**Question1: Using default network.**

Plot the train images and its corresponding labels. From the below labels we are not able to figure out the classes to which a particular group of images belongs.

The dataset consists of 70000 grayscale images of 28x28 pixels each, representing 5 mystery classes. The images are split into 60000 training samples and 10000 test samples. Visualizing the input samples and their corresponding pixels.

A picture containing text

Description automatically generated A picture containing text, red, light, brick

Description automatically generated

**Plotting the data distribution**

Background pattern

Description automatically generatedChart, bar chart

Description automatically generated

Distribution of the class shows that for 3 classes we have equal amount of data. But for class 1 and class 4, the dataset is not balanced properly.

We also tested for outliers’ detection following which we got that there are few outliers. These outliers may be because of the mystery classes. Following are the results after removing outliers. This is just for analysis purpose, for training the CNN we will be using the entire dataset provided.

Shape of x\_train data after outlier removal: (24553, 784)

Shape of y\_train data after outlier removal: (24553,)

**Plotting the default CNN Model**: Test accuracy: 86.15000247955322

Plotting the training loss, validation loss, training accuracy and validation accuracy for default CNN. Epoch: 10, LR: 0.01, Loss: Categorical Cross entropy

Chart, line chart

Description automatically generated Diagram

Description automatically generated

The output shows the training and validation accuracy and loss for each epoch during the training of a neural network model. The model was trained for 10 epochs and achieved an accuracy of 86.15% on the test set.

It seems that the model is learning, as the training and validation accuracies increase with each epoch, while the loss decreases. However, it appears that the model may be overfitting to the training data, as the validation accuracy starts to plateau after the 6th epoch, while the training accuracy continues to increase. This suggests that the model may not generalize well to new data.

To address this issue, techniques such as regularization, dropout, or early stopping could be used to prevent overfitting and improve the model's performance on new data.

**Runtime Performance**

Table

Description automatically generated

**Question2: modifying the model.**

To do this we have considered few models. Out first approach is to not to change the model, instead modify the hyperparameters such as learning rate. Reducing the learning rate from 0.01 to 0.001 which resulted in higher test accuracy. Reducing LR increased the computation.

Model 1: Using Learning Rate: 0.001,Epoch: 10, Test accuracy: 92.40000247955322

Chart, line chart

Description automatically generated

Table

Description automatically generatedDiagram, table

Description automatically generated

The above model is a neural network that has been trained for 10 epochs on a certain dataset, with a learning rate of 0.001. Based on the output, the model seems to be performing fairly well on the given task, achieving an accuracy of 92.47% on the validation set and 91.87% on the test set.

During training, the model was able to steadily improve its accuracy on the validation set from 89.35% in the first epoch to 92.47% in the tenth epoch. This suggests that the model was able to effectively learn from the training data and generalize well to unseen data.

**Modified Model 2**: Test accuracy: 92.75000095367432

Learning Rate: 0.001, Epoch:10, Optimizer: SGD, Loss: Categorical cross entropy

Chart, line chart

Description automatically generatedTable

Description automatically generatedDiagram

Description automatically generated

Based on the given output, the model seems to perform well with an increasing accuracy over the training epochs. The training accuracy is at 96.90% while the validation accuracy is at 93.12%, indicating that the model generalizes well to unseen data.

The model also seems to avoid overfitting, as the validation loss remains relatively consistent throughout the training process. The test accuracy of 92.75% is also quite good, indicating that the model performs well on new and unseen data.

Overall, this model seems to be performing well and could be considered a good model for the given task.

**Chart, line chart

Description automatically generatedModel3: Using 16 layer CNN** : Test accuracy: 89.48000073432922

Table

Description automatically generatedTable

Description automatically generated

The two models (Model2 and Model3) provided have significantly different architectures and training histories.

Model 2 is a CNN model that has 10 epochs of training with a decreasing learning rate. It achieves a test accuracy of 92.75%. Model 3 is also a CNN model with 10 epochs of training but with a different architecture, achieving a test accuracy of 89.11%.

In comparing the two models, we can see that Model 2 has a higher test accuracy than Model 3. It is also noteworthy that the validation accuracy of Model 2 is consistently higher than that of Model 3, indicating that it is more effective in preventing overfitting.

Additionally, the training times for each epoch of Model 2 is much faster than that of Model 3, suggesting that it may be a more efficient model to use for larger datasets.

Overall, we can conclude that Model 2 is the better model of the two due to its higher accuracy, faster training time, and better prevention of overfitting.

