Upload the Dataset

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files mnist_test.csv
• mnist_test.csv(text/csv) - 18303650 bytes, last modified: 5/13/2025 - 100% done
Saving mnist_test.csv to mnist_test.csv
```

Load the Dataset

```
import pandas as pd

df = pd.read_csv('mnist_test.csv')
df.head()
```

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	...	28x19	28x20	28x21	28x22	28x23	28x24	28x25	28x26	28x27	28x28
0	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

Data Exploration

```
df.info()
df.describe()
df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Columns: 785 entries, label to 28x28
dtypes: int64(785)
memory usage: 59.9 MB
(10000, 785)
```

Check for Missing Values and Duplicates

```
print(df.isnull().sum())
print(f"Duplicate Rows: {df.duplicated().sum()}")
```

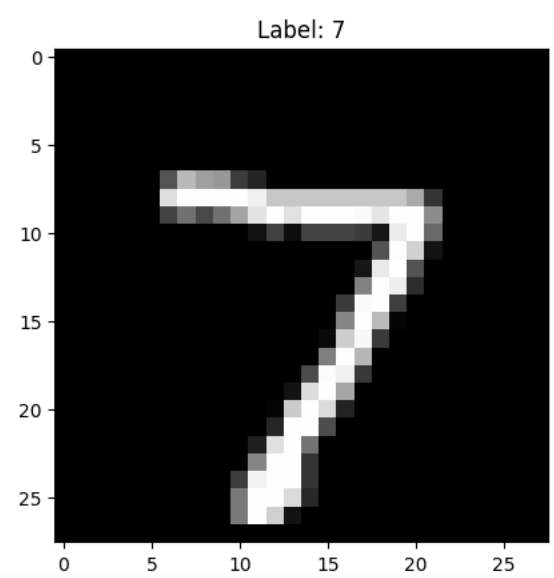
```
label    0
1x1      0
1x2      0
1x3      0
1x4      0
        ..
28x24    0
28x25    0
28x26    0
28x27    0
28x28    0
Length: 785, dtype: int64
Duplicate Rows: 0
```

Visualize a Few Features

```
import matplotlib.pyplot as plt

# Visualize a digit image
def visualize_digit(index):
    image = df.iloc[index, 1:].values.reshape(28, 28)
    label = df.iloc[index, 0]
    plt.title(f"Label: {label}")
    plt.imshow(image, cmap='gray')
    plt.show()
```

```
visualize_digit(0)
```



Identify Target and Features

```
X = df.drop('label', axis=1)
y = df['label']
```

One-Hot Encoding

```
from tensorflow.keras.utils import to_categorical

y_encoded = to_categorical(y)
```

Feature Scaling

```
X = X / 255.0
```

Train-Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

Model Building

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten

model = Sequential([
    Flatten(input_shape=(28*28,)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_c
  super().__init__(**kwargs)
```

Evaluation

```
model.fit(X_train, y_train, epochs=5, validation_data=(X_test, y_test))
```

```
Epoch 1/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - accuracy: 0.7331 - loss: 0.9401 - val_accuracy: 0.9105 - val_loss: 0.3167
Epoch 2/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9262 - loss: 0.2579 - val_accuracy: 0.9300 - val_loss: 0.2457
Epoch 3/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9513 - loss: 0.1773 - val_accuracy: 0.9400 - val_loss: 0.2091
Epoch 4/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - accuracy: 0.9637 - loss: 0.1236 - val_accuracy: 0.9320 - val_loss: 0.2248
Epoch 5/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.9720 - loss: 0.1028 - val_accuracy: 0.9460 - val_loss: 0.1804
<keras.src.callbacks.history.History at 0x7964571abf10>
```

Make Predictions from New Input

```
predictions = model.predict(X_test)
predicted_labels = predictions.argmax(axis=1)
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

Convert to DataFrame and Encode

```
import numpy as np

result_df = pd.DataFrame({
    'True Label': y_test.argmax(axis=1),
    'Predicted Label': predicted_labels
})
result_df.head()
```

	True Label	Predicted Label
0	6	6
1	2	2
2	3	3
3	7	7
4	2	2

Next steps: Generate code with result_df View recommended plots New interactive sheet

No results

Deployment - Building an Interactive App

```
!pip install gradio
import gradio as gr
```

```
'/local/lib/python3.11/dist-packages (from gradio) (11.2.1)
isr/local/lib/python3.11/dist-packages (from gradio) (2.11.4)

 (1.4 kB)

ietadata (1.8 kB)
```

```
.1b/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0.16.0)
.b/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
.b/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
ıl/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.67.1)
'usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (1.1.0)
ı /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2.9.0.post0)
ıl/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
ɔcal/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
ı /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (0.7.0)
 /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (2.33.2)
 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (0.4.0)
ıl/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.8)
ɪr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4)
ɔal/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
.b/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->gradio) (1.17.0)
 /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0.12->gradio) (3.0.0)
.n /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0.12->gradio) (2.19.1)
 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.28.1->gradio) (3.4.2)
ɪr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.28.1->gradio) (2.4.0)
'lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->gradio) (0.1.2)

ł 12.0 MB/s eta 0:00:00
:B)
 kB 15.9 MB/s eta 0:00:00


ł 4.9 MB/s eta 0:00:00

ł kB)
ıanylinux2014_x86_64.whl (11.5 MB)
```

reate a Prediction Function

```
def predict_digit(image):
    import numpy as np
    image = image.reshape(1, 28*28) / 255.0
    pred = model.predict(image)
    return pred.argmax()
```

Create the Gradio Interface

```
import numpy as np
import gradio as gr

# Make sure your model is already trained and assigned to cnn_model
# Example: cnn_model = Sequential([...])
# cnn_model.compile(...)
# cnn_model.fit(...)

def predict_digit(img):
    try:
        # Convert PIL image to numpy array
        img = np.array(img)

        # Resize to 28x28 if needed
        if img.shape != (28, 28):
            from PIL import Image
            img = Image.fromarray(img).resize((28, 28))
            img = np.array(img)

        # Invert colors (white digit on black background like MNIST)
        img = 255 - img

        # Normalize
        img = img / 255.0

        # Reshape for CNN input
        img = img.reshape(1, 28, 28, 1)

        # Predict
        prediction = cnn_model.predict(img)
        return str(np.argmax(prediction))

    except Exception as e:
        return f"Error: {str(e)}"
```

Handwritten Digits Recognition using CNN

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Reshape

X_cnn = X.values.reshape(-1, 28, 28, 1)

cnn_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')
])

cnn_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
cnn_model.fit(X_cnn, y_encoded, epochs=5, validation_split=0.2)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 9s 28ms/step - accuracy: 0.7706 - loss: 0.7737 - val_accuracy: 0.9540 - val_loss: 0.1410
Epoch 2/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 6s 23ms/step - accuracy: 0.9531 - loss: 0.1462 - val_accuracy: 0.9675 - val_loss: 0.1038
Epoch 3/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 11s 25ms/step - accuracy: 0.9717 - loss: 0.0878 - val_accuracy: 0.9670 - val_loss: 0.0934
Epoch 4/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 22ms/step - accuracy: 0.9836 - loss: 0.0532 - val_accuracy: 0.9785 - val_loss: 0.0774
Epoch 5/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 10s 23ms/step - accuracy: 0.9929 - loss: 0.0273 - val_accuracy: 0.9825 - val_loss: 0.0664
<keras.src.callbacks.history.History at 0x796442713910>
```

```
predict_digit
```

```
predict_digit
def predict_digit(img)

<no docstring>
```