Implement neural network and convolutional neural network for the task of classification.

Shwetasree Chowdhury

Person Id: 50296995

Department of Computer Science

State University of New York at Buffalo

shwetasr@buffalo.edu

Abstract

The classification task will be that of recognizing an image and identify it as one of ten classes specified. The dataset to be used is the Fashion-MNIST dataset which uses clothing images of size 28 x 28 pixels (grayscale images). The dataset has been classified/trained using three approaches:

1. Neural Network with one hidden layer, fully implemented in python
2. Neural network with multiple hidden nodes, using Keras libraries.
3. Convolutional Neural Network using Keras libraries.

The takeaway from this project is to be able to successfully predict the labels (with maximum accuracy and minimize the loss) from a test dataset of 10000 features after successful training on the training dataset of 60000 features and 784 attributes/labels.

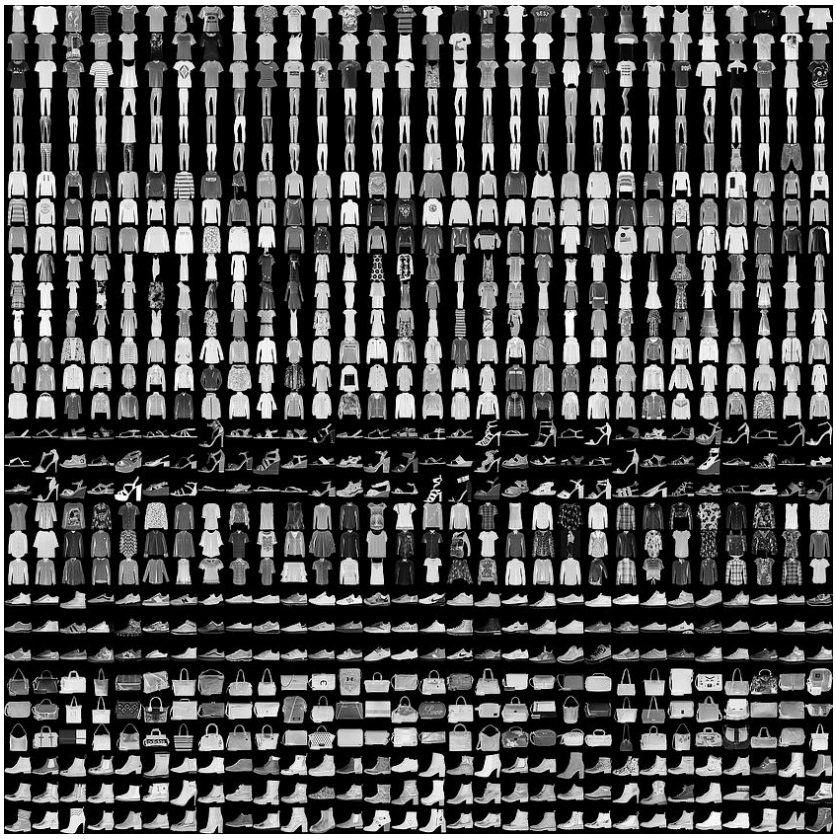
Problem Statement

The fashion MNIST is a dataset comprising of 70000 features. Each image is a 28x28 grayscale reproduction, associated with a label from 10 classes. Labels so represented are noted in the next section. We aim to train our neural network/CNN models on this dataset and use the training to predict the labels from another test data set comprising of 10000 examples. Neural Network and CNN has been implemented in a mix of 3 different approaches, using libraries and also coded from scratch. It is a sort of supervised learning problem where we are given inputs and corresponding correct outputs and our task is to find the mapping between the inputs and the outputs, learn from them and then test that training on an unknown data set to see if the algorithm can ascertain the correct labels. Neural Network is an excellent choice to solve such classification problems because of the complexity of the database and the kind of classification required for this dataset, unlike logistic regression, which works well for a binary classification.

1.1 Dataset Used

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. We intend Fashion-MNIST to serve as a direct **drop-in replacement** for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

Here's an example how the data looks (*each class takes three-rows*):



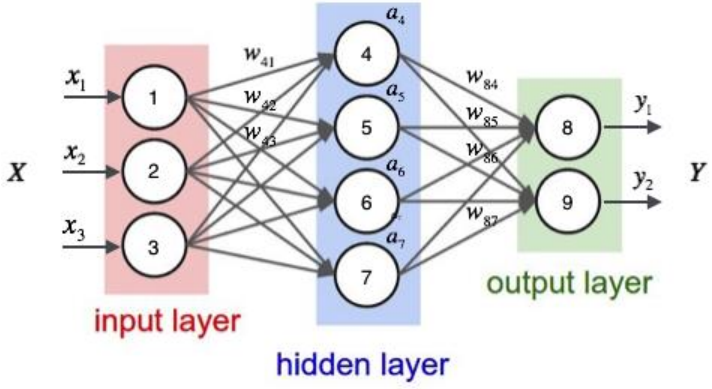
Label	Description
0	T-shirt/top
1	Trouser
2	Pullover

Label	Description
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

1.2 What are neural networks and how do we implement them?

A simple Perceptron (single hidden layer) is simply a logistic regression classifier where the inputs are first transformed using a non-linear function so that it becomes linearly separable and are then multiplied with weights to get the final outputs. In multilayer perceptron, there are more than 1 hidden layers.

The ability to predict comes from the multi-layered structure of the network. Neural network models generally have the same structure of an input layer, one or more hidden layers, and an output layer and the hidden layers perform a non-linear transformation of its input. We assemble a collection of layers to form models and the simplest one is the sequential model that consists of linear stack of layers.



Where X is the input given, "w" are the associated weights "a" is the activation function. The perceptron has the following features:

- **Dense layer** : it is the layer that declares the number of neurons, weight and biases to perform the linear transformation on the data.
- **Hidden Layers**: these are the layers present after the first visible/dense layer and these are not directly exposed to the input values
- **Activation function** : It is required to perform non-linear transformation on the data so that it can learn and solve more complex tasks. This output is sent to the next layer.
- **Dropout** : it's used to prevent overfitting and lies in [0,1]. Some fraction of the nodes are dropped randomly so that machine learning algorithm doesn't get biased, and is able to generalize.
- **Loss function**: it is the wrapper around our prediction function that tells us how good our model is. Its slope tells us how to change our parameters to increase accuracy. The aim is to minimize the loss function.
- **Optimizer**: this parameter specifies how the training is going to take place like the learning rate.
- **Metrics**: this parameter is used to monitor the training process. We use accuracy.
- **Output layer** : this is the last dense layer that is responsible for giving the output values in the desired form.

How does neural network work?

- In the early phases of the training, the neural network makes some random predictions, these predictions are matched with the correct output and the error or the difference between the predicted values and the actual values is calculated and is called as the loss/ cost function. The goal of the training is to learn to minimise the cost/error associated with each prediction with each epoch.

- Neural networks function in two phases: feed-forward and back propagation.
- In the feed-forward part of a neural network, predictions are made based on the values in the input nodes and the weights. The number of input nodes are generally the number of attributes associated with the dataset.
- Initially, random weights and bias are assigned.
- The nodes in the input layer are connected with the output layer via three weight parameters. In the output layer, the values in the input nodes are multiplied with their corresponding weights and are added together. Finally, the bias term is added to the sum.
- In the multi-class classification, the output layer will have K needs corresponding to K number of classes. Softmax function is used in the last layer for this purpose and the class is determined by taking the maximum probability.
- To find the minima of a function, we can use the **stochastic gradient descent** algorithm which works by finding the partial derivative of the cost function with respect to each weight and bias and subtract the result from the existing weight values to get the new weight values.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial s_i}\frac{\partial s_i}{\partial w_{ji}} \quad \text{……. (1)}$$

$$\begin{aligned}
\frac{\partial E}{\partial y_i} &= \frac{-t_i}{y_i} + \frac{1-t_i}{1-y_i}, \\
&= \frac{y_i - t_i}{y_i(1-y_i)}, \\
\frac{\partial y_i}{\partial s_i} &= y_i(1-y_i) \\
\frac{\partial s_i}{\partial w_{ji}} &= h_j
\end{aligned}$$

..... (2)

- The derivative of a function gives us its slope at any given point. To find if the cost increases or decreases, given the weight value, we can find the derivative of the function at that particular weight value. If the cost increases with the increase in weight, the derivative will return a positive value which will then be subtracted from the existing value.

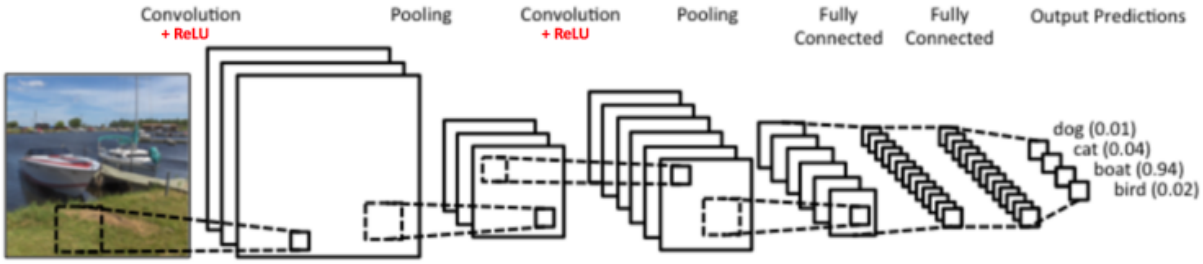
$$\frac{\partial E}{\partial s_i} = y_i - t_i$$

$$\frac{\partial E}{\partial w_{ji}} = (y_i - t_i)h_j$$

..... (3)

How does convolutional neural network work?

- A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is a m x m x r image where m is the height and width of the image and r is the number of channels
- The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m−n+1.
- Each map is then subsampled typically with mean or max pooling over p x p contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger inputs.
- Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map



2. Implementation

In order to work on about any dataset, there are a few steps that we have to follow in order for NN or CNN to be implemented on.

1. Importing datasets
2. Dataset normalization and preprocessing
3. Data partitioning into Training, Validation and Testing Sets
4. Training dataset with Neural Network of one hidden layer Using Stochastic Gradient
5. Training using multi-layer neural network- Keras
6. Training using CNN - Keras
7. Tuning of hyper-parameters.
8. Testing of each test-dataset against the same hyper-parameters to ascertain which tuning fits well.

2.1 Importing Datasets

The Fashion MNIST datasets can be imported using keras library by invoking *keras.datasets.fashion_mnist.load_*data(). It can also be directly downloaded from the official git page of fashion MNIST dataset as a .gz file and imported locally. We have used the latter approach. Four different datasets comprise of the fashion MNIST package. Train data, train label, test data and test label, i.e the label for the test and the train data are already bifurcated into separate datasets. Given below are the dataset sizes.

```
train_data shape: (60000, 28, 28) train_label shape: (60000,)
test_data shape: (10000, 28, 28) test_label shape: (10000,)
```

2.2 Dataset normalization and Preprocessing

The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values, usually required for datasets with different kinds of values. This is required since if the features in the data set have values in different ranges, it takes the model a long time to converge and achieve the global/local minima. Hence, we do normalization to get the feature values in similar ranges and help the model to converge more quickly.

- For the purpose of this project, normalisation is done by dividing each image by 255 (range of pixels). The data is then concatenated together and shuffled using a common seed for the model to train well and learn on randomised data.
- The next step followed is to separate the data back into its original dataset size for ease of training and testing.
- Once done, we use one hot encoding on the train label and test label. For categorical variables where no such ordinal relationship exists between one category and the others in one dataset, the integer encoded variable is removed and a new binary variable is added for each unique integer value. This helps in better representation and manipulation of categorical data.

2.3 Data partitioning into Training, Validation and Testing

The data sets in this project are already separated into training and test data sets and labels.

NN using Stochastic gradient:
Training Data: 60,000 rows, 784 labels | **Testing data**: 10000 rows, 784 labels

NN using Keras:
Training Data: 50,400 rows, 784 labels | **Validation:** 0.16%~ 9600 rows | **Testing data**: 10000 rows, 784 labels

CNN using Keras
Training Data: 50,400 rows, 784 labels | **Validation:** 0.16% ~ 9600 rows | **Testing data**: 10000 rows, 784 labels

2.4 Training dataset with Neural Network of one hidden layer Using Stochastic Gradient

As previously explained, Neural network works on a very simple approach. It has two phase- the feedforward and back propagation. At the end of the feedforward phase, the cost or loss is calculated and sent back to the initial layers in order to come to a conclusive value of weights and biases which will have minimum loss.
For the purpose of our project, we have constructed a basic neural network with one hidden layer.

- The no of hidden nodes in the hidden layer is taken 128. After working with 64 nodes, the model was not training poorly as compared to 128.
- Initial weights and biases are considered using the random function and are of the same dimension as the input dataset so as to enable matrix multiplication.

```
num_hidden_nodes = 128
output_nodes = 10
input_nodes = sgd_nn_train_data.shape[1]

wh = np.random.rand(input_nodes,num_hidden_nodes)
wo = np.random.rand(num_hidden_nodes,output_nodes)
bh = np.zeros((1,num_hidden_nodes))
bo = np.zeros((1,output_nodes))
```

- The forward propagation is simple dot multiplication of weights with input and summation of bias. The sigmoid value of this, is then transported to the next layer and so on. Since we have just one hidden layer, we tabulate the activation value of the data using a sigmoid function and then a softmax function from the hidden to output traversal. Activation functions provide nonlinear transformations on data and the softmax in particular compresses the value so as the summation of the entire probability of the whole row remains at 1. This approach is useful for categorical classification in our case.

```
def forwardPropagation(data):
    zh = np.dot(data, wh) + bh
    ah = sigmoid(zh)
    zo = np.dot(ah, wo) + bo
    ao = softmax(zo)
    return ah,ao
```

- The back propagation works by calculation of derivatives. Loss/Error is calculated by the difference between the predicted value and target value. The error is minimised by differentiating it with respect to the weights. This gives rise to newer weights and bias, which use the learning rate to expedite the learning process.

```
# BACK PROPAGATION
# calculating cross entropy
ao_delta = ao - sgd_nn_train_labels_1hot
ao_delta = ao_delta/sgd_nn_train_labels_1hot.shape[0]

#calculating derivatives
zh_delta = np.dot(ao_delta,wo.T)
ah_delta = zh_delta * sigmoid_derivative(ah)

#updating weights
wo -= lr * np.dot(ah.T,ao_delta)
bo -= lr * np.sum(ao_delta,axis=0,keepdims=True)
wh -= lr * np.dot(sgd_nn_train_data.T,ah_delta)
bh -= lr * np.sum(ah_delta, axis=0)
```

- The loss is calculated and printed for every 100 epochs.
- Accuracy is calculated using the confusion matrix.
- Plots and confusion matrix are printed.

