NCKU MiinWu
School of Computing
敏求智慧運算學院

AI Robotics

How to Learn Deep Learning? (2/2)
(如何學習人工智慧-深度學習?)

National Cheng Kung University
Dept. of Computer Science and Information Engineering
Robotics Lab.
國立成功大學 資訊工程系 機器人實驗室

(Jenn-Jier James Lien / 連震杰)
http://robotics.csie.ncku.edu.tw
jjlien@csie.ncku.edu.tw

國立成功大學

3.0 History: Computer Vision and Deep Learning

1. Industrial Revolution:

3)1969, Industry 3.0:
Computing and Automation

4)2013, Industry 4.0:
Smart Manufacturing

1)1981, 生產力1.0：程式化

2)1991, 生產力2.0：整線化 3)2001, 生產力3.0：電子化 4)2011, 生產力4.0：智慧化

2. OpenCV: Image Processing, Computer Vision, Machine Learning, and Deep Learning

0)1999, OpenCV0.0 Started

1)2007, OpenCV1.0: IP+CV

2)2009, OpenCV2.0: IP+CV

3)2016, OpenCV3.0: IP+CV+ML

4)201x, OpenCV 4.0: IP+CV+ML+DL?

1)1999, OpenCV Supported by Intel

2)2010, OpenCV Supported by NVidia

3)2011, OpenCV Supported by Google

3. Deep Learning:

1)1989, LeCun tPami1989 **2)1998, LeNet IEEE1998** **3)2012, AlexNet NIPS2012**

4. Robotics Lab.: Computer Vision and Machine Learning – 1991 ~ 2013

1)1993, Facial Expression Recognition at CMU, USA

2)1999, Face Detection and Recognition at a Distance Using Stereo at Visionics, USA

3)2004, Automatic Optical Inspection (AOI System Integration): TFT-LCD, Solar Cell…

4)2009, eCV Surveillance with TI & Faraday

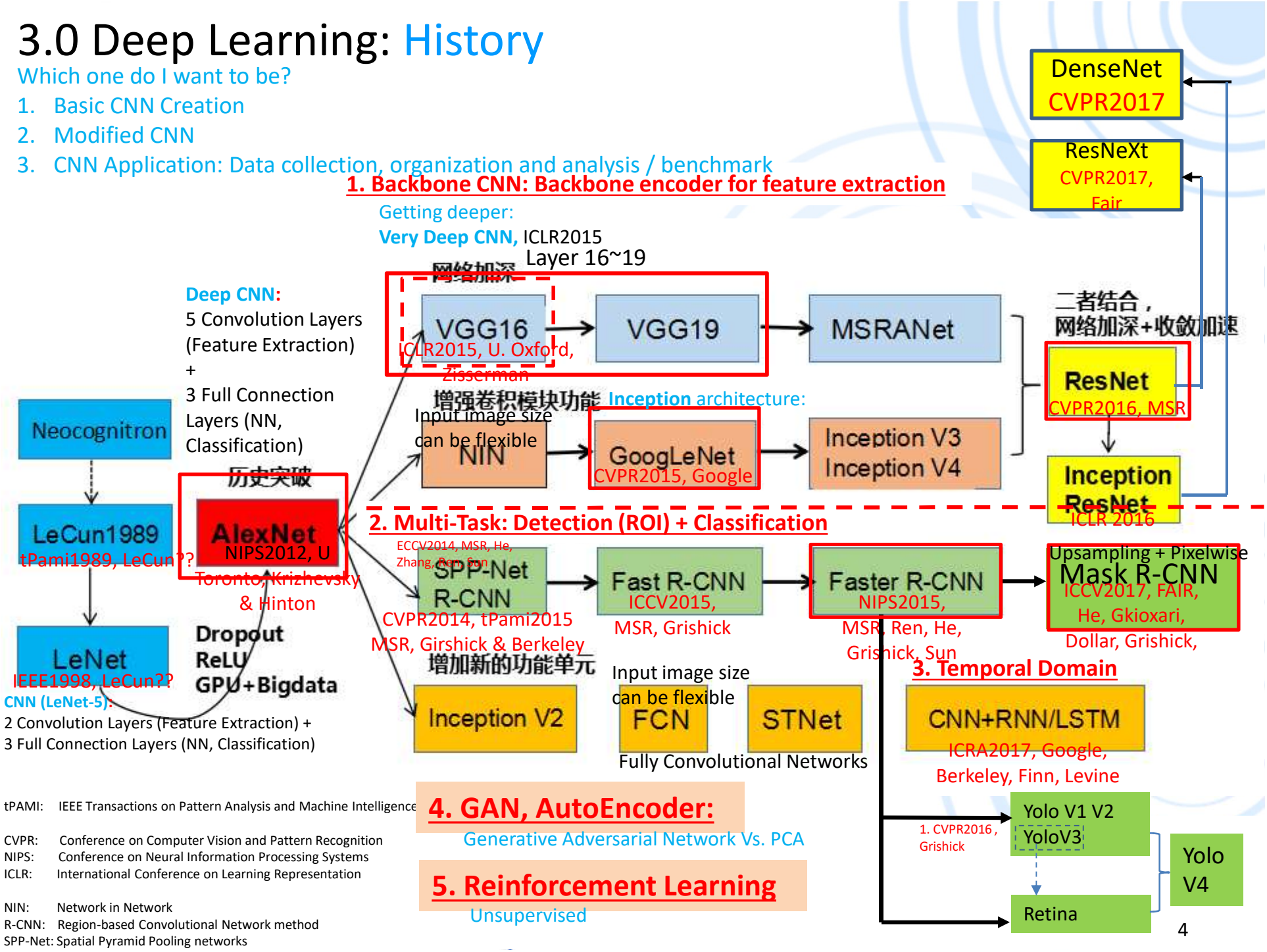
5. Robotics Lab.: Computer Vision and Deep Learning - Since 2014: AI智慧製造+AI半導體+AI醫材+AI

Content

3.0 Deep Learning: History

Which one do I want to be?

1. Basic CNN Creation
2. Modified CNN
3. CNN Application: Data collection, organization and analysis / benchmark



tPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence

CVPR: Conference on Computer Vision and Pattern Recognition
NIPS: Conference on Neural Information Processing Systems
ICLR: International Conference on Learning Representation

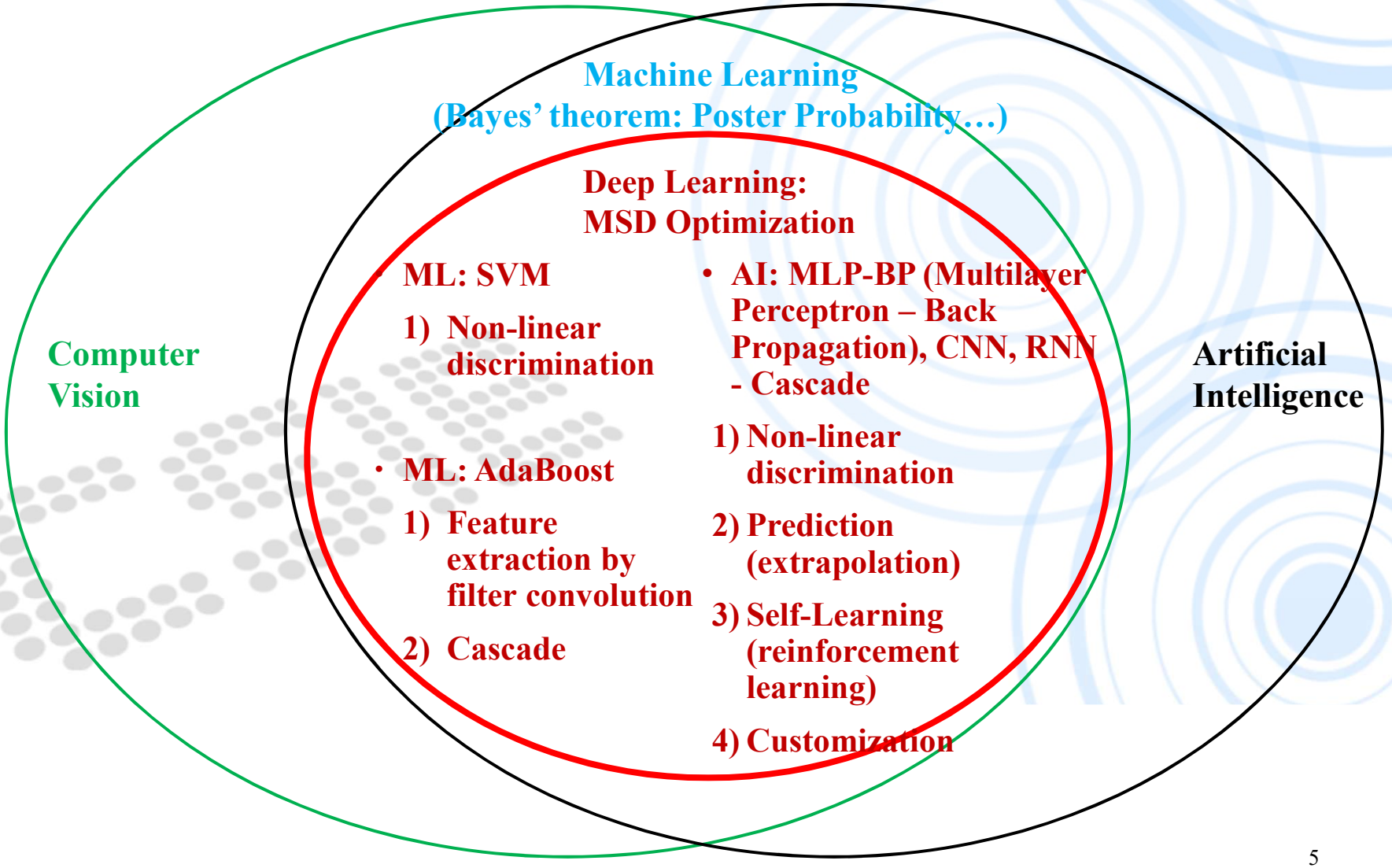
NIN: Network in Network
R-CNN: Region-based Convolutional Network method
SPP-Net: Spatial Pyramid Pooling networks

3.0.0 Deep Learning:

Computer Vision Vs. Artificial Intelligence Network

Machine Learning: Need Data + Label

Deep Learning: Need Big Data + Label



3.0.0 Deep Learning: AI MLP-BP (Multilayer Perceptron Back-Propagation)

寫程式不會以equation的方式，會以vector , matrix (Ax = b), homogenous matrix, 的方式撰寫。

WX = Y'

Batch size *N*

Loss Function: 1) MSE: $I = \min \frac{1}{N}\sum_{j=1}^{N} |yj - y'_j|^2 +$ Continuous, regression

2) Cross-Entropy: $I = \min -[\sum_{j=1}^{N}(y_j \log y'_j) + \ldots]$ Discrete, classification

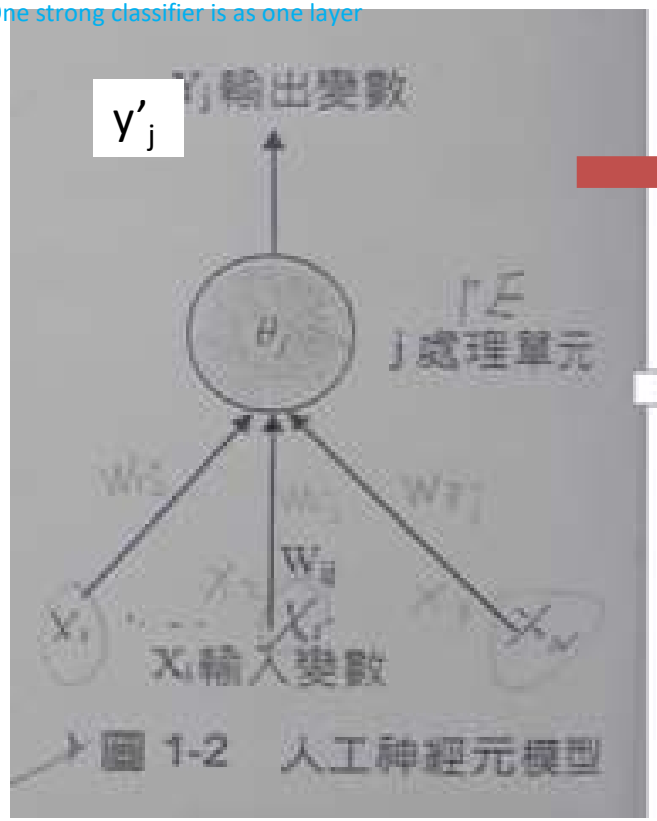
3) Softmax

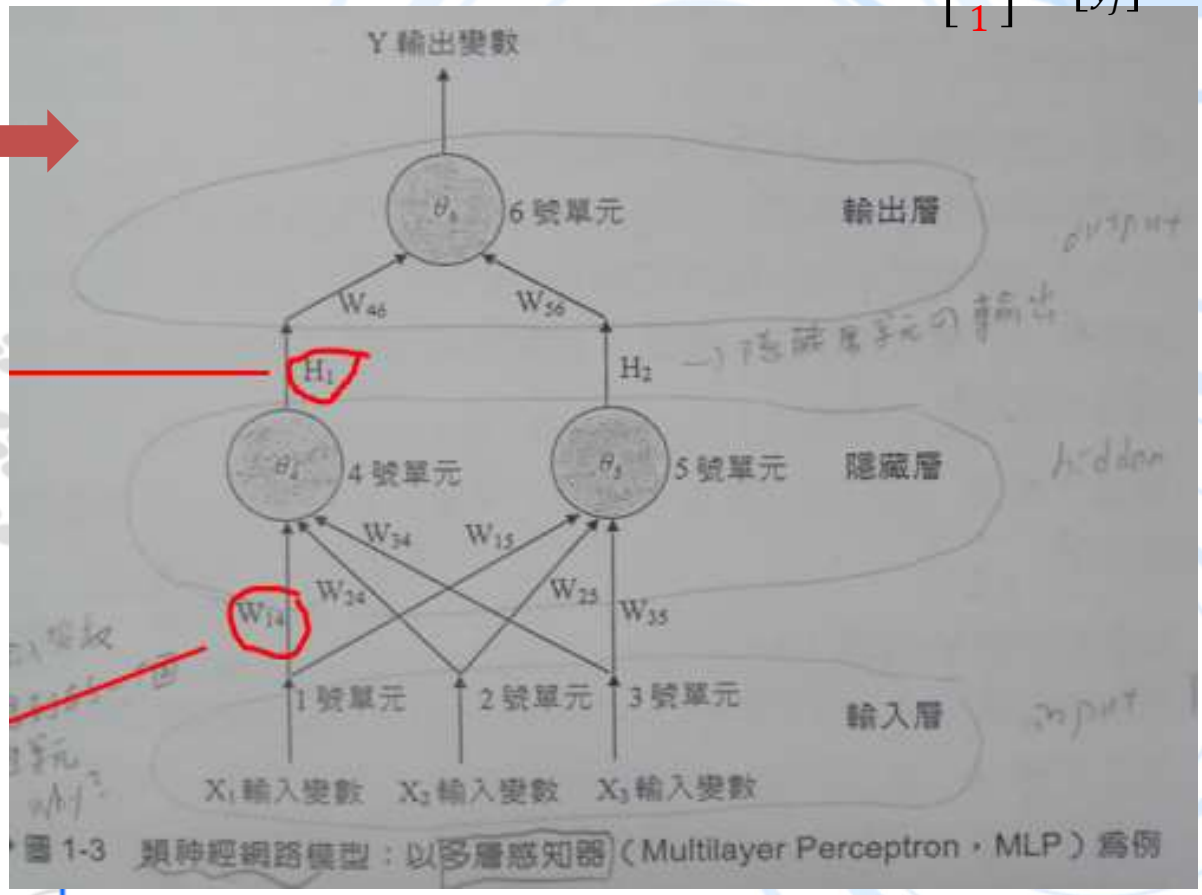
Homogenous Matrix

$$\begin{bmatrix} w_{11} w_{21} \cdots w_{k1} \cdots w_{N1} & b_1 \\ w_{12} w_{22} \cdots w_{k2} \cdots w_{N2} & b_2 \\ \vdots & \\ w_{1j} w_{2j} \cdots w_{kj} \cdots w_{Nj} & b_j \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ \vdots \\ x_N \\ 1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \\ \vdots \\ y_j \end{bmatrix}$$

-One weak classifier is one haar filter is also as one neuron unit

-One strong classifier is as one layer





$y'_j = f(\sum_i W_{ij} x_i - \theta_j)$

Threshold, bias

W: As coefficient of edge detection/feature extraction filter

b: As the threshold of edge detection/feature extraction filter, it is a bias

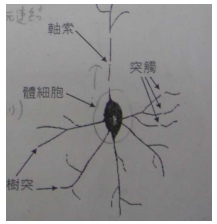
3.0.0 Deep Learning: AI MLP-BP

寫程式不會以equation的方式，會以vector, matrix (Ax = b), homogenous matrix, 的方式撰寫。

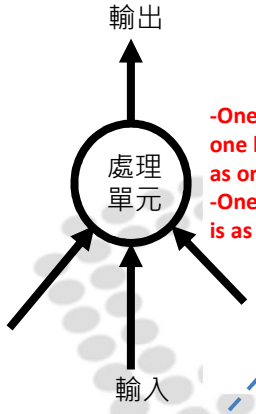
W: As coefficient of edge detection/feature extraction filter
b: As the threshold of edge detection/feature extraction filter, it is a bias

- For the input, why are extraction features better than raw image??

