Project 3: Classification
Introduction to Machine Learning

Yash Narendra Saraf
UB Person No. 50290453
MS Computer Science and Engineering
School of Engineering and Applied Science
ysaraf@buffalo.edu

Abstract

In this project we have are working with two similar datasets i.e. USPS handwritten numeral characters and MNIST human numeral characters. The task to train the model for the task of classification on MNIST dataset and test the output on USPS dataset based on multiple classification techniques like Logistic Regression, Neural Networks, Random Forest Trees, and Support Vector Machine. Compare the performance of different models and create an ideal ensemble classifier based on all the classification models.

1 Datasets

The two datasets used over here are-

1.1 Modified National Institute of Standards and Technology or MNIST Numeral Handwritten Characters

The dataset is widely used to test different models and is a corpus of 60000 images each of dimensions 28*28. The images are of numeral characters 0 to 9.

1.2 United States Postal Service Numeral Handwritten characters

The USPS dataset has the training set of 20000 with 2000 images from each class. The USPS testing set is a 1500 size test set with mixed classes. The good thing about this dataset is that the dataset is way random than the MNIST dataset and has random size images. This implies that the dataset requires a little bit of preprocessing and also training on such an dataset might build up a better model.

2 Theory - Methods Used

According to the project description we are expected to work with four types of models i.e. logistic regression, Neural Networks, Random Forest Trees, and Support Vector Machine. There four are the fundamentals of Machine Learning and have been experimented with thoroughly throughout the project.

2.1 Logistic Regression

The logistic regression is a classification technique. This is used when we are dealing with a classification problem. Since we have 10 output classes we have implemented softmax logistic regression by encoding the target using one hot encoding. This implies that we essentially we have 10 output nodes each corresponding to each output class. Each input neuron is connected to the output bucket by a single link. So since we have added the bias along with the input. We have over here (785*10) weights. The input is a 784 length vector which is formed by flattening the image.

$$prediction = softmax(W^T X)$$

$$prediction = \frac{\mathrm{e}^{\mathrm{x}}}{\sum_{n=1}^{10} e^x}$$

Over here both the prediction and the softmax are (10x1) vectors. The softmax is calculated over the prediction.

The main function of the loss function is to get the idea of how far are we from the actual target and this job is better done by the Cross Entropy loss function which will be shown below. Also the Mean square error is non convex when it comes to logistic regression problem and this might cause us to converge at a point that is not the required minima. Also using the Cross Entropy provides us with the same gradient function which is shown below.

$$CostFunction = -\sum_{n}^{allclasses} target * log(prediction))$$

Now to minimize the cost function we will differentiate the cost function with respect to weights. So that we can use Gradient Descent to update the weights to the optimal values.

Similar to bi-class classification problem after finding the derivative of the cost function we get the a simple gradient equation which is given by

$$Gradient = -(target - prediction)X_j$$

Here we can see that the gradient equation remains the same as the linear regression. Now we use the same L2 regularizer term and the equation for the gradient now is

$$Gradient = -(target - prediction)X_j + \lambda * \sum_{n}^{allWeights} W_j$$

Using the above gradient equation we update all the weights.

To optimize the calculation batch gradient descent has been implemented where the batches of data has been passed to the model one by one. The gradient has been calculated for the entire batch and averaged out and then the gradient has been added to the weights. The calculation for error has also been done for each batch and then the desired graphs has been plotted.

2.2 Neural Networks

Regression is method dealing with linear dependencies, neural networks can deal with nonlinearities. So if your data will have some nonlinear dependencies, neural networks should perform better than regression. Linear regression can be thought as the 2 layer neural network which connects all the inputs to a single neuron in the output layer which has no activation function, and the error is calculated using the mean square error. Now since we have linear equation we can only draw a hyperplane to map the input features to the output features.

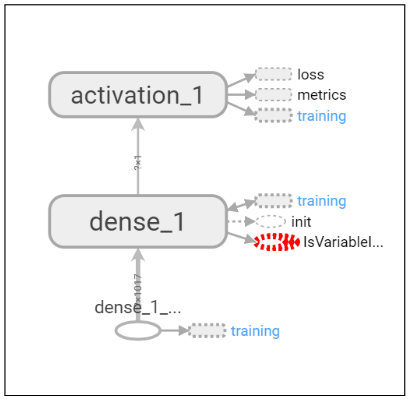


Figure 1: Model for Logistic Regression - Here we have only only layer i.e. the output layer i.e. dense_1 on which softmax is applied i.e. activation_1

But the neural networks are capable of a lot more, i.e. they can successfully map non linearities present in the data as the linear combinations of inputs are first passed to the hidden layer which is passed to a suitable activation function, now this represents nothing but a set of new features which are formed by the linear combination of original features. So this helps in mapping non-linearities in the neural networks.

Here we have implemented two types of neural network based models i.e. Multilayer perceptron network with one hidden layer and the Convolutional neural networks with multiple pooling and convolutional layers. Both these networks perform far better than the basic logistic regression.

2.2.1 Multilayer Perceptron network

This is a simple Neural Network with one hidden layer. The model has been explained further. For the sake of simplicity keras library has been used to implement the neural networks.

Layer	**Number of nodes**	**Weights associated or application**
Input Layer	784 nodes	None
dense_1 (Dense)	784 nodes	(784+1)*512=401920
activation_1 (Activation)	512 nodes	Relu applied to all dense_1 outputs
dense_2 (Dense)	10 nodes	(512+1)*10=5130
activation_2 (Activation)	10 nodes	SOftmax applied to find probablity

Table 1: Model Description for Neural Networks, the conditions considered over here are we have 1 hidden layer and relu has been applied to the hidden layer and also we use ten output neurons for the output buckets.

The simple 3 layer neural network suffices for the above dataset as the data is easily separable into classes. This can also be notices as the logistic regression also performs pretty well on the above dataset. So the keeping the model complexity low the model can be made converge faster and also perform well for the above dataset.

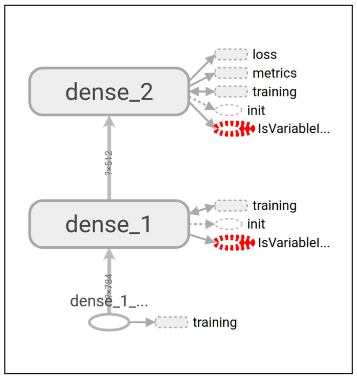


Figure 2: Model for Multilayer Perceptron Neural Network

2.2.2 Convolutional Neural Networks

ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet architecture.

To create a CNN based classifier I referred the following blog

The blog discusses various details on optimizing the results-

Some of the methods used are randomizing the data using the ImageDataGenerator keras function which shifts the images and rotates by small factor to introduce randomness. This makes sure that the model is well generalized.

Other smart trick used here is use of function ReduceLRonPlateau which is used to manipulate learning rate using the model callback. This lets us define the minimum learning rate for the model .

Also multiple convolution layers are stacked so that the feature extraction is optimized.

The model is very expensive computationally. But when it comes to image data opting for Convolutional Neural Networks provide us with the best results.

2.3 Random Forest Classification

Random forest classification is nothing but using ensemble classification over multiple decision trees. The decision trees can be used to create a map of the given dataset. But the issue with the decision trees is they do not have good accuracy on testing as they map the training data perfectly. Decision trees are as the name suggests binary trees with each node as some decision over some parameter in the dataset. So going down the decision trees the model can be evaluated.

The decision tress are created by something known as the gini index. The gini index is scoring metric for building of the decision trees. The main task is to make sure that the value of gini index is 1 for that parameter. The value 1 means that the data can be easily classified i.e. the class is highly biased. For example for all the values of parameter greater than the threshold the class is same. This property is used to make the decision trees. Now if the decision trees are build then we end up overfitting the data given.

So the data is divided into multiple splits and then multiple decision trees are built based on this. The decision trees each then predict the output class and based on some ensemble classification method the output is generated. This method is usually preferred for smaller amounts of data as the training is very quick and also the model is very good in its performance. The model can be used for both the classification and regression tasks.

In this also hyperparameter tuning can be done to make sure the model converges faster or to get a higher accuracy. One of the most important parameters is the number of estimators i.e. the number of decision trees to be used.

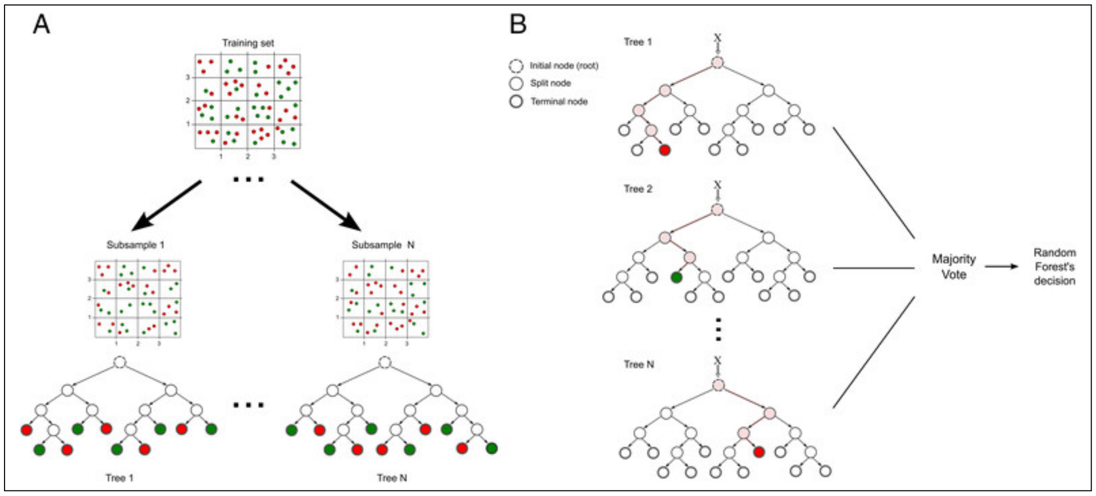


Figure 3: Random forest Working

Firstly, there is the ,n_estimators hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

Another important hyperparameter is max_features, which is the maximum number of features Random Forest is allowed to try in an individual tree. Sklearn provides several options, described in their documentation.

The last important hyper-parameter we will talk about in terms of speed, is min_sample_leaf. This determines, like its name already says, the minimum number of leafs that are required to split an internal node.

2.4 Support Vector Machine

Support Vector Machine are used to find the best hyperplane to divide the classes. The SVM's basic logic involves maximizing the margin between the two classes. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

The hyperplane between the two classes might not always be a linear boundary so using various kernals like polynomial, RBF, or sigmoid kernals the classification boundary can be drawn more clearly. The margin is found by considering only a few terms close to the boundary. Now all these optimizations are defined using the hyperparameters. This is considered one of the best models as this can be used to ignore the outliers and draw the maximum margin decision boundary.

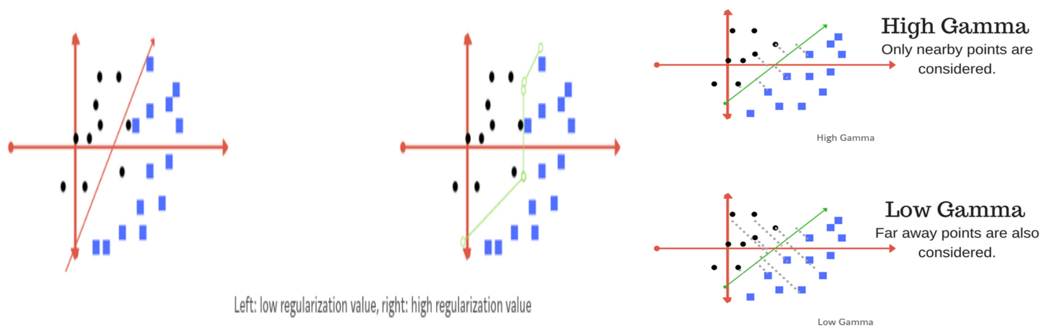

High Gamma
Only nearby points are considered.
High Gamma
Low Gamma
Far away points are also considered.
Low Gamma
Left: low regularization value, right: high regularization value

Figure 4: a.Impact of parameter C on the SVM. b. Impact of value gamma on SVM.

C hyperparameter : large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Gamma hyperparameter : The gamma parameter defines how far the influence of a single training example reaches, with low values meaning far and high values meaning close. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Where as high gamma means the points close to plausible line are considered in calculation.

2.5 Ensemble Classification

When we have multiple classifiers and we want the best predictions, we can essentially combine the strengths of all the classifiers to build a classifier which can give the highest accuracy.

Now there are many ways to perform this. There are many basic ways like maximum voting and weighted averaging which might also result in the drop of accuracy.

The following are the some of the ensemble classifiers which are highly used. But the use of the ensemble classifier is highly dependent on the models which are being used.

Maximum Voting : Maximum Voting is the maximum consensus theorem and it states that the class is the one which most of the models has predicted.

Consider a scenario where we have a classifier like CNN which gives a very high accuracy and other two classifiers which do not perform as well and give an accuracy of about 80-90 percent. So if the maximum voting is implemented then the accuracy will probably be dragged down by the two ill performing classifiers by a bit.

Weighted Averaging : It is an very interesting concept which can be used when the classes adjacent to each other have correlation. So when the classes have correlation then the prediction is made by all the classifiers and then the weighted average of all the models is taken. Now if the classes has nothing to do with correlation then this might be a very bad model.

Consider our MNIST dataset where we have the numeral character recognition. In this dataset the probability of class 1 to be predicted as class 7 is very high compared to the probability of class 1 to be predicted as class 0 or 2 based on the way the are written. So if weighted average is taken then we might end up predicting something which lies between 1 and 7 which is not the correct class. This might even reduce the accuracy value to be lower than the accuracy of the model with lowest accuracy. This essentially highlights the mis-predictions of all the models for the given dataset.

Bagging : Bagging is one of the advanced techniques where the data is first split into multiple portions and then each model then gets trained on a particular dataset. After this is done then for the test set all the models make a prediction and then we use maximum voting or some other combining method for result. This is also used for the Random Forest implementation.

Boosting : Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model. The boosting is nothing but the model is first trained on some dataset and then tested.A new model then is made in such a way so as to correct the errors made by the previous classifier. The final model (strong learner) is the weighted mean of all the models (weak learners).

There are also some other ways to implement the ensemble classifiers i.e. stacking and blending.

According to my intuition a good implementation for ensemble classifiers is probably by checking out the confusion matrix and then checking the strengths of each of the classifiers and using them only for those classes. This can be done using the analysis on the testing dataset.

2.6 No Free Lunch Theorem

In layperson's terms, the No Free Lunch theorem states that no optimization technique (algorithm/heuristic/meta-heuristic) is the best for the generic case and all special cases (specific problems/categories). You can have a technique that is better for the generic case, but it will lose to certain techniques for specific problems/categories. Likewise, a technique for a given problem/category will not be comparable to some techniques for generic cases.

3 Experimentations and Observations

The dataset has been used in 3 ways-

- The training has been done on MNIST dataset and the testing has been performed on MNIST, USPS and also preprocessed USPS dataset.
- The training has been done on the USPS dataset and the testing has been done on the MNIST and USPS dataset.
- The training has been done on the preprocessed USPS dataset and the testing has been done on the MNIST and preprocessed USPS dataset.

3.1 Logistic Regression

Logistic Regression has been implemented using the batch gradient descent using simple mathematical libraries like numpy and math. The implementation has been done by slicing the input array and performing Gradient descent on each batch. The weights has been updated after each batch. The advantages of using the batch gradient descent is that since the dimensionality of the batch is low the computations are faster and also the amount of memory it takes up is low.

Logistic regression is a simple model which uses softmax to give probabilities of the output classes. The model performs quite well when trained on the MNIST and tested on MNIST dataset. But the performance on the USPS dataset is terrible i.e. around 30-35 percent accuracy. This is because the model is very simple and does not extract characteristics pertinent to the images and features. The only way to improve on the performance would be use a complex model or preprocess the data such that it becomes identical to the MNIST dataset.

The simple preprocessing has been done based on the image characteristics to see that is there any improvement in the accuracy. On just resizing the image and placing it into the 28*28 frame gives an accuracy boost of about 10 percent for the logistic regression problem.

The following table summarizes the performance of the model on all the datasets -

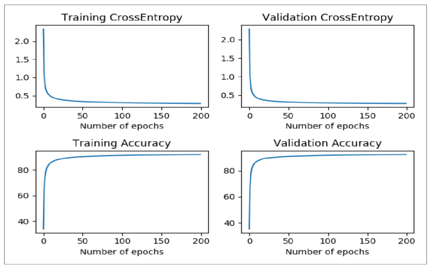
Trained On	Tested On	Accuracy	Cross Entropy
MNIST	MNIST	91.97	0.2856101886
MNIST	USPS	29.8666666667	4.1567802616
MNIST	EDIT_USPS	39.0666666667	2.5640376656
USPS	USPS	77.6	0.7676159481
USPS	MNIST	56.7	1.5165071588
EDIT_USPS	EDIT_USPS	76.1333333333	0.8845204287
EDIT_USPS	MNIST	54.83	1.7707633683

Table 2: Logistic Regression accuracy and error function

3.2 Neural Networks - Multilayer Perceptron

A simple Multilayer perceptron has been implemented with a single hidden layer with 512 neurons. The MLP has the strength to map the non-linearities of the model. So the accuracy for the MLP is way better than the accuracy of the simpler models like Logistic Regression. Using MLP the performance boost is seen for all the three training and testing datasets.

The following table summarizes the performance of the neural network on all the datasets -



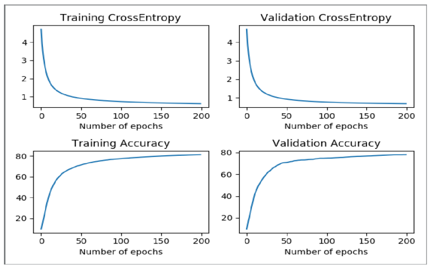
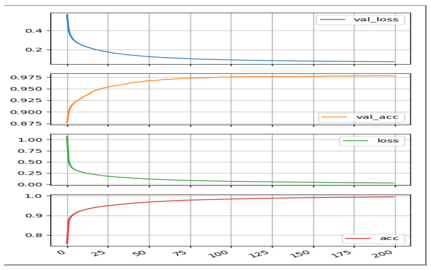


Figure 5: a.Training of logistic Regression model on MNIST b. Training of logistic Regression model on USPS



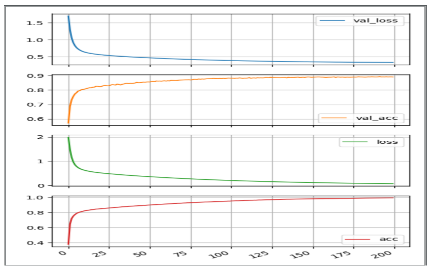


Figure 6: a.Training of MLP model on MNIST b. Training of MLP model on USPS

Trained On	Tested On	Accuracy	Cross Entropy
MNIST	MNIST	0.9782	0.0696089809
MNIST	USPS	0.4573333329	2.5066796637
MNIST	EDIT_USPS	0.5453333333	1.7390202897
USPS	USPS	0.8993333335	0.3373532187
USPS	MNIST	0.7349	0.9192908364
EDIT_USPS	EDIT_USPS	0.9106666662	0.3456383088
EDIT_USPS	MNIST	0.7066	1.3308510105

Table 3: Neural Networks accuracy and error function

3.3 Convolutional Neural Networks

When it comes to image data one of the best models which captures all the features is the CNN. A deep neural network has been created using multiple pooling and convolution layers.

This model has been just been used for experimentation and understand how it works. So before passing the data to the model the data has been randomized i.e. the samples have been zoomed in and out by a factor of 0.1 and also some samples are rotated clockwise or anti-clockwise by an angle of 15 degrees.

The performance of the model is far superior as the main purpose of Convolutional neural networks is to find out the filters which are found using the correlation between the adjacent pixels. This is very important when we deal with images as the main features are how the object is bound together or what are the boundaries which can be considered as the neighboring features.

The performance of the Convolutional Neural Network has been tabulated below -

Trained On	Tested On	Accuracy	Cross Entropy
MNIST	MNIST	0.9957	0.0141327588
MNIST	USPS	0.9233333329	0.2783758962
MNIST	EDIT_USPS	0.9253333332	0.2405052352
USPS	USPS	0.986	0.0496195917
USPS	MNIST	0.9596	0.1441656456
EDIT_USPS	EDIT_USPS	0.9860000003	0.0456802426
EDIT_USPS	MNIST	0.9432	0.2186911916

Table 4: CNN Training and testing accuracy and error function

3.4 Support Vector Machine

Support Vector Machine has been imported from the well known sklearns library which is a repository for many machine learning libraries. The sklearns library parameters has been set by calling the svm method and then the model is fit.

The parameters which are used are-

- C = 5
- gamma = 0.05
- kernal = RBF

The performance of SVM on the MNIST dataset is great as it essentially finds out the hyperplane between the classes. But the performance of the data on the USPS data is bad as the USPS data is much more random data and the SVM does not perform well for that data. For linear kernal accuracies close to 90 are obtained. So using the non linear kernal and considering the closer values only gives a good accuracy. For higher values of gamma the accuracy starts to drop as the outliers now affect the output.

Trained On	Tested On	Accuracy
Trained On	Tested On	Accuracy
MNIST	MNIST	0.9828
MNIST	USPS	0.254
MNIST	EDIT_USPS	0.4966666667
USPS	USPS	0.874
USPS	MNIST	0.6999
EDIT_USPS	EDIT_USPS	0.92
EDIT_USPS	MNIST	0.6372

Table 5: SVM performance table

Performance of the SVM can be seen in the above table.

3.5 Random Forest Classifier

The random forest is one of the fastest classification techniques available. Using 10 number of decision trees the model is constructed which gives an ideal performance for the MNIST dataset. But the USPS dataset performance of the model is not that good as the model is not able to generalize on the USPS dataset because of the randomness in the data.

The performance of the Random Forest Classifier can be studied from the table below -

Trained On	Tested On	Accuracy
MNIST	MNIST	0.9447
MNIST	USPS	0.298
MNIST	EDIT_USPS	0.2886666667
USPS	USPS	0.7573333333
USPS	MNIST	0.5313
EDIT_USPS	EDIT_USPS	0.7506666667
EDIT_USPS	MNIST	0.5864

Table 6: Random Forest Classifiers performance

3.6 Ensemble Classifier

For ensemble classifier a simple version of the maximum voting has been defined where the priority has been given to the classifier with the maximum accuracy. The ensemble classification has been done only for models MLP, RF, SVM, and Logistic Regression. The classification accuracy on testing set among the following models has been for the neural networks model.

Here when implementing the maximum voting as the ensemble method it is observed that even when the training and testing set are the same a accuracy drop has been observed. This is because the models like Logistic Regression and and the Random forest which has slightly lower testing accuracy are pulling the accuracy of the entire model down. This can be avoided if a model is developed which uses the strength of each model. Complex techniques like bagging and boosting can also show some improvement in the performance. Maximum voting is one of the most primitive and in some cases most ideal technique used is not suitable for this scenario.

Trained On	Tested On	Accuracy
MNIST	MNIST	0.973
MNIST	USPS	0.3773333333
MNIST	EDIT_USPS	0.486
USPS	USPS	0.8773333333
USPS	MNIST	0.6887
EDIT_USPS	EDIT_USPS	0.8946666667
EDIT_USPS	MNIST	0.7102

Table 7: Ensemble classifier performance

3.7 Model performance visualized

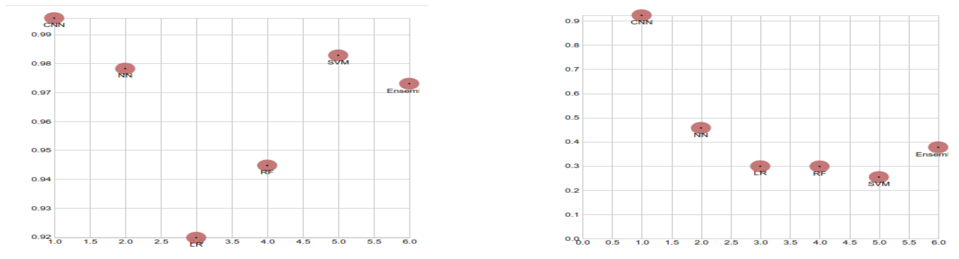


Figure 7: a.Training on MNIST and testing on MNIST b. Training on MNIST and testing on USPS

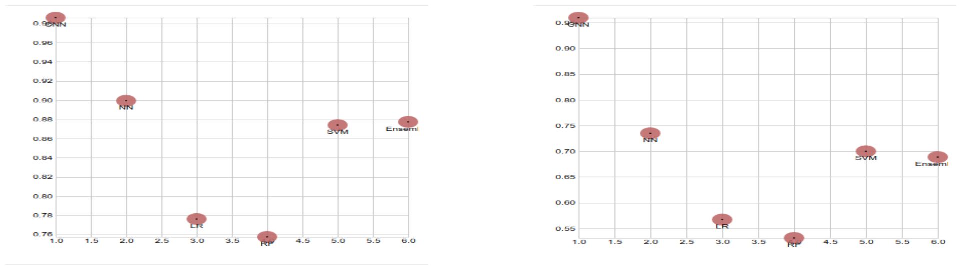


Figure 8: a.Training on USPS and testing on USPS b. TTraining on USPS and testing on MNIST

4 Questions to be answered

- **We test the MNIST trained models on two different test sets: the test set from MNIST and a test set from the USPS data set. Do your results support the No Free Lunch theorem?**

  Ans- Yes, the results do support the ”No free lunch theorem”. When trained on MNIST data for most of the models the accuracy of the order of about 95 percent was obtained on the testing set of MNIST dataset. But then when tested on the USPS dataset then the model does not perform well. The model provides an accuracy of about 30 percent for simpler models which are more focused on mapping the given data with soe generalization. The neural network gives an accuracy of about 50 percent as the model is able to map the non-linearities a little better.

  The convolutional neural network gives an accuracy of 93 percent when the data has been randomized before feeding to the network. This is because the CNN has the capability to learn the relation between the neighboring by learning the filter kernals.

  But for the sake of experimentation when the model is trained on the USPS dataset which in itself is quite random and then the testing is done on MNIST dataset, all models do perform significantly better than their performance earlier. For instance the Logistic regression gives the accuracy of about 50 percent. Thus with proper randomization of data good accuracy can be obtained on the MNIST data.

The model should be trained on the more diverse dataset i.e. here for example an randomized version of the USPS dataset where some basic processing like zooming and rotation is done for random samples. This make sure that the sample is more close to the real world scenario. But performing this kind of processing is not possible for all the datasets. And so we can clearly see how the **"No free lunch theorem"** is valid.

- **Observe the confusion matrix of each classifier and describe the relative strengths/weaknesses of each classifier. Which classifier has the overall best performance?**

  Ans- Looking at the confusion matrix of all the classifiers here are all the conclusions drawn from the data -

  The main reason we plot a confusion matrix is to study the false negatives and false positives.

  On y-axis confusion matrix has the actual values, and on the x-axis the values given by the predictor.

  Convolutional Neural Networks: For the training and testing on the MNIST dataset there is no significant conclusions that can be drawn as the number of mis-classifications are quite low. But when the model is tested on the USPS dataset the accuracy drops to 92 percent from 99.5 percent. Now this is because of the reason that most of the confusion is caused by randomness in the digits 0, 8, and 9. As all of the three contains loops the chance of mis-classification on these classes increases and which is clearly observed from the confusion matrix. Another pair of similar characters are 2 and 7. A lot of randomness is seen in people's handwriting when they draw these two characters. So also the misclassification are observed in these classes.

  Logistic Regression: Even for the logistic regression similar pattern is observed. 8 being misclassified as 0 and 3. All these randomness is because of the randomness in the data. Also there is a significant amount of randomness in the misclassification's in Logistic Regression evenly spread out apart from significant misclassification's for which one of the major reasons is the small complexity of the model. The same pattern has been observed in the cross dataset classifications. The misclassification count for the cross dataset detection is very high, but the accuracies when the model is trained on MNIST and tested on USPS and when the model is trained on USPS and tested on MNIST is different. This is because the the data is USPS is more general so training on USPS and testing on MNIST will obviously give better results.

  MLP: For MLP the results are very similar to the Logistic Regression model as the MLP is nothing but an extension of Logistic regression by introducing a new hidden layer the model complexity has been increased which results in better convergence of the model. All the accuracies for the Neural Network is just slightly better than the Logistic Regression and the pattern observed in their confusion matrix is also same.

  Random Forest: The random forest classifier is based on the decision trees concepts which does not scale well for generalization which can be clearly observed from the cross dataset accuracies. The model is trained by building multiple decision trees and then using the majority voting. So in this the model becomes a little bit more general than the decision trees which map the data exactly but still can't be scaled to other datasets. So even viewing the confusion matrix it is clear that the errors are very high and randomly distributed.

  Support Vector Machine: The SVM is a model very similar to logistic regression but the major difference is the error function. The error function over here tries to maximize the margin between the two classes and then draws the decision boundary. The performance of SVM is better than the Logistic regression model and almost equal to Neural Networks as the decision boundary it chooses it way optimized. So here the randomness in the confusion matrix is very similar to the randomness in the Neural Networks.

- **Combine the results of the individual classifiers using a classifier combination method such as majority voting. Is the overall combined performance better than that of any individual classifier?**

  Ans- For the specific problem the majority voting is not the ideal ensemble model as the accuracies we have for models are of very high order i.e. 98 percent for the neural networks and around 98.5 percent for SVM when trained and tested on the MNIST dataset. The accuracy has been dragged down by a little bit by the other 2 models i.e. Logistic Regression and the RF classifier. For such models an complex ensemble classifier can be used by studying the confusion matrices and understanding the strengths and weakness of each model. Also the increase in accuracy even after implementing such complex algorithms would be quite low as the accuracy of the model is already of very high order of 98.5 percent.

Conclusion

The project dealt with understanding four classification models in depth and able to understand the differences between them. Also to understand how to use data two datasets were provided to experiment with of the same type but different kind of randomness and preprocessing. So concepts like ”No free lunch theorem” and all were discussed.

It is quite clear that for the image data using the convolutional neural networks after scrambling the data can provide the best performance For example here for training on the MNIST and testing on USPS the accuracies of the order of 95 percent were obtained.

But when we are dealing with such datasets individually there is no need for such complex algorithms as the simple algorithms also give a decent amount of accuracy with very low computation compared to the models like CNN. So essentially there is a model complexity vs model accuracy trade-off which is very low. To increase the accuracy even by a minute percentage the complexity can increase by many folds.

For the experimentation of the data the USPS data was preprocessed a bit to observe does that improve the accuracy. On simple preprocessing of resizing it as 22*22 and placing in a 28*28 frame gave a little better accuracy on all the models. So if the data is properly processed like multiple rotated and zoomed in and zoomed out and other processing is performed than the data would be more general to the task and will perform better on any such kind of dataset given that there is some initial proper preprocessing.

