AIM:

Develop a neural network model utilizing the MNIST Fashion dataset by employing a sequential Keras model. Include 2 to 3 intermediate hidden layers in the model architecture.

-Abhinaya Rajarajan

ABSTRACT

The primary objective will be to build a classification model which will be able to identify the different categories of the fashion industry from the Fashion MNIST dataset using Keras. The model architecture includes 2 intermediate hidden layers to classify the fashion items accurately. We will utilize the Sequential model from Keras to build the neural network.

INTRODUCTION

The MNIST database of handwritten digits is one of the most widely used data sets used to explore Neural Networks and became a benchmark for model comparison. The Fashion MNIST dataset is meant to be a replacement for the original MNIST and promises to be more challenging, so that machine learning algorithms have to learn more advanced features to correctly classify the images. The dataset consists of 70,000 images, of which 60,000 are for training, and the remaining are for testing purposes. The images are in grayscale format. Each image consists of 28×28 pixels, and the number of categories is 10. Hence there are 10 labels available to us, and they are as follows:

- T-shirt/top
- Trouser
- Pullover
- Dress
- Coat
- Sandal
- Shirt
- Sneaker
- Bag
- Ankle boot

This assignment focuses on developing a neural network model using the MNIST Fashion dataset and the Keras framework. The goal is to create a sequential model with 2 to 3 intermediate hidden layers for accurately classifying fashion items based on their images. The assignment covers loading the dataset, preprocessing the data, building the model architecture, training the model, and evaluating its performance. By the end, the aim is to have a trained model capable of effectively classifying fashion items, showcasing the practical application of deep learning in image classification tasks.

CODE

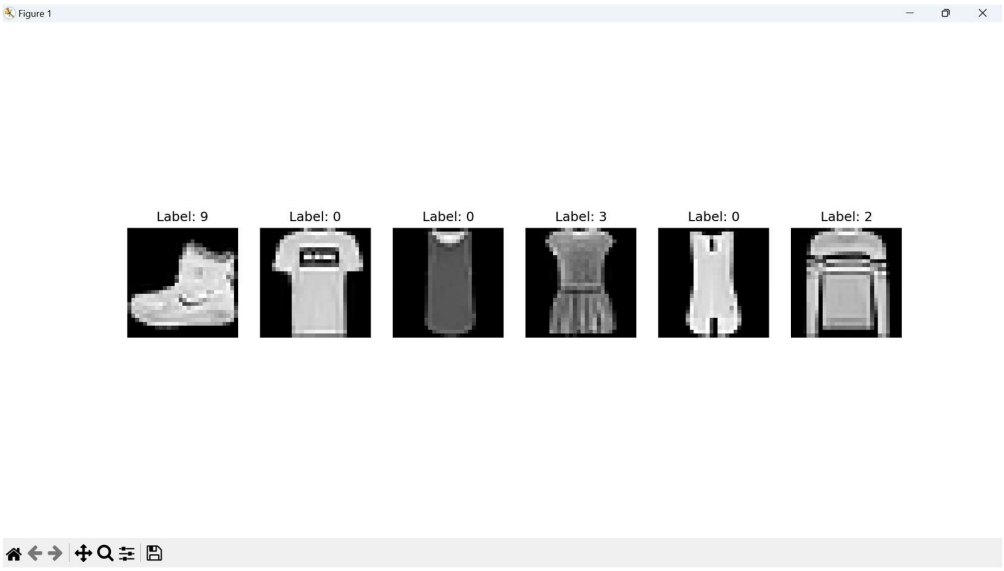
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from keras.datasets import fashion_mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Flatten

#Load Fashion MNIST dataset
(train_images, train_labels),(test_images, test_labels) =
fashion_mnist.load_data()

#Display Sample images
def display_samples(images, labels, num_samples=6):
    plt.figure(figsize=(10, 2))
    for i in range(num_samples):
       plt.subplot(1, num_samples, i+1)
       plt.imshow(images[i], cmap='gray')
       plt.title(f'Label: {labels[i]}')
       plt.axis('off')
    plt.show()
display_samples(train_images, train_labels)

