In [2]:
```
pip install tensorflow
```

```
Collecting tensorflow
  Obtaining dependency information for tensorflow from https://files.python
hosted.org/packages/e4/14/d795bb156f8cc10eb1dcfe1332b7dbb8405b634688980aa9b
e8f885cc888/tensorflow-2.16.1-cp311-cp311-win_amd64.whl.metadata (https://f
iles.pythonhosted.org/packages/e4/14/d795bb156f8cc10eb1dcfe1332b7dbb8405b63
4688980aa9be8f885cc888/tensorflow-2.16.1-cp311-cp311-win_amd64.whl.metadat
a)
  Downloading tensorflow-2.16.1-cp311-cp311-win_amd64.whl.metadata (3.5 kB)
Collecting tensorflow-intel==2.16.1 (from tensorflow)
  Obtaining dependency information for tensorflow-intel==2.16.1 from http
s://files.pythonhosted.org/packages/e0/36/6278e4e7e69a90c00e0f82944d8f2713d
d85a69d1add455d9e50446837ab/tensorflow_intel-2.16.1-cp311-cp311-win_amd64.w
hl.metadata (https://files.pythonhosted.org/packages/e0/36/6278e4e7e69a90c0
0e0f82944d8f2713dd85a69d1add455d9e50446837ab/tensorflow_intel-2.16.1-cp311-
cp311-win_amd64.whl.metadata)
  Downloading tensorflow_intel-2.16.1-cp311-cp311-win_amd64.whl.metadata
(5.0 kB)
Collecting absl-py>=1.0.0 (from tensorflow-intel==2.16.1->tensorflow)
  Obtaining dependency information for absl-py>=1.0.0 from https://files.py
```

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
```

In [7]:
```
# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [8]:
```
# Preprocess the data
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1).astype('float32') / 255
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1).astype('float32') / 255
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

In [9]:
```
# Define the model
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')
])
```

```
D:\anaconda\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:10
7: 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 fi
rst layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [10]:
```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accu
```

In [11]:
```
# Train the model
model.fit(x_train, y_train, batch_size=128, epochs=5, validation_data=(x_test,
```

```
Epoch 1/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 19s 37ms/step - accuracy: 0.8805 - loss: 0.4299
- val_accuracy: 0.9753 - val_loss: 0.0784
Epoch 2/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 38ms/step - accuracy: 0.9794 - loss: 0.0714
- val_accuracy: 0.9812 - val_loss: 0.0549
Epoch 3/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 18s 38ms/step - accuracy: 0.9875 - loss: 0.0425
- val_accuracy: 0.9831 - val_loss: 0.0514
Epoch 4/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 36ms/step - accuracy: 0.9906 - loss: 0.0299
- val_accuracy: 0.9823 - val_loss: 0.0503
Epoch 5/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 17s 36ms/step - accuracy: 0.9935 - loss: 0.0232
- val_accuracy: 0.9852 - val_loss: 0.0420
```

Out[11]:
```
<keras.src.callbacks.history.History at 0x208e1979850>
```

In [12]:
```
# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc}')
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9807 - loss: 0.0520
Test accuracy: 0.9851999878883362
```

In [13]:
```
# Predict labels for test images
predictions = model.predict(x_test)
predicted_labels = np.argmax(predictions, axis=1)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
```

In [14]:
```
# Display images along with predicted and actual labels
plt.figure(figsize=(15, 15))
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)
    predicted_label = predicted_labels[i]
    true_label = np.argmax(y_test[i])
    if predicted_label == true_label:
        color = 'green'
        title = f'Predicted: {predicted_label}\nActual: {true_label}'
    else:
        color = 'red'
        title = f'Predicted: {predicted_label}\nActual: {true_label}'
    plt.xlabel(title, color=color)
plt.show()
```