Assignment No: 3B

Title of the Assignment:

Roll No. : - 19121028

> Use MNIST Fashion Dataset and create a classifier to classify fashion clothing into categories.

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

# Load the dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

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

# Plot some of the images
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```

```
# Preprocess the data
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

x_train = np.expand_dims(x_train, axis=-1)
x_test = np.expand_dims(x_test, axis=-1)
```

```
# Define the model architecture
model = keras.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

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

```
# Train the model
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
```

```
Epoch 1/10
1875/1875 [==============================] - 82s 43ms/step - loss: 0.4309 - accuracy: 0.8434 - val_loss: 0.3573 - val_accuracy: 0.8
Epoch 2/10
1875/1875 [==============================] - 83s 45ms/step - loss: 0.2939 - accuracy: 0.8917 - val_loss: 0.2896 - val_accuracy: 0.8
Epoch 3/10
1875/1875 [==============================] - 71s 38ms/step - loss: 0.2479 - accuracy: 0.9092 - val_loss: 0.2802 - val_accuracy: 0.8
Epoch 4/10
1875/1875 [==============================] - 74s 40ms/step - loss: 0.2185 - accuracy: 0.9180 - val_loss: 0.2717 - val_accuracy: 0.9
Epoch 5/10
1875/1875 [==============================] - 72s 39ms/step - loss: 0.1904 - accuracy: 0.9283 - val_loss: 0.2532 - val_accuracy: 0.9
Epoch 6/10
1875/1875 [==============================] - 70s 37ms/step - loss: 0.1687 - accuracy: 0.9380 - val_loss: 0.2650 - val_accuracy: 0.9
Epoch 7/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.1492 - accuracy: 0.9437 - val_loss: 0.2446 - val_accuracy: 0.9
Epoch 8/10
1875/1875 [==============================] - 72s 38ms/step - loss: 0.1311 - accuracy: 0.9515 - val_loss: 0.2682 - val_accuracy: 0.9
Epoch 9/10
1875/1875 [==============================] - 72s 38ms/step - loss: 0.1150 - accuracy: 0.9566 - val_loss: 0.2945 - val_accuracy: 0.9
Epoch 10/10
1875/1875 [==============================] - 72s 38ms/step - loss: 0.1009 - accuracy: 0.9621 - val_loss: 0.3016 - val_accuracy: 0.9
```

```
# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)

print('Test accuracy:', test_acc)
```

```
313/313 - 3s - loss: 0.3016 - accuracy: 0.9122 - 3s/epoch - 9ms/step
Test accuracy: 0.9121999740600586
```

8m 22s completed at 12:47 PM