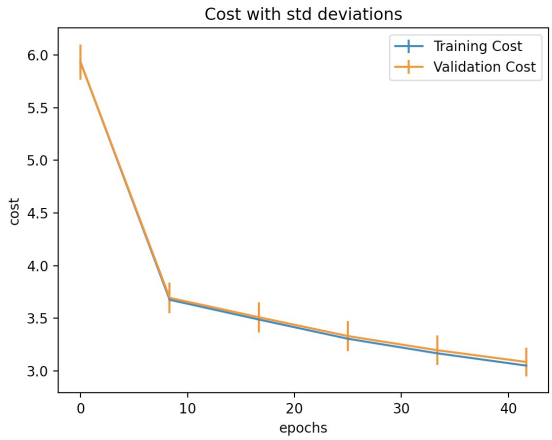
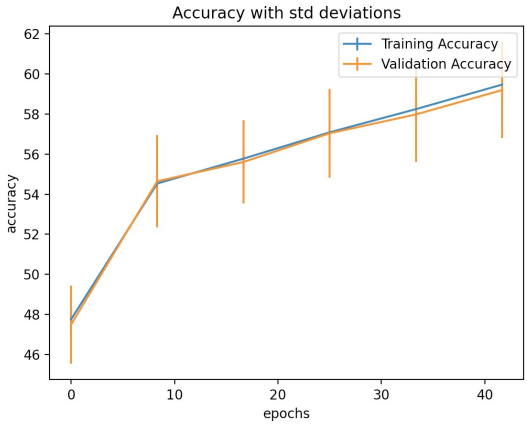


Figure 4: PCA visual on Aligned data Class 7-8 , leftmost is 1st PC, rightmost in 4th PC



(a)Cost vs Epochs for Logistic



(b) Accuracy vs Epochs for Logistic

Figure 5: Plots for Question 5c(ii) , Logistic Classification on Class 7-8 for Aligned Dataset with PCA using lr=0.07, epochs = 50, dim = 60

4.3 Q-5c Logistic Regression Classifier on Aligned Dataset 100km/h (class 7) vs Speed limit 120km/h (class 8)

4.3.1 Q-5c(i)PCA components comparison between aligned and unaligned dataset

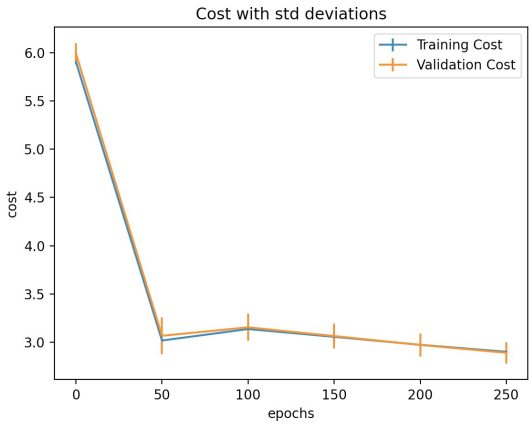
The plots of the first 4 principal components for aligned datset can be seen in **Figure 4**. When the data is unaligned the images are not centered so each pixel index has no significance as its related actual feature varies when not centered properly. But when the data is aligned, each pixel location has a significance as its same in all of the data. So when applying PCA for many data which is unaligned we wont be getting sharp borders or features. This can be compared in **Figure 3,4** the unaligned components are little blurred and have no borders, whereas aligned once have sharper borders and meaningful features.

4.3.2 Q-5C(ii) Loss and accuracy plots for aligned dataset

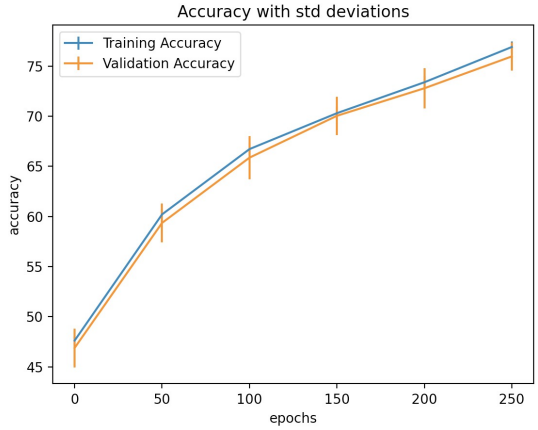
The plots for the **question (5C(ii)))** about the loss and accuracies vs epoch can be seen in **Figure(5-8)**. We have explored two learning rates and have shown them in two figures. We have also done for maximum epochs = 50, 100. The best testing accuracy we got for aligned data set is **76.1(1.4)%** for epochs = 300 and **61.5(1.9)%** for epochs =50

4.3.3 Q-5c(iii) Cost and Accuracy plots for aligned data set for 3 different learning rates

The accuracy values for the **question (5C(iii)))** for different learning rates is listed below in **Table(1)**. The loss and accuracy values are compared in the **Figure(9)**. For this we have repeated the above process with 3 different values of learning rate.

