Ex No: 3 **BUILD A CONVOLUTIONAL NEURAL NETWORK**

AIM:

To build a simple convolutional neural network with Keras/TensorFlow.

PROCEDURE:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

PROGRAM:

```
from tensorflow.keras.datasets import fashion_mnist

# Load the Fashion MNIST dataset
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
from tensorflow.keras.utils import to_categorical

# Reshape the images to add the channel dimension (28x28x1)
train_images = train_images.reshape((train_images.shape[0], 28, 28, 1))
test_images = test_images.reshape((test_images.shape[0], 28, 28, 1))

# Normalize the pixel values between 0 and 1
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

# One-hot encode the labels
train_labels = to_categorical(train_labels, 10)
test_labels = to_categorical(test_labels, 10)



from tensorflow.keras import layers, models
```

```
# Build the CNN model
model = models.Sequential()

# Add Convolutional layers and MaxPooling
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# Add Fully Connected layers
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
# Train the model
history = model.fit(train_images, train_labels,epochs=10, batch_size=64, validation_split=0.2)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f"Test accuracy: {test_acc}")

predictions = model.predict(test_images)
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt

# Generate confusion matrix
y_pred = predictions.argmax(axis=1)
y_true = test_labels.argmax(axis=1)
```

```
cm = confusion_matrix(y_true, y_pred)
print(classification_report(y_true, y_pred))

# Plotting accuracy and loss curves
plt.plot(history.history['accuracy'], label='train accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.legend()
plt.show()
```

OUTPUT:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 64)	36,928
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 64)	36,928
dense_1 (Dense)	(None, 10)	650

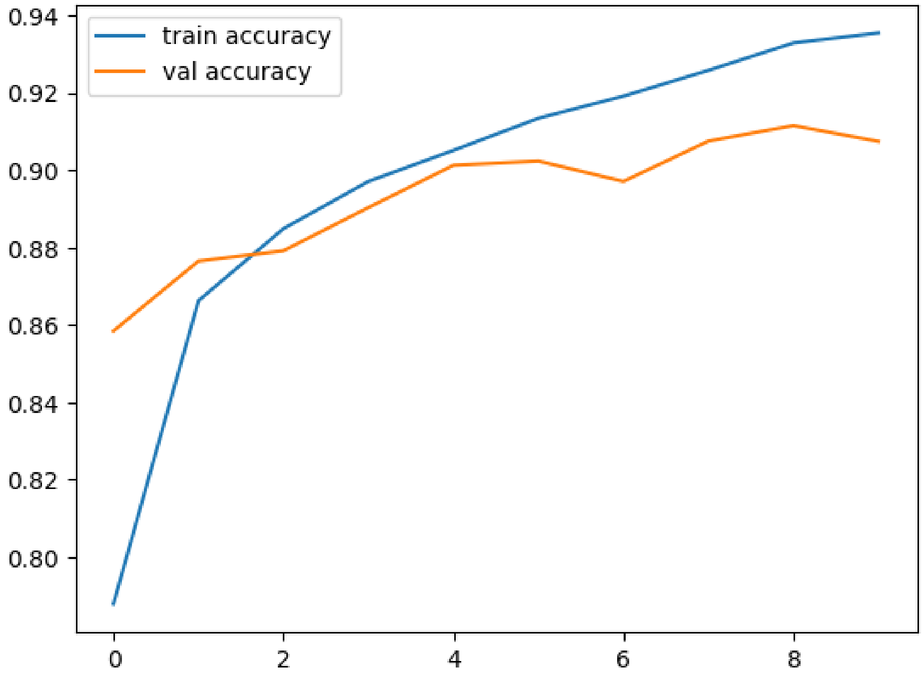
```
Total params: 93,322 (364.54 KB)
Trainable params: 93,322 (364.54 KB)
Non-trainable params: 0 (0.00 B)
```

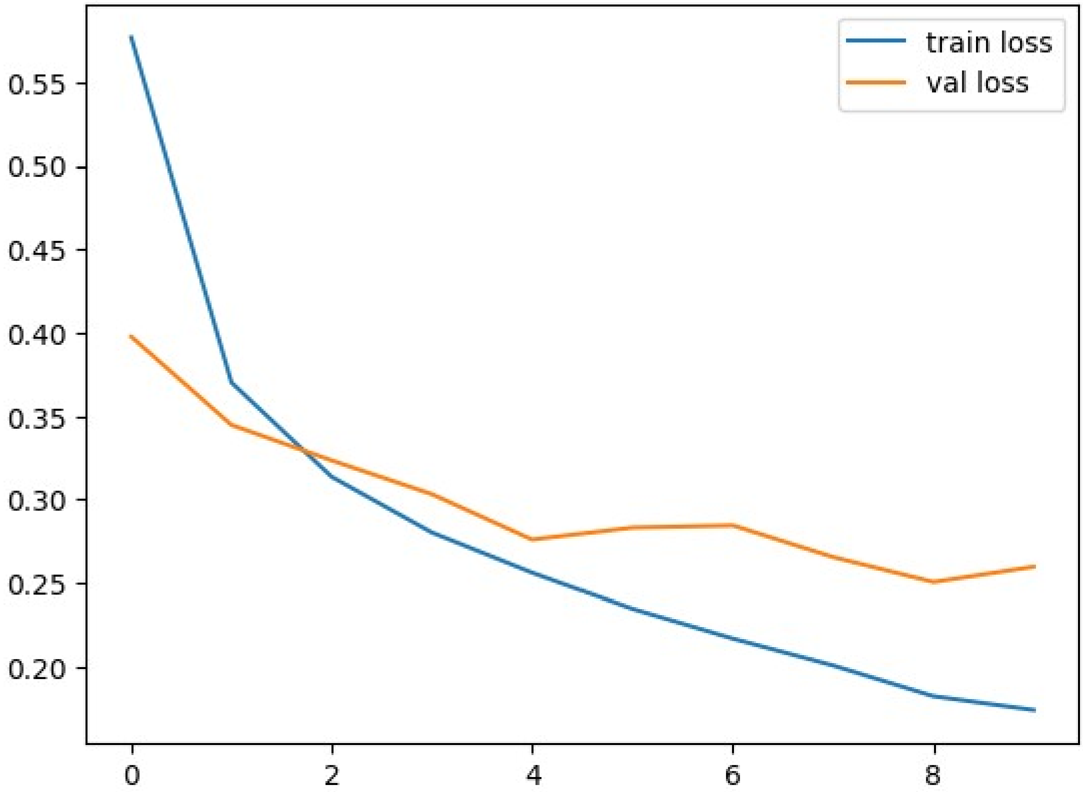
ROLL NUMBER: 210701105

```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85      1000
           1       0.99      0.98      0.98      1000
           2       0.81      0.89      0.85      1000
           3       0.93      0.89      0.91      1000
           4       0.88      0.83      0.86      1000
           5       0.98      0.97      0.98      1000
           6       0.72      0.74      0.73      1000
           7       0.96      0.96      0.96      1000
           8       0.96      0.98      0.97      1000
           9       0.96      0.97      0.96      1000

    accuracy                           0.90     10000
   macro avg       0.91      0.90      0.90     10000
weighted avg       0.91      0.90      0.90     10000
```





RESULT:

Thus a simple convolutional neural network with Keras/TensorFlow is built.