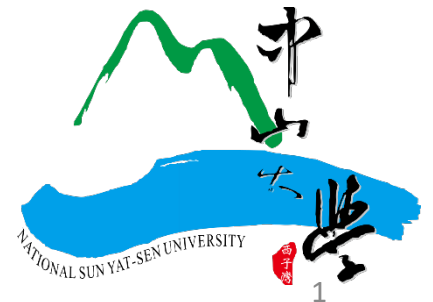


# Computer Vision Assignment 2

魏家博 (Chia-Po Wei)

Department of Electrical Engineering  
National Sun Yat-sen University





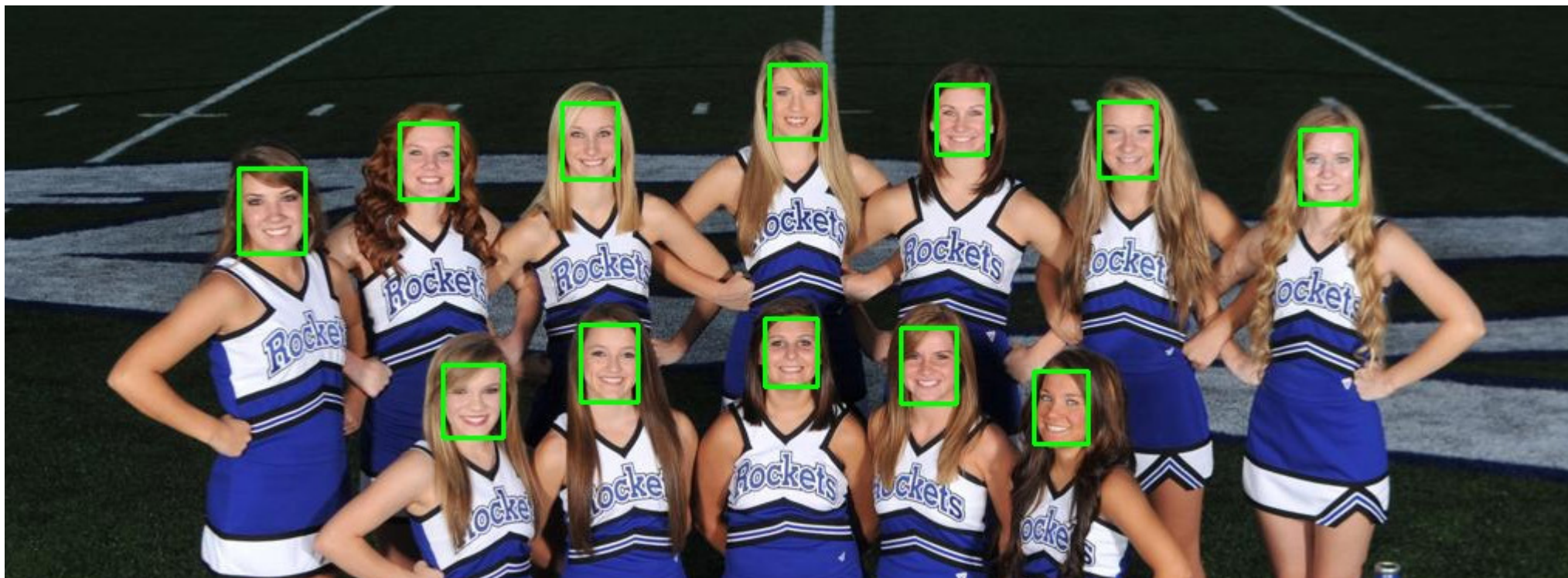
# Outline

Assignment 2a: Image Processing & Face Detection

Assignment 2b: Data Statistics

# Assignment 2a: Image Processing

- 讀取 annotation.pkl 中定界框(bounding box)之資訊，將定界框畫在原圖，結果如下圖。  
接著將所有綠色框區域裁切下來存成檔案至指定目錄。



## Steps for Assignment 2a 於原始影像畫出臉之定界框

### Step 2a.1: Open [assignment\\_2a.ipynb](#)

Ensure that [assignment\\_2a.ipynb](#), [annotation.pkl](#), [data\\_dir/](#) are in the same folder.

Step 2a.2: Implement the following pseudo code to draw bounding boxes of faces in function `draw_bbox()`

```
for img_name in annotate_dict:
    # get the image path from img_name
    # use cv2.imread() to read from the image path
    for bbox in annotate_dict[img_name]:
        # draw bounding boxes on the image
        # use plt.imshow() to show the image
```

- Notice that in Step 2a.2

`annotate_dict` is a python dictionary





- Use `for key in annotate_dict:` to access the keys of `annotate_dict`
- Each key is the name of an image in the folder `data_dir`, e.g.

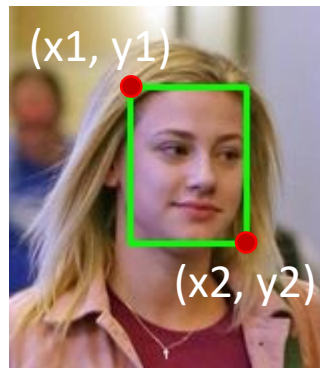
`7_Cheering_Cheering_7_74.jpg`

`7_Cheering_Cheering_7_543.jpg`

`7_Cheering_Cheering_7_889.jpg`

- Each element of `annotate_dict` is a list of bounding boxes.
- Use `for bbox in annotate_dict[key]:` to access each bounding box
- Samples of bounding boxes are as below

62	294	79	319
152	141	196	193
255	103	298	158
			
x1	y1	x2	y2



## Hints for Step 2a.2

- To get the image path from `img_name`
  - `img_path = os.path.join(data_dir, img_name)`
- To read an image from an image path
  - `img = cv2.imread(img_path)`
  - if `img` is `None`, this means that `img_path` is not a legal path of an image
- To draw bounding boxes
  - `cv2.rectangle(img, (x1,y1), (x2,y2), (0,255,0), thickness=2)`
- To show the image
  - `img = img[:, :, ::-1]` # convert from BGR to RGB
  - `plt.imshow(img)`
  - `plt.show()`

從原始影像根據定界框裁切所有臉部區域，將每一臉部區域存成檔案至指定目錄

**Step 2a.3: Implement the following pseudo code to save the cropped faces in function `save_bbox()`**

```
# create a directory named save_dir
for img_name in annotate_dict:
    # get the image path from img_name
    # use cv2.imread() to read from the image path
    for bbox in annotate_dict[img_name]:
        # crop faces according to bbox
        # create the save_path for the cropped face
        # use cv2.imwrite() to save the cropped face
```

