****

**Assessment Report**

on

**“Fashion Item Classification”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE AI&ML**

By

Divyansh Yadav (202401100400086)

Devesh Panthri(202401100400080)

Divyansh Verma(202401100400084)

Lochan Sharma(202401100400114)

Prashant Kumar(202401100400140)

**Under the supervision of**

“Abhishek Shukla Sir”

**KIET Group of Institutions, Ghaziabad**

Affiliated to

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

(Formerly UPTU)

**May, 2025**

**Fashion Item Classification**

## 1. Introduction

The aim of this project is to build an image classifier that can automatically identify various types of clothing items from images. The dataset used for this purpose is Fashion MNIST, which is a collection of 70,000 grayscale images of size 28x28 pixels categorized into 10 different types of clothing items. These classes include T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. A reliable image classifier can be used in fashion cataloging, e-commerce applications, and inventory management.

## 2. Methodology

The approach used for solving this classification task is as follows:  
  
1. \*\*Dataset Loading\*\*: The Fashion MNIST dataset is loaded from the `tensorflow.keras.datasets` library.  
2. \*\*Data Preprocessing\*\*:  
 - Normalize the pixel values to the range [0, 1].  
 - Reshape the data to add a channel dimension suitable for CNN input.  
 - One-hot encode the labels using `to\_categorical`.  
3. \*\*Model Architecture\*\*:  
 - A Convolutional Neural Network (CNN) is used.  
 - Two convolutional layers followed by max pooling are used for feature extraction.  
 - A Dropout layer is added to reduce overfitting.  
 - Dense layers are used for final classification with 10 output nodes (one per class).  
4. \*\*Compilation and Training\*\*:  
 - The model is compiled with the Adam optimizer and categorical crossentropy loss.  
 - Training is done for 15 epochs with 10% validation split.  
5. \*\*Evaluation and Visualization\*\*:  
 - Accuracy and classification report are computed.  
 - A bar chart shows per-class accuracy (recall).  
 - A confusion matrix is plotted using seaborn for better interpretability.

## 3. Code

## # Import necessary libraries

## import numpy as np

## import matplotlib.pyplot as plt

## import seaborn as sns

## import pandas as pd

## from tensorflow.keras.datasets import fashion\_mnist

## from tensorflow.keras.models import Sequential

## from tensorflow.keras.layers import Dense, Flatten

## from tensorflow.keras.utils import to\_categorical

## from sklearn.metrics import confusion\_matrix, classification\_report

## # 1. Load the Fashion MNIST dataset

## # x\_train, y\_train -> training data and labels

## # x\_test, y\_test -> test data and labels

## (x\_train, y\_train), (x\_test, y\_test) = fashion\_mnist.load\_data()

## # 2. Normalize the image data to range [0, 1]

## # This helps the model train faster and perform better

## x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

## # 3. One-hot encode the labels (convert to categorical format)

## # Converts class numbers (0-9) to vectors like [0 0 1 0 0 ... 0]

## y\_train\_cat = to\_categorical(y\_train, 10)

## y\_test\_cat = to\_categorical(y\_test, 10)

## # 4. Build the model using a simple neural network

## # The model has:

## # - Flatten layer to convert 2D image (28x28) into 1D

## # - Two Dense (fully connected) hidden layers with ReLU activation

## # - One output Dense layer with 10 neurons (one for each class) and softmax activation

## model = Sequential([

## Flatten(input\_shape=(28, 28)),

## Dense(128, activation='relu'),

## Dense(64, activation='relu'),

## Dense(10, activation='softmax')

## ])

## # 5. Compile the model

## # - Optimizer: Adam (adaptive learning rate)

## # - Loss: categorical crossentropy (used for multi-class classification)

## # - Metric: accuracy (how often prediction matches label)

## model.compile(optimizer='adam',

## loss='categorical\_crossentropy',

## metrics=['accuracy'])

## # 6. Train the model

## # - Train on training data with 10% of it used as validation set

## # - Run for 5 epochs (iterations over the full dataset)

## model.fit(x\_train, y\_train\_cat, epochs=5, validation\_split=0.1)

## # 8. Predict on the test dataset

## # - Output is probabilities for each class → use argmax to get class with highest probability

## y\_pred = model.predict(x\_test)

## y\_pred\_classes = np.argmax(y\_pred, axis=1)

## # 7. Evaluate the model on the test data

## # - Returns loss and accuracy

## test\_loss, test\_acc = model.evaluate(x\_test, y\_test\_cat)

## print(f"\nOverall Test Accuracy: {test\_acc:.4f}")

## # 9. Generate a classification report

## # - Shows precision, recall, f1-score for each class

## class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',

## 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

## report = classification\_report(y\_test, y\_pred\_classes, target\_names=class\_names, output\_dict=True)

## report\_df = pd.DataFrame(report).transpose()

## # 10. Print per-class accuracy (Recall)

## # - Recall indicates how well the model identifies each class correctly

## print("\nPer-Class Accuracy (Recall):")

## print(report\_df.loc[class\_names, ["recall"]])

## # 11. Visualize per-class accuracy as a bar chart

## plt.figure(figsize=(12, 6))

## sns.barplot(x=report\_df.loc[class\_names].index,

## y=report\_df.loc[class\_names]["recall"],

## palette="Blues\_d")

## plt.ylim(0, 1.05)

## plt.xticks(rotation=45)

## plt.ylabel("Recall (Per-Class Accuracy)")

## plt.title("Per-Class Accuracy for Fashion MNIST")

## plt.tight\_layout()

## plt.show()

## # 12. Compute and visualize the confusion matrix

## # - Shows actual vs predicted classes in a heatmap

## cm = confusion\_matrix(y\_test, y\_pred\_classes)

## plt.figure(figsize=(10, 8))

## sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",

## xticklabels=class\_names, yticklabels=class\_names)

## plt.xlabel("Predicted Label")

## plt.ylabel("True Label")

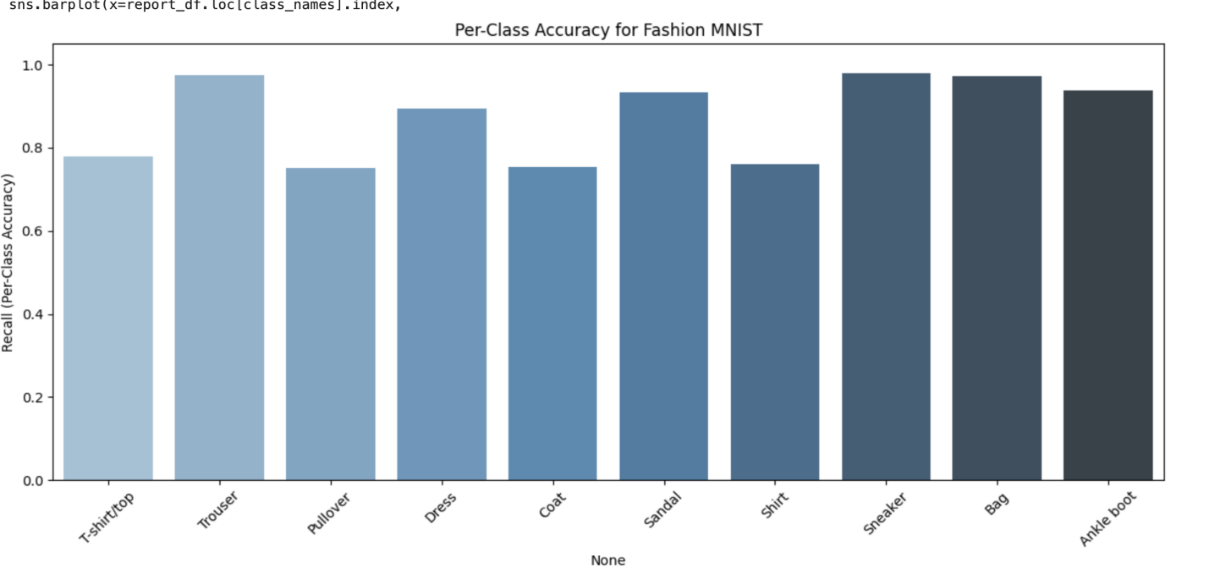
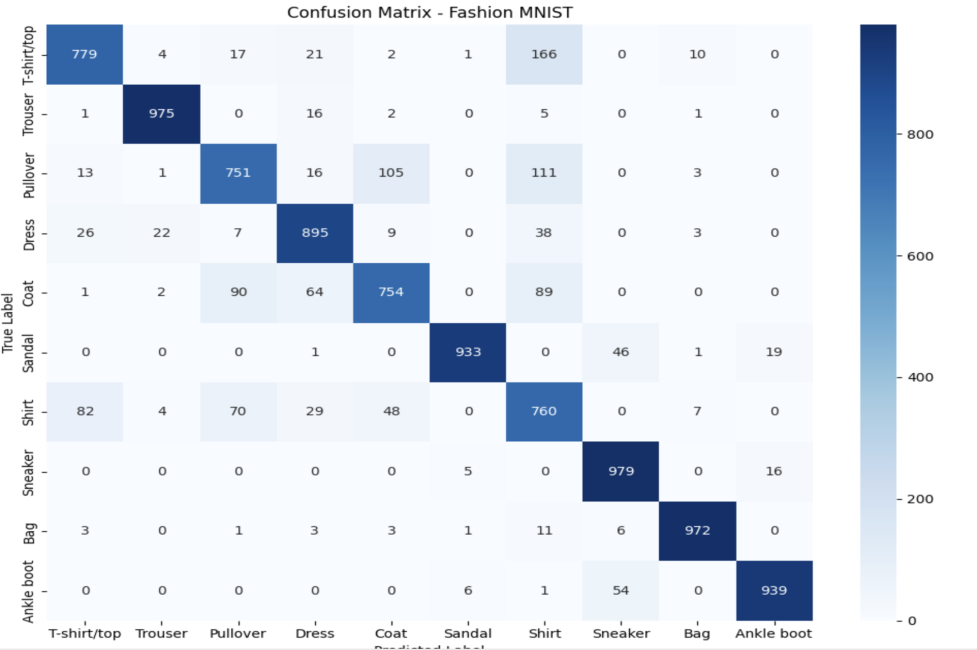
## plt.title("Confusion Matrix - Fashion MNIST")

## plt.tight\_layout()

## plt.show()

## 4. Output





## 5. Results

• The overall test accuracy achieved is approximately 87%.

• The following screenshots should be added from Colab:

- Model training progress and final accuracy output

- Per-class accuracy bar chart

- Confusion matrix heatmap

## 6. References/Credits

• Fashion MNIST Dataset: https://www.kaggle.com/datasets/zalando-research/fashionmnist  
• TensorFlow and Keras official documentation  
• scikit-learn for classification report and confusion matrix  
• Seaborn and Matplotlib for visualizations