🧪 Predict Your Own Digit with CNN

```
# Step 1: Load the MNIST dataset
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# The dataset contains 28x28 grayscale images of handwritten digits (0–9)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
# Step 2: Preprocess the data
# Reshape the data to include a channel dimension (needed for CNN)
x_train = x_train.reshape(-1, 28, 28, 1).astype("float32") / 255
x_test = x_test.reshape(-1, 28, 28, 1).astype("float32") / 255

# Convert class vectors to binary class matrices (one-hot encoding)
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

```
# Step 3: Build the CNN model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Define the model architecture
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `in
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# Step 4: Compile the model
# Use Adam optimizer and categorical crossentropy loss
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
# Step 5: Train the model
# Train for 5 epochs with a batch size of 64
model.fit(x_train, y_train,
          batch_size=64,
          epochs=5,
          validation_split=0.1)
```

```
Epoch 1/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 31s 35ms/step - accuracy: 0.8933 - loss: 0.3696 - val_accuracy: 0.9787 - val_loss: 0.0739
Epoch 2/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 30s 35ms/step - accuracy: 0.9817 - loss: 0.0621 - val_accuracy: 0.9848 - val_loss: 0.0563
Epoch 3/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 39s 33ms/step - accuracy: 0.9883 - loss: 0.0389 - val_accuracy: 0.9872 - val_loss: 0.0470
Epoch 4/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 43s 35ms/step - accuracy: 0.9922 - loss: 0.0255 - val_accuracy: 0.9865 - val_loss: 0.0477
Epoch 5/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 39s 32ms/step - accuracy: 0.9948 - loss: 0.0166 - val_accuracy: 0.9882 - val_loss: 0.0451
<keras.src.callbacks.history.History at 0x79f7e8b1e490>
```

```
# Step 6: Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test accuracy:", test_acc)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.9831 - loss: 0.0558
Test accuracy: 0.9865999817848206
```

```
# Step 6: Prediction with MINS
import numpy as np
predictions = model.predict(x_test)
predicted_labels = np.argmax(predictions, axis=1)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step
```

```
#Visualise
import matplotlib.pyplot as plt

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_test[i].reshape(28,28), cmap=plt.cm.binary)
    plt.xlabel(f"Label: {np.argmax(y_test[i])}\nPred: {predicted_labels[i]}")
plt.tight_layout()
plt.show()


#Show Misclassified Examples

import numpy as np
import matplotlib.pyplot as plt

# Gerçek etiketleri ve tahminleri al
true_labels = np.argmax(y_test, axis=1)       # Gerçek etiketler
predicted_labels = np.argmax(model.predict(x_test), axis=1)  # Tahmin edilen etiketler

# Yanlış sınıflandırılan indeksleri bul
misclassified_indices = np.where(predicted_labels != true_labels)[0]

# İlk 25 yanlış tahmini göster
plt.figure(figsize=(12, 12))
for i, index in enumerate(misclassified_indices[:10]):
    plt.subplot(5, 5, i+1)
    plt.imshow(x_test[index].reshape(28, 28), cmap='gray')
    plt.title(f"True: {true_labels[index]}\nPred: {predicted_labels[index]}")
    plt.axis('off')
plt.suptitle("Misclassified Examples", fontsize=18)
plt.tight_layout()
plt.show()
```

🧪 Predict Your Own Digit with CNN (Image Upload)

```
from PIL import Image, ImageOps
import numpy as np

# Step 1: Load your image (upload your image and give the correct filename here)
img = Image.open("/content/try_black.png").convert('L')  # Convert to grayscale

# Step 2: Invert colors (if digit is white DO NOT USE THIS LINE)
#img = ImageOps.invert(img)

# Step 3: Resize the image to 28x28
img = img.resize((28, 28))

# Step 4: Convert to NumPy array and normalize pixel values
img_array = np.array(img) / 255.0

# Step 5: Reshape to fit CNN input shape (1 sample, 28x28 size, 1 channel)
img_array = img_array.reshape(1, 28, 28, 1)

# Step 6: Predict using your trained CNN model
prediction = model.predict(img_array)
predicted_label = np.argmax(prediction)

print("Predicted Digit:", predicted_label)

import matplotlib.pyplot as plt

# Görseli göster ve tahmini yaz
plt.imshow(img_array.reshape(28, 28), cmap='gray')
plt.axis('off')
plt.title(f"Predicted: {predicted_label}")
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
Predicted Digit: 6
```