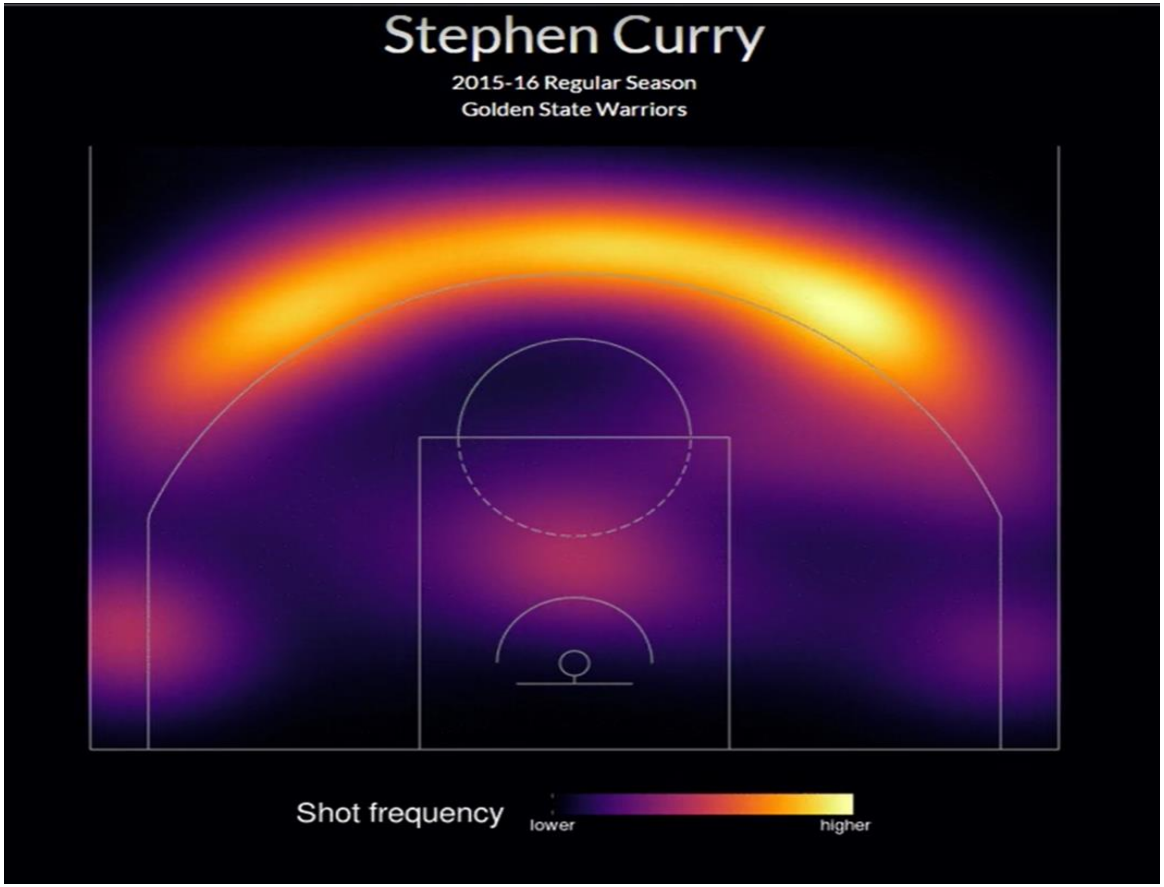
深度學習智慧應用
Lab 4: FCN、U-Net and TrackNet

Yu Ching Lin
Network Optimization Lab (NOL)
Department of Computer Science
National Chiao Tung University

Outline

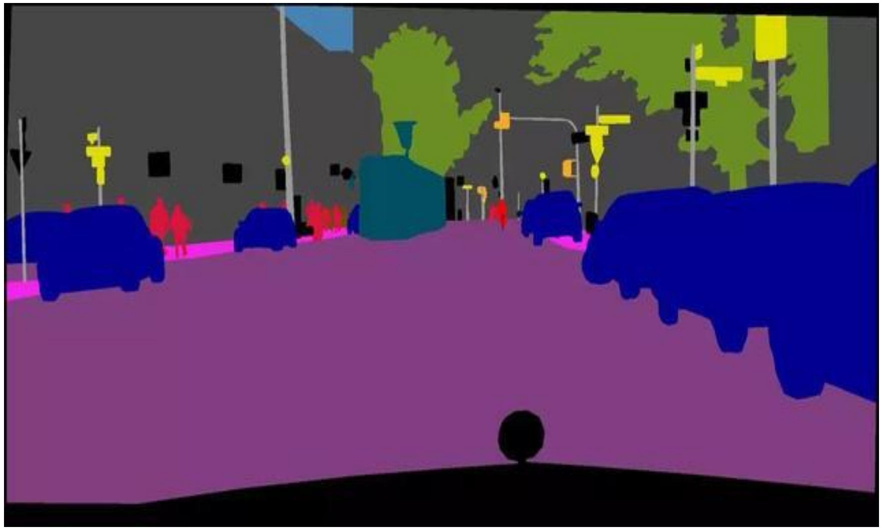
- Heatmap
- Semantic Segmentation
- FCN
- U-Net
- TrackNet

HeatMap


Stephen Curry
2015-16 Regular Season
Golden State Warriors
Shot frequency
lower
higher

