Neural Networks

DSC 550 - Week Eleven

McKenzie Payne

In this exercise, you will build a convolutional neural network (CNN) to classify handwritten digits from the MNIST dataset. The steps to build a CNN classifier are outlined in section 20.15 of the Machine Learning with Python Cookbook, but keep in mind that your code may need to be modified depending on your version of Keras.

Step 1. Load the MNIST data set.

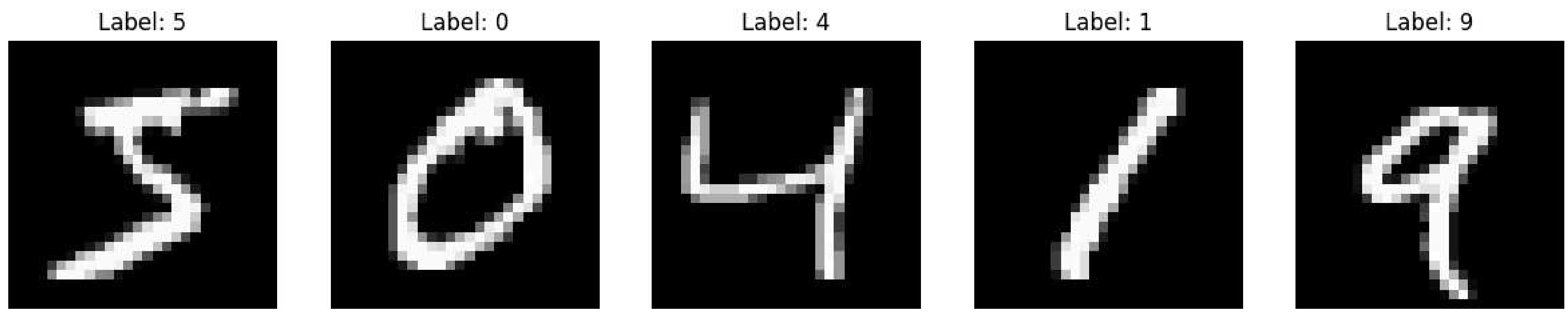
In [1]:
```
from tensorflow.keras.datasets import mnist

# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

Step 2. Display the first five images in the training data set (see section 8.1 in the Machine Learning with Python Cookbook). Compare these to the first five training labels.

In [2]:
```
import matplotlib.pyplot as plt

# Display the first five images and labels
fig, axes = plt.subplots(1, 5, figsize=(15, 3))
for i, ax in enumerate(axes):
    ax.imshow(train_images[i], cmap='gray')
    ax.set_title(f"Label: {train_labels[i]}")
    ax.axis('off')
plt.show()
```



Step 3. Build and train a Keras CNN classifier on the MNIST training set.

In [3]:

```
from tensorflow.keras import models, layers

# Reshape and normalize the input data
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

# Define the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(train_images, train_labels, epochs=5, batch_size=64, validation_split=0.2)
```

```
Epoch 1/5
```

```
C:\Users\mcken\anaconda3\Lib\site-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)
750/750 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - accuracy: 0.8574 - loss: 0.4806 - val_accuracy: 0.9794 - val_loss: 0.0741
Epoch 2/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 8s 11ms/step - accuracy: 0.9826 - loss: 0.0597 - val_accuracy: 0.9846 - val_loss: 0.0522
Epoch 3/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - accuracy: 0.9871 - loss: 0.0409 - val_accuracy: 0.9853 - val_loss: 0.0466
Epoch 4/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 8s 11ms/step - accuracy: 0.9910 - loss: 0.0291 - val_accuracy: 0.9861 - val_loss: 0.0488
Epoch 5/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 9s 12ms/step - accuracy: 0.9926 - loss: 0.0221 - val_accuracy: 0.9860 - val_loss: 0.0481
```

Step 4. Report the test accuracy of your model.

In [4]:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f'Test accuracy: {test_acc}')
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9870 - loss: 0.0489
Test accuracy: 0.9886999726295471
```

Step 5. Display a confusion matrix on the test set classifications.

In [5]:

```
from sklearn.metrics import confusion_matrix
import numpy as np

# Get the model's predictions on the test set
predictions = np.argmax(model.predict(test_images), axis=-1)

# Generate and display the confusion matrix
conf_matrix = confusion_matrix(test_labels, predictions)
print(conf_matrix)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
[[ 974    1    0    0    0    3    0    1    1    0]
 [   0 1133    0    0    0    1    1    0    0    0]
 [   4    2 1018    0    0    0    0    3    5    0]
 [   1    0    2  991    0    9    0    0    7    0]
 [   0    0    0    0  982    0    0    0    0    0]
 [   1    0    0    2    0  887    2    0    0    0]
 [   3    2    0    0    4    1  946    0    2    0]
 [   0    5    3    2    3    0    0 1009    2    4]
 [   2    0    0    0    0    1    0    0  970    1]
 [   3    1    0    0   12    8    0    1    7  977]]
```

Step 6. Summarize your results.

> Overall, the CNN model achieved high accuracy on the MNIST dataset, demonstrating its effectiveness in classifying handwritten digits. The confusion matrix allowed us to identify specific areas where the model could be improved, such as distinguishing between similar-looking digits.

In [ ]:

```

```