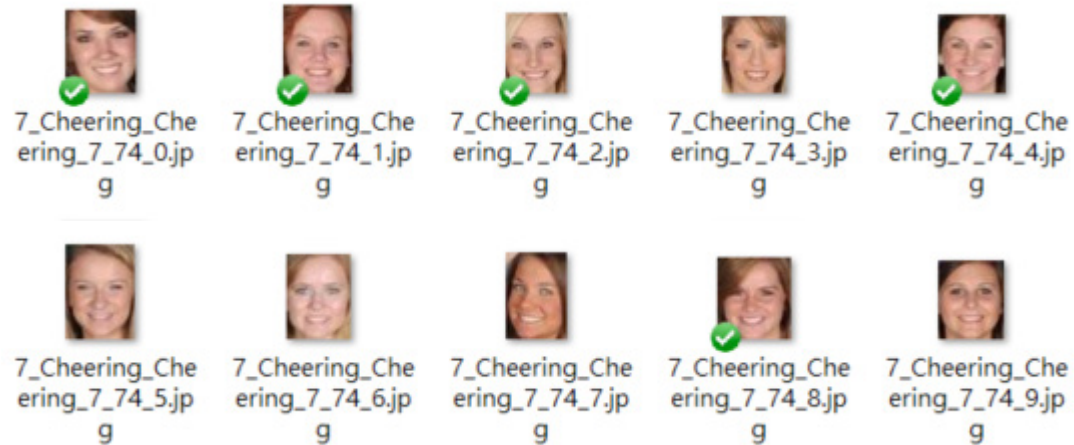
## Hints for Step 2a.3

- To create a directory
  - Use `os.path.exists(save_dir)` to determine whether `save_dir` exists.
  - If `save_dir` does not exist, use `os.mkdir(save_dir)` to create a directory.
  - If `save_dir` exists, then do nothing.
- To crop faces
  - `x1, y1, x2, y2 = bbox`
  - `img_crop = img[y1:y2, x1:x2]`
- The `save_name` of each face is as follows
  - `7_Cheering_Cheering_7_74_0.jpg`
  - `7_Cheering_Cheering_7_74_1.jpg`
  - `7_Cheering_Cheering_7_74_2.jpg`
  - You can create `save_name` from `img_name` with a counter `j`
  - The counter `j` can be created using `enumerate()` as below
    - `for j, bbox in enumerate(annotate_dict[img_name]):`
- To create `save_path` with `save_dir` and `save_name`, use
  - `save_path = os.path.join(save_dir, save_name)`



## Hints for Step 2a.3

- To save the cropped image, use `cv2.imwrite()`
- The samples for cropped faces with their file names are as follows



Step 2a.4: Implement the function `test_dlib()` that applies `dlib` to detect faces in the input image (`7_Cheering_Cheering_7_889.jpg`), and then draw the bounding box of each detected face as done in Step 2a.2. You also need to display the following information in `test_dlib()`

```
7_Cheering_Cheering_7_889.jpg #det1
```

where `#det1` is the number of faces detected by `dlib`.

Next, resize the input image such that its height is 1400, and the aspect ratio (height/width) should remain the same as that of the original input image. Then, repeat the above step to draw the bounding boxes detected by `dlib`, and display the number of faces detected by `dlib`.

Hints for using `dlib`:

Install the `dlib` library and use the following code

```
detector = dlib.get_frontal_face_detector()
img = cv2.imread(img_path)
dets = detector(img, 1)
for det in dets:
    x1, y1, x2, y2 = det.left(), det.top(), det.right(), det.bottom()
    # draw bounding box based on x1, y1, x2, y2
```

# Assignment 2b: Data Statistics

## Step 2b.1: Open [assignment\\_2b.ipynb](#)

Ensure that [assignment\\_2b.ipynb](#), [annotation.txt](#) are in the same folder.

前面練習都是從 `annotate.pkl` 來讀取定界框資訊，此練習改成從 `annotate.txt` 來讀取定界框資訊 (網路上下載之資料庫不一定都會將資料庫資訊存成 `pkl` 檔)

`annotate.txt` 內容如下

7_Cheering_Cheering_7_74.jpg	152	141	196	193	255	103	298	158	...
file name		x1	y1	x2	y2	x1	y1	x2	y2

每一行開頭是影像檔案名稱，接著是人臉定界框 (bounding box) 座標，每一張影像通常包含多張人臉，所以每一行通常會有多組定界框座標

- A bounding box is called **valid** if the following four inequalities are satisfied

$$x1 \geq 0, y1 \geq 0, x2 > x1, y2 > y1$$

- The **width** of a bounding box is defined as  $x2 - x1 + 1$
- The **height** of a bounding box is defined as  $y2 - y1 + 1$

# Assignment 2b: Data Statistics

## Step 2b.2: Display the following information

```
width < 10: #1
10 <= width < 20: #2
20 <= width < 30: #3
30 <= width < 40: #4
40 <= width < 50: #5
50 <= width      : #6
```

where #1 is the number of **valid** bounding boxes belonging to range 1 (`width < 10`), and #2 is the number of **valid** bounding boxes belonging to range 2 (`10 <= width < 20`), etc.

