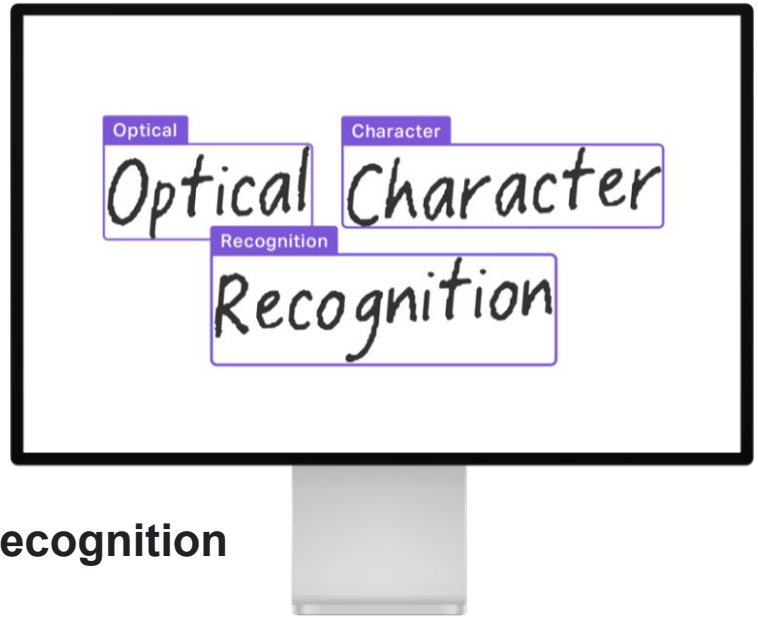
Text recognition (Numbers)

What is Text Recognition?

- Electronic conversion of images
- From Typed / Handwritten / Printed text
- Or Photos of a document
- To Machine-encoded text



https://en.wikipedia.org/wiki/Optical_character_recognition

Our basic idea of the project

Give the program data from a dataset ->

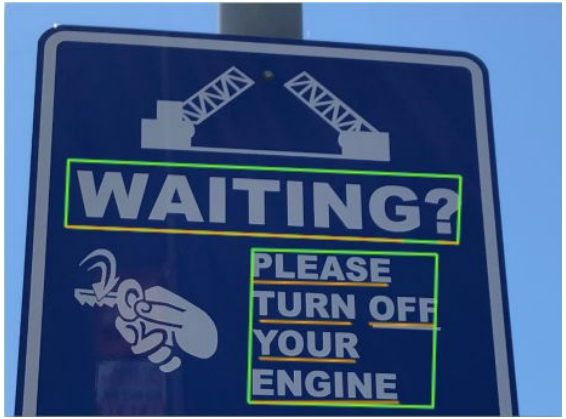
Make neural network model (Training) ->

The program gets a photo ->

Output a single number detected

Why are we using OCR as our project?

- Provides high accuracy (Reduce errors)
- Has fast speed (Higher productivity)
- Easy, convenient to use
- Our product can be used as barcode scanner



Example: Google Cloud OCR

Vision API: detect and extract text from images

What is artificial neural network?

Composed of artificial neurons or nodes

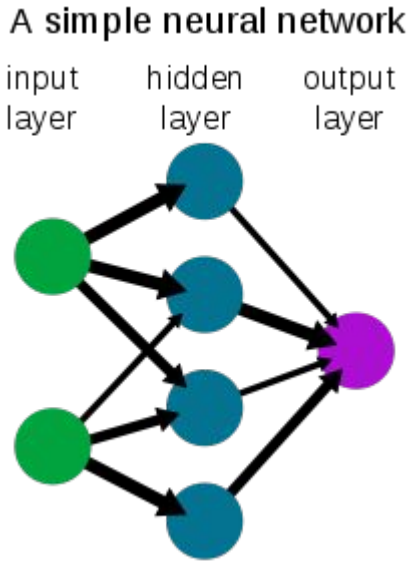
Trained by processing examples (Input & Result)

Probability-weighted associations between the two

"Learn" themselves to perform tasks

(Without being programmed with task-specific rules)

Applications: Image / Text / Sound recognition etc.