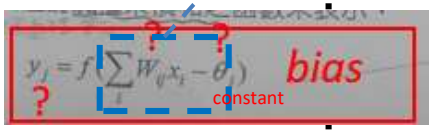
Review NN (Ch1) :



生物神經元模型



人工神經元模型



A neuron unit

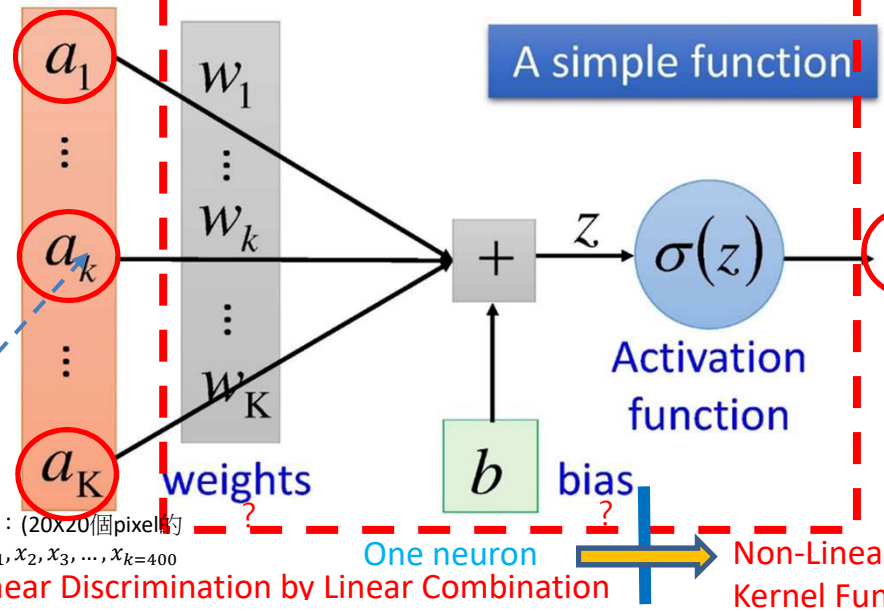
Neural Network

One **Neuron**

Linear Combination:

$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$

Constant, bias



input：(20X20個pixel的圖) $x_1, x_2, x_3, \ldots, x_{k=400}$

Linear Discrimination by Linear Combination

One neuron

Non-Linear Discrimination by Kernel Function / Activation Function as SVM phi()

Linear Combination:

J input data vectors (here J=1)

$$Aw = z \quad ?$$

I/P O/P

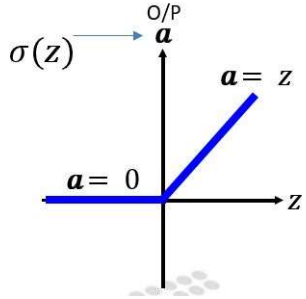
$$\begin{bmatrix} w_{11} w_{21} \cdots w_{k1} \cdots w_{N1} \; b_1 \\ w_{12} w_{22} \cdots w_{k2} \cdots w_{N2} \; b_2 \\ \vdots \\ w_{1j} w_{2j} \cdots w_{kj} \cdots w_{Nj} \; b_j \end{bmatrix}_{J \times (N+1)} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \\ \vdots \\ a_N \\ 1 \end{bmatrix}_{(N+1) \times 1} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \\ \vdots \\ z_j \end{bmatrix}_{J \times 1}$$

1 J-dim output result vector, here J=1)

J: Can solve by EM optimization, random (such as Particle filter), ...
(可見此份投影片 p.38+3)

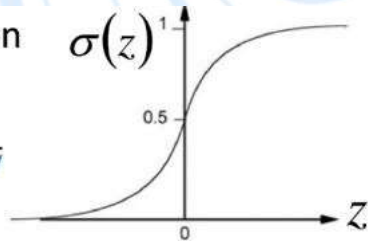
a=σ(z)
Here, the output is *a*, which is one input element for next neuron

-One weak classifier is one haar filter is also as one neuron unit
-One strong classifier is as one layer



Sigmoid Function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

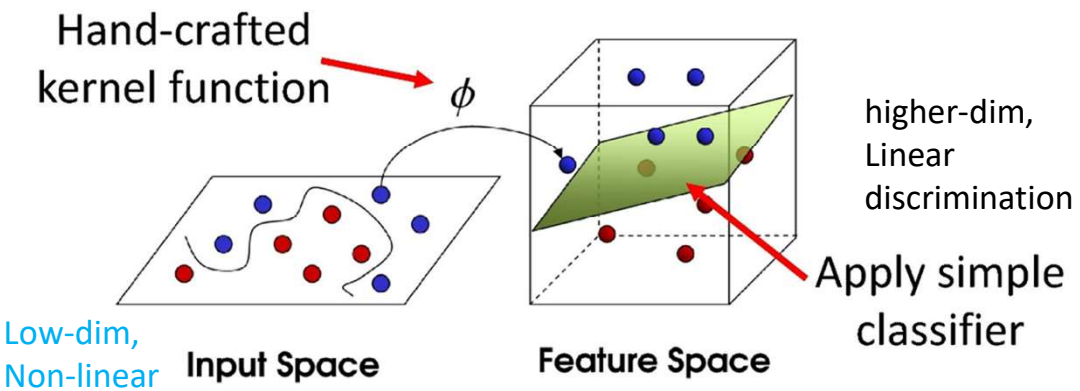


3.0.1 Deep Learning with Machine Learning – Non-Linear Discrimination: As Learnable Kernel (SVM)

Deep Learning from ML:
Networks as
1) Non-linear discrimination: As learnable kernel (SVM) phi()
2) AdaBoost Cascade: As boosted weak / strong classifiers

SVM

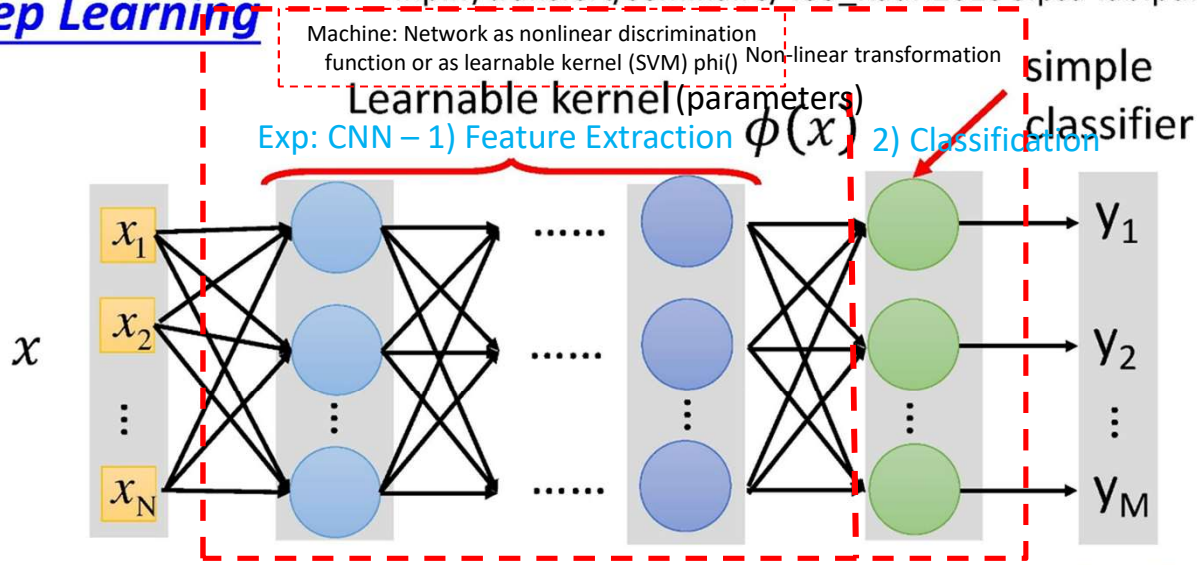
J: Without cascade concept



Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

Deep Learning

J: With cascade concept



3.0.1 Deep Learning with Machine Learning – AdaBoost Cascade: As boosted weak/strong classifiers

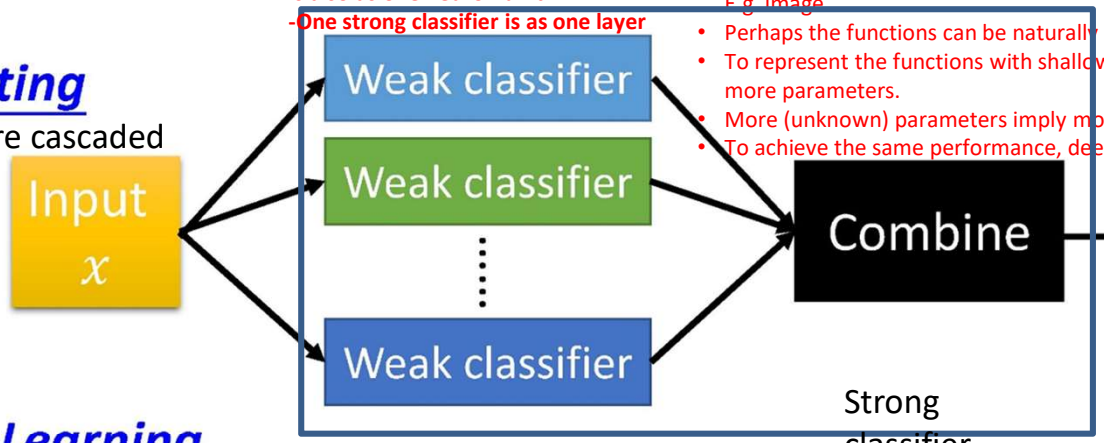
Deep Learning from ML:
Networks as
1) Non-linear discrimination: As learnable kernel (SVM) phi()
2) AdaBoost Cascade: As boosted weak / strong classifiers

Boosting

Cascade: Strong classifiers are cascaded

Strong classifier

-One weak classifier is one haar filter is also as one neuron unit
-One strong classifier is as one layer

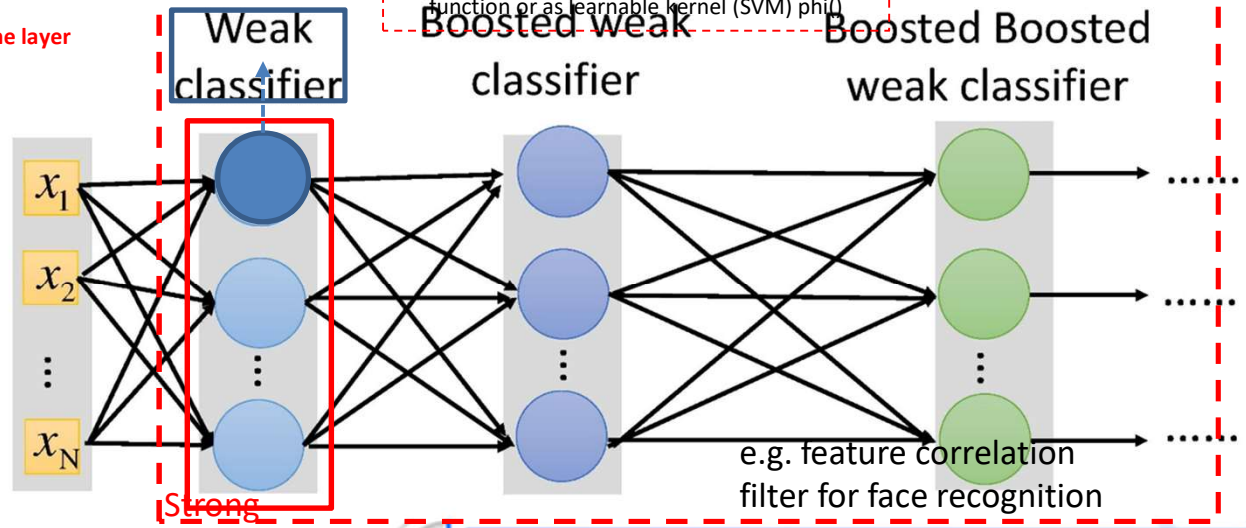


- Some functions can be easily represented by deep structure (topology). E.g. image
- Perhaps the functions can be naturally decomposed into several steps.
- To represent the functions with shallow (fat and short) structure needs more parameters.
- More (unknown) parameters imply more training data needed.
- To achieve the same performance, deep learning needs less training data.

Deep Learning

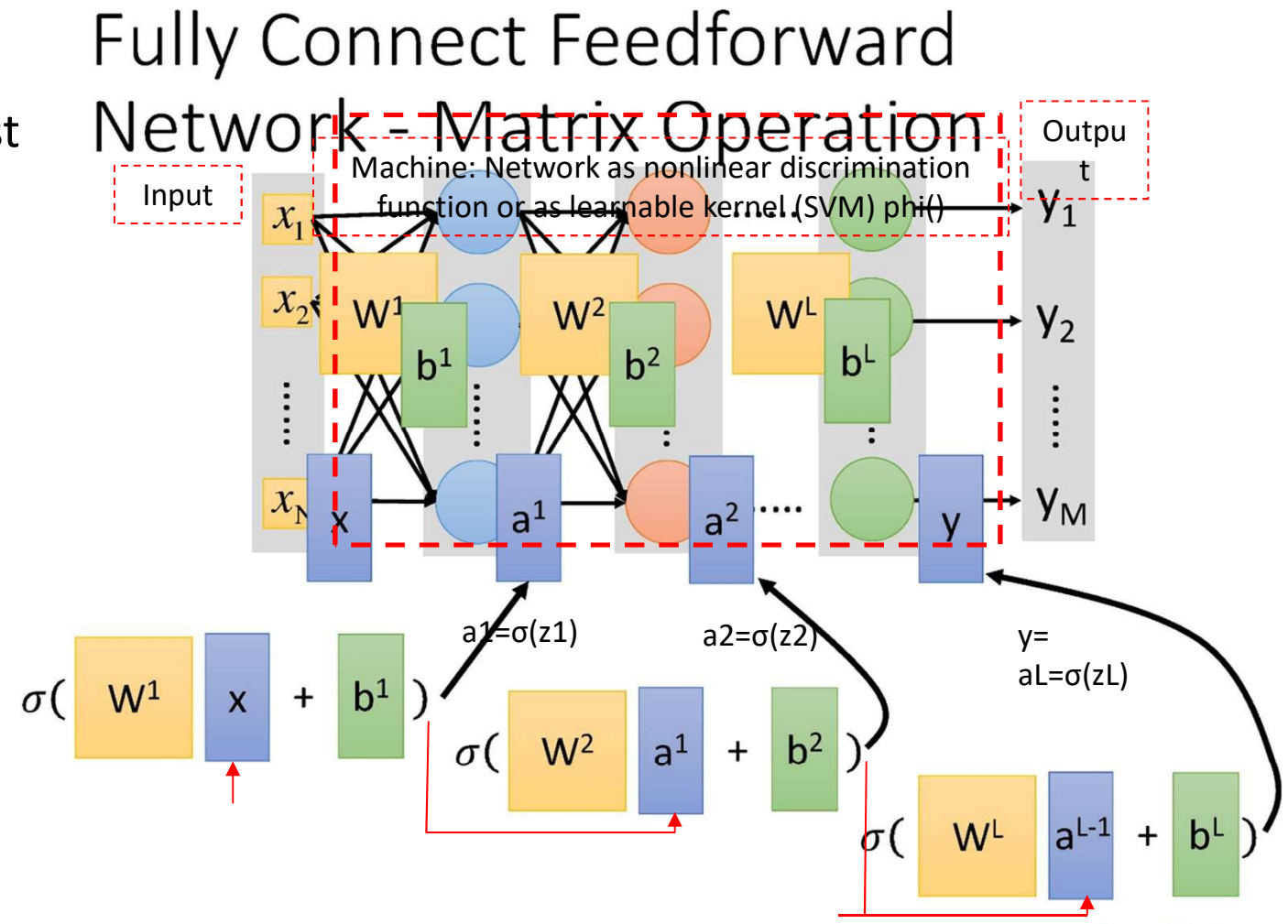
-One weak classifier is one haar filter is also as one neuron unit
-One strong classifier is as one layer

Machine: Network as nonlinear discrimination function or as learnable kernel (SVM) phi()



