電腦視覺與深度學習
(Computer Vision and Deep Learning)
Final Project

TA:

Kevin: i1007673219@gmail.com

Office Hour: 19:00~21:00, Mon.
09:00~11:00, Wed.
At CSIE 9F Robotics Lab.

Notice (1/1)

- ❑ Copy is strictly prohibited!! Penalty: Grade will be zero for both persons!!
- ❑ Due date => 2020/01/09 (Thu.) 23:59:59
  - No delay. If you submit project after deadline, you will get 0.
- ❑ Demo date => 2020/01/10 (Fri.) 09:10 – 12:00
  - Please check the time table on the moodle.
- ❑ Upload to => 140.116.154.1 -> /Upload/FinalProject
  - User ID: opencvdl2019 Password: opencvdl2019
- ❑ Format
  - Filename: FinalProject_GroupNumber_Version.rar
    - FinalProject_01_v1.rar
    - If you want to update your file, you should update your version to be v2, ex: FinalProject_01_v2.rar
    - Only group leader needs to hand in the final project file.
  - Content: project folder*(including ppt file)
- ❑ Topic of final project is unlimited.
- ❑ All group members must attend the demo.

Grading

1. (50%) Source code & PPT file

2. (50%) Project Demo

1. (50%) Source code & PPT file

1) Source code

The source code of your final project: You can use any programming languages.

2) PPT file

PPT file: Please check following example slices.
(x pages)

2. (50%) Project Demo

1) Project Demo

The project demonstration time is 9:10 ~ 12:00 on 2020/01/10. You can check your demo time on Moodle. Remember to bring your notebook to demo your project.

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Neural Information Processing Systems (NIPS 2015)

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Keywords: Object Detection, Region Proposal, Convolutional Neural Network.

Group Number:	01		
Group Members:	連震杰	P00001111	電機所
	黃鼎元	P46061411	資訊所

1. Introduction

1) Motivation:

- Usually it is at the first paragraph of Introduction Session.
- Why does this paper want to do this research? Application?
- Like why is face detection research important? Its application is…..

2) Objective:

- What is the goal/objective of this paper?
- Like what kind of face detection this research can complete? Frontal view? 45^0 view? …. Detection in the cloud?

3) Contribution:

- What is its the contribution?
- Because this paper develops xxx methods to solve yyy existing problem.

2. System Framework: Training Process

Images N_t, N_v | Feature maps $512 \times N_t$ | Feature maps $512 \times N_v$

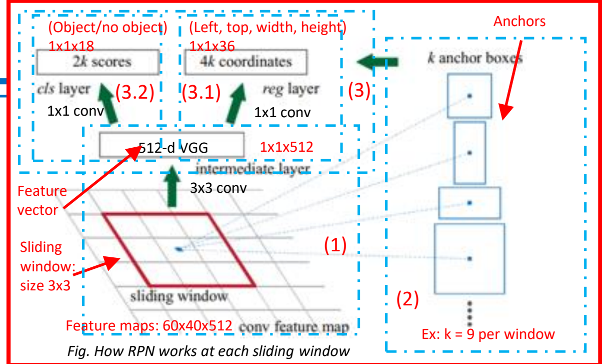


Fig. How RPN works at each sliding window

3. Region Proposal Networks (1/4):

1. Sliding window and low feature vector extraction:

- Slide a window (size nxn, ex: n=3) over the convolutional feature map
- Map window to a lower-dimensional feature vector (512-d VGG).
- The vectors contain the location information on original image. (an receptive field of 228 pixels for VGG)
- This process works as a 3x3-kernel convolution with 512 output feature maps
- Vectors contain the location information on original image.

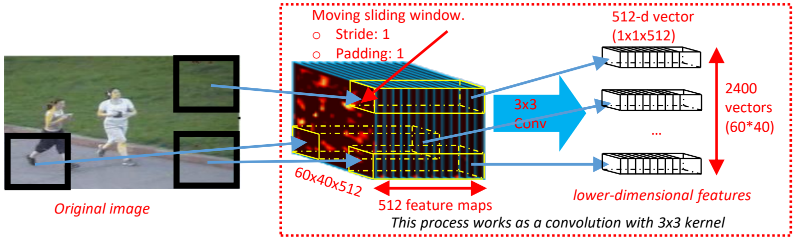


Fig. lower-dimensional features extraction

2400 low-dimensional vectors

2. Anchor generation:

- Anchors: (possible proposals)
  - Pre-defined reference boxes.
  - Multi scales and ratios.
  - Translation invariant. (Same set of anchors at every location)
- **At each sliding-window position on image, k anchor boxes is generated.**
  - k: number of maximum possible proposals for each location.
- **In our case:**
  - k = 9 anchors.
  - 3 scales: 128x128, 256x256, 512x512.
  - 3 ratios: 1:1, 1:2, 2:1 (each scale)

❖ **Ignore unnecessary Anchors:**

i. Total: w*h*k anchor (60x40x9 = 21600)
ii. (Actually) Ignore all cross-boundary anchors.
iii. The final number of anchors: ~6000

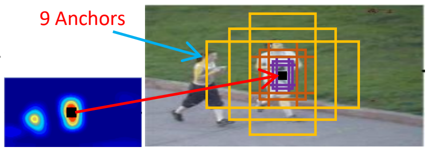


Fig. Anchors are generated at the position re-projected from a sliding window center on feature map.

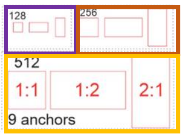


Fig. Sample Anchors

~6000 anchors: x_a, y_a, w_a, h_a

3. Feeding data into Sibling networks:

- Simultaneously classifying (Cls) and regressing (Reg) anchors based on their corresponding feature vectors.

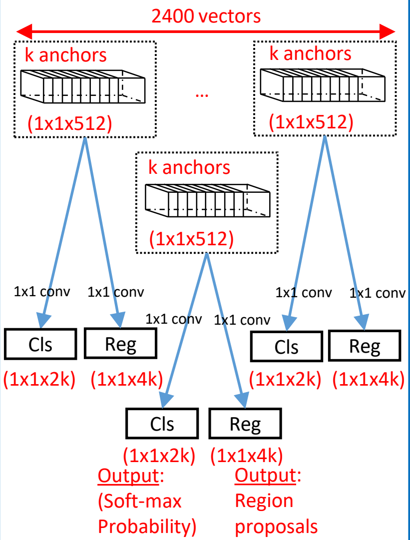


Fig. Feed lower dimensional vectors and anchors into sibling networks

3.1. Regression:

- **Object**: Compute offsets from anchor boxes.
- **Method**: linear regression
- **Process:**

x, y: center of box
w, h: width, height
x, y, w, h: Predicted box
x_a, y_a, w_a, h_a: Anchor box
x*, y*, w*, h*: Ground-truth Box

1. *Bbox regression:*

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

i: index of an anchor in a mini-batch.
t_i: vector represents 4 parameterized coordinates of the predicted box
t_i^*: the ground-truth box associated with a positive anchor.

2. *Loss function:*

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$$

R: robust loss function (smooth L_1)

$$loss(x,y) = \sum \begin{cases} 0.5 * (x_i - y_i)^2, if |x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, otherwise \end{cases}$$

Fig. 1x1-kernel convolutional layer for regression. Output: 1x1x4k.

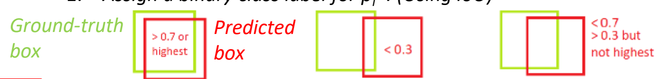
$\mathcal{L}_{cls}(p_i, p_i^*)$

$L_{reg}(t_i, t_i^*)$

3.2. Binary Classification:

- **Output:** Probability that each anchor shows an object.
- **Method**: Binary classification.
- **Process**:

1. *Assign a binary class label for p_i*: (Using IoU)*



$p_i^* = 1$ (positive) | $p_i^* = 0$ (negative) | No concern

2. *Minimize loss function: log-loss (can use sigmoid, instead)*

$$\mathcal{L}_{cls}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i)$$

i: index of an anchor in a mini-batch.
p_i: the predicted probability of i^{th} anchor being an object.
p_i^*: the ground-truth label. (from IOU)

Fig. 1x1-kernel convolutional layer for classification. Output: 1x1x2k.

