Comparison of Random Forest, K-Nearest Neighbour, Multi-Layer Perceptron and Convolutional Neural Network classifiers

Tutors:
Jiayan Qiu
Jue Wang
Seid Miad Zandavi

Contributors
TongTong Liu tliu8353
Hitesh Wagle hwag3386
Dhanu Jhala djha5117

Abstract

Random Forest, K-Nearest Neighbour, Multi-Layer Perceptron and Convolutional Neural Network are popularly used classifiers to solve many real life problems. In this report all these classifiers are implemented and compared on MNIST Fashion dataset based on the accuracy and other performance parameters like precision, recall, f1 score and confusion matrix. Based on the evaluation metrics result, we will depict and benchmark the performance of all the classifiers mentioned to select the best suitable classifier for MNIST image data set. To improve time and space complexity, Principal Component Analysis (PCA) is used for dimensionality reduction.The analysis shows that k-NN performed better than LR for some classes

1 Introduction

Nowadays, number of images increased dramatically with the development of information technology. So how to make use of trillions of images generated every year is becoming a hot topic. By using supervised classification techniques in machine learning, we can see significant improvement in categorization of images. In this paper we work on comparison of different classification algorithm on the Fashion-Mnist data set. Accuracy, running time, confusion matrix and some metrics like precision, recall are discussed later on.

Classifier chosen in this data set can be broken down into two categories, simple classifiers, Random Forest(RF) and K-Nearest Neighbors(KNN), and complex classifiers, Multi-Layer Perceptron(MLP) and Convolutional Neural Network(CNN), all of them are popular classifiers that widely used in the academia and industry areas. We mainly evaluate performance of classifiers base on accuracy as well as its speed and we working on trade-offs of those two metrics to choose the best parameter for this data set.

Further more, we also look into optimization could be made to any of classifiers and explore how each classifier handled different preprocessing techniques. Preprocessing procedure like normalization, Zero-Center, data augmentation and Principle Component Analysis were talked about to find the best way for classification task.

2 Previous work

Fashion-MNIST data set has been developed by the Zalando Research Team as clothes product database and as an alternative to the original MNIST handwritten digits database[1]. It is a popular baseline for studying the performance of neural networks and other image classification algorithm.

Different methods have been applied to Fashion-MNIST by different teams. There are 260 academic papers cite or conduct experiment on this data set until now according to Google Scholar[2]. Currently, CNN designs have reached up to 95 percent accuracy by implementing together with some other advanced techniques like dense-like connectivity. Support Vector Machines go up to around 88 percent accuracy, both K-Nearest Neighbor and Logistic Regression method have also been applied and both reach an accuracy of around 86 percent as well, which is very close to the test accuracy we obtained in this experiment.

It is worth noting that the even more higher accuracy reached by ResNet(Deep Residual Network) and DenseNets(Dense Convolutional Network). Both of them developed based on CNN model, but made some improvement. Main idea of ResNet is instead of hoping each layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping by using shortcut connections[3]. Compared to ResNet, DenseNet involves for each layer, the feature maps of all preceding layers are treated as separate inputs whereas its own feature maps are passed on as inputs to all subsequent layers. These relevant algorithms can reach nearly 96-97 percent accuracy on Fashion-MNIST data set.

3 Dataset Description

3.1 Data Properties

We have selected MNIST fashion data set to analyze and compare different image classifiers. MNIST fashion data set contains 60000 grayscale images which belongs to 10 classes namely T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag and Ankle boot. Each image has a resolution of 28*28 pixels with each pixel value represents the pixel density. A list of labels is provided which shows the label of a respective image.[8]

Another set of 10000 grayscale images and labels are provided which is only used to test and benchmark the performance of all the image classifiers. The given test images also belongs to a respective class with a resolution property of 28*28 pixels with each pixel value represents the pixel intensity.

3.2 Preliminary Analysis

The data is loaded using an object of MNIST library. Post importing the training and test data set, the initial step is to check the data type and shape of the given data set. It is found that the shape of training image data set is 60000 * 784 where 60000 denotes the number of grayscale images and 784 denotes the feature vector of each image (28*28 pixels). Training label set is in the form of 1 dimensional array which contains number representing a class for the respective image. Similarly the test image data set is provided for testing the performance of each classifier. To train each classifier, it is required to convert the label set which is in a form of class vector (integers ranging from 0-9) to binary class matrix. Below table shows distinct labels and description of each class correspond to the respective label.

4 Methods

4.1 Pre-processing

4.1.1 Zero-Centre and Normalization

Zero-Centre and normalization are both simple but commonly applied preprocessing techniques to regulate the image data. The Zero-Centre procedure involves subtracting the mean of each feature in the data, with images specifically, for convenience it can be common to subtract a single value from all pixels. Normalization refers to scales data points to the unit length[6]. There are two common

Table 1: Labels

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

ways of achieving this normalization in case of images. One is to divide each dimension by its standard deviation, the other involves dividing the maximum values of all data so that the min and max along the dimension is from zero to one. Figure 1 gives a geometric interpretation of common data preprocessing pipeline.

