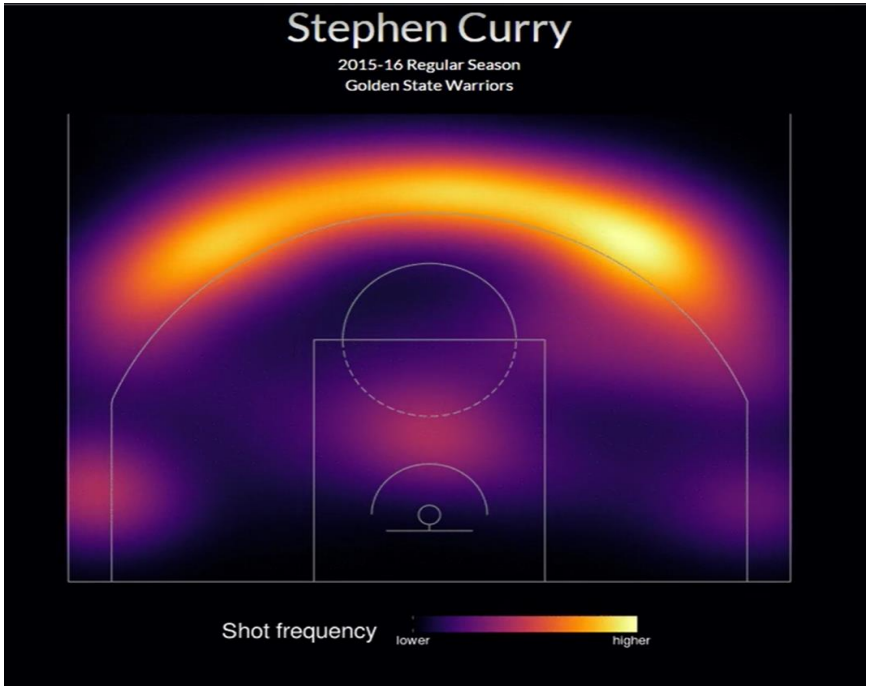
深度學習智慧應用
Lab 1: TrackNet

Network Optimization Lab (NOL)
Department of Computer Science
National Yang Ming Chiao Tung University

Outline

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

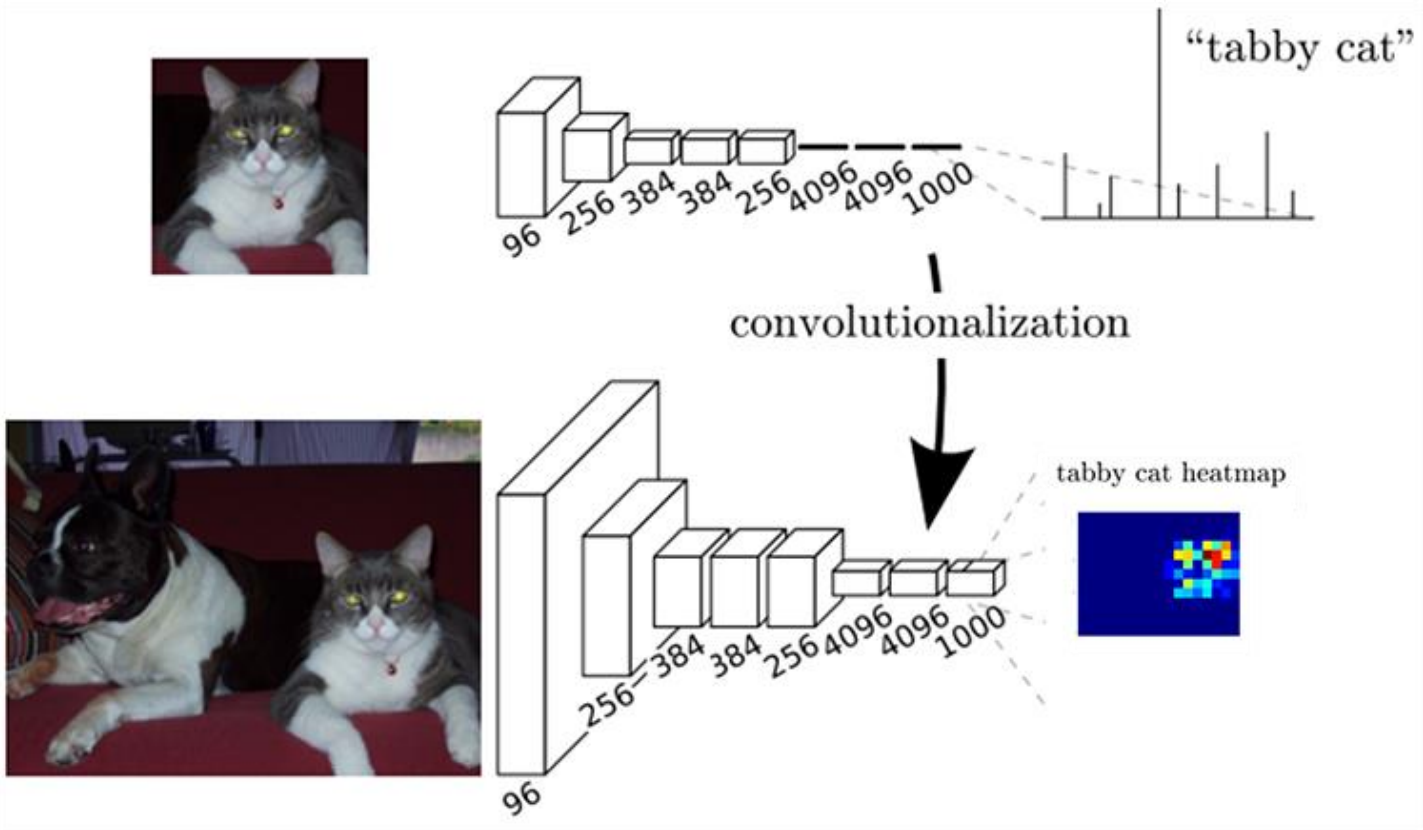
HeatMap


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

Semantic Segmentation

Pixel-wise classification

FCN


"tabby cat"
96
256
384
384
256
4096
4096
1000
convolutionalization
tabby cat heatmap
96
256
384
384
256
4096
4096
1000