**Q3: Analysis on Question2 model1.**

Plotting the Class Activation Map for all the classes along with their SoftMax probabilities:

A picture containing text

Description automatically generated

**Class Activation Map of CNN**

Class activation maps (CAMs) are used in convolutional neural networks (CNNs) to visualize the regions of an input image that are important for a particular classification decision.

To generate a CAM, we first obtain the feature maps from the last convolutional layer of the CNN. These feature maps represent the learned features of the input image. We then compute the class activation map by taking a weighted sum of these feature maps, where the weights are the learned coefficients of the last fully connected layer of the CNN. The resulting activation map highlights the regions of the input image that are most important for the predicted class.

From the plot, we have a CNN that has been trained to classify images into 5 classes: t-shirt, sandal, trouser, bag, and coat. Let's assume that the input image has been classified as t-shirts with a probability of 0.99. We can generate a CAM for this class by taking the weighted sum of the feature maps, where the weights are the coefficients of the last fully connected layer that correspond to the t-shirts class.

The resulting CAM will highlight the regions of the input image that were most important for predicting the t-shirts class. This can give us insight into what features the CNN is looking for when it makes its predictions. Similarly, we can generate CAMs for the other classes to see what regions of the input image are important for each class.

For example, if we generate a CAM for the trouser class, which was predicted with a probability of 1.00, the resulting activation map will highlight the regions of the input image that were most important for predicting the trouser class. We can repeat this process for each of the other classes to get a better understanding of how the CNN is making its predictions.

Conclusion: the region which has higher intensity has contributed the most in classifying one class from another class

Analysing the CNN architecture:

Calendar

Description automatically generated

From the above figure we can analyse the CNN architecture in detail. Given the input image, we first apply the convolution with 32 3x3 filter, resulting image can be analysed form the above figure. Then we apply max pooling to further reduce the dimension capturing only the pixel values that contains the most information. After max pooling we again apply a layer of convolution to further reduce the dimension and to extract the features. Once the features has been extracted, we then flatten the pixel values. This flattened layer is known as encodings.

"encoding" refers to the process of transforming input data (usually images) into a more compact and abstract representation that captures important features and patterns in the input. This is typically done using a series of convolutional and pooling layers that progressively reduce the spatial dimensions of the input while increasing the number of feature maps or channels.

The output of the encoding layers is often flattened and fed into one or more fully connected layers that performs the classification task. In our case, we have used the dense layer.

The fully connected layer is then connected to the SoftMax layer. the SoftMax activation function is typically used in the final layer to convert the output of the network into a probability distribution over the classes. This allows us to interpret the output of the network as a probability that the input belongs to each class, which can be used to make predictions or perform classification.

**Analysing the CNN feature map for layer 1**

Table

Description automatically generated with medium confidence

**Analysing the feature map for layer 2**

Table

Description automatically generated with medium confidence

**Plot the ROC and Precision Recall Curve for Q2 model1.**

**ROC Curve interpretation**

In this ROC curve, there are five classes (Class 0, Class 1, Class 2, Class 3, and Class 4), and each class has an associated AUC value. The AUC (Area Under the Curve) is a metric that represents the overall performance of the model. An AUC value of 1.0 indicates a perfect classifier, while an AUC value of 0.5 indicates a classifier that performs no better than random chance.

Looking at the AUC values for each class in this ROC curve, we can see that they are all quite high, ranging from 0.99 to 1.0. This suggests that the model is performing very well for each of the five classes. Specifically, Class 1 and Class 3 have an AUC value of 1.0, which indicates that the model can perfectly distinguish between positive and negative examples for these classes.

Chart, line chart

Description automatically generated

A picture containing chart

Description automatically generated

**PR Curve interpretation**

In this PR curve, there are five classes, and each class has an associated average precision (AP) value. An AP value of 1.0 indicates a perfect classifier, while an AP value of 0.0 indicates a classifier that performs no better than random chance.

Looking at the AP values for each class in this PR curve, we can see that they are all relatively high, ranging from 0.95 to 1.0. This suggests that the model is performing very well for each of the five classes. Specifically, Class 1 has an AP value of 1.0, which indicates that the model can perfectly distinguish between positive and negative examples for this class.

It also highlights the areas where the model's performance could potentially be improved, such as for Class 0 and Class 2 where the AP values are slightly lower than the others.