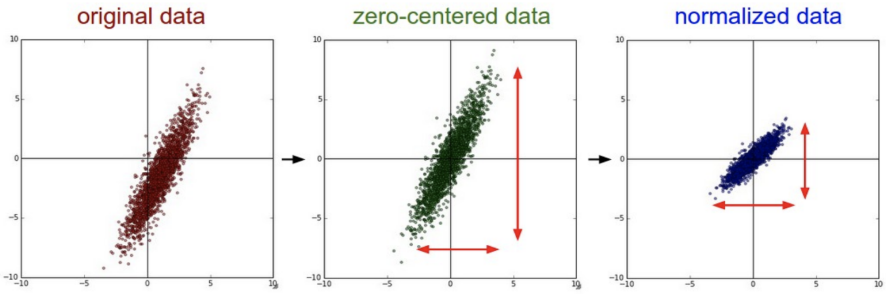


Figure 1: Zero-Centre and Normalization

Applying Zero-Centre and normalization aims to avoid over or under compensate corrections on data features during training. Further, normalization also helps to reduce running time. In case of images, even though the value of each pixel is fixed to be within 0-255, we still need to apply normalization to reduce running time as well as obtain the best possible performance.

4.1.2 Principal Component Analysis (PCA)

PCA is used for dimensionality reduction by using classes to find the variance captured in each of the feature dimension. The usage of PCA is to find a low-dimension set of axes that summarize the whole data. It is used to remove the redundant data which has related property. Instead of dropping down the property, PCA reconstructs the set of properties based on the combination of old ones. PCA looks for properties that show as much variation across classes as possible to build the principal component space. The algorithm use the concepts of variance matrix, covariance matrix, eigenvector and eigenvalues pairs to perform PCA, providing a set of eigenvectors and its respectively eigenvalues as a result. This provides the visualisation of higher dimension data along the principal component (or eigenvectors).

4.2 Classifiers

This section explains different image classifiers implemented on various parameters and functions to compare the performance matrix for MNIST fashion image data set.

4.2.1 Random Forest

Random forest is one of the most popular and powerful supervised machine learning algorithm. It can be used for both classification as well as regression application. As the name suggests, random

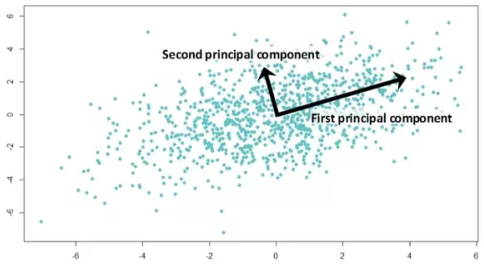


Figure 2: Principal Component Analysis

forest is a forest made up of many decision trees. It works on a divide and conquer methodology. Each tree predicts for a given sample. A weighted voting is carried out over the responses of all decision trees to decide the label for the sample. The forest chooses the classification for which most votes are obtained. In the case of regression, an average of the output of decision trees is taken. Random forest involves ensembling methodology. It is a collection of a group of weak learners, to form a strong learner[1].

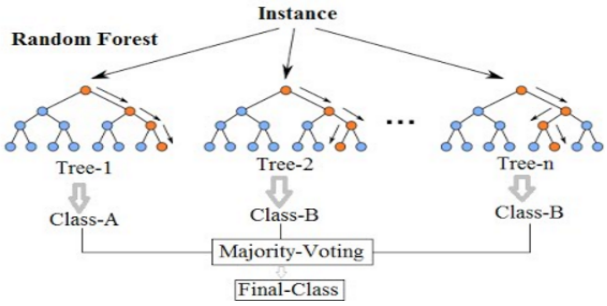


Figure 3: Random Forest

Random forest for N samples with d features, is grown in the following method:

1. Sample of the N training samples is taken at random with replacement.
2. From d features of a sample, m number of features are selected randomly. The best split of these m features is used to split the node. Now as the forest is added with more trees, m remains constant.
3. Each of the decision tree is grown to the maximum extent and no pruning is done.
4. Prediction is made on new data from the predictions of decision trees in the forest.

There are various advantages of using random forest. It is robust to missing values and has a stable accuracy. It seldom overfits the data especially in the classification algorithm. Some of the disadvantages include its not very good at regression as it does not predicts beyond the training data. Random forest is greatly used in banking and finance industry. It can be used to predict loyal and fraud customers. It is used in health sector for disease identification. It is also used in stock markets to predicted expected loss and stock behaviours[].

4.2.2 K Nearest Neighbour

K Nearest Neighbor(KNN) is a non-parametric and lazy learning algorithm. There is no assumption for underlying data distribution and it does not need any training data points for model generation. All training data used in the testing phase, which means training is faster but testing is slower and costlier[4].