4) Loss function of RPN:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i \boxed{p_i^* L_{reg}}(t_i, t_i^*).$$

Lambda: balancing parameter (10)
N_{cls}: Normalized term by mini-batch (512)
N_{reg}: Normalized term by number of anchors (2400)

Reg only for positive anchors

Objectness score (P_c) | Region proposals (t_x, t_y, t_w, t_h)

3. Deep Learning Architecture

Fig. Faster R-CNN scheme. A single, unified network for object detection.

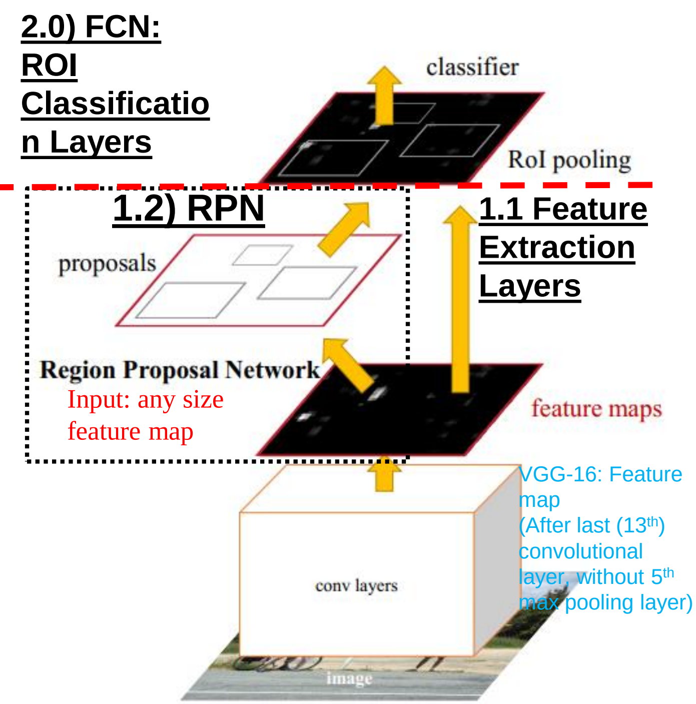


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

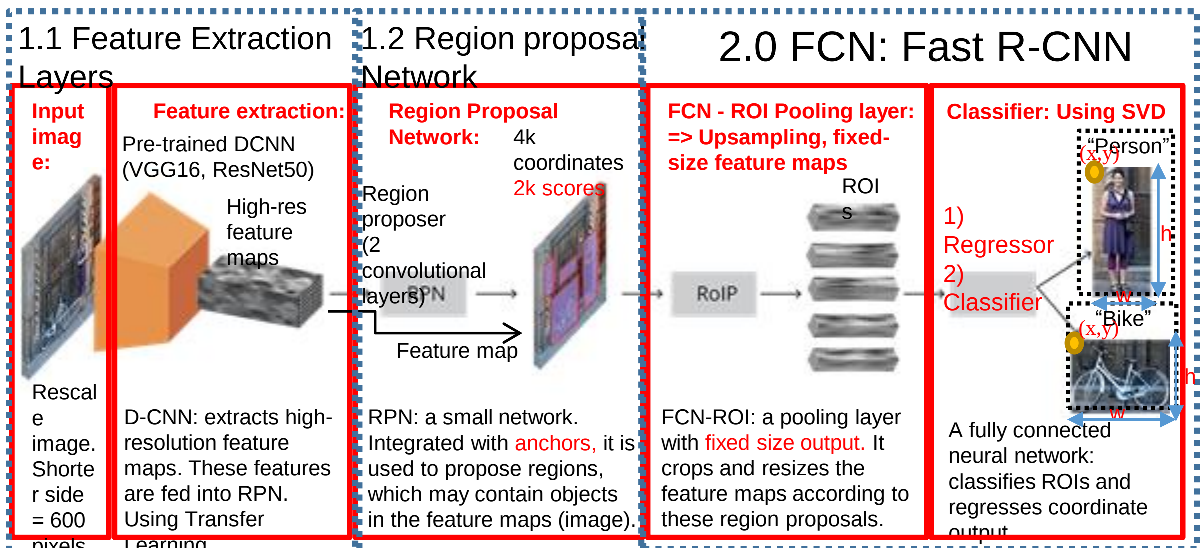


Fig. Overview of Faster R-CNN

Output:
x,y (top left point),
w (width), h (height)
-> Object class
對每個ROI output
一個object class

