ASSIGNMENT 5

NAME: VANI SETH

ER NO.: 201B299

1. Create a DNN model of at least three hidden layer and one output layer. Train this model to identify eight classes of Fashion items (Pick any eight classes of your choice). Show accuracy and loss after training the model.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
import numpy as np

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

x_train= x_train.reshape(x_train.shape[0], 28,28,1)
x_test= x_test.reshape(x_test.shape[0], 28,28,1)

x_train = x_train[y_train<8]
x_test = x_test[y_test<8]
y_train = y_train[y_train<8]
y_test = y_test[y_test<8]

# Normalize pixel values to range 0-1
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

#One-hot encode
y_train = keras.utils.to_categorical(y_train,8)
y_test = keras.utils.to_categorical(y_test,8)

# Arch
model = Sequential()
model.add(Flatten(input_shape=(28,28,1)))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(8, activation='softmax'))

#compile
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

#training
training = model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_test))

loss, acc = model.evaluate(x_test, y_test)
print('Accuracy:', acc)
print('Loss:', loss)
```

```
Epoch 1/10
1500/1500 [==============================] - 5s 3ms/step - loss: 0.5276 - accuracy: 0.8058 - val_loss: 0.4718 - val_accuracy: 0.8254
Epoch 2/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3980 - accuracy: 0.8512 - val_loss: 0.4383 - val_accuracy: 0.8394
Epoch 3/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3644 - accuracy: 0.8624 - val_loss: 0.3855 - val_accuracy: 0.8537
Epoch 4/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3428 - accuracy: 0.8711 - val_loss: 0.3735 - val_accuracy: 0.8621
Epoch 5/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3269 - accuracy: 0.8759 - val_loss: 0.3897 - val_accuracy: 0.8544
Epoch 6/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3122 - accuracy: 0.8809 - val_loss: 0.4114 - val_accuracy: 0.8443
Epoch 7/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.2980 - accuracy: 0.8862 - val_loss: 0.3759 - val_accuracy: 0.8648
Epoch 8/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.2881 - accuracy: 0.8909 - val_loss: 0.3616 - val_accuracy: 0.8715
Epoch 9/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.2787 - accuracy: 0.8934 - val_loss: 0.3666 - val_accuracy: 0.8664
Epoch 10/10
1500/1500 [==============================] - 5s 3ms/step - loss: 0.2730 - accuracy: 0.8969 - val_loss: 0.3559 - val_accuracy: 0.8686
250/250 [==============================] - 0s 1ms/step - loss: 0.3559 - accuracy: 0.8686
```

```
     Accuracy: 0.8686249852180481
     Loss: 0.3558727204799652
```

```
x_train.shape
```

```
     (60000, 28, 28)
```

```
x_test.shape
```

```
     (10000, 28, 28)
```

```
y_train.shape
```

```
     (60000,)
```

```
x_train.shape[0]
```

```
     60000
```

```
set(y_train)
```

```
     {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

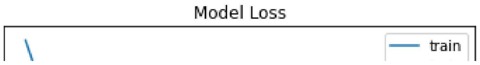
```
set(y_train[2:5])
```

```
     {0, 3}
```

```
import matplotlib.pyplot as plt

# Loss
plt.plot(training.history['loss'], label='train')
plt.plot(training.history['val_loss'], label='test')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Accuracy
plt.plot(training.history['accuracy'], label='train')
plt.plot(training.history['val_accuracy'], label='test')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



2. Use above model to classify rest of the fashion items. Remaining fashion classes are only two. Make suitable changes in above model wherever required and show accuracy and loss for the modified model.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
import numpy as np

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

x_train= x_train.reshape(x_train.shape[0], 28,28,1)
x_test= x_test.reshape(x_test.shape[0], 28,28,1)

# Filtering out remaining two classes
# x_train = x_train[(y_train == 8) | (y_train == 9)]
# y_train = y_train[(y_train == 8) | (y_train == 9)]
# x_test = x_test[(y_test == 8) | (y_test == 9)]
# y_test = y_test[(y_test == 8) | (y_test == 9)]

x_train = x_train[y_train >= 8]
y_train = y_train[y_train >=8] - 8
x_test = x_test[y_test>=8]
y_test = y_test[y_test >=8] - 8


# Normalize
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# One-hot encode
y_train = keras.utils.to_categorical(y_train, num_classes=None, dtype='float32')
y_test = keras.utils.to_categorical(y_test,num_classes=None, dtype='float32')

# Arch
model = Sequential()
model.add(Flatten(input_shape=(28,28,1)))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(2, activation='softmax'))

# Freeze all layers except the output layer
for layer in model.layers[:-1]:
    layer.trainable = False

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

# Train the model
training1 = model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_test))

# Evaluate the model
loss, acc = model.evaluate(x_test, y_test)
print('Accuracy:', acc)
print('Loss:',loss)
```

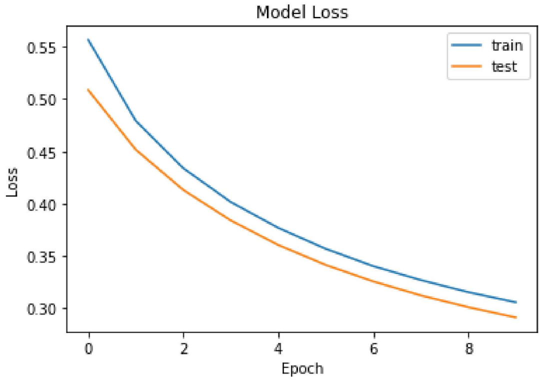
```
Epoch 1/10
375/375 [==============================] - 1s 2ms/step - loss: 0.5567 - accuracy: 0.7909 - val_loss: 0.5087 - val_accuracy: 0.8225
Epoch 2/10
375/375 [==============================] - 1s 2ms/step - loss: 0.4793 - accuracy: 0.8263 - val_loss: 0.4517 - val_accuracy: 0.8370
Epoch 3/10
375/375 [==============================] - 1s 2ms/step - loss: 0.4340 - accuracy: 0.8380 - val_loss: 0.4132 - val_accuracy: 0.8485
Epoch 4/10
375/375 [==============================] - 1s 2ms/step - loss: 0.4016 - accuracy: 0.8504 - val_loss: 0.3840 - val_accuracy: 0.8600
Epoch 5/10
375/375 [==============================] - 1s 2ms/step - loss: 0.3768 - accuracy: 0.8618 - val_loss: 0.3604 - val_accuracy: 0.8660
Epoch 6/10
375/375 [==============================] - 1s 2ms/step - loss: 0.3568 - accuracy: 0.8713 - val_loss: 0.3414 - val_accuracy: 0.8745
Epoch 7/10
375/375 [==============================] - 1s 2ms/step - loss: 0.3403 - accuracy: 0.8773 - val_loss: 0.3256 - val_accuracy: 0.8805
Epoch 8/10
```

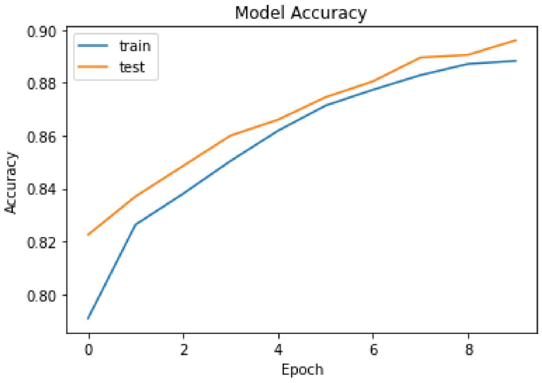
```
375/375 [==============================] - 1s 2ms/step - loss: 0.3269 - accuracy: 0.8828 - val_loss: 0.3122 - val_accuracy: 0.8895
Epoch 9/10
375/375 [==============================] - 1s 2ms/step - loss: 0.3153 - accuracy: 0.8871 - val_loss: 0.3010 - val_accuracy: 0.8905
Epoch 10/10
375/375 [==============================] - 1s 2ms/step - loss: 0.3056 - accuracy: 0.8882 - val_loss: 0.2911 - val_accuracy: 0.8960
63/63 [==============================] - 0s 2ms/step - loss: 0.2911 - accuracy: 0.8960
Accuracy: 0.8960000276565552
Loss: 0.2910982668399811
```

```
import matplotlib.pyplot as plt

# Loss
plt.plot(training1.history['loss'], label='train')
plt.plot(training1.history['val_loss'], label='test')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Accuracy
plt.plot(training1.history['accuracy'], label='train')
plt.plot(training1.history['val_accuracy'], label='test')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```


Model Loss
train
test
Loss
Epoch


Model Accuracy
train
test
Accuracy
Epoch