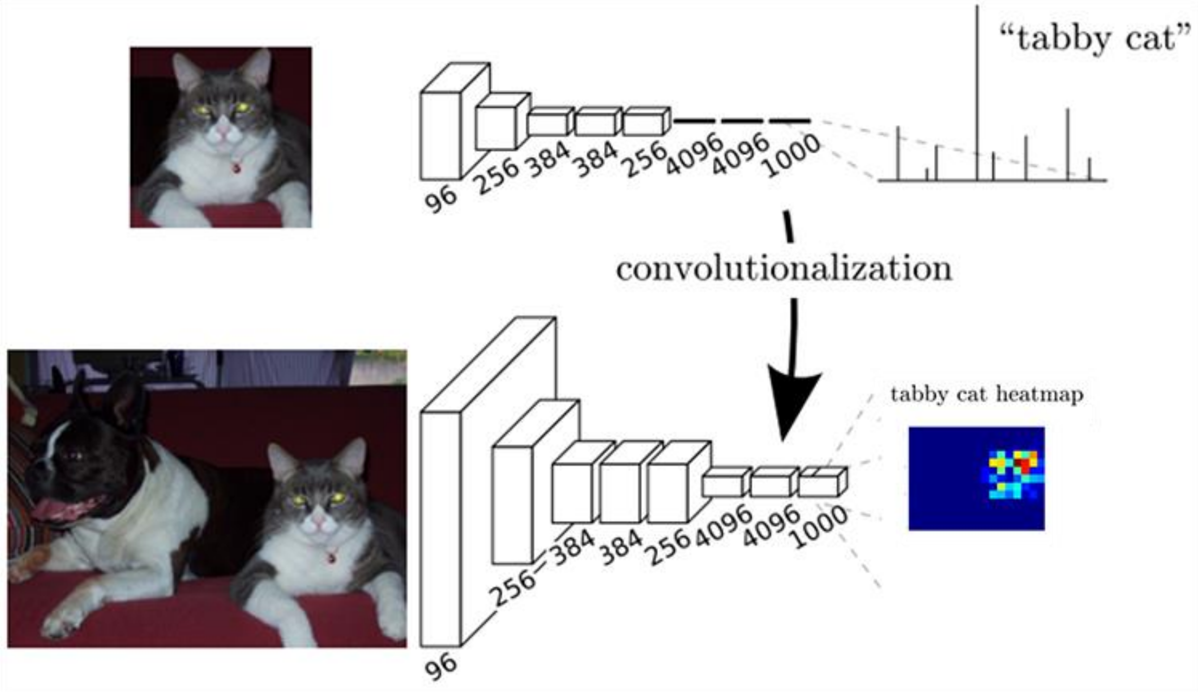
Semantic Segmentation

Pixel-wise classification

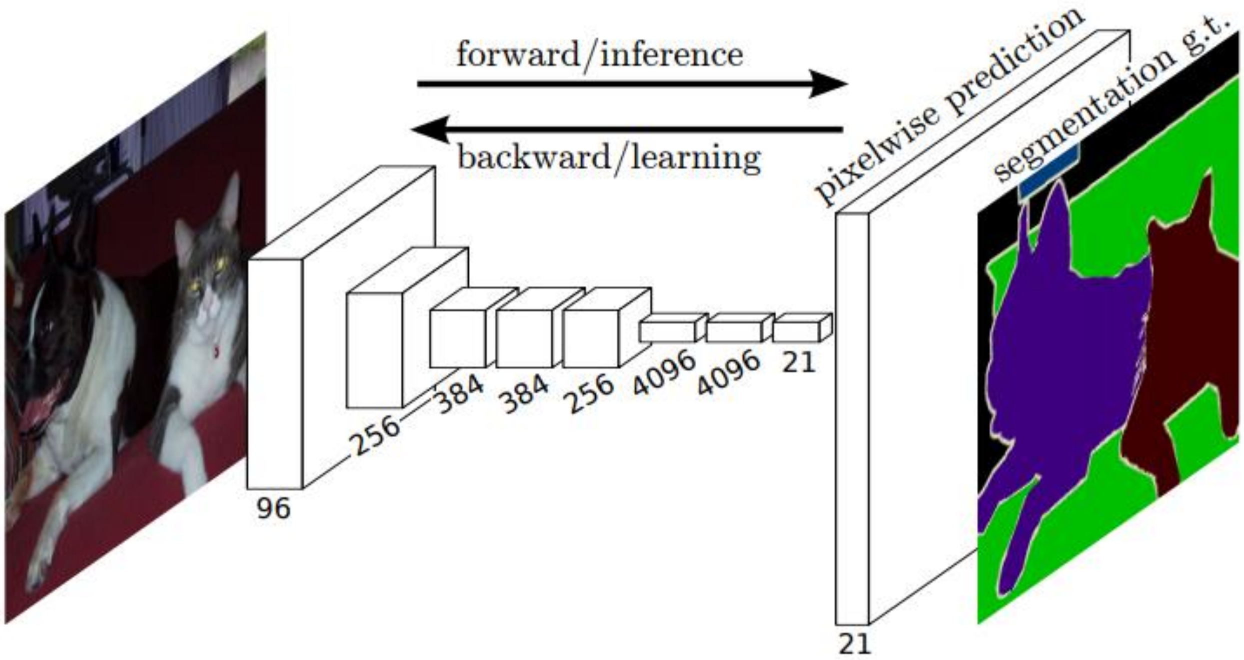


FCN

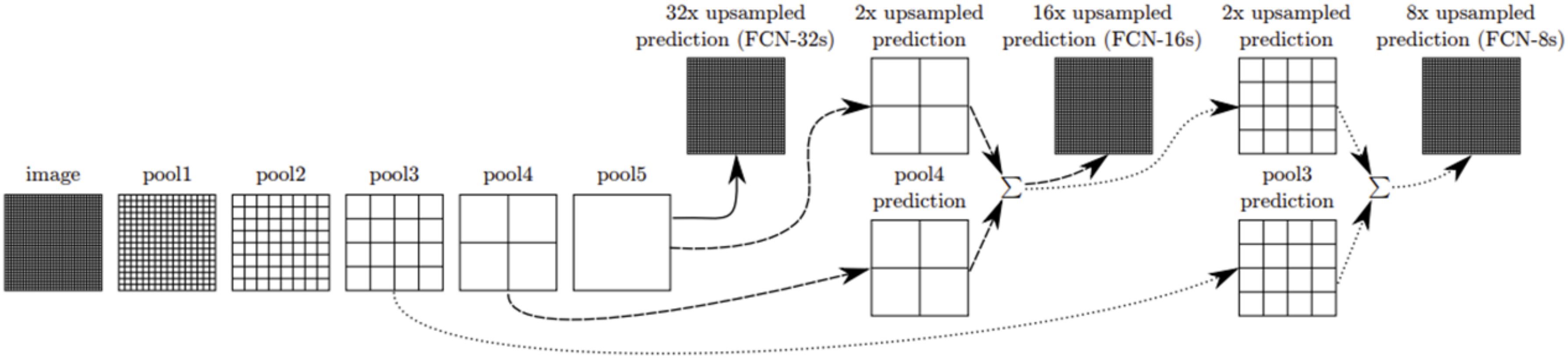
Convolutionalization： 全連接層->卷積層

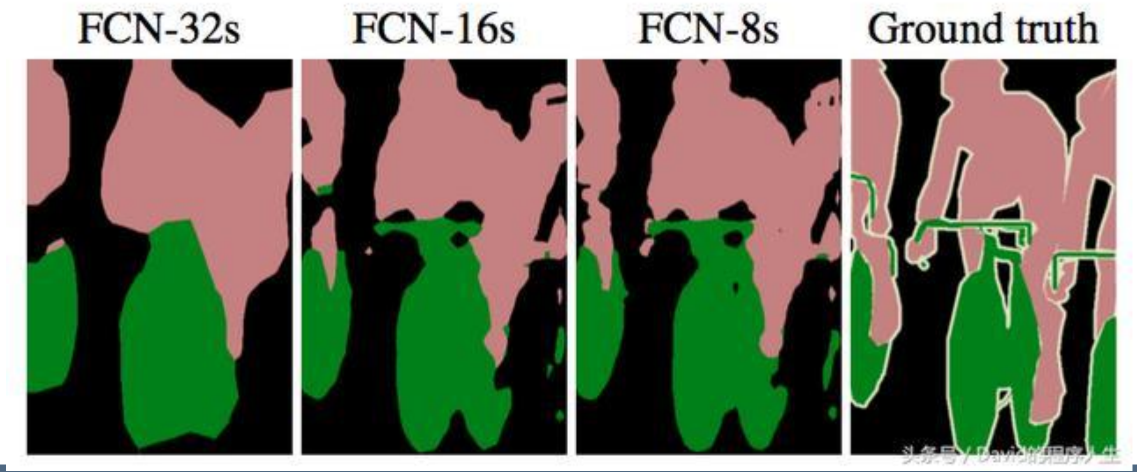


FCN


forward/inference
backward/learning
pixelwise prediction
segmentation g.t.
96
256
384
384
256
4096
4096
21
21

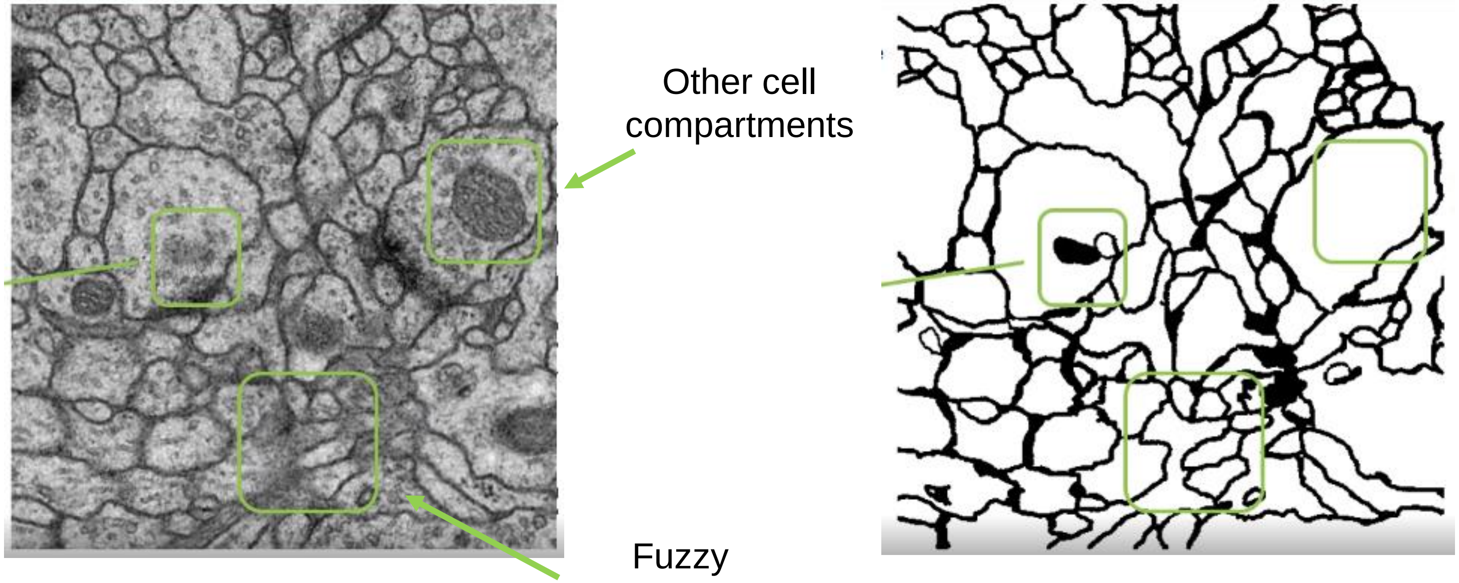
FCN


32x upsampled prediction (FCN-32s)
2x upsampled prediction
16x upsampled prediction (FCN-16s)
2x upsampled prediction
8x upsampled prediction (FCN-8s)
image
pool1
pool2
pool3
pool4
pool5
pool4 prediction
pool3 prediction
∑
∑


FCN-32s
FCN-16s
FCN-8s
Ground truth
头条号 / David的程序人生

U-Net

Modify and extend the architecture of fully convolutional network such that it works with very few training images and yields more precise segmentations



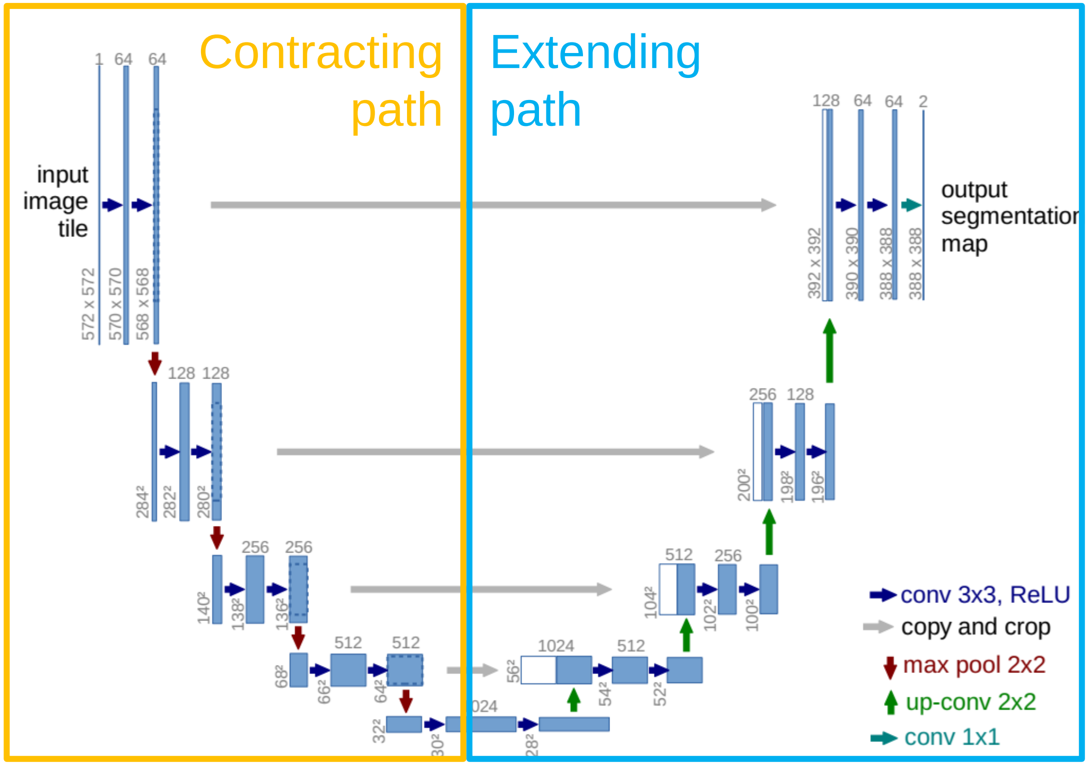
Other cell compartments

Structures with very low contrast

Fuzzy membranes

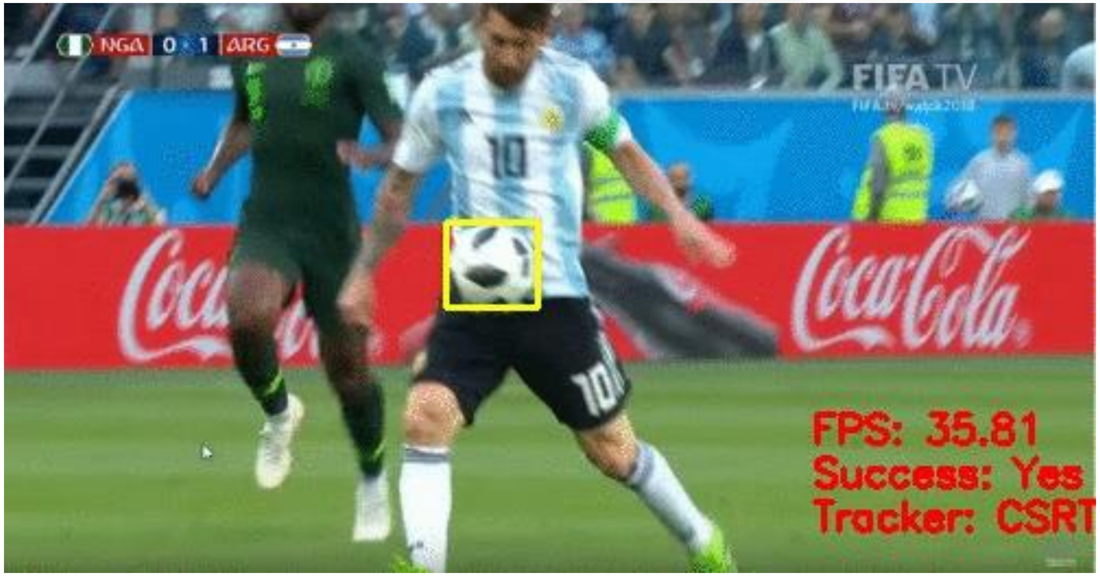
U-Net Architecture

Concatenation with the correspondingly cropped feature map from the contracting path


Contracting path

Extending path

TrackNet

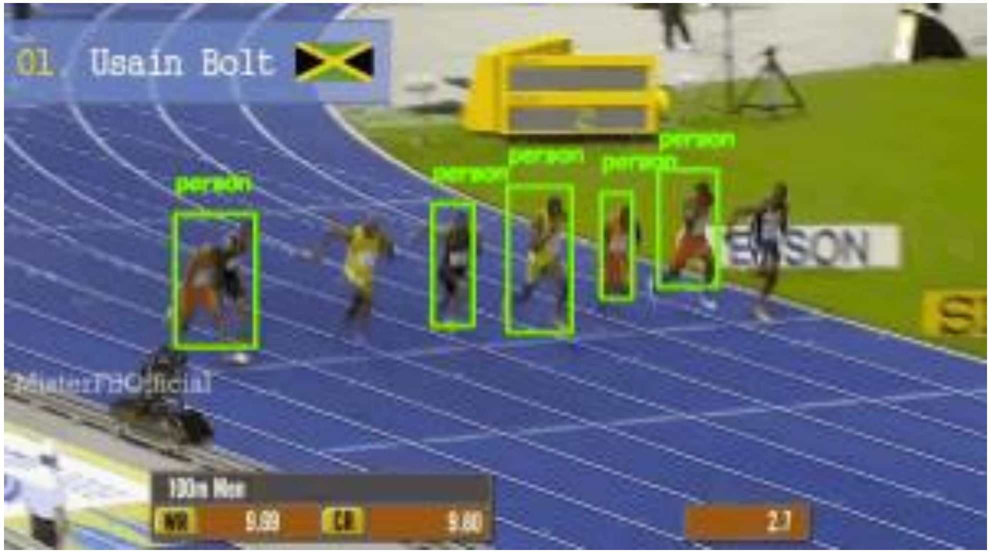


Single Object Tracking

https://sandipanweb.wordpress.com/category/computer-vision/

Multiple Object Tracking

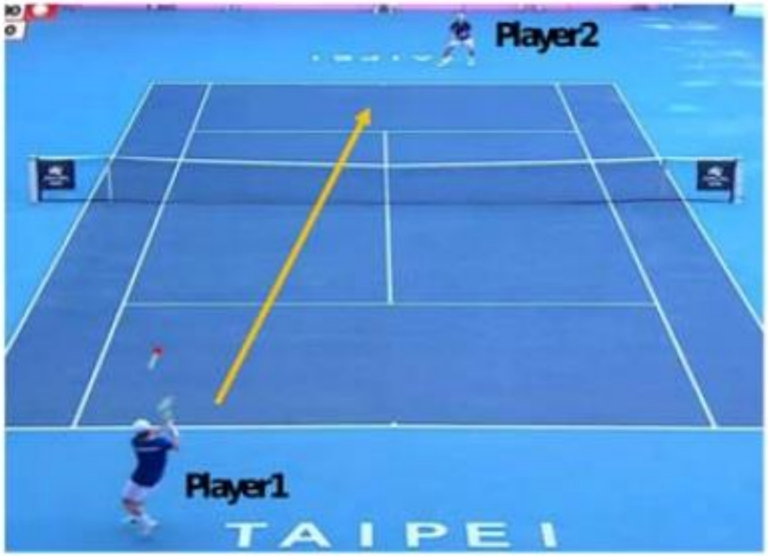
https://www.pyimagesearch.com/2018/10/29/multi-object-tracking-with-dlib/



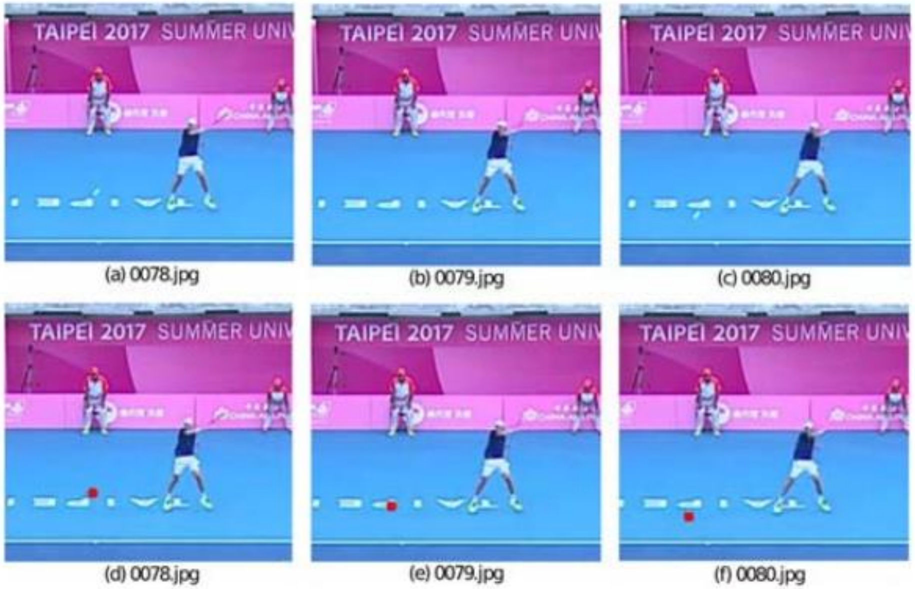
https://nanonets.com/blog/object-tracking-deepsort/

TrackNet

https://people.cs.nctu.edu.tw/~yi/TechReports/TrackNet.v1_Final_NOL.pdf



標記網球位置的時，網球呈現軌跡拖延的案例

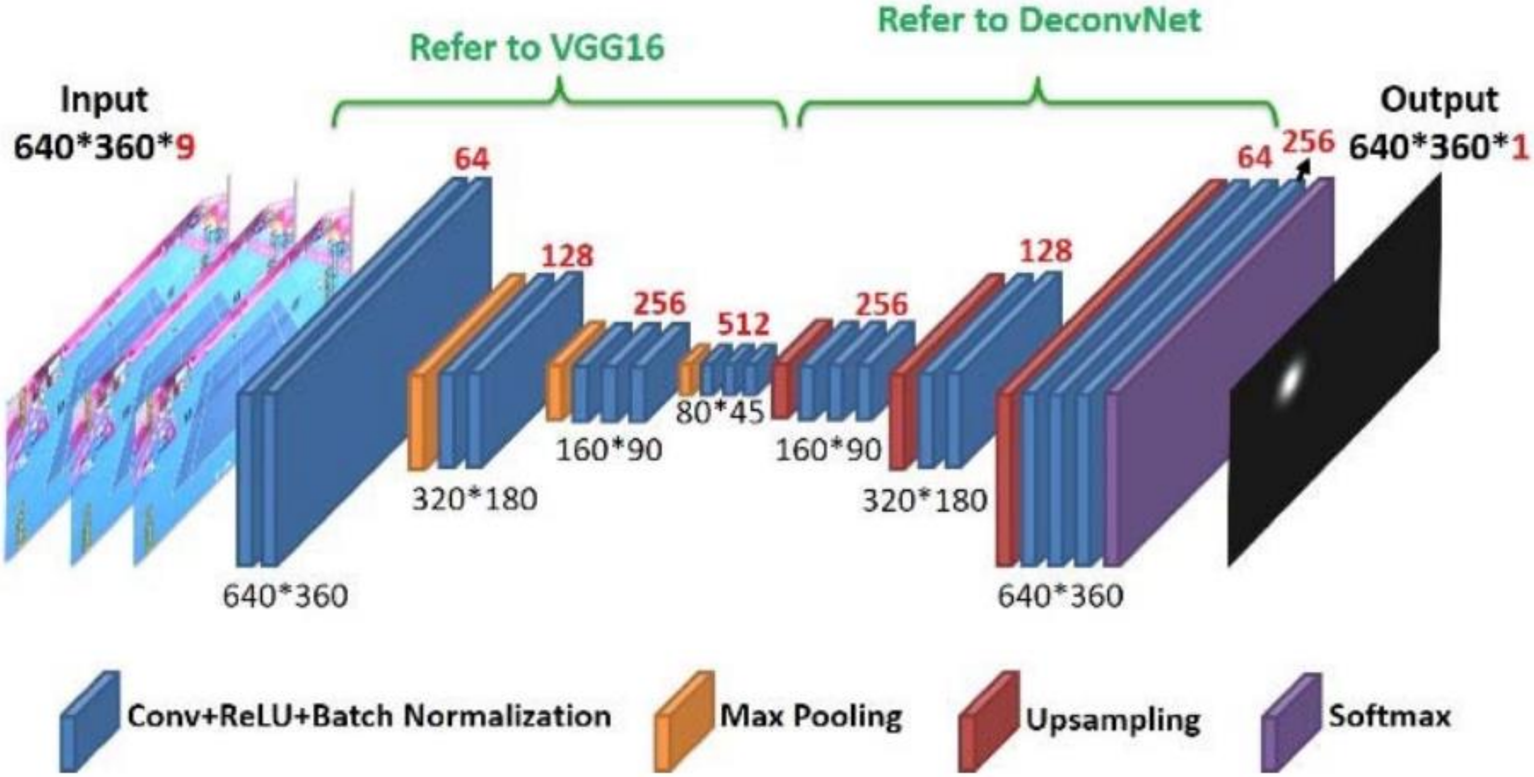


(a) 0078.jpg (b) 0079.jpg (c) 0080.jpg

(d) 0078.jpg (e) 0079.jpg (f) 0080.jpg

網球影像無法被識別

TrackNet


Refer to VGG16
Refer to DeconvNet
Input
640*360*9
Output
640*360*1
64
128
256
512
256
128
64
256
640*360
320*180
160*90
80*45
160*90
320*180
640*360
Conv+ReLU+Batch Normalization
Max Pooling
Upsampling
Softmax

TrackNet

- Performance

Lab 4

Outline

- Data Label
- Data Generate
- Model Architecture
- Evaluation standard
- Ways to improve
- Score

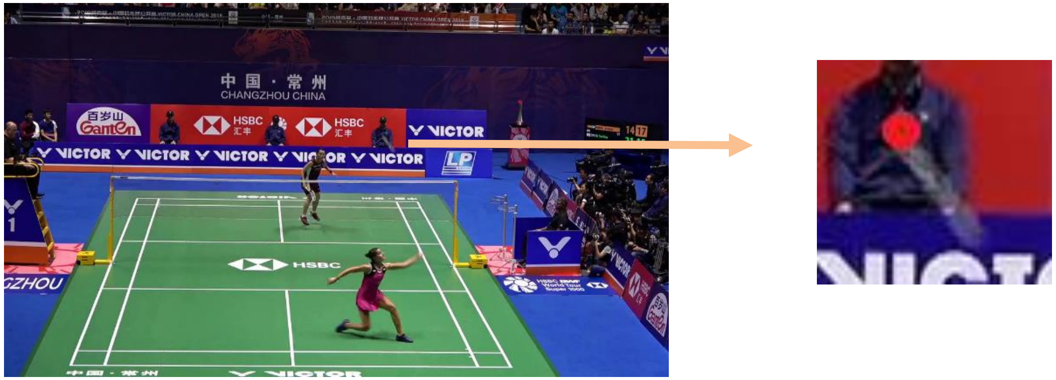
Data Label

Label Tool

Just follow the step, you will get label result as csv file
HackMD : https://hackmd.io/@JVTTHln7SmqjN39abZ9DLQ/B1lpZCBPj

Rules:

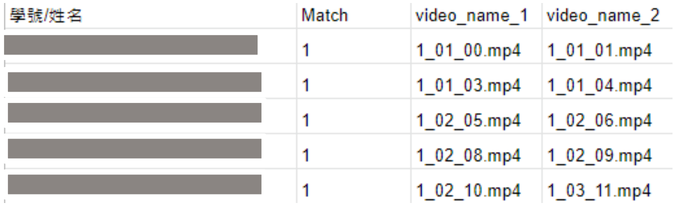
1. Please mark the ball head of shuttlecock (follow forward direction)



2. If you can’t see the ball, skip that.

Input : mp4, output : csv

Data Label



學號/姓名	Match	video_name_1	video_name_2
[redacted]	1	1_01_00.mp4	1_01_01.mp4
[redacted]	1	1_01_03.mp4	1_01_04.mp4
[redacted]	1	1_02_05.mp4	1_02_06.mp4
[redacted]	1	1_02_08.mp4	1_02_09.mp4
[redacted]	1	1_02_10.mp4	1_03_11.mp4

- Label list :

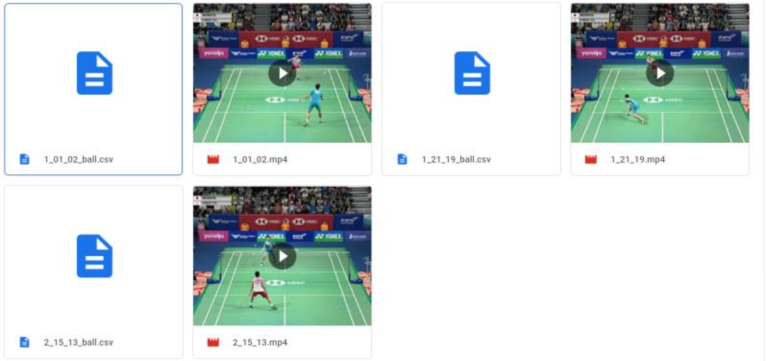
https://docs.google.com/spreadsheets/d/1V2BTacTkfsGQEgjU_1Yfs4c4a_7GOKWzLELAcjEaJJs/edit#gid=1385358667

- Data :

https://drive.google.com/drive/folders/1wrODR97BXw9Rd3uPY-bQVwI0s5K0I1Pr?usp=share_link

- Everyone need to label 2 videos

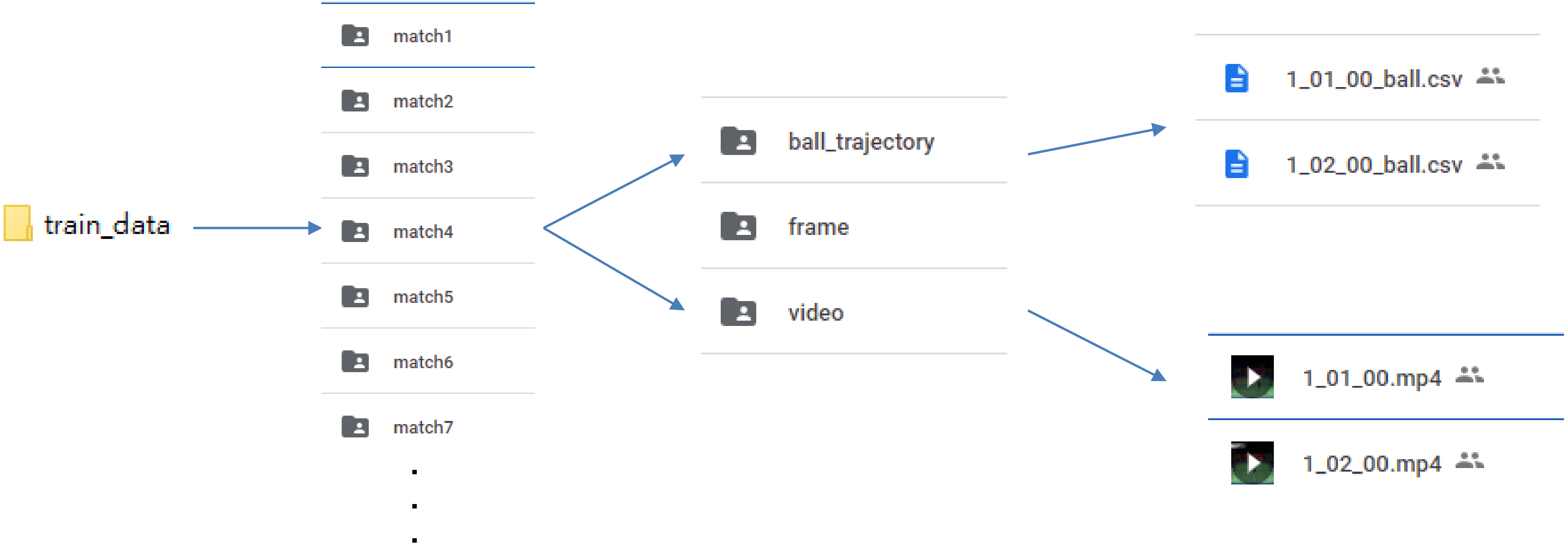
- Upload your csv file (after labeling) in the same match folder



Data Generate

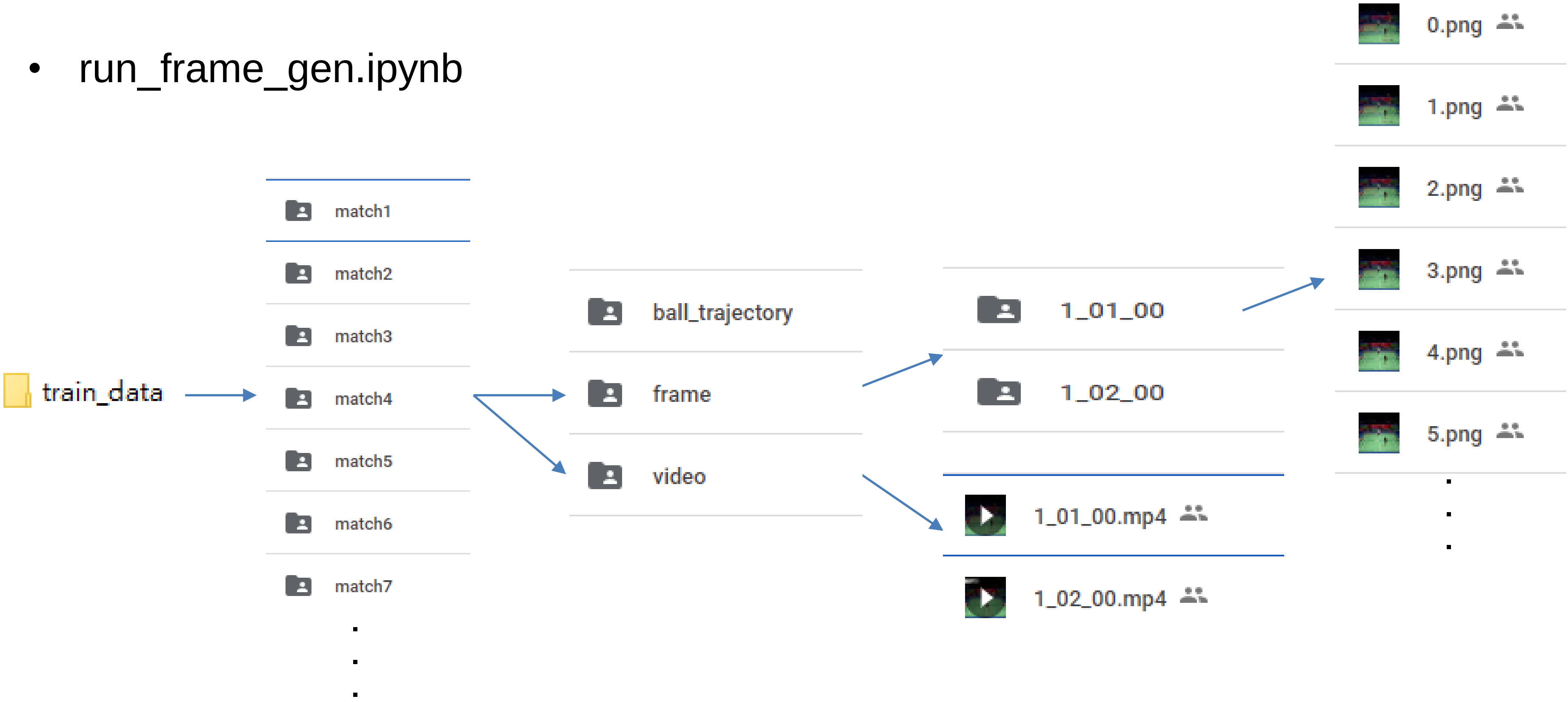
• 資料結構

- 將Label完的csv檔放到對應的比賽資料夾下方



• run_frame_gen.ipynb

- run_frame_gen.ipynb



train_data

match1
match2
match3
match4
match5
match6
match7
.
.
.

ball_trajectory
frame
video

1_01_00
1_02_00

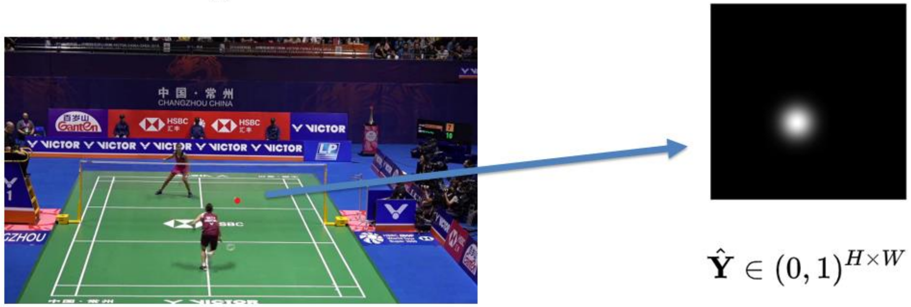
1_01_00.mp4
1_02_00.mp4

0.png
1.png
2.png
3.png
4.png
5.png
.
.
.

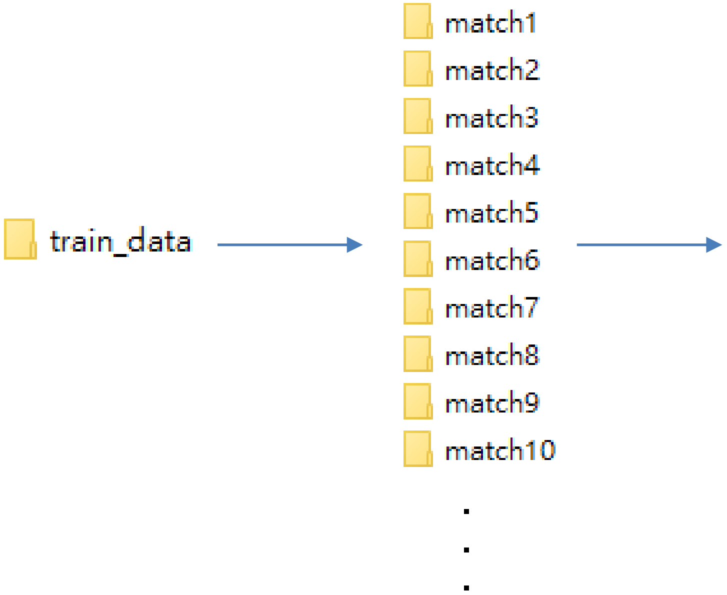
- 將train_data底下的所有比賽影片切成一個一個frame存起來

• run_gen_data.ipynb

- run_gen_data.ipynb

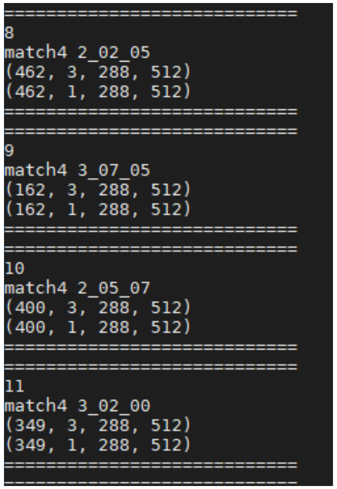


$\hat{\mathbf{Y}} \in (0,1)^{H\times W}$

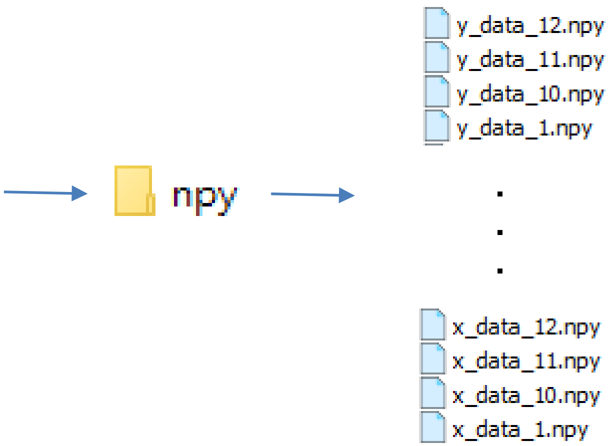


train_data

match1
match2
match3
match4
match5
match6
match7
match8
match9
match10
.
.
.



```
============================
8
match4 2_02_05
(462, 3, 288, 512)
(462, 1, 288, 512)
============================
============================
9
match4 3_07_05
(162, 3, 288, 512)
(162, 1, 288, 512)
============================
============================
10
match4 2_05_07
(400, 3, 288, 512)
(400, 1, 288, 512)
============================
============================
11
match4 3_02_00
(349, 3, 288, 512)
(349, 1, 288, 512)
============================
```

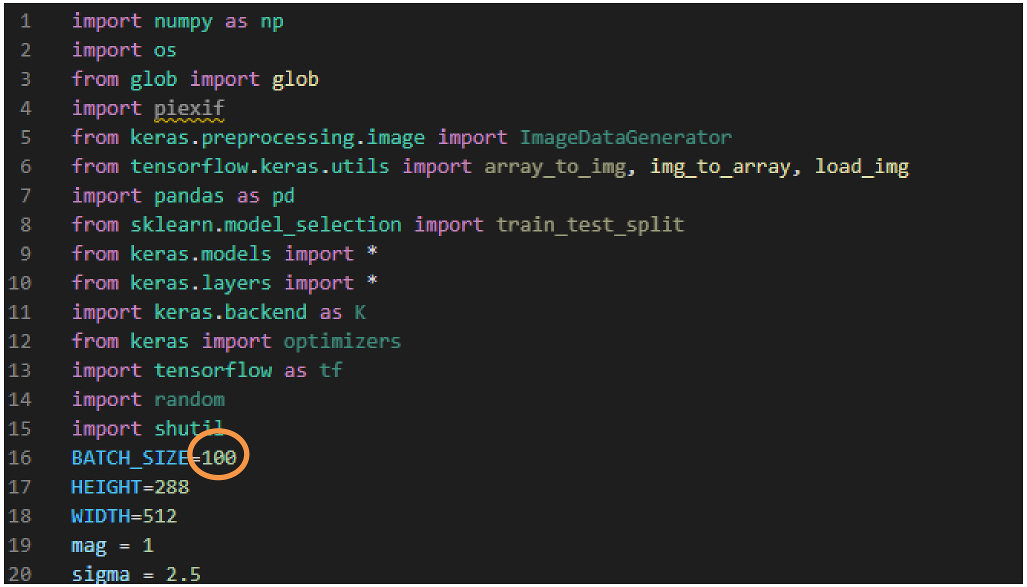


npy

y_data_12.npy
y_data_11.npy
y_data_10.npy
y_data_1.npy
.
.
.
x_data_12.npy
x_data_11.npy
x_data_10.npy
x_data_1.npy

Customizing batch size

- 由於Colab的GPU記憶體空間不足，預設的batch size皆為100，若想調整batch size，更改的地方如下。



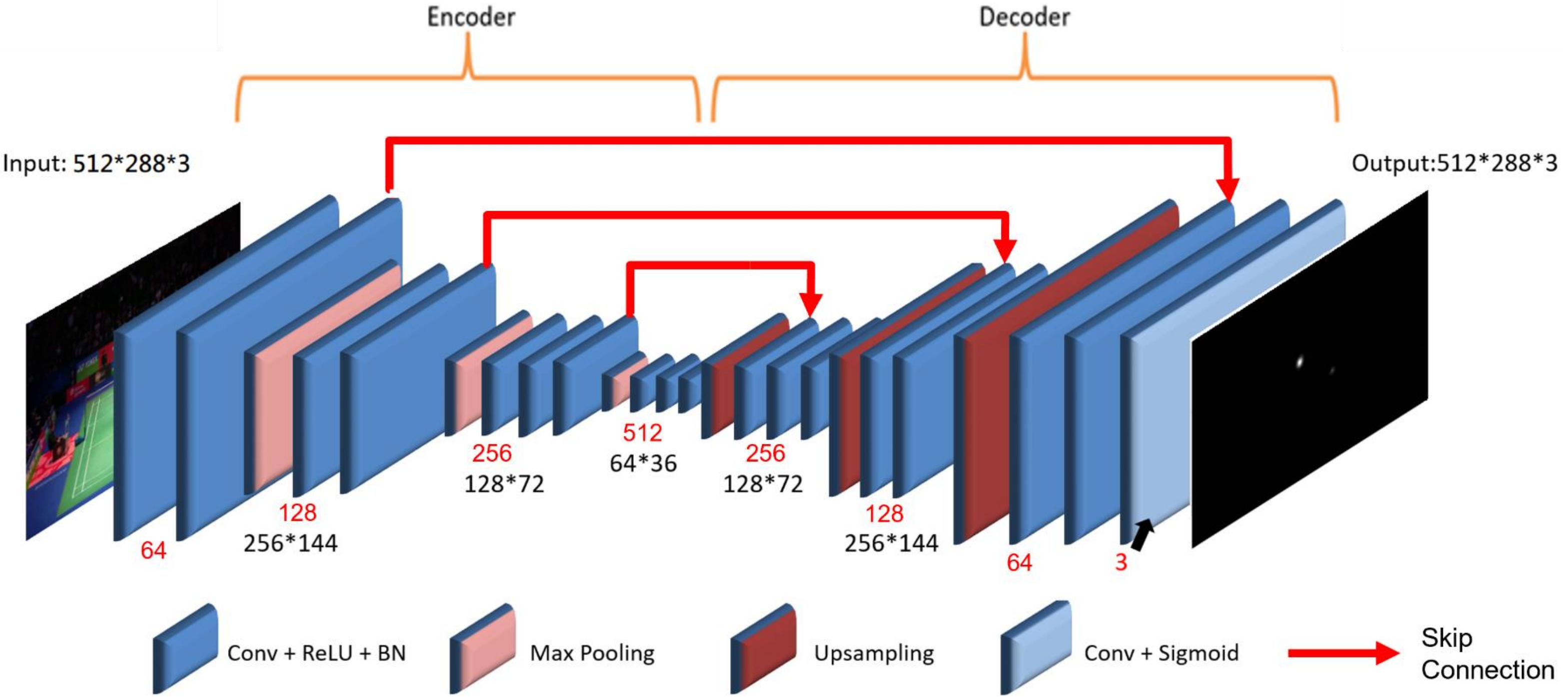
```python
import numpy as np
import os
from glob import glob
import piexif
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import array_to_img, img_to_array, load_img
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import *
from keras.layers import *
import keras.backend as K
from keras import optimizers
import tensorflow as tf
import random
import shutil
BATCH_SIZE=100
HEIGHT=288
WIDTH=512
mag = 1
sigma = 2.5
```

gen_data_rally.py

Model architecture

TrackNet.py

Modify model by yourself ! (If you need)



