

Program No:-12

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

Program Code

```
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', '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_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()
```

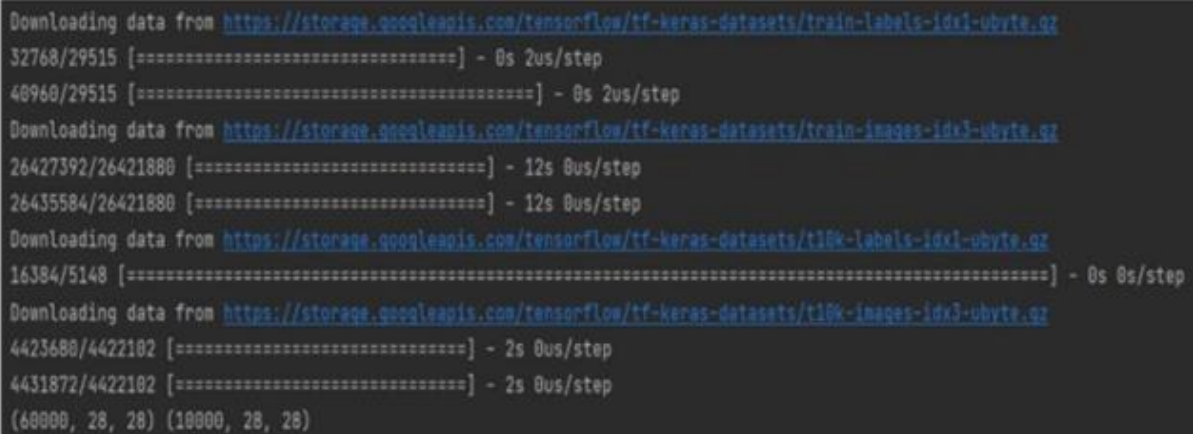
```
model_CNN.compile(loss='sparse_categorical_crossentropy',  
optimizer='adam', metrics=['accuracy'])
```

```
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')
```

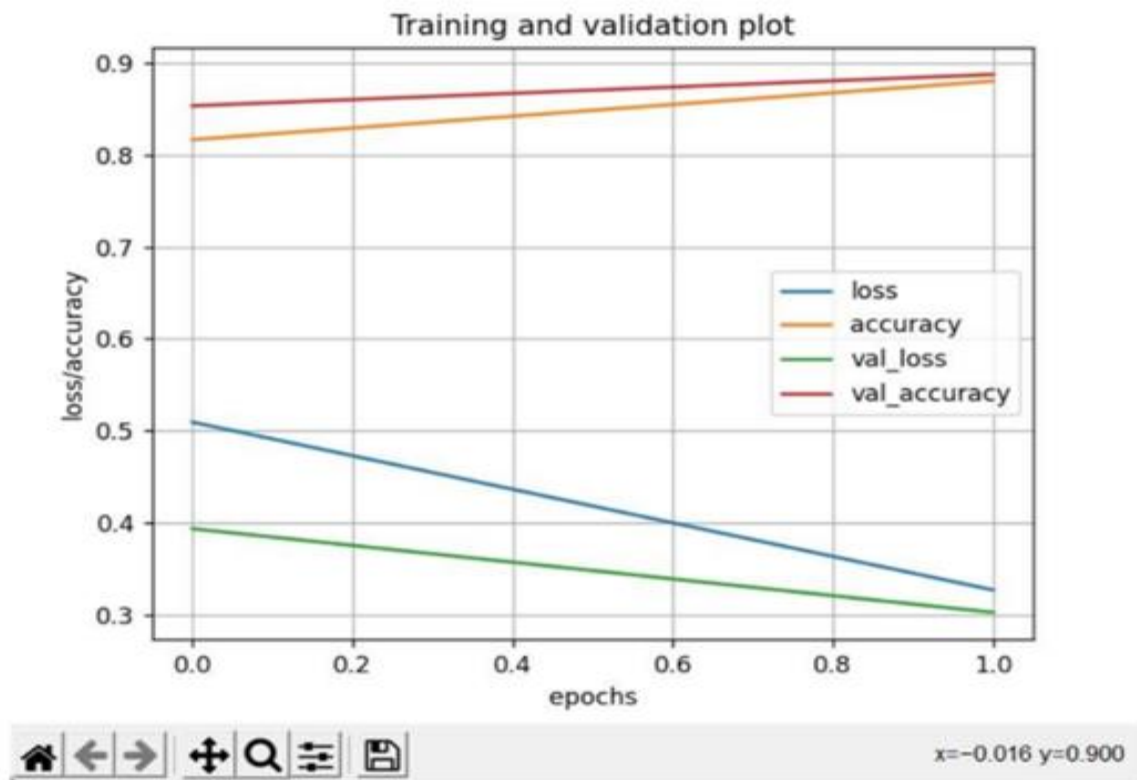
```
plt.title('Training and validation plot')
plt.show()
test_loss, test_accuracy = model_CNN.evaluate(X_test, y_test)
print(' Test Loss :{ }, Test Accuracy : { }'.format(test_loss,
test_accuracy))
```

Output



```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [=====] - 0s 2us/step
48960/29515 [=====] - 0s 2us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [=====] - 12s 0us/step
26435584/26421880 [=====] - 12s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
16384/5148 [=====] - 0s 0s/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [=====] - 2s 0us/step
4431872/4422102 [=====] - 2s 0us/step
(60000, 28, 28) (10000, 28, 28)
```





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 (MaxPooling2D)	(None, 7, 7, 64)	0
conv2d_2 (Conv2D)	(None, 7, 7, 32)	18464
max_pooling2d_2 (MaxPooling2D)	(None, 3, 3, 32)	0

=====
Total params: 38,560
Trainable params: 38,560
Non-trainable params: 0

Model: "sequential"

Layer (type)	Output Shape	Param #
--------------	--------------	---------

