# Fashion MNIST Data Classification Project

**SHAILESH.R**
**cb.en.u4ece20254@cb.students.amrita.edu**

## 1. ABSTRACT:

In this project, we have built a fashion apparel recognition using the Convolutional Neural Network (CNN) model. To train the CNN model, we have used the Fashion MNIST dataset. After successful training, the CNN model can predict the name of the class given apparel item belongs to. This is a multiclass classification problem in which there are 10 apparel classes the items will be classified. The fashion training set consists of 70,000 images divided into 60,000 training and 10,000 testing samples.Dataset sample consists of 28x28 grayscale images, associated with a label from 10 classes. So the end goal is to train and test the model using Convolution neural network.

## 2. OBJECTIVE:

The objective is to identify (predict) different fashion products from the given images using Convolutional neural network.

## 3. INTRODUCTION:

This project focuses on predictive analysis of different fashion products using a convolutional neural network.Now Convolutional Neural Networks, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on Convolutional Neural Networks. Neural networks, as its name suggests, is a machine learning technique which is modeled after the brain structure. It comprises of a network of learning units called neurons. These neurons learn how to convert input signals (e.g. picture of a cat) into corresponding output signals (e.g. the label “cat”), forming the basis of automated recognition.From an example of automatic image recognition , the process of determining whether a picture contains a cat involves an activation function can be determined. If the picture resembles prior cat images the neurons have seen before, the label “cat” would be activated. Hence, the more labeled images the neurons are exposed to, the better it learns how to recognize other unlabelled images. Hence this is the process of training neurons.

*Convolutional Layer:*
This layer is the main layer of CNN. When an image is fed into the convolution layer, a filter or a kernel of varying size but generally of size 3×3 is used to detect the features. The dot product is carried out with the image, and the kernel is the output is stored in a cell of a matrix which is called a feature map or an activation map. Once the operation is done, the filter moves by a distance and then repeats the process. This distance is called a stride. After each convolution operation, a ReLu transformation is applied to the feature map to introduce non-linearity into the model.

*Pooling Layer:*
This layer is responsible for reducing the number of parameters in the next layer. It is also known as downsampling or dimensionality reduction.

*Fully Connected Layer:*
Neurons in this layer have full connectivity to all the neurons in the preceding layer and the succeeding layer. FC layer helps to map the input with the output.

## 4. METHODOLOGY :

### 4.1 Machine Intelligence Library:

_Keras with Google TensorFlow backend was used to implement the deep learning algorithms in this study, with the aid of other scientific computing libraries: matplotlib, numpy, and seaborn.

## 4.2 The Datasets:

In this section, we describe the datasets used for the deep learning models used in the experiments.

### *4.2.1 MNIST:*

MNIST is one of the established standard datasets for benchmarking deep learning models. It is a 10-class classification problem having 60,000 training examples, and 10,000 test cases – all in grayscale, with each image having a resolution of 28 × 28. 2.2.2

### *4.2.2Fashion-MNIST:*

A dataset as an alternative to the conventional MNIST. The new dataset consists of 28 × 28 grayscale images of 70, 000 fashion products from 10 classes, with 7, 000 images per class.

## 4.3 Data Preprocessing:

For the case of MNIST and Fashion-MNIST, we employed Principal Component Analysis (PCA) for dimensionality reduction. That is, to select the representative features of image data. Hence the process involved:

1. Import Libraries
2. Load Data
3. Show Image from Numbers
4. Change Dimension /Feature Scaling

## 4.4 The Model:

We will create a straightforward CNN architecture with three convolutional layers followed by three max-pooling layers for this dataset. Convolutional layers will perform the convolutional operation and extract the features, while the max-pooling layer will down sample the features.Once the model architecture is defined, we will compile and build the model.We use Adam optimizers in most CNN architectures because it is very efficient on larger problems and helps us achieve correct weights and learning rates with minimum loss.Once all the model parameters are set, the model is ready to be trained. We will train the model for ten epochs, with each epoch having 100 steps.

