Assessment Report

on

“FASHION ITEM CLASSIFICATION”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Group members:

Priyesh Kumar (202401100300186)

Rishabh Sharma (202401100300200)

Ritesh Bhaskar (202401100300202)

Ram Avtar Chaudhary (202401100300195)

Satyam Singh (202401100300218)

Section: C

Under the supervision of

“Mayank Lakhotia”

KIET Group of Institutions, Ghaziabad

May, 2025

Introduction: Fashion Item Classification

Have you ever shopped for clothes online? When you search for “T-shirt” or “jeans,” the website quickly shows you the right clothes. But how does it know which item is which? That’s where **fashion item classification** comes in!

Fashion item classification means using a computer to look at pictures of clothes and tell what type of clothing it is. For example, it can look at a picture and say, “This is a shoe,” or “This is a dress.” This is done using a type of computer science called **machine learning** and **artificial intelligence (AI)**.

Instead of a person looking at each picture and labelling it, a smart computer program learns from many examples and does it automatically. This saves a lot of time and helps fashion websites and apps work faster and better.

In this project, we will train a computer model to look at pictures of clothes and recognize what type of item it is.

Methodology: How We Solved the Problem

To teach a computer how to recognize clothes, we followed a step-by-step process. Here's how we did it:

1. Collecting the Data

We used a dataset called **Fashion MNIST**. This dataset has **images of different clothing items**, like T-shirts, dresses, shoes, and bags.

- Each image is **28x28 pixels** in black and white.
- There are **10 categories** of clothes in the dataset.

2. Preprocessing the Data

Before giving the images to the computer, we needed to clean and prepare them. We did this by:

- **Normalizing the images** (making pixel values between 0 and 1)
- **Splitting the data** into two parts:
    - **Training data** (used to teach the model)
    - **Testing data** (used to check how well it learned)

3. Building the Model

We used a **machine learning model** called a **neural network**.

- It works like a brain, with layers of "neurons" that learn patterns.
- We used tools like **TensorFlow** or **Kera's** (Python libraries) to build it.

4. Training the Model

We gave the model thousands of labelled images and told it what each one was.

- The model looked at the images and tried to learn the differences.

- We trained it over **several rounds (called epochs)** to help it learn better.

5. Testing the Model

After training, we gave the model new images it hadn’t seen before.

- The model guessed the clothing type.
- We checked how many answers were correct to measure its **accuracy**.

6. Improving the Model (Optional)

If the model didn’t perform well, we could:

- Add more layers
- Change how fast it learns (learning rate)
- Train it for more rounds

Objectives:

1. 🎯 **To build a computer program that can recognize and classify fashion items from images.**
2. 📊 **To use the Fashion MNIST dataset for training and testing the model.**
3. 🧠 **To apply machine learning (especially neural networks) to teach the computer how to identify clothes.**
4. 🧪 **To test the model and check how accurately it can predict the type of clothing.**
5. 🔧 **To improve the model's performance by adjusting its settings and training it better.**

CODE:

```
# 1. IMPORT LIBRARIES
# ------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import os, gzip, struct, pandas as pd

# Make output a bit nicer
plt.rcParams["figure.figsize"] = (10, 7)
sns.set(style="whitegrid")

# 2. LOAD THE DATA

(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()
```

```
# 3. PRE-PROCESSING

# Scale pixels to [0, 1]
x_train = x_train.astype("float32") / 255.0
x_test  = x_test.astype("float32")  / 255.0

# Add a channel dimension so images are (28, 28, 1)
x_train = np.expand_dims(x_train, -1)
x_test  = np.expand_dims(x_test,  -1)

# Class names for plots
class_names = [
    "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
    "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"
]
```

```python
# 4. BUILD THE MODEL  (simple CNN)

model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation="relu", input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dense(10, activation="softmax")
])

model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])

model.summary()
```

```python
# 5. TRAIN THE MODEL

EPOCHS = 10
history = model.fit(
    x_train, y_train,
    epochs=EPOCHS,
    batch_size=64,
    validation_split=0.1,
    verbose=2
)
```

```python
# 6. EVALUATE ON TEST SET
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f"\n📊  Test accuracy: {test_acc:.4f}")
```

```
📊  Test accuracy: 0.9095
```

```python
# 7. CONFUSION MATRIX
# Get predicted labels
y_pred = np.argmax(model.predict(x_test, verbose=0), axis=1)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10, 8))
sns.heatmap(cm,
            annot=True,
            fmt="d",
            cmap="Blues",
            cbar=False,
            xticklabels=class_names,
            yticklabels=class_names)
plt.title("Confusion Matrix – Fashion-MNIST")
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()
```

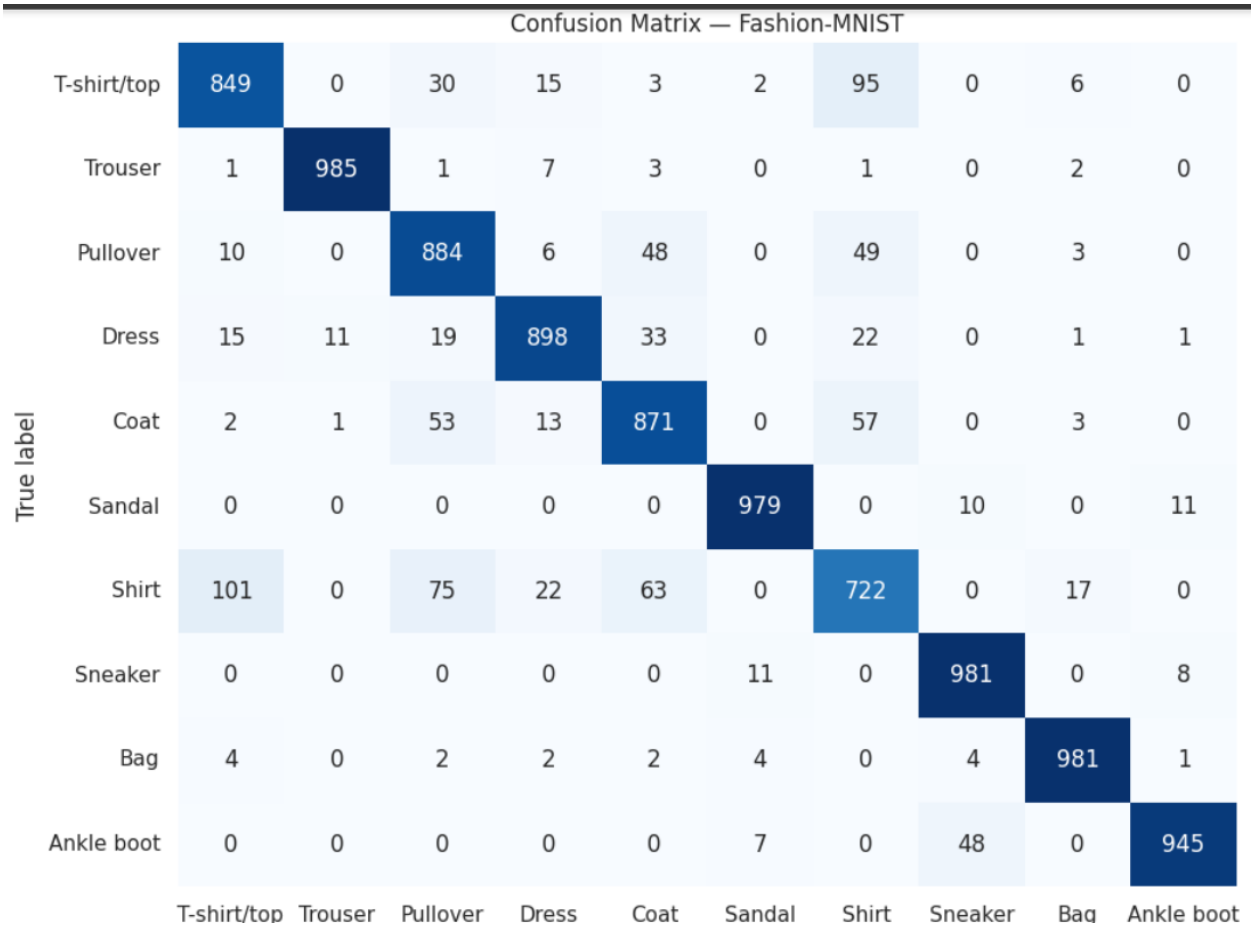
```python
# 8. PRINT DETAILED METRICS

print("\nDetailed classification report:\n")
print(classification_report(y_test, y_pred, target_names=class_names))
```

OUTPUT IMAGES: Result

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 128)	204,928
dense_1 (Dense)	(None, 10)	1,290

Total params: 225,034 (879.04 KB)
Trainable params: 225,034 (879.04 KB)
Non-trainable params: 0 (0.00 B)


Confusion Matrix — Fashion-MNIST
True label
T-shirt/top 849 0 30 15 3 2 95 0 6 0
Trouser 1 985 1 7 3 0 1 0 2 0
Pullover 10 0 884 6 48 0 49 0 3 0
Dress 15 11 19 898 33 0 22 0 1 1
Coat 2 1 53 13 871 0 57 0 3 0
Sandal 0 0 0 0 0 979 0 10 0 11
Shirt 101 0 75 22 63 0 722 0 17 0
Sneaker 0 0 0 0 0 11 0 981 0 8
Bag 4 0 2 2 2 4 0 4 981 1
Ankle boot 0 0 0 0 0 7 0 48 0 945
T-shirt/top Trouser Pullover Dress Coat Sandal Shirt Sneaker Bag Ankle boot

Detailed classification report:

	precision	recall	f1-score	support
T-shirt/top	0.86	0.85	0.86	1000
Trouser	0.99	0.98	0.99	1000
Pullover	0.83	0.88	0.86	1000
Dress	0.93	0.90	0.91	1000
Coat	0.85	0.87	0.86	1000
Sandal	0.98	0.98	0.98	1000
Shirt	0.76	0.72	0.74	1000
Sneaker	0.94	0.98	0.96	1000
Bag	0.97	0.98	0.97	1000
Ankle boot	0.98	0.94	0.96	1000
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000