In [ ]:

```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow.keras.utils as tku
```

In [ ]:

```
tf.__version__
```

Out[ ]:

```
'2.6.0'
```

In [ ]:

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
training_set = train_datagen.flow_from_directory('/home/jovyan/binder/MNIST/training_set'
,
                                                 target_size = (28, 28),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 37340 images belonging to 10 classes.
```

In [ ]:

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_set = train_datagen.flow_from_directory('/home/jovyan/binder/MNIST/test_set',
                                             target_size = (28, 28),
                                             batch_size = 32,
                                             class_mode = 'categorical')
```

```
Found 4660 images belonging to 10 classes.
```

In [ ]:

```
cnn = tf.keras.models.Sequential()
```

In [ ]:

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape
=[28, 28, 3]))
```

In [ ]:

```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

In [ ]:

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

In [ ]:

```
cnn.add(tf.keras.layers.Flatten())
```

In [ ]:

```
cnn.add(tf.keras.layers.Dense(units=64, activation='relu'))
```

In [ ]:

```
cnn.add(tf.keras.layers.Dense(units=10, activation='softmax'))
```

In [ ]:

```
cnn.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'
])
```

In [ ]:

```
trained_model = cnn.fit(x = training_set, validation_data = test_set, epochs = 20)
```

```
Epoch 1/20
1167/1167 [==============================] - 96s 81ms/step - loss: 0.3725 - accuracy: 0.8
785 - val_loss: 0.1686 - val_accuracy: 0.9464
Epoch 2/20
1167/1167 [==============================] - 95s 81ms/step - loss: 0.1359 - accuracy: 0.9
580 - val_loss: 0.1141 - val_accuracy: 0.9635
Epoch 3/20
1167/1167 [==============================] - 95s 81ms/step - loss: 0.1043 - accuracy: 0.9
677 - val_loss: 0.0823 - val_accuracy: 0.9738
Epoch 4/20
1167/1167 [==============================] - 103s 88ms/step - loss: 0.0865 - accuracy: 0.
9725 - val_loss: 0.0760 - val_accuracy: 0.9770
Epoch 5/20
1167/1167 [==============================] - 98s 84ms/step - loss: 0.0770 - accuracy: 0.9
755 - val_loss: 0.0635 - val_accuracy: 0.9796
Epoch 6/20
1167/1167 [==============================] - 98s 84ms/step - loss: 0.0631 - accuracy: 0.9
798 - val_loss: 0.0673 - val_accuracy: 0.9775
Epoch 7/20
1167/1167 [==============================] - 103s 88ms/step - loss: 0.0623 - accuracy: 0.
9804 - val_loss: 0.0561 - val_accuracy: 0.9824
Epoch 8/20
1167/1167 [==============================] - 101s 87ms/step - loss: 0.0555 - accuracy: 0.
9825 - val_loss: 0.0667 - val_accuracy: 0.9796
Epoch 9/20
1167/1167 [==============================] - 100s 86ms/step - loss: 0.0520 - accuracy: 0.
9834 - val_loss: 0.0658 - val_accuracy: 0.9792
Epoch 10/20
1167/1167 [==============================] - 163s 140ms/step - loss: 0.0471 - accuracy: 0
.9849 - val_loss: 0.0592 - val_accuracy: 0.9843
Epoch 11/20
1167/1167 [==============================] - 94s 81ms/step - loss: 0.0465 - accuracy: 0.9
852 - val_loss: 0.0640 - val_accuracy: 0.9788
Epoch 12/20
1167/1167 [==============================] - 98s 84ms/step - loss: 0.0424 - accuracy: 0.9
864 - val_loss: 0.0599 - val_accuracy: 0.9822
Epoch 13/20
1167/1167 [==============================] - 92s 79ms/step - loss: 0.0387 - accuracy: 0.9
877 - val_loss: 0.0606 - val_accuracy: 0.9828
Epoch 14/20
1167/1167 [==============================] - 92s 79ms/step - loss: 0.0377 - accuracy: 0.9
875 - val_loss: 0.0496 - val_accuracy: 0.9837
Epoch 15/20
1167/1167 [==============================] - 92s 79ms/step - loss: 0.0345 - accuracy: 0.9
885 - val_loss: 0.0628 - val_accuracy: 0.9792
Epoch 16/20
1167/1167 [==============================] - 94s 80ms/step - loss: 0.0347 - accuracy: 0.9
881 - val_loss: 0.0570 - val_accuracy: 0.9809
Epoch 17/20
1167/1167 [==============================] - 92s 79ms/step - loss: 0.0339 - accuracy: 0.9
890 - val_loss: 0.0510 - val_accuracy: 0.9845
Epoch 18/20
1167/1167 [==============================] - 91s 78ms/step - loss: 0.0296 - accuracy: 0.9
```

```
910 - val_loss: 0.0528 - val_accuracy: 0.9843
Epoch 19/20
1167/1167 [==============================] - 91s 78ms/step - loss: 0.0317 - accuracy: 0.9
895 - val_loss: 0.0541 - val_accuracy: 0.9841
Epoch 20/20
1167/1167 [==============================] - 92s 79ms/step - loss: 0.0289 - accuracy: 0.9
903 - val_loss: 0.0630 - val_accuracy: 0.9830
```

In [ ]:

```
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('/home/jovyan/binder/MNIST/single_prediction/img_11.jpg', tar
get_size = (28, 28))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
results = (cnn.predict(test_image))
training_set.class_indices
if results[0][0] == 1:
  prediction = 'Zero'
elif results[0][1] == 1:
  prediction = 'One'
elif results[0][2] == 1:
  prediction = 'Two'
elif results[0][3] == 1:
  prediction = 'Three'
elif results[0][4] == 1:
  prediction = 'Four'
elif results[0][5] == 1:
  prediction = 'Five'
elif results[0][6] == 1:
  prediction = 'Six'
elif results[0][7] == 1:
  prediction = 'Seven'
elif results[0][8] == 1:
  prediction = 'Eight'
else:
  prediction = 'Nine'
```

In [ ]:

```
print(results)
```

```
[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]
```

In [ ]:

```
print(prediction)
```

```
Five
```

In [ ]: