# 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))

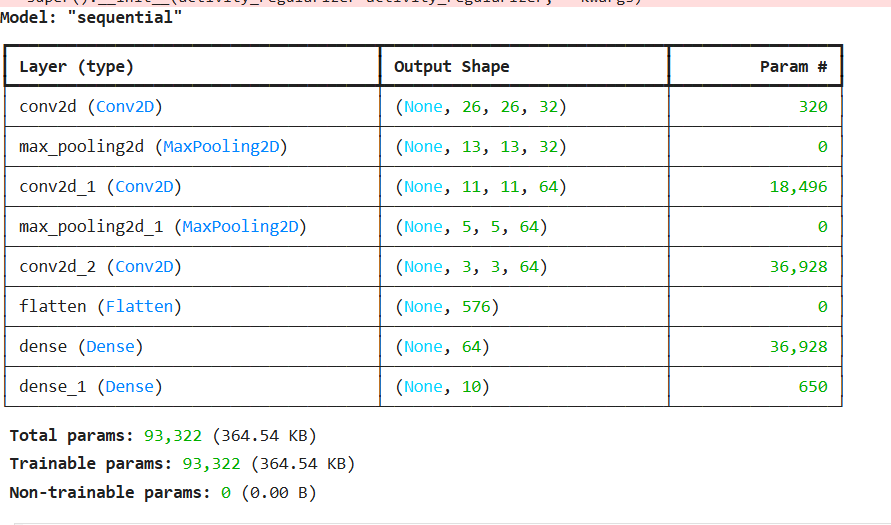
# Plotting accuracy and loss curves plt.plot(history.history['accuracy'], label='train accuracy') plt.plot(history.history['val\_accuracy'], label='val accuracy') plt.legend()

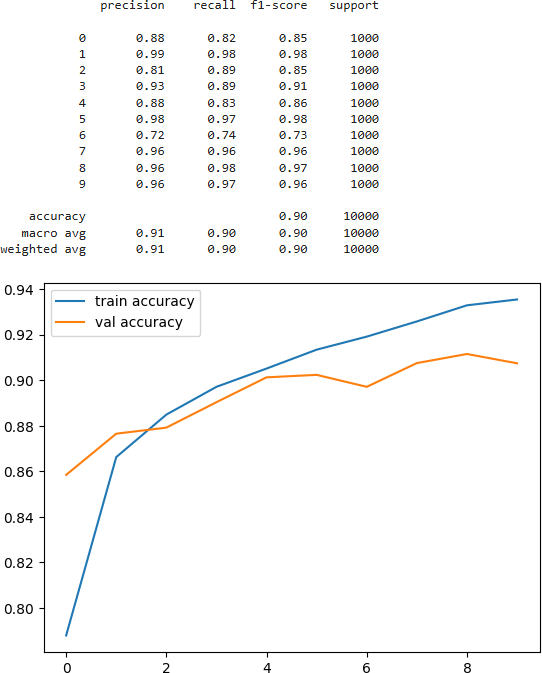
plt.show()

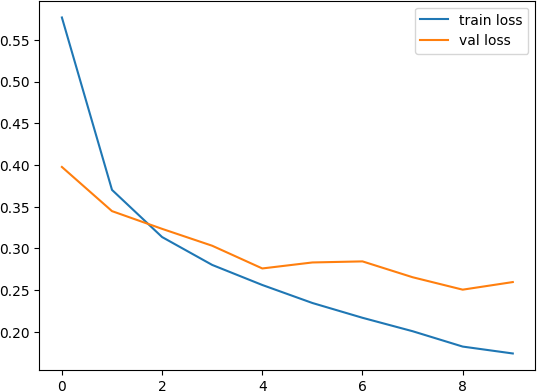
plt.plot(history.history['loss'], label='train loss') plt.plot(history.history['val\_loss'], label='val loss') plt.legend()

plt.show()

# OUTPUT:







**RESULT:**

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