FCN

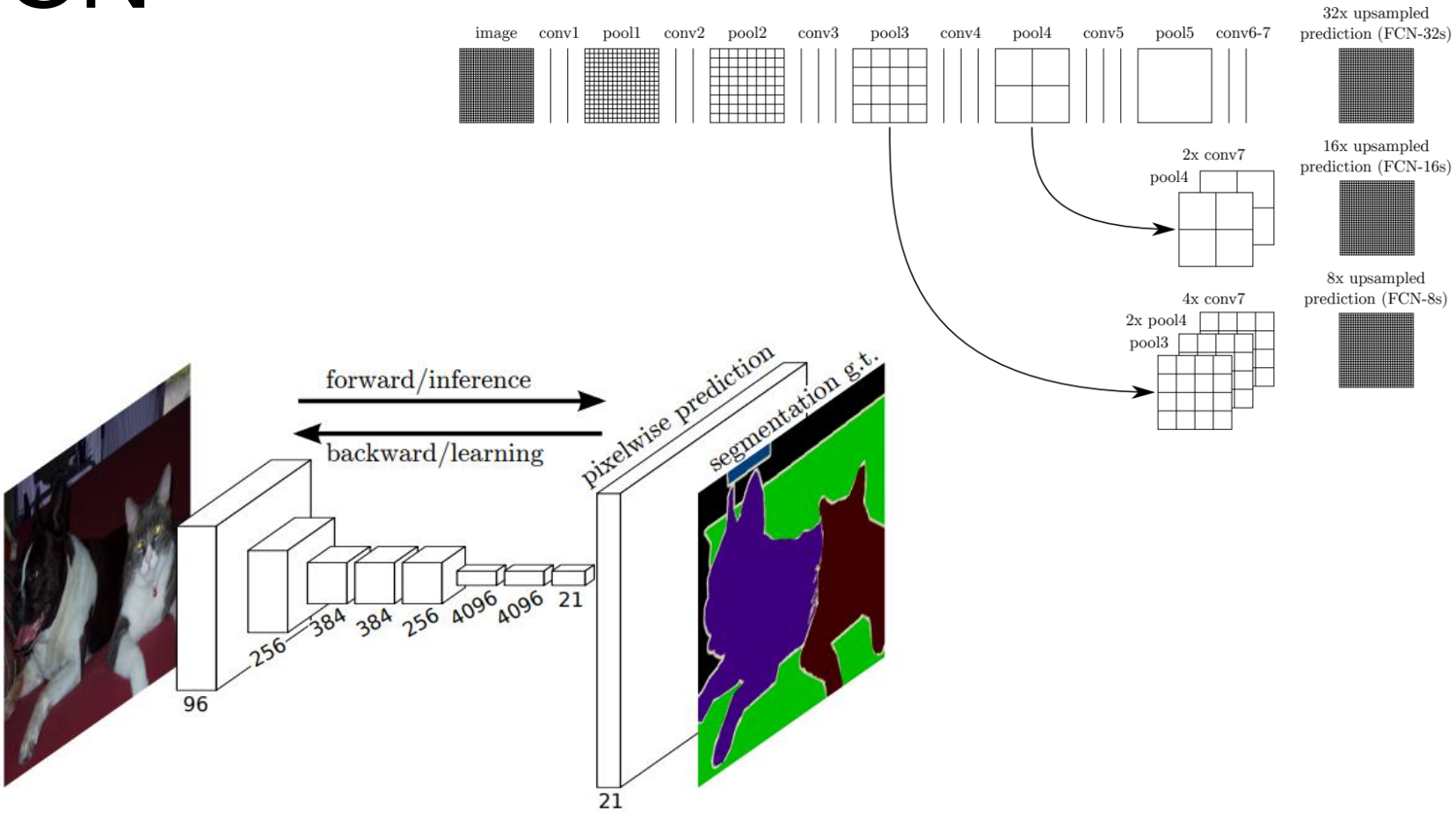
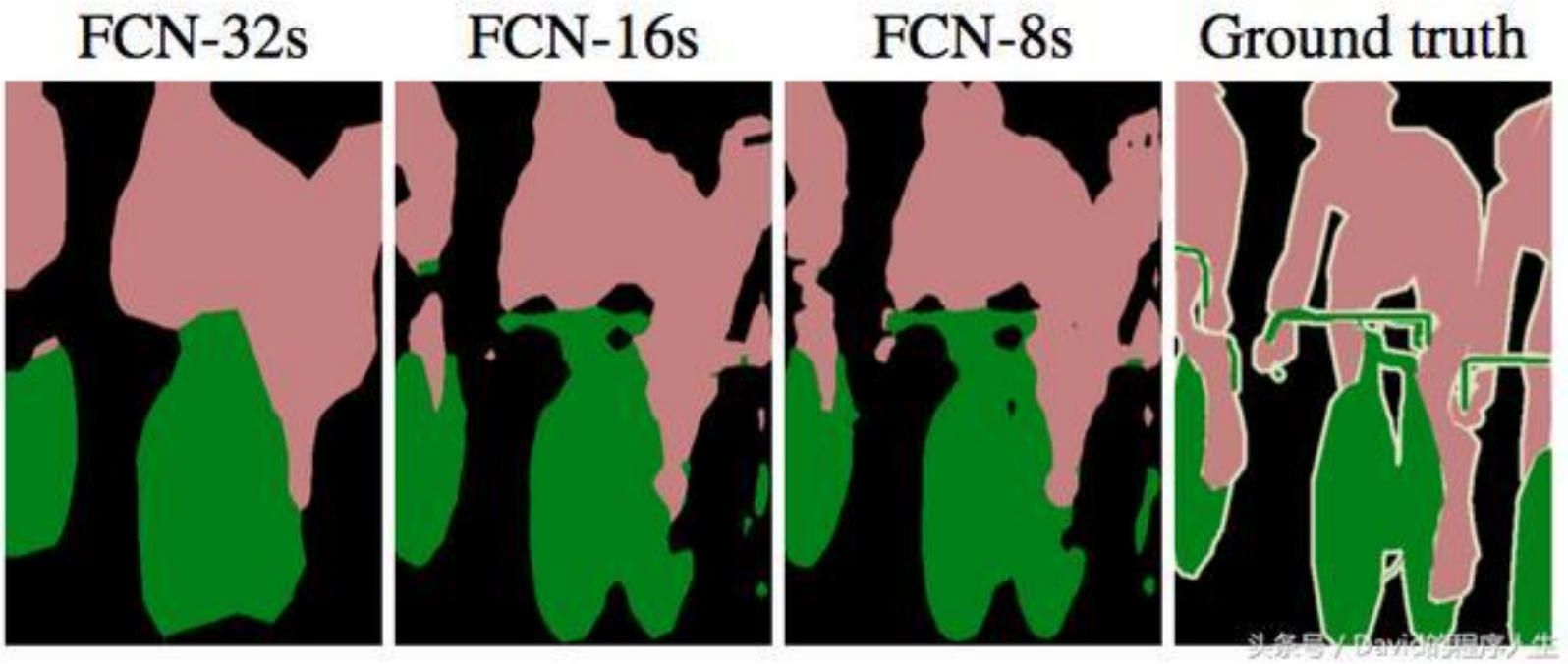


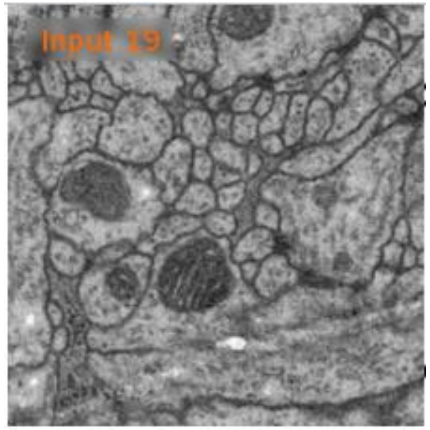
image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)
forward/inference
backward/learning
pixelwise prediction
segmentation g.t.
96
256
384
384
256
4096
4096
21
21

FCN

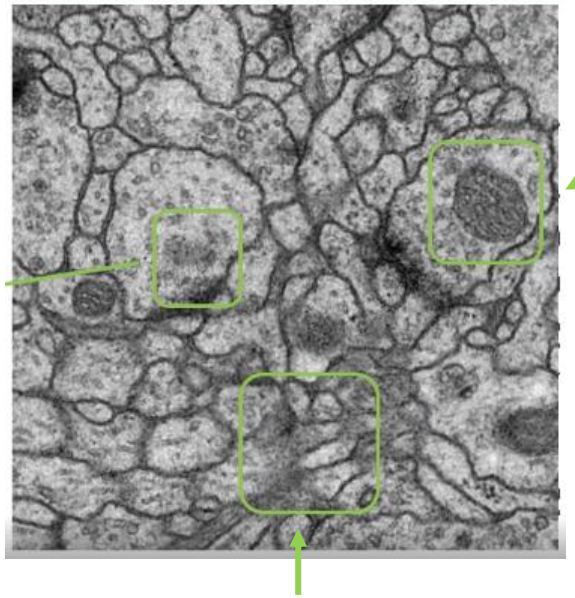


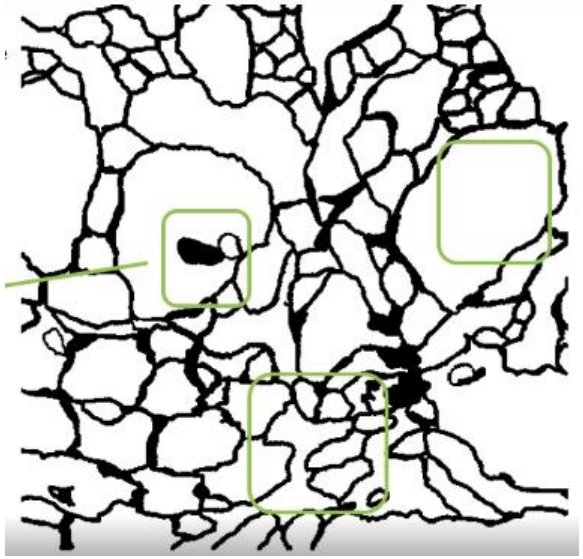
U-Net

Modify and extend the architecture of fully convolutional network such that it works with very few training images and yields more precise segmentations
Won the ISBI cell tracking challenge 2015 in these categories by a large margin