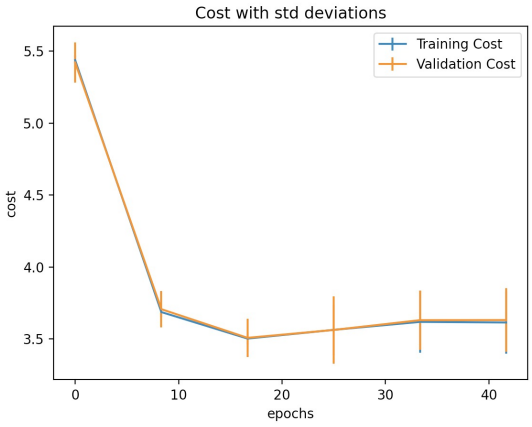


(a)Cost vs Epochs for Logistic

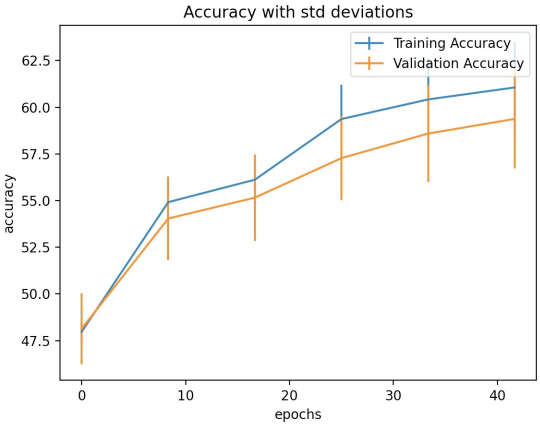


(b) Accuracy vs Epochs for Logistic

Figure 6: Plots for Question 5c(ii) , Logistic Classification on Class7-8 for Aligned Dataset with PCA using lr=0.07, epochs = 300, dim = 60



(a)Cost vs Epochs for Logistic

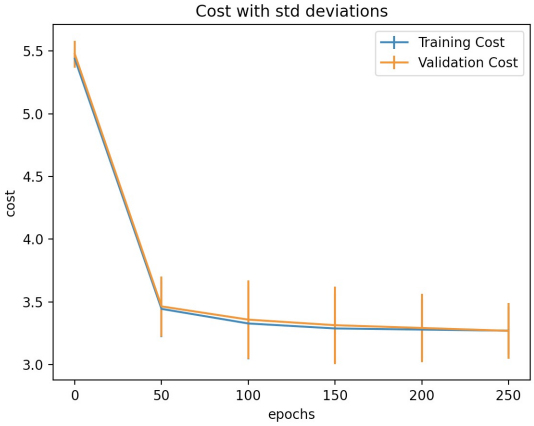


(b) Accuracy vs Epochs for Logistic

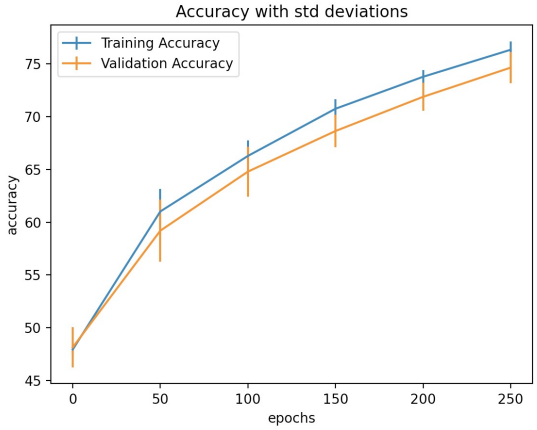
Figure 7: Plots for Question 5c(ii) , Logistic Classification on Class7-8 for Aligned Dataset with PCA using lr=0.09, epochs = 50, dim = 100

Number of Epochs	Learning Rate	Input Dimension	Test Accuracy
50	0.0014	50	59.2%
50	0.07	50	61.4%
50	0.35	50	59.9%
300	0.0014	50	71.4%
300	0.07	50	76.8%
300	0.35	50	72.4%

Table 1: Logistic Regression Model Performance for Different learning rates

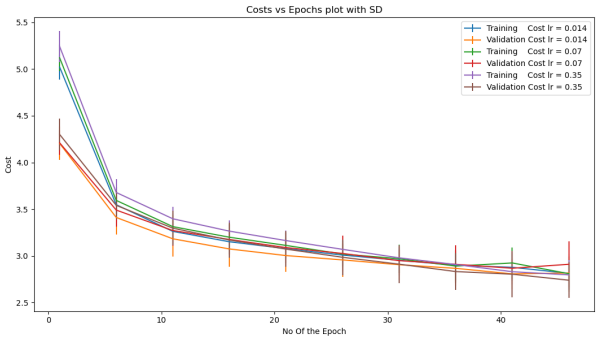


(a)Cost vs Epochs for Logistic

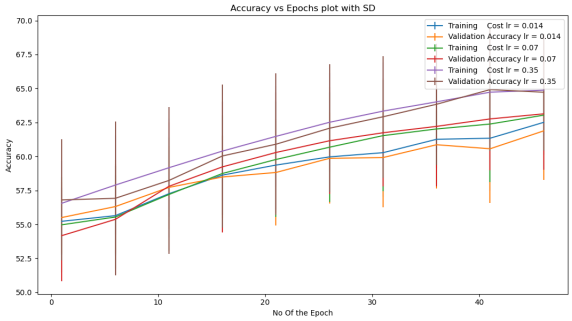


(b) Accuracy vs Epochs for Logistic

Figure 8: Plots for Question 5c(ii) , Logistic Classification on Class7-8 for Aligned Dataset with PCA using lr=0.09, epochs = 300, dim = 100