#Preprocess the data
train_images=
train_images.reshape((60000,28,28,1)).astype('float32')/255
test_images=
test_images.reshape((10000,28,28,1)).astype('float32')/255

train_labels=to_categorical(train_labels)
test_labels=to_categorical(test_labels)

# Build a neural network model with 3 intermediate hidden layers
model = Sequential()
model.add(Flatten(input_shape=(28, 28, 1)))    # Flatten the input
images
```

```
model.add(Dense(128, activation='relu'))       # First hidden layer
with 128 neurons and ReLU activation
model.add(Dense(64, activation='relu'))        # Second hidden layer
with 64 neurons and ReLU activation
model.add(Dense(32, activation='relu'))        # Third hidden layer
with 32 neurons and ReLU activation
model.add(Dense(10, activation='softmax'))     # Output layer with
10 neurons for 10 classes and softmax activation

#Compile the model
model.compile(optimizer='adam',loss='categorical_crossentropy',metri
cs=['accuracy'])

#Train the model
history=model.fit(train_images,train_labels,epochs=5,batch_size=64,v
alidation_split=0.2)

#Evaluate the model on the test set
test_loss,test_acc=model.evaluate(test_images,test_labels)
print(f'Test Loss:{test_loss}')
print(f'Test accuracy:{test_acc}')

#Make predictions on a few test images
predictions=model.predict(test_images[:6])
predicted_labels=np.argmax(predictions,axis=1)
actual_labels=np.argmax(test_labels[:6],axis=1)

#Display the test images and their predictions
def display_predictions(images,actual,predicted):
    plt.figure(figsize=(10,2))
    for i in range(len(images)):
        plt.subplot(1,len(images),i+1)
        plt.imshow(images[i].reshape(28,28),cmap='gray')
        title=f'Actual:{actual[i]}\nPredicted:{predicted[i]}'
        plt.title(title)
        plt.axis('off')
    plt.show()

display_predictions(test_images[:6],actual_labels,predicted_labels)

# Plotting accuracy vs epoch
plt.figure(figsize=(10, 6))
```

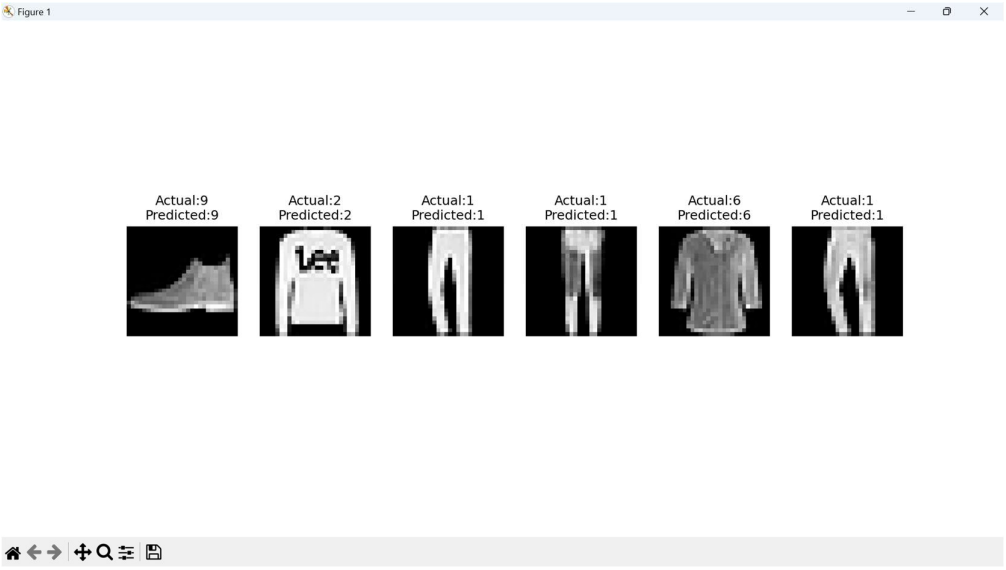
```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Test'], loc='upper left')