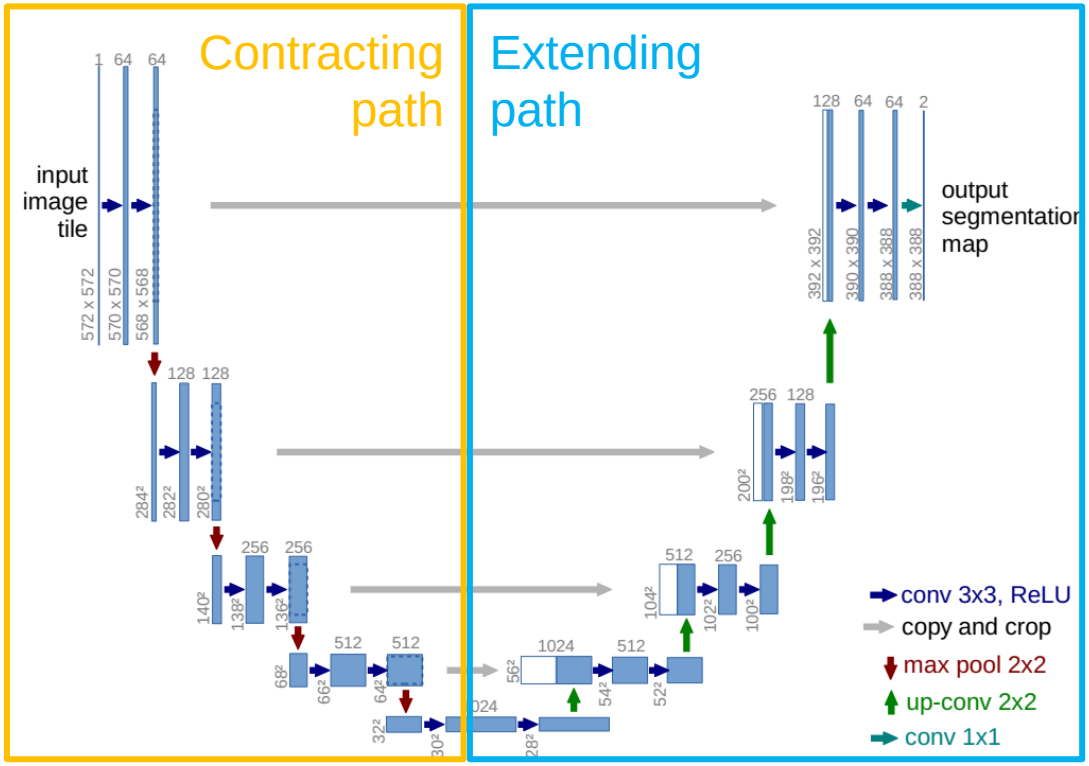
U-Net


Other cell compartments
Structures with very low contrast
Fuzzy membranes



U-Net Architecture

Concatenation with the correspondingly cropped feature map from the contracting path



EM segmentation challenge

The training data is a set of 30 images
Cells (white) and membranes (black)

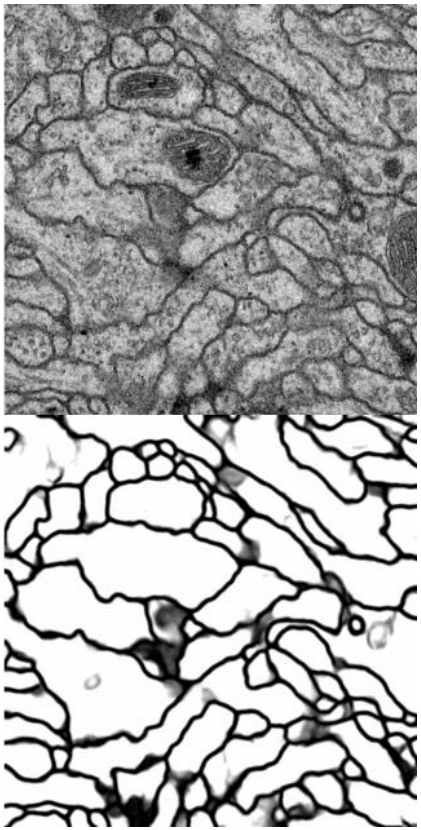


Table 1. Ranking on the EM segmentation challenge [14] (march 6th, 2015), sorted by warping error.

Rank	Group name	Warping Error	Rand Error	Pixel Error
	** human values **	0.000005	0.0021	0.0010
1.	u-net	**0.000353**	0.0382	0.0611
2.	DIVE-SCI	0.000355	0.0305	0.0584
3.	IDSIA [1]	0.000420	0.0504	0.0613
4.	DIVE	0.000430	0.0545	**0.0582**
⋮				
10.	IDSIA-SCI	0.000653	**0.0189**	0.1027

TrackNet

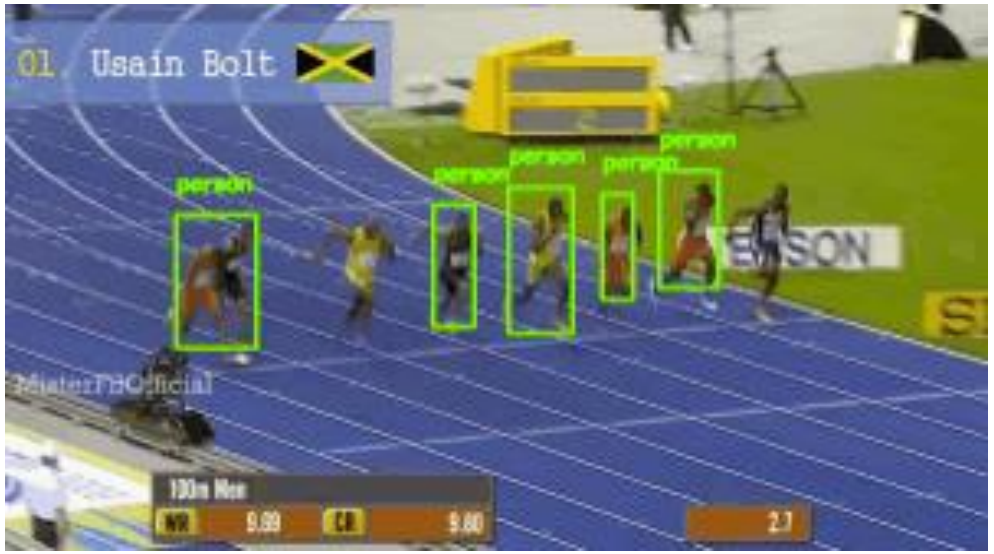
https://www.pyimagesearch.com/2018/10/29/multi-object-tracking-with-dlib/