Base model

Loss function

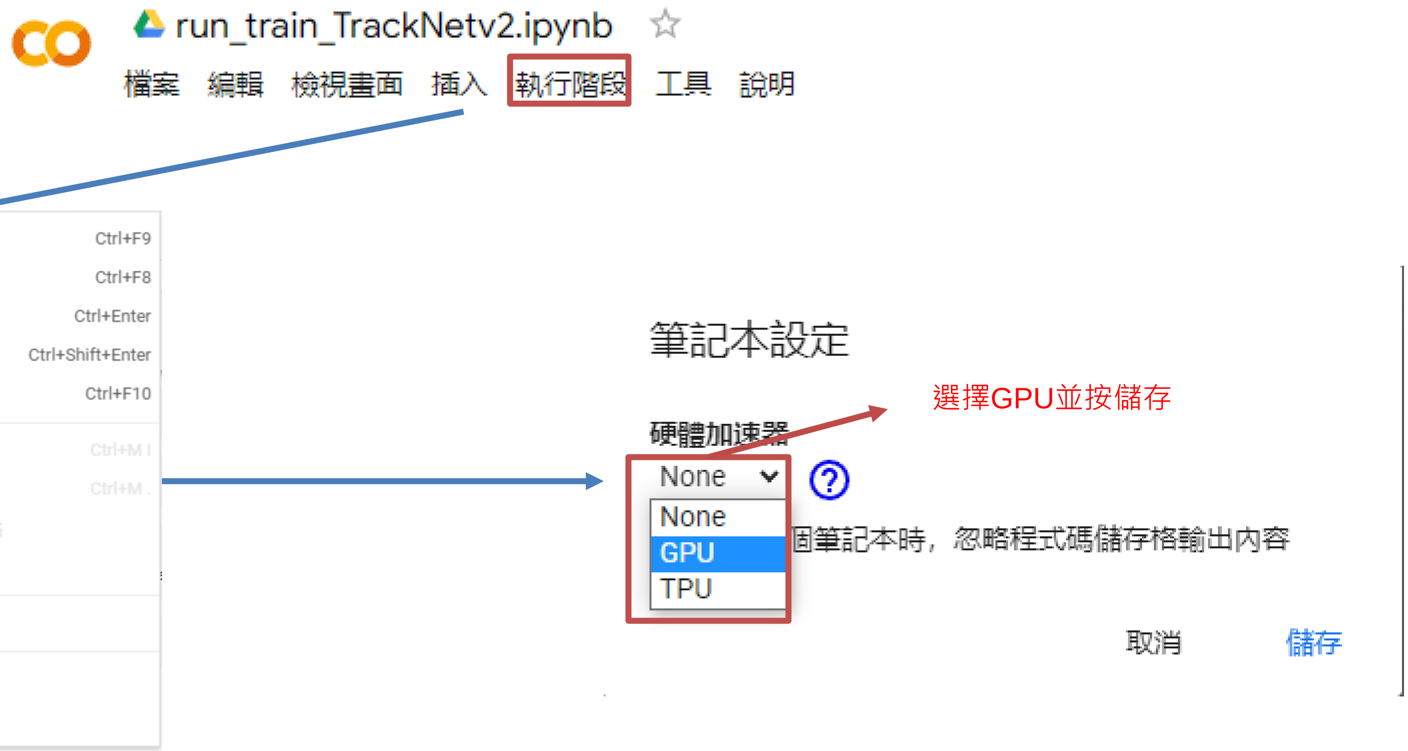
Binary cross entropy

y = ground truth
$\hat{y}$ = predict result

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

run_train_TrackNetv2.ipynb

- Colab選擇GPU環境



硬體加速器

固筆記本時，忽略程式碼儲存格輸出內容

選擇GPU並按儲存

run_train_TrackNetv2.ipynb

- python3 train_TrackNet.py –load_weights = **<previousWeightPath>** --save_weights = **<weightPath>** --dataDir = **<npyDataDirectory>** --epochs = **<trainingEpochs>** --tol = **<toleranceValue>**

- <**previousWeightPath**> is TrackNet weight you had trained before.
- <**weightPath**> is TrackNet weight after this training
- <**npyDataDirectory**> is the directory of the .npy training data
- <**toleranceValue**> means tolerance value of true positive (we use 5)

EX :

- 從頭開始train

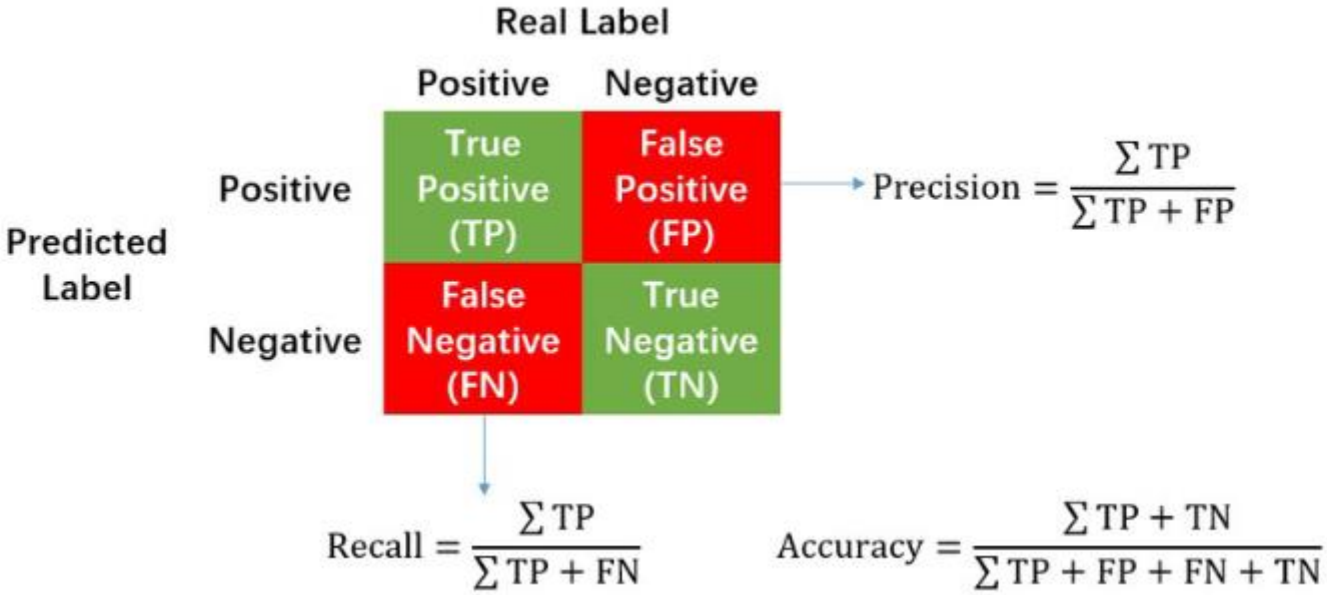
```
python3 train_TrackNet3.py --save_weights=mymodel --dataDir=npy --epochs=30 --tol=5
```

- 加train

```
python3 train_TrackNet3.py --load_weights=mymodel/mymodel_3
--save_weights=mymodel --dataDir=npy --epochs=30 --tol=5
```

Evaluation standard

Metric



		Real Label	
		Positive	Negative
Predicted Label	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FN}}$$

$$\text{Accuracy} = \frac{\sum \text{TP} + \text{TN}}{\sum \text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

- FP1 :

Both prediction and ground truth are ball existing, but the distance is out of tolerance value.

- FP2 :

The prediction is ball existing, but the ground truth is no ball

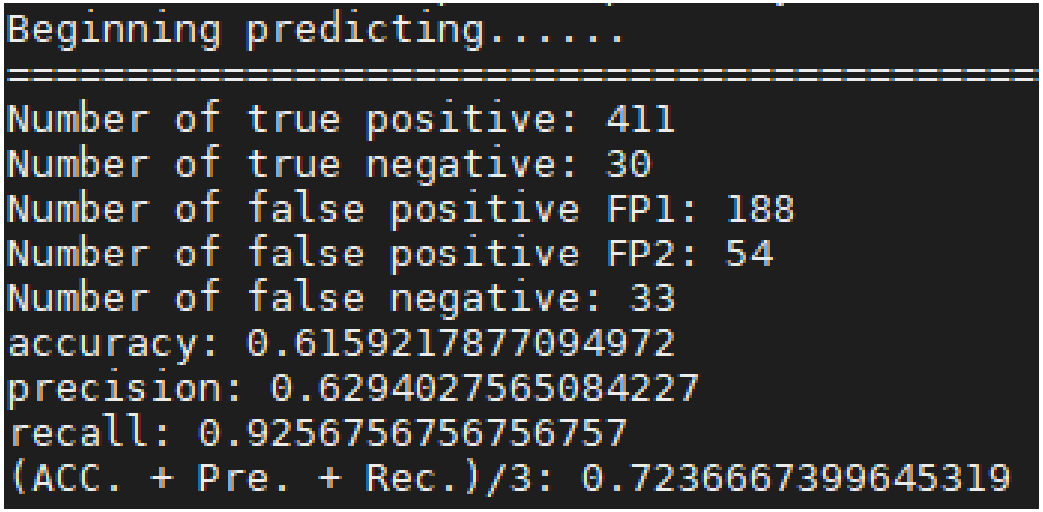
run_predict.ipynb

- Colab也一樣要先選GPU環境
- python3 predict.py --video_name = **<videoPath>** --load_weights = **<weightPath>** --label = **<csvFile>** --tol = **<toleranceValue>**
- <**videoPath**> is video you want to predict.
- <**weightPath**> is weight of model
- <**csvFile**> is the ground truth csv of video
- <**toleranceValue**> means tolerance value of true positive (we use 5)

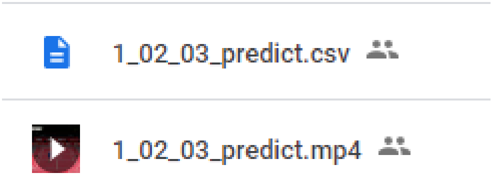
EX:

```
python predict.py --video_name=1_02_03.mp4 --load_weights=model_27 --label=1_02_03_ball.csv --tol=5
```

run_predict.ipynb



```
Beginning predicting.......
===========================================
Number of true positive: 411
Number of true negative: 30
Number of false positive FP1: 188
Number of false positive FP2: 54
Number of false negative: 33
accuracy: 0.615921787709497 2
precision: 0.629402756508422 7
recall: 0.925675675675675 7
(ACC. + Pre. + Rec.)/3: 0.7236667399645319
```



1_02_03_predict.csv

1_02_03_predict.mp4

Ways to improve

Model Architecture

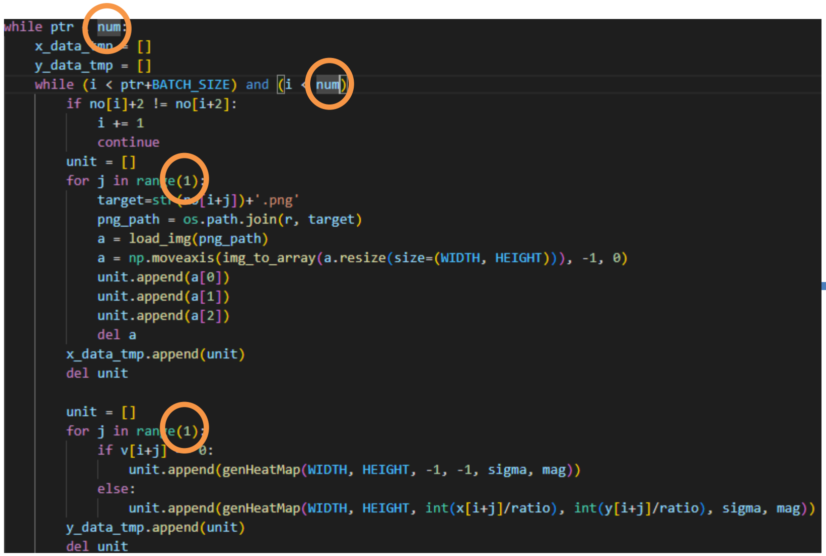
- Multiple Input (or output)
- Channel (Ex: 32 -> 64, 64 -> 128 …)
- Skip connection (while upsampling, like Unet)
- Drop out layer (If overfitting)
- Different activation function

It may cause GPU out of memory …

Multiple input (or output)

