**DATE:02/02/2022**

**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

import tensorflow as tf

from tensorflow import keras

np.random.seed(42)

# tf.set.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\_names = ['T-Shirt/Top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', '8ag', '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\_names[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')

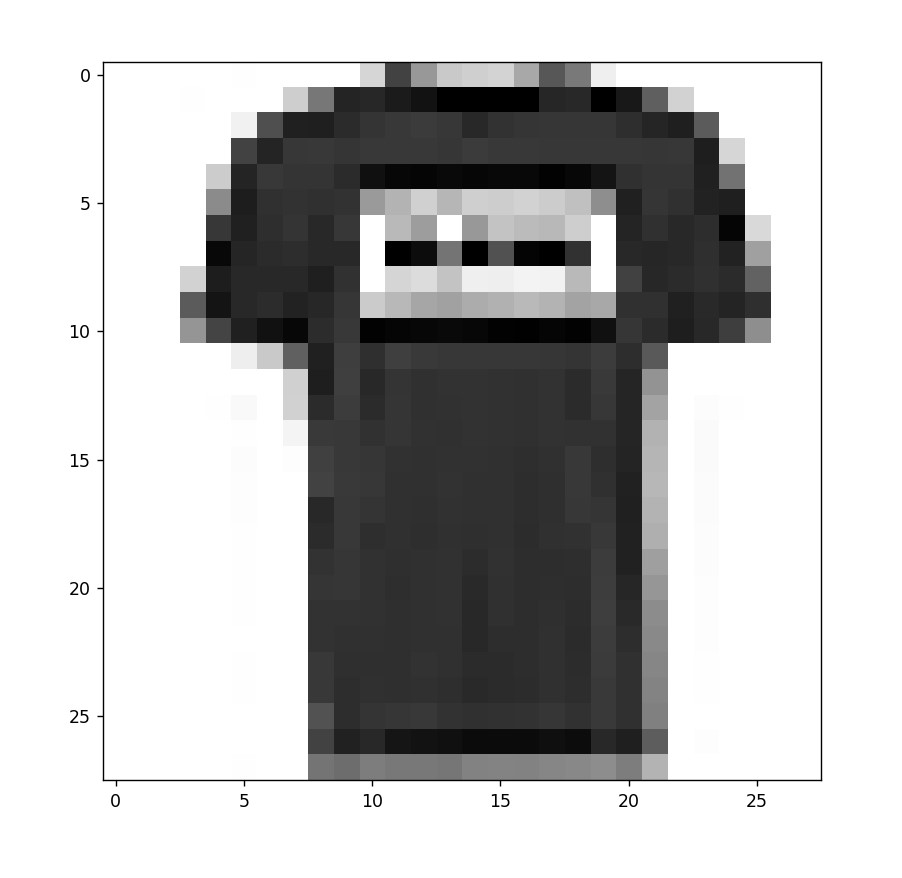
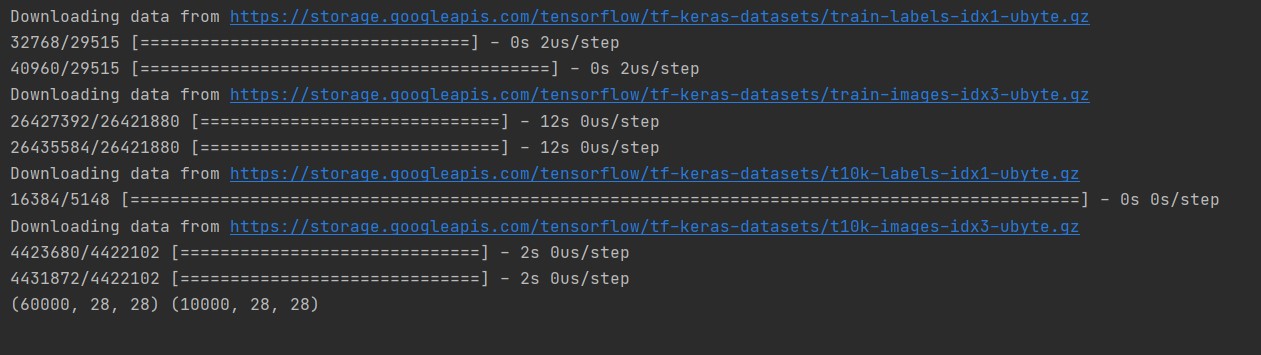
plt.title('Training and validation plot')