3.0.2 Deep Learning with Machine Learning – AX=b

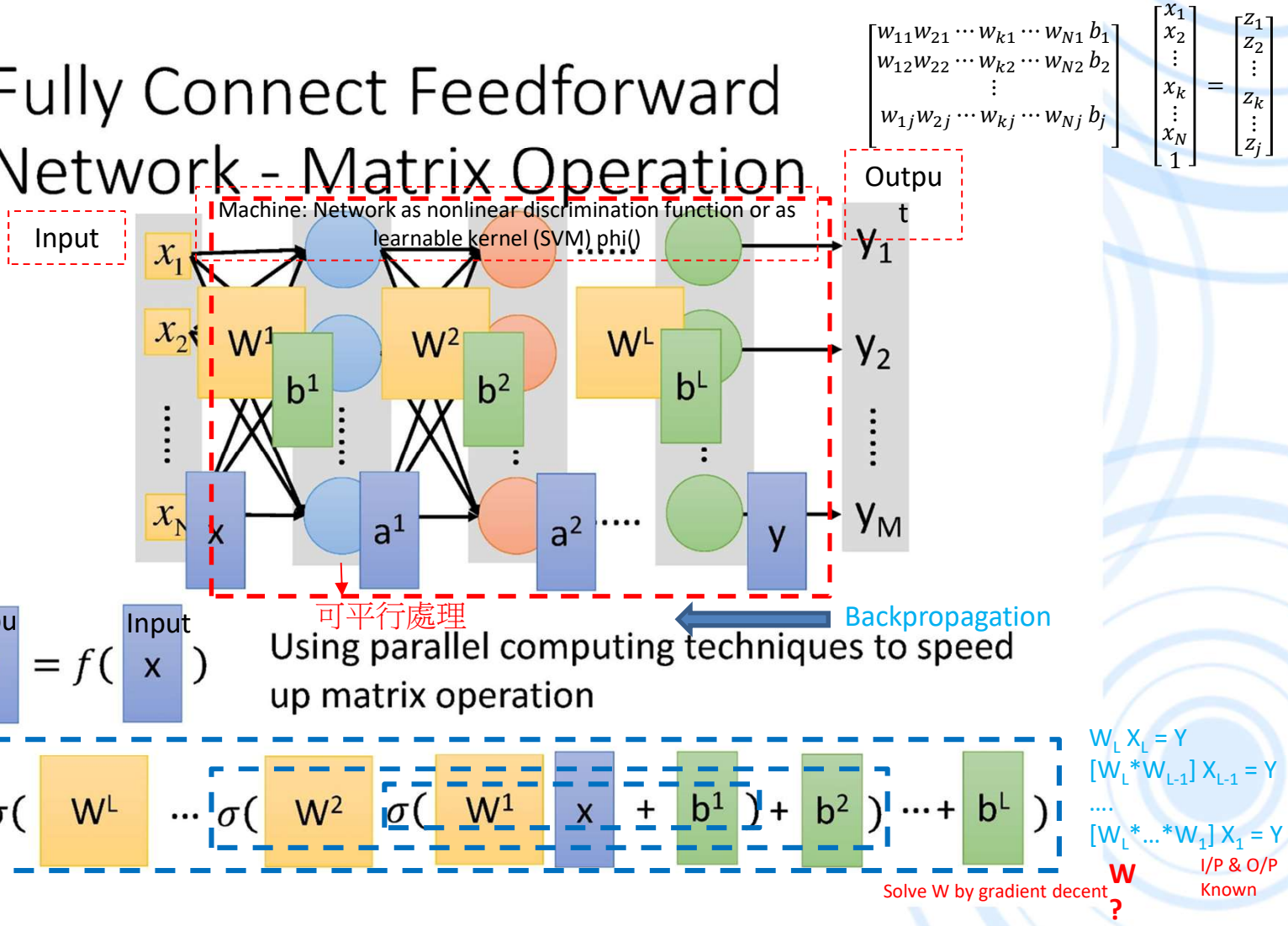
: AdaBoost
- Cascade



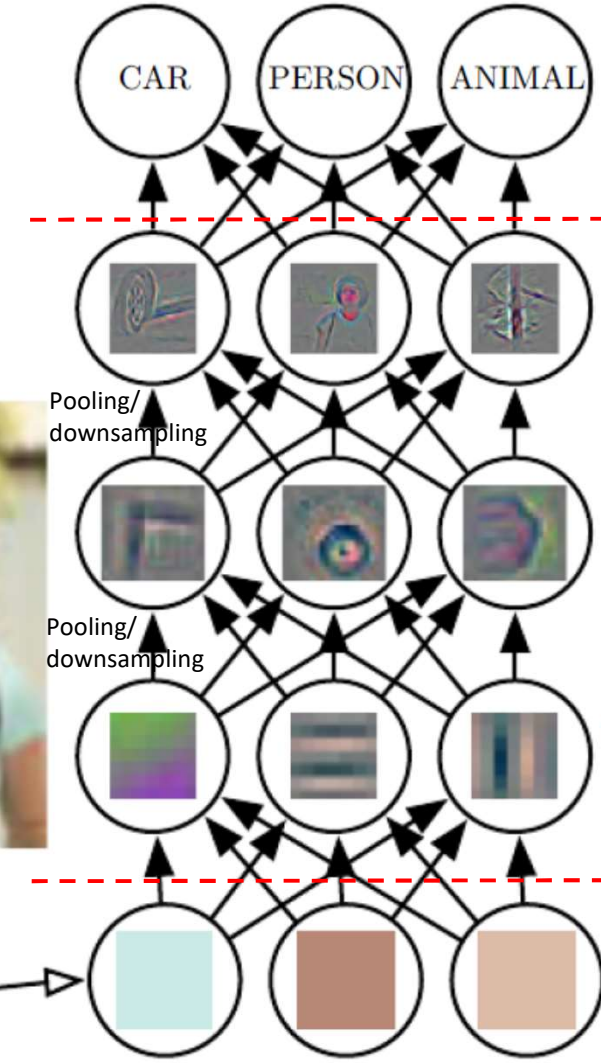
3.0.2 Deep Learning with Machine Learning – AX=b

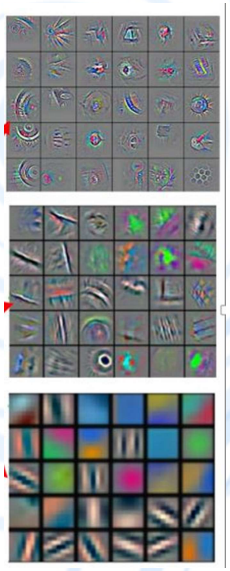
: AdaBoost

- Cascade



3.0.2 Deep Learning: CNN Classification



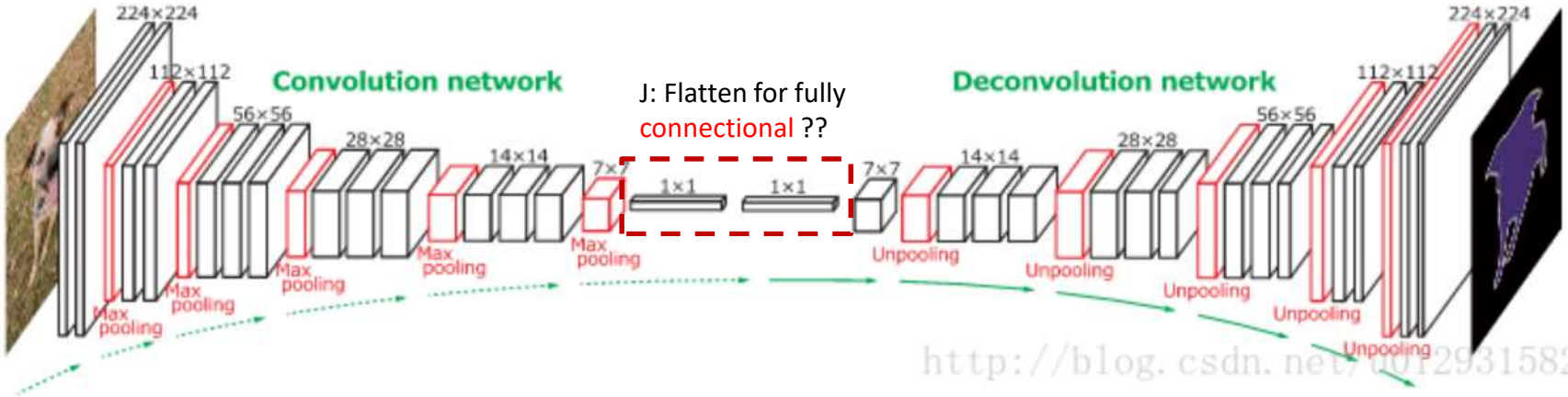


2. Classification Layers (NN)

1. CNN: Feature Extraction Layers

0. Input Data:
   1) Texture > Sensitive to light
   2) Shape > Sensitive to noise
   3) Color > RGB to HSV
      > RGB to YUV for compression
   4) RGB+Depth

3.0.2.1 Convolution Encoder Decoder Network



Reference paper: Noh, H., Hong, S., Han, B.\ 2015.\ Learning Deconvolution Network for Semantic Segmentation.\ arXiv e-prints arXiv:1505.04366.

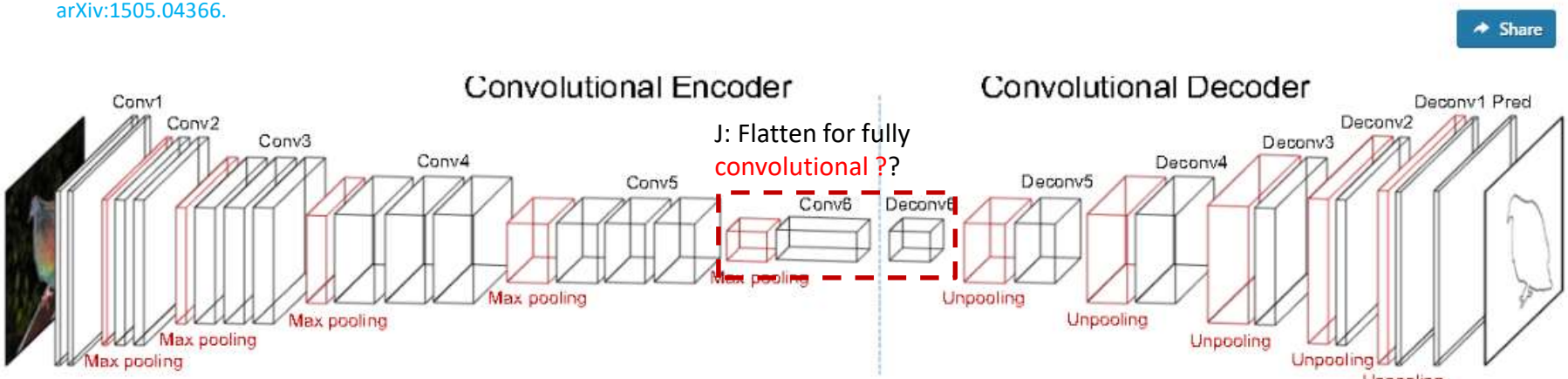
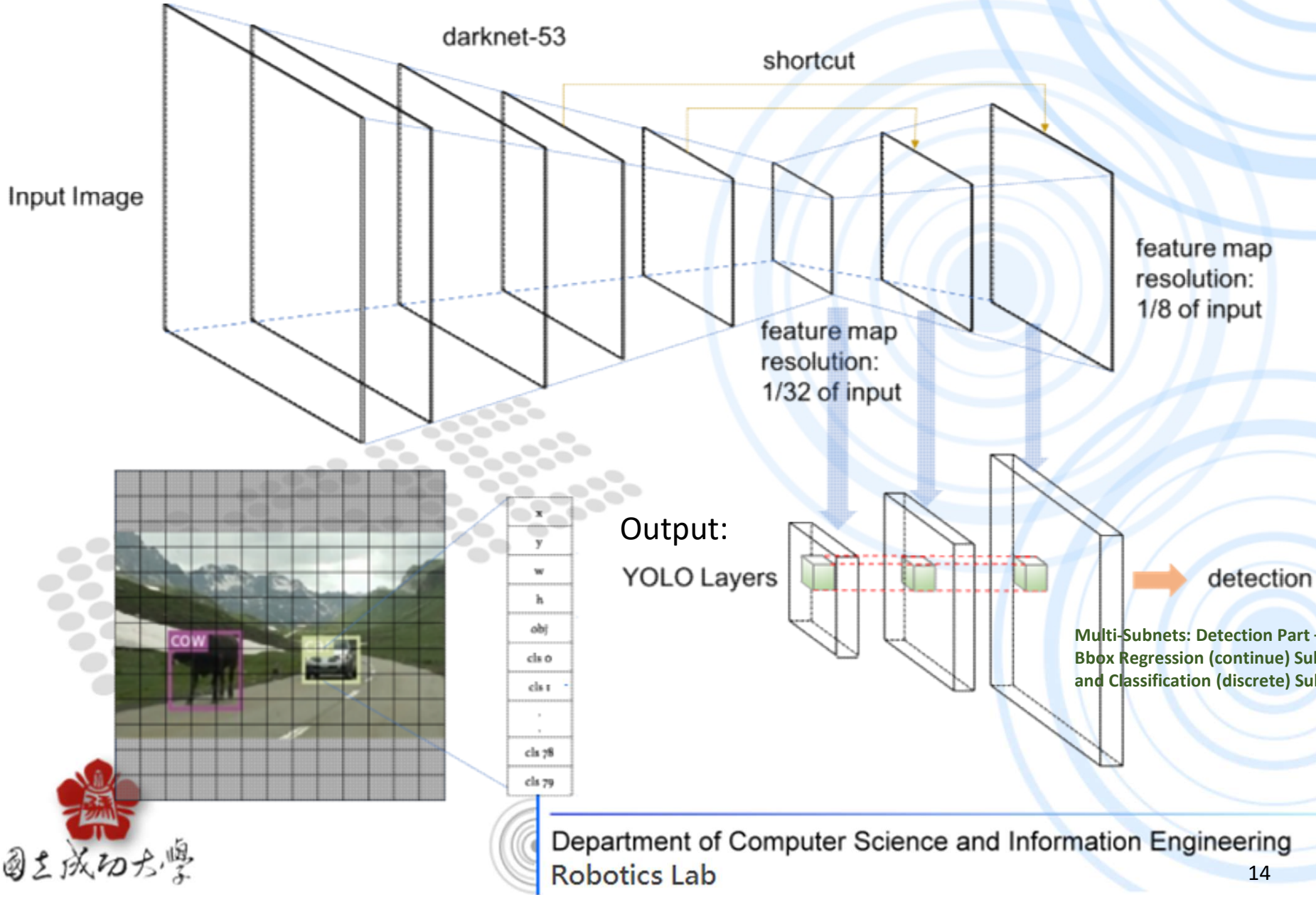


Figure 2. Architecture of the proposed fully convolutional encoder-decoder network.

Reference paper: Yang, J., Price, B., Cohen, S., Lee, H., Yang, M.-H.\ 2016.\ Object Contour Detection with a Fully Convolutional Encoder-Decoder Network.\ arXiv e-prints arXiv:1603.04530.

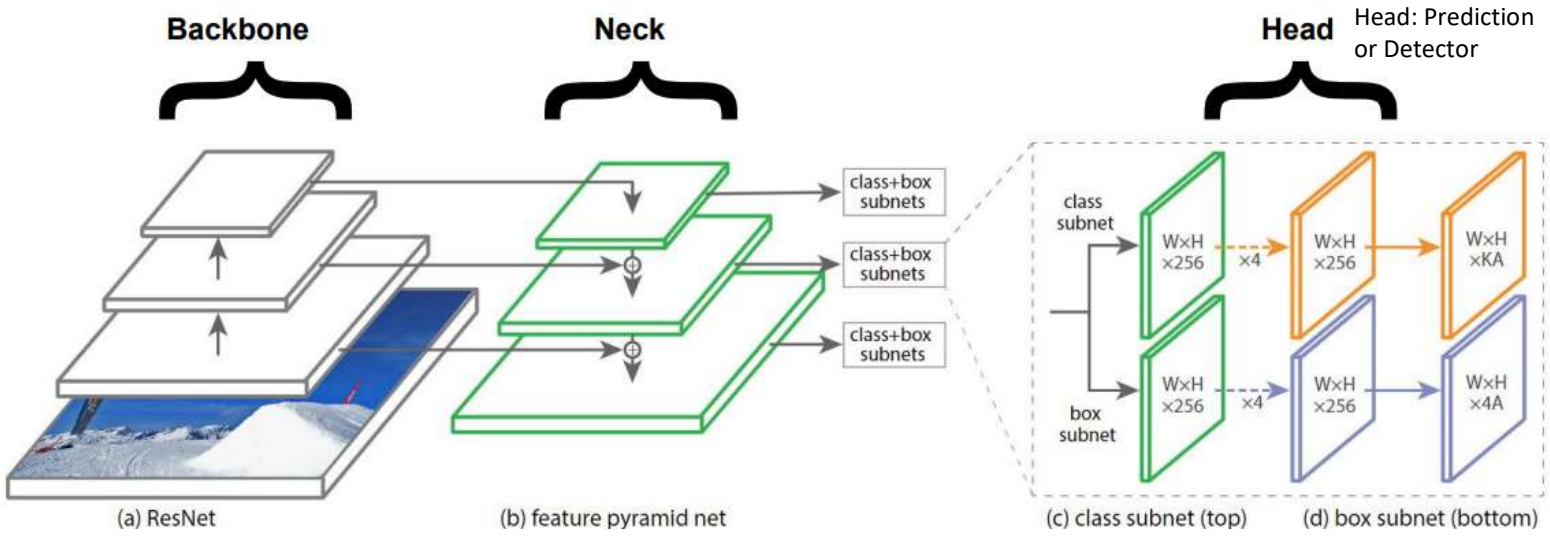
3.0.2.1 YoloV3 in 3D Scenes



3.0.2.2 RetinaNet

- **Purpose**: Handles the Bboxes prediction and Categories classification
- This is a network in charge of actually doing the detection part (classification and regression (Bbox Detection)) of bounding boxes.
- A single output may look like (depending on the implementation):
  - 4 values describing the predicted bounding box coordinates (x, y, h, w)
  - Possibly, a confident value for each bounding box (like in YOLO)
  - The probability of k classes + 1 (possibly, one extra for background like in Faster R-CNN).
- Objected detectors *anchor-based*, like YOLO, apply the head network to each anchor box.
  - Popular one-stage detectors, which are *anchor-based*, are: Single Shot Detector[6] and RetinaNet[4].

- **Following illustration combines the three modules: Backbone, Neck, and Head: RetinaNet.**



3. Multi-Subnets: Detection Part – Bbox Regression (continue) Subnet and Classification (discrete) Subnet

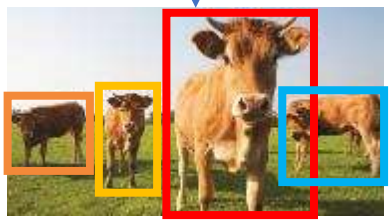
3.0.2.2 Region Proposal Network (RPN)

RPN: Region Proposal Network
SSD: Single Shot MultiBox Detector

One-stage

input image

CNN



Two-stage

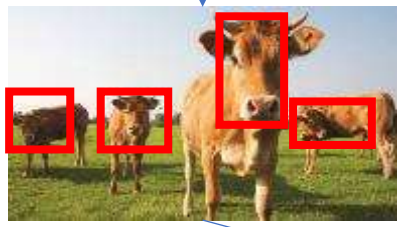
input image

CNN

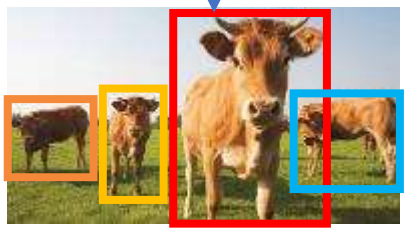
(or Selective Search, etc…)

RPN

region proposals



classifier

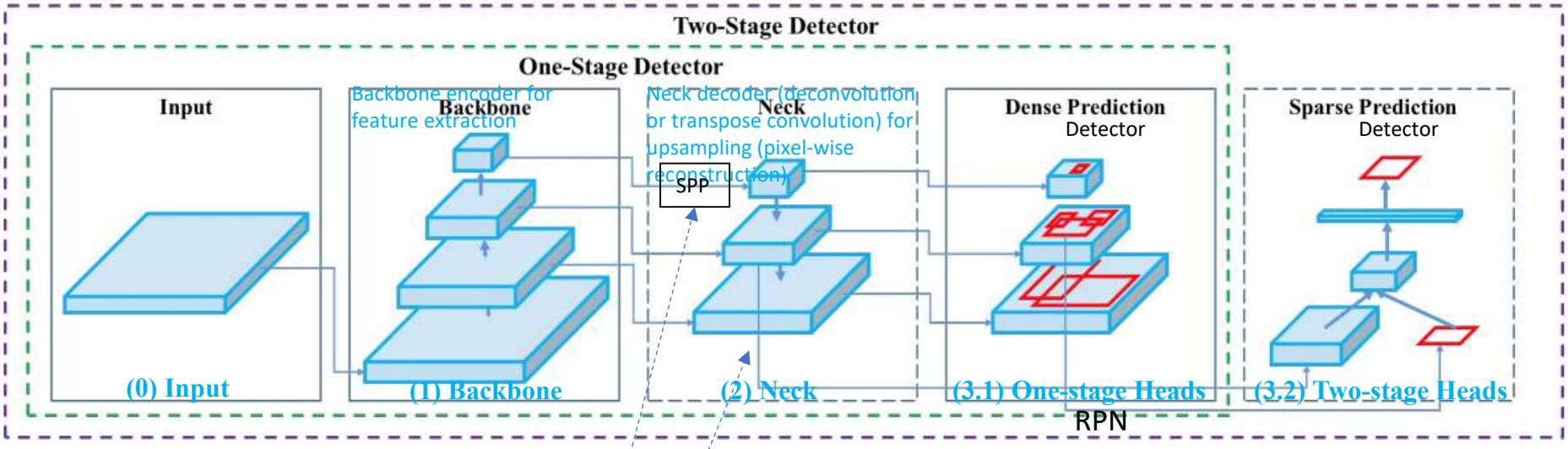


In general,

Shan: Why one stage accuracy is lower??
The extreme foreground-background class imbalance,
Two stage can focus on foreground classes.

	Speed	Accuracy	Examples
One-stage	faster	Lower ??	Yolo, SSD, RetinaNet
Two-stage	slower	higher	Fast R-CNN, Faster R-CNN

3.0.2.3 YoloV4



- Object Detection can be divided into 2 kinds:
  - (1) One-Stage: anchor-based vs anchor free.
  - (2) Two-Stage: anchor-based vs anchor free.
- The right side is all of technologies that the author have been evaluated, modified and integrated into YOLOv4.

(0) • **Input**: Image, Patches, Image Pyramid

(1) • **Backbones**: VGG16 [68], ResNet-50 [26], SpineNet [12], EfficientNet-B0/B7 [75], CSPResNeXt50 [81], CSPDarknet53 [81]

(2) • **Neck**:
- **Additional blocks**: SPP [25], ASPP [5], RFB [47], SAM [85]
- **Path-aggregation blocks**: FPN [44], PAN [49], NAS-FPN [17], Fully-connected FPN, BiFPN [77], ASFF [48], SFAM [98]

(3) • **Heads:**:
- **Dense Prediction (one-stage)**:
  - RPN [64], SSD [50], YOLO [61], RetinaNet [45] (anchor based)
  - CornerNet [37], CenterNet [13], MatrixNet [60], FCOS [78] (anchor free)
- **Sparse Prediction (two-stage)**:
  - Faster R-CNN [64], R-FCN [9], Mask R-CNN [23] (anchor based)
  - RepPoints [87] (anchor free)

(3) Multi-Subnets: Detection Part – Bbox Regression (continue) Subnet and Classification (discrete) Subnet

2) Bag of specials (BoS):
- **Activations**: ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, or Mish

1) Bag of freebies (BoF):
O/P • **Bounding box regression loss**: MSE, IoU, GIoU, CIoU, DIoU

I/P • **Data augmentation**: CutOut, MixUp, CutMix

- **Regularization method**: DropOut, DropPath [36], Spatial DropOut [79], or DropBlock
- **Normalization of the network activations by their mean and variance**: Batch Normalization (BN) [32], Cross-GPU Batch Normalization (CGBN or SyncBN) [93], Filter Response Normalization (FRN) [70], or Cross-Iteration Batch Normalization (CBN) [89]
- **Skip-connections**: Residual connections, Weighted residual connections, Multi-input weighted residual connections, or Cross stage partial connections (CSP)

3.0.2.3 YoloV4 – Backbone + Neck

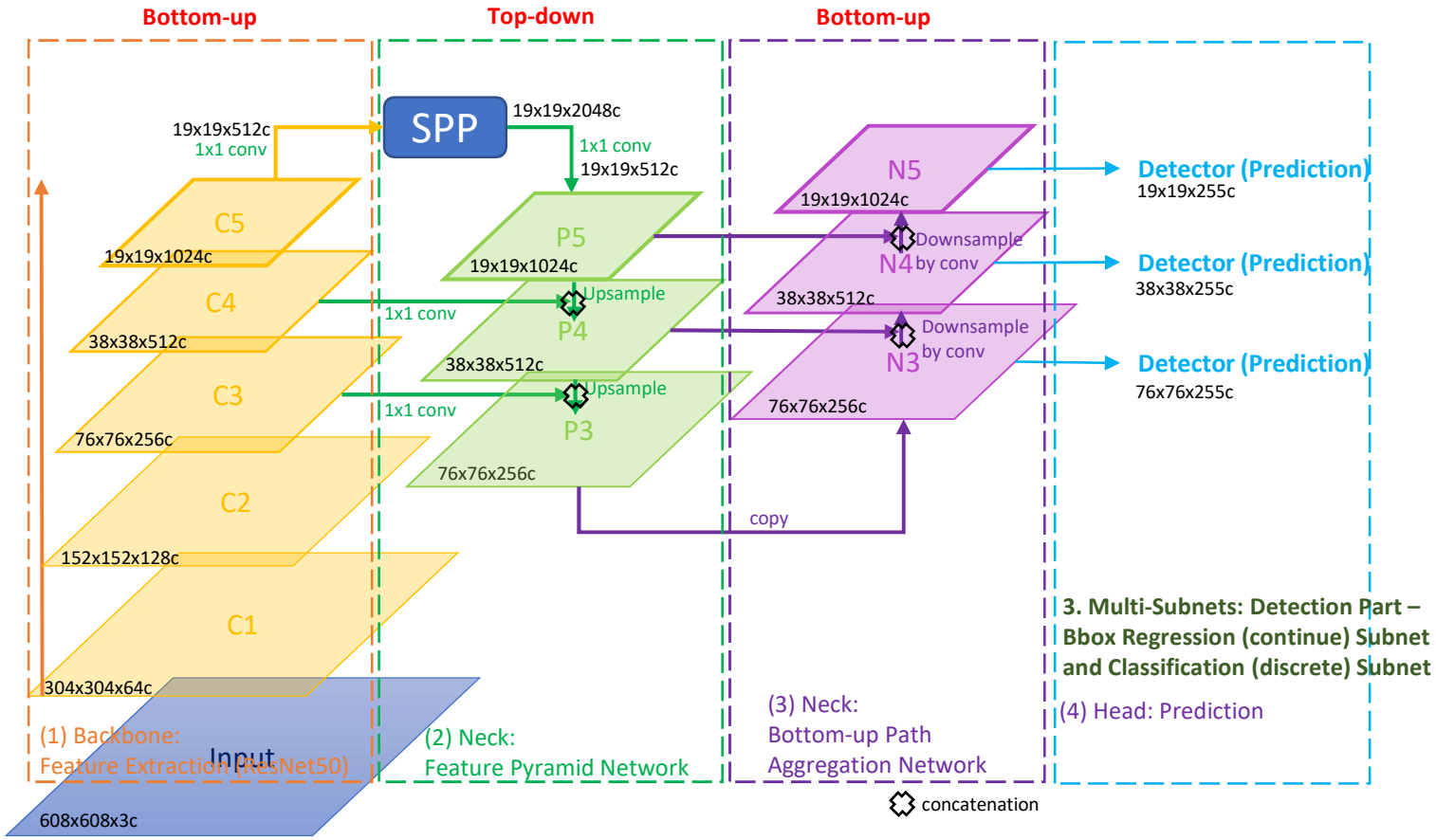
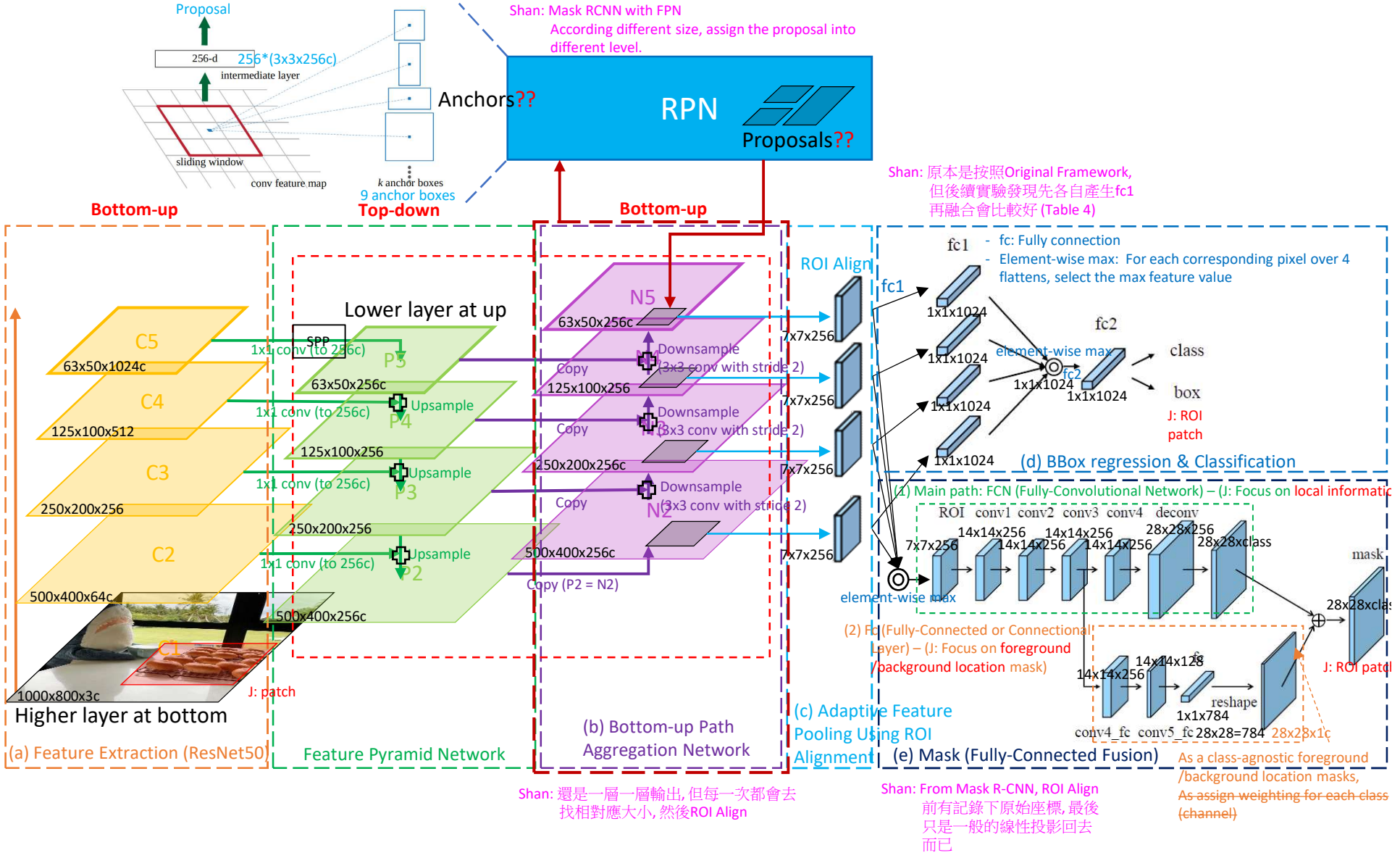


Fig. YOLOv4 demonstration

3.0.2.3 YoloV4 - Neck - PANet (FPN+PAN)



Path Aggregation Network

參數皆為假設論文中沒有提及, wxhxc

⊕ : element addition

3.1 LeNet 1998 – CNN (Convolutional Neural Networks, 2+3 layers) for Optical Character Recognition (OCR)

- LeNet-5: LeCun Network

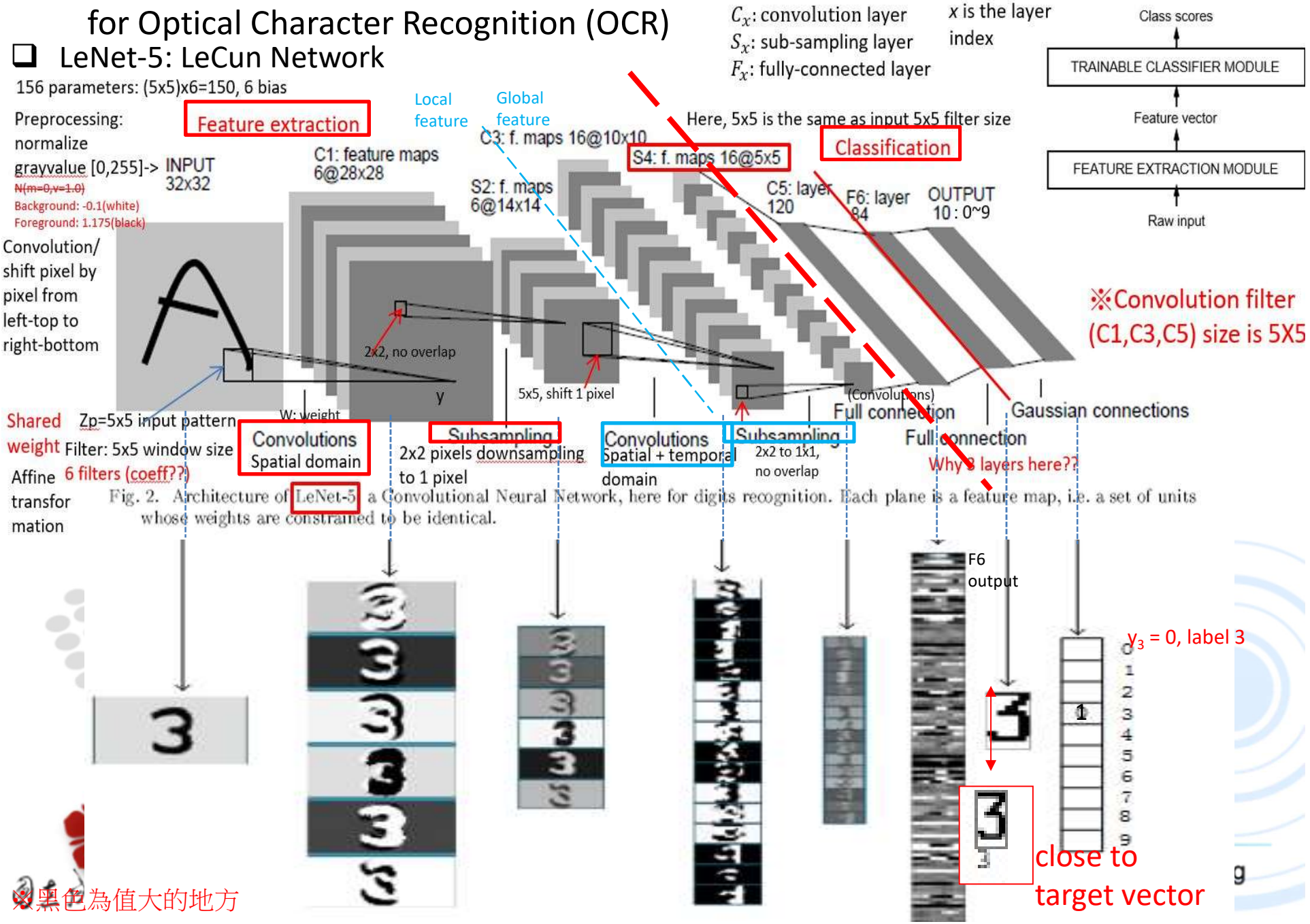
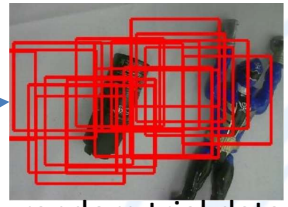


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

3.2 AlexNet:
5 + 3 Fully Connectional Layers (1/2)

δ(j, θ_i) = 1 when gripper angle Θi corresponds to jth bin for ith image patch
δ(j, θ_i) = 0, otherwise
B: batch size, J: exists image patch i=1~B
A_{ji}: forward pass binary activations
l_i: the label corresponding to angle Θi
AlexNet: 2012年, ImageNet比赛冠军的model
i : current iteration ??

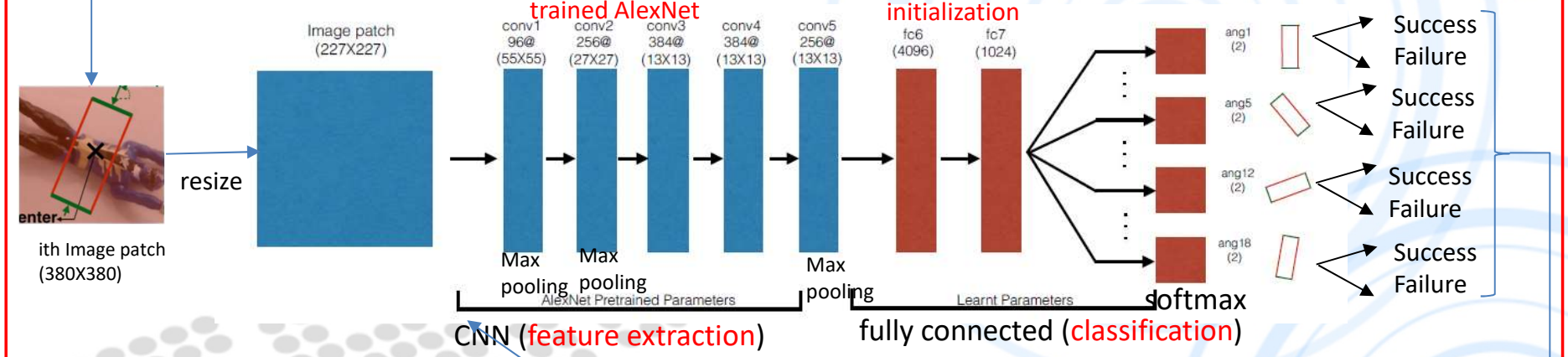
1) Training Stage 1
- the network trained use the random trial data (Collection Stage1 data).
- Learning rate :0.01
- trained over 20 epochs


random trial data

-18 angles x 2 (0 or 1) =36 output
-For ith image patch, each time only one of 36 outputs is labelled *li*
F=1 (successful grasp),
others F=0 (fail grasp)
-For ith image patch:
Angle Θi=1~18
Bin *j*=1~18

Pretrained using Transfer learning from ImageNet-trained AlexNet

Gaussian initialization



??

$$L_B = \frac{1}{N} \sum_{B=1}^{B} \sum_{j=1}^{N=18} \delta(j, \theta_i) \cdot \text{softmax}(A_{ji}, l_i)$$

Batch loss

deep network learnt in the random trial data (Stage1 data).
- Used to collect data in Collection Stage2
- As an initialization parameter use in Training Stage 2

Fig. 5. Our CNN architecture is similar to AlexNet [6]. We initialize our convolutional layers from ImageNet-trained Alexnet.

3.2 Modified Alex: 5 + 3 Fully Convolutional Layers/Networks (FCN) (2/2)

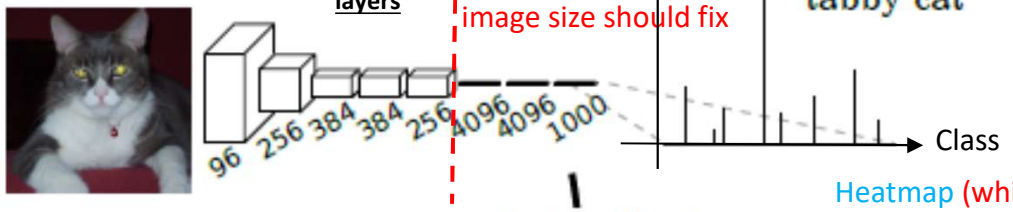
➢ FCN does NOT reduce total parameter number compared with Fully connectional network (MLP)??

J: AlexNet -
Because during the training process,
- the input is fixed size of image,
- the output has 1000 classes, each of which (each class) contains pre-labeling classification image set

Typical recognition nets: LeNet [23], AlexNet [22].

1. fixed-sized inputs
2. non-spatial outputs

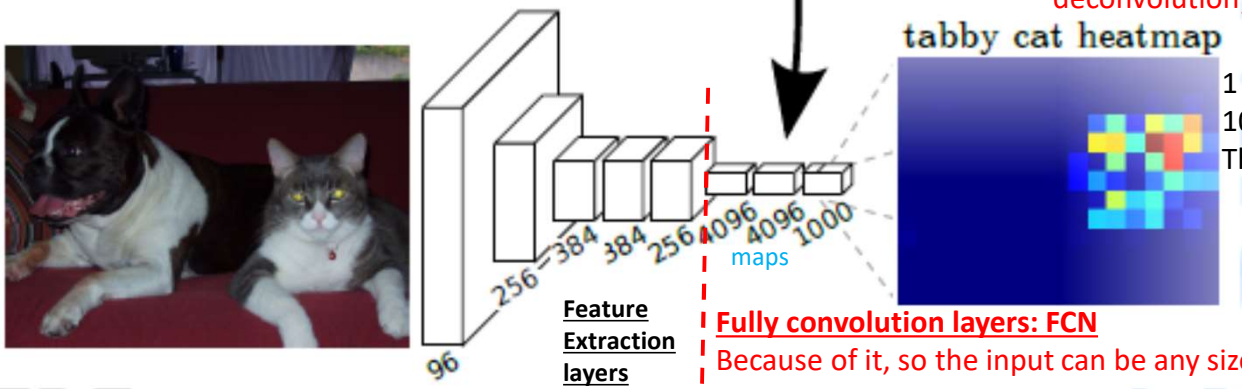
CNN: AlexNet



convolutionalization

Heatmap (which is a Gaussian distribution as probability) as upsampling (or deconvolution) segmentation image

FCN: Modified AlexNet



1 Class -> 1 heatmap
1000 classes -> 1000 heatmaps
The value of pixel: Probability

Figure 2 from [30-Cvpr2015, tPami2017]

Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient ma-chine for end-to-end dense learning.

> Learnable upsampling (or deconvolution) segmentation heatmaps (which is the Gaussian distribution - probability) through
> transpose convolution (initialized with bilinear interpolation) > Fixed-size feature map

J: FCN -
Because during the training process,
- the input is any size of image,
- the output has 1000 classes, each of which (each class) contains pre-labeling classification image set.
> Each pre-classified object / target region at image is also labeled its RoI.
> That is, the output target is labeled 1) classification result with 2) target RoI at image.

3.3 VGG16 or VGG19 2015 – Very Deep CNN (13+3 or 16+3 Layers (as 5 + 3 Layers)) (1/3)

Application:

- Given image → find object name in the image
- It can detect any one of 1000 images
- It takes input image of size 224 * 244 * 3 (RGB image)

Advantage:

- VGG adopts the simplest. (Only 3x3 convolution and 2x2 pooling are used)
- The depth of the network plays an important role.

Built using:

(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

Drawback:

- Network is usually big. (containing around 160M parameters).
- Most of the parameters are consumed in the fc layers.

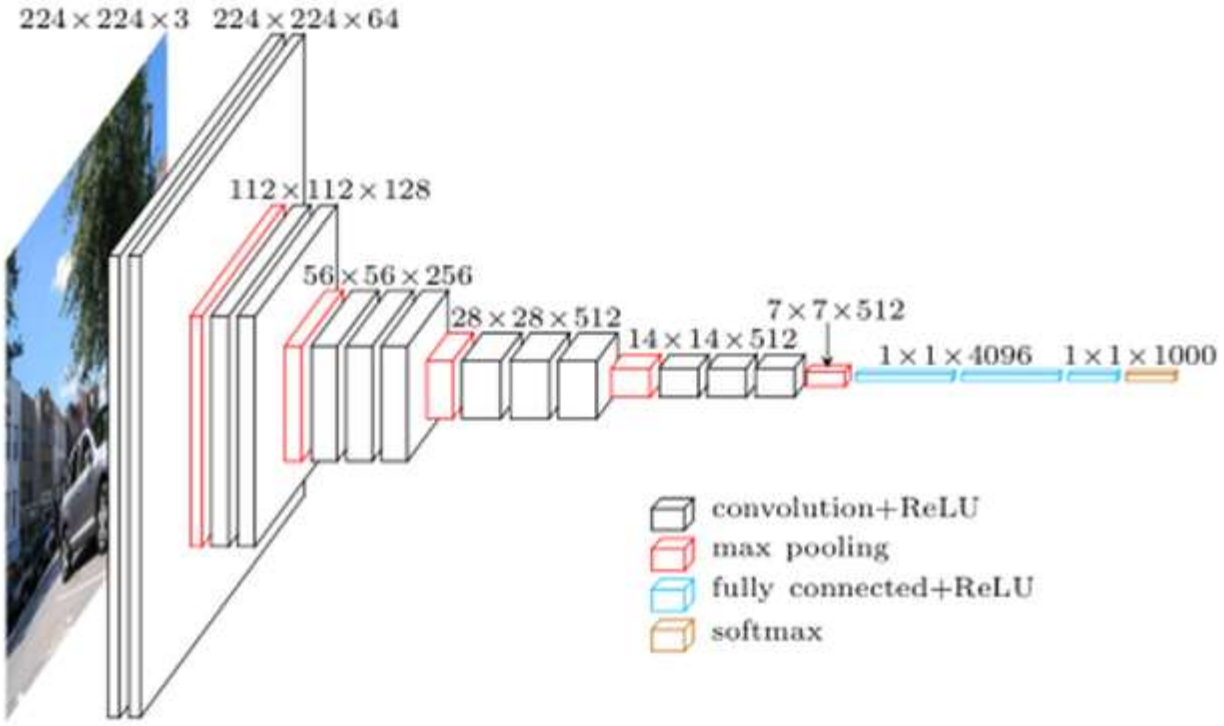


Fig. VGG16 for feature extraction

[Ref] DeepLearn_Rcnn_Faster_DetectObjSeg_Nips2015_20180726_V1Beta4_Phu.pptx

3.3 VGG16 or VGG19 2015 – Very Deep CNN (13+3 or 16+3 Layers (as 5 + 3 Layers)) (1/2)

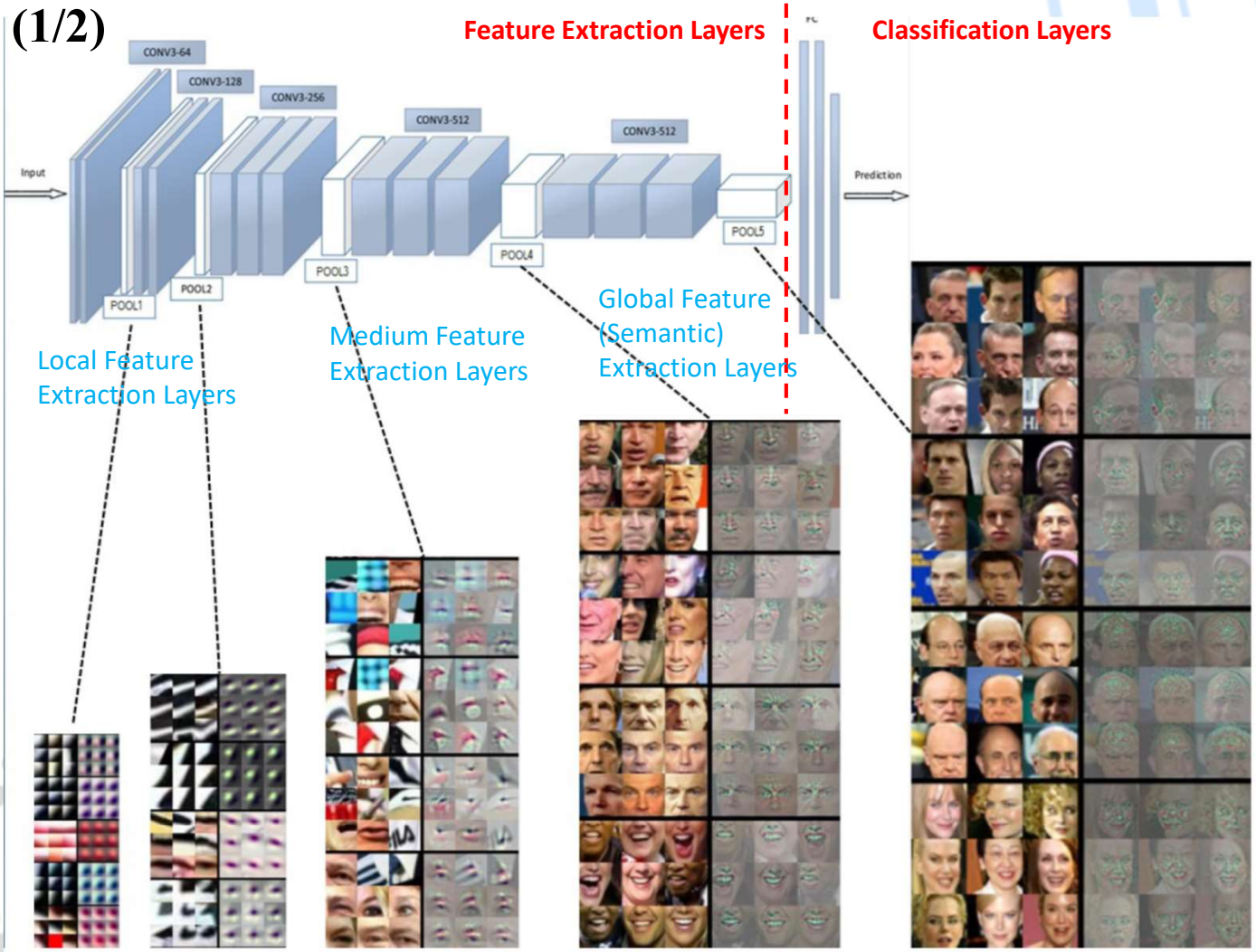


Fig. 2. The hierarchical architecture that stitches together pixels into invariant face representation. Deep model consists of multiple layers of simulated neurons that convolute and pool input, during which the receptive-field size of simulated neurons are continually enlarged to integrate the low-level primary elements into multifarious facial attributes, finally feeding the data forward to one or more fully connected layer at the top of the network. The output is a compressed feature vector that represent the face. Such deep representation is widely considered the state-of-the-art technique for face recognition.

3.3 VGG16: 13 + 3 Layers (as 5 + 3 Layers) (2/2)

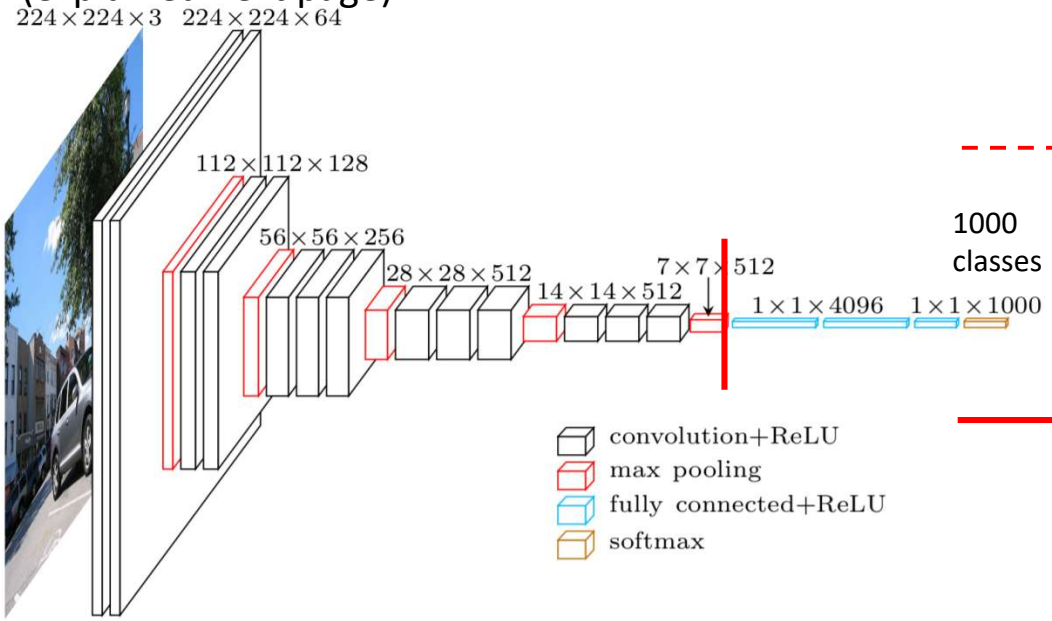
- Spatial filter size: 3×3
- Stride: 1
- Padding: 1
- Non-overlap pooling
- Total parameters:

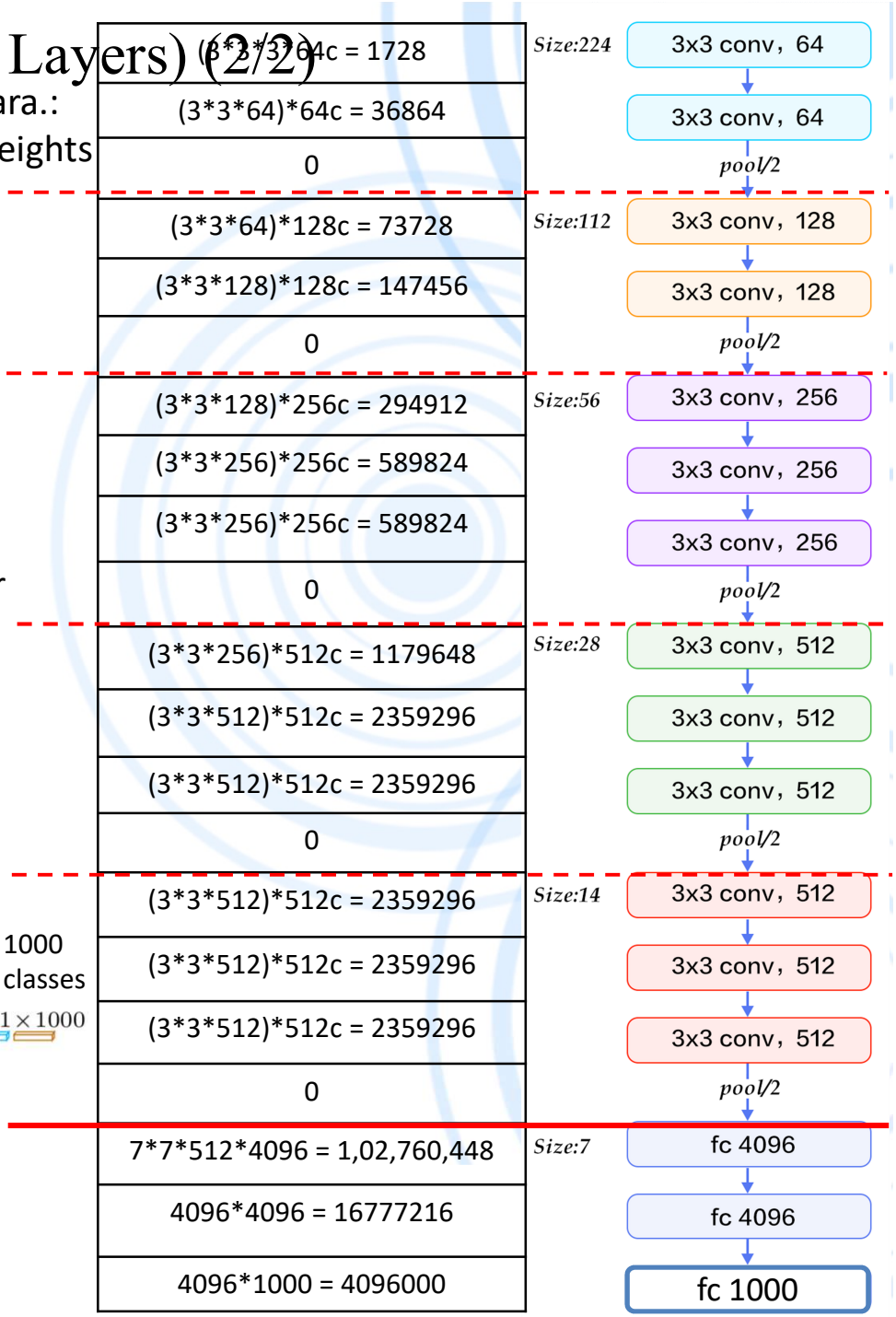
$$params = weights + biases\ (in\ FC)$$
$$= 138344128 + 4096 * 3 = 138{,}356{,}416$$

→ Instead of using large spatial filter size (7×7), use small one 3×3

→ reduce parameters, and increase the net depth (or # of ReLU)

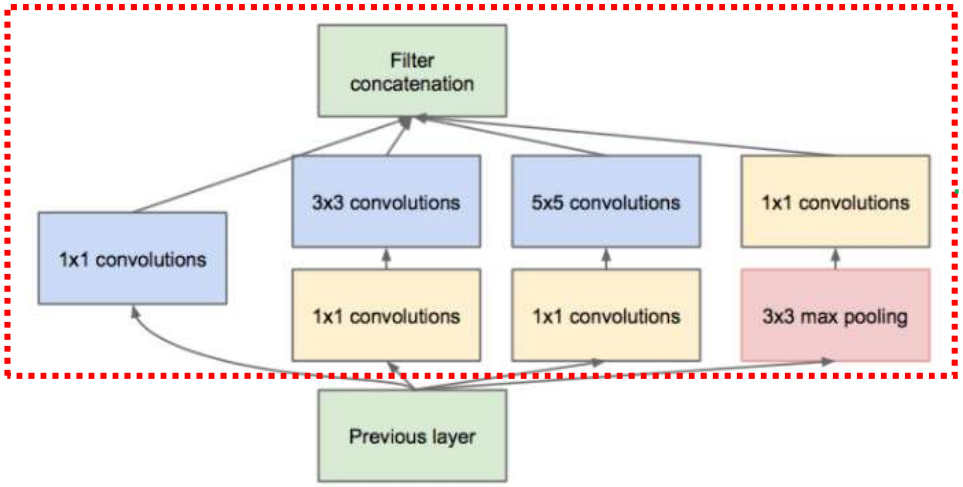
→ make the decision boundary more discriminative (explained next page)





3.4 GoogleNet

1) Inception module:



(b) Inception module with dimensionality reduction

2) GoogleNet:

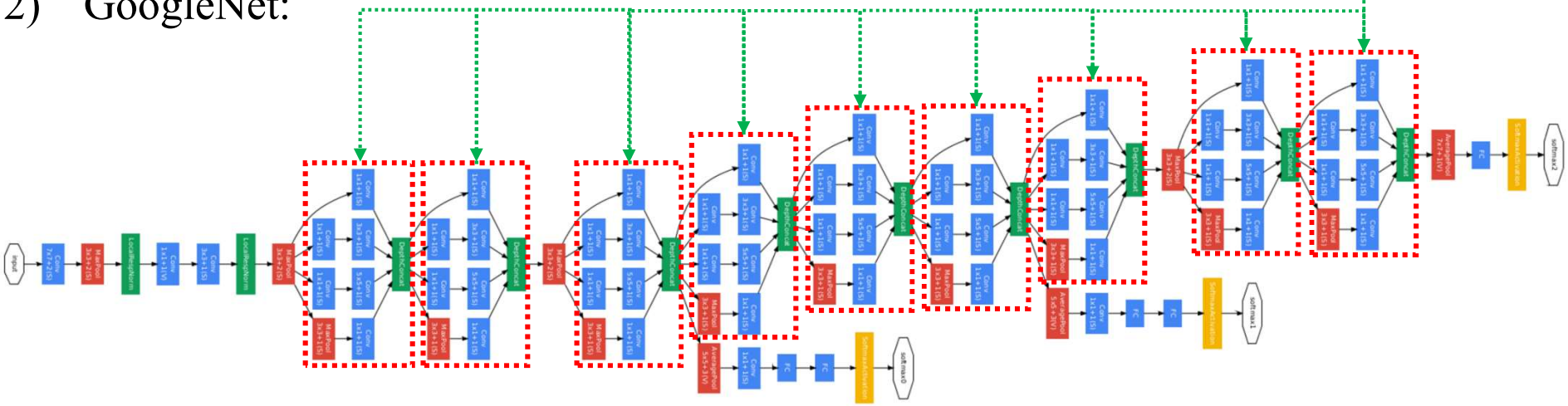


Figure 3: GoogLeNet network with all the bells and whistles.

3.5 ResNet

1) Residual module:

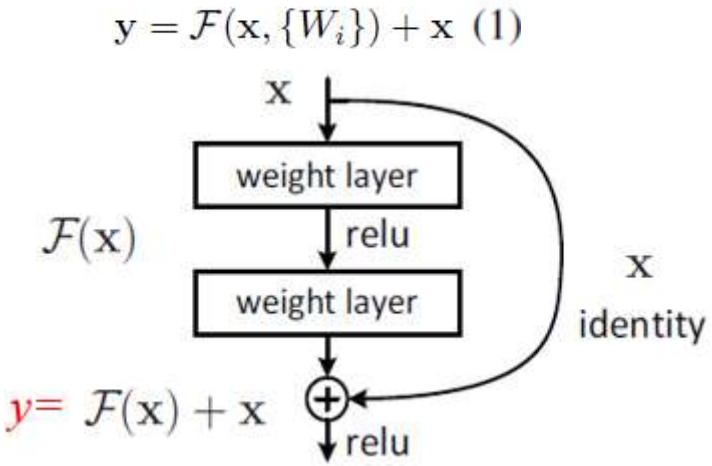
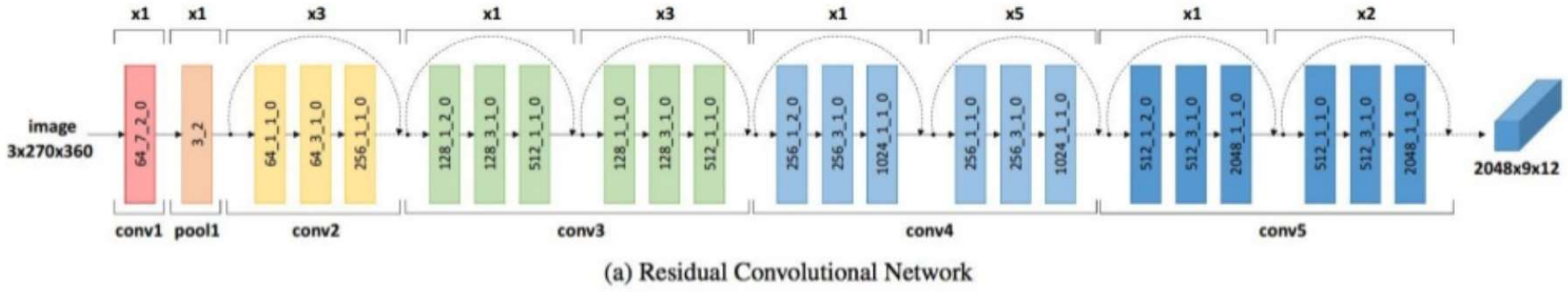


Figure 2. Residual learning: a building block.

Built using :

- Stack residual blocks (Identity_blocks)
- Periodically, double filter and down-sample spatially using stride 2. (/2 in each dimension)
- Additional conv layer at the beginning.
- No FC layers at the end (only FC 1000 to output classes)

2) ResNet:



(a) Residual Convolutional Network

Figure 3: RestNet network

3.5 ResNet: Deep Residual Learning – 50, 101…

Network Architectures: /2: Stride 2

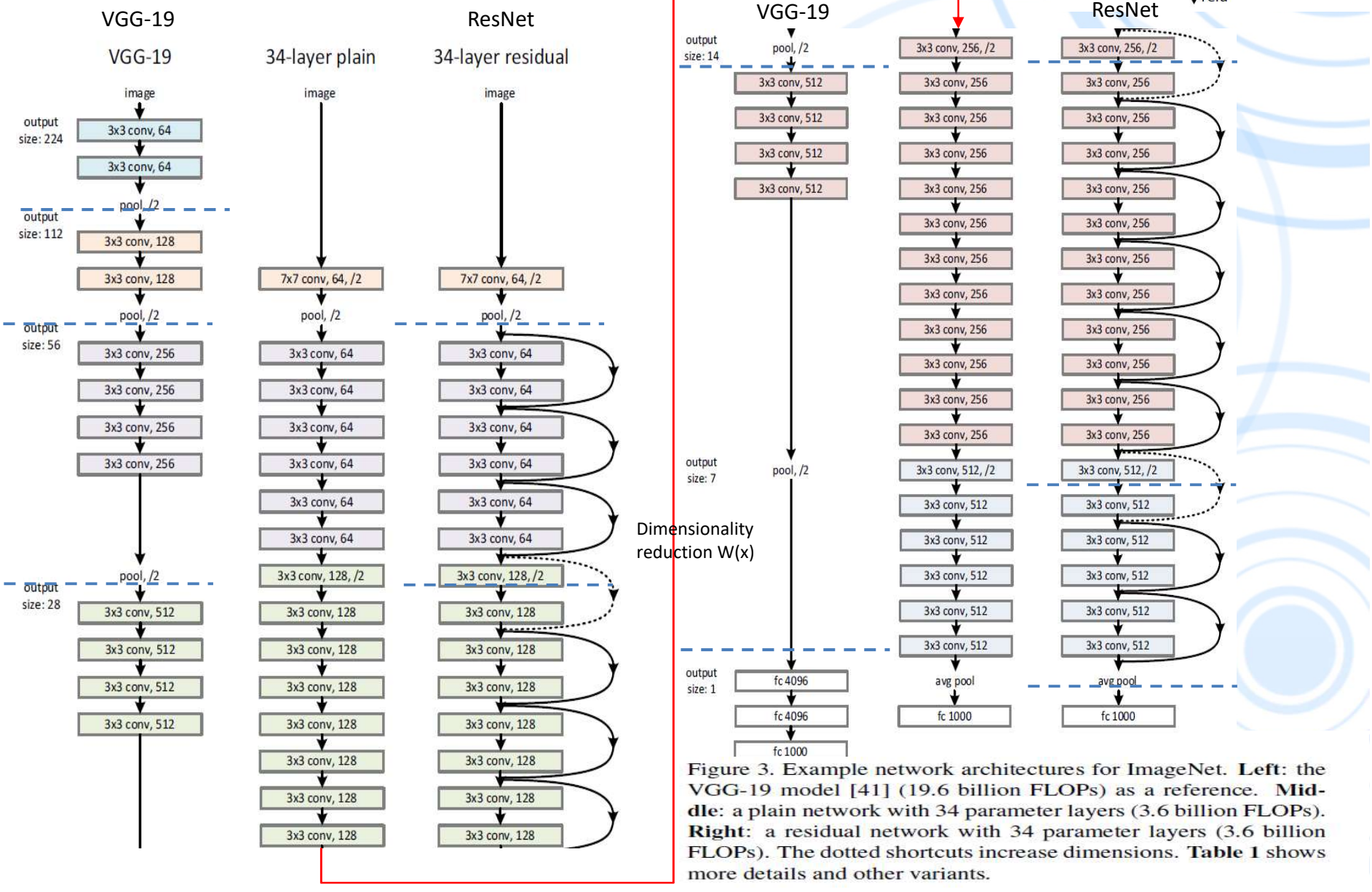
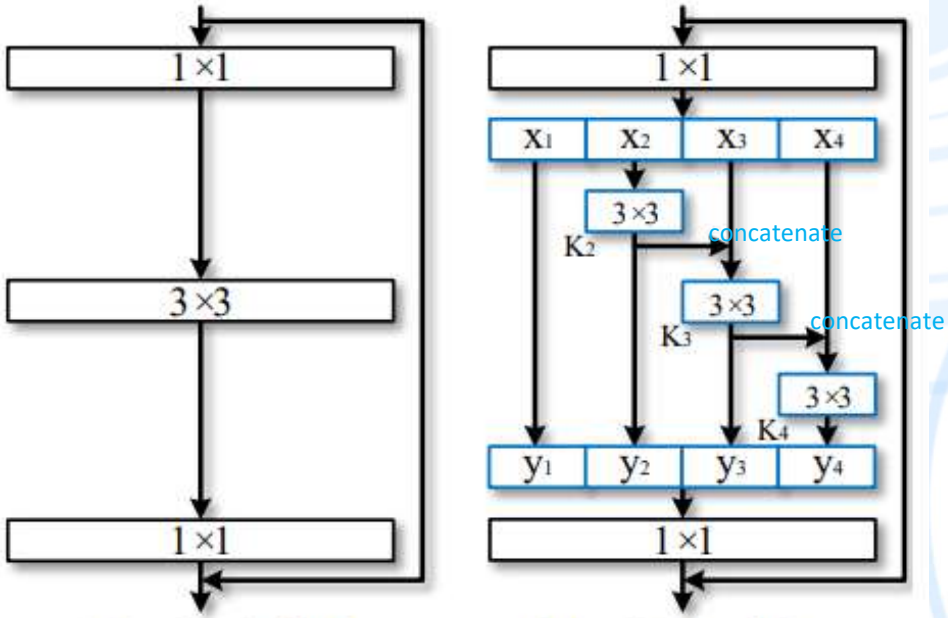


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

3.6 Res2Net

- Different from improving the multi-scale ability by utilizing features with different resolutions, the multi-scale of our proposed method refers to the **multiple available receptive fields** at a more granular level. $3 \times 3 \times n = s * 3 \times 3 \times w$, $n = s \times w$, n: Channels, w: Channels, s: Scales
- We replace the 3×3 filters 1 of n channels, with a set of smaller filter groups, each with w channels (without loss of generality we use $n = s \times w$).
- These smaller filter groups are connected in a hierarchical residual-like style to increase the number of scales that the output features can represent.



X1~x4, S=4

3x3xw (channel)

J: similar to Xception

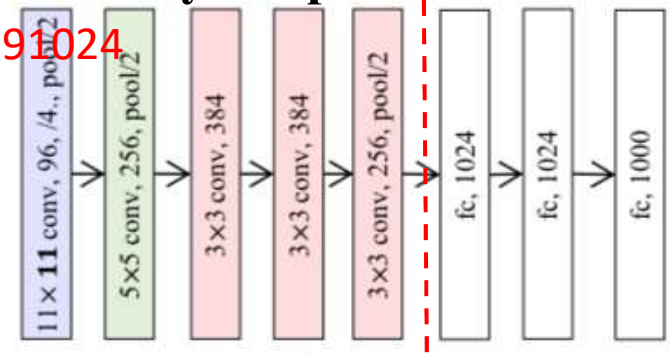
Why bottleneck is better??

Shan: 可以降維減少參數，增加深度

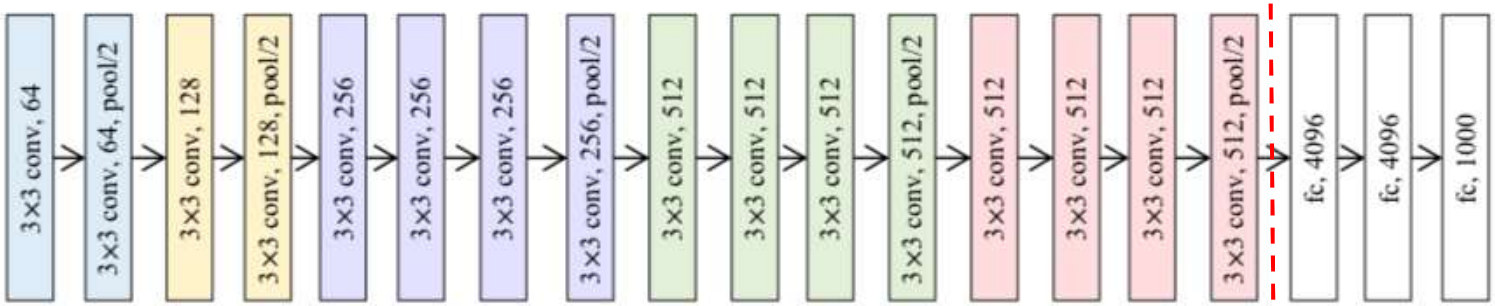
Fig. 2: Comparison between the bottleneck block and the proposed Res2Net module (the scale dimension s = 4).

3.7 Other Very Deep CNNs

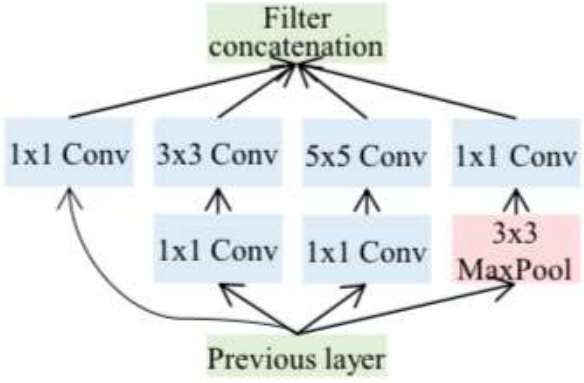
G: 20191024



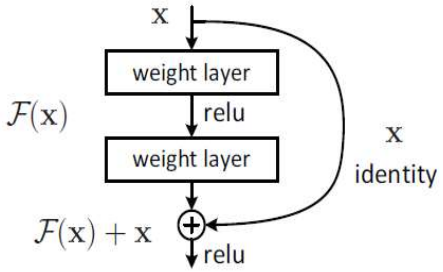
(a) Alexnet 5+3 layers



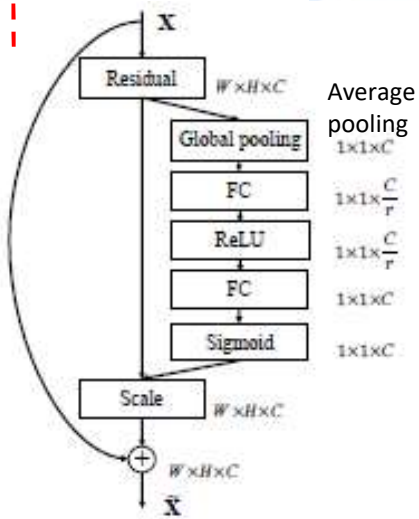
(b) VGGNet 16 (5 blocks) +3 = 19 layers



(c) GoogleNet



(d) ResNet

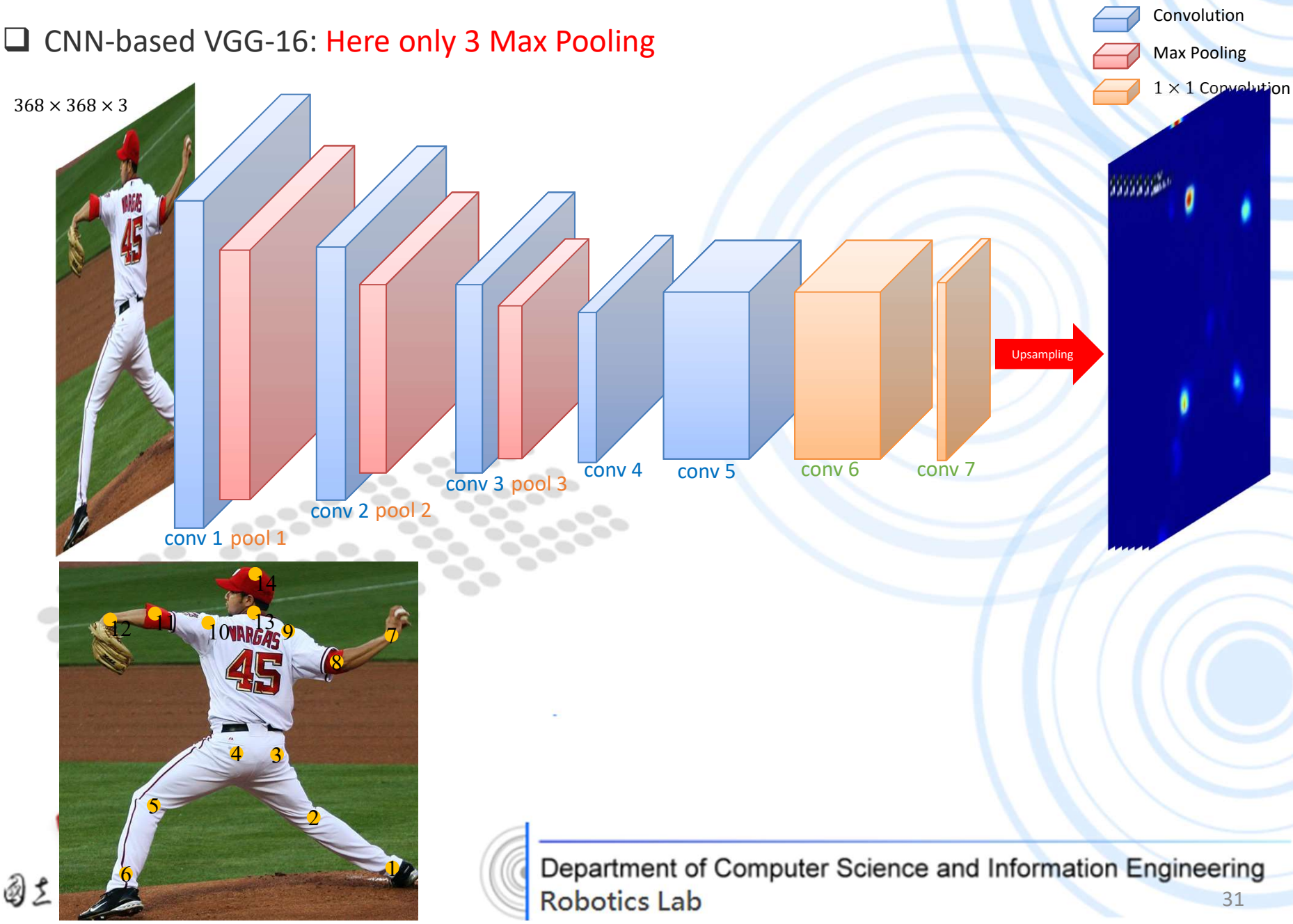


(e) SENet

Fig. 9. The architecture of Alexnet, VGGNet, GoogleNet, ResNet, SENet.

3.8.1 Application 1 - Body Joint Point Estimation Using VGG-16 (1/2)

- CNN-based VGG-16: Here only 3 Max Pooling



3.8.1 Application 1 - Body Joint Point Estimation Using VGG-16 (2/2) 01:12.55

3.8.1 Application 1 - Body Joint Point Estimation Using Deep Learning

Real-time Multi-Person 2D Pose Estimation Using Part Affinity Fields

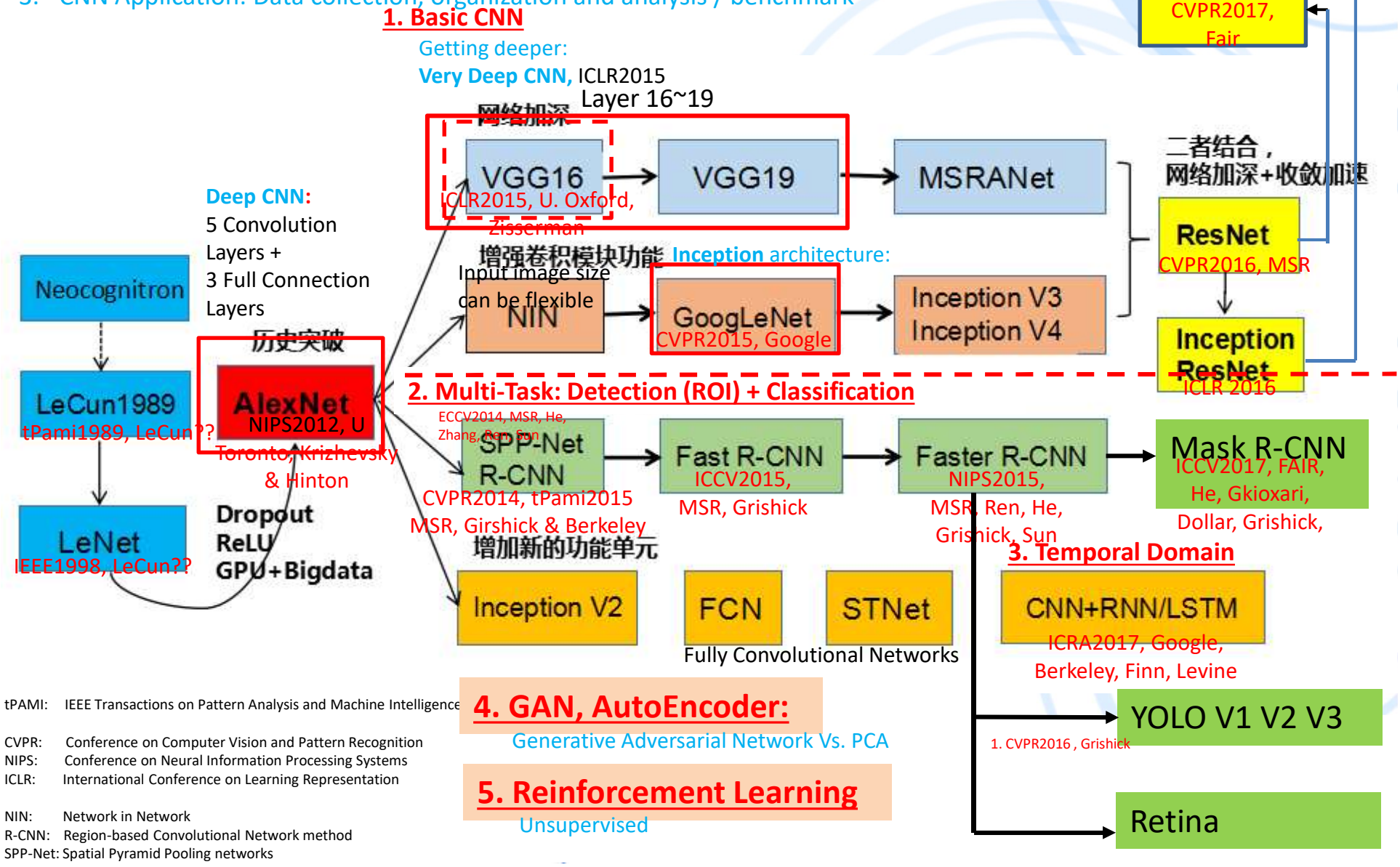
Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh

Carnegie Mellon University

4.0 Deep Learning: 2. Multi-Task

Which one do I want to be?

1. Basic CNN Creation
2. Modified CNN
3. CNN Application: Data collection, organization and analysis / benchmark



tPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence

CVPR: Conference on Computer Vision and Pattern Recognition
NIPS: Conference on Neural Information Processing Systems
ICLR: International Conference on Learning Representation

NIN: Network in Network
R-CNN: Region-based Convolutional Network method
SPP-Net: Spatial Pyramid Pooling networks

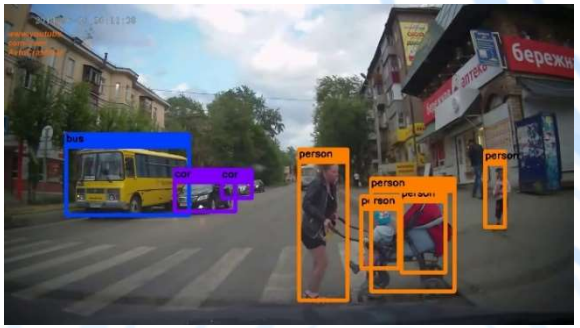
4. GAN, AutoEncoder:

Generative Adversarial Network Vs. PCA

5. Reinforcement Learning

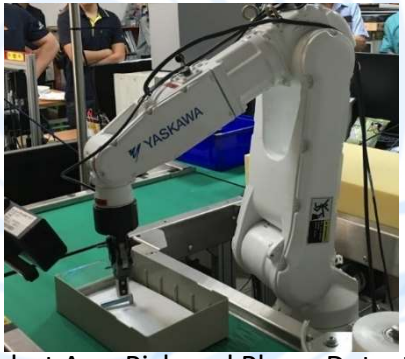
Unsupervised

4.0 Multi-Task: Application

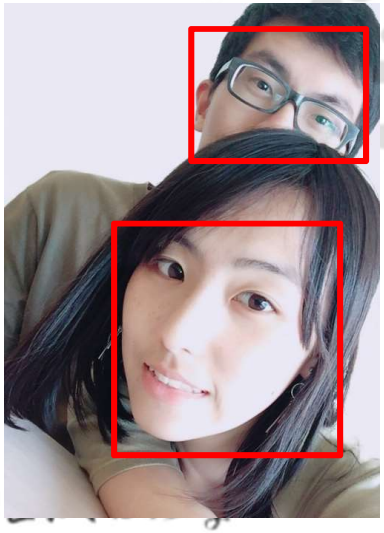


1. ADAS: Detect and classify cars, people, scooters…

5. Sports: Detect and classify people and sports balls

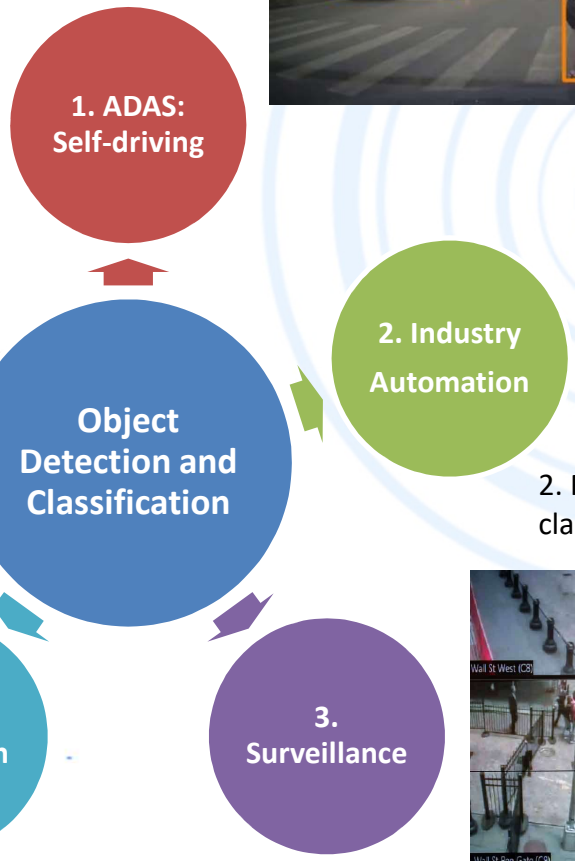


2. Robot Arm Pick and Place: Detect and classify objects (ex: screws, screwdrivers…)



4. Face recognition: includes both detection and recognition

3. Surveillance: Detect and classify people, cars…



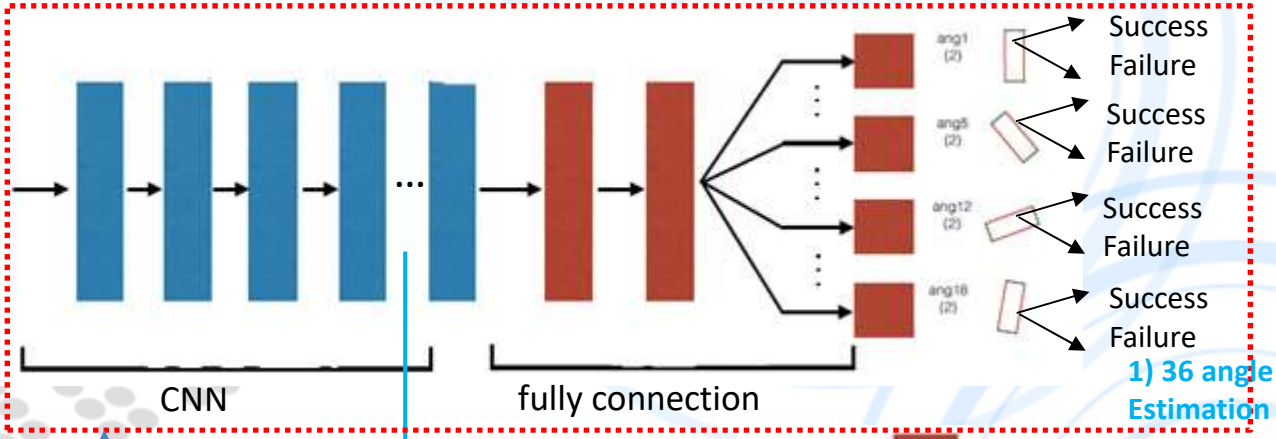
4.1 Multi-Task: Visual-Guided Robot Arm Using VGG-16 – Object Detection, Classification and Picking Angle Estimation

Training:
- the network trained use the random trial data.
- Learning rate :0.0001
- trained over 10 epochs

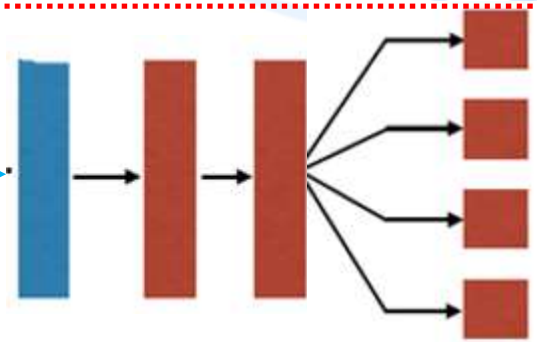
random trial data

Image patch
(224*224)

224*224

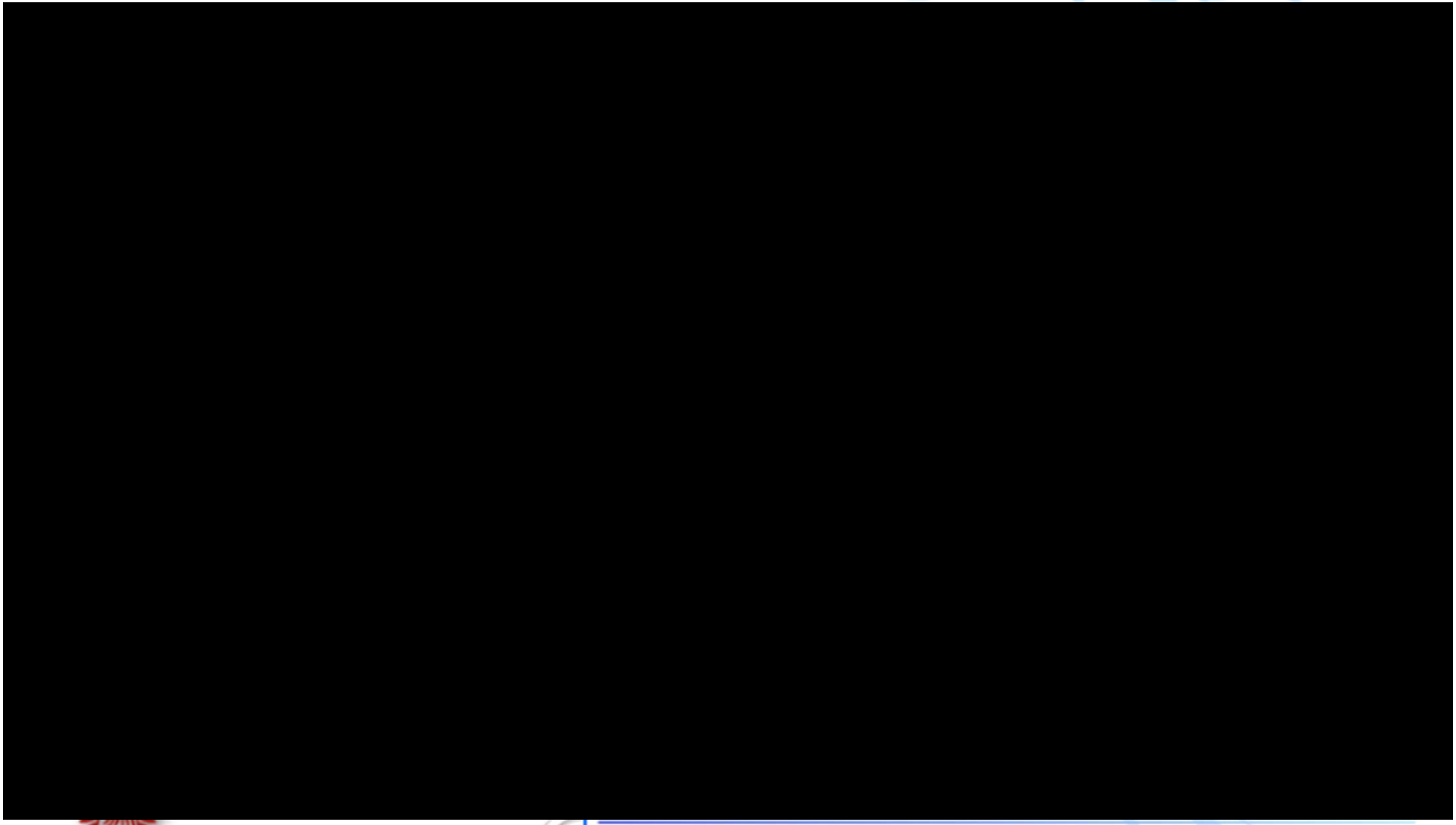


Loss: Cross-entropy



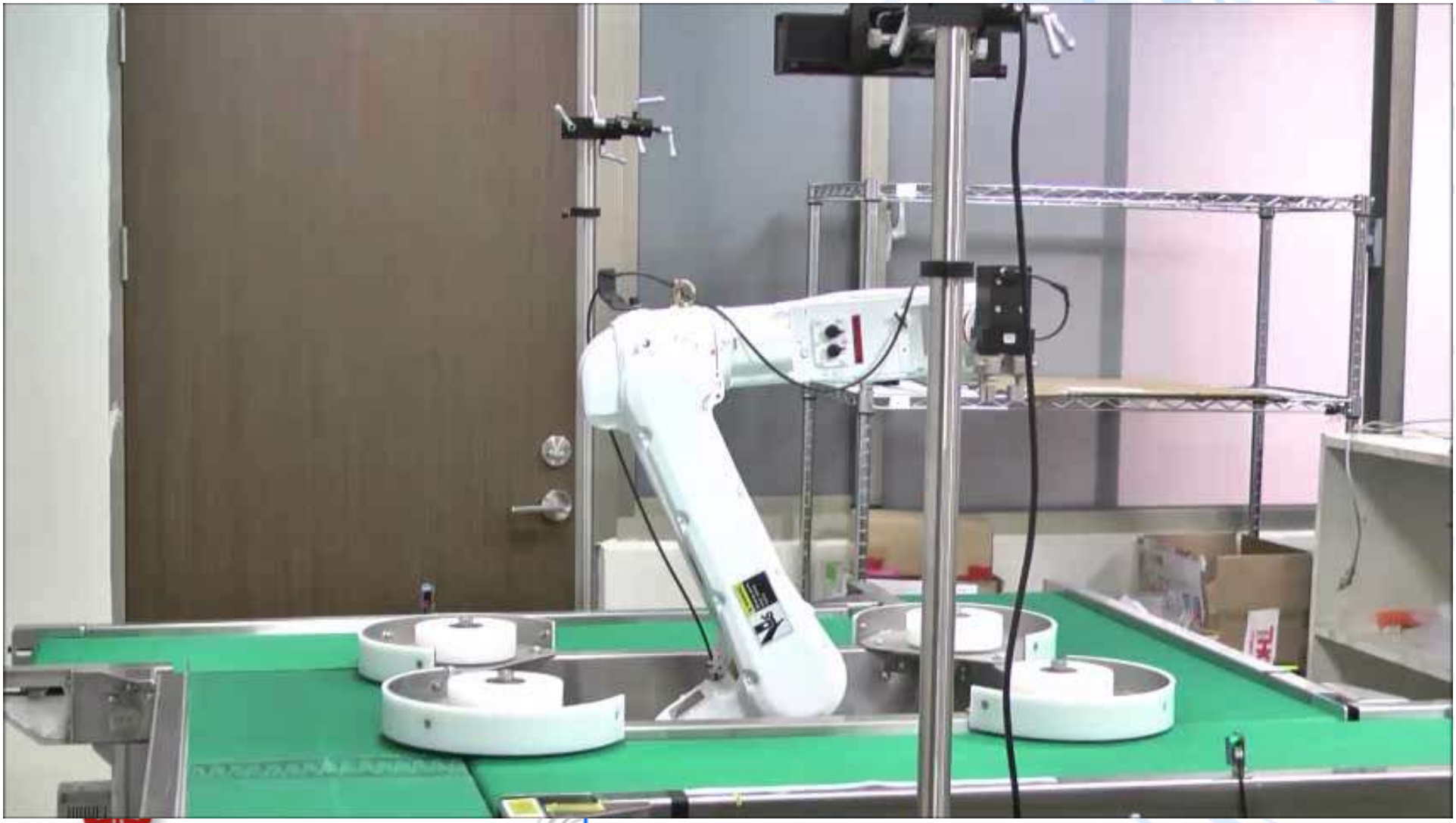
4.1 Multi-Task: Visual-Guided Robot Arm Using VGG-16 - Object Detection, Classification and Picking Angle Estimation: Training Process 01:31.61

4.1 Multi-Task: Visual-Guided Robot Arm Using VGG-16 - Object Detection, Classification and Picking Angle Estimation: Test Process 02:14.35



4.1 Visual-Guided Robot Arm Using Machine Learning: Static Object

4.1 Multi-Task: Visual-Guided Robot Arm Using Deep Learning - Object Detection, Tracking and Picking Angle Estimation 02:14.35



4.2 Multi-Task: Detection, Segmentation, and Classification

Which order is better?

1) BBox 2) Mask 3) Classification

1) BBox 2) Classification 3) Mask

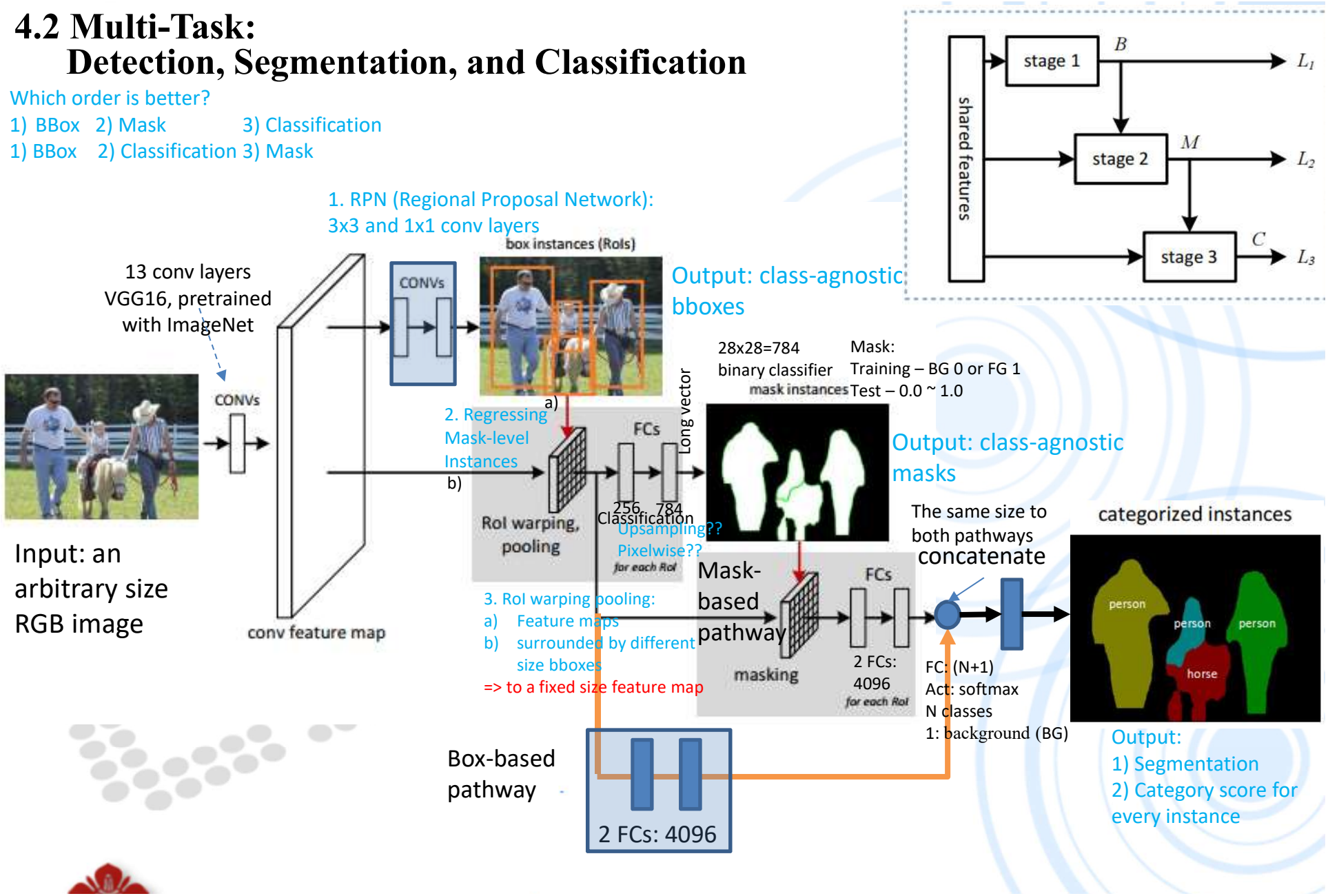


Figure 2. Multi-task Network Cascades for instance-aware semantic segmentation. At the top right corner is a simplified illustration.

4.3 R-CNN Background Knowledge and Comparison ??

1.1) R-CNN (Region-CNN, CVPR2014, He):

- Use Selective Search + CNN (AlexNet)
- RoI: Bounding Box Segmentation

1.2) SPPNet (Spatial Pyramid Pooling Network):

2) Fast R-CNN (ICCV2015, Girshick):

- Use Selective Search + FCN (Modified AlexNet or VGG16)
- RoI: Bounding Box Segmentation

3) Faster R-CNN (NIPS2015, Girshick+He):

- Use RPN + FCN (Modified ??)
- RoI: Bounding Box Segmentation

4) Mask R-CNN (ICCV2017, Girshick+He):

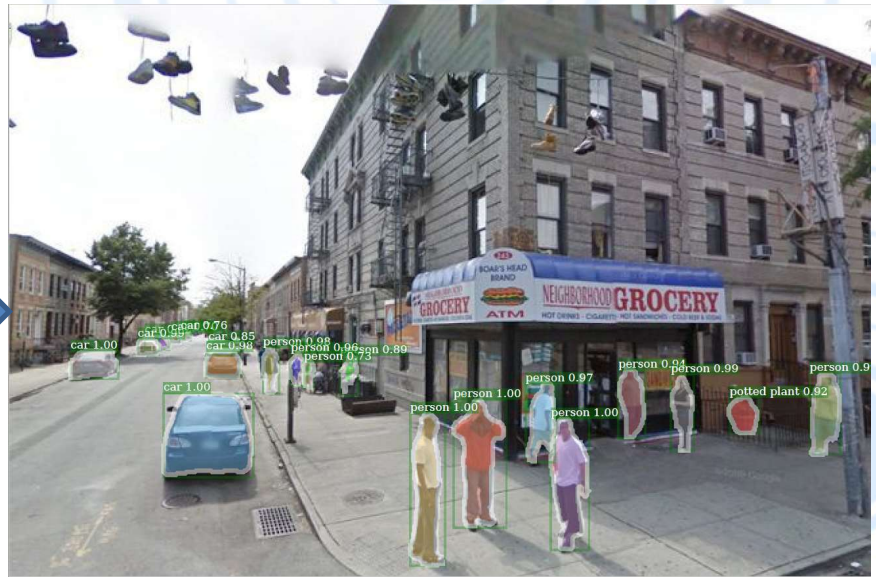
- Use RPN + FCN (Modified ResNet)
- RoI: Pixel-wise Segmentation

For later layers:
(1) CNN - AlexNet: Fully Connectional Network (MLP) for classification
(2) FCN – Modified AlexNet or VGG16: Fully Convolutional Network for classification + RoI

RPN: Region Proposal Network
-Input: An image (of any size)
-Output: A set of rectangular object proposals (RoI), each (RoI) with an object score

4.3 Multi-Task: Faster R-CNN: Goal – Detection + Bbox Segmentation + Classification
Mask R-CNN: Goal – Detection + Pixel-Wise Segmentation + Classification

Faster R-CNN: Output -
(1) Classification label ~~+ Score~~
(2) RoI: Bounding box
~~(3) Mask~~



Input: Image

Mask R-CNN: Output -
(1) Classification label + Score
(2) RoI: Bounding box
(3) Mask: Pixel-wise

4.3 Multi-Task: Faster R-CNN: Goal – Detection + Bbox Segmentation + Classification
Mask R-CNN: Goal – Detection + Pixel-Wise Segmentation + Classification

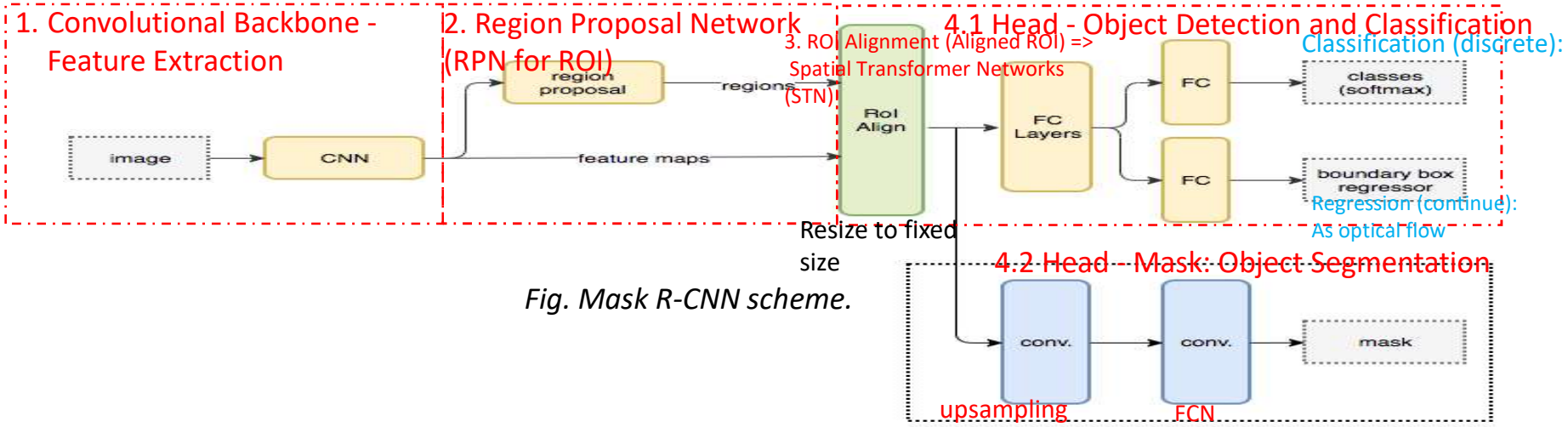
