# 1. WORKING OF CNN ARCHITECTURE TO CLASSIFY IMAGES

|  |  |
| --- | --- |
| **EX.N0 : 1** | **WORKING OF CNN ARCHITECTURE TO CLASSIFY IMAGES** |
| **DATE : 21/01/2025** |

**AIM:**

To implement and demonstrate a Convolutional Neural Network (CNN) for image classification using the MNIST dataset in TensorFlow and Keras.

# ALGORITHM:

Step 1: Import necessary libraries.

Step 2: Load and pre-process the MNIST dataset.

Step 3: Build the CNN model using Keras Sequential API.

Step 4: Compile the model with appropriate loss function and optimizer. Step 5: Train the model on the training dataset.

Step 6: Evaluate the model on the test dataset.

Step 7: Visualize a few predictions to validate performance.

# PROGRAM:

import tensorflow as tf

from tensorflow.keras import layers, models from tensorflow.keras.datasets import mnist import matplotlib.pyplot as plt

import numpy as np

# Load MNIST dataset for simplicity

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data() train\_images = train\_images.reshape((60000, 28, 28, 1)).astype('float32') / 255.0

test\_images = test\_images.reshape((10000, 28, 28, 1)).astype('float32') / 255.0

# Create a simple CNN model model = models.Sequential([

layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'), layers.Flatten(),

layers.Dense(64, activation='relu'), layers.Dense(10, activation='softmax')

])

# Compile the model model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

# Train the model

history = model.fit(train\_images, train\_labels, epochs=5, validation\_data=(test\_images, test\_labels))

# Evaluate the model

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels) print(f"Test Accuracy: {test\_acc}")

# Plotting some images along with model predictions

def plot\_images(images, labels, predictions=None, num\_images=5): fig, axes = plt.subplots(1, num\_images, figsize=(12, 3))

for i in range(num\_images):

img = images[i].reshape(28, 28) ax = axes[i]

ax.imshow(img, cmap='gray') if predictions is not None:

ax.set\_title(f"Label: {labels[i]} \nPred: {predictions[i]}")

else:

ax.set\_title(f"Label: {labels[i]}") ax.axis('off')

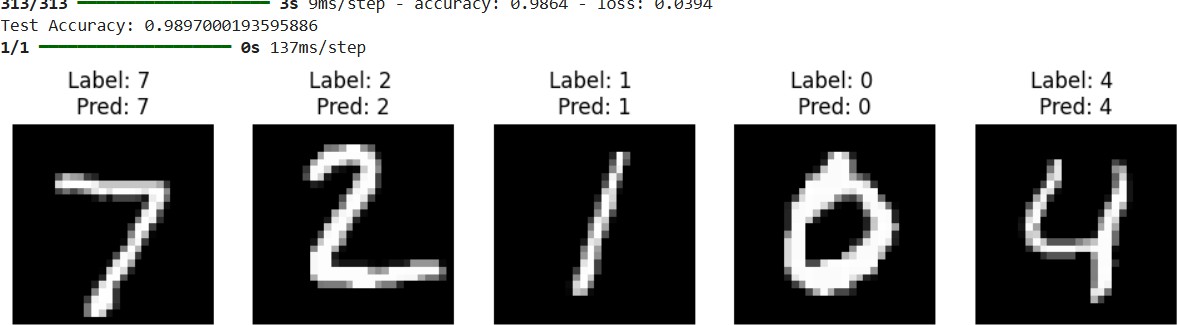
plt.show()

# Get predictions for test images

predictions = np.argmax(model.predict(test\_images[:5]), axis=-1)

# Display some images with their predicted labels plot\_images(test\_images, test\_labels, predictions)

# OUTPUT:

****

**RESULT:**

Thus, to implement and demonstrate a Convolutional Neural Network (CNN) for image classification using the CIFAR-10 dataset in TensorFlow and Keras.