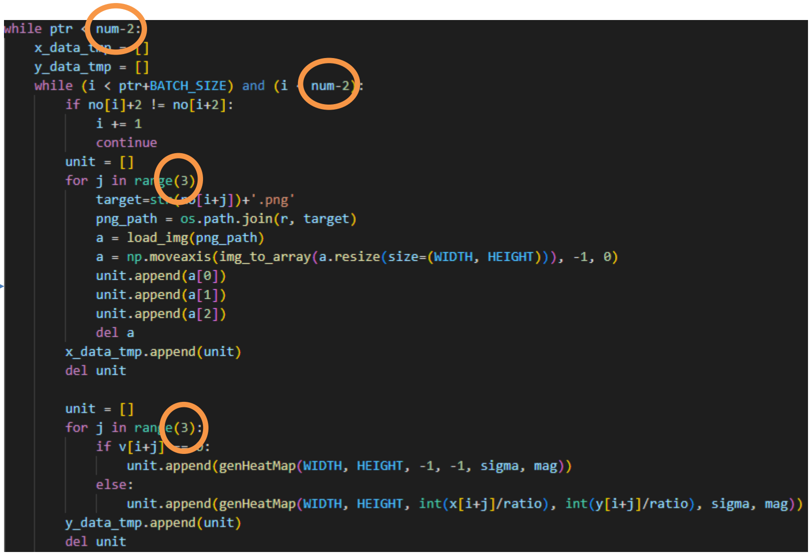
The devil is in the details

gen_data_rally.py



```
while ptr < num:
    x_data_tmp = []
    y_data_tmp = []
    while (i < ptr+BATCH_SIZE) and (i < num):
        if no[i]+2 != no[i+2]:
            i += 1
            continue
        unit = []
        for j in range(1):
            target=str(no[i+j])+'.png'
            png_path = os.path.join(r, target)
            a = load_img(png_path)
            a = np.moveaxis(img_to_array(a.resize(size=(WIDTH, HEIGHT))), -1, 0)
            unit.append(a[0])
            unit.append(a[1])
            unit.append(a[2])
            del a
        x_data_tmp.append(unit)
        del unit

        unit = []
        for j in range(1):
            if v[i+j] == 0:
                unit.append(genHeatMap(WIDTH, HEIGHT, -1, -1, sigma, mag))
            else:
                unit.append(genHeatMap(WIDTH, HEIGHT, int(x[i+j]/ratio), int(y[i+j]/ratio), sigma, mag))
        y_data_tmp.append(unit)
        del unit
```

1-in-1-out



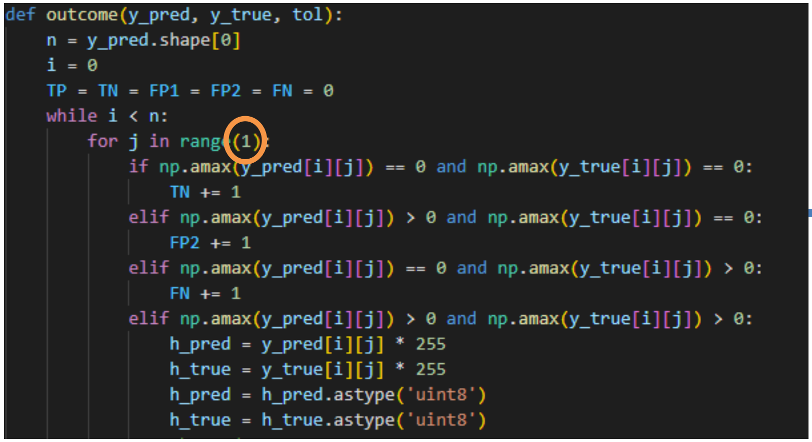
```
while ptr < num-2:
    x_data_tmp = []
    y_data_tmp = []
    while (i < ptr+BATCH_SIZE) and (i < num-2):
        if no[i]+2 != no[i+2]:
            i += 1
            continue
        unit = []
        for j in range(3):
            target=str(no[i+j])+'.png'
            png_path = os.path.join(r, target)
            a = load_img(png_path)
            a = np.moveaxis(img_to_array(a.resize(size=(WIDTH, HEIGHT))), -1, 0)
            unit.append(a[0])
            unit.append(a[1])
            unit.append(a[2])
            del a
        x_data_tmp.append(unit)
        del unit

        unit = []
        for j in range(3):
            if v[i+j] == 0:
                unit.append(genHeatMap(WIDTH, HEIGHT, -1, -1, sigma, mag))
            else:
                unit.append(genHeatMap(WIDTH, HEIGHT, int(x[i+j]/ratio), int(y[i+j]/ratio), sigma, mag))
        y_data_tmp.append(unit)
        del unit
```

3-in-3-out

Multiple input (or output)

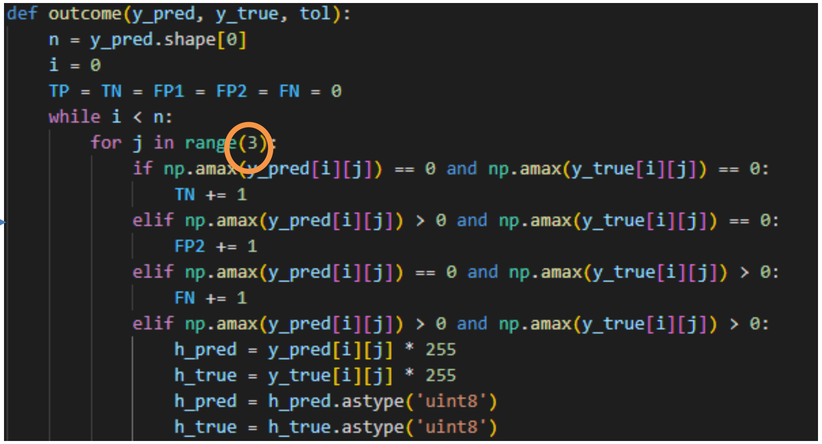
The devil is in the details

train_TrackNet3.py



```
def outcome(y_pred, y_true, tol):
    n = y_pred.shape[0]
    i = 0
    TP = TN = FP1 = FP2 = FN = 0
    while i < n:
        for j in range(1):
            if np.amax(y_pred[i][j]) == 0 and np.amax(y_true[i][j]) == 0:
                TN += 1
            elif np.amax(y_pred[i][j]) > 0 and np.amax(y_true[i][j]) == 0:
                FP2 += 1
            elif np.amax(y_pred[i][j]) == 0 and np.amax(y_true[i][j]) > 0:
                FN += 1
            elif np.amax(y_pred[i][j]) > 0 and np.amax(y_true[i][j]) > 0:
                h_pred = y_pred[i][j] * 255
                h_true = y_true[i][j] * 255
                h_pred = h_pred.astype('uint8')
                h_true = h_true.astype('uint8')
```

1-in-1-out



```
def outcome(y_pred, y_true, tol):
    n = y_pred.shape[0]
    i = 0
    TP = TN = FP1 = FP2 = FN = 0
    while i < n:
        for j in range(3):
            if np.amax(y_pred[i][j]) == 0 and np.amax(y_true[i][j]) == 0:
                TN += 1
            elif np.amax(y_pred[i][j]) > 0 and np.amax(y_true[i][j]) == 0:
                FP2 += 1
            elif np.amax(y_pred[i][j]) == 0 and np.amax(y_true[i][j]) > 0:
                FN += 1
            elif np.amax(y_pred[i][j]) > 0 and np.amax(y_true[i][j]) > 0:
                h_pred = y_pred[i][j] * 255
                h_true = y_true[i][j] * 255
                h_pred = h_pred.astype('uint8')
                h_true = h_true.astype('uint8')
```

3-in-3-out

Other Loss function

- Mean square error
- α-balanced cross-entropy loss
- Focal loss

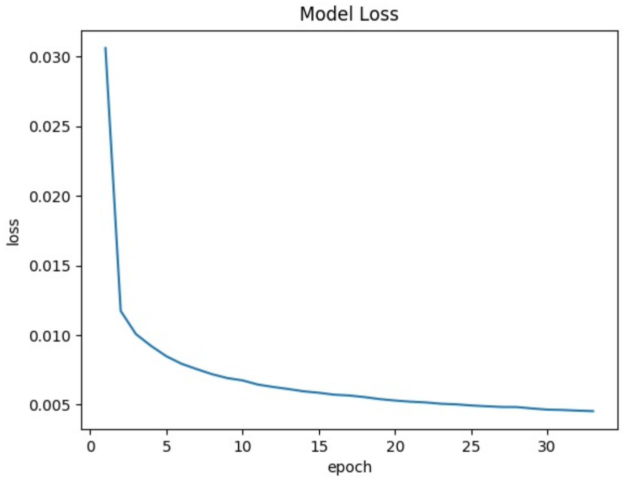
https://github.com/umbertogriffo/focal-loss-keras/blob/master/src/loss_function/losses.py

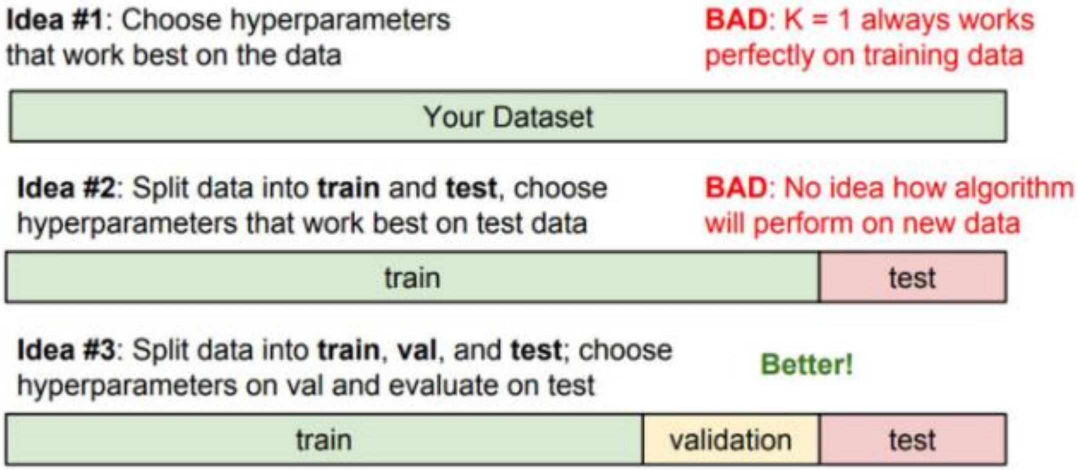
- Weighted Hausdorff distance

https://github.com/N0vel/weighted-hausdorff-distance-tensorflow-keras-loss/blob/master/weighted_hausdorff_loss.py

Training tips

- Learning rate
- Batch size
- Data splitting
- Visualize loss





Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train | test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train | validation | test

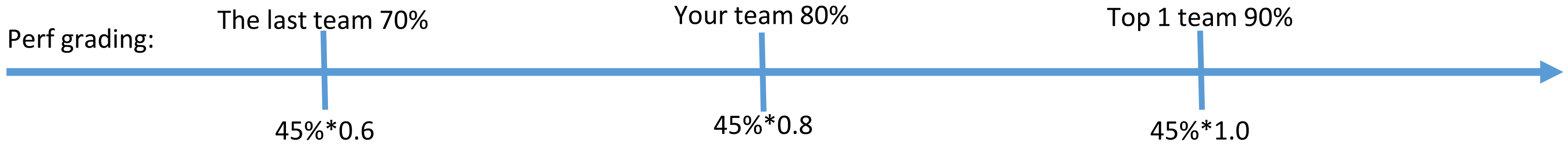
Score

How we score

- Label (30%)
- Model performance (45%)

  45% for Performance benchmark: (accuracy + precision + recall) / 3

- Demo & QA(25%)



Perf grading:

The last team 70%

45%*0.6

Your team 80%

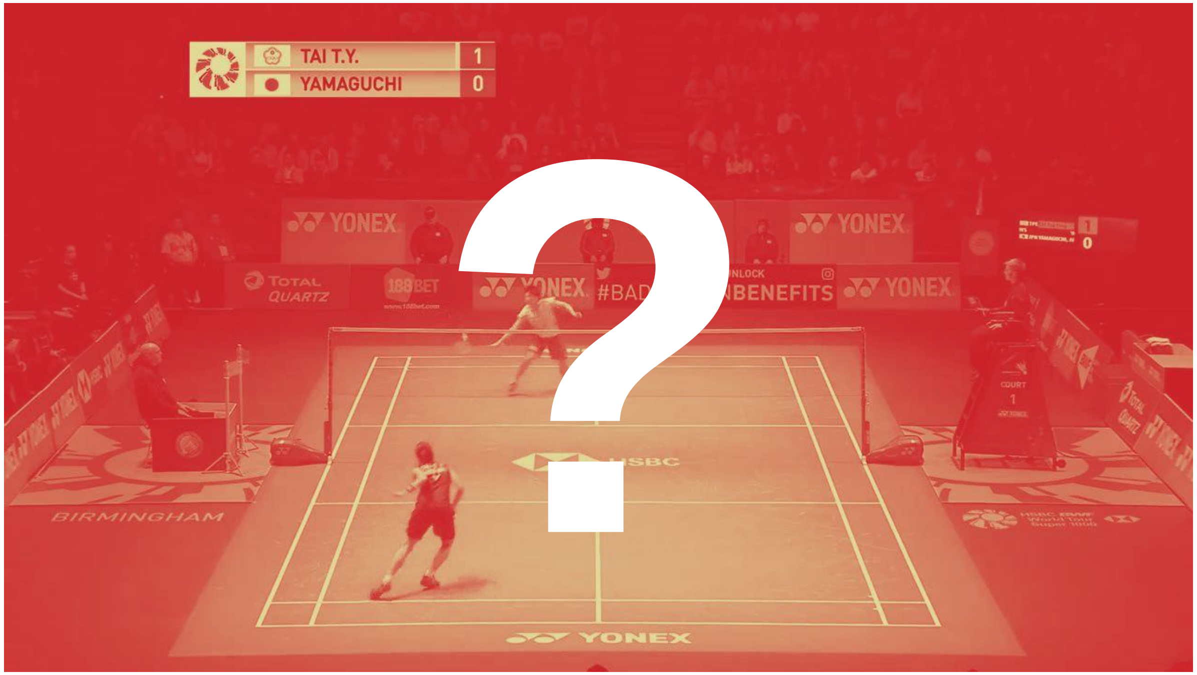
45%*0.8

Top 1 team 90%

45%*1.0

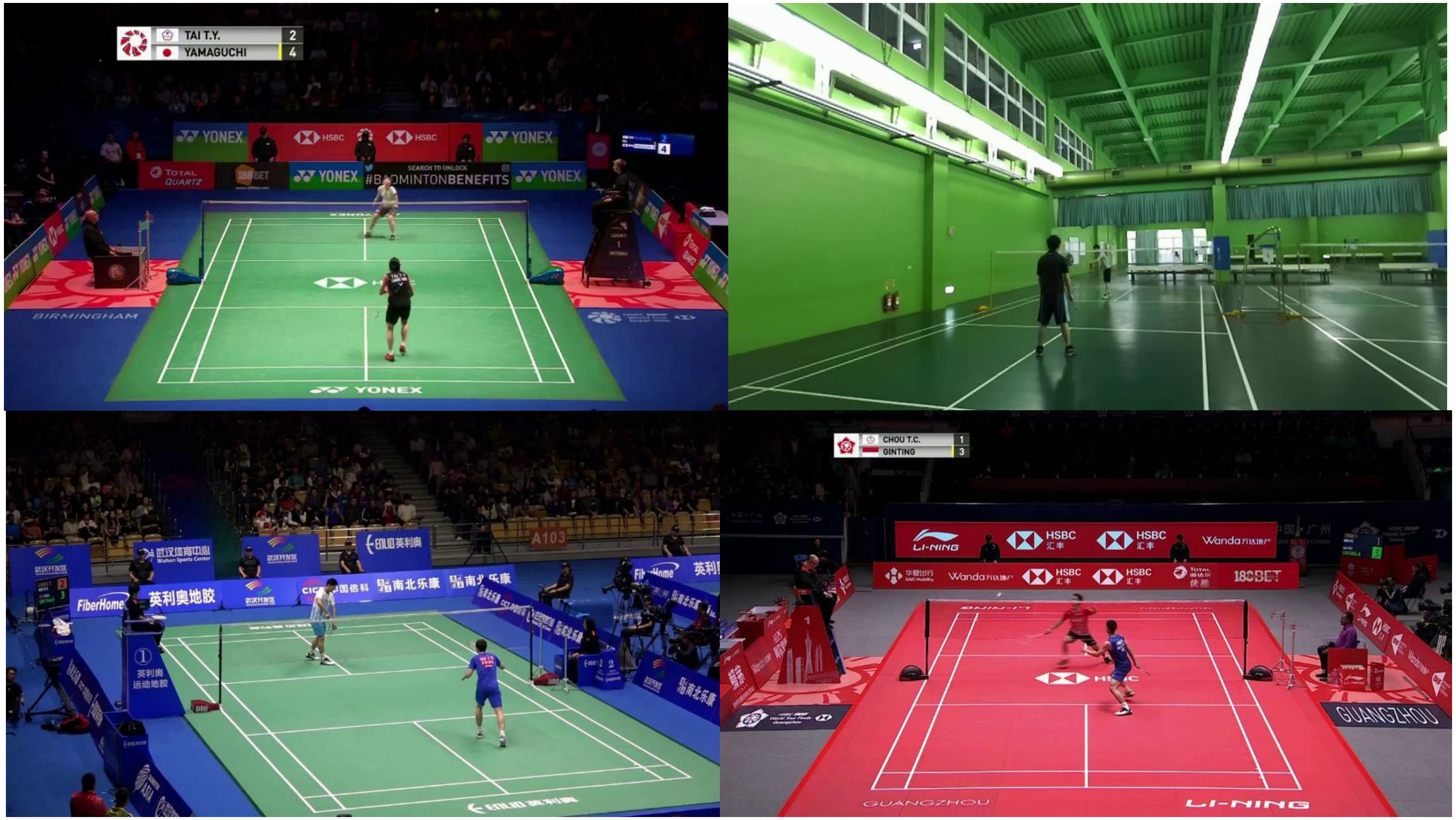
Test Data

Match not in train data



Test Data

Maybe …

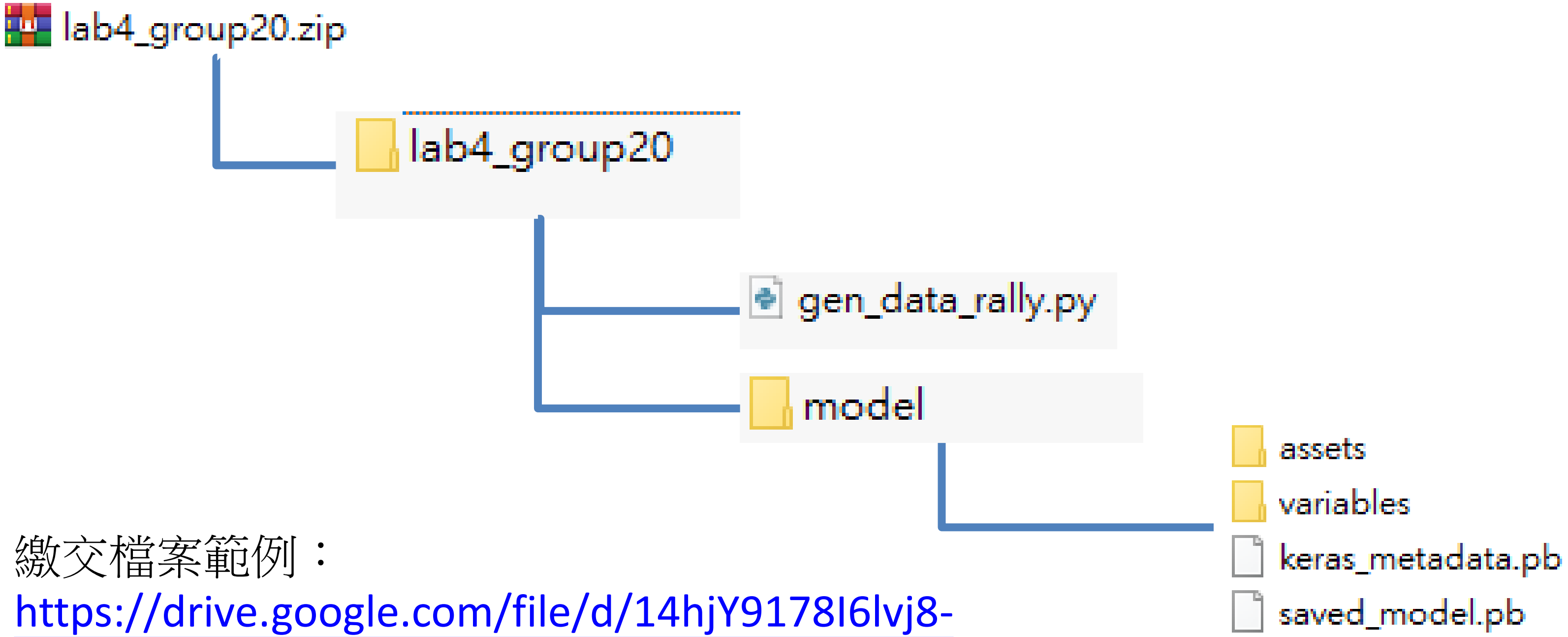


Deadline

- Create data
  - Before 12/04 23:59 (half less per day, e.g. 100 → 50 → 25 …)
- Demo
  - On 12/15 after TA's lecture (about 19:30)
- Each group needs at least 1 member to demonstrate
- Lab Submission
  - Before 12/15 23:59
- If you have any problems about submission & demo please contact TAs before deadline. ([深度學習智慧應用] 主旨開頭請加這個)
  - bobhsiao0306.cs10@nycu.edu.tw
  - jason880102.cs10@nycu.edu.tw

繳交檔案格式

- lab4_group{GROUP_NUMBER}.zip
- EX：lab4_group20.zip



lab4_group20.zip

lab4_group20

gen_data_rally.py

model

assets

variables

keras_metadata.pb

saved_model.pb

繳交檔案範例：
https://drive.google.com/file/d/14hjY9178I6lvj8-9D4Hc7FUKO65jDvwr/view?usp=sharing

Further Reading

TrackNet : A Deep Learning Network for Tracking High-speed and Tiny Objects in Sport Applications
https://people.cs.nctu.edu.tw/~yi/TechReports/TrackNet.v1_Final_NOL.pdf

High Performance Visual Tracking with Siamese Region Proposal Network (CVPR 2018)
http://www.zhengzhu.net/upload/P6938bc861e8d4583bf47d47d64ed9598.pdf

Object as points (CVPR 2019)
https://arxiv.org/pdf/1904.07850.pdf

Locating Objects Without Bounding Boxes (CVPR 2019)
https://arxiv.org/pdf/1806.07564.pdf