(a)Cost vs Epochs for Logistic



(b) Accuracy vs Epochs for Logistic

Figure 9: Plots for Question 5c(iii) , Logistic Classification on Class 7-8 for Aligned Dataset with PCA using lr=0.0014, lr=0.07, lr=0.35 with epochs = 50 , dim = 50

4.4 Q-5d Logistic Regression Classifier on Aligned Dataset (class 19) vs (class 20)

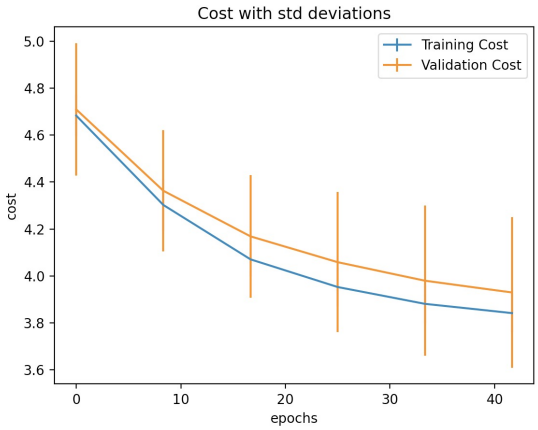
The same process done in **section (4.2)** is repeated for class 19 and 20. The plots for the **question (5d))** of the loss and accuracy are shown in **Figure(10-11)**. We have got a test accuracy is **78.2(1.9)%** for same learning rate and best of **81.1(2.1)%** for a different learning rate . Here we have explored 2 learning rates. The first four prime components have been shown in **figure(12)** By taking the same learning rate as above we see the accuracy is better for 19-20 as they can be distinguished easily.

5 Classification of Multiple Traffic Signs

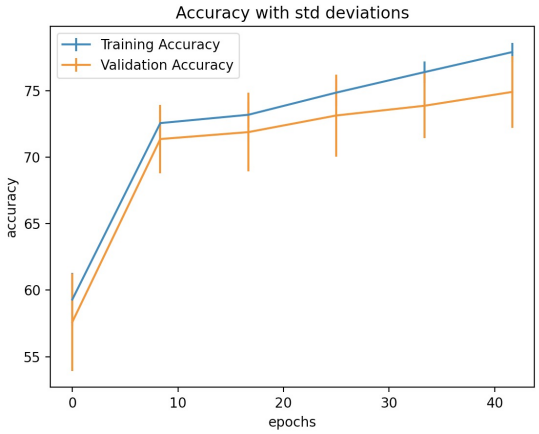
In this problem statement, the goal is to distinguish and classify 43 traffic signals. This is a multi class classification problem. Just like Logistic regression classifier which we used before to classify 2 traffic signals, we have implemented Softmax Regression Classifier to classify 43 traffic signals. In this case, we have designed a 43 output feature model.

5.1 Design of Softmax Regression Classifier

Softmax is a generalization of Logistic classifier. We have defined Multi Class Cross Entropy Cost function for the Softmax Regression Classifier. We have designed the model to generate 43 output

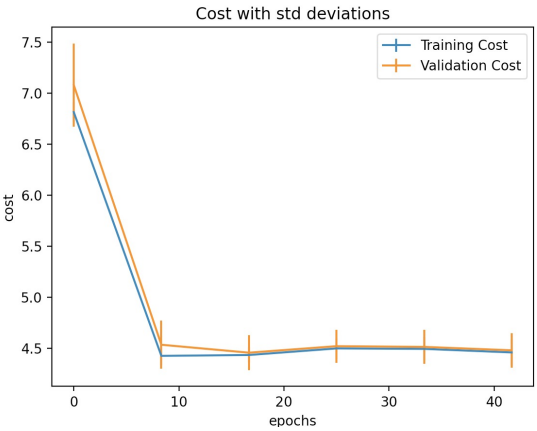


(a)Cost vs Epochs for Logistic

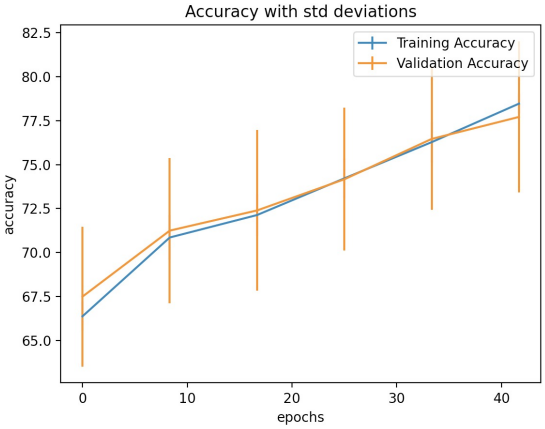


(b) Accuracy vs Epochs for Logistic

Figure 10: Plots for Question 5d , Logistic Classification on Class 19-20 for Aligned Dataset with PCA using lr=0.07, epochs = 50, dim = 50, kfold=10