2.4 Training using multi-layer neural network- Keras

- For this implementation, we are using the Keras open-source network library to define the neural network model.
- We use a function getNNmodel() to define the architecture of the neural network model- Sequential in nature (implying the layers are placed sequentially), the activation is defined differently for each layer, with softmax being defined in the last layer, the reason for which is discussed above.
- Dropout factor is defined to avoid overfitting as neurons develop co-dependency among themselves which leads to less power among neurons, leading to over fitting
- The model is compiled by defining the optimiser, loss type and metrics.
- Optimisers say how the model is updated and is based on the data and the loss function. Adam is an extension to the classic stochastic gradient descent and is popular because it's shown to be effective and efficient.
- Cross-entropy is the default loss function to use for a multi-class classification problem and it's sparse because our targets are not one-hot encodings but are integers.
- Metrics — monitors the training and testing steps. *Accuracy* is a common metric and it measures the fraction of images that are correctly classified.
- The model is then trained by specifying a **validation split of 16%** and then specifying no of epochs.
- On noticing overfitting, while training, **EarlyStopping** was initiated to prevent the same.
- Accuracies and performances are plotted using the history feature of the model.

2.4 Training using CNN- Keras

- The function getCNNModel() is used to define the architecture of the CNN model which comprises of convolutional and pooling layers.
- Number of convolutional layers are 3 in number, starting with one of size (3,3), interspersed with 2 pooling layers. The filter maps can then be flattened to provide features to the classifier.
- Max pooling helps to prevent over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.
- A flatten operation on a tensor reshapes the tensor to have the shape that is equal to the number of elements contained in tensor
- Given that the problem is a multi-class classification, we know that we will require an output layer with 10 nodes in order to predict the probability distribution of an image belonging to each of the 10 classes. This will also require the use of a softmax activation function. Between the feature extractor and the output layer, we can add a dense layer to interpret the features, in this case with 128 nodes.
- Model is compiled using the same metrics, categorical cross entropy loss and same optimizer as neural network defined in above implementation.

2.5 Tuning of Hyper-parameter

Parameters are tuned to bring out the change in the accuracy, cost, precision and recall. Gradient Descent is the process of minimizing a function by following the gradients of the cost function.

This evaluates and updates the coefficients every iteration called stochastic gradient descent to minimize the error of a model on our training data.

- ***Learning rate, number of epochs, no of nodes and no of layers, optimiser are changed to find the best accuracy***
- Cost is calculated using the log loss function. Cost is also differentiated with respect to weights, in order to obtain the optimal weights for the training in the next epoch
- Confusion matrix is calculated using the inbuilt sklearns metrics library and the accuracy, recall and precision are calculated for the test set.
  Thus represented as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

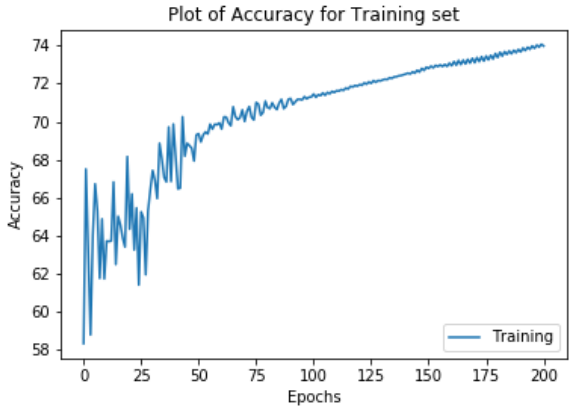
Where TP = true positive, TN = true negative, FP = false positive and FN = false negative.

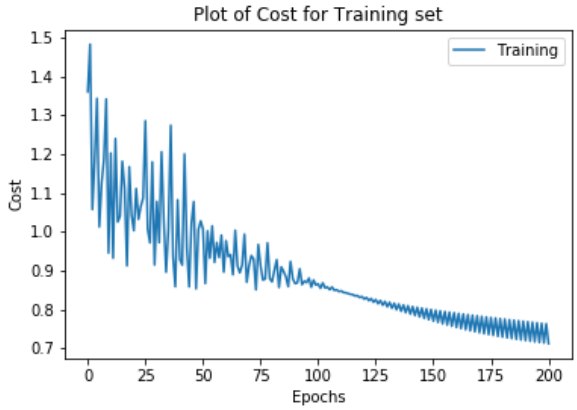
- The results are reported and analysed.

HYPER PARAMETER TUNING RESULTS

Neural Network with Gradient Descent

Case 1: Learning rate 0.5 epoch 200 nodes = 128

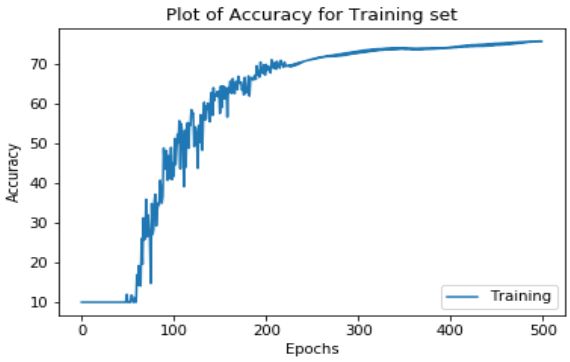


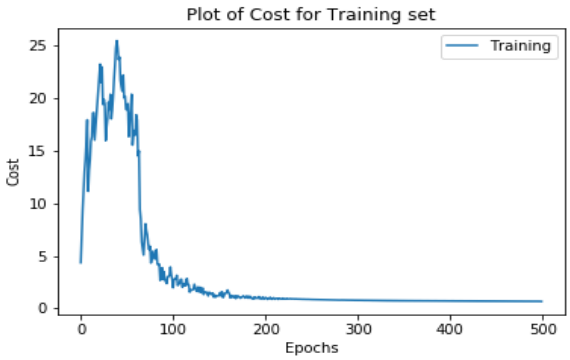


```
Neural Network with SGD accuracy for training set:  69.42%
Neural Network accuracy with SGD for Test set: 67.9%

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	905	0	38	10	0	40	0	2	0	0
Trouser	1	0	4	477	1	491	0	23	0	0
Pullover	28	0	753	90	0	114	0	2	0	0
Dress	1	0	22	772	0	199	0	6	0	0
Coat	0	0	3	5	479	4	203	93	212	0
Sandal	12	0	31	262	0	566	0	46	0	0
Shirt	0	0	0	0	5	0	886	3	106	0
Sneaker	1	0	20	10	7	36	8	912	2	0
Bag	0	0	1	6	2	0	69	10	912	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 2: learning rate 0.5 epoch 500 nodes = 128

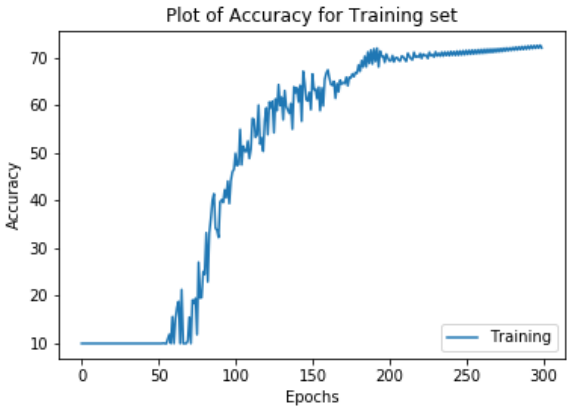


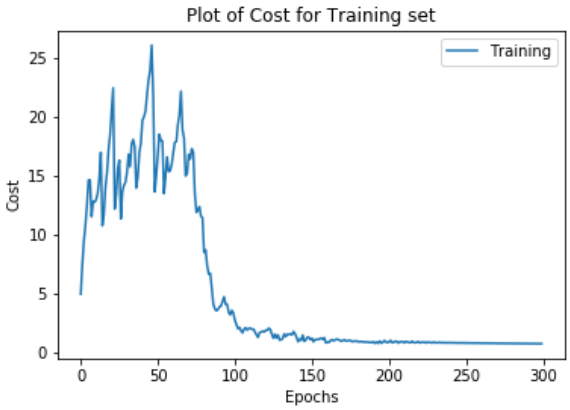



```
Neural Network with SGD accuracy for training set:  75.74%
Neural Network accuracy with SGD for Test set: 75.41%

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	906	15	61	9	0	0	0	2	0	0
Trouser	2	706	14	244	1	4	0	15	0	0
Pullover	12	18	863	53	0	11	0	5	0	0
Dress	2	177	60	736	1	9	0	11	0	0
Coat	1	0	2	0	768	0	96	23	108	0
Sandal	5	233	78	352	1	63	0	44	0	0
Shirt	0	0	0	0	39	0	874	4	83	0
Sneaker	2	36	11	14	16	2	5	909	2	0
Bag	0	1	0	0	17	0	49	3	928	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 3: learning rate 0.5 epoch 300 nodes = 128

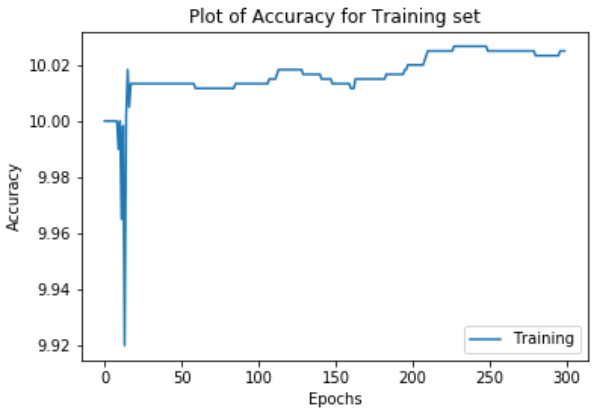


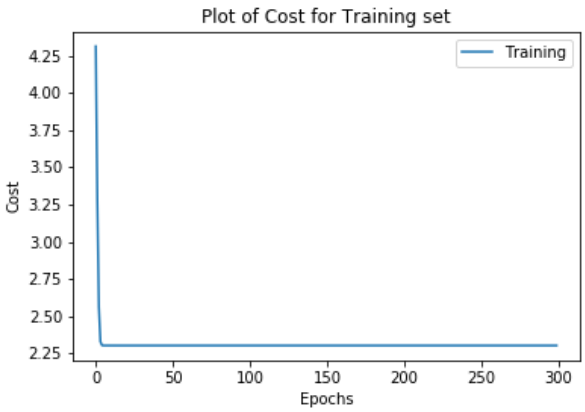



```
Neural Network with SGD accuracy for training set:  72.14%
Neural Network accuracy with SGD for Test set: 72.53%

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	906	21	53	3	0	0	0	2	0	0
Trouser	0	879	14	77	2	2	0	9	0	0
Pullover	8	24	848	41	0	4	0	4	0	0
Dress	3	552	53	372	2	0	0	10	0	0
Coat	0	1	4	0	773	0	86	20	115	0
Sandal	3	512	77	98	1	13	0	46	0	0
Shirt	0	0	0	0	46	0	866	5	83	0
Sneaker	1	55	21	4	18	0	4	887	3	0
Bag	0	3	2	2	23	0	58	4	907	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 4: learning rate 0.1 epoch 300 nodes = 128



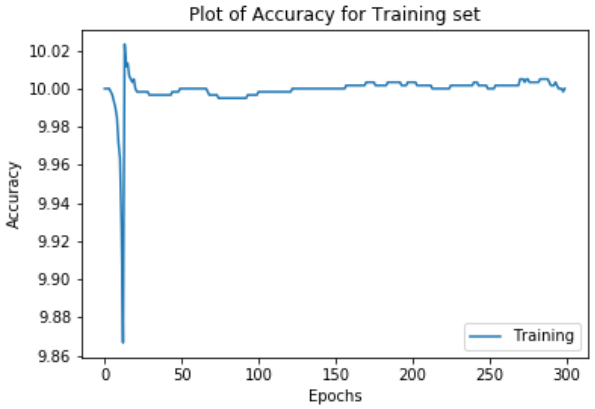


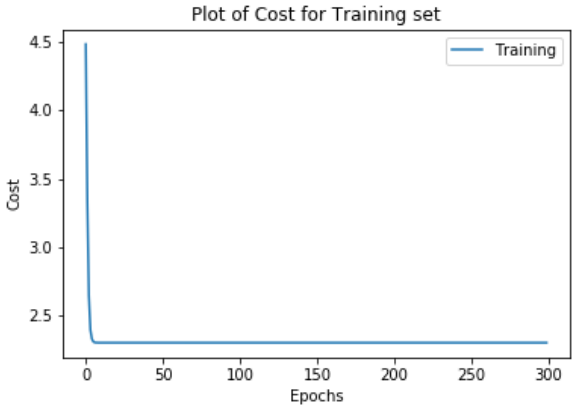
Neural Network with SGD accuracy for training set: 10.02%
Neural Network accuracy with SGD for Test set: 10.02%

Confusion Matrix:

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	1000	0	0	0	0	0	0	0	0	0
Trouser	1000	0	0	0	0	0	0	0	0	0
Pullover	1000	0	0	0	0	0	0	0	0	0
Dress	999	1	0	0	0	0	0	0	0	0
Coat	956	2	2	1	0	0	13	16	8	0
Sandal	1000	0	0	0	0	0	0	0	0	0
Shirt	998	0	0	0	0	0	0	0	2	0
Sneaker	999	0	1	0	0	0	0	0	0	0
Bag	1000	0	0	0	0	0	0	0	0	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 5: learning rate 0.05 epoch 300 nodes = 128

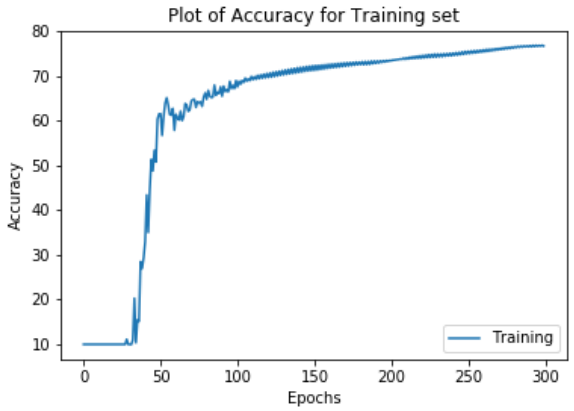


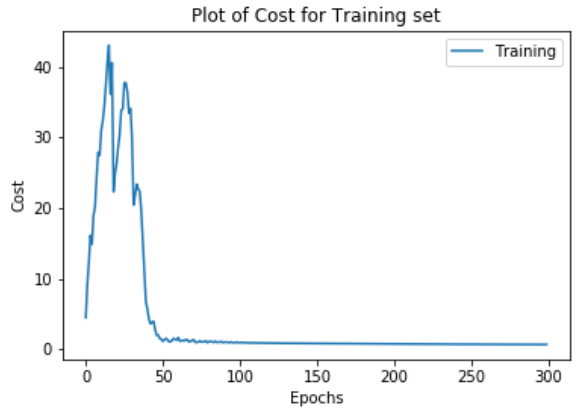