Code explanation (Pt 1)

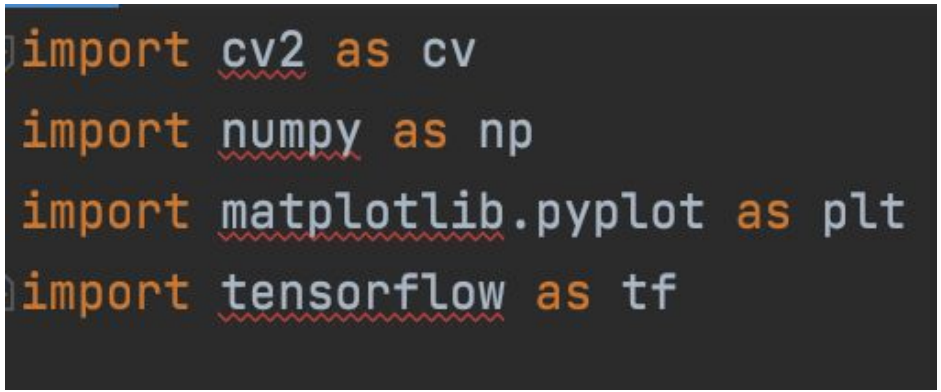
Import Open-source Libraries for the program

Cv2: Import own images as input & result data

NumPy (Numerical Python): scientific computing and working with arrays

Matplotlib: makes some change to a figure (Visualisation)

Tensorflow: machine learning and artificial intelligence



Additional: What is TensorFlow (and its benefits)?

Can be used across a range of tasks

Particular focus on training and inference of deep neural networks.

Developed by the Google Brain team for internal Google use

A wide variety of programming languages

Most notably Python, as well as Javascript, C++, and Java.

A range of applications in many different sectors.

Code Explanation (Pt 2)

Keras: A Python interface for the TensorFlow library

Create a variable (dtset) that store the MNIST dataset

Get & Load data from the dataset

Split them into “training’ & “testing” (Tuples)

60000 digital photo samples of real handwritten numbers

Normalisation: Scaling data to values of 0s & 1s

Ratios & Percentages

Easier to compute in next steps

```
dtset = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = dtset.load_data()
x_train = tf.keras.utils.normalize(x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis = 1)
```

Code Explanation (Part 3)

Create a basic sequential model

Ordinary feed-forward neural network

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape = (28, 28)))
model.add(tf.keras.layers.Dense(units = 128, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(units = 64, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(units = 32, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(units = 10, activation = tf.nn.softmax))
```

Adding layers:

Flattened layer (One-dimensional) (Input):

28 x 28 -> pixels of our input images (shape)

Hidden layer (Four dense layers):

Neurons connected to previous & next one

Units = 128: number of neurons

Activation relu: Applies the rectified linear unit activation fun

Code Explanation (Part 4)

Output layer: (adding 1 dense layer)

Activation softmax: Take all outputs & calculate probability

Compiling model:

Configures the model for training

Optimizer: String (name of optimizer) or optimizer instance

Loss: Compute quantity, seek to minimize during training

Metrics: evaluated by the model during training and testing

```
model.add(tf.keras.layers.Dense(units = 10, activation = tf.nn.softmax))

model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
model.fit(x_train, y_train, epochs = 4)
```

Code Explanation (Part 4)

```
model.fit(x_train, y_train, epochs = 4)

loss, accuracy = model.evaluate(x_test, y_test)
print(accuracy)
print(loss)

model.save('digits.model')
```

Model.fit: Train the model

Epochs: No. of times the model repeat the process

Model.evaluate: Test the model with the data stored before

Model.save: save the model

Code Explanation (Part 5)

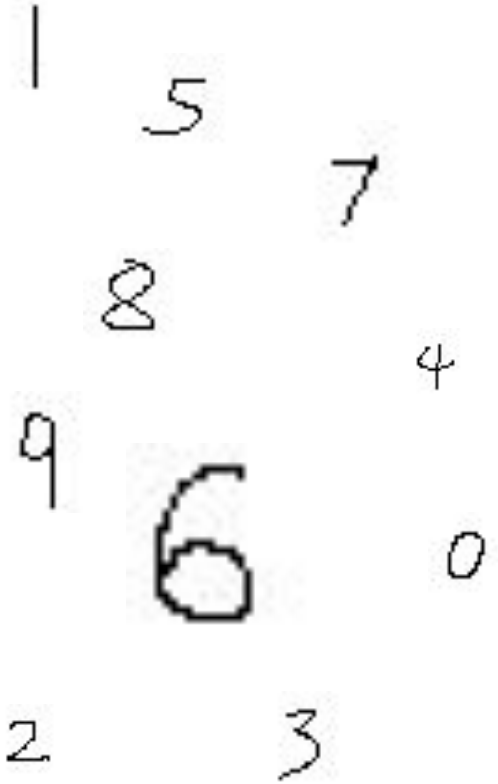
Loads an image from the specified file (size: 28x28 pixels)

Let the model try to predict which number is in the photo

Print the result, also show it through a map drawn (plt)

```
for i in range(0, 10):
    img = cv.imread(f'{i}.png')[:,:,0] # Give image file name
    img = np.invert(np.array([img]))
    prediction = model.predict(img)
    print(f'The result is probably: {np.argmax(prediction)}')

    plt.imshow(img[0], cmap = plt.cm.binary)
    plt.show()
```



Major references:

https://youtu.be/Zi4i7Q0zrBs

https://ithelp.ithome.com.tw/aticles/10191404