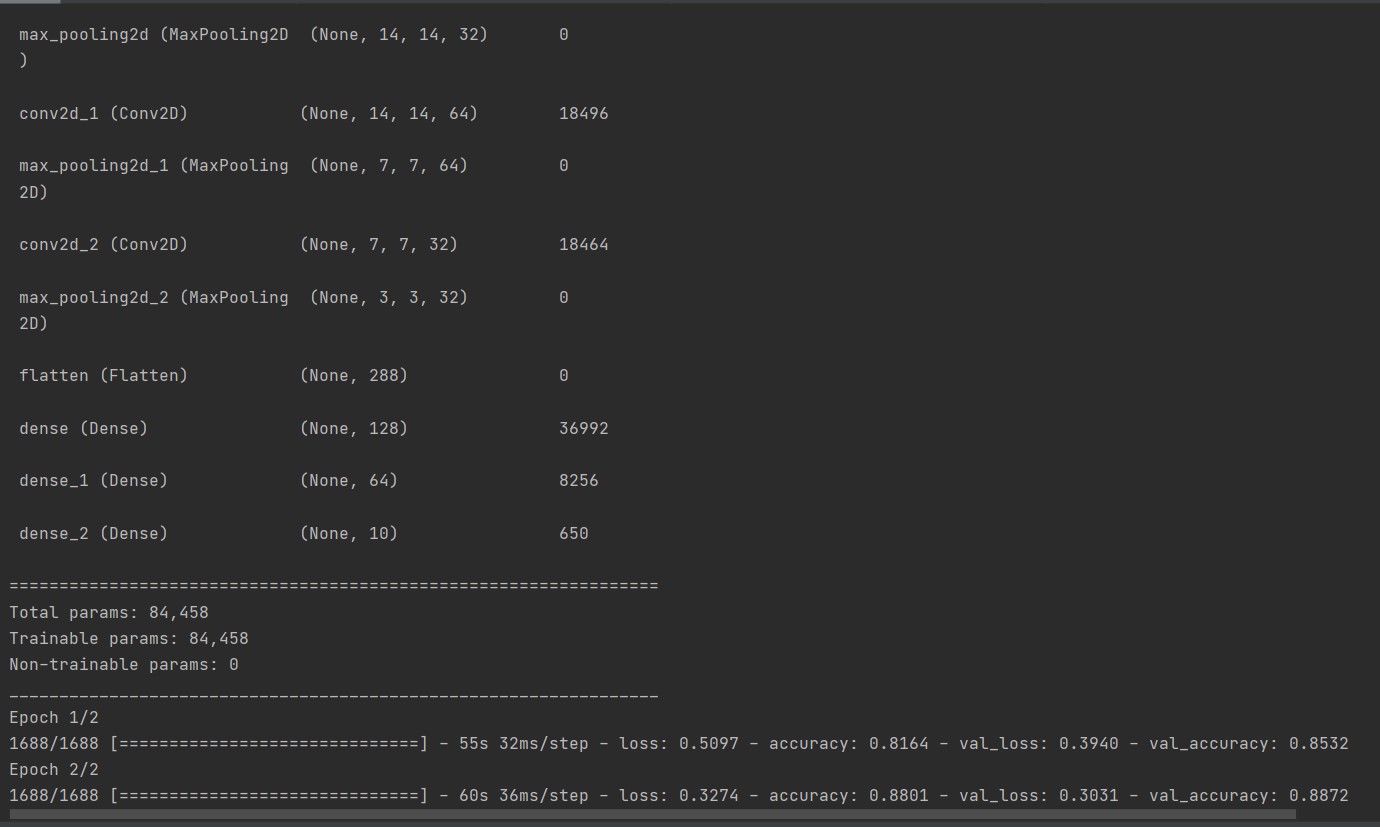
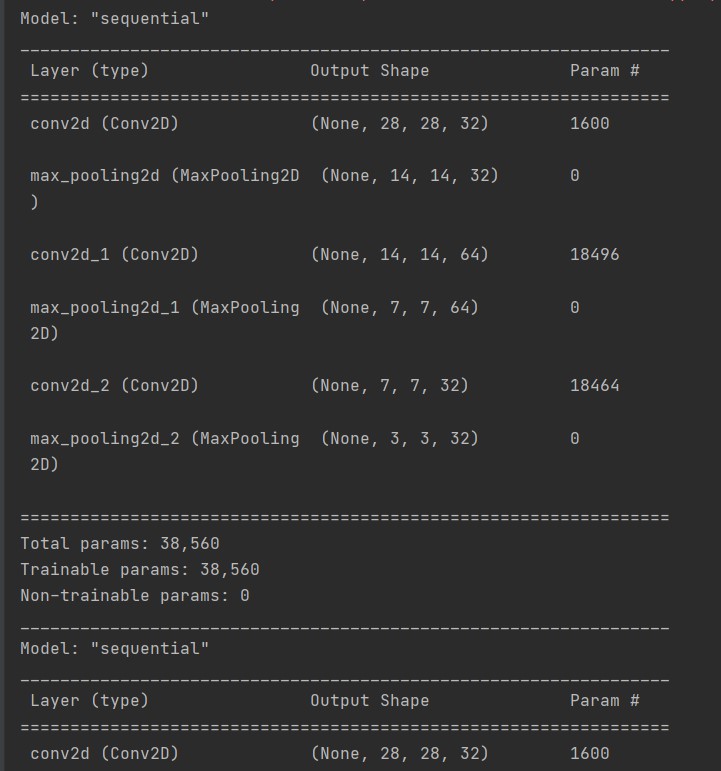
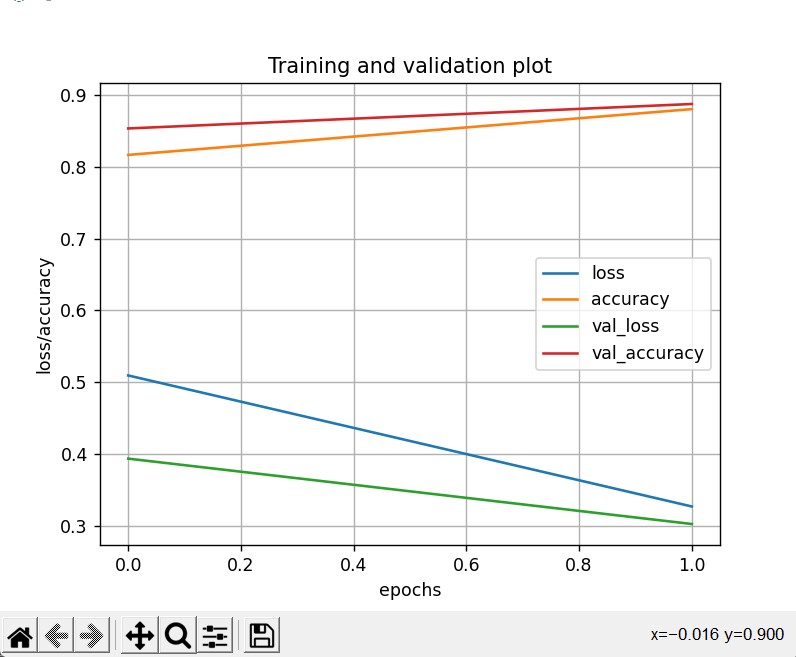
plt.show()

test\_loss, test\_accuracy = model\_CNN.evaluate(X\_test, y\_test)

print(' Test Loss :{}, Test Accuracy : {}'.format(test\_loss, test\_accuracy))

**OUTPUT**





**RESULT:** THE PROGRAM HAS BEEN EXECUTED AND OUTPUT VERIFIED