```
Neural Network with SGD accuracy for training set:  10.0%
Neural Network accuracy with SGD for Test set: 9.98%

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	0	0	0	0	0	0	0	0	0	0
Trouser	0	0	0	0	0	0	0	0	0	0
Pullover	0	0	1	0	0	0	0	0	0	0
Dress	0	1	0	0	0	0	0	0	0	0
Coat	3	0	18	8	0	0	2	4	8	0
Sandal	0	0	2	0	0	0	0	0	0	0
Shirt	0	0	0	0	0	0	0	0	1	0
Sneaker	0	0	1	0	0	0	0	0	0	0
Bag	0	0	0	0	0	0	0	0	0	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 6: learning rate 0.7 epoch 300 nodes = 128

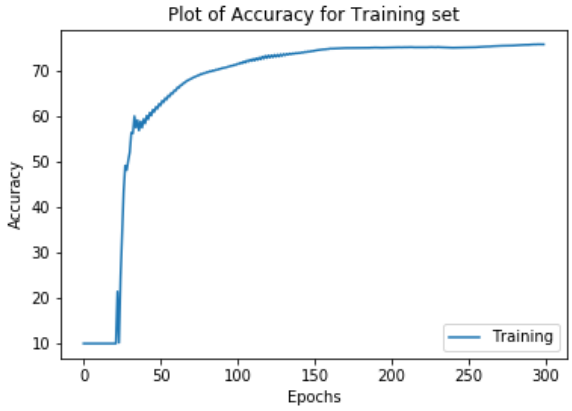


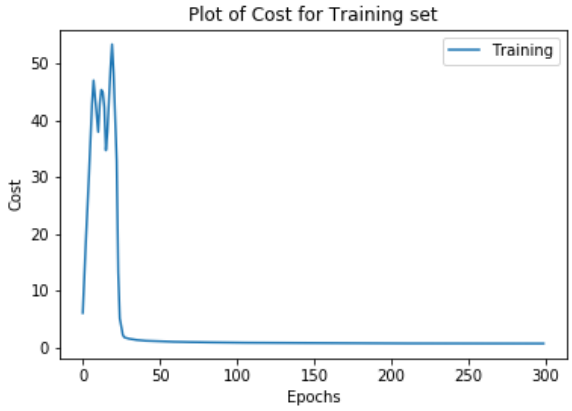


```
Neural Network with SGD accuracy for training set:  76.72%
Neural Network accuracy with SGD for Test set: 75.76%

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	911	9	53	14	0	6	0	2	0	0
Trouser	2	538	16	236	2	173	0	19	0	0
Pullover	6	5	882	30	0	48	0	5	0	0
Dress	1	112	99	662	2	116	0	8	0	0
Coat	1	1	2	0	787	0	92	27	88	0
Sandal	3	98	108	197	8	389	0	40	0	0
Shirt	0	0	0	0	31	0	898	2	69	0
Sneaker	1	4	37	7	17	27	5	898	2	0
Bag	1	1	2	1	19	0	71	3	902	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 7: learning rate 0.9 epoch 300 nodes = 128

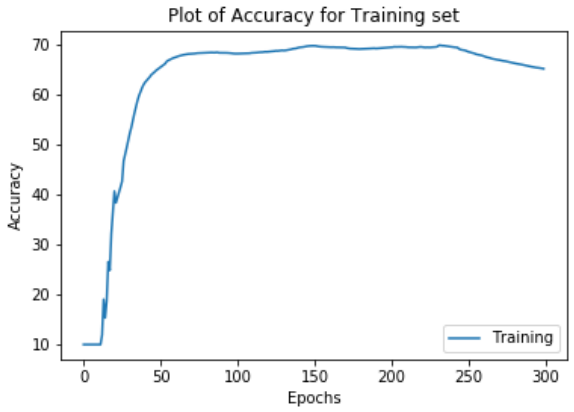


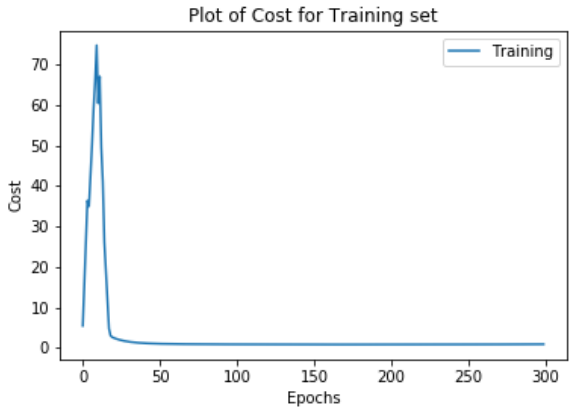


```
Neural Network with SGD accuracy for training set:  75.79%
Neural Network accuracy with SGD for Test set: 74.89%

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	899	15	65	15	0	2	0	2	0	0
Trouser	1	600	14	203	3	131	0	21	1	0
Pullover	19	5	824	97	0	41	0	2	0	0
Dress	2	139	57	656	5	133	0	7	0	0
Coat	1	0	10	0	720	3	116	35	114	0
Sandal	3	134	102	196	7	411	0	43	0	0
Shirt	0	0	0	0	36	0	864	1	99	0
Sneaker	2	14	25	13	10	21	7	907	1	0
Bag	1	0	2	2	23	0	41	2	929	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 8: learning rate 1.2 epoch 300 nodes = 128



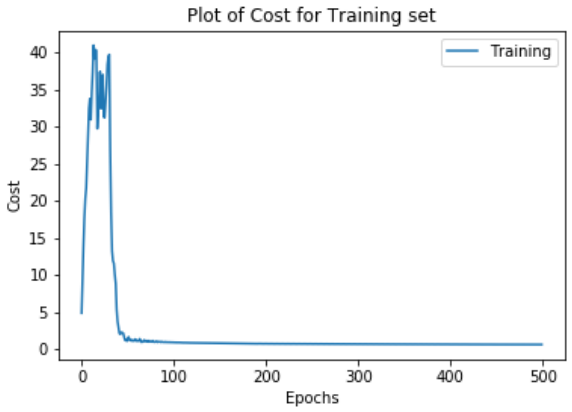


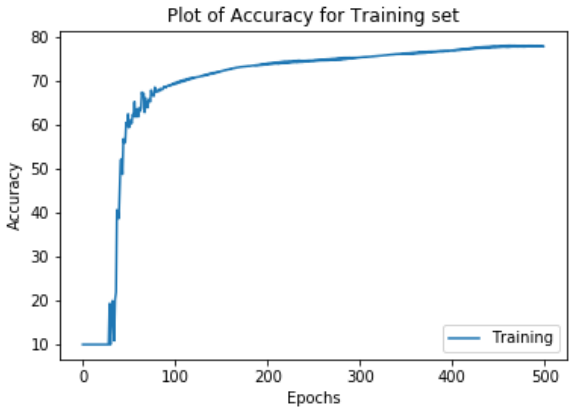
Neural Network with SGD accuracy for training set: 65.26%
Neural Network accuracy with SGD for Test set: 64.23%

Confusion Matrix:

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	887	25	59	15	0	10	0	2	0	0
Trouser	2	680	6	189	0	92	0	23	0	0
Pullover	8	8	759	150	0	37	0	22	0	0
Dress	1	324	26	467	0	164	0	18	0	0
Coat	1	0	2	1	224	0	363	48	360	0
Sandal	2	196	74	316	0	269	0	70	0	0
Shirt	0	0	0	0	23	0	957	8	12	0
Sneaker	2	31	9	4	10	16	3	919	1	0
Bag	1	0	5	0	22	0	249	30	693	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 9: learning rate 0.7 epoch 500 nodes = 128

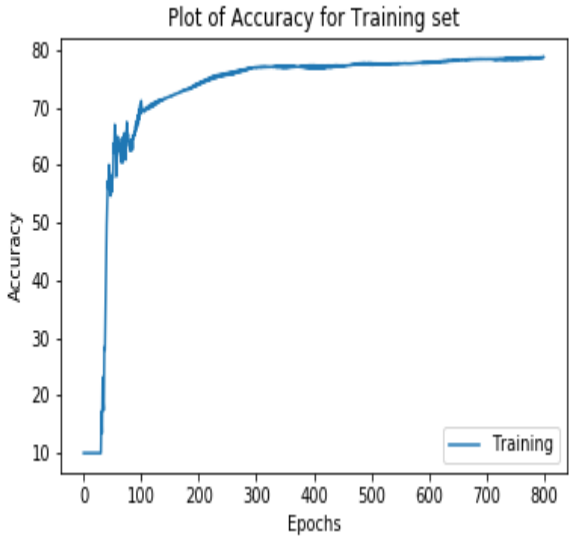


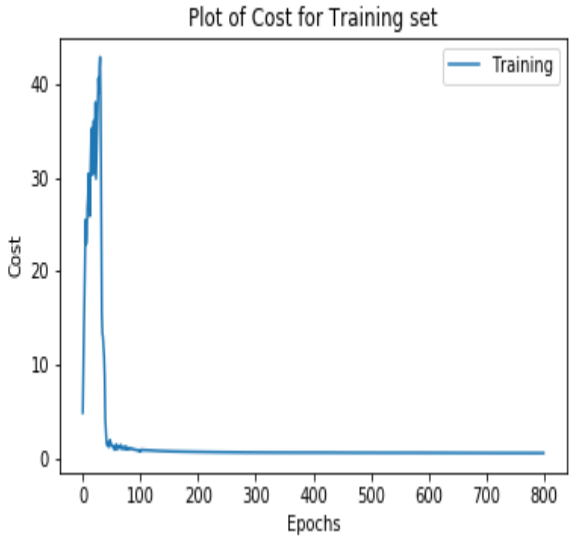


```
Neural Network with SGD accuracy for training set:  77.89%
Neural Network accuracy with SGD for Test set: 76.71%

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	925	10	45	11	0	5	0	2	0	0
Trouser	4	618	15	225	1	100	0	21	0	0
Pullover	22	9	822	71	0	49	0	3	0	0
Dress	2	109	35	717	3	126	0	8	0	0
Coat	0	1	5	0	770	1	102	18	102	0
Sandal	3	149	66	246	4	362	0	40	0	0
Shirt	0	0	0	0	27	0	890	1	82	0
Sneaker	2	13	15	6	19	20	7	913	3	0
Bag	0	1	3	0	15	0	52	2	927	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 10: learning rate 0.7 epoch 5=800 nodes = 128



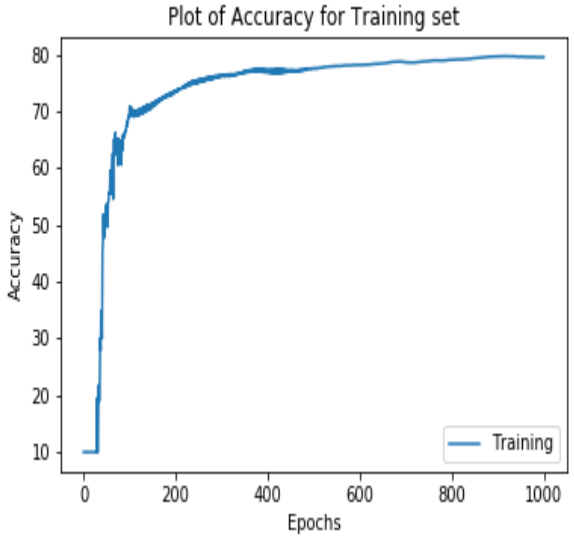


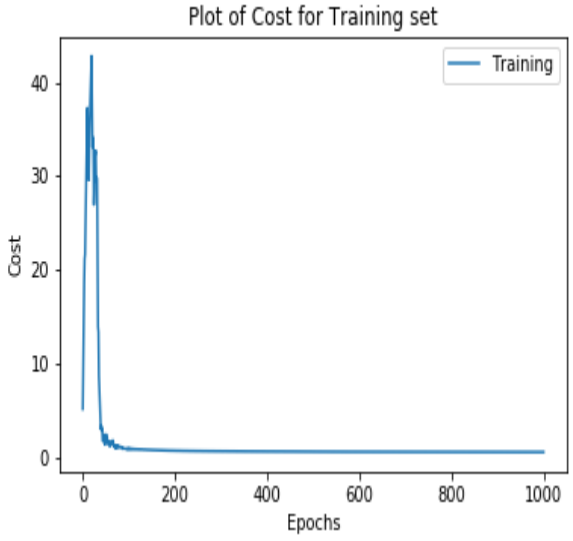
Neural Network with SGD accuracy for training set: 78.78%
Neural Network accuracy with SGD for Test set: 77.76%

Confusion Matrix:

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	919	16	53	5	0	4	0	2	0	0
Trouser	2	725	19	169	4	53	0	13	0	0
Pullover	9	13	877	40	0	24	0	4	0	0
Dress	1	172	75	666	1	79	0	6	0	0
Coat	0	0	8	2	787	1	78	18	105	0
Sandal	5	157	54	237	4	350	0	37	0	0
Shirt	0	0	0	0	31	0	866	3	100	0
Sneaker	1	17	22	7	19	23	5	901	3	0
Bag	0	0	1	0	16	2	46	3	932	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 11: lr 0.7 epoch 1000, nodes = 128



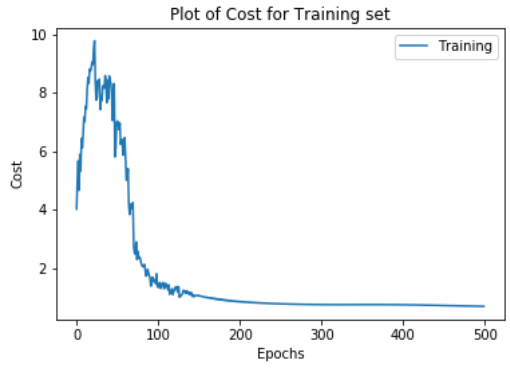


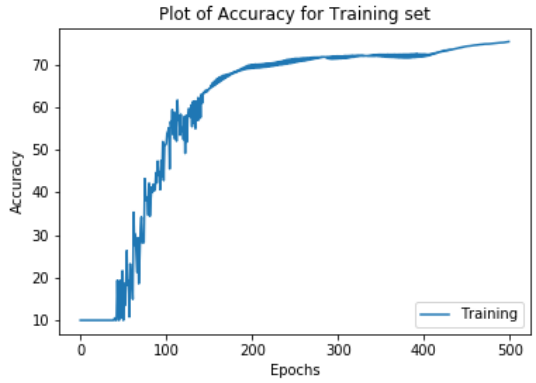
Neural Network with SGD accuracy for training set: 79.52%
Neural Network accuracy with SGD for Test set: 78.19%

Confusion Matrix:

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	919	12	40	8	0	9	0	2	0	0
Trouser	0	679	13	165	1	107	0	24	0	0
Pullover	11	13	847	56	0	29	0	4	0	0
Dress	1	146	68	670	1	100	0	12	0	0
Coat	1	0	5	1	785	0	76	27	104	0
Sandal	5	138	42	195	2	379	0	45	0	0
Shirt	0	0	0	0	35	0	886	1	78	0
Sneaker	1	10	15	8	8	16	5	933	1	0
Bag	2	0	1	0	25	1	38	3	930	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

case 12: lr 0.7 epoch 1000, nodes = 64