(a)Cost vs Epochs for Logistic



(b) Accuracy vs Epochs for Logistic

Figure 11: Plots for Question 5d , Logistic Classification on Class 19-20 for Aligned Dataset with PCA using lr=0.09, epochs = 50, dim = 50, kfold=10

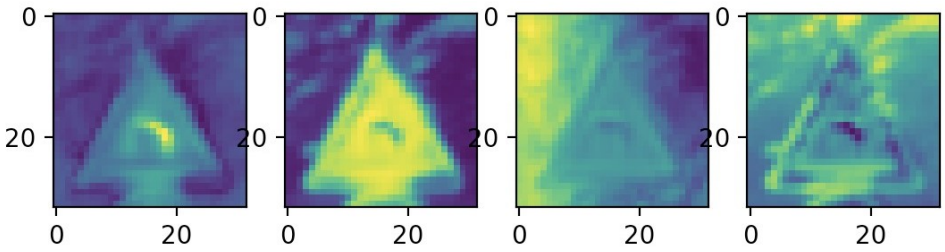
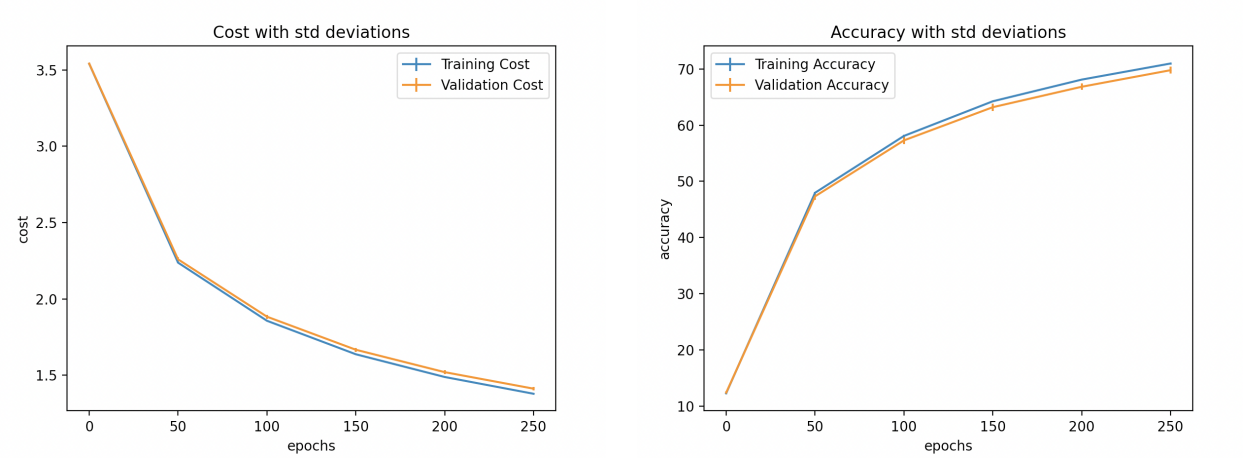


Figure 12: PCA visual on Aligned data Class 19-20 , leftmost is 1st PC, rightmost in 4th PC



(a)Cost vs Epochs for Softmax, lr = 0.07

(b) Accuracy vs Epochs for Softmax, lr=0.07

Figure 13: Plots for Question 6a(i) , Softmax Classification for Aligned Dataset with PCA, lr=0.07, epochs = 300, dim = 300

features and only one of the 43 output features will be predicted as 1 and the rest will be predicted as 0. The output feature whose prediction is 1 is the label of the image that is to be classified.

5.2 Q 6a(i) Softmax Regression Classifier on Aligned Dataset with PCA

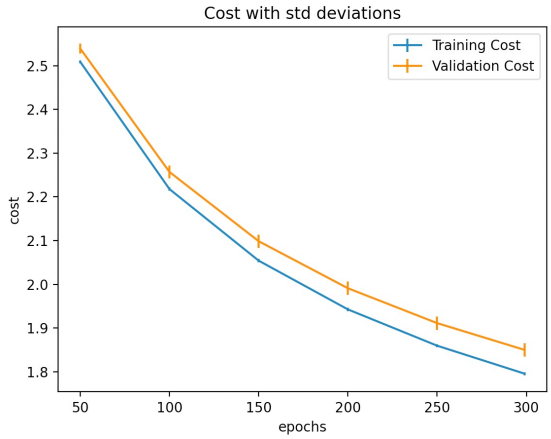
The plots for the **question (6a(i))** about the loss and accuracies vs epoch can be seen in **Figure(13)**. The testing accuracy we got is **71.2(0.2)%**. By doing the above exercises, we observed that PCA works best on aligned dataset and thus the importance of preprocessing the data to achieved aligned data followed by PCA has been apprehended. We also have done normalization of the input features obtained after doing PCA. This is done because the loss function and the gradient on it have been derived under the assumption that each input feature is uni-variant Gaussian distribution.