## 5.CODE:

```
                          import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import keras

(X_train, y_train), (X_test, y_test)=tf.keras.datasets.fashion_mnist.load_data()

# Print the shape of data
X_train.shape,y_train.shape, "***************" , X_test.shape,y_test.shape

X_train[0]
y_train[0]

class_labels = [   "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt",     "Sneaker",          "B
class_labels

# show image
plt.imshow(X_train[0],cmap='Greys')

plt.figure(figsize=(16,16))

j=1
for  i in np.random.randint(0,1000,25):
  plt.subplot(5,5,j);j+=1
  plt.imshow(X_train[i],cmap='Greys')
  plt.axis('off')
  plt.title('{} / {}'.format(class_labels[y_train[i]],y_train[i]))

  X_train.ndim

X_train = np.expand_dims(X_train,-1)
X_train.ndim
```

```
X_test=np.expand_dims(X_test,-1)
# feature scaling
X_train = X_train/255
X_test= X_test/255
# Split dataset
from sklearn.model_selection import  train_test_split
X_train,X_Validation,y_train,y_Validation=train_test_split(X_train,y_train,test_size=0.2,random_state=2020)
X_train.shape,X_Validation.shape,y_train.shape,y_Validation.shape

model=keras.models.Sequential([

keras.layers.Conv2D(filters=32,kernel_size=3,strides=(1,1),padding='valid',activation='relu',input_shape=[28,28,
1]),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Flatten(),
                         keras.layers.Dense(units=128,activation='relu'),
                         keras.layers.Dense(units=10,activation='softmax')
])
model.summary()

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(X_train,y_train,epochs=10,batch_size=512,verbose=1,validation_data=(X_Validation,y_Validation))

y_pred = model.predict(X_test)
y_pred.round(2)

y_test

model.evaluate(X_test, y_test)


j=1
for i in np.random.randint(0, 1000,25):
  plt.subplot(5,5, j); j+=1
  plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
  plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]], y_test[i],
class_labels[np.argmax(y_pred[i])],np.argmax(y_pred[i])))
  plt.axis('off')

plt.figure(figsize=(16,30))

j=1
for i in np.random.randint(0, 1000,60):
  plt.subplot(10,6, j); j+=1
  plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
  plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]], y_test[i],
class_labels[np.argmax(y_pred[i])],np.argmax(y_pred[i])))
  plt.axis('off')

"""## Confusion Matrix"""

from sklearn.metrics import confusion_matrix
plt.figure(figsize=(16,9))
y_pred_labels = [ np.argmax(label) for label in y_pred ]
cm = confusion_matrix(y_test, y_pred_labels)
sns.heatmap(cm, annot=True, fmt='d',xticklabels=class_labels, yticklabels=class_labels)

from sklearn.metrics import classification_report
cr= classification_report(y_test, y_pred_labels, target_names=class_labels)
print(cr)

"""# Save Model"""
model.save('fashion_mnist_cnn_model.h5')

#Building CNN model
cnn_model2 = keras.models.Sequential([
                         keras.layers.Conv2D(filters=32, kernel_size=3, strides=(1,1),
padding='valid',activation= 'relu', input_shape=[28,28,1]),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Conv2D(filters=64, kernel_size=3, strides=(2,2), padding='same',
activation='relu'),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Flatten(),
                         keras.layers.Dense(units=128, activation='relu'),
                         keras.layers.Dropout(0.25),
                         keras.layers.Dense(units=256, activation='relu'),
                         keras.layers.Dropout(0.25),
                         keras.layers.Dense(units=128, activation='relu'),
                         keras.layers.Dense(units=10, activation='softmax')
                         ])

# complie the model
cnn_model2.compile(optimizer='adam', loss= 'sparse_categorical_crossentropy', metrics=['accuracy'])

#Train the Model
cnn_model2.fit(X_train, y_train, epochs=20, batch_size=512, verbose=1, validation_data=(X_Validation,
y_Validation))

cnn_model2.save('fashion_mnist_cnn_model2.h5')

"""######## very complex model"""

#Building CNN model
cnn_model3 = keras.models.Sequential([
                         keras.layers.Conv2D(filters=64, kernel_size=3, strides=(1,1),
padding='valid',activation= 'relu', input_shape=[28,28,1]),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Conv2D(filters=128, kernel_size=3, strides=(2,2), padding='same',
activation='relu'),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Conv2D(filters=64, kernel_size=3, strides=(2,2), padding='same',
activation='relu'),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Flatten(),
                         keras.layers.Dense(units=128, activation='relu'),
                         keras.layers.Dropout(0.25),
                         keras.layers.Dense(units=256, activation='relu'),
                         keras.layers.Dropout(0.5),
                         keras.layers.Dense(units=256, activation='relu'),
                         keras.layers.Dropout(0.25),
                         keras.layers.Dense(units=128, activation='relu'),
                         keras.layers.Dropout(0.10),
                         keras.layers.Dense(units=10, activation='softmax')
                         ])

# complie the model
cnn_model3.compile(optimizer='adam', loss= 'sparse_categorical_crossentropy', metrics=['accuracy'])

#Train the Model
cnn_model3.fit(X_train, y_train, epochs=50, batch_size=512, verbose=1, validation_data=(X_Validation,
y_Validation))

cnn_model3.save('fashion_mnist_cnn_model3.h5')

cnn_model3.evaluate(X_test, y_test)
```

## **6.CONCLUSION:**

From our code we were able to understand the basics of deep learning of CNN networks with the help of this project.Also from the built model we could be able to say that we built the model for the classification of MNIST fashion and the code was implemented successfully.It Concluded the fashion MNIST Clothing classification problem is a new standard data set used in computer version and deep learning.