```
Neural Network with SGD accuracy for training set:  75.43%
Neural Network accuracy with SGD for Test set: 74.35%

Confusion Matrix:
```

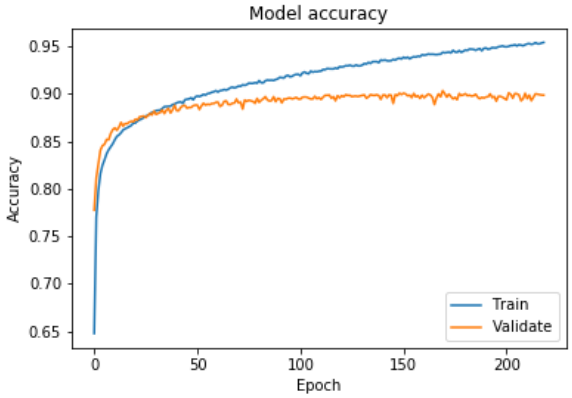
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	901	21	48	11	0	3	0	2	0	0
Trouser	0	613	10	206	2	133	0	21	0	0
Pullover	9	10	837	43	0	70	0	2	0	0
Dress	4	152	62	640	1	133	0	7	0	0
Coat	1	0	4	0	726	0	102	39	126	0
Sandal	2	159	64	262	1	287	0	42	0	0
Shirt	0	0	0	0	29	0	860	10	101	0
Sneaker	2	14	10	8	13	15	4	925	1	0
Bag	0	0	0	0	20	0	58	0	919	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

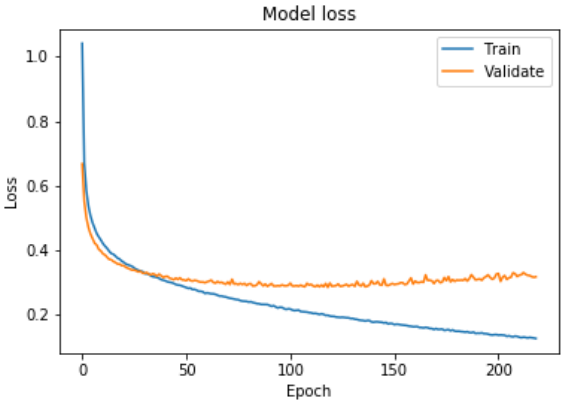
Multi Layer Neural Network

Input Dimension of first layer : 784 (number of features in each sample)

Case 1

Parameter	Value	Parameter	Value
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	SGD
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.02
Nodes 3	10		
Activation 3	softmax		





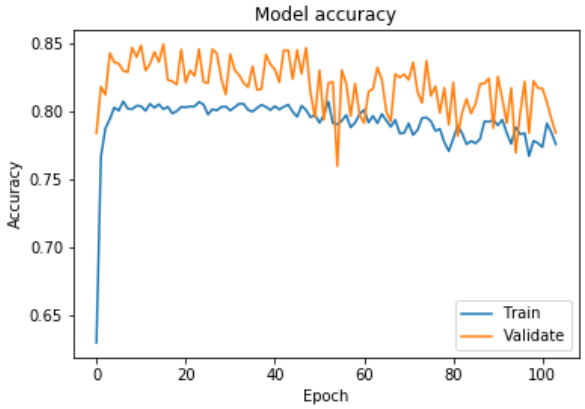

```
Neural Network accuracy for training set:  95.40079365079364
Neural Network accuracy for validation set:  89.85416666666667
Neural Network accuracy for Test set: 89.05

Confusion Matrix:
```

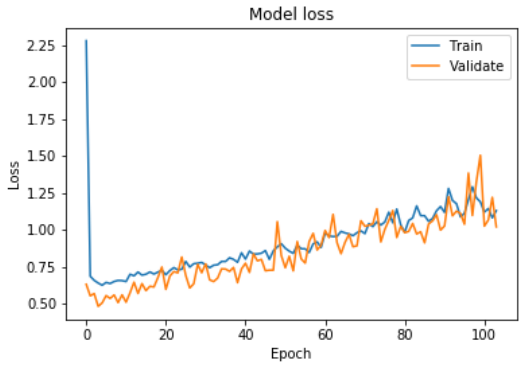
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	969	0	16	5	0	3	0	0	0	0
Trouser	2	816	12	78	1	77	0	0	0	0
Pullover	6	11	894	25	0	34	0	4	0	0
Dress	1	69	37	818	0	74	0	0	0	0
Coat	0	0	1	0	974	0	13	1	11	0
Sandal	2	65	22	55	0	761	0	6	0	0
Shirt	0	0	0	0	22	0	935	1	42	0
Sneaker	0	5	4	6	2	13	2	961	1	0
Bag	0	0	0	0	11	1	22	0	966	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 2

Parameter	**Value**	**Parameter**	**Value**
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	RMSprop
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.02

Nodes 3	10		
Activation 3	softmax		





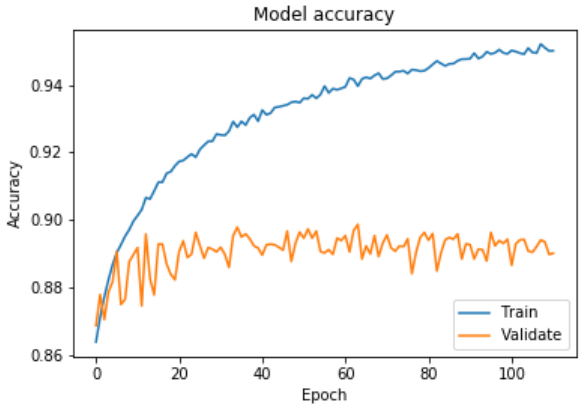

```
Neural Network accuracy for training set:  77.58531746031746
Neural Network accuracy for validation set:  78.41666666666667
Neural Network accuracy for Test set: 77.0

Confusion Matrix:
```

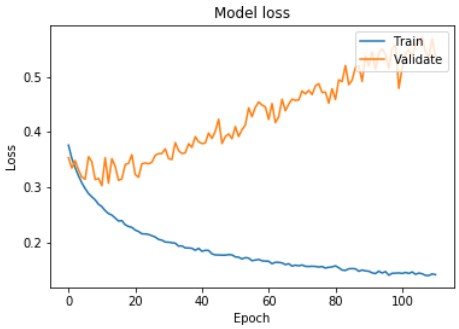
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	940	10	44	4	0	0	0	2	0	0
Trouser	2	684	80	185	2	31	1	5	2	0
Pullover	2	5	937	44	2	4	0	1	0	0
Dress	0	120	64	795	1	13	0	6	1	0
Coat	0	2	17	1	936	0	14	4	19	0
Sandal	1	157	348	140	1	256	0	20	2	0
Shirt	0	0	237	0	25	0	726	0	12	0
Sneaker	1	9	44	1	5	1	2	927	4	0
Bag	0	0	45	0	13	0	15	0	927	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 3

Parameter	Value	Parameter	Value
Nodes 1	128	Epochs	300

Activation 1	relu		
Dropout 1	0.1	Optimizer	RMSprop
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.001
Nodes 3	10		
Activation 3	softmax		





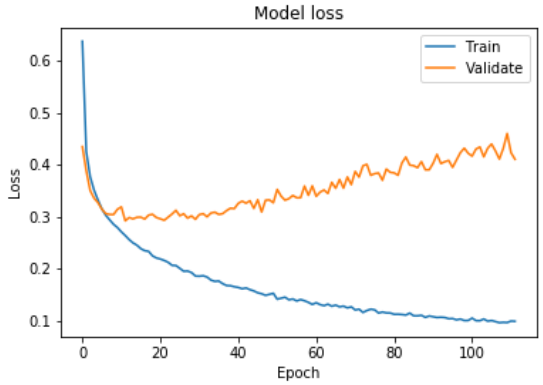

```
Neural Network accuracy for training set:  95.00992063113621
Neural Network accuracy for validation set:  89.02083333333334
Neural Network accuracy for Test set: 87.89

Confusion Matrix:
```

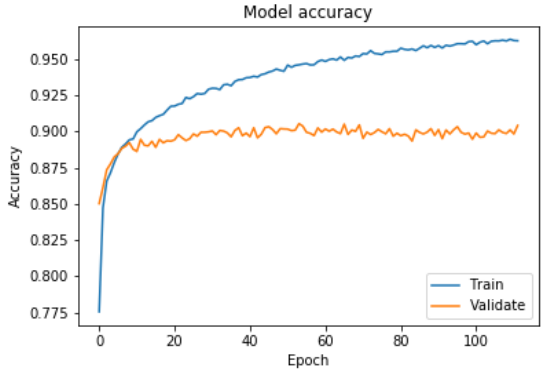
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	966	0	20	3	0	7	0	0	0	0
Trouser	2	714	11	180	1	75	1	2	0	0
Pullover	9	8	911	29	1	17	0	7	0	0
Dress	1	38	34	888	0	35	0	3	0	0
Coat	0	0	0	0	973	0	10	3	14	0
Sandal	0	62	31	135	0	671	0	12	0	0
Shirt	0	0	0	0	34	0	940	0	26	0
Sneaker	0	2	7	5	5	4	3	969	0	0
Bag	0	0	0	0	7	1	31	0	961	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 4

Parameter	Value	Parameter	Value

Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	Adam
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.001
Nodes 3	10		
Activation 3	softmax		



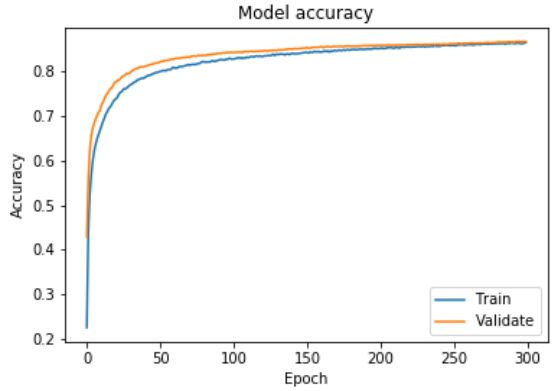


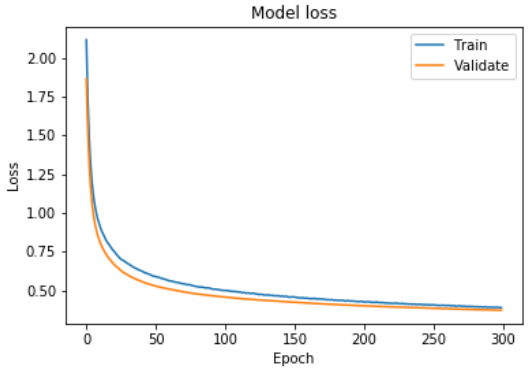

```
Neural Network accuracy for training set:  96.27380952380953
Neural Network accuracy for validation set:  90.41666666666667
Neural Network accuracy for Test set: 89.54

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	977	4	11	0	0	2	0	2	0	0
Trouser	1	838	11	66	0	66	0	1	1	0
Pullover	10	11	901	28	0	25	0	3	0	0
Dress	0	87	37	805	0	68	0	3	0	0
Coat	0	0	0	0	972	0	19	1	8	0
Sandal	0	78	29	53	0	741	0	8	0	0
Shirt	0	0	0	0	13	0	963	0	24	0
Sneaker	1	9	3	3	4	4	2	970	0	0
Bag	0	0	0	0	6	1	31	0	962	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 5