A sample output has been added for reference. The sequence of the models is CNN, Logistic Regression, Neural Networks, Random Forest, SVM, and the ensemble model. The same has been repeated over for different datasets. Also find the **references** after the sample output.

```
--------------------------------------------------
Reading all the datasets
--------------------------------------------------
Getting MNIST dataset
Getting USPS entire dataset
Getting USPS entire dataset and editing it as required
----------------------------------------------------------------------------------------------
2018-11-21 19:13:38.097186: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that
this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
2018-11-21 19:13:38.099015: E tensorflow/stream_executor/cuda/cuda_driver.cc:406] failed call to cuInit: CUDA_ERROR_UNKNOWN
2018-11-21 19:13:38.099045: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:158] retrieving CUDA diagnostic
information for host: yash
2018-11-21 19:13:38.099050: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:165] hostname: yash
2018-11-21 19:13:38.099082: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:189] libcuda reported version is: 410.73.0
2018-11-21 19:13:38.099106: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:369] driver version file contents: """NVRM
version: NVIDIA UNIX x86_64 Kernel Module  410.73  Sat Oct 20 22:12:33 CDT 2018
GCC version:  gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.10)
"""
2018-11-21 19:13:38.099116: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:193] kernel reported version is: 410.73.0
2018-11-21 19:13:38.099121: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:300] kernel version seems to
match DSO: 410.73.0
Test CrossEntropy using Convolutional Neural Networks on dataset mnist after training on mnist is:
0.01413275692098796
Test Accuracy using Convolutional Neural Networks on dataset mnist after training on mnist is:  0.9957
Testing Confusion Matrix:
[[ 978    0    1    0    0    0    0    0    1    0]
 [   0 1127    0    0    0    0    3    5    0    0]
 [   0    0 1029    1    0    0    0    1    1    0]
 [   0    0    0 1009    0    0    0    0    1    0]
 [   0    0    0    0  975    0    0    0    0    7]
 [   0    0    0    2    0  888    2    0    0    0]
 [   1    1    1    1    1    0  952    0    1    0]
 [   0    1    4    0    0    0    0 1021    0    2]
 [   0    0    1    1    0    0    0    0  972    0]
 [   0    0    0    0    2    0    0    0    1 1006]]
----------------------------------------------------------------------------------------------
Testing Confusion Matrix:
[[ 957    0    1    2    0    8    8    1    3    0]
 [   0 1109    2    2    0    3    4    2   13    0]
 [   8    7  915   16   11    5   15    9   40    6]
 [   2    1   24  920    1   22    2   11   18    9]
 [   1    3    3    1  911    1   12    4    7   39]
 [   9    2    5   36   10  768   18    8   29    7]
 [  11    3    7    2   12   13  907    0    3    0]
 [   3    7   19   10    9    0    0  945    4   31]
 [   5    8    6   23   10   27   12   12  864    7]
 [  11    4    2   12   36    7    0   27    9  901]]
The Accuracy for mnist when trained on mnist dataset are: 91.97
The Cross Entropy for mnist when trained on mnist dataset are: 0.2856101886248586
----------------------------------------------------------------------------------------------
Test CrossEntropy using Neural Networks on dataset mnist trained on mnist is: 0.06960898444135673
Test Accuracy using Neural Networks on dataset mnist trained on mnist is: 0.9782
Testing Confusion Matrix:
[[ 970    0    0    1    1    1    2    1    3    1]
 [   0 1127    2    1    0    1    2    0    2    0]
 [   4    2 1012    2    1    0    2    6    3    0]
 [   0    0    6  991    0    3    0    3    4    3]
 [   2    0    3    0  962    0    1    2    1   11]
 [   3    0    0    7    1  866    7    1    5    2]
 [   6    3    1    1    2    3  940    1    1    0]
 [   1    6    9    3    0    0    0 1002    1    6]
 [   3    0    6   10    4    5    3    4  937    2]
 [   3    4    0    7    8    4    1    5    2  975]]
----------------------------------------------------------------------------------------------
When the training has been done on: mnist and tested on mnist
Testing Accuracy: 0.9447
Testing Confusion Matrix:
[[ 966    1    1    0    1    2    5    1    2    1]
 [   0 1119    1    5    1    1    2    0    5    1]
 [   9    2  987    9    3    1    3    9    8    1]
 [   2    1   17  942    1   21    0   12    9    5]
 [   5    3    4    0  925    0    6    0   12   27]
 [   7    1    1   35    5  817   13    2    6    5]
 [  13    4    2    1    6   10  915    0    6    1]
 [   0    8   26    5    3    0    2  971    3   10]
 [   7    4    9   18   11   17    6    2  882   18]
 [   9    5    2   16   29    8    3    6    8  923]]
----------------------------------------------------------------------------------------------
When the training has been done on: mnist and tested on mnist
Testing Accuracy: 0.9828
Testing Confusion Matrix:
[[ 974    0    1    0    0    1    1    1    2    0]
 [   0 1128    3    1    0    1    0    1    1    0]
 [   4    0 1015    1    1    0    0    6    5    0]
 [   0    0    1  996    0    4    0    5    4    0]
 [   0    1    3    0  965    0    4    0    2    7]
 [   2    0    1    7    1  872    3    1    4    1]
[    5    2    0    0    2    3  945    0    1    0]
[    0    3    9    1    1    0    0 1004    2    8]
[    2    0    1    6    1    2    0    2  958    2]
[    4    4    2    8    6    2    0    6    6  971]]
```

```
Testing Accuracy: 0.973
Testing Confusion Matrix:
[[  973    0    1    0    0    1    2    1    2    0]
[    0 1126    2    2    0    1    1    1    2    0]
[    4    0 1014    2    1    0    1    6    4    0]
[    0    0   10  987    0    4    0    5    4    0]
[    2    1    1    0  966    0    3    0    2    7]
[    5    0    0   16    1  856    8    1    4    1]
[    8    3    1    1    4    7  934    0    0    0]
[    1    6   16    5    1    0    0  990    1    8]
[    3    0    4   11    6    9    5    4  930    2]
[    9    6    2    9   12    4    0    8    5  954]]
```

```
Test CrossEntropy using Convolutional Neural Networks on dataset usps after training on mnist is:
0.27837590261300404
Test Accuracy using Convolutional Neural Networks on dataset usps after training on mnist is:
0.9233333336512247
Testing Confusion Matrix:
[[122   0   4   0   0   1   0   0  11  12]
[   0 111   9   2  17   1   3   4   1   2]
[   0   0 148   1   0   0   0   0   1   0]
[   0   0   0 150   0   0   0   0   0   0]
[   0   0   0   0 142   0   0   2   3   3]
[   0   0   0   2   0 141   0   0   7   0]
[   0   0   2   0   0   0 148   0   0   0]
[   0   0  11   0   0   1   0 131   3   4]
[   0   0   0   0   0   4   0   0 146   0]
[   0   0   0   0   1   0   0   2   1 146]]
```

```
Testing Confusion Matrix:
[[39  0  9 13 10 24  1  7 20 27]
[15 24  7 16 10 33  0 22 21  2]
[ 5  0 69 14  2 43  7  3  6  1]
[ 7  0 14 73  0 43  0  2  7  4]
[ 0  5  5 16 40 28  2 19 27  8]
[11  0  9 35  0 85  2  4  3  1]
[10  0 30  8  2 34 62  0  1  3]
[24  8  4 63  4 15  0 21  4  7]
[28  0 11 33  8 39  5  3 17  6]
[ 2  7  4 61 11  2  2 18 25 18]]
The Accuracy for usps when trained on mnist dataset are: 29.866666666666667
The Cross Entropy for usps when trained on mnist dataset are: 4.156780261634452
```

```
Test CrossEntropy using Neural Networks on dataset usps trained on mnist is: 2.506679692586263
Test Accuracy using Neural Networks on dataset usps trained on mnist is: 0.45733333357175193
Testing Confusion Matrix:
[[ 40   0  14   6  10   6   2  16  15  41]
[   9  34  22  10  16  12   1  25  20   1]
[   3   1 124   1   2   9   6   0   4   0]
[   6   0  14 102   0  21   0   0   2   5]
[   1   5   6   2  63  18   0  29  23   3]
[   3   0  12  16   0 106   1   1   9   2]
[   8   1  22   2   6  14  83   0   5   9]
[   4  13  14  28   8   2   1  61  16   3]
[  11   0  14  17  12  27   6   4  58   1]
[   1   4   7  28  22   1   0  37  35  15]]
```

```
When the training has been done on: mnist and tested on usps
Testing Accuracy: 0.298
Testing Confusion Matrix:
[[39  2 27  7 22  9 12 11  0 21]
[10 52 13 11 21 11  3 29  0  0]
[11  8 74 13  3 17  5 18  0  1]
[10 13 13 84  2 19  0  7  0  2]
[ 6 24  9  8 43 15  4 32  3  6]
[17  3 14 34  1 68  4  7  0  2]
[26 12 27  8  8 20 39  6  1  3]
[ 3 46 17 30  7 14  3 29  0  1]
[15  9 20 33 14 38  6  3  9  3]
[ 1 22 27 32 19 13  2 23  1 10]]
```

```
When the training has been done on: mnist and tested on usps
Testing Accuracy: 0.254
Testing Confusion Matrix:
[[ 11   0 125   1   2   1   0   0   3   7]
[  10  21  52  12  20  14   1   7  13   0]
[   0   0 149   0   0   1   0   0   0   0]
[   1   0  97  49   0   2   0   0   1   0]
[   0   0  92   2  26   4   0   3  23   0]
[   0   0 106   2   0  42   0   0   0   0]
[   4   0 109   0   0   2  35   0   0   0]
[   2   0 108  13   0  17   1   8   1   0]
[   0   0 103   0   0  12   0   0  35   0]
[   0   0 113   7   0   0   0   3  22   5]]
```

```
Testing Accuracy: 0.37733333333333335
Testing Confusion Matrix:
[[ 38   1  36   3  15   6   7  11   5  28]
 [ 13  34  24  10  17  15   1  23  13   0]
 [  4   2 132   2   0   7   2   1   0   0]
 [  8   1  24  95   0  17   0   1   2   2]
 [  1  10  18   3  51  18   3  21  22   3]
 [  8   0  22  25   0  91   1   3   0   0]
 [ 10   0  54   2   6  14  58   1   1   4]
 [  4  23  28  45   4  16   1  26   1   2]
 [ 17   0  32  22   8  36   3   2  29   1]
 [  1   8  29  41  14   3   1  19  22  12]]
```

```
Test CrossEntropy using Convolutional Neural Networks on dataset edit_usps after training on mnist is:
0.2405052351951599
Test Accuracy using Convolutional Neural Networks on dataset edit_usps after training on mnist is:
0.9253333331743876
Testing Confusion Matrix:
[[116   0   9   2   0   1   3   0   4  15]
 [  0 143   5   1   0   0   1   0   0   0]
 [  0   2 143   3   0   0   0   0   2   0]
 [  0   0   0 148   0   2   0   0   0   0]
 [  0   2   7   0 134   0   0   2   1   4]
 [  0   0   0   0   0 150   0   0   0   0]
 [  0   0   0   0   0   0 150   0   0   0]
 [  0   4  18   9   1   2   0 114   0   2]
 [  0   0   2   0   0   4   0   0 144   0]
 [  0   0   1   2   0   0   0   1   0 146]]
```

```
Testing Confusion Matrix:
[[ 15   0   0   0  30   3   2  12   1  87]
 [  6  52   0   1  11  22   3  39   8   8]
 [  5  11  65   7  17   3  11  18   1  12]
 [  1   2  10  86   0  16   2  26   0   7]
 [  0  16   0   0  59   8   4  12   7  44]
 [  5   4   3   4   4 101   7  12   1   9]
 [  1   1   9   0  37  12  84   2   0   4]
 [  3  20   3   8   3  34   0  50   2  27]
 [  7   6   7   7  23  52  22   2  14  10]
 [  1  14   0   4  17   8   0  33  13  60]]
The Accuracy for edit_usps when trained on mnist dataset are: 39.06666666666667
The Cross Entropy for edit_usps when trained on mnist dataset are: 2.567080098775904
```

```
Test CrossEntropy using Neural Networks on dataset edit_usps trained on mnist is: 1.739020289738973
Test Accuracy using Neural Networks on dataset edit_usps trained on mnist is: 0.5453333333333333
Testing Confusion Matrix:
[[ 10   0   1   0  24   4   1  11   0  99]
 [  0  70   6   2  16  11   1  39   0   5]
 [  2   7  92   5   3   2   5  29   0   5]
 [  1   1   8 119   0   5   0  12   0   4]
 [  0   2   0   0  97  13   1  17   0  20]
 [  5   0   0   4   0 112   0   8  14   7]
 [  1   0   9   1  34   1  99   2   1   2]
 [  1   6   4  12   2   7   0 107   3   8]
 [  5   0  13   5  19  43  10   4  44   7]
 [  0   1   1   6  17   4   0  38  15  68]]
```

```
When the training has been done on: mnist and tested on edit_usps
Testing Accuracy: 0.2886666666666667
Testing Confusion Matrix:
[[ 8  6  6  3 33 15  5 40  0 34]
 [ 0 67  9  6  5 18  0 44  0  1]
 [ 1 17 54  7 11 16  3 36  1  4]
 [ 5 17 10 58  8 27  1 23  1  0]
 [ 0 34  7  7 49 24  0 22  3  4]
 [ 2  6  6 11  5 82  3 30  0  5]
 [ 1 14 19  4 37 18 33 22  0  2]
 [ 1 50  4  6  1 25  1 58  0  4]
 [ 9 10 12 13 20 48 12 13  6  7]
 [ 1 30  5 12 14 12  1 56  1 18]]
```

```
When the training has been done on: mnist and tested on edit_usps
Testing Accuracy: 0.49666666666666665
Testing Confusion Matrix:
[[ 14   0  11   0  19  20   0   1   1  84]
 [  0  51   5   6   9  42   1  34   0   2]
 [  1   1 130   2   2   6   1   7   0   0]
 [  1   0  21 109   0  12   0   4   1   2]
 [  0   0  10   0  94  20   0  11   8   7]
 [  0   0  13   4   0 128   0   3   1   1]
 [  2   0  30   0  26  10  80   0   0   2]
 [  0   5  20  30   2  45   0  43   0   5]
 [  1   0  29   5   4  66   1   1  43   0]
 [  0   0  21  10  18   5   0  19  24  53]]
```

```
Testing Accuracy: 0.486
Testing Confusion Matrix:
[[ 15   0   2   0  30   4   2   8   0  89]
 [  0  67   5   3  11  26   1  34   0   3]
[   1   12  105    4    7    5    2   11    0    3]
[   2    1   10  113    1    6    0   15    0    2]
[   0   15    4    3   91   13    1   13    0   10]
[   1    1    2    6    2  124    0   10    0    4]
[   1    2   17    0   37    6   83    2    0    2]
[   0   26    6   13    1   46    0   54    0    4]
[   5    4   12    6   17   60    8    6   27    5]
[   1   11    1    9   22    5    1   36   14   50]]
```

```
Test CrossEntropy using Convolutional Neural Networks on dataset usps after training on usps is:
0.04961960233127077
Test Accuracy using Convolutional Neural Networks on dataset usps after training on usps is:
0.9860000003178915
Testing Confusion Matrix:
[[146    0    0    0    0    0    0    0    4    0]
 [   0  145    1    0    0    0    1    2    0    1]
 [   0    0  149    0    0    0    1    0    0    0]
 [   0    0    0  150    0    0    0    0    0    0]
 [   0    0    0    0  149    0    0    0    0    1]
 [   0    0    0    2    0  148    0    0    0    0]
 [   0    0    0    0    0    0  150    0    0    0]
 [   0    0    0    0    1    0    0  145    1    3]
 [   0    0    0    0    0    0    0    0  150    0]
 [   0    0    0    0    0    0    0    1    2  147]]
```

```
Testing Confusion Matrix:
[[126    0    2    2    7    1    1    3    3    5]
 [   5  125    0    4    4    3    2    2    3    2]
 [   4    7  112   13    1    2    2    5    1    3]
 [   1    5    9  121    0    9    0    4    0    1]
 [  13    3    1    0   93    9    1    6    3   21]
 [   8    0    1   10    1  116    5    0    3    6]
 [   1    0    9    1    0    4  132    0    1    2]
 [  13    0    0    0    3    0    1  114   13    6]
 [   4    3    4    4    1   11    5    3  112    3]
 [   3    0    0    1    3    1    0   13   16  113]]
The Accuracy for usps when trained on usps dataset are: 77.6
The Cross Entropy for usps when trained on usps dataset are: 0.7676159481459606
```

```
Test CrossEntropy using Neural Networks on dataset usps trained on usps is: 0.33735321402549745
Test Accuracy using Neural Networks on dataset usps trained on usps is: 0.8993333331743876
Testing Confusion Matrix:
[[141    0    2    0    2    4    0    0    1    0]
 [   3  140    0    2    1    1    1    0    1    1]
