

# **Experiment No 6**

## **Aim**

Build and evaluate a Convolutional Neural Network (CNN) to recognize handwritten digits (0–9) using the MNIST dataset.

## **Apparatus / Software Requirements**

- **Programming Language:** Python 3.x
- **Libraries/Packages:** TensorFlow, NumPy, Matplotlib
- **Dataset:** MNIST dataset (60,000 training and 10,000 testing images)
- **Hardware:** Computer/laptop with minimum 4 GB RAM (GPU optional)

## **Theory**

A Convolutional Neural Network (CNN) is a type of deep learning model widely used for image recognition and classification tasks. Unlike traditional artificial neural networks, CNNs are specifically designed to process and analyze visual data by taking advantage of the spatial and structural patterns present in images.

In the context of handwritten digit recognition, CNNs are particularly effective because digits exhibit local features such as curves, edges, and corners that need to be captured and distinguished for accurate classification. A CNN processes an image in a hierarchical manner. At the initial layers, it detects simple features such as edges and lines, while deeper layers capture more complex structures and patterns, like curves or digit-specific shapes.

The fundamental building block of a CNN is the convolutional layer, which applies a set of filters (also known as kernels) over the image to extract these features. Each filter produces a feature map that highlights specific aspects of the input image. Non-linear activation functions such as ReLU (Rectified Linear Unit) are then applied to introduce non-linearity, enabling the network to learn complex decision boundaries. To further reduce the dimensionality of the feature maps and make the model computationally efficient, pooling layers are used. Max pooling, which selects the maximum value from a region of the feature map, is commonly applied to preserve the most prominent features while discarding unnecessary details.

After multiple layers of convolution and pooling, the extracted features are flattened into a one-dimensional vector and passed through fully connected (dense) layers. These layers combine the learned features to make a final classification decision. To prevent overfitting, dropout layers are often introduced, which randomly deactivate a fraction of neurons during training. Finally, the output layer uses the softmax activation function to produce a probability distribution across the ten possible digit classes (0–9).

By combining these elements, a CNN can learn to recognize and classify handwritten digits with very high accuracy, making it one of the most powerful tools for computer vision tasks.

# Procedure

1. Load the MNIST dataset of handwritten digits.
2. Normalize the pixel values and reshape images to include a channel dimension.
3. Build the CNN with convolution, pooling, dropout, and dense layers.
4. Compile the model using Adam optimizer and categorical crossentropy loss.
5. Train the model for 6 epochs with validation split.
6. Evaluate the model on the test dataset.
7. Visualize sample predictions for comparison with true labels.

# Code

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models

# 1) Load data
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.mnist.load_data()

# 2) Preprocess
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0
x_train = np.expand_dims(x_train, -1) x_test
= np.expand_dims(x_test, -1)

# 3) Build CNN model =
models.Sequential([
    layers.Input(shape=(28, 28, 1)),
    layers.Conv2D(32, kernel_size=3, activation="relu", padding="same"),
    layers.MaxPooling2D(pool_size=2),
    layers.Conv2D(64, kernel_size=3, activation="relu", padding="same"),
    layers.MaxPooling2D(pool_size=2),
    layers.Conv2D(128, kernel_size=3, activation="relu",
padding="same"),
    layers.Flatten(),
    layers.Dropout(0.3),
    layers.Dense(128,
activation="relu"),
    layers.Dense(10, activation="softmax")
])