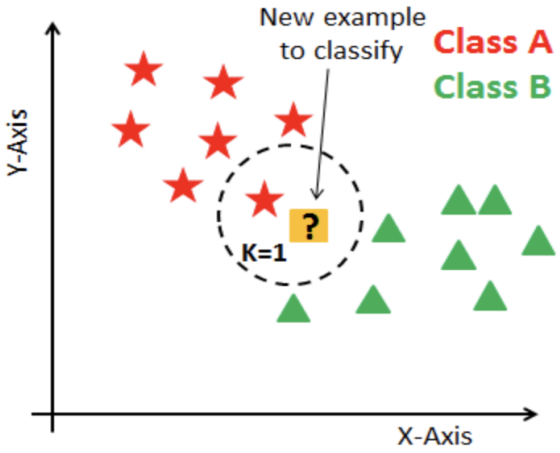


Figure 4: knn

Main steps of KNN algorithm include calculating distance, finding closest k neighbors and voting for labels. As for similarity calculation, the most widely used method is Euclidean Distance. Euclidean distance between two vectors $\vec{x}$ and $\vec{y}$ of m dimension is given by

$$Distance(d) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{1}$$

As for choice of the hyper parameter k, there is no optimal number of neighbors suits all kinds of data set. In the case of a small number of neighbors, the noise will influence on the result; while a large number of neighbors make computation expensive. KNN performs better with lower number of features because the testing phase is costly in terms of time and memory. It requires large memory for storing the whole training dataset for prediction.

4.2.3 Multi-layer Perceptron

A multilayer perceptron (MLP) is a class of feed forward artificial neural network. It consists of at least 3 layers namely, input layer, hidden layer and output layer. To avoid over fitting of the data, multiple hidden layers can be defined by creating pseudo input and output variables which take in differing weights dependent on the layer it is found in. For an m-layered MLP, the formula is:

$$z = \sum_{i=0}^{m} w^n_{j\,i} x_i + b_j \tag{2}$$

Where z is the non linear function for the layer, $w^n_{j\,i}$ is the weights of a layer n and x_i being an input node and b_j is the bias.

$$G(w^T x + b) \tag{3}$$

In case the activation function G is the sigmoid function. Each unit of input layer corresponds to feature element of input image vector and Each output unit of logistic classifier generate a prediction probability that input vector belong to a specified class.

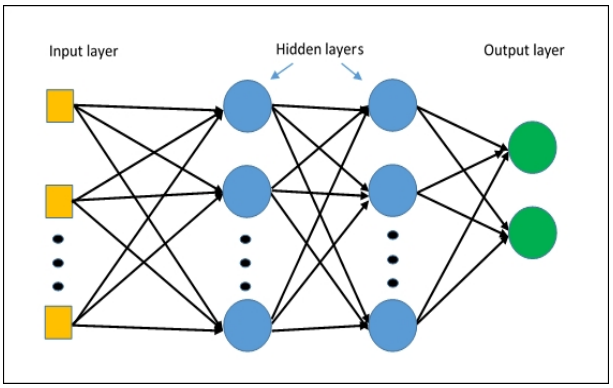


Figure 5: Multi-Layer Perceptron layout

4.2.4 Convolutional Neural Network

The idea of Convolutional Neural Network (CNN) first found its root in the works of Hubel and Wiesel[2]. They carried out series of experiments to detect the response of simple cells to simple attributes like light orientation to complex cells responsive to both light orientation and movement. Thereafter, in 1980, Neurocognitron provided a network architecture formed of simple and complex layer. Later in 1998 Lecun et al[3] provided a CNN architecture that did really well on document recognition. However, the major breakthrough in CNN was in 2012 when Alex Krizhevsky gave the modern implementation of CNN called as AlexNet Alex et al[6].

Convolution Neural Network (CNN) is one of the widely used classifier in the academia and industry. At the core of the classifier lies its powerful capability to learn a model that is invariant to certain transformations of input data. The advantage of CNN over traditional neural network lies in its capability to learn the local information. It exploits the strong correlation that exists between localised data.

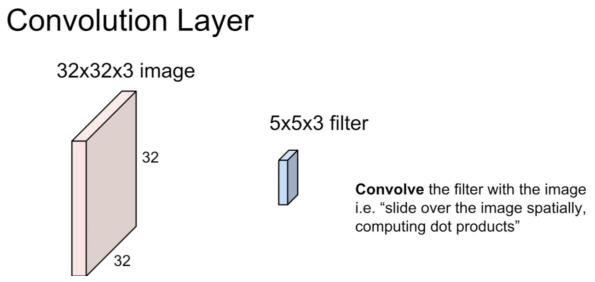


Figure 6: Convolution of input image with filter

The CNN architecture is composed of several hidden layers and a fully connected layer, connected to the output. Each hidden layer is composed of a convolution layer followed by a pooling/down-sampling layer. The original image is convoluted over by a unique filter by taking a dot product between the filter and a part of input. The filter is designed to capture a unique property of the input data. For example it can be used to capture edges, blobs, corners etc. The weights of the filter are learned by the algorithm. After applying the filter we receive activation maps. Activation maps input has an analogy to neuron. An activation function such as relu or sigmoid is applied to the activation map. Further, down-sampling is done by pooling (max, mean). In this way multiple layers are formed leading to learning of hierarchical filters[7]. They enable learning of simple to complex features. For eg: in image data, it can be learning edges to corners and then to blobs. Padding is done to ensure the learning of data around the edges of the input data. Popularly zero padding is used.

