國立清華大學
NATIONAL TSING HUA UNIVERSITY

AIMS - deep learning

Application of CNNs – Drawing Recognition

TA course 03

TAs: 蘇時頤, 廖柄淦

- Course website: https://github.com/matteosoo/aimsfellows_DL

Catalog

- Drawing recognition
  - Quick Draw dataset
  - Data preprocessing
  - Build model (CNNs)
  - Training model
  - Validation model
- TensorFlowJS
  - Model convert
  - Interface
    1. Apache server deployment
    2. IDEs with PyCharm

- Drawing recognition

The Quick, Draw! Dataset

The Quick Draw Dataset is a collection of 50 million drawings across 345 categories, contributed by players of the game Quick, Draw!.

- Data preprocessing

- Numpy bitmaps (.npy)
  - All the simplified drawings have been rendered into a 28x28 grayscale bitmap in numpy .npy format.
  - The files can be loaded with np.load().
- Reshape()
  - In this project, they used 28x28 as the input size.
- ☑[TODO]
  - Convert the data(png/jpg) you crawl to the appropriate format (.npy).

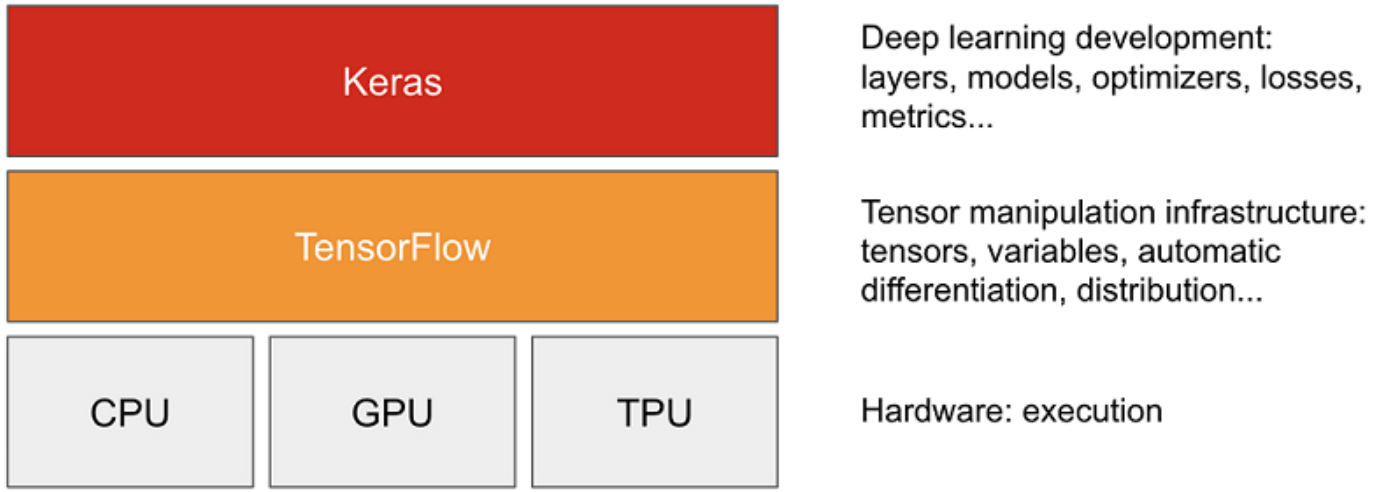
- Summary the model architecture

- The below architecture is the detail of the model.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               73856
_________________________________________________________________
dense_2 (Dense)              (None, 100)               12900
=================================================================
Total params: 110,052
Trainable params: 110,052
Non-trainable params: 0
_________________________________________________________________
None
```

- Keras

- **Sequential** class
  - Sequential provides training and inference features on this model.
  - `>>> tf.keras.Sequential(layers=None, name=None)`



- **add** method
  - Adds a layer instance on top of the layer stack.
  - `>>> Sequential.add(layer)`

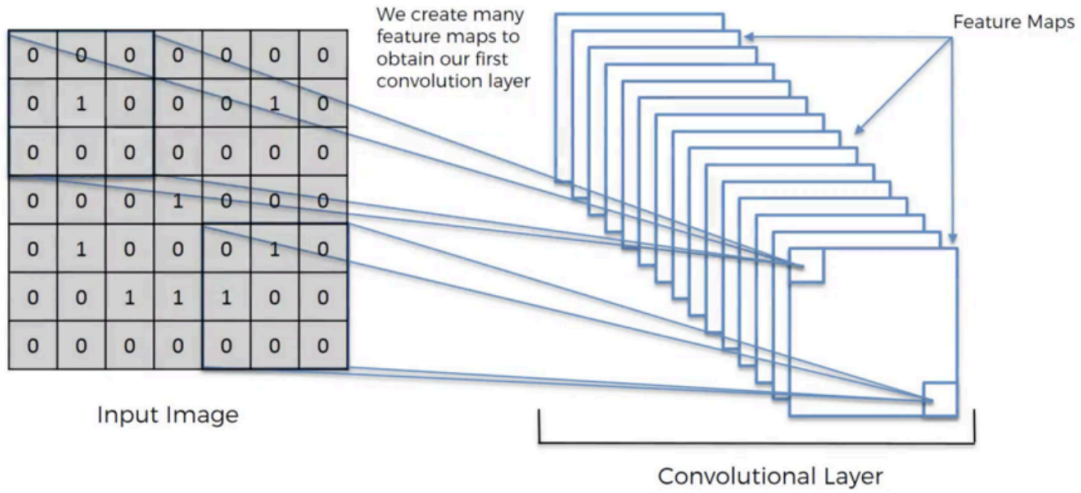
- Convolution layers

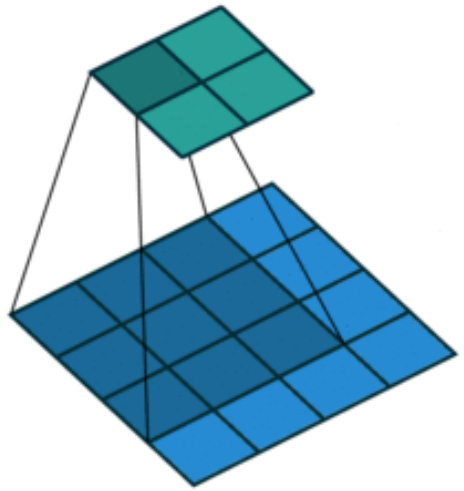
- 2D convolution layer (e.g. spatial convolution over images).
- Arguments
  - Filters
  - Kernel size
  - Stride
  - padding

```
tf.keras.layers.Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding="valid",
    data_format=None,
    dilation_rate=(1, 1),
    groups=1,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

- Convolution layers

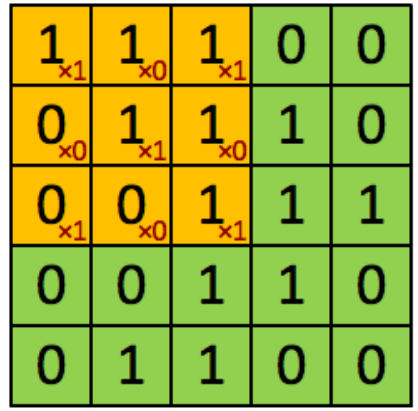
- Filters (a.k.a. kernel)
  - It's an integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
  - That means a convolution is how the input is modified by a filter (Feature Detector).
  - "Input Image" ⊗ " Filter" = Feature Map
- Kernel size
  - An integer or tuple/list of 2 integers. (e.g. 3*3可用一 integer "3"表示)
- Padding
  - Valid
  - Same