5.3 Q 6a(ii) Softmax Regression Classifer on Unaligned Dataset with PCA

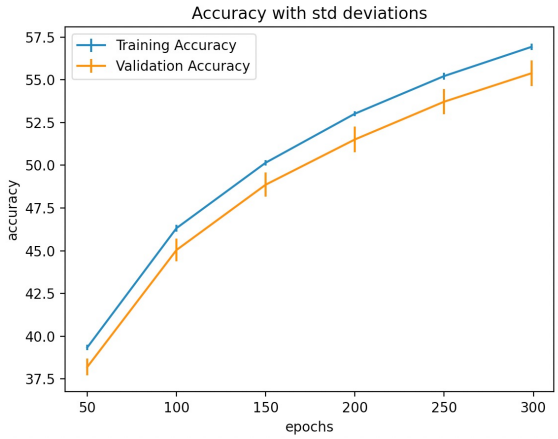
The plots for the **question 6a(ii)** of the accuracy and loss vs epochs can be seen in **Figure(14)**. The testing accuracy we got is **57.2(0.9)%** We have trained the model using unaligned dataset with PCA and observed that PCA doesn't work at its best on unaligned dataset. Reason was mentioned in the logistic regression model part. This is because the PCA tries to maximize covariance among each dimension that is used in representing the input. Since the input images are not aligned, they dont have a fixed center. So when PCA is done on such input data, the output data doesn't necessarily extract good input features. So applying PCA on an unaligned dataset would be like raining on a buffalo. It doesn't make sense like the previous statement, right! We have attached the first 4 PCA outputs to get a better visualization. Like mentioned, since the input features doesn't have any meaningful features, all we can see is some noise.

5.4 Q 6a(ii) Softmax Regression Classifier on Aligned Dataset with PCA disabled

The plots for the **question 6a(ii)** of the accuracy and loss vs epochs can be seen in **Figure(15)**. The testing accuracy we got is **70.2(0.2)%**. We have trained the model using aligned dataset but without PCA. We observed that the aligned dataset when flattened in of length 1024. When this is used as input feature for the model, the computations were quite expensive. We only have some meaningful features which are definetely not 1024 but much lesser. So when we implemented the model , the number of weights to be trained are much higher and so the model took forever to converge the weights. Thus the number of epochs will not be sufficient to arrive at the best model possible.

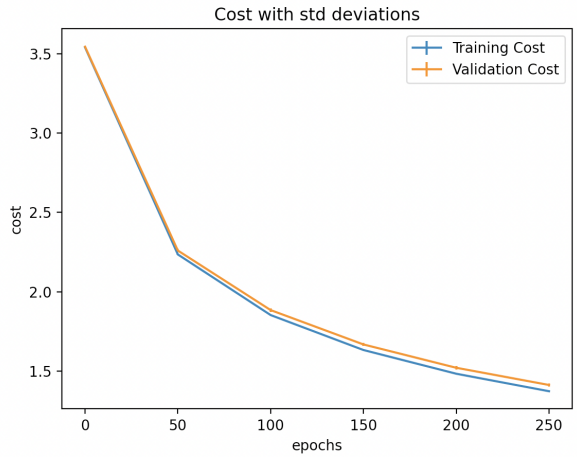


(a)Cost vs Epochs for Softmax, lr = 0.07

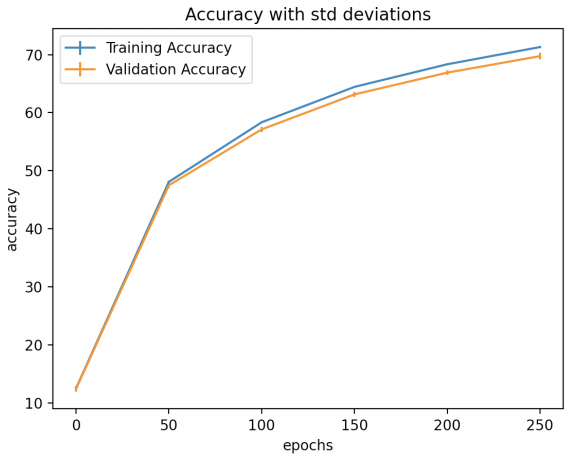


(b) Accuracy vs Epochs for Softmax, lr=0.07

Figure 14: Plots for Question 6a(ii) , Softmax Classification for Unaligned Dataset with PCA, lr=0.07, epochs = 300, dim = 300



(a)Cost vs Epochs for Softmax, lr = 0.07



(b) Accuracy vs Epochs for Softmax, lr=0.07

Figure 15: Plots for Question 6a(ii) , Softmax Classification for Aligned Dataset without PCA, lr=0.07, epochs = 300, dim = 300