Convolutional Neural Networks finds widespread application. They are used for image retrieval and image detection tasks. They are also used for segmentation in self driving cars. CNNs are also used

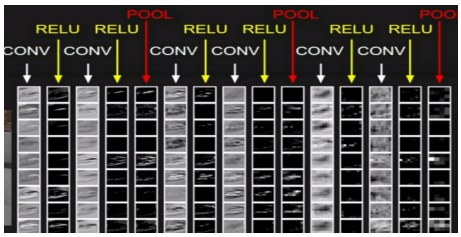


Figure 7: Different layers in CNN

for video classification, pose recognition, medical diagnosis, street sign recognition etc. Thus CNN forms a widespread application domain.

5 Experimental results and Discussion

5.1 Experiment Plan

5.1.1 Random Forest

In the experiment, we run random forest over the MNIST fashion dataset. We use sklearn.ensemble.RandomForest library to implement random forest. Data pre-processing includes normalization and principal component analysis. The dataset is randomly divided in training and validation data in the ratio of 80:20. The classifier is fine tuned to perform well over validation data. The number of components in the random forest and the number of principal components in PCA are changed over different trials. Results were compared to select the best configuration of random forest. We find that the Random forest performed best with 100 decision trees and 120 principal components (0.95% variance). This was obtained from comparison between accuracy and run time of different configurations (Table 4).

5.1.2 K Nearest Neighbour

In this experiments, we use KNeighborsClassifier library provided by Scikit-learn. All Parameters like *weights*, *algorithm,* are set as default, i.e. *weights*='uniform', *algorithm*='auto'. In the first step, data prepossessing, including data normalization and PCA (variance = 0.95), is implemented on the original training data set, then we split it into training data and validation data on ratio of 4: 1. After that, as shown in figure 4, training accuracy, testing accuracy and running time are calculated and compared based on different k (from 1 to 10). Finally k=6 is selected as the best k for this data set.

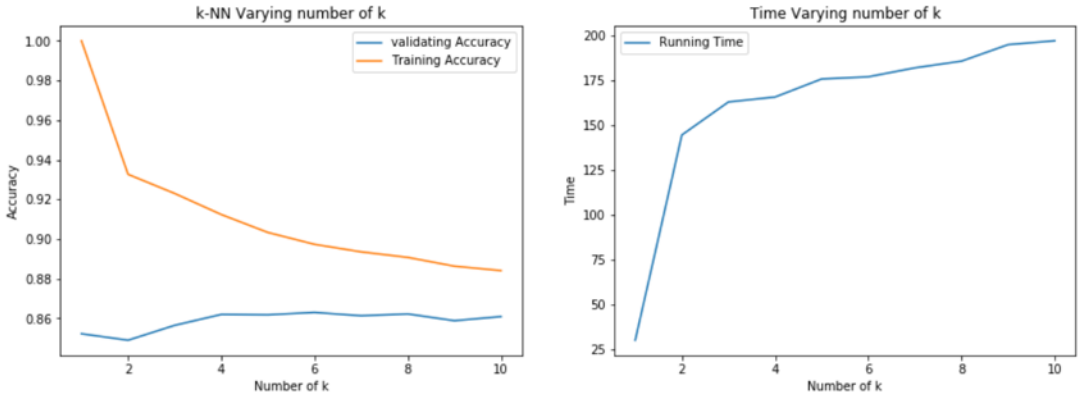


Figure 8: Plot of Accuracy/ Time VS K

We implement KNN (k=6) algorithm on test data set, recording accuracy and running time with respect to different number of components of PCA (n), then n=120 is decided considering trade-off

Table 2: Accuracy and Time

PCA	Accuracy(%)	Time(s)
NO PCA	85.15	2036.53
PCA(N=80)	86.20	42.61
PCA(N=100)	86.13	55.73
PCA(N=120)	86.23	69.67
PCA(N=160)	86.16	94.98
PCA(N=200)	86.16	130.59

of accuracy and running time. Finally we provide the confusion matrix, precision, recall, ROC curve for model evaluation.