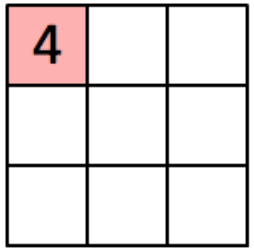


- Example of Convolution

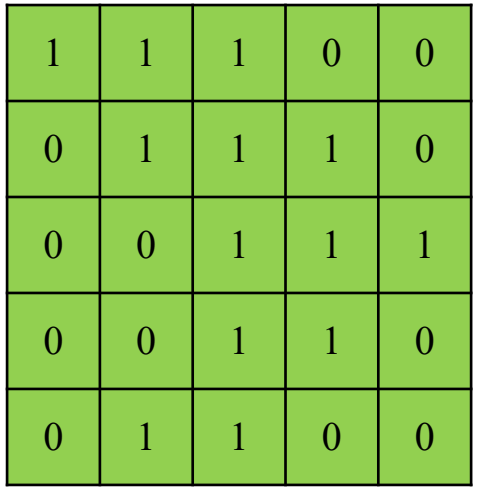
Stride = 1



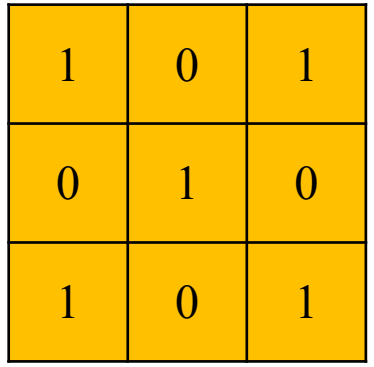
Image



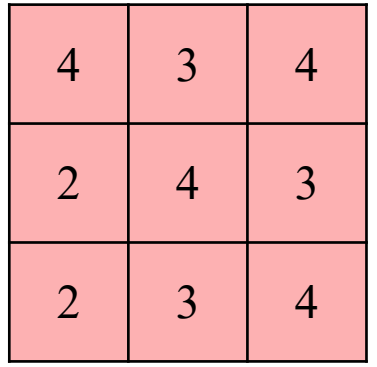
Convolved Feature



Input image



Filter
(This is just one of the filters)



Feature map

- Padding

In Keras, they divided 2 categories

- Same: make same height/width dimension as the input.
- Valid: vice versa

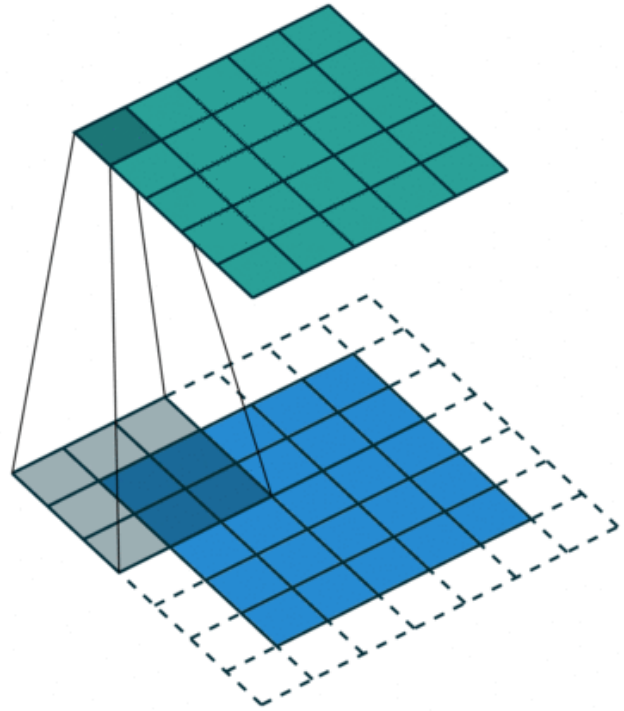
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	6	0	0
0	3	1	2	0	0
0	1	0	2	2	0
0	0	0	0	0	0

⊗

2	2	1
1	0	2
1	0	4

=

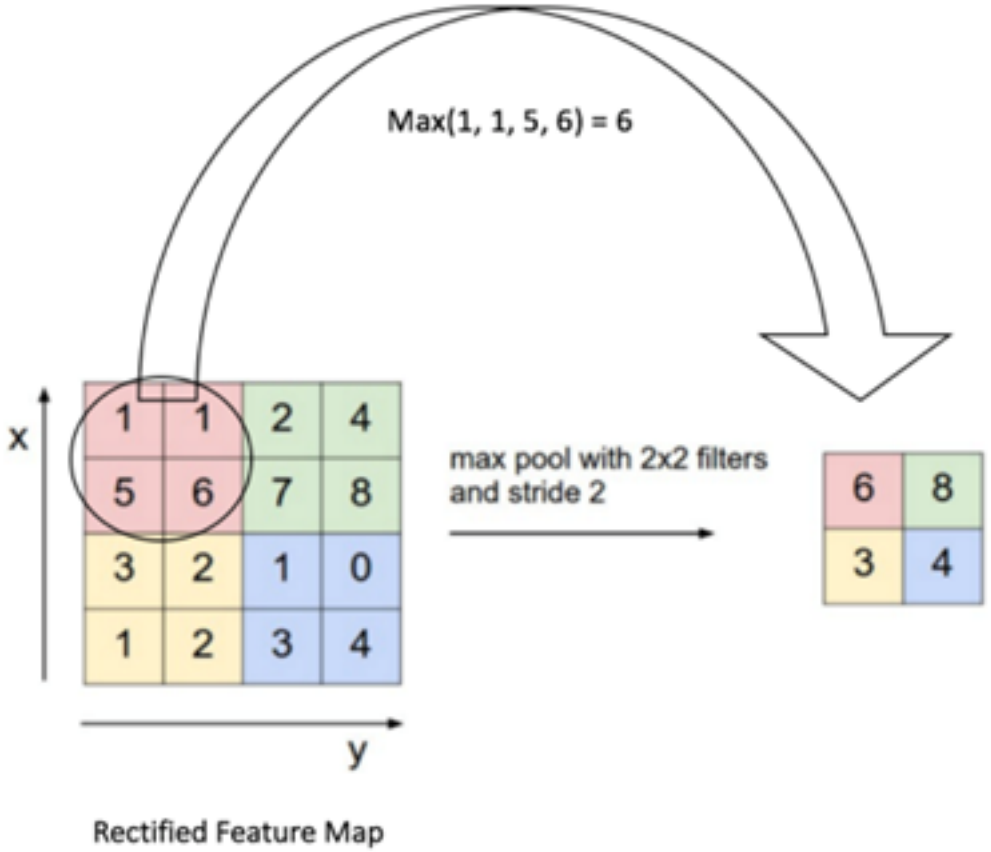
a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}