Chart

Description automatically generated

**Confusion Matrix interpretation**

Looking at the values in the matrix, we can see that the majority of the diagonal values are high, indicating that the model is making correct predictions for those classes. For example, the value of 1697 in the (1,1) position indicates that there were 1697 instances where the true class was class 0, and the model correctly predicted it as class 0. Similarly, the value of 977 in the (2,2) position indicates that there were 977 instances where the true class was class 1, and the model correctly predicted it as class 1.

However, there are also some off-diagonal values, indicating misclassifications. For example, the value of 54 in the (3,1) position indicates that there were 54 instances where the true class was class 2, but the model incorrectly predicted it as class 0. Similarly, the value of 34 in the (4,5) position indicates that there were 34 instances where the true class was class 4, but the model incorrectly predicted it as class 5.

**Question4:**

Chart, scatter chart

Description automatically generated

**PCA plot of features with true labels interpretation**

In this plot, the encoding features of a multi-class classification problem have been visualized using the first two principal components obtained from PCA. The plot has five different colors to represent the labels or classes.

The points or clusters that are closer together in the plot represent encoding features that are more similar in nature. Conversely, points that are further apart from each other indicate encoding features that are more distinct from each other.

The fact that some clusters are overlapping suggests that the features belonging to those classes are not clearly separable based on the first two principal components. This indicates that the classification model may have difficulty in accurately distinguishing between those classes based on those features.

Chart, scatter chart

Description automatically generated

**PCA plot of features with K means cluster interpretation.**

The PCA plot of features with K-means clustering algorithms shows the distribution of the encoded features after clustering them into 5 different groups using the K-means algorithm. Each data point in the plot represents a single encoded feature and is colored according to its assigned cluster. The plot shows that the clusters are well separated, indicating that the K-means algorithm has successfully grouped similar features together.

The clusters are visually separable, which suggests that the features have been extracted and clustered effectively. The fact that the clusters are not overlapping suggests that the K-means algorithm was able to identify distinct patterns in the feature space. Overall, the plot shows that the encoding and clustering processes were successful and that the resulting clusters are well-defined.

Chart, scatter chart

Description automatically generated

**PCA plot of features with DB Scan cluster interpretation.**

we applied dbscan clustering algorithms on the features extracted from our designed model to cluster the data points based on their similarities.

We visualized the clustering results using a PCA plot of the first two components, with each data point colored according to its true class label.

The plot showed that for the five classes, only one cluster was obtained, indicating that the data points in each class were not well separated based on the selected features.

This suggests that further feature engineering or the use of more sophisticated clustering algorithms may be necessary to achieve better separation and clustering of the data points in this classification task.

Chart

Description automatically generated

**TSNE plot of features with true labels interpretation.**

The plot shows the features in a two-dimensional space, where each point represents a feature and the color represents the true label of that feature. There are five classes in the data, and it appears that some of the classes have overlapping points, indicating that the features for those classes may be similar or difficult to distinguish from one another based on the extracted features.

Here, we can see that that features from the few classes are overlapped with each other which indicates that the few encodings of the classes are very similar. For example, sneakers and ankle boots or sweatshirts vs coats.

Chart, map

Description automatically generated

**TSNE plot of features with K means cluster interpretation.**

The resulting plot shows the data points colored according to the K-means clustering algorithm. The plot has 5 classes and it appears that the classes and clusters are well separated, which indicates that the K-means algorithm was able to successfully group similar data points together. This could suggest that the model's feature extraction is effective in separating the different classes, and that K-means clustering is a suitable method for further analysis of the data.

Chart

Description automatically generated

**TSNE plot of features with DB Scan cluster interpretation.**

The t-SNE plot of features with DBSCAN clusters suggests that the DBSCAN algorithm was not able to effectively separate the features into distinct clusters for the five classes. Instead, all the points appear to be grouped into a single cluster. This may indicate that the features are too closely grouped together or that the algorithm was not set up correctly for the given data.

**Conclusion:**

The results obtained from the different analysis of the given dataset have revealed some interesting findings. The class activation map has provided insights into the 5 classes present in the dataset, namely t shirt, Sandals, trousers, bag, and coats.

based on the results obtained from the PCA plot of features with K-means clustering and t-SNE plots, it can be concluded that the 5 identified classes are valid and well-separated. Class 0 corresponds to tshirt/shirt/top, class 1 to Sandal, class 2 to trousers/pullovers, class 3 to bags/sneakers, and class 4 to coat/dress/ankle boots.

Using the known labels and visualizing the samples.

A picture containing text

Description automatically generated