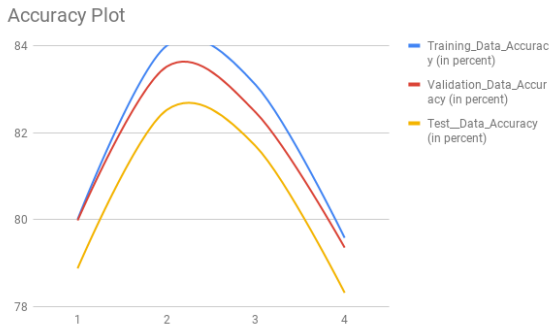
5.1.3 Multi-layer Perceptron

In MLP classifier, MLPClassifier library from sklearn.neural_network is used to predict the labels of test data for evaluating confusion matrix, precision and recall. Initially, data pre-processing is done to normalize the data and post normalization, the given training set of 60000 images is bifurcated into 4:1 ratio to create training data set and validation data set. A custom function is used to perform the bifurcation task which is time efficient in comparison to existing library. In order to implement MLP, Placeholder, weights and Bias needs to be defined explicitly. In our model, we have used 1 input layer, 2 hidden layers and 1 output layer.

Input layer contains 784 nodes to accept all the possible features of a given image of resolution 28*28. The 2 hidden layers have 256 nodes each which represents the pixel density ranging from [0-255] and lastly the output layer contains 10 nodes depicting all the 10 distinct classes.

The main calculation is encapsulated inside the hidden layer where prediction, classification, training, gradient computation and error propagation takes place. MLP is a reliable classifier when non-linear data set is provided in which the intake data has N-dimensions and output signal has M-dimensions. MLP has a disadvantage of selecting the global minima as it does not guarantee the global minima due to its training fashion and hence the code sometimes get stuck with the local minima and treats it as the global minima.

MLP is further performed on training, validation and test data on learning rate at 0.01 with different parameter permutation as shown in below figure 5. Based on the curve having optimum time and accuracy, respective parameters have been selected to obtain best result for the given MNIST image data set.



(a) Plot of Accuracy on various parameters

1	2 hidden layers, 0.01 learning rate, 50 epochs, softmax	78.88
2	2 hidden layers, 0.01 learning rate, 50 epochs, softmax, sigmoid	82.52
3	2 hidden layers, 0.01 learning rate, 75 epochs, sigmoid	81.69
4	2 hidden layers, 0.01 learning rate, 50 epochs	78.32

(b) representation of above parameters

As per the figure 9, we have selected the combination 'b' which has parameters like, 2 hidden layers, 50 epochs, softmax noise reduction and sigmoid activation function at 0.01 learning rate.

5.1.4 Convolutional Neural Network

We implement CNN using Keras library. In the experiment we start with a CNN with one convolution layer and a fully connected layer. Activation maps are obtained by applying a filter of size (3,3) with a stride of (1,1). We select 32 activation maps at the first layer. Post this we implement a fully connected layer of 512 nodes followed by output layer with 10 nodes. This becomes the starting point of our analysis. In next step we increase the number of convolutional layers sequentially. We observe that the training accuracy is close to 99% and validation accuracy around 89%. The results are shown in Figure 8 and Figure 9. Therefore, we observe over fitting on the data.

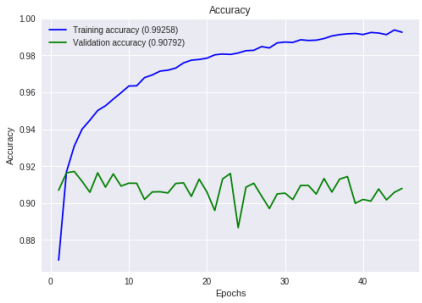


Figure 10: Training and validation accuracy

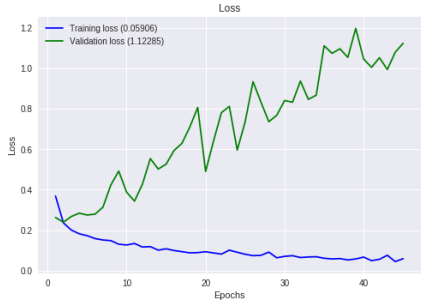


Figure 11: Training and validation loss

To reduce overfitting and simplify our model, we decide to implement dropout and max pooling. We try this for different number of hidden layers and dropout and max pooling. We observe results attached below

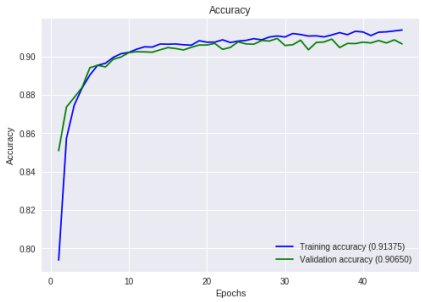


Figure 12: Accuracy with dropout and pooling

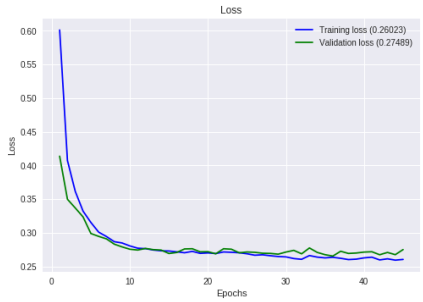


Figure 13: Loss with dropout and pooling

The above results shows that dropout and pooling helps in avoiding overfitting on the data. Therefore, we tried different configurations of dropouts, maxpooling and number of hidden layers to fine tune our CNN model. The learning rate was fine tuned by looking at the loss function. We tried learning rate 0.001 but we observed oscillations in the loss function. Therefore, we took learning rate as 0.0001 which smoothed the loss function. We run the CNN on GPU virtual machine provided by google research colaboratory. The specifications for the same are provided (Table 3). Furthermore, we sum up the results obtained from different configurations of CNN (Table 6). Finally, we decided to build the model used with 2 convolution layer with dropout (0.25) and max pooling. 45 epochs and learning rate of 0.0001 were used.