Parameter	Value	Parameter	Value
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	SGD
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.001
Nodes 3	10		
Activation 3	softmax		



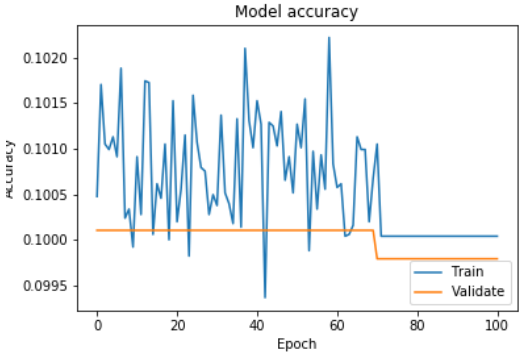


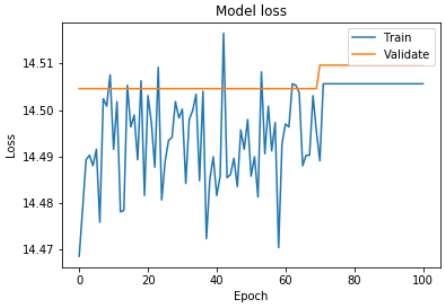

```
Neural Network accuracy for training set:  86.30555555177114
Neural Network accuracy for validation set:  86.5
Neural Network accuracy for Test set: 85.68

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	955	5	26	5	0	2	0	2	0	0
Trouser	2	771	9	121	1	71	0	6	0	0
Pullover	9	10	878	36	2	28	0	6	1	0
Dress	0	115	31	787	0	60	0	7	0	0
Coat	0	0	1	0	924	0	43	4	27	0
Sandal	1	116	38	92	0	583	0	19	0	0
Shirt	0	0	0	0	29	0	933	0	38	0
Sneaker	1	8	6	5	4	12	5	955	0	0
Bag	0	0	0	0	8	0	38	1	953	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 6

Parameter	Value	Parameter	Value
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	Adam
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	1
Nodes 3	10		
Activation 3	softmax		



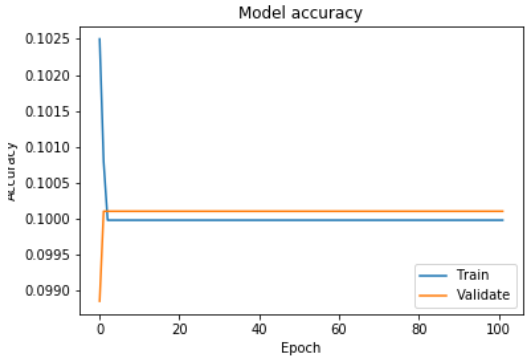


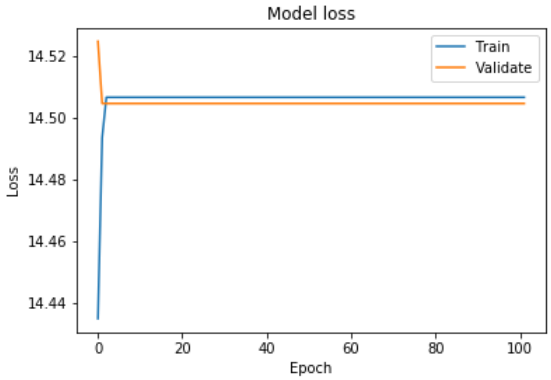

```
Neural Network accuracy for training set:  10.003968253495202
Neural Network accuracy for validation set:  9.979166666666666
Neural Network accuracy for Test set: 10.0

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	0	0	0	0	0	0	0	1000	0	0
Trouser	0	0	0	0	0	0	0	1000	0	0
Pullover	0	0	0	0	0	0	0	1000	0	0
Dress	0	0	0	0	0	0	0	1000	0	0
Coat	0	0	0	0	0	0	0	1000	0	0
Sandal	0	0	0	0	0	0	0	1000	0	0
Shirt	0	0	0	0	0	0	0	1000	0	0
Sneaker	0	0	0	0	0	0	0	1000	0	0
Bag	0	0	0	0	0	0	0	1000	0	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 7

Parameter	Value	Parameter	Value
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	Adam
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.5
Nodes 3	10		
Activation 3	softmax		



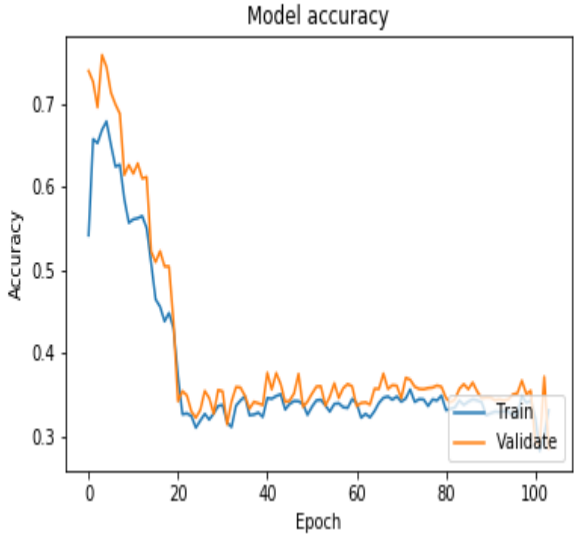


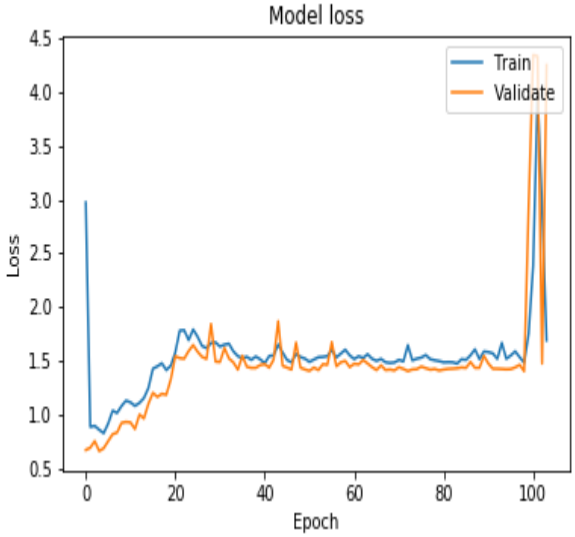

```
Neural Network accuracy for training set:  9.998015873488926
Neural Network accuracy for validation set:  10.010416666666666
Neural Network accuracy for Test set: 10.0

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	0	0	1000	0	0	0	0	0	0	0
Trouser	0	0	1000	0	0	0	0	0	0	0
Pullover	0	0	1000	0	0	0	0	0	0	0
Dress	0	0	1000	0	0	0	0	0	0	0
Coat	0	0	1000	0	0	0	0	0	0	0
Sandal	0	0	1000	0	0	0	0	0	0	0
Shirt	0	0	1000	0	0	0	0	0	0	0
Sneaker	0	0	1000	0	0	0	0	0	0	0
Bag	0	0	1000	0	0	0	0	0	0	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 8

Parameter	Value	Parameter	Value
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	Adam
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.05
Nodes 3	10		
Activation 3	softmax		



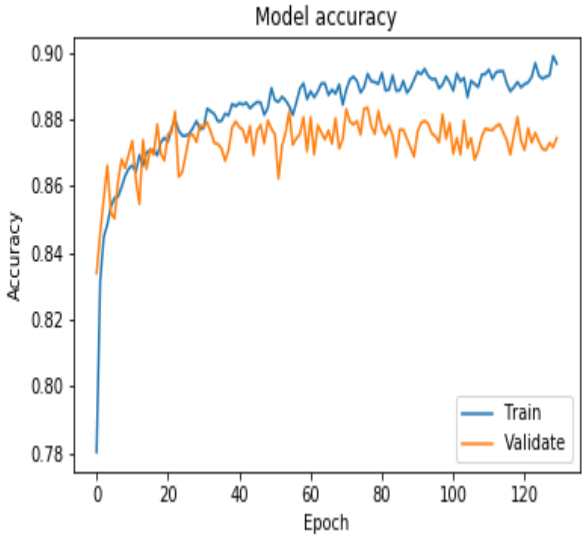


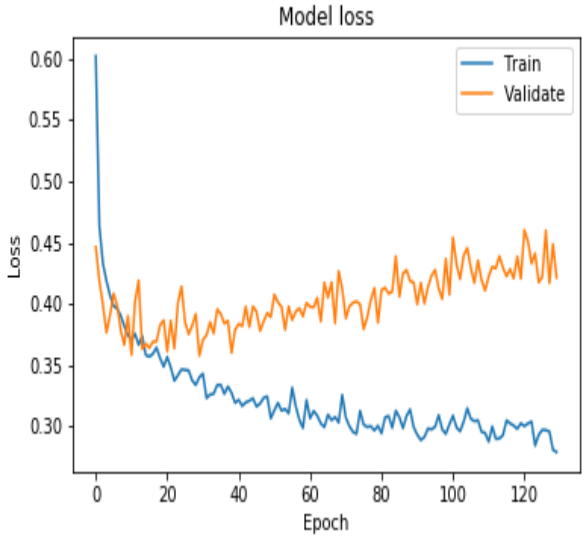
```
Neural Network accuracy for training set:  33.142857140964935
Neural Network accuracy for validation set:  28.572916666666664
Neural Network accuracy for Test set: 28.37

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	883	0	0	0	0	117	0	0	0	0
Trouser	0	0	0	0	0	1000	0	0	0	0
Pullover	3	0	0	0	0	997	0	0	0	0
Dress	1	0	0	0	0	999	0	0	0	0
Coat	0	0	0	0	953	47	0	0	0	0
Sandal	0	0	0	0	0	1000	0	0	0	0
Shirt	0	0	0	0	998	2	0	0	0	0
Sneaker	1	0	0	0	5	993	0	1	0	0
Bag	0	0	0	0	904	96	0	0	0	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 9

Parameter	Value	Parameter	Value
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	Adam
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.01
Nodes 3	10		
Activation 3	softmax		



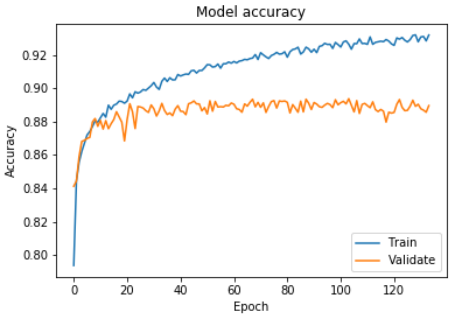


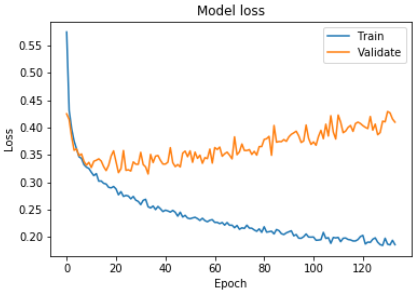
```
Neural Network accuracy for training set:  89.66071428949871
Neural Network accuracy for validation set:  87.42708333333333
Neural Network accuracy for Test set: 86.25

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	971	1	20	3	0	3	0	2	0	0
Trouser	1	676	15	193	0	102	1	2	0	0
Pullover	7	8	899	43	0	28	2	1	0	0
Dress	1	43	23	877	0	54	1	1	0	0
Coat	0	0	1	0	943	0	38	2	16	0
Sandal	1	61	46	132	0	632	2	6	0	0
Shirt	0	0	0	0	13	0	980	0	7	0
Sneaker	1	0	5	3	2	20	5	955	0	0
Bag	0	0	0	0	5	1	80	0	914	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 10

Parameter	Value	Parameter	Value
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	Adam
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.005
Nodes 3	10		
Activation 3	softmax		



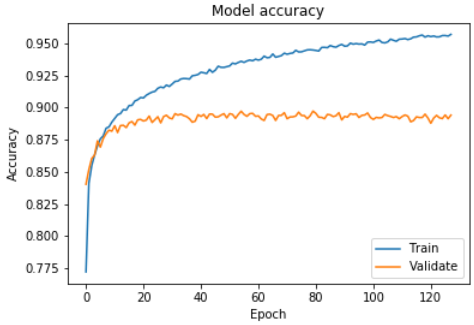


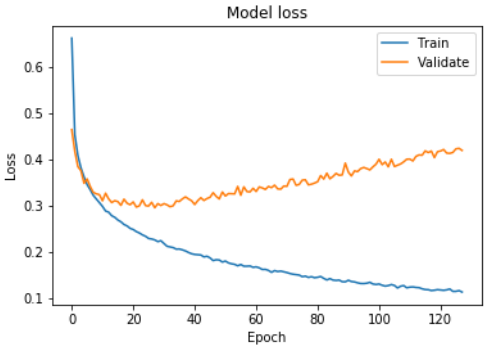
```
Neural Network accuracy for training set:  93.19246032124474
Neural Network accuracy for validation set:  88.95833333333333
Neural Network accuracy for Test set: 87.87

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	964	2	15	6	0	8	0	0	0	0
Trouser	1	733	10	152	0	86	0	1	0	0
Pullover	7	13	891	45	1	21	0	3	0	0
Dress	0	57	24	880	0	36	0	1	0	0
Coat	0	0	1	0	962	2	28	0	7	0
Sandal	0	71	32	97	0	658	0	5	0	0
Shirt	0	0	0	0	11	0	974	1	14	0
Sneaker	1	3	2	4	4	11	4	968	0	0
Bag	0	0	2	0	5	1	63	0	929	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 11

Parameter	Value	Parameter	Valu
Nodes 1	64	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	Adam
Nodes 2	10	η	0.001
Activation 2	softmax		



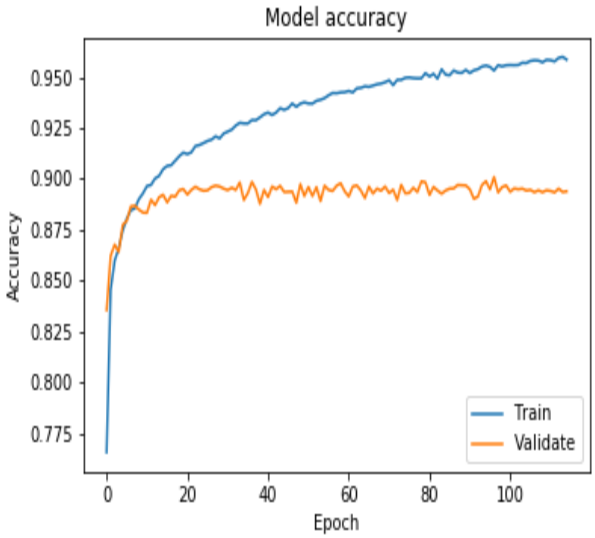


```
Neural Network accuracy for training set:  95.68055555933998
Neural Network accuracy for validation set:  89.40625
Neural Network accuracy for Test set: 88.31

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	972	5	16	2	1	2	0	1	0	0
Trouser	2	794	9	101	0	68	0	5	0	0
Pullover	8	15	875	43	0	35	0	4	0	0
Dress	1	70	26	825	0	73	0	5	0	0
Coat	0	0	0	0	972	0	12	5	11	0
Sandal	1	76	28	60	2	715	0	11	0	0
Shirt	0	0	0	0	33	0	936	2	29	0
Sneaker	0	6	5	4	4	8	2	967	0	0
Bag	0	0	1	0	13	1	36	0	949	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 12

Parameter	**Value**	**Parameter**	**Value**
Nodes 1	128	Epochs	300
Activation 1	relu		
Dropout 1	0.1	Optimizer	Adam
Nodes 2	64		
Activation 2	relu		
Dropout 2	0.1	η	0.001
Nodes 3	64		
Activation 3	relu		
Dropout 3	0.1		
Nodes 4	10		
Activation 4	softmax		




```
Neural Network accuracy for training set:  95.86309524187966
Neural Network accuracy for validation set:  89.38541666666666
Neural Network accuracy for Test set: 89.33

Confusion Matrix:
```

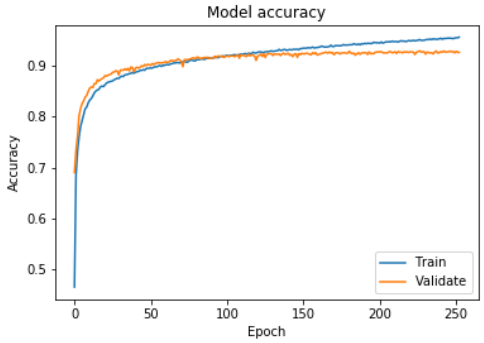
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	972	0	18	2	0	2	0	1	0	0
Trouser	1	855	11	68	1	47	0	0	0	0
Pullover	7	12	898	29	1	16	0	4	0	0
Dress	1	110	32	815	0	37	0	2	0	0
Coat	0	0	1	0	974	0	14	1	10	0
Sandal	0	94	26	73	1	659	0	4	0	0
Shirt	0	0	0	0	10	0	966	0	24	0
Sneaker	0	5	4	4	4	5	5	962	1	0
Bag	0	0	0	0	14	1	35	0	950	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

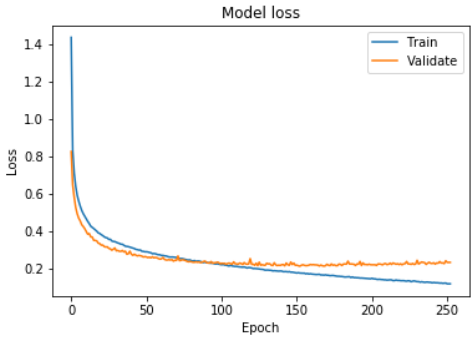
CNN

Input Dimension of first layer : 784 (number of features in each sample)

Case 1

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	SGD
Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.02
Dropout	0.25		
Conv2D	128		
Dense	128		
Dropout	0.5		
Dense	10		
Activation 3	softmax		



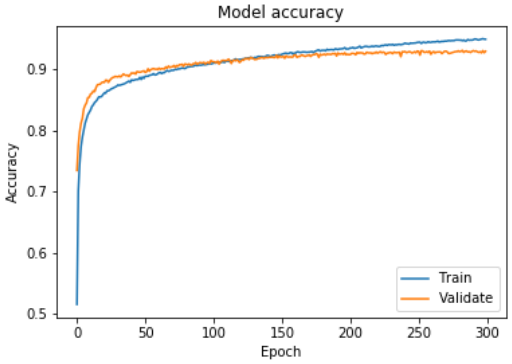


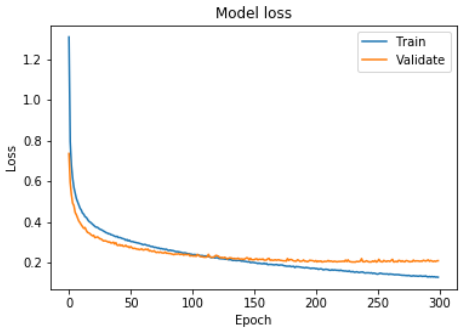
```
Convulational Neural Network accuracy for training set:  95.5932539720384
Convulational Neural Network accuracy for validation set:  92.61458333333333
Convulational Neural Network accuracy for Test set: 92.41

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	984	1	8	2	0	1	0	2	0	0
Trouser	1	867	6	46	0	65	0	1	0	0
Pullover	5	11	917	28	0	29	0	1	0	0
Dress	1	22	18	895	0	62	0	1	0	0
Coat	0	0	0	0	990	0	7	0	3	0
Sandal	0	42	23	51	0	809	0	7	0	0
Shirt	0	0	0	0	10	0	978	0	12	0
Sneaker	1	0	3	1	1	1	4	987	1	0
Bag	0	0	0	0	6	1	34	0	959	0

Case 1

Parameter	**Value**	**Parameter**	**Value**
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	SGD
Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.02
Dropout	0.25		
Dense	128		
Dropout	0.5		
Dense	10		
Activation 3	softmax		





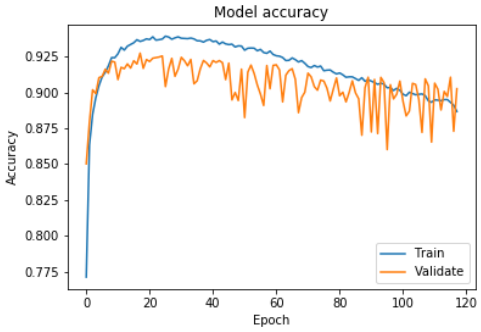

```
Convulational Neural Network accuracy for training set:  94.95833333711775
Convulational Neural Network accuracy for validation set:  92.98958333333334
Convulational Neural Network accuracy for Test set: 92.28

Confusion Matrix:
```

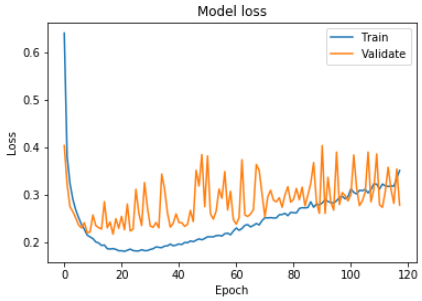
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	983	0	11	2	0	3	0	1	0	0
Trouser	1	916	8	25	0	34	0	1	0	0
Pullover	2	10	936	19	0	20	0	4	0	0
Dress	1	63	20	862	0	53	0	1	0	0
Coat	0	0	0	0	985	0	8	0	7	0
Sandal	0	65	31	55	0	737	0	9	0	0
Shirt	0	0	0	0	9	0	977	0	14	0
Sneaker	1	4	4	1	1	0	2	983	1	0
Bag	0	1	0	0	4	0	20	0	975	0

Case 1

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	RMSprop
Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.001
Dropout	0.25		
Conv2D	128		
Dense	128		
Dropout	0.5		

Dense	10		
Activation 3	softmax		





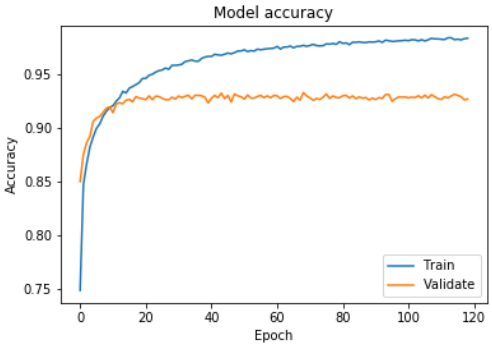

```
Convulational Neural Network accuracy for training set:  88.6686507974352
Convulational Neural Network accuracy for validation set:  90.25
Convulational Neural Network accuracy for Test set: 89.74

Confusion Matrix:
```

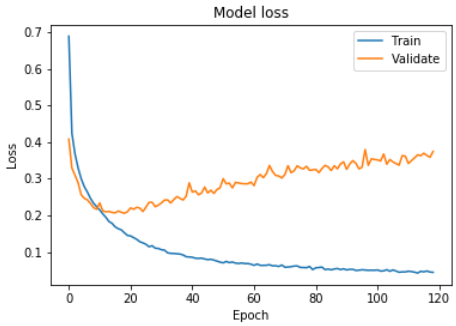
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	968	0	18	5	0	8	0	0	0	0
Trouser	1	864	5	68	0	51	0	0	0	0
Pullover	0	18	896	44	0	23	0	0	0	0
Dress	1	71	16	862	0	48	0	2	0	0
Coat	0	0	0	0	965	0	18	0	17	0
Sandal	0	75	30	86	0	675	0	7	0	0
Shirt	0	0	0	0	2	0	951	0	47	0
Sneaker	1	5	1	5	2	8	4	971	1	0
Bag	1	0	0	0	3	0	22	0	974	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 1

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	Adam
Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.001
Dropout	0.25		

Conv2D	128		
Dense	128		
Dropout	0.5		
Dense	10		
Activation 3	softmax		





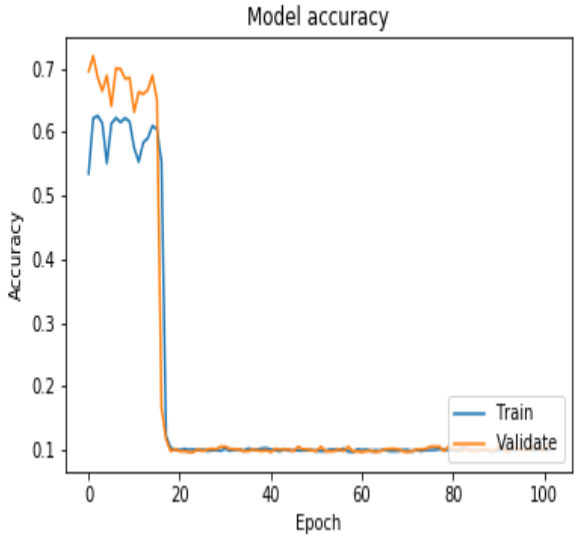

```
Convulational Neural Network accuracy for training set:  98.3908730120886
Convulational Neural Network accuracy for validation set:  92.70833333333334
Convulational Neural Network accuracy for Test set: 92.54

Confusion Matrix:
```

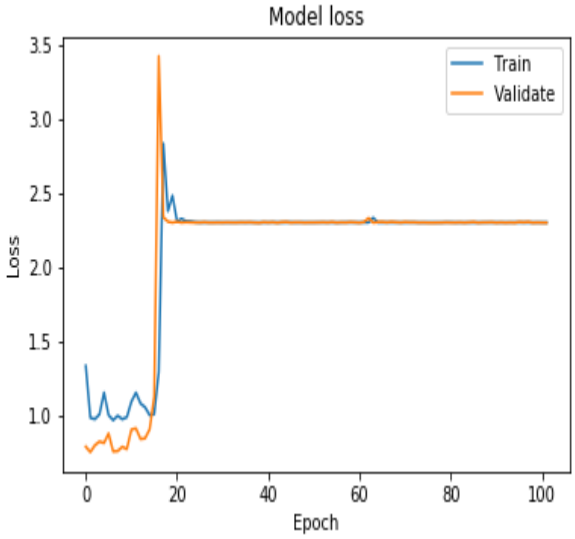
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	983	0	12	2	0	1	0	2	0	0
Trouser	0	891	8	47	0	34	0	0	0	0
Pullover	1	13	937	19	0	18	0	0	0	0
Dress	1	32	23	896	0	47	0	0	0	0
Coat	0	0	0	0	987	0	9	0	4	0
Sandal	0	61	20	74	0	762	0	5	0	0
Shirt	0	0	0	0	5	0	986	0	9	0
Sneaker	0	0	1	4	1	0	3	985	1	0
Bag	0	1	0	0	3	0	35	0	961	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 1

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	Adam

Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.05
Dropout	0.25		
Conv2D	128		
Dense	128		
Dropout	0.5		
Dense	10		
Activation 3	softmax		



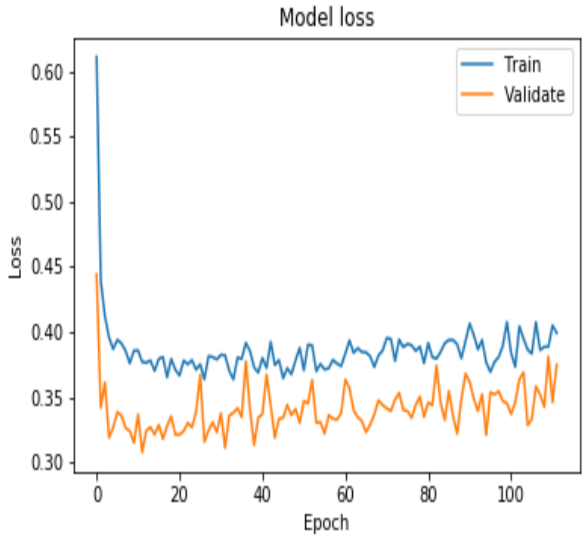


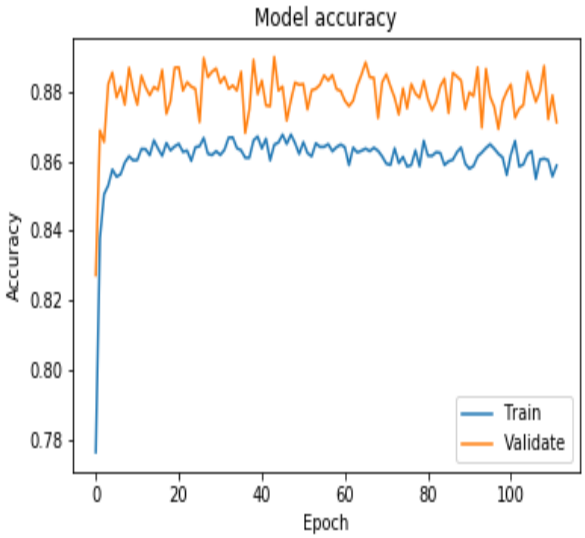
```
Convulational Neural Network accuracy for training set:  9.988095237622186
Convulational Neural Network accuracy for validation set:  10.510416666666666
Convulational Neural Network accuracy for Test set: 10.0

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	0	0	0	0	0	1000	0	0	0	0
Trouser	0	0	0	0	0	1000	0	0	0	0
Pullover	0	0	0	0	0	1000	0	0	0	0
Dress	0	0	0	0	0	1000	0	0	0	0
Coat	0	0	0	0	0	1000	0	0	0	0
Sandal	0	0	0	0	0	1000	0	0	0	0
Shirt	0	0	0	0	0	1000	0	0	0	0
Sneaker	0	0	0	0	0	1000	0	0	0	0
Bag	0	0	0	0	0	1000	0	0	0	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 1

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	Adam
Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.01
Dropout	0.25		
Conv2D	128		
Dense	128		
Dropout	0.5		
Dense	10		
Activation 3	softmax		





Convulational Neural Network accuracy for training set: 85.87301587301587
Convulational Neural Network accuracy for validation set: 87.125
Convulational Neural Network accuracy for Test set: 85.78

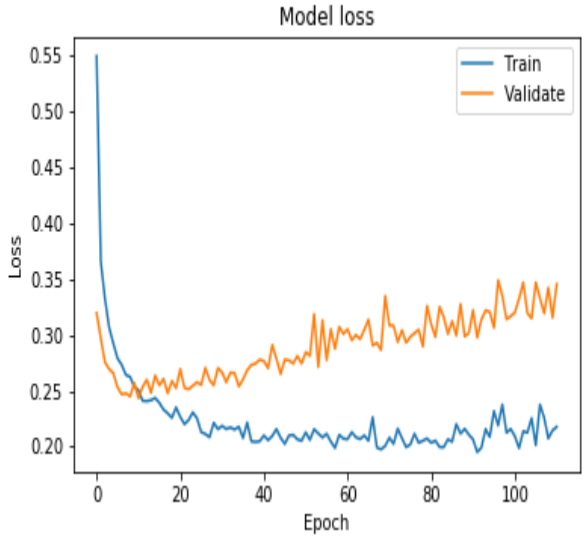
Confusion Matrix:

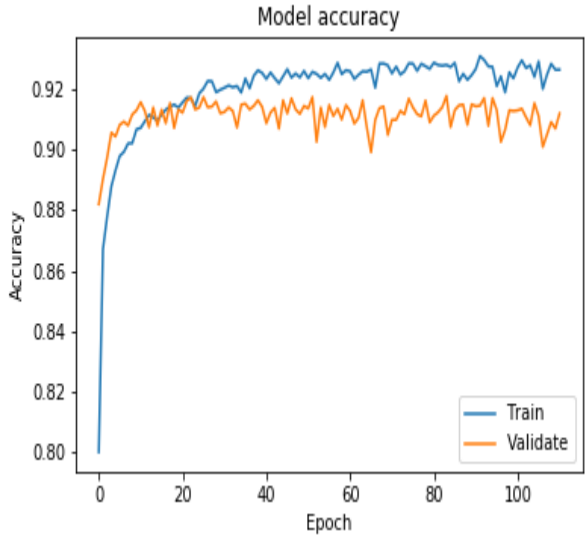
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	972	0	19	3	0	5	0	0	0	0
Trouser	0	723	10	167	0	77	0	1	0	0
Pullover	2	9	865	62	0	38	0	2	0	0
Dress	0	84	17	842	0	56	0	1	0	0
Coat	0	1	0	0	929	0	53	1	16	0
Sandal	0	96	29	133	0	551	0	6	0	0
Shirt	0	0	0	0	5	0	977	0	18	0
Sneaker	0	2	3	6	2	30	5	938	1	0
Bag	0	0	0	0	3	2	64	0	931	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 1

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	Adam

Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.005
Dropout	0.25		
Conv2D	128		
Dense	128		
Dropout	0.5		
Dense	10		
Activation 3	softmax		



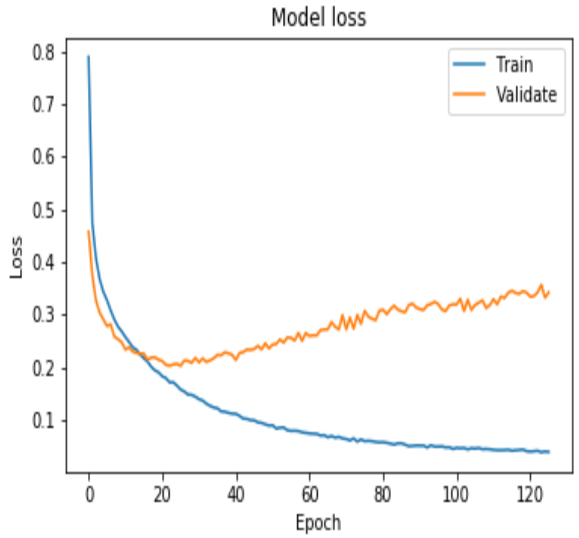


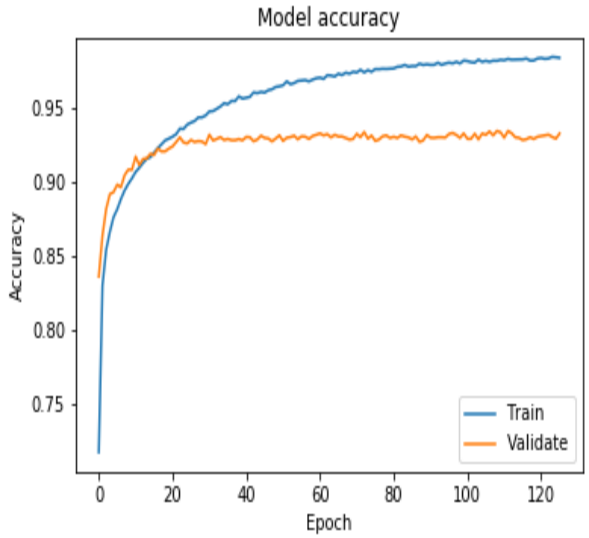
```
Convulational Neural Network accuracy for training set:  92.64880952759394
Convulational Neural Network accuracy for validation set:  91.19791666666667
Convulational Neural Network accuracy for Test set: 90.21

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	981	1	8	2	0	7	0	1	0	0
Trouser	1	854	6	41	0	85	0	0	0	0
Pullover	4	7	917	20	0	37	0	0	0	0
Dress	0	74	35	835	0	54	0	1	0	0
Coat	0	0	0	0	967	0	26	0	7	0
Sandal	1	49	23	74	0	722	0	5	0	0
Shirt	0	0	0	0	1	0	990	0	9	0
Sneaker	1	0	3	4	1	9	2	977	0	0
Bag	0	0	0	0	5	0	48	0	946	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case 1

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	Adam
Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.0005
Dropout	0.25		
Conv2D	128		
Dense	128		
Dropout	0.5		
Dense	10		
Activation 3	softmax		





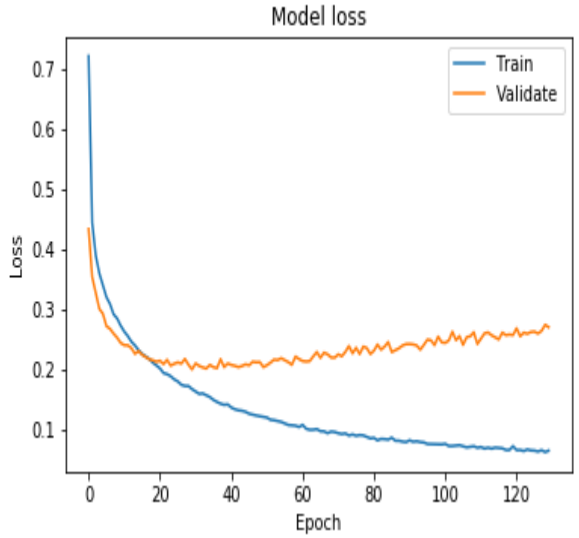

```
Convulational Neural Network accuracy for training set:  98.44642856764415
Convulational Neural Network accuracy for validation set:  93.30208333333333
Convulational Neural Network accuracy for Test set: 92.81

Confusion Matrix:
```

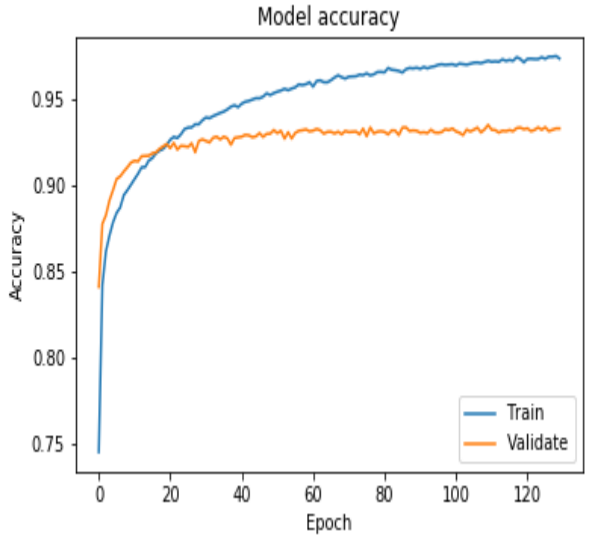
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	983	1	11	4	0	1	0	0	0	0
Trouser	0	890	7	38	0	49	0	0	0	0
Pullover	3	9	925	23	0	28	0	1	0	0
Dress	1	21	25	897	0	55	0	1	0	0
Coat	0	0	0	0	984	0	11	1	4	0
Sandal	0	47	22	49	0	802	0	3	0	0
Shirt	0	0	0	0	3	0	984	0	13	0
Sneaker	1	0	3	1	0	2	2	990	1	0
Bag	0	0	0	0	6	1	31	0	962	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	Adam
Conv2D	64		
Activation 2	relu		

MaxPooling2D	2	η	0.0005
Dropout	0.25		
Dense	128		
Dropout	0.5		
Dense	10		
Activation 3	softmax		



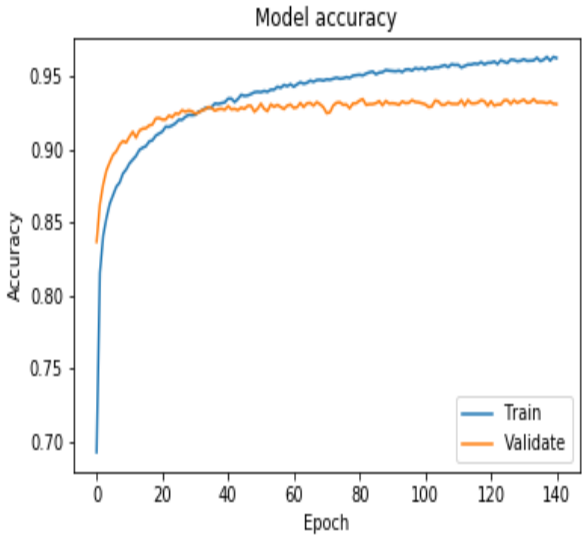


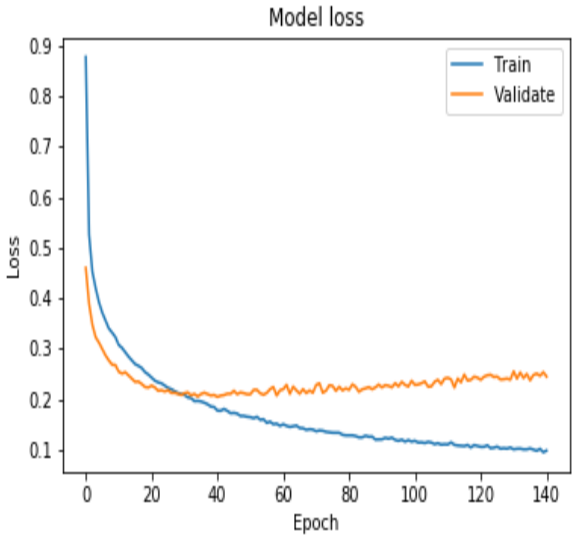
```
Convulational Neural Network accuracy for training set:  97.42460317460318
Convulational Neural Network accuracy for validation set:  93.34375
Convulational Neural Network accuracy for Test set: 93.02

Confusion Matrix:
```

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	983	0	11	1	0	1	0	2	0	0
Trouser	1	911	6	24	0	35	0	2	0	0
Pullover	1	9	938	13	0	19	0	1	1	0
Dress	0	53	25	874	0	48	0	0	0	0
Coat	0	0	0	0	987	0	10	0	3	0
Sandal	0	51	21	45	0	770	1	10	0	0
Shirt	0	0	0	0	2	0	975	0	23	0
Sneaker	2	0	1	1	2	1	2	989	0	0
Bag	0	0	0	0	6	0	24	0	970	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case

Parameter	**Value**	**Parameter**	**Value**
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	Adam
Conv2D	64		
Activation 2	relu		
MaxPooling2D	2	η	0.0005
Dropout	0.25		
Dense	64		
Dropout	0.5		
Dense	10		
Activation 3	softmax		