### Hints for Step 2b.2:

- Use `annotation = line.strip().split(' ')` to convert each line to a list
- Convert the type of each element of `annotation[1:]` to integer

## Hints for Step 2b.2

- Suppose `line` (a string) is

```
0--Parade/0_Parade_marchingband_1_799 78 221 85 229 78 237 92 255 112  
212 123 226
```

- After the steps in the previous page, we obtain a list of integers as

```
[78, 221, 85, 229, 78, 237, 92, 255, 112, 212, 123, 226]
```

- Use `np.array()` and `np.reshape()` to convert the above to an array of bounding boxes as below

```
[[ 78 221 85 229]      -> bbox1 [x1 y1 x2 y2]
```

```
[ 78 237 92 255]      -> bbox2 [x1 y1 x2 y2]
```

```
[112 212 123 226]]    -> bbox3 [x1 y1 x2 y2]
```

- Compute the **width** of each bounding box,  $x_2 - x_1 + 1$ , to obtain a list of widths

```
[8, 5, 12]
```

- Display the required information in Step 2 based on the above list.

# Assignment 2b: Data Statistics

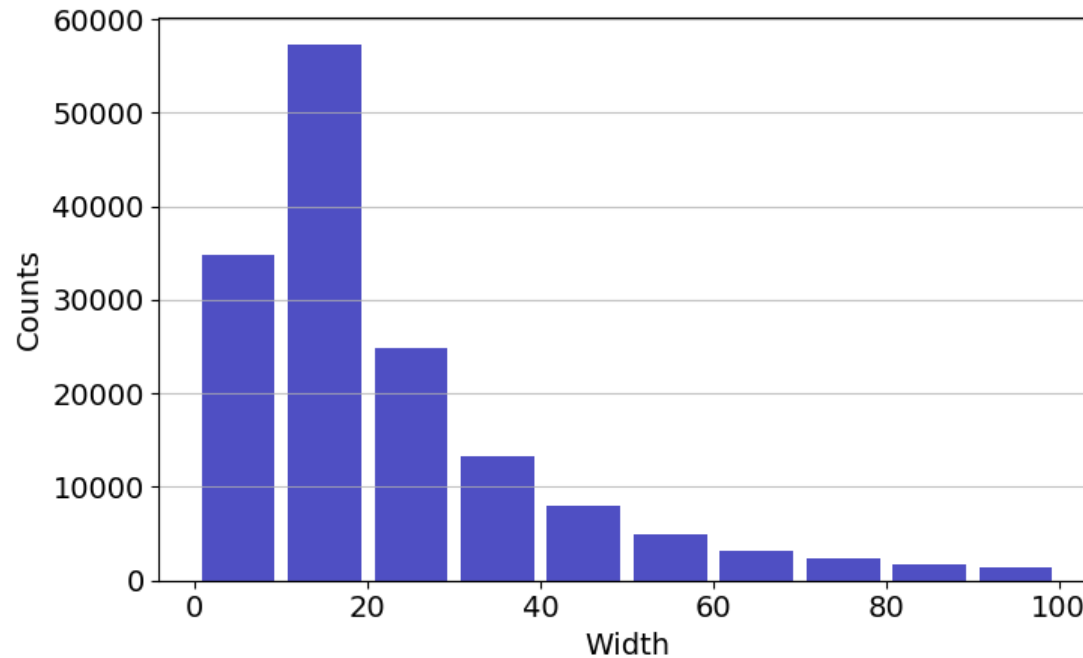
## Step 2b.3: Display the following information

```
width/height < 0.6: #1
0.6 <= width/height < 0.7: #2
0.7 <= width/height < 0.8: #3
0.8 <= width/height < 0.9: #4
0.9 <= width/height < 1.0: #5
1.0 <= width/height      : #6
```

where #1 is the number of **valid** bounding boxes belonging to range 1 ( $\text{width/height} < 0.6$ ), and #2 is the number of **valid** bounding boxes belonging to range 2 ( $0.6 \leq \text{width/height} < 0.7$ ), etc.

## Assignment 2b: Data Statistics

Step 2b.4: Display the following figure



**Hint:** Use `plt.hist(widths, range(0,101,10), rwidth=0.85)`

where `widths` is a python list of the width of valid bounding boxes.

Use `xlabel` to set the label for the x-axis, and `ylabel` to set the label for the y-axis.

# Assignment 2b: Data Statistics

## Step 2b.5:

In Step 2b.1, we consider only the valid bounding boxes. Modify your code to find the number of **invalid** bounding boxes.