5.5 q 6a(iii) Confusion matrices for the above cases

The confusion matrix for the best performance has been taken and shown below in **Figure(16)**. This was batch in next section we discuss for stochastic case so, we have also shown them here in **Figure(17)**

5.6 Q 6b Batch Gradient Descent Vs Stochastic Gradient Descent for Softmax Regression Classifier

The plots for the **question 6b** of the accuracy and loss vs epochs can be seen in **Figure(18)**. We have also plotted the loss and accuracy plots in**Figure (19)**We have used the Multiclass Cross Entropy Cost function in the softmax regression classifier. The loss function is derived and gradient of loss function is used in obtained the updated weights. The question is when to update the weights. So we have used two types of Weight updation techniques namely Stochastic gradient descent and Batch gradient descent. In case of Batch gradient descent, the weights are updated after using the data points. But in case of Stochastic gradient descent, the weights are updated at each data point every time. The use of stochastic gradient descent we observed is that convergence is achieved much faster

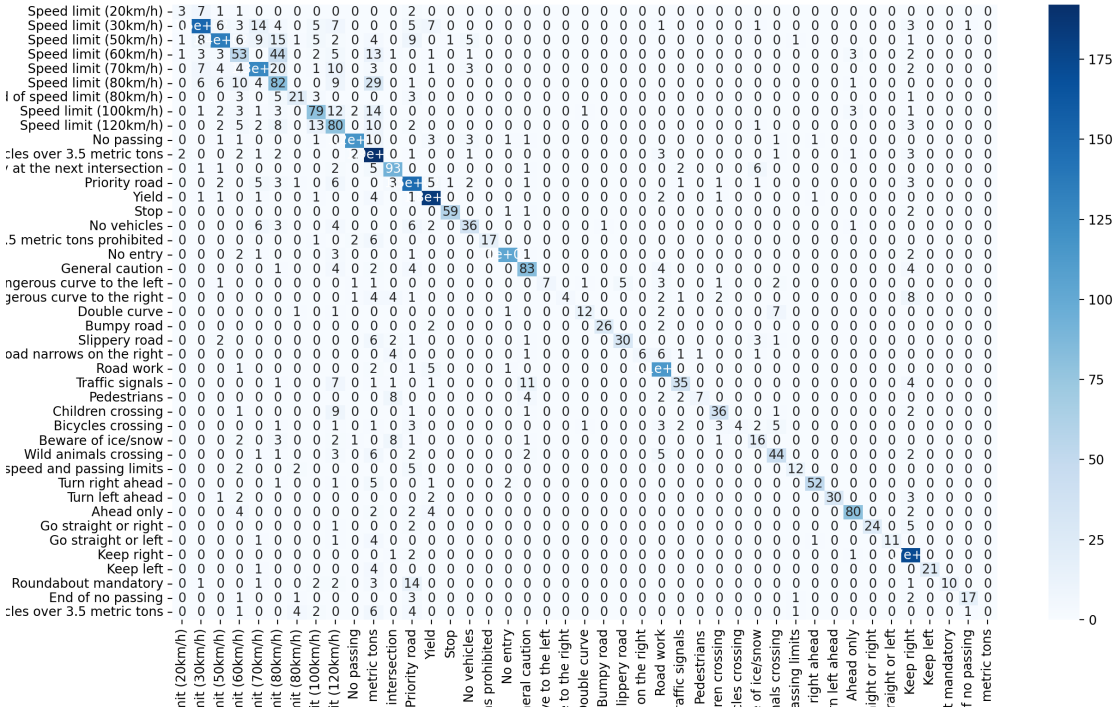


Figure 16: Plots for Question 6a(iii) ,Confusion Matrix on Aligned Data with PCA in Softmax Classification using Batch Gradient Descent