 [   2    2  130    8    0    1    2    3    2    0]
 [   0    0    1  144    0    4    0    0    0    1]
 [   6   11    0    0  115    2    1    2    1   12]
 [   1    0    0    4    0  140    2    0    0    3]
 [   1    0    3    0    0    0  145    0    0    1]
 [   4    0    0    0    0    0    0  131    9    6]
 [   0    0    5    4    0    6    1    1  127    6]
 [   1    0    1    1    1    1    0    5    4  136]]
```

```
When the training has been done on: usps and tested on usps
Testing Accuracy: 0.7573333333333333
Testing Confusion Matrix:
[[131    1    2    1    5    5    1    1    0    3]
 [   1  141    2    0    1    2    2    1    0    0]
 [  11    9  105   16    0    2    3    3    1    0]
 [   1    2   13  119    0    7    1    3    1    3]
 [  33    3    3    1   86   10    0    2    1   11]
 [  11    1    7   11    7  100    3    0    4    6]
 [   4    2    8    3    1    3  127    0    0    2]
 [   8    1    0    0    3    2    0  125    5    6]
 [   9    0    5    5    4   11    3    5   96   12]
 [   6    1    1    1   14    3    2    9    7  106]]
```

```
When the training has been done on: usps and tested on usps
Testing Accuracy: 0.874
Testing Confusion Matrix:
[[143    0    0    2    1    1    1    0    2    0]
 [   2  133    0    3    2    3    1    4    2    0]
 [   4    0  120   15    1    5    1    0    3    1]
 [   0    0    2  146    0    1    0    0    0    1]
 [  18    0    0    0  108    4    1    3    3   13]
 [   0    0    1   20    0  125    0    0    2    2]
 [   0    0    5    0    0    2  140    0    3    0]
 [   2    0    0    0    1    0    0  130    8    9]
 [   0    0    2   16    0    2    0    1  127    2]
 [   1    0    1    1    2    1    0    1    4  139]]
```

```
Testing Accuracy: 0.8773333333333333
Testing Confusion Matrix:
[[142    1    1    1    2    2    0    0    1    0]
 [   3  138    0    2    1    2    2    1    1    0]
 [   5    4  129    7    0    2    0    2    0    1]
 [   0    0    4  142    0    3    0    0    0    1]
 [  20    2    1    0  108    3    1    2    2   11]
 [   3    0    1   10    0  133    0    0    1    2]
 [   1    0    6    0    0    2  140    0    0    1]
[   6    0    0    0    1    0    0 129    8    6]
[   0    0    5    8    0    5    1    1 126    4]
[   2    0    1    1    1    1    0    7    8 129]]
```

```
Test CrossEntropy using Convolutional Neural Networks on dataset mnist after training on usps is:
0.14416566390832886
Test Accuracy using Convolutional Neural Networks on dataset mnist after training on usps is:  0.9596
Testing Confusion Matrix:
[[ 977    1    0    0    0    0    2    0    0    0]
 [    0 1128    5    1    0    1    0    0    0    0]
 [    1   21  985   11    1    0    6    5    0    2]
 [    0    6    6  992    0    1    0    3    1    1]
 [    1   16    0    0  948    0    1    1    0   15]
 [    2    3    0   30    0  844    4    0    8    1]
 [    5   28    1    0    1    0  920    0    3    0]
 [    0   57    6    0    5    0    0  908    1   51]
 [   23   13    1    1    1    0    0    0  921   14]
 [    0   24    0    1    5    4    0    1    1  973]]
```

```
Testing Confusion Matrix:
[[458   3 256  72   1  52  84   0  52   2]
 [  0 878 111 123   0   0   8  12   1   2]
 [  2   4 921  20   9   5  52  14   2   3]
 [  0   3  66 883   2   7   3  22   3  21]
 [  5 101  15  23 474   1 111   2 243   7]
 [  1  54  39 246   7 410  21   6 105   3]
 [  2  23  80   1   1   2 835   0  13   1]
 [ 16  28 105  96  68  10   3 296  19 387]
 [  1  26 182 352   2  23  20   7 309  52]
 [ 34 106  39 264  62  10  18   9 261 206]]
The Accuracy for mnist when trained on usps dataset are: 56.7
The Cross Entropy for mnist when trained on usps dataset are: 1.5165071588466192
```

```
Test CrossEntropy using Neural Networks on dataset mnist trained on usps is: 0.91929083776474
Test Accuracy using Neural Networks on dataset mnist trained on usps is: 0.7349
Testing Confusion Matrix:
[[ 754    1   94   28    0    5   74    0   24    0]
 [    0 1114   12    3    0    0    1    5    0    0]
 [    0    1  976    5    7    0   27    8    6    2]
 [    0    4   67  917    1    1    1    8    7    4]
 [   13   64   27    7  634    0   87    0  128   22]
 [    3    7   19  126    0  628   20    1   88    0]
 [    3   12   16    2    2    2  890    0   31    0]
 [   41   23  120   30   35   12    1  454   17  295]
 [    1   11  182  118    2    8    5    1  619   27]
 [  115   90   35  160   74   18   14    2  138  363]]
```

```
When the training has been done on: usps and tested on mnist
Testing Accuracy: 0.5313
Testing Confusion Matrix:
[[ 606    5   39   49   13   18  220    1   25    4]
 [    0 1124    8    0    0    1    1    1    0    0]
 [   16   23  815   52    5    7   40   44   20   10]
 [    1   26  146  736    2   19    8   45   13   14]
 [  214   78   43   34  349   25  104   15   78   42]
 [   15   48   91  274   11  339   38    4   65    7]
 [   27   97   56   37   14   16  672    5   34    0]
 [   41   80  146   44  124   30    8  352   21  182]
 [   28  151  208  140   27   59   45   21  186  109]
 [  291   67   85   49  149   59   75   22   78  134]]
```

```
When the training has been done on: usps and tested on mnist
Testing Accuracy: 0.6999
Testing Confusion Matrix:
[[ 549    2  128   83    0    5  208    1    4    0]
 [    0 1118    5    3    0    1    0    8    0    0]
 [    1    3  930   47    8    0   13   10   18    2]
 [    0   20   75  882    0    4    0   23    1    5]
 [    8  108   53    4  669    0   50    4   48   38]
 [    0   49   46  104    5  612   13    2   60    1]
 [    1   39   67    3    1    2  842    0    3    0]
 [    5   35   67   10   34    1    1  505    2  368]
 [    1   43  267  241    3   18    5    2  336   58]
 [   64  106   90   36   77   13   11   33   23  556]]
```

```
Testing Accuracy: 0.6887
Testing Confusion Matrix:
[[ 677    3  109   36    2    8  128    0   15    2]
 [    0 1123    8    2    0    0    0    2    0    0]
 [    2    2  981    4    7    1   21    9    3    2]
 [    0   10   84  891    1    3    1   15    0    5]
 [   27   99   31    4  633    2   79    3   94   10]
 [    3   46   37  200    3  523   17    1   61    1]
 [    4   34   43    4    1    2  867    0    3    0]
 [   31   32  111   28   53    8    1  453    7  304]
 [    4   55  236  203    4   25    6    6  395   40]
 [  153  111   50  105   83   30   35   11   87  344]]
```

```
Test CrossEntropy using Convolutional Neural Networks on dataset edit_usps after training on edit_usps is:
0.04568024255304287
Test Accuracy using Convolutional Neural Networks on dataset edit_usps after training on edit_usps is:
0.9860000003178915
Testing Confusion Matrix:
[[148   0   0   0   0   0   1   0   1   0]
 [  0 146   1   0   1   0   1   0   0   1]
 [  0   0 150   0   0   0   0   0   0   0]
 [  0   0   0 150   0   0   0   0   0   0]
 [  0   0   0   0 145   0   0   0   0   5]
 [  0   0   0   0   0 150   0   0   0   0]
 [  0   0   0   0   0   0 150   0   0   0]
 [  0   0   0   0   1   0   0 144   0   5]
 [  0   0   0   0   0   1   0   0 149   0]
 [  0   0   0   1   0   0   0   1   1 147]]
```

```
Testing Confusion Matrix:
[[123   5   2   0   3   2   4   2   7   2]
 [  9 105   1   2   4   6   3   8  10   2]
 [  0  10 115   9   2   4   5   2   1   2]
 [  2   3   5 128   0   5   1   2   1   3]
 [  5   8   0   0  91   1   8   1   6  30]
 [  8   0   1  14   0 105   7   0   6   9]
 [  2   0   6   0   4   3 133   0   1   1]
 [ 14   8   1   1   1   2   0 112   4   7]