Multiple Object Tracking

Single Object Tracking

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

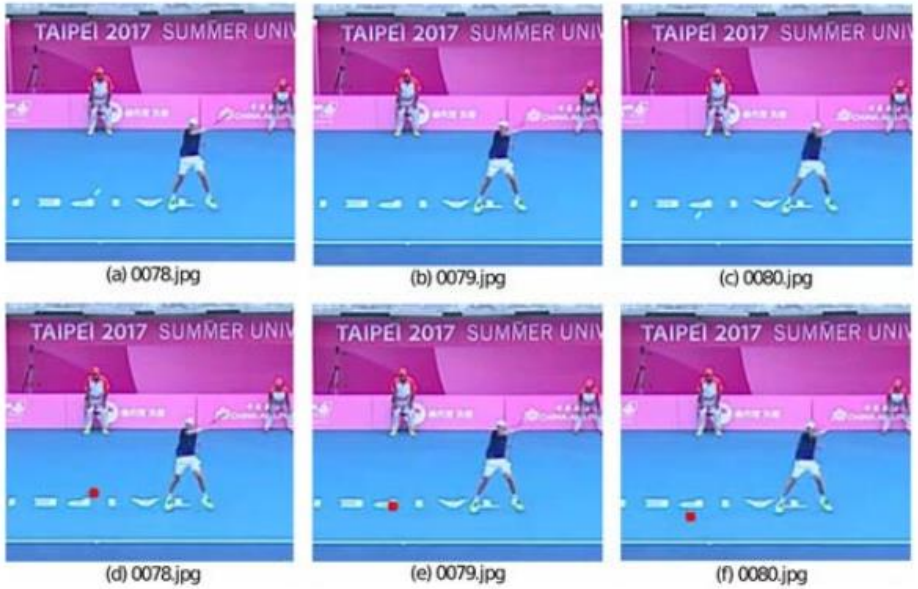
https://nanonets.com/blog/object-tracking-deepsort/



TrackNet

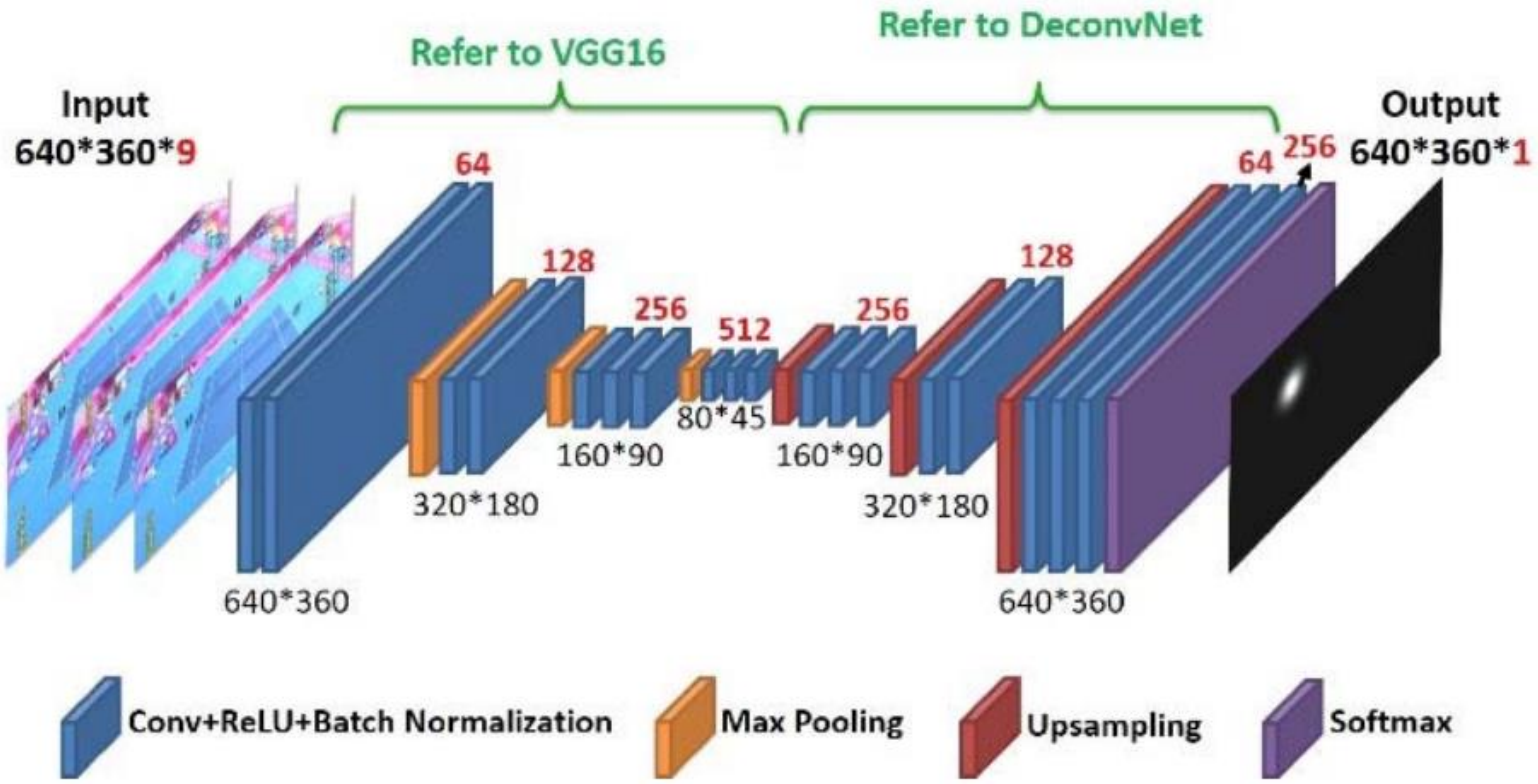
https://people.cs.nctu.edu.tw/~yi/TechReports/TrackNet.v1_Final_NOL.pdf

標記網球位置的時，網球呈現軌跡拖延的案例



網球影像無法被識別

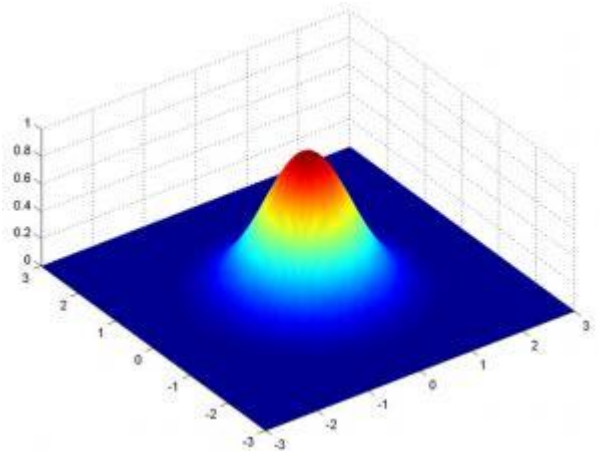
TrackNet

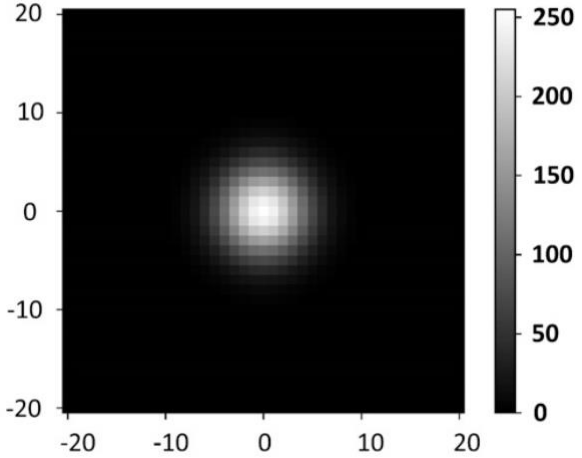

Refer to VGG16
Refer to DeconvNet
Input
640*360*9
Output
640*360*1
64
128
256
512
256
128
64
256
640*360
320*180
160*90
80*45
160*90
320*180
640*360
Conv+ReLU+Batch Normalization
Max Pooling
Upsampling
Softmax

TrackNet

- Ground Truth

$$G(x,y) = \left\lfloor \left(\frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \right) (2\pi \cdot \sigma^2 \cdot 255) \right\rfloor$$





TrackNet

- Performance

Lab 1

Congratulation!