Image source: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detect

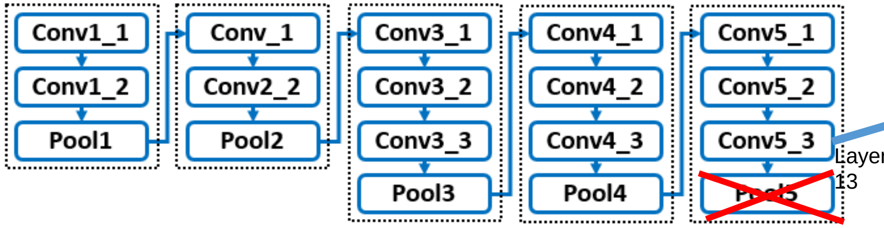
Or 3. Deep Learning Architecture

Images N_t, N_v

1.1-2) Feature Extraction Using Deep Convolutional Neural Network: (ex: VGG16, ZF, ResNet)

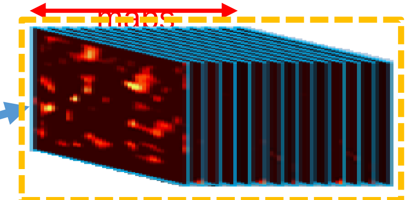
- **Object:** Extract feature maps from input images.

Size: 1000 x 600 x 3



Deep Convolutional network: (ex VGG16)

512 feature maps

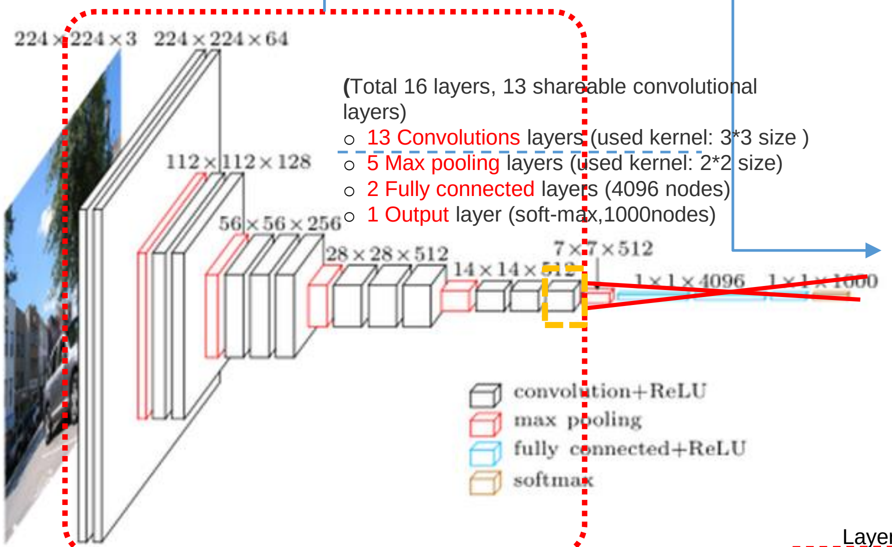


VGG-16: Feature map (After last (13th) convolutional layer without 5th max pooling layer)

Size: (60x40 grids) x 512c
cf. same as (14x14 grids) x 512c in VGG16

Fig. Feature extraction process

用ImageNet train過的weight當初始值



(Total 16 layers, 13 shareable convolutional layers)
- 13 Convolutions layers (used kernel: 3*3 size)
- 5 Max pooling layers (used kernel: 2*2 size)
- 2 Fully connected layers (4096 nodes)
- 1 Output layer (soft-max,1000nodes)

Layer 13

Fig. VGG16 for feature extraction. (until the last convolutional layer)