 [  2   1   3   3   2  13   5   1 114   6]
 [  5   1   1   2   3   4   0   7  11 116]]
The Accuracy for edit_usps when trained on edit_usps dataset are: 76.13333333333334
The Cross Entropy for edit_usps when trained on edit_usps dataset are: 0.8845204287435372
```

```
Test CrossEntropy using Neural Networks on dataset edit_usps trained on edit_usps is: 0.3456383088032405
Test Accuracy using Neural Networks on dataset edit_usps trained on edit_usps is: 0.9106666661898295
Testing Confusion Matrix:
[[145   0   0   0   0   0   2   0   3   0]
 [  4 137   0   0   0   2   0   1   6   0]
 [  0   4 137   5   0   1   1   0   1   1]
 [  0   0   3 142   0   4   0   0   1   0]
 [  2   2   0   1 116   0   7   0   2  20]
 [  0   0   0   7   0 140   0   0   0   3]
 [  2   1   0   0   1   0 145   0   1   0]
 [  1   5   0   1   2   0   0 132   4   5]
 [  1   0   2   4   0   7   1   0 134   1]
 [  0   0   0   1   1   1   0   5   4 138]]
```

```
When the training has been done on: edit_usps and tested on edit_usps
Testing Accuracy: 0.7506666666666667
Testing Confusion Matrix:
[[134   0   0   1   0   2   9   0   3   1]
 [  3 138   0   0   0   4   5   0   0   0]
 [  6   6 116   9   0   4   3   4   1   1]
 [  2   1  10 124   1   7   1   3   1   0]
 [ 19   4   0   0  95   2   4   3   3  20]
 [  8   0   5  18   3 104   1   1   6   4]
 [ 13   1   5   2   3   2 121   1   2   0]
 [  3   5   2   1   4   2   2 120   6   5]
 [  9   0   9   7   5  20   6   3  80  11]
 [ 11   0   2   0  18   7   0   8  10  94]]
```

```
When the training has been done on: edit_usps and tested on edit_usps
Testing Accuracy: 0.92
Testing Confusion Matrix:
[[147   0   0   1   0   0   0   0   2   0]
 [  3 141   0   0   0   2   0   2   1   1]
 [  0   2 140   4   0   1   2   0   1   0]
 [  0   0   1 145   0   4   0   0   0   0]
 [  1   2   0   0 118   0   6   3   1  19]
 [  0   0   0  10   0 134   0   0   4   2]
 [  1   0   2   0   1   0 145   0   1   0]
 [  1   1   0   0   1   1   0 134   6   6]
 [  0   1   2   3   0   7   1   0 136   0]
 [  0   0   1   1   0   1   0   4   3 140]]
```

```
Testing Accuracy: 0.8946666666666667
Testing Confusion Matrix:
[[146   0   0   0   0   0   1   0   3   0]
 [  2 141   0   0   0   3   0   1   3   0]
 [  0   5 135   5   0   1   1   0   1   2]
 [  1   0   3 141   0   4   0   0   1   0]
 [  4   3   0   0 117   1   7   1   0  17]
 [  2   0   0  11   1 131   0   0   2   3]
 [  2   0   2   0   2   1 142   0   1   0]
 [  4   5   0   1   1   1   0 129   5   4]
 [  1   0   3   4   0  11   1   0 130   0]
 [  3   0   0   1   2   2   0   6   6 130]]
```

```
Test CrossEntropy using Convolutional Neural Networks on dataset mnist after training on edit_usps is:
0.21869119163146242
Test Accuracy using Convolutional Neural Networks on dataset mnist after training on edit_usps is:  0.9432
Testing Confusion Matrix:
[[  899     0     0     1     1     2    27     1    44     5]
[    2 1112    8    1    0    0    5    2    5    0]
[    0    0  962    9    4    0   11    1   43    2]
[    0    0    6  995    0    5    0    0    3    1]
[    0    1    0    0  966    1    2    0    3    9]
[    0    0    0    6    0  878    1    0    7    0]
[    0    2    1    0    0    1  953    0    1    0]
[    0   11   22   11   15    0    0  720   12  237]
[    0    0    3    0    0    0    0    0  969    2]
[    1    1    0    3    5    7    0    0   14  978]]
--------------------------------------------------------------------------------
Testing Confusion Matrix:
[[297   0   6 142   0 354  80   1  96   4]
 [  0 562 124 429   1   2   2   4  11   0]
 [  3   0 626 195   2  12 107  25  50  12]
 [  0   2  22 891   0   5   5  31  12  42]
 [ 98  10  25  13 441   2 211   0 137  45]
 [ 11   2   8 224   4 486  31   5  78  43]
 [  3   1  28  17   1  14 857   0  37   0]
 [ 60   5  88  60  20  21   6 434   2 332]
 [  6   3  27 268   1  61  24  11 434 139]
 [304   7  38  89  17  22  14   1  62 455]]
The Accuracy for mnist when trained on edit_usps dataset are: 54.83
The Cross Entropy for mnist when trained on edit_usps dataset are: 1.770763368333747
--------------------------------------------------------------------------------
Test CrossEntropy using Neural Networks on dataset mnist trained on edit_usps is: 1.3308510105371476
Test Accuracy using Neural Networks on dataset mnist trained on edit_usps is: 0.7066
Testing Confusion Matrix:
[[732   0   0  42   1  62  74   0  63   6]
 [  0 884  23  78   2   2   6   3 133   4]
 [  3   0 553 143   2   2 102  13 191  23]
 [  1   1  13 920   0   4   3   3  48  17]
 [ 30   2   4   5 620   0 155   0 125  41]
 [  2   0   2  91   0 725  26   0  44   2]
 [  5   0   2   7   1   3 892   0  47   1]
 [ 20   1  62 181  12  13   3 349  14 373]
 [  5   0   4  67   1  10  14   1 844  28]
 [116   3  16  94  22  28  31   0 152 547]]
--------------------------------------------------------------------------------
When the training has been done on: edit_usps and tested on mnist
Testing Accuracy: 0.5864
Testing Confusion Matrix:
[[ 709    0   27   18   11   61  111    0   40    3]
 [    4 1041   52    4    0    5    4   18    7    0]
 [   29   26  678  130    4   13   64   46   30   12]
 [   14   20   84  718    2   73    9   40   37   13]
 [  293    6   15   19  453   28   85    3   40   40]
 [   49   18   20  144   26  538   37    2   51    7]
 [   46   20   43   27    7   34  743    1   34    3]
 [   98   22  126   66   62   43    3  421    8  179]
 [   77   43  144   97   52   66   33   15  363   84]
 [  552   17    8   30   89   46   23    5   39  200]]
--------------------------------------------------------------------------------
When the training has been done on: edit_usps and tested on mnist
Testing Accuracy: 0.6372
Testing Confusion Matrix:
[[ 297    0    4   80    0   24   68    0  506    1]
 [    0 1057    2   15    0    1    0   20   39    1]
 [    0    0  339  170    0    2   31    5  476    9]
 [    0    4   16  912    0    2    0    1   69    6]
 [   38    6   20    4  631    0   88    0  164   31]
 [    2    4    1  140    0  558   18    0  167    2]
 [    2    3   10   11    1    1  805    0  125    0]
 [   20    6   66  155   10    7    0  352   67  345]
 [    0    1   26   91    1   11    5    1  826   12]
 [  151    7   19   47    5   14   15    2  154  595]]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Testing Accuracy: 0.7102
Testing Confusion Matrix:
[[ 685    0    6   46    7   83   83    0   68    2]
 [    0 1066   11   23    0    1    0    4   29    1]
 [    4    0  664  129    1    4   79   23  115   13]
 [    1    2   19  934    0    5    1    7   25   16]
 [   91    5   11    7  600    3  145    1   91   28]
 [    9    4    3  161    6  635   28    0   44    2]
 [    5    4   10   10    2    4  897    0   26    0]
 [   38    8   90  124   20    9    1  403    9  326]
 [    5    3   25  125    2   21   12    6  750   25]
 [  313    7    6   73   19   22   17    0   84  468]]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

Process finished with exit code 0
```

References

[1] Support Vector Machine Medium.com Blogpost *Savan Patel*
`https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72`

[2] Pattern Recognition And Machine Learning, *Book by Christopher Bishop*

[3] Keras Documentation
`https://keras.io/`

[4] Ensemble Classifiers Analytics Vidhya Blogpost *Aishwarya Singh*
`https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/`

[5] Kaggle Blogpost Keras CNN on MNIST *Aditya Soni*
`https://www.kaggle.com/adityaecdrid/mnist-with-keras-for-beginners-99457`