# Plotting loss vs epoch
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Test'], loc='upper left')

#Show Plots
plt.tight_layout()
plt.show()
```

OUTPUT

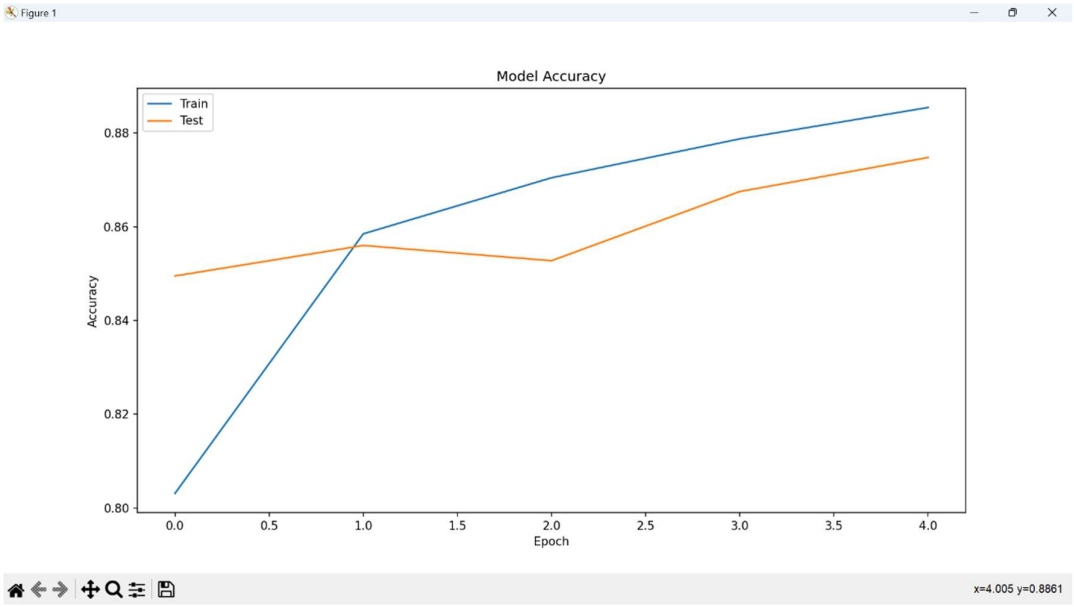
Displaying of sample images



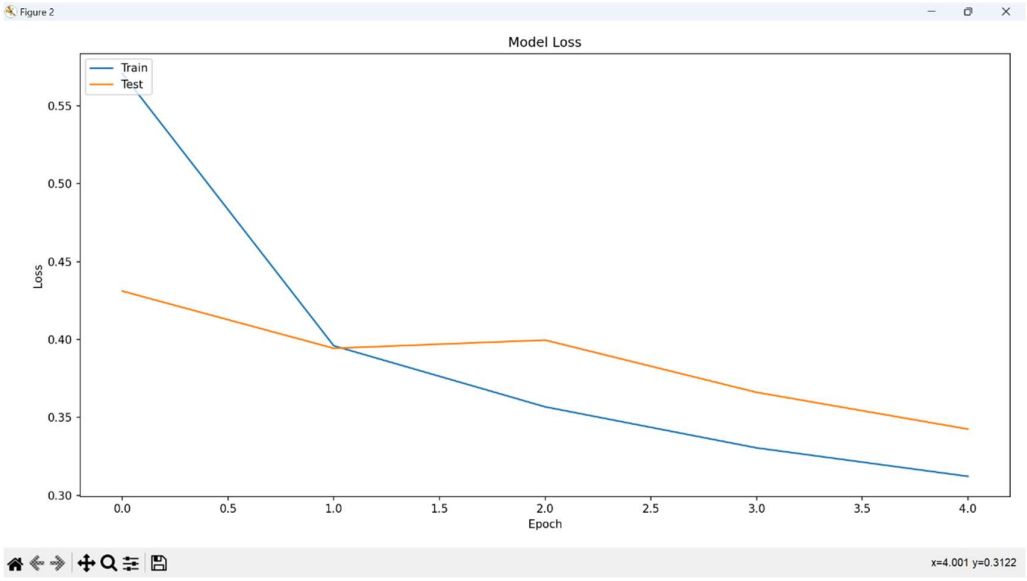
Predictions on Few Test Images



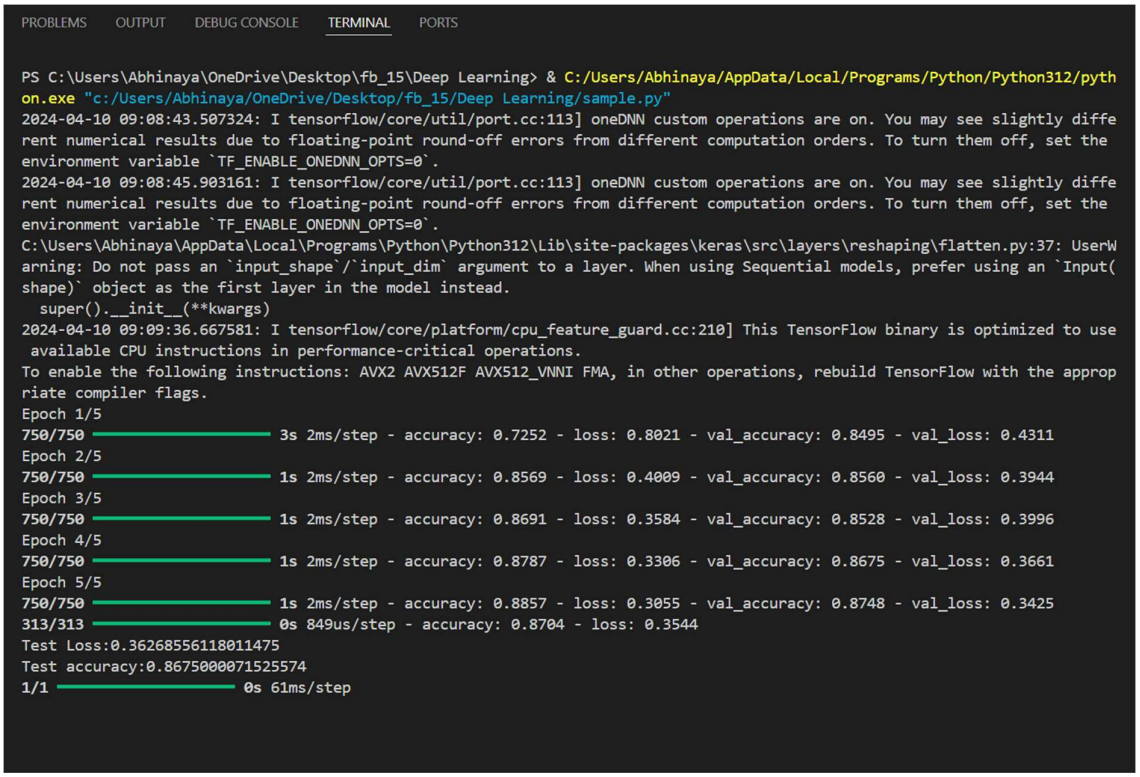
Plotting accuracy vs epoch



Plotting loss vs epoch



Terminal Window



CONCLUSION

The neural network model developed using the MNIST Fashion dataset and employing a sequential Keras model with 3 intermediate hidden layers has demonstrated promising results. The model was trained on the training dataset for 5 epochs with a batch size of 64.

Upon evaluating the model on the test dataset, it achieved a test accuracy of approximately 87.04% and a test loss of 0.3627. These metrics indicate that the model performs well on unseen data and can effectively classify fashion items based on their images.

The model's predictions on sample test images were accurate, showcasing its ability to correctly classify fashion items into their respective categories. The plots depicting training and validation accuracy/loss over epochs provided insights into the model's learning dynamics. The increasing accuracy and decreasing loss trends demonstrated that the model was learning from the data and avoiding overfitting.