You don’t have to

- Design model from scratch
- Write Data Generater
- Write dataloader

Outline

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

System Environment

Suggest Environment

- Ubuntu 20.04
- Python 3.8.10 / pandas / numpy / Sklearn
- Pytorch 1.13.1 / torchvision 0.14.1 / CUDA 11.7 / Cudnn 8.5.0
- Opencv 4.2.0

Use “ == “ to install a specific version
EX : pip3 install h5py==2.9.0

Code

https://drive.google.com/drive/folders/1-lJH_49-zeNE_-97rC_OR5L5meyDS6az?usp=drive_link

- This link contains all the code needed for this lab.
- You should be able to run the entire project without modifying any code.

However, you're still encouraged to make some code modifications for improvement.

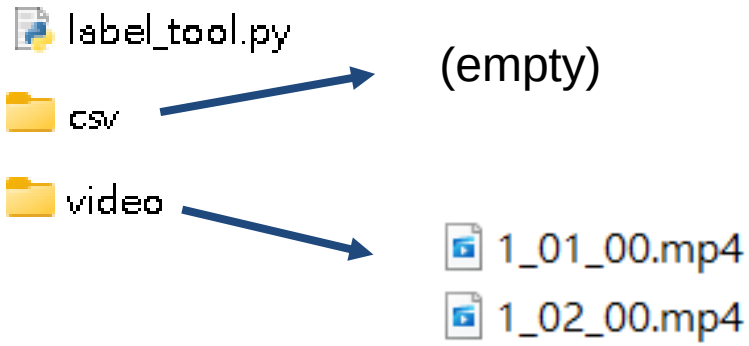
Improvement scores still count! :)

Data Label

Label Tool

Command Usage:

python3 label_tool.py [video_path]



It is recommended to organize your files like this before labeling

Label Tool Output

Input : mp4, output : csv

The output csv will look like:

1	Frame	Visibility	X	Y
2	0	0	0	0
3	1	1	652	257
4	2	1	652	257

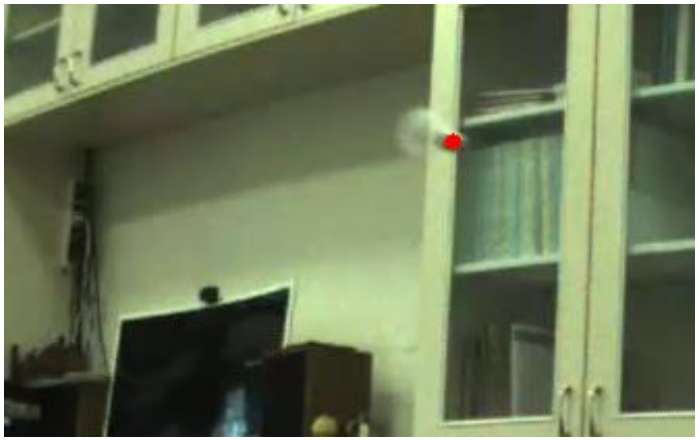
Frame represent the frame of video.
Visibility means the shuttlecock is visible or not at this frame(0 : invisible, 1 : visible).
X, Y is the coordinate of shuttlecock. If shuttlecock is invisible now, then X, Y is 0.

Label Tool Usage

- Last Frame: z
- Next Frame: x
- Last 50 Frame: d
- Next 50 Frame: f
- Label: left click
- Zoom: a (centered on the last label)
- Clear Label: c
- Quit & Save: q
- Quit without saving: Esc

Labeling Rules

1. Please mark the head of shuttlecock
2. No need to mark the shuttlecock if it is blocked and not visible
3. No need to mark the ball if it is stationary or held by a person



Your folders should look like this after labeling

Data Labeling Task

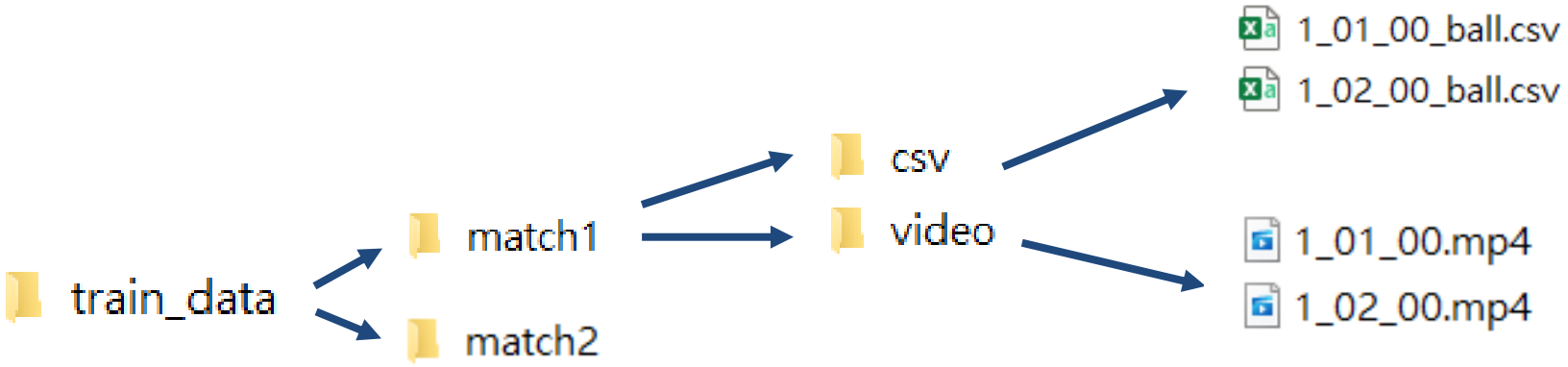
- Unlabeled data :
https://drive.google.com/drive/folders/1lB0ooeKV4R2w5DnyDKOuKwjwDo32YxYj?usp=drive_link
- Every group needs to label 4 videos
- Upload your csv files to e3 in a zip file (only compress csv folder)

Data Generate

Arrange the structure as below

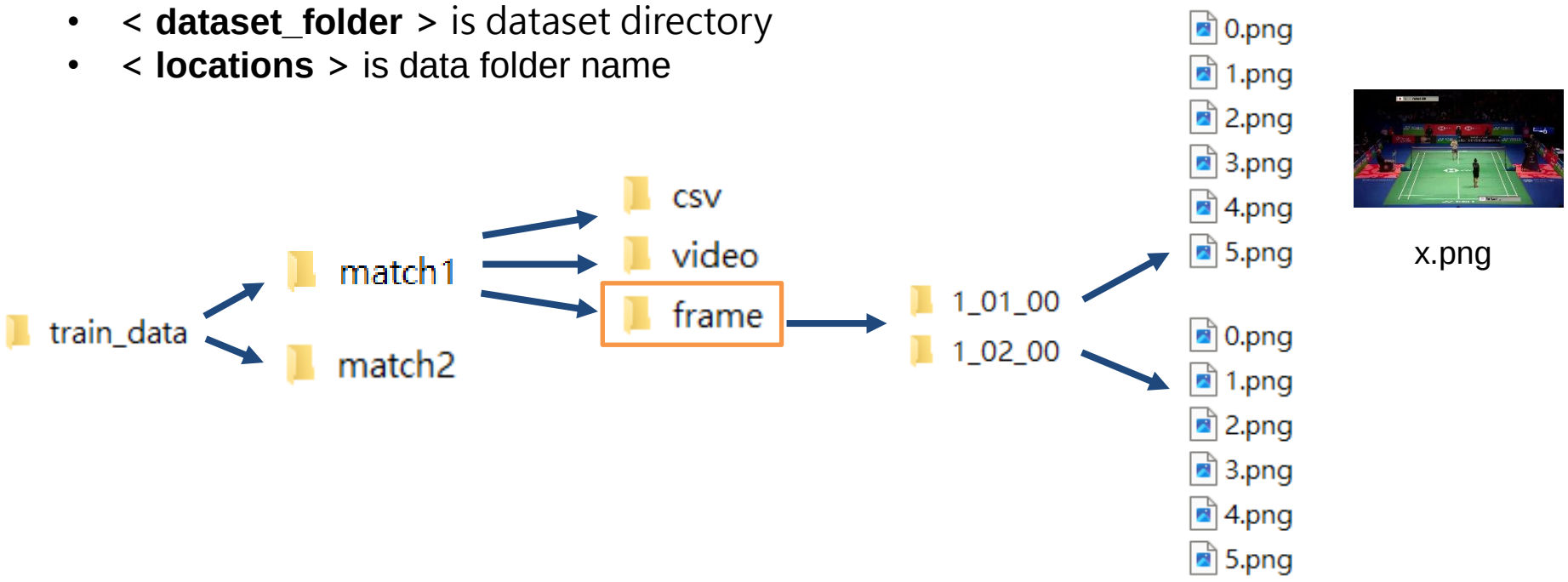
- Use the data you labeled in your labeling task

- You can also use these additional data that are already labeled:
https://drive.google.com/drive/folders/1DD7_AQUcjE1eUoAe32U_ixNaT1UubG5g?usp=drive_link



frame_generator.py

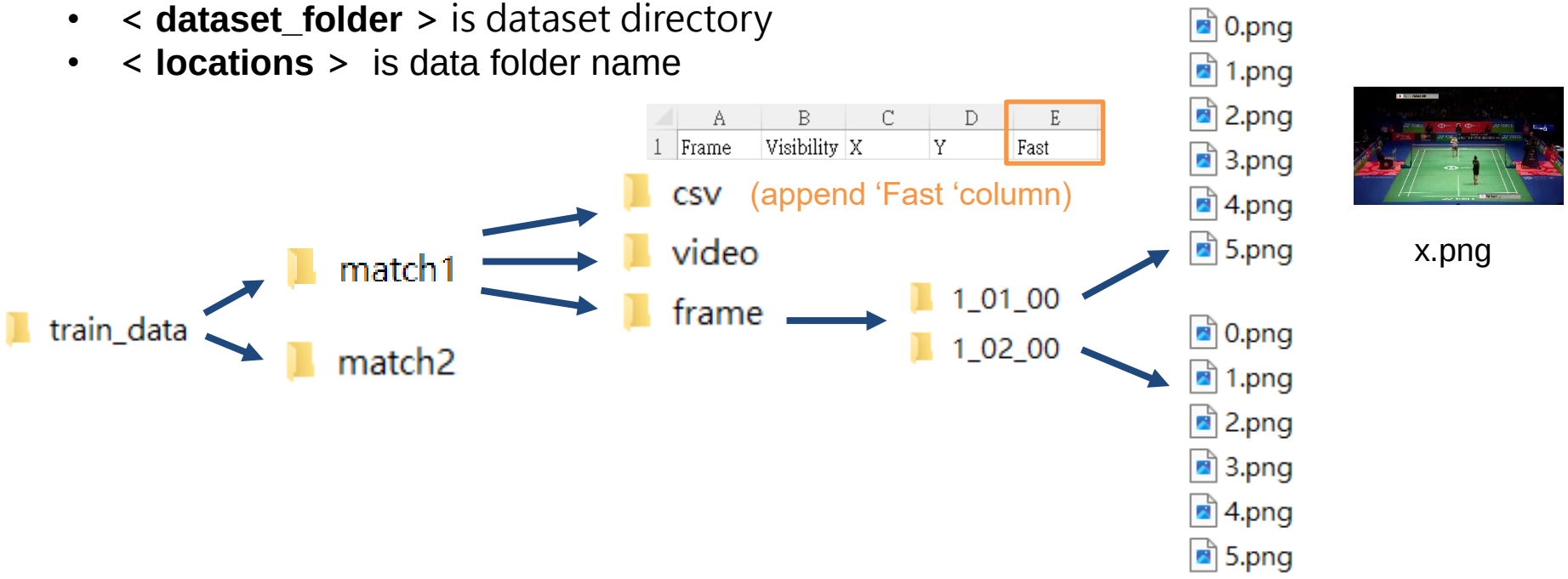
- python3 frame_generator.py --dataset_folder --locations
- < **dataset_folder** > is dataset directory
- < **locations** > is data folder name



EX : python3 frame_generator.py --dataset_folder train_data --locations match_1 match_2

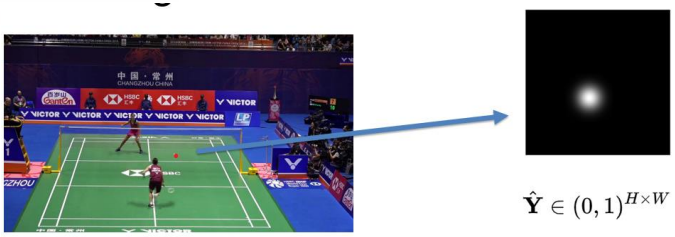
detect_fast.py

- python3 detect_fast.py --dataset_folder --locations
- < **dataset_folder** > is dataset directory
- < **locations** > is data folder name

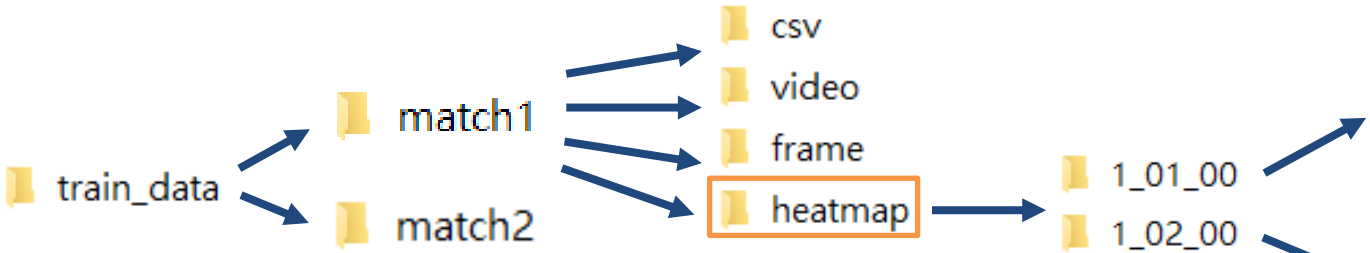


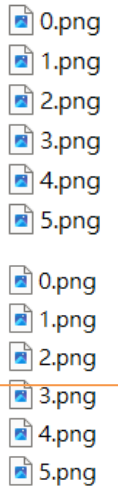
EX : python3 detect_fast.py --dataset_folder train_data --locations match_1 match_2

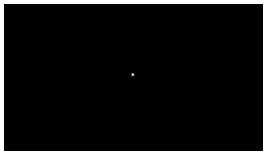
preprocess.py



- python3 preprocess.py --dataset_folder --locations [--info_csv] [--result_csv] --test
- < **dataset_folder** > is dataset directory
- < **locations** > is data folder name
- < **info_csv** > csv file record location and the number of frames
- < **result_csv** > csv file record model's evaluation results
- < **test** > 0 means train data, 1 means test data







x.png

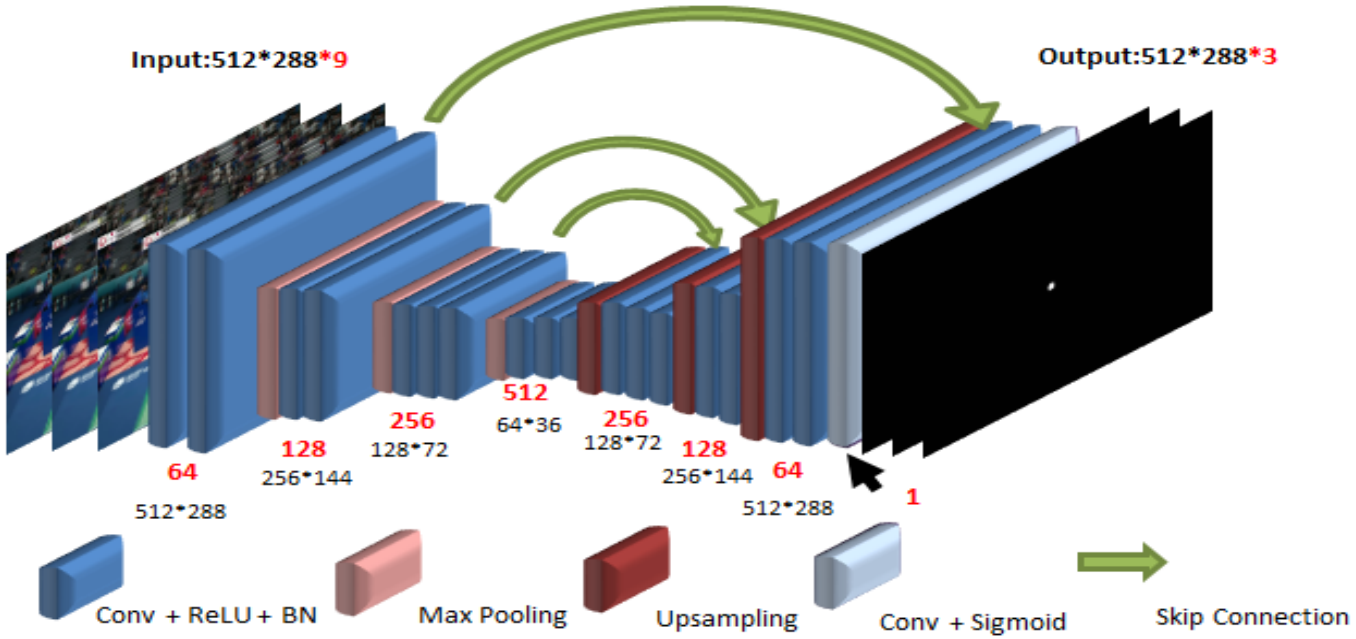
tracknet_train_list_x_10.csv : list of train data image path
tracknet_train_list_y_10.csv : list of ground truth heatmap image path
tracknet_train_list_z_10.csv : list of labels indicating the "Fast" for each frame (1 for normal speed, 5 for fast movement)

EX : python3 preprocess.py --dataset_folder train_data --locations match_1 match_2 --info_csv train_data_info.csv --result_csv train_data_result.csv --test 0

Model architecture

TrackNet.py

Modify model by yourself ! (If you need)



Base model

Loss function

Weighted Binary Cross Entropy (WBCE)

WBCE is a loss function that adds weights to balance class importance. It helps when one class is more common than the other.

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

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

train.py

- python3 train.py --batchsize --epochs --lr --tol [--load_ weight] --save_weights [--result_csv]
  - < **batchsize**> default=8
  - < **epochs**> default=30
  - < **lr**>default=1
  - < **tol**>default=4, tolerance value of true positive
  - < **load_ weight** > weight you had trained before
  - < **save_weights** > TrackNet weight after this training, default='TrackNet10'
  - < **result_csv** > csv file record model's evaluation results

EX :
python3 train.py --result_csv train_data_result.csv

Pretrained Weight

- python3 train.py --batchsize --epochs --lr --tol [--load_ weight] --save_weights [--result_csv]
  - < **batchsize**> default=8
  - < **epochs**> default=30
  - < **lr**>default=1
  - < **tol**>default=4
  - < **load_ weight** > weight you had trained before
  - < **save_weights** > TrackNet weight after this training, default='TrackNet10'
  - < **result_csv** > csv file record model's evaluation results

You can use our pretrained weight to fine-tune the model with new dataset:
https://drive.google.com/drive/folders/1BdP9DEcoxU_AZXOO4Fg71fkyEHmBT44Z?usp=drive_link

train.py result

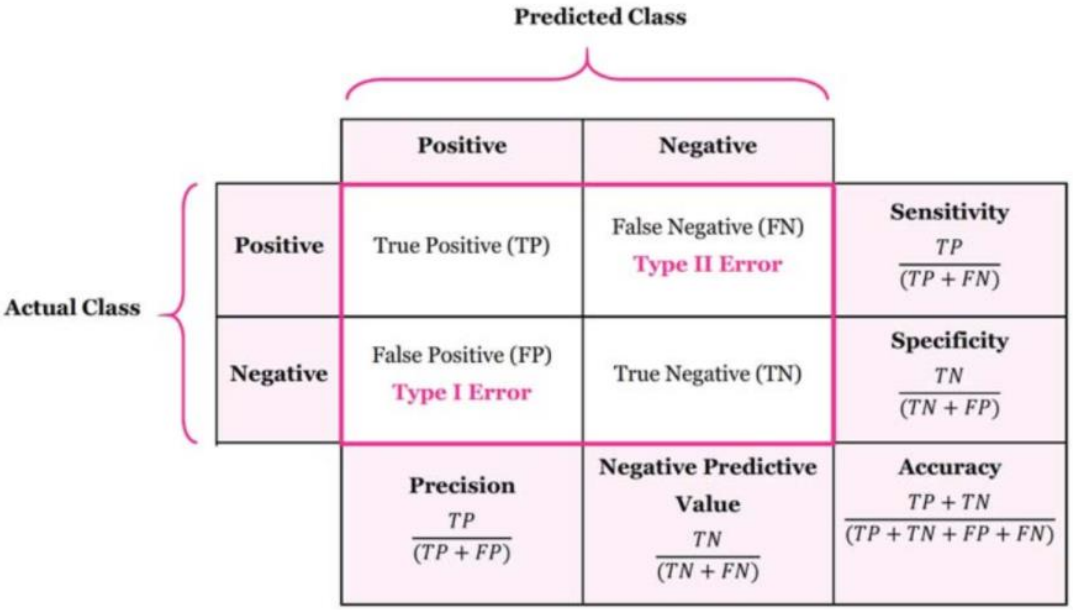
```
Train Epoch" 1 [12/1716 (1%)]   Loss : 0.000212
Fast means = 1.0
Batch Weighted: 0.5841666666666666
Train Epoch" 1 [24/1716 (1%)]   Loss : 0.000115
Fast means = 1.0
Batch Weighted: 0.7875
Train Epoch" 1 [36/1716 (2%)]   Loss : 0.000093
Fast means = 1.0
Batch Weighted: 1.0833333333333335
Train Epoch" 1 [48/1716 (3%)]   Loss : 0.000150
Fast means = 1.0
Batch Weighted: 0.8275000000000001
Train Epoch" 1 [60/1716 (3%)]   Loss : 0.000118
Fast means = 1.0666666666666667
Batch Weighted: 0.5349999999999999
Train Epoch" 1 [72/1716 (4%)]   Loss : 0.000067
Fast means = 1.0
Batch Weighted: 0.5666666666666668
Train Epoch" 1 [84/1716 (5%)]   Loss : 0.000044
Fast means = 1.0
Batch Weighted: 0.8249999999999998
Train Epoch" 1 [96/1716 (6%)]   Loss : 0.000142
Fast means = 1.0666666666666667
Batch Weighted: 0.5808333333333334
Train Epoch" 1 [108/1716 (6%)]  Loss : 0.000072
Fast means = 1.0
Batch Weighted: 0.5941666666666666
```

```
======================Evaluate=======================
Number of true positive: 11631
Number of true negative: 4724
Number of false positive FP1: 134
Number of false positive FP2: 164
Number of false negative: 507
Loss: 2.7240246705578458e-05
Accuracy: 0.953088578088578
Precision: 0.975018861597787
Recall: 0.958230350963915
=====================================================
```