Convulational Neural Network accuracy for training set: 96.24206349584792
Convulational Neural Network accuracy for validation set: 93.09375
Convulational Neural Network accuracy for Test set: 92.61

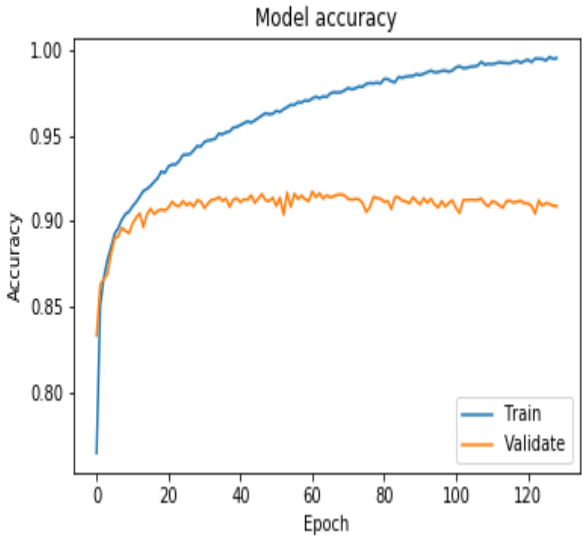
Confusion Matrix:

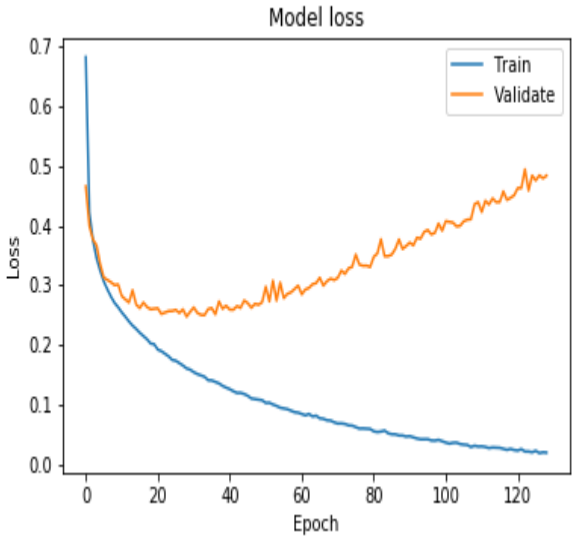
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	985	0	12	1	0	1	0	1	0	0
Trouser	1	891	5	40	0	46	0	0	0	0
Pullover	1	11	914	32	0	26	0	2	0	0
Dress	0	33	16	909	0	40	0	0	0	0
Coat	0	0	0	0	986	0	9	0	5	0
Sandal	0	43	17	70	0	758	0	10	0	0
Shirt	0	0	0	0	6	0	983	0	11	0
Sneaker	0	1	3	1	3	0	2	987	0	0
Bag	0	0	0	0	4	1	27	0	968	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

Case

Parameter	Value	Parameter	Value
Conv2D	32	Epochs	300
Activation 1	relu		
MaxPooling2D	2	Optimizer	Adam
Conv2D	64		

Activation 2	relu		
MaxPooling2D	2	η	0.0005
Dense	10		
Activation 3	softmax		





Convulational Neural Network accuracy for training set: 99.51388888888889
Convulational Neural Network accuracy for validation set: 90.875
Convulational Neural Network accuracy for Test set: 90.66

Confusion Matrix:

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle-Boot
T-shirt	982	1	10	2	0	3	0	0	0	0
Trouser	1	870	8	46	0	58	0	2	0	0
Pullover	6	17	882	28	0	23	0	1	0	0
Dress	0	70	24	866	0	31	0	2	0	0
Coat	0	0	0	0	975	0	15	0	10	0
Sandal	0	65	26	63	0	723	0	6	0	0
Shirt	0	0	0	0	9	0	970	1	20	0
Sneaker	1	2	4	1	2	1	4	982	0	0
Bag	0	0	0	0	6	1	19	0	974	0
Ankle-Boot	0	0	0	0	0	0	0	0	0	0

EVALUATIONS

Observations

- The confusion matrices are created for all the classifiers and it tells the number of times that classifier predicts the correct class and the number of times it predicts other classes.
- Comparing the confusion matrices we can see that the model is better when the numbers are higher on the diagonal. The value on the diagonal tells us the number of times the model predicted the correct class, and the numbers in the other columns tells us how many times the model predicted other class instead of the correct one.
- Hence, the higher the values in the diagonal, more is the accuracy of our model
- It can be seen that the best results are obtained using **Convolutional Neural Network,** followed by **Multilayer Neural Network** and then **Single layer implementation of Neural Network using SGD.**

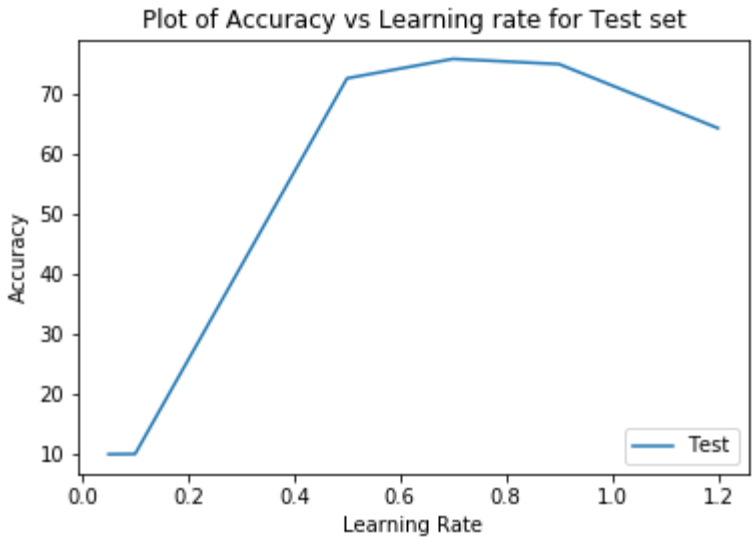
The optimum accuracies achieved for the classifiers are as follows:

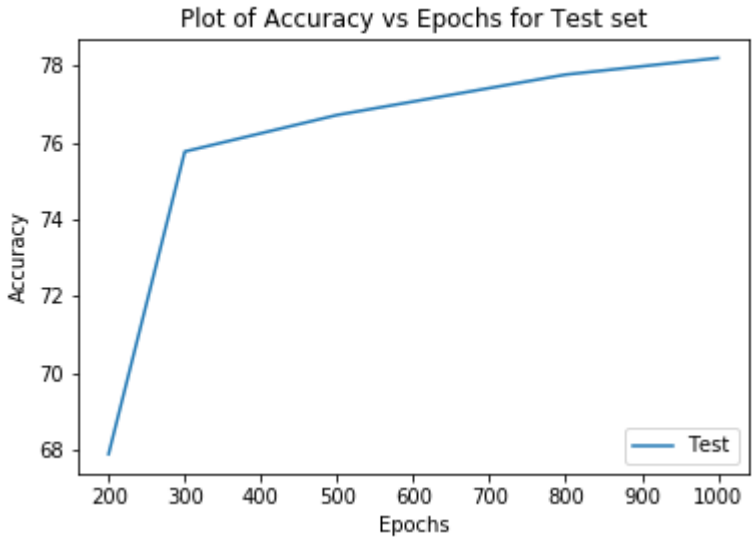
Classifier	MNIST Accuracy (%)
Neural Network using SGD	78.19
Multilayer Neural Network	89.54
CNN	93.02

Individual Classifier Observations

<u>Single Hidden Layer Neural Network using SGD</u>

1. It is observed that when we increase the learning rate η, the accuracy also increases. We vary the learning rate from 0.05 to 1.2 and it is observed that after a learning rate of 0.9, the accuracy tends to decrease.
2. The best accuracy is achieved at **0.7**
3. Also, the number of epochs are varied from 200 to 1000 and it is observed that as the number of epochs are increased, the accuracy is increased.
4. The best accuracy is achieved for **1000** epochs
5. The number of nodes in the hidden layer are also varied and it is observed that increasing the number of nodes increases the accuracy. The best accuracy is achieved for the number of hidden nodes **128**.
6. Thus, for this classifier, the best accuracy achieved is where η is 0.7, epochs are 1000 and nodes are 128.
7. As, can be seen from the confusion matrix many classes have been correctly classified but some have been not.
8. These observations can also be seen from below graphs.

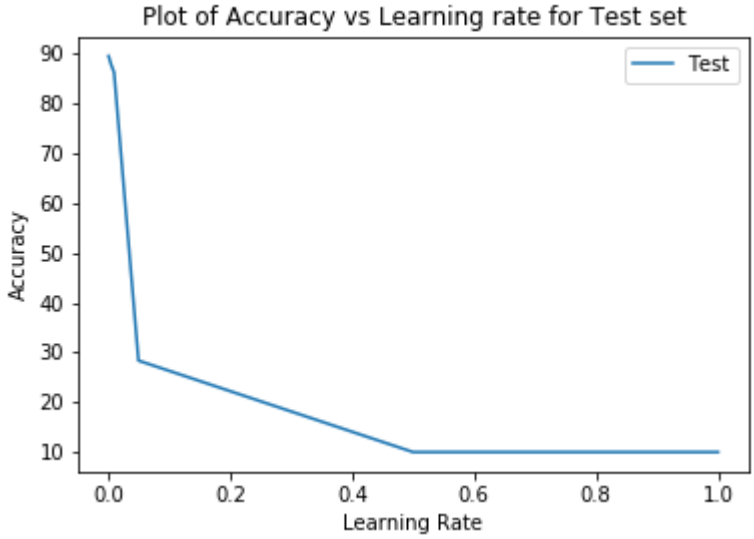




Multilayer Neural Network

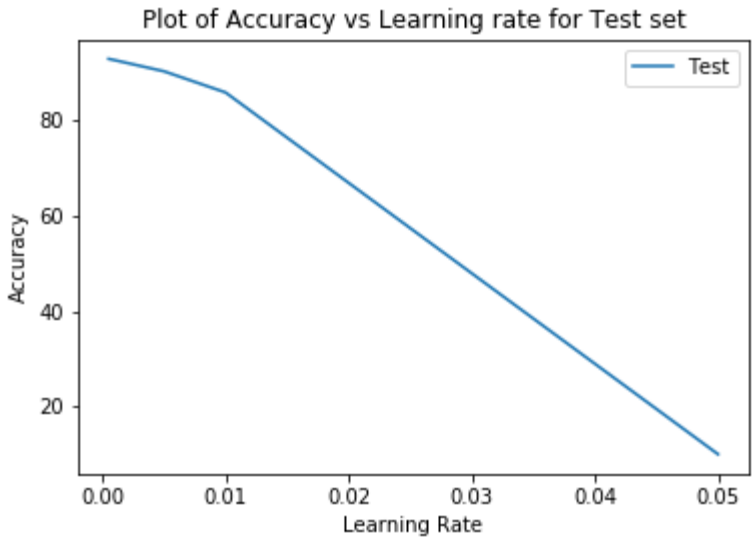
1. The optimizer functions are varied.
2. SGD gives a better accuracy as compared to RMSProp optimiser function
3. Adam is the more efficient choice out of all the other optimers. Provides better accuracy as compared to RMSProp and SGD for comparable epochs and learning rate.
4. Also, increasing the number of layers present increases performance of our model but optimum accuracy is obtained for two layer network
5. The learning rate is varied from 0.001 to 1.
6. Learning rate =0.001 and Optimiser = Adam are the most optimal combination.
7. Best observed case is Case 4 with 3 nodes and 3 layers. Hence, the number of layers and nodes in each layer affect the performance as well
8. Dropout also plays a role in better accuracy.
9. The accuracies increase slightly with variations to learning rate and epochs, however a radically steep learning rate of 1 would rapidly decrease the accuracy even with the optimum number of epochs or optimiser.

10. Varying the number of nodes and layers can also decrease the performance as is visible in case 8-9
11. The effect of learning rate on accuracy can be observe from the graphs given below.



Convolutional Neural Network

1. It is seen that the activation of the input and the hidden layers as ‘relu’ gives better accuracy.
2. Increasing the number of convolution layers and pooling layer also tends to increase the accuracy but taking more than 2 convolution layers tended to decrease the performance.
3. The best performance is achieved when **2 convolution layers** are taken with max pooling layer in between and 2 dense layers, out of which 1 being the output layer.
4. Incorporating dropout between the layers also increased the accuracy which is correct since it tends to prevent over-fitting.
5. The learning rate is varied from 0.01 to 0.0005 and it is observed that for lower learning rates the accuracy tends to increase and the optimum accuracy is obtained for a value of **0.0005**.
6. The optimizer is also varied and the network is tested with SGD, RMSprop, ADAM and Adadelta optimizers. RMSprop and Adadelta don’t tend to give good performance. The best performance is given by **Adam** followed by SGD.
7. The number of nodes are also varied the best accuracy obtained.
8. Hence, the best accuracy is obtained for **learning rate of 0.0005,** optimizer **Adam,** epochs 300 and for **2 convolution layer** and max pooling layers.
9. It can be seen from the confusion matrix that more numbers are present on the diagonal and hence more accuracy.
10. The trend of learning rate with accuracy can be seen from below.



Comparing the confusion matrices of all three networks it can be seen that the numbers are more saturated on the diagonal for the CNN and thus it tends to give better accuracy than the other two.

Acknowledgments

Thankful to Professor Srihari and the group of Teaching Assistants who have helped us ace the projects.

References

https://keras.io/examples/mnist_cnn/
https://keras.io/layers/convolutional/
https://keras.io/optimizers/
https://stackoverflow.com/questions/43237124/what-is-the-role-of-flatten-in-keras/43237727#43237727

https://www.ics.uci.edu/~pjsadows/notes.pdf

https://stackabuse.com/creating-a-neural-network-from-scratch-in-python/

https://towardsdatascience.com/neural-networks-from-scratch-easy-vs-hard-b26ddc2e89c7