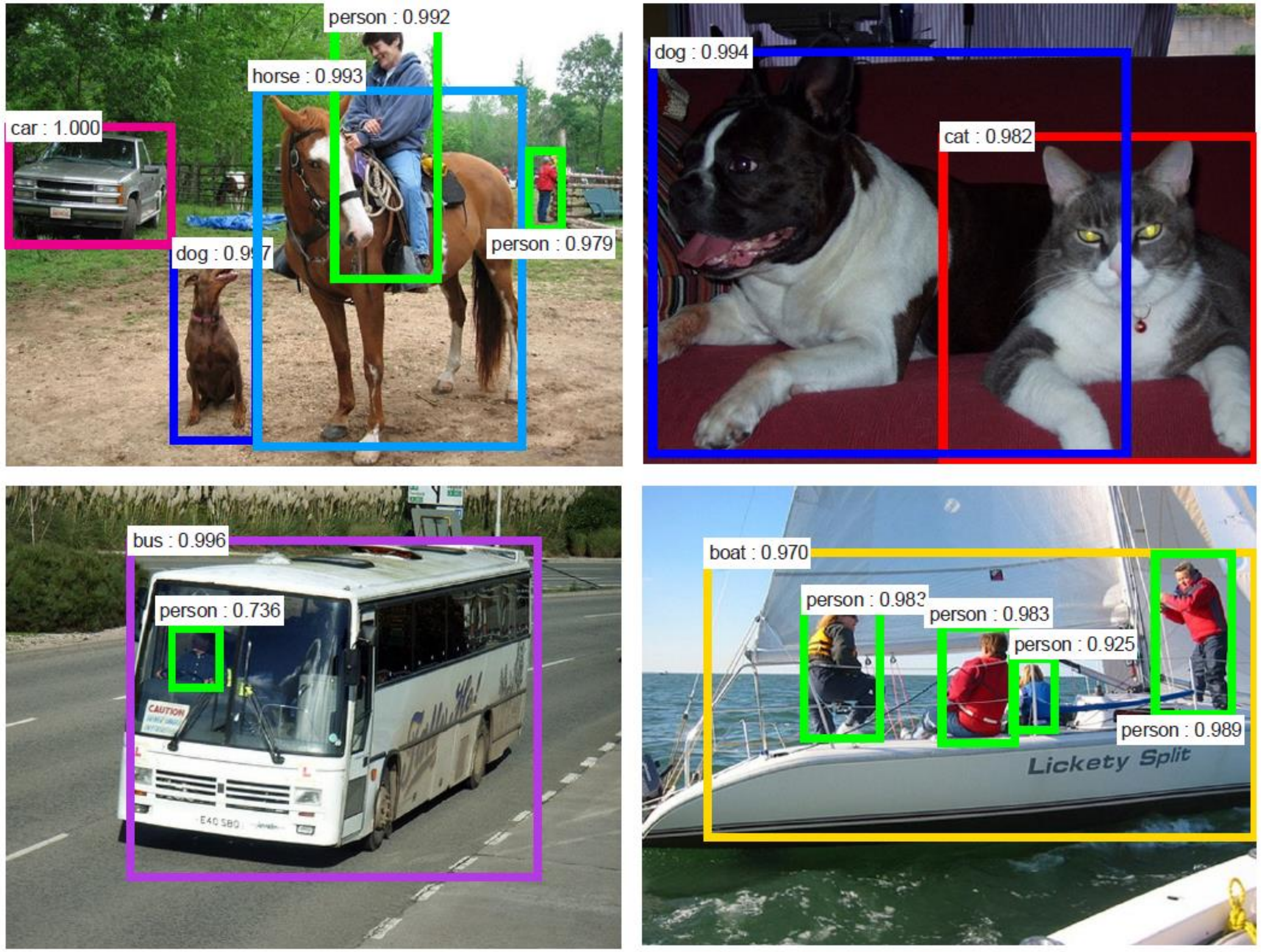
Very Deep Convolutional Networks for Large-Scale Image Recognition
https://www.quora.com/What-is-the-VGG-neural-network

N_t x (60x40 grids)x512c feature maps

N_v x (60x40 grids)x512c feature maps

4.1 Experimental Result – Correct Examples

- Why is it correct?



4.2 Experimental Result – Incorrect Examples

- Why is it incorrect?

5. Conclusion