- Pooling layers

- MaxPooling2D layer

```
tf.keras.layers.MaxPooling2D(
    pool_size=(2, 2), strides=None, padding="valid", data_format=None, **kwargs
)
```

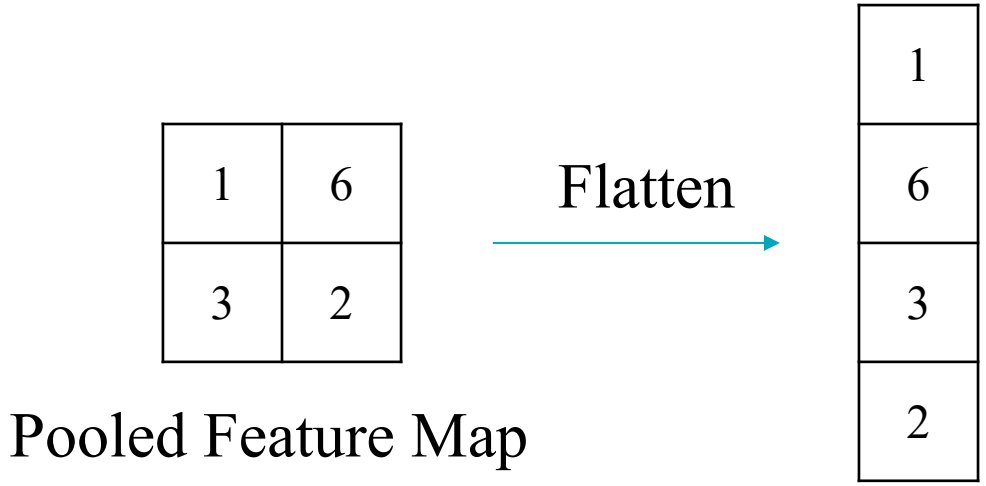


- Reshaping layers

- Flatten layer

```
tf.keras.layers.Flatten(data_format=None, **kwargs)
```

  - Remove all of the dimensions except for one.



- Example of Flatten layer

Example

```
>>> model = tf.keras.Sequential()
>>> model.add(tf.keras.layers.Conv2D(64, 3, 3, input_shape=(3, 32, 32)))
>>> model.output_shape
(None, 1, 10, 64)
```

```
>>> model.add(Flatten())
>>> model.output_shape
(None, 640)
```

- Core layers

- Dense layer

```
tf.keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

output = activation(dot(input, kernel) + bias)

- Training model

- Model.fit

```
model.fit(x = x_train, y = y_train,
validation_split=0.1, batch_size = 256,
verbose=2, epochs=5)
```

  - validation_split: Float between 0 and 1. Fraction of the training data to be used as validation data.
  - batch_size: Number of samples per gradient update. If unspecified, batch_size will default to 32.
  - verbose: 0, 1, or 2. Verbosity mode. 0 = silent, 1 = progress bar, 2 = one line per epoch.

- Validation model

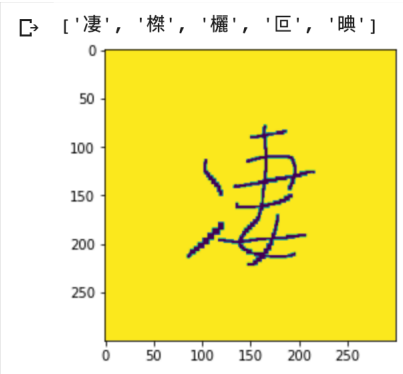
- Model.evaluate

```
score = model.evaluate(x_test, y_test,
verbose=0)
print('Test accuracy:
{:0.2f}%'.format(score[1] * 100))
```

  - >>> Test accuracy: 92.17%

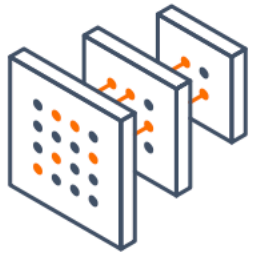
- Inference
  - Using matplotlib library to see the label of image is match to the real image.

```
1 import matplotlib.pyplot as plt
2 from random import randint
3 %matplotlib inline
4 idx = randint(0, len(x_test))
5 img = x_test[idx]
6 plt.imshow(img.squeeze())
7 pred = model.predict(np.expand_dims(img, axis=0))[0]
8 ind = (-pred).argsort()[:5]
9 latex = [class_names[x] for x in ind]
10 print(latex)
```



TensorFlow.js

- According to the official website, we can easily use TensorFlow.js to convert a Python TensorFlow models to run in the browser or under Node.js.



執行現有模型

使用現成的 JavaScript 模型或轉換 Python TensorFlow 模型，以便在瀏覽器作業或利用 Node.js 執行。

使用官方 TensorFlow.js 模型 →

轉換 Python 模型 →

- Installation

- Save the model you train
  - >>> model.save('keras.h5')
- Install the package
  - >>> !pip install tensorflowjs
- Create a folder and convert the model to the format that available to TensorFlow.js.
  - >>> !mkdir model #創立一個名稱為model的資料夾
  - >>> !tensorflowjs_converter --input_format keras keras.h5 model/ #將keras.h5的model轉換
- Copy the class name.txt to the target under folder
  - >>> !cp class_names.txt model3/class_names.txt
- Make all the files to a .zip package
  - >>> !zip -r model3.zip model3

- Apache server deployment

- ☑[TODO] (option1)

- Apache HTTP Server
  - A free and open-source cross-platform web server software
  - You can deploy Apache by installing XAMPP

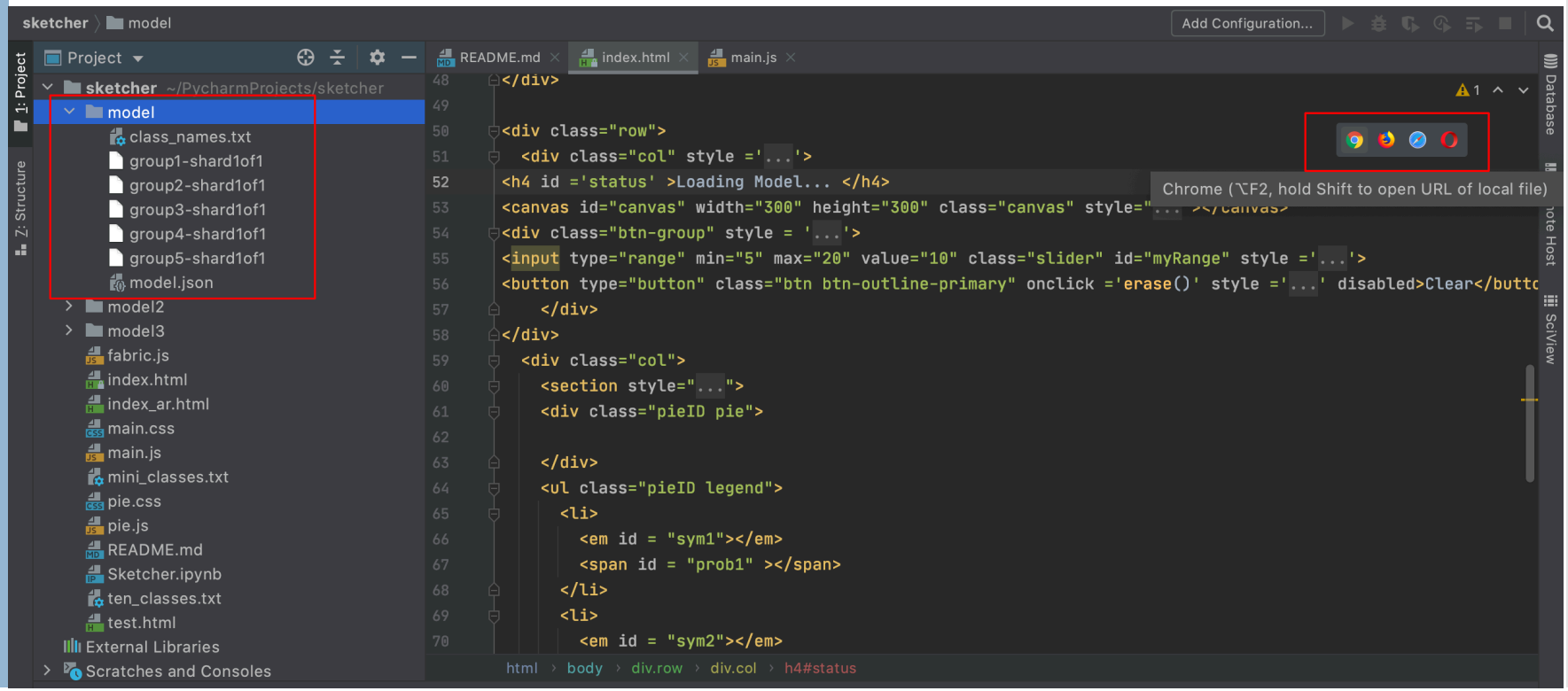
- XAMPP (to see more detail 👉 AIMS_TA_Course03_xampp.pdf)
  - Easy to install Apache distribution containing MariaDB, PHP, and Perl
  - Download and install

- IDEs with PyCharm

- Downloads PyCharm
  - Community (free, open source)
  - Professional (it's free if you have a .edu email account)

- PyCharm deployment

- ☑ [TODO] (option2)
  - After converting the model, please put the related files under the model folder (as the left red frame).
  - Then, choose the browser you want (as the right red frame) to use to show the demo page.



- Traditional Chinese Handwriting

- Original dataset was produced based on Tegaki, an open-source package. Total 13,065 different Chinese characters, with average of 50 samples for each character.
- 原始資料集基於 Tegaki 開源套件下產出，總計 13,065 個不同的中文字，每一個字體平均有 50 個樣本。

- ☑ **[TODO]**
  - **Mission: try to predict the character you write (draw?).**
  - TAs have prepared the dataset (each image with 300*300), so we'll choose an article and assign the characters to each of you.

- Assignment allocation

Data download ⇣
1. Zip: link1🔗
2. Rar: link2🔗

☞ choose another way in case of any decoding problem!

編號	姓名	題目
1	張*鈞	台中除了慶記還有什麼
2	張*銘	正妹的男友不是臉帥身材好就是臉普有點錢
3	邱*民	三十歲有什麼一定要做不然會後悔的八卦
4	鍾*	如何激怒夜市門口化緣的和尚
5	張*義	想問一下病人告贏醫生的案例多嗎
6	莊*皓	我老婆直播自己看書是不是香到爆
7	李*峯	欸要上場了還在宿醉啦
8	王*晴	認真問現在買電車好嗎
9	涂*桂	真的戰爭被共軍俘虜會被怎麼對待
10	謝*燕	副都心在新莊是不是瞧不起古亭啊
11	謝*安	吳怡農建議在家當兵
12	簡*吉	中國比較像漢還是比較像楚
13	林*茂	請問如果戰爭被徵兵不去會怎樣
14	林*騏	貓有犀利眼神有加分嗎

- Project assignments

- Please check out the ☑ [TODO] list to complete the project we assign.
- Any of question about the class is welcome.
- You could
  - 1. issue on the course website
  - 2. ask TAs in class
- Deadline
  - 2020/11/11 (Wed.) demo in class.
  - 2020/11/11 (Wed.) 23:59 written report.

- Score assessment

- 30%: 能夠將model訓練並存為(.h5)模型
- 20%: 能夠從(.h5)模型轉為tensorflow.js支持格式並在網頁demo
- 10%: 精準度至少85%(?)以上
- 40%: written report (上傳區點此)
  - 2頁(±0.5頁)a4, 形式不拘
  - 除了模型架構和精準度等客觀評比外，可以寫下project或課堂所學、嘗試改進部分、所遇到問題)

- References

- Quick draw dataset
  - https://github.com/googlecreativelab/quickdraw-dataset
- Keras
  - https://keras.io/api/models/
- Matplotlib
  - https://matplotlib.org/
- PyCharm
  - https://www.jetbrains.com/pycharm/
- TensorFlow.js
  - https://www.tensorflow.org/js
- Traditional-Chinese-Handwriting-Dataset
  - https://github.com/AI-FREE-Team/Traditional-Chinese-Handwriting-Dataset