Evaluation standard

Metric

- Accuracy
- Precision
- Recall



- 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.

test.py

You need to perform the same steps as in the training process first.
1. frame_generator.py
2. detect_fast.py
3. preprocess.py

get tracknet_test_x.csv, trscknet_test_y.csv, tracknet_test_z.csv

- python3 test.py --batchsize --lr --tol --load_ weight --location [--info_csv]

  - < **batchsize**> default=8
  - < **lr**>default=1
  - < **tol**>default=4
  - < **load_ weight** > weight you had trained before
  - < **location**> test data folder name
  - < **info_csv**> csv file record testing results

EX :
python3 test.py --load_weight TrackNet10_30.tar --location match_test _1 --info_csv match_1_test_result.csv

test.py result

```
Weight: TrackNet10_30.tar
=====================Evaluate=====================
Number of true positive: 10465
Number of true negative: 1123
Number of false positive FP1: 465
Number of false positive FP2: 7
Number of false negative: 4120
Number of fast true positive: 0
Number of fast true negative: 0
Number of fast false positive FP1: 0
Number of fast false positive FP2: 0
Number of fast false negative: 0
Accuracy: 0.7161928306551298
Precision: 0.9568437414281796
Recall: 0.7175179979430922
F1-score: 0.8200767964893033
fast Accuracy: 0
fast Precision: 0
fast Recall: 0
==================================================
```

We will not consider the 'fast' results.

Testing Data

- Testing data will be uploaded one week later via e3
- You should use them to evaluate your model and include the results in report

predict.py

- python3 predict.py --video_name --load_weight --output_dir

  - < **video_name**> video path
  - < **load_ weight** > input model weight for predict
  - < **output_dir**> output video directory

EX :
python3 predict.py --video_name match_test_1 /video/1_02_01.mp4 --load_weight TrackNet10_30.tar --output_dir output

predict.py result

1_02_01_ball.csv

1_02_01_with_TrackNet10_30.mp4

Predict video

Ways to improve

Model Architecture

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

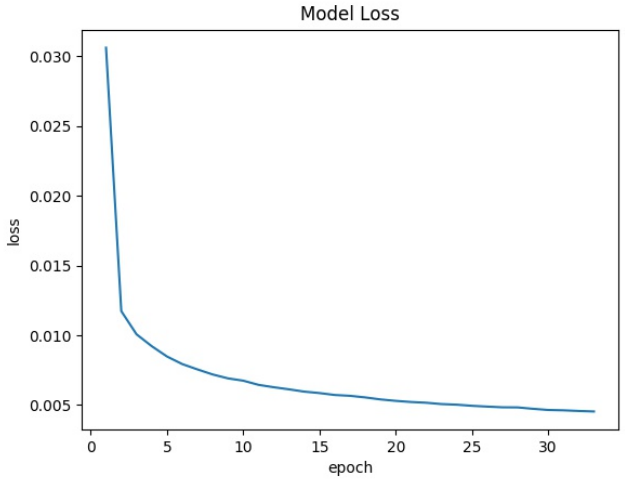
It may cause GPU out of memory …

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

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

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

Better!

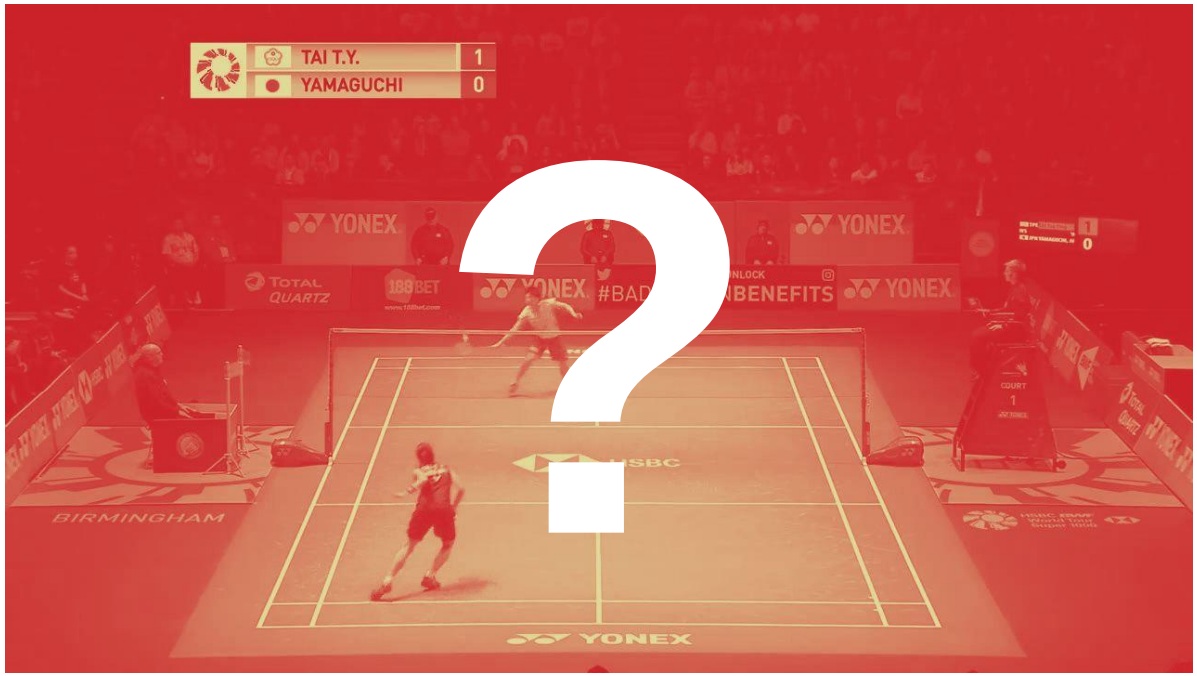
Grading

How we score

- Data Label (10%)
- Successfully executable (40%)
  (including training and testing)
- Improvement (30%)
- Performance (20%)
  The testing data will be announced during the demo. Each group will use its model to predict, and the F1 score will be used to evaluate performance.

Test Data

Match not in train data



Test Data

Maybe …

Label Data Submission Requirements

- Upload the labeled CSV files, with all files for each group compressed into a zip file.

Name the zip file as Lab1_LabelData_{group number}.zip

Lab Submission Requirements

- Report (.pdf)
  - Implementation Results
    - Document the results based on the Training and Testing processes taught in the course
  - Improvement Methods and Results
    - Describe the methods used to improve the model.
    - Report the performance outcomes after implementing improvements.
- Code

Name the zip file as Lab1_Report_{group number}.zip

Lab Resources

- You can find all the provided code and data here:

https://drive.google.com/drive/folders/1TrvXqDRoDVZbZjGixdl5_olNSFmfaq_v?usp=sharing

Deadline

- Label Data Submission
  - On 10/21 23:59 (10% less per day)
- Lab Submission
  - On 10/28 23:59 (10% less per day)
- Demo
  - On 10/29 after TA‘s lecture

- If you have problems with the submission & demo, please contact TAs before the deadline.
  - jeremy926.cs13@nycu.edu.tw & clcheng.cs11@nycu.edu.tw ([深度學習智慧應用] 主旨開頭請加這個)

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