**Practical 3:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.utils import to\_categorical

# Load the dataset

data = pd.read\_csv('C:/Users/Administrator/Documents/BE/DL Pratical/fashion-MNIST.csv')

# Split features and labels

X = data.iloc[:, 1:].values.astype('float32')  # Features (pixels)

y = data.iloc[:, 0].values.astype('int32')     # Labels (categories)

# Preprocess the data

X /= 255.0  # Normalize pixel values to be between 0 and 1

X = X.reshape(-1, 28, 28, 1)  # Reshape the features into 28x28 grayscale images

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, *test\_size*=0.2, *random\_state*=42)

# Convert labels to one-hot encoding

num\_classes = len(np.unique(y))

y\_train = to\_categorical(y\_train, num\_classes)

y\_test = to\_categorical(y\_test, num\_classes)

# Build the CNN model

model = Sequential([

    Conv2D(32, (3, 3), *activation*='relu', *input\_shape*=(28, 28, 1)),

    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), *activation*='relu'),

    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), *activation*='relu'),

    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(128, *activation*='relu'),

    Dropout(0.5),

    Dense(num\_classes, *activation*='softmax')

])

# Compile the model

model.compile(*optimizer*='adam', *loss*='categorical\_crossentropy', *metrics*=['accuracy'])

# Train the model

model.fit(X\_train, y\_train, *epochs*=10, *batch\_size*=128, *validation\_split*=0.2)

# Evaluate the model

test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test)

print(*f*'Test Accuracy: {test\_accuracy}')

# Make predictions

predictions = model.predict(X\_test)

# Print some actual and predicted classes

print("Some actual and predicted classes:")

for i in range(10):  # Print predictions for the first 10 images

    actual\_class = np.argmax(y\_test[i])

    predicted\_class = np.argmax(predictions[i])

    print(*f*"Sample {i+1}: Actual cloth class: {actual\_class}, Predicted cloth class: {predicted\_class}")