5.1.5 Discussion

We try to compare the performance of different classifiers on the fashion MNIST dataset. We fine tune each of these models by varying their hyper parameters. We observe that Random forest performs best for 100 trees and 120 Principal components, KNN does well for k=6 and principal components 120. Similarly, Multi Layer Perceptron does well for 2 hidden layers with sigmoid activation

and softmax at output. CNN performed well for 2 convolutional layers along with dropout and max pooling. After doing individual classifier tuning we proceed to compare these classifiers over the total accuracy and running time, confusion matrix and receiver operating characteristic.

Accuracy and Execution time

We find that the execution time taken for Random Forest was less than the time taken for KNN classifier. However the accuracy of KNN is more than Random Forest. This higher accuracy comes at a cost of higher computational and space complexity. KNN do not have a model built up from training data unlike Random Forest. Each time a test is done, it computes distance of test point from all the points in the training data. However, random forest builds a model from training data (eager learner) and use this model to evaluate on testing data. Therefore KNN has limitations over scalability when the dataset is of high dimension. Random Forest on the other hand is scalable and can be applied to large datasets. When we compare the running time of the above two algorithm with Muli Layer Perceptron(MLP), we find that MLP takes more time. This is expected as MLP has large number of weights to learn compared to other two classifiers. Though the accuracy of MLP is less than KNN and RF but MLP can be a robust classifier for complex input data. It can learn powerful features from the input data and can make the classification robust to transformations of input. Therefore we can expect the MLP to be a reliable classifier even if the input data becomes complex, say if the images becomes coloured and contains other objects. However a downside of MLP is that it requires large amount of data to train. This restricts its implementation when the dataset has less samples. The Convolutional Neural Network on the other hand has the highest accuracy among all the classifiers compared. This is because it exploits the spatial correlations existing in the data. It learns a hierarchial features from simple to complex features. Therefore, if accuracy is the prime concern and running time is not then one can choose CNN over other classifiers.

Confusion Matrix

Confusion Matrix can be another performance metric to compare our classifiers. We observe that the Random forest (Table 8) gets confused for the Tshirt/top class and classifies several of Tshirts to Shirt. This is expected due to low discriminability between the two classes. It also confuses Pullover class to Coat and Shirt. The Coat class is also confused by Random Forest and classified to Pullover and Shirt class. A similar kind of confusion exists for KNN classifier (Table 9). The Multi Layer Perceptron also is confused to Pullover and Shirt, when it comes to classifying the Tshirt class. However, it performs well than KNN and Random Forest on Pullover, Coat and Shirt class. The CNN performs the best among all the classifiers. It's least confused between Tshirt, Pullover, Coat and Shirt class. This can be attributed to its capability to learn from simple to complex features involving edges, corners and blobs (Table 11).

Receiver Operating Characteristic curves ROC curves characterise the trade off between positive hits and false alarms. They are a plot of true positive rate versus false positive rate. A point (1,1) will denote the ideal classifier with 100% accuracy. It classifies all the cases correctly. The more the curve is closer to this point, the better the classifier. A comparison of ROC of all the four classifiers from the figure[] leads denotes CNN as the best classifer among all the others on all the classes except class label 6 (Shirt). However for all other classes, the area of ROC curve for CNN is highest.

5.2 Conclusions and Future Work

In conclusion, we can observe that each classifier has its own strength and weakness and therefore their use is task specific. Also, we realised the importance of pre-processing. It not only helped in extracting the most meaningful information, but also help reduce the computation time. This aspect becomes very important due to ever increasing digital data. In the classifiers compared, CNN performed the best over the given dataset. Among KNN and Random Forest, Random Forest can be considered as a better classifier. This is because of its scalability potential and lower runtime. We also learnt the importance of validation which can be very useful to fine tune the hyper parameters of a classifier. We carried out validation performance over different settings to find the best classifier. In the context of future work, multi layer perceptron and convolutional neural network can be further improved by performing data augmentation especially over class label 6 (Shirt). Furthermore, encoders and neural nets can be used for dimensionality reduction. This can help improve the accuracy as it can provide discriminative powerful features to our classifiers like KNN and Random Forest.

