labten

September 13, 2023

[ ]:
```
import numpy as np
import cv2
import tensorflow as tf
import matplotlib.pyplot as plt
```

[ ]:
```
# Load the MNIST dataset
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize pixel values to␣
 ↪be between 0 and 1
# Reshape the data to have a single channel (grayscale)
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
```

[ ]:
```
# Create a CNN model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28,␣
 ↪1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

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

[ ]:
```
# Train the model
model.fit(x_train, y_train, epochs=3,batch_size=100)

# Save the model
#model.save('handwritten_cnn.model')

# Load the model
#model = tf.keras.models.load_model('handwritten_cnn.model')
```

```
Epoch 1/3
2023-09-13 22:43:41.539596: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83]
Allocation of 188160000 exceeds 10% of free system memory.
600/600 [==============================] - 28s 46ms/step - loss: 0.0168 -
accuracy: 0.9950
Epoch 2/3
600/600 [==============================] - 27s 45ms/step - loss: 0.0099 -
accuracy: 0.9970
Epoch 3/3
600/600 [==============================] - 27s 45ms/step - loss: 0.0055 -
accuracy: 0.9984
```

[ ]:
```
<keras.src.callbacks.History at 0x7f77e6646450>
```

[ ]:
```python
# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print("Test loss:", loss)
print("Test accuracy:", accuracy)
```

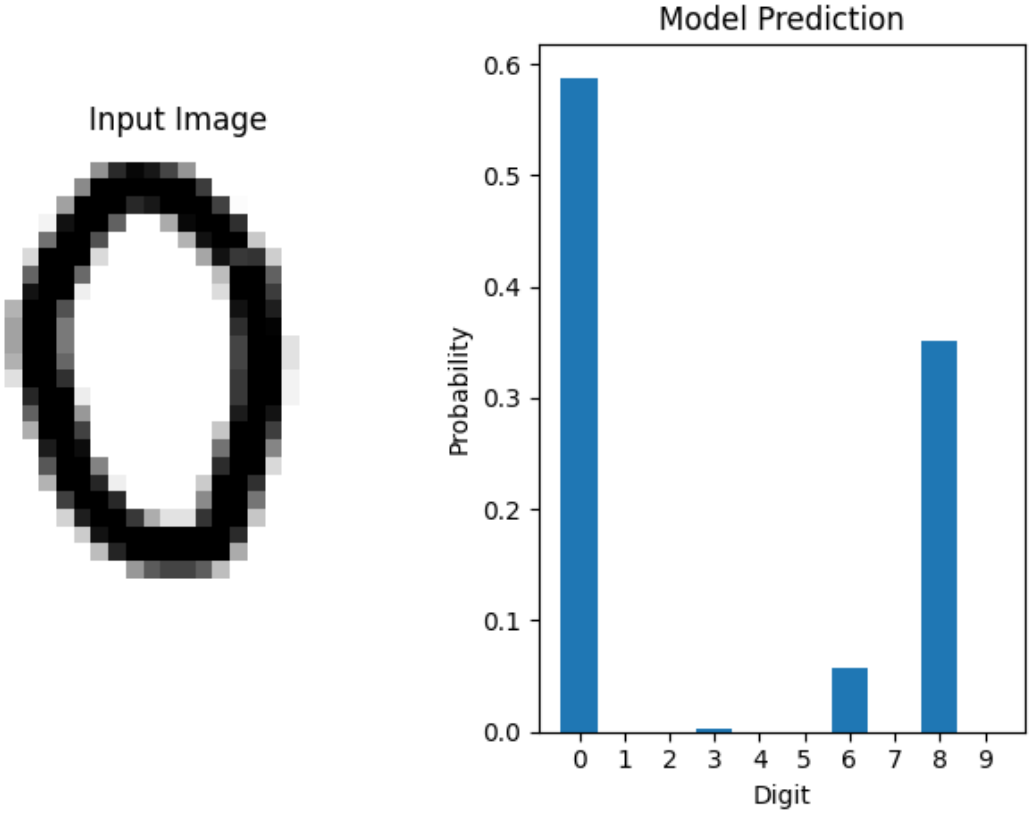
```
313/313 [==============================] - 1s 4ms/step - loss: 0.0459 -
accuracy: 0.9866
Test loss: 0.04588093236088753
Test accuracy: 0.9865999817848206
```

[ ]:
```python
# Load and preprocess the image for prediction
image_path = "9.png"
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
img = cv2.resize(img, (28, 28))
img = img / 255.0  # Normalize pixel values
img = img.reshape(1, 28, 28, 1)
```

[ ]:
```python
# Make a prediction
prediction = model.predict(img)
predicted_digit = np.argmax(prediction)
print(f"This digit is probably a {predicted_digit}")


# Display the original image
plt.subplot(1, 2, 1)
plt.imshow(img.reshape(28, 28), cmap='gray', vmin=0, vmax=1)
plt.title('Input Image')
plt.axis('off')

# Display the model's prediction probabilities
plt.subplot(1, 2, 2)
plt.bar(range(10), prediction[0])
plt.xticks(range(10))
```

```
plt.title('Model Prediction')
plt.xlabel('Digit')
plt.ylabel('Probability')

plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 0s 56ms/step
This digit is probably a 0
```



[ ]: