PROGRAM NO : 13

AIM : Programs on convolutional neural network to classify images from any standard dataset in the public domain

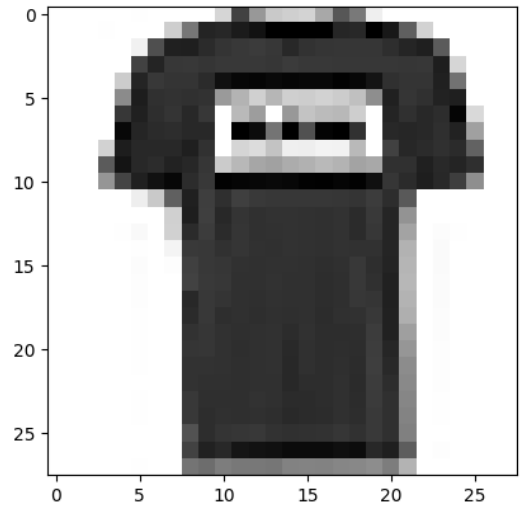
PROGRAM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from tensorflow import keras

np.random.seed(42)

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

x_train = x_train/255.0
x_test = x_test/255.0

plt.imshow(x_train[1], cmap ='binary')
plt.show()

np.unique(y_test)

class_name = ['T-
shirt/Top','Trouser','Pullover','Dress','Cost','Sandal','Shirt','Sneaker','Bag','Ankle
Boot']
n_rows = 5
n_cols = 10

plt.figure(figsize=(n_cols * 1.4, n_rows * 1.6))

for row in range(n_rows):
    for col in range(n_cols):
        index = n_cols * row + col
        plt.subplot(n_rows, n_cols,index + 1)
        plt.imshow(x_train[index], cmap='binary', interpolation='nearest')
        plt.axis('off')
        plt.title(class_name[y_train[index]])

plt.show()

model_CNN = keras.models.Sequential()

model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size = 7,padding='same',
activation='relu', input_shape=[28, 28, 1]))
model_CNN.add(keras.layers.MaxPooling2D(pool_size= 2))

model_CNN.add(keras.layers.Conv2D(filters=64, kernel_size = 3,padding='same',
activation='relu'))
model_CNN.add(keras.layers.MaxPooling2D(pool_size= 2))

model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size = 3,padding='same',
activation='relu'))
model_CNN.add(keras.layers.MaxPooling2D(pool_size= 2))

model_CNN.summary()

model_CNN.add(keras.layers.Flatten())
model_CNN.add(keras.layers.Dense(units=128,activation='relu'))
model_CNN.add(keras.layers.Dense(units=64,activation='relu'))
model_CNN.add(keras.layers.Dense(units=10,activation='softmax'))
```

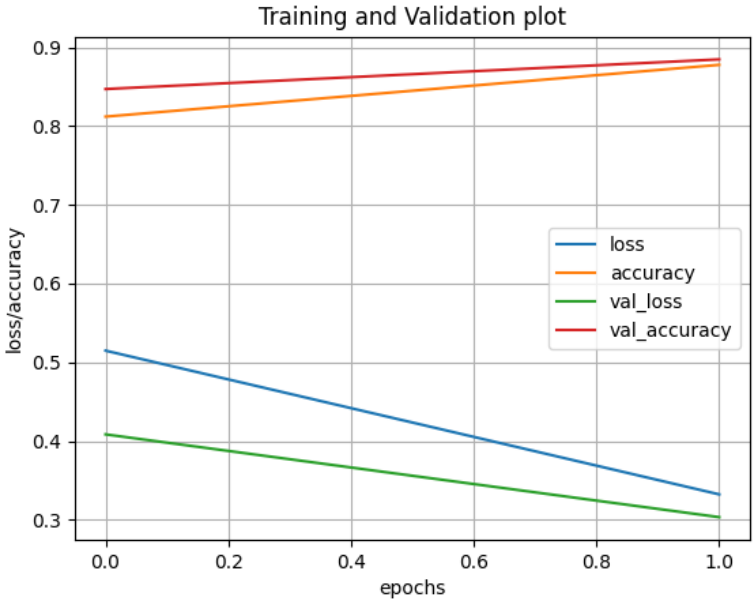
```
model_CNN.summary()

model_CNN.compile(loss='sparse_categorical_crossentropy',
optimizer='adam',metrics=['accuracy'])

x_train = x_train[...,np.newaxis]
x_test = x_test[...,np.newaxis]

history_CNN = model_CNN.fit(x_train, y_train, epochs=2,validation_split=0.1)
pd.DataFrame(history_CNN.history).plot()
plt.grid(True)
plt.xlabel('epochs')
plt.ylabel('loss/accuracy')
plt.title('Training and Validation plot')
plt.show()
test_loss, test_accuracy = model_CNN.evaluate(x_test, y_test)

print('Test Loss : {},Test Accuracy : {}'.format(test_loss, test_accuracy))
```

OUTPUT

```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        1600

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 32)          18464

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 32)         0
 2D)

=================================================================
Total params: 38,560
Trainable params: 38,560
Non-trainable params: 0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        1600

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        1600

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 32)          18464

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 32)         0
 2D)

 flatten (Flatten)           (None, 288)               0

 dense (Dense)               (None, 128)               36992

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 10)                650

=================================================================

=================================================================
Total params: 84,458
Trainable params: 84,458
Non-trainable params: 0
_________________________________________________________________
Epoch 1/2
1688/1688 [==============================] - 38s 22ms/step - loss: 0.5148 - accuracy: 0.8119 - val_loss: 0.4085 - val_accuracy: 0.8470
Epoch 2/2
1688/1688 [==============================] - 38s 22ms/step - loss: 0.3323 - accuracy: 0.8777 - val_loss: 0.3034 - val_accuracy: 0.8847
313/313 [==============================] - 2s 6ms/step - loss: 0.3215 - accuracy: 0.8789
Test Loss : 0.32149529457092285,Test Accuracy : 0.8788999915122986
```


0
5
10
15
20
25
0
5
10
15
20
25


Ankle Boot
T-shirt/Top
T-shirt/Top
Dress
T-shirt/Top
Pullover
Sneaker
Pullover
Sandal
Sandal
T-shirt/Top
Ankle Boot
Sandal
Sandal
Sneaker
Ankle Boot
Trouser
T-shirt/Top
Shirt
Cost
Dress
Trouser
Cost
Bag
Cost
Dress
T-shirt/Top
Pullover
Cost
Cost
Sandal
Dress
Shirt
Shirt
T-shirt/Top
Bag
Sandal
Pullover
Trouser
Shirt
Shirt
Sneaker
Ankle Boot
Sandal
Ankle Boot
Pullover
Sneaker
Dress
T-shirt/Top
Dress


Training and Validation plot
loss/accuracy
epochs
loss
accuracy
val_loss
val_accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0