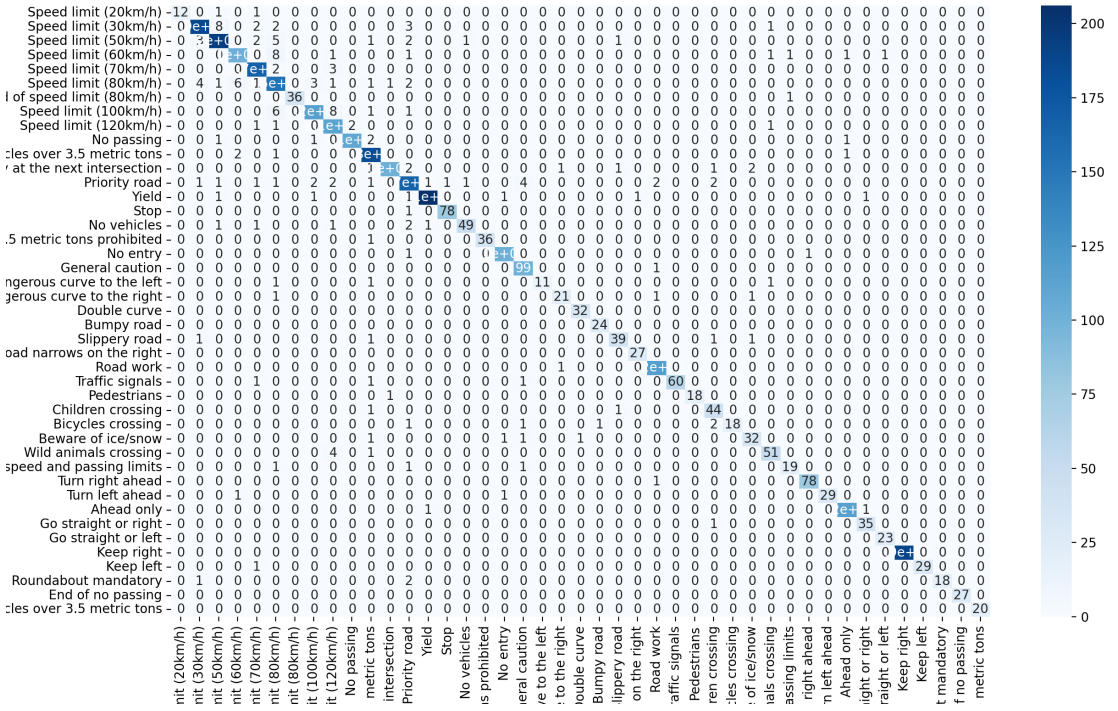


Figure 17: Plots for Question 6a(iii) ,Confusion Matrix on Aligned Data with PCA in Softmax Classification using Stochastic Gradient Descent

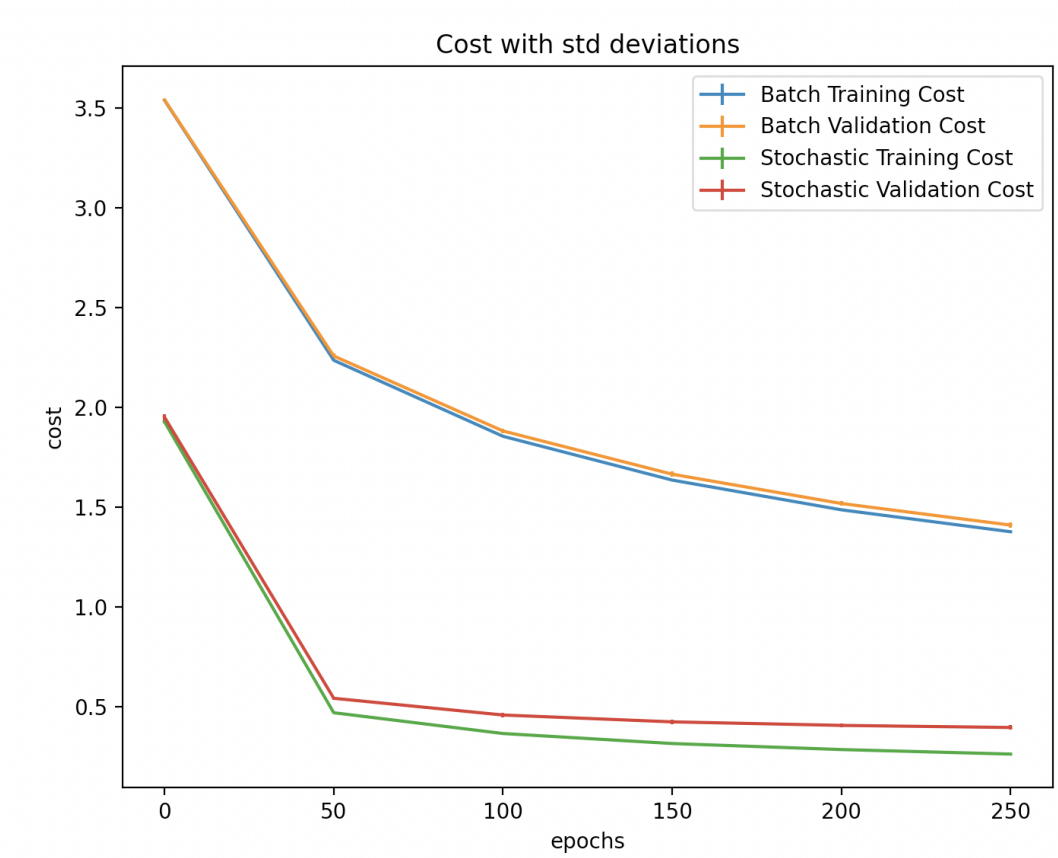


Figure 18: Plots for Question 6b , Softmax Classification using Stochastic Vs Batch Gradient Descent on Aligned Dataset with PCA, lr=0.07, epochs = 300, dim = 300

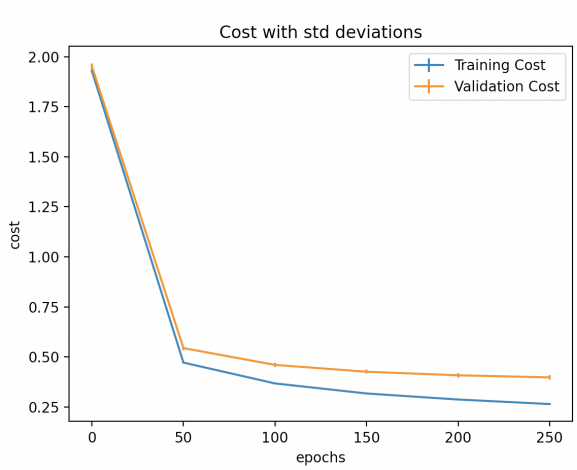
than the Batch gradient descent method. We achieved **71.2(0.2)%** accuracy in case of batch gradient descent and **92.1(0.1)%** accuracy in case of stochastic gradient descent.

