```
# Import required libraries
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Print shapes
print("Train images:", x_train.shape)
print("Test images:", x_test.shape)
```

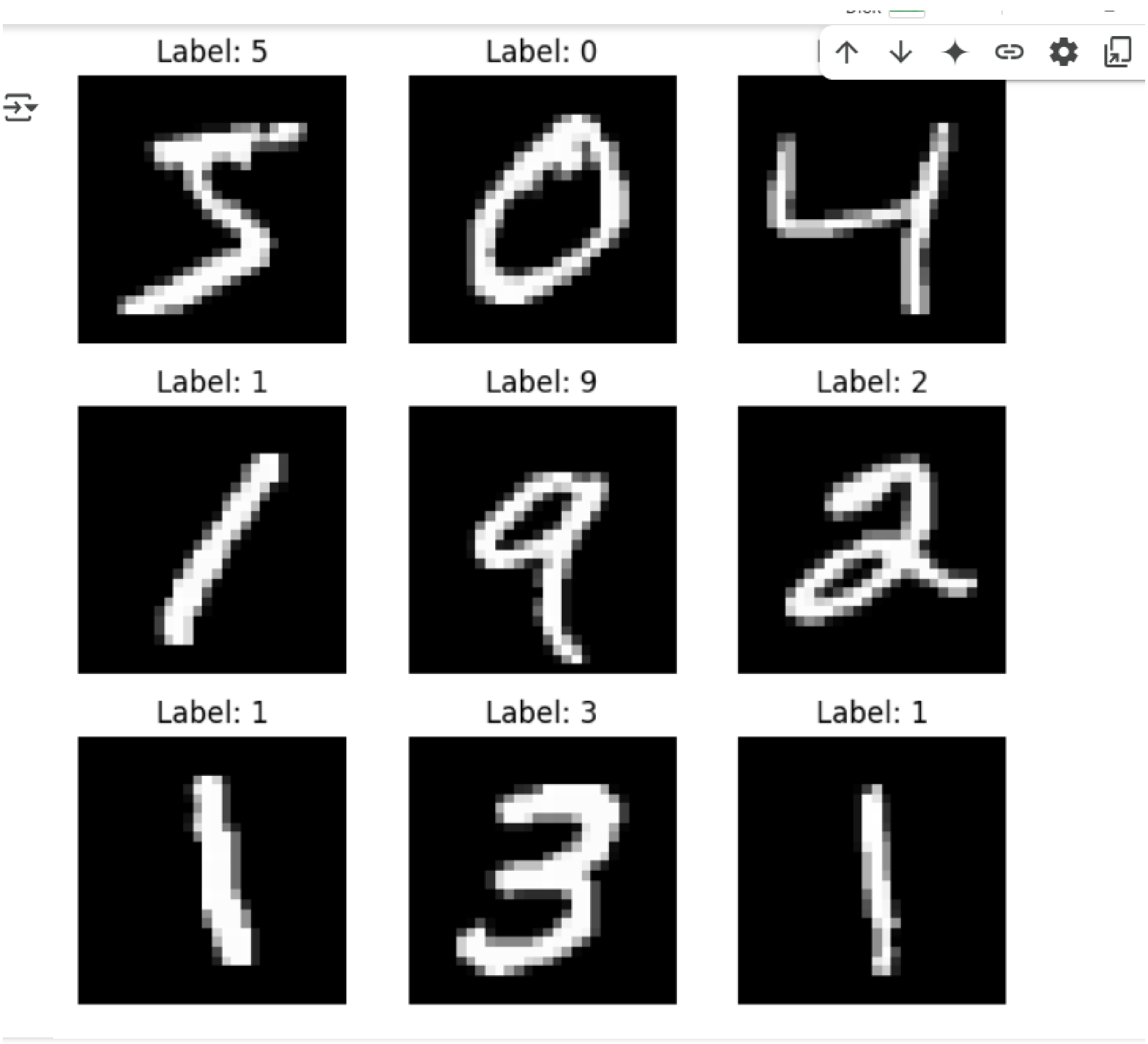
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Train images: (60000, 28, 28)
Test images: (10000, 28, 28)
```

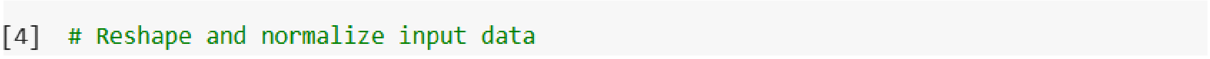
---

[2]

0s

```
# Show first 9 sample images
plt.figure(figsize=(6, 6))
for i in range(9):
   plt.subplot(3, 3, i + 1)
   plt.imshow(x_train[i], cmap='gray')
   plt.title(f"Label: {y_train[i]}")
   plt.axis('off')
plt.tight_layout()
plt.show()
```





```
[4]  # Reshape and normalize input data
```

```
 Reshape and normalize input data
x_train = x_train.reshape(-1, 28, 28, 1).astype("float32") / 255
x_test = x_test.reshape(-1, 28, 28, 1).astype("float32") / 255

# One-hot encode labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

print("Data normalized and labels encoded.")
```

```
Data normalized and labels encoded.
```

```
[5]
0s
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
# Create the model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(10, activation='softmax')
])
# Compile model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

---

[7]
0s

```
# Reshape and normalize input data
x_train = x_train.reshape(-1, 28, 28, 1).astype("float32") / 255
x_test = x_test.reshape(-1, 28, 28, 1).astype("float32") / 255
# One-hot encode labels if they are not already in categorical format
if len(y_train.shape) == 1:  # Check if y_train is not already one-hot encoded
    y_train = to_categorical(y_train, 10)
    y_test = to_categorical(y_test, 10)
print("Data normalized and labels encoded.")
```

```
Data normalized and labels encoded.
```

---

[13]
error
0s

```
# Plot accuracy and loss curves
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label="Train Accuracy")
plt.plot(history.history['val_accuracy'], label="Validation Accuracy")
plt.title("Model Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label="Train Loss")
plt.plot(history.history['val_loss'], label="Validation Loss")
plt.title("Model Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.tight_layout()NameError                         Traceback (most recent call last)
<ipython-input-13-4eb34c6d27c1> in <cell line: 0>()
    3
    4 plt.subplot(1, 2, 1)
----> 5 plt.plot(history.history['accuracy'], label="Train Accuracy")
    6 plt.plot(history.history['val_accuracy'], label="Validation Accuracy")
    7 plt.title("Model Accuracy")

plt.show()
```