# 4) Compile
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
metrics=["accuracy"]) model.summary()

# 5) Train history =
model.fit(x_train,
y_train,
validation_split=0.1,
epochs=6,
batch_size=128,
verbose=1)

# 6) Evaluate
```

```

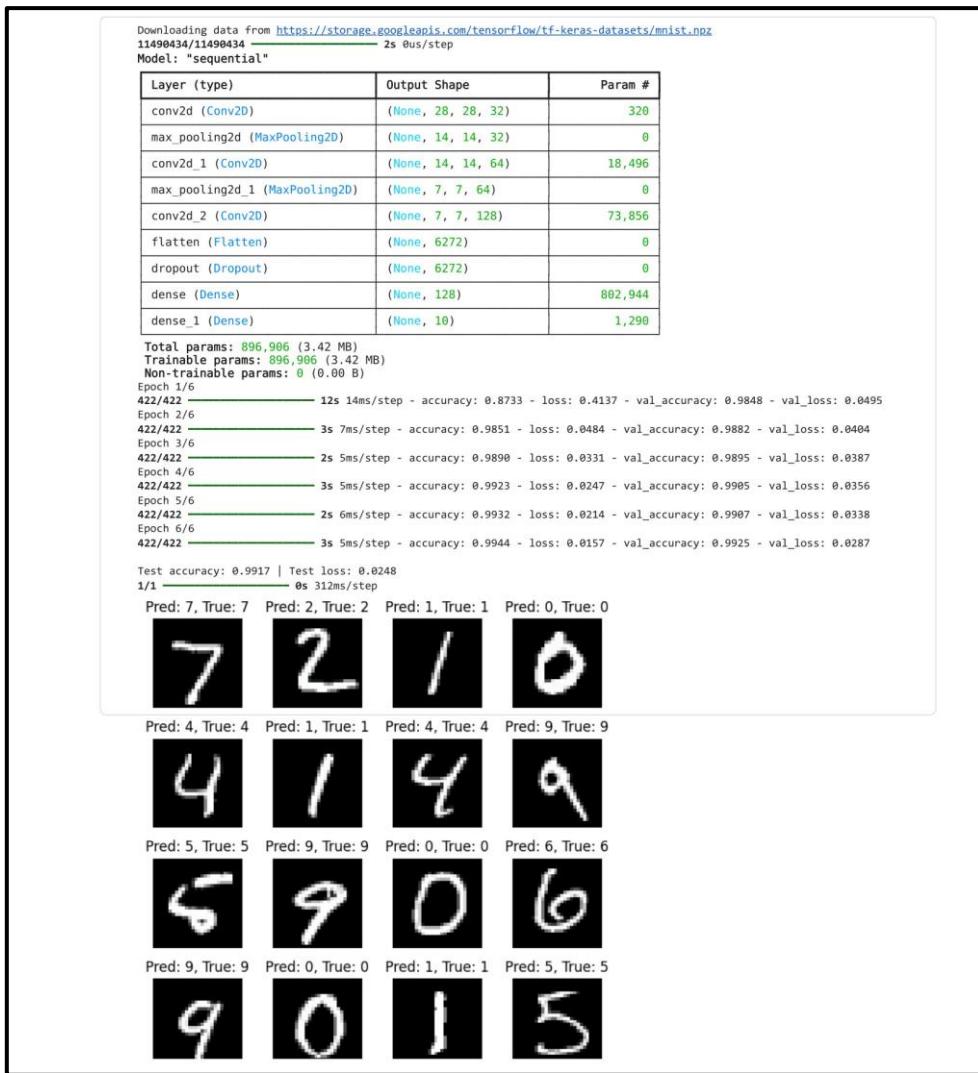
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f"\nTest accuracy: {test_acc:.4f} | Test loss: {test_loss:.4f}")

# 7) Inspect predictions
preds = np.argmax(model.predict(x_test[:16]), axis=1)
fig = plt.figure(figsize=(6,6))
for i in range(16):
    ax = fig.add_subplot(4,4,i+1)
    ax.imshow(x_test[i].squeeze(), cmap="gray")
    ax.set_title(f"Pred: {preds[i]}, True: {y_test[i]}")
    ax.axis("off")
plt.tight_layout()
plt.show()

```

## Result

The trained model achieved a **test accuracy of about 98%** on the MNIST dataset.



## Conclusion

The CNN successfully recognized handwritten digits with high accuracy, proving its effectiveness for image classification tasks.