5.6.1 Is stochastic gradient faster compared to batch gradient descent in terms of number of epochs

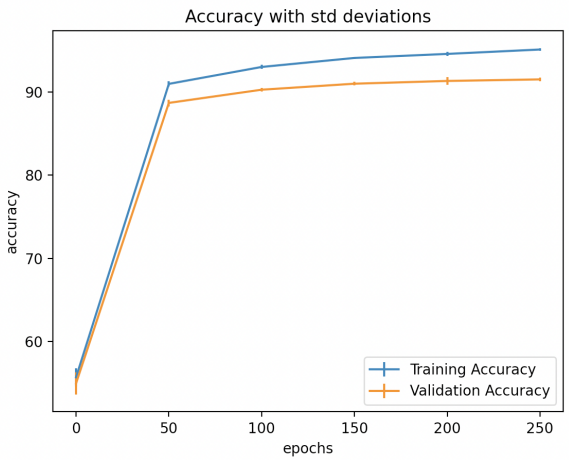
From the **Figure(18)** we can see that the stochastic is converging a little faster in terms of epochs, can be observed from the slopes, i.e it takes lesser epochs to reach to same accuracy. The reason is that in stochastic the weights are updated for every data in each epoch, which is basically that weights goes down the gradient more times in a single epoch. So, we can think it that there are many mini descents in a single epoch descent. Whereas as in batch weights goes down the gradient only once for the whole data once in each epoch. So, stochastic converges faster. The drawback of stochastic is that it takes a little more time to run on hardware as there are many calculations involved in each epoch. So we generally settle between these 2 methods by creating minibatches.

5.7 6c Visualisation of weights

The **Figure(20)** shows the visualisation of weights for four classes. Note that these are without PCA because if we apply PCA the feature vectors get tranformed and the weights relate to the new features. We can observe that the weights are in general same pattern as the actual sign. So classifier basically matches the pattern with the actual sign. Note that the values of weights are higher for lower pixel value of the pattern and vice versa. So the classifier in general normalises the inputs from each pixel and add them up to compare.



(a)Cost vs Epochs for Softmax, lr = 0.07



(b) Accuracy vs Epochs for Softmax, lr=0.07

Figure 19: Plots for Question 6b , Softmax Classification for Aligned Dataset with PCA using Stochastic Gradient Descent, lr=0.07, epochs = 300, dim = 300

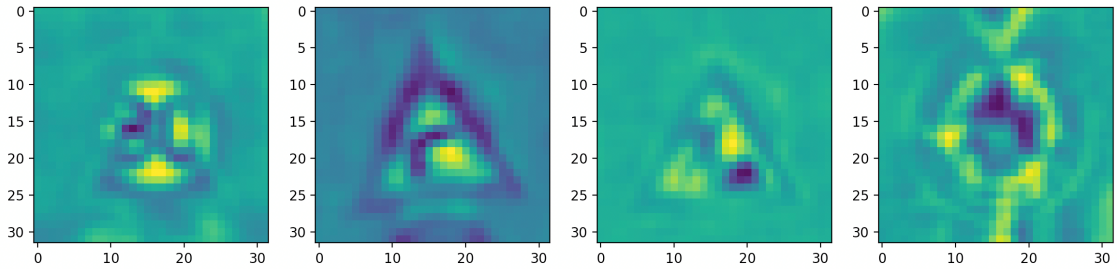


Figure 20: Plots for Question 6c ,Weights Visual without PCA on Class 5(leftmost), Class 20, Class 25, Class 40(rightmost)

6 Team Effort

Everyone contributed almost equally to all the parts.

6.1 Srinivas Rao Daru

Implemented the complete PCA.py file, stochastic gradient descent functions, Softmax gradient descent functions and other functions in network.py file. I also added on related information about these in the final report. I also worked on writing different looping functions to get different plots and graphs.

6.2 Sri Harsha Pamidi

Implemented complete data.py and in network.py, plot functions for all the experiments and the stochastic gradient descent update algorithm. Performed experiments on unaligned and aligned datasets from sections 5b and 5c. Visualized the weights and PCA components for. Generated the multi-class classification performance results report using the confusion matrix.

6.3 Venkata Sai Abhiram

Implemented the complete main.py, Binary Cross Entropy Cost function, MultiClass Cross Entropy Cost function, Logistic Gradient Descent and Softmax gradient descent functions. Put together the Report for Part2.