Appendix A

EXPERIMENT PLAN

Table 3: Summary of experiments

Algorithm	Experiment Params	Feature Reduction	Implementation	Testing Environment
RF	tree size from 10-1000	PCA	sklearn.ensemble. RandomForest	2.3 GHz Intel Core i5
KNN	default setting	PCA	sklearn.neighbors. KNeighborsClassifier	2.3 GHz Intel Core i5
MLP	0.01 learning rate, 50 epoch	None	sklearn.neural_network MLPClassifier	2.6 GHz Intel Core i5
CNN	2 CNN, Dropout, Max Pooling	None	keras.layers	2.3 GHz Intel Xeon

Appendix B

INDIVIDUAL CLASSIFIER FEATURES

Table 4: RF Accuracy and Time

Model	Validate Accuracy(%)	Test Accuracy(%)	Time(s)
10 trees+NO PCA	86.26	84.96	6
100 trees+NO PCA	88.47	87.43	62
300 trees+NO PCA	88.18	87.70	186
500 trees+NO PCA	88.58	87.51	294
10 trees+120 PCA	82.75	81.89	5
100 trees+120 PCA	86.84	85.81	55
300 trees+120 PCA	87.50	86.02	168
500 trees+120 PCA	86.85	85.93	277
10 trees+160 PCA	81.44	81.44	6
100 trees+160 PCA	85.48	85.81	67
300 trees+160 PCA	85.35	86.02	208
500 trees+160 PCA	85.55	85.93	341

Table 5: KNN Accuracy and Time

Model	Accuracy(%)	Time(s)
NO PCA	85.15	2036.53
PCA(N=80)	86.20	42.61
PCA(N=100)	86.13	55.73
PCA(N=120)	86.23	69.67
PCA(N=160)	86.16	94.98
PCA(N=200)	86.16	130.59

Table 6: CNN Accuracy and Time

Model	Accuracy(%)	Time(s)
2 CNN	89.06	1316
4 CNN	90.40	2363
2 CNN + Dropout + Max Pooling	91.17	783
4 CNN + Dropout + Max Pooling	89.50	843

RESULTS

Table 7: Summary of experiments

Model	Description	Accuracy	Total Precision	Total Recall	F-Score	Time
RF	120PCA	85.88	0.86	0.86	0.86	57
KNN	120PCA	86.23	0.87	0.86	0.86	62.31
MLP	0.01 learning rate and 50 epoch	76.47	0.88	0.87	0.87	125
CNN	2CNN + Dropout +Max Pooling	91.17	0.91	0.90	0.91	783

CONFUSION MATRIX

Table 8: **RF(100 trees) + 120PCA**

K value	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankleboot
T-shirt/top	850	0	16	30	6	7	74	0	17	0
Trouser	4	958	5	24	5	0	2	0	2	0
Pullover	13	0	788	10	113	0	65	0	11	0
Dress	28	4	7	890	31	2	32	0	5	1
Coat	1	1	104	38	799	1	48	0	8	0
Sandal	0	0	0	0	0	923	0	44	5	28
Shirt	170	0	116	30	100	2	557	0	25	0
Sneaker	0	0	0	0	0	30	0	930	0	40
Bag	2	0	4	9	3	14	5	5	956	2
Ankle boot	0	0	0	0	0	23	0	39	1	937

Table 9: **KNN(k=6) + 120PCA**

K value	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankleboot
T-shirt/top	832	0	15	16	5	1	119	1	11	0
Trouser	6	969	4	14	4	0	2	0	1	0
Pullover	19	2	785	10	101	0	82	0	1	0
Dress	27	5	11	874	41	0	37	0	5	0
Coat	1	1	107	22	781	0	85	0	3	0
Sandal	0	0	0	0	0	896	0	55	1	48
Shirt	156	0	105	24	82	0	618	0	15	0
Sneaker	0	0	0	0	0	9	0	955	0	36
Bag	1	0	5	5	6	1	12	6	962	2
Ankle boot	0	0	0	0	0	2	0	27	1	970

Table 10: **MLP(learning rate = 0.01, epochs = 50)**

K value	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankleboot
T-shirt/top	899	1	13	20	2	2	58	0	5	0
Trouser	20	960	0	13	4	0	3	0	0	0
Pullover	113	1	758	8	78	1	40	0	1	0
Dress	76	5	11	872	24	0	11	0	1	0
Coat	55	1	64	35	810	1	33	0	1	0
Sandal	28	0	0	1	0	950	0	10	2	9
Shirt	218	1	65	23	73	0	612	0	8	0
Sneaker	21	0	0	0	0	16	0	940	0	23
Bag	21	1	8	4	7	5	9	6	939	0
Ankle boot	22	0	0	1	0	9	1	41	0	926

Table 11: **CNN(2 Convolution layer+ dropout + maxpooling)**

K value	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankleboot
T-shirt/top	759	0	10	30	9	2	185	0	5	0
Trouser	0	986	0	9	2	0	1	0	2	0
Pullover	14	1	820	9	58	0	98	0	0	0
Dress	7	6	10	918	26	0	33	0	0	0
Coat	0	3	24	28	882	0	63	0	0	0
Sandal	0	0	0	0	0	991	0	7	0	2
Shirt	56	6	25	31	75	0	802	0	5	0
Sneaker	0	0	0	0	0	23	0	956	1	20
Bag	0	3	2	4	2	1	8	2	977	1
Ankle boot	0	0	0	0	0	5	0	30	0	965

ROC curves

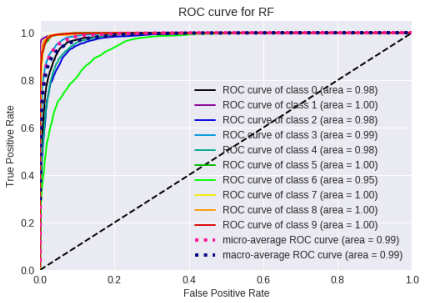


Figure 14: ROC curve for Random Forest

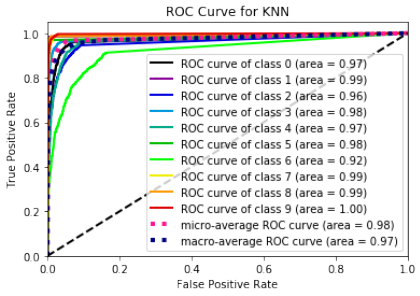


Figure 15: ROC curve for K- Nearest Neighbor

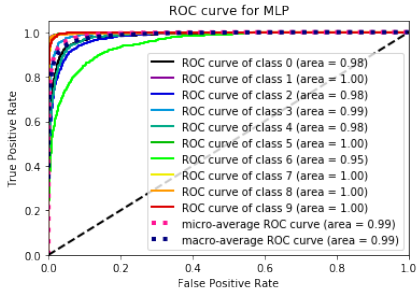


Figure 16: ROC curve for Multi-Layer Perceptron

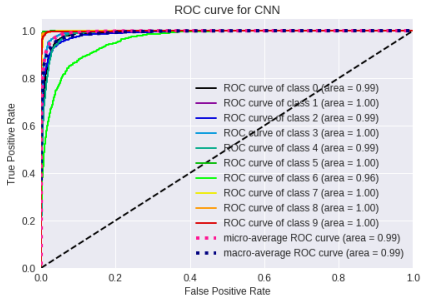


Figure 17: ROC curve for Convolutional Neural Network

A Instructions to run code

To run all the 4 classifiers, Jupyter notebook is required with python version 3 which can be installed through below steps,

1. Download Python v3 from ”https://www.python.org/downloads/release/python-371/” and install.
2. once the python is installed, kindly verify from command prompt by using ”python” command. If version is displayed on command prompt then python is installed successfully or else, kindly visit python community to debug the issue.
3. Post installing Python, open command prompt and run ”pip3 install jupyter”.
4. Once the Jupyter is installed, use the same command to install required repositories to be used in the classifiers like ”pip3 install numpy, pip3 install pandas”.
5. Open Jupyter notebook from command prompt by using ”jupyter notebook” command.

To run the code, open the classifiers notebook from the respective path and select it to open the code. Once the code is open, change the path of input data in each classifier notebook by providing the complete path in cell 2.

Post providing the path, the code is ready to run and should be compiled and run to obtain the results. Please note, the parameter configuration is set to best suitable configuration which is obtained after the analysis on various parameter combinations. Also, the result may vary depending upon the system configuration. To obtain similar results mentioned in this report, kindly use the system with configuration mentioned in the table 3.

B Contributions

All: Report

Dhanu Jhala: Implementation of Random Forest and CNN

Hitesh Wagle: Implementation of Multi Layer Perceptron

Tongtong Liu: Implementation of data pre-processing and KNN classifier.

References

[1] Corinne Dahinden, Isabelle Guyon *An Improved Random Forests Approach with Application to the Performance Prediction Challenge Datasets*, Citeseer, 2009.

[2] D. H. Hubel and T. N. Wiesel *Receptive fields of single neurons in the cat's striate cortex*, The Journal of Physiology 1959.

[3] Y. Lecun ; L. Bottou ; Y. Bengio ; P. Haffner *Gradient-based learning applied to document recognition*, IEEE Journal 1998

[4] Akshay Padmanabha, Christopher Williams, *K-nearest Neighbors*, `https://brilliant.org/wiki/k-nearest-neighbors/`

[5] Ahmad Basheer Hassanat, *Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach* , International Journal of Computer Science and Information Security, `https://arxiv.org/ftp/arxiv/papers/1409/1409.0919.pdf`

[6] Alex Krizhevsk, Ilya Sutskever, Geoffrey E. Hinton *ImageNet Classification with Deep Convolutional Neural Networks*, NIPS 2012

[7] Introduction to convolutional Neural Networkhttps://www.youtube.com/watch?v=vT1JzLTH4G4, Oxford University 2017

[8] https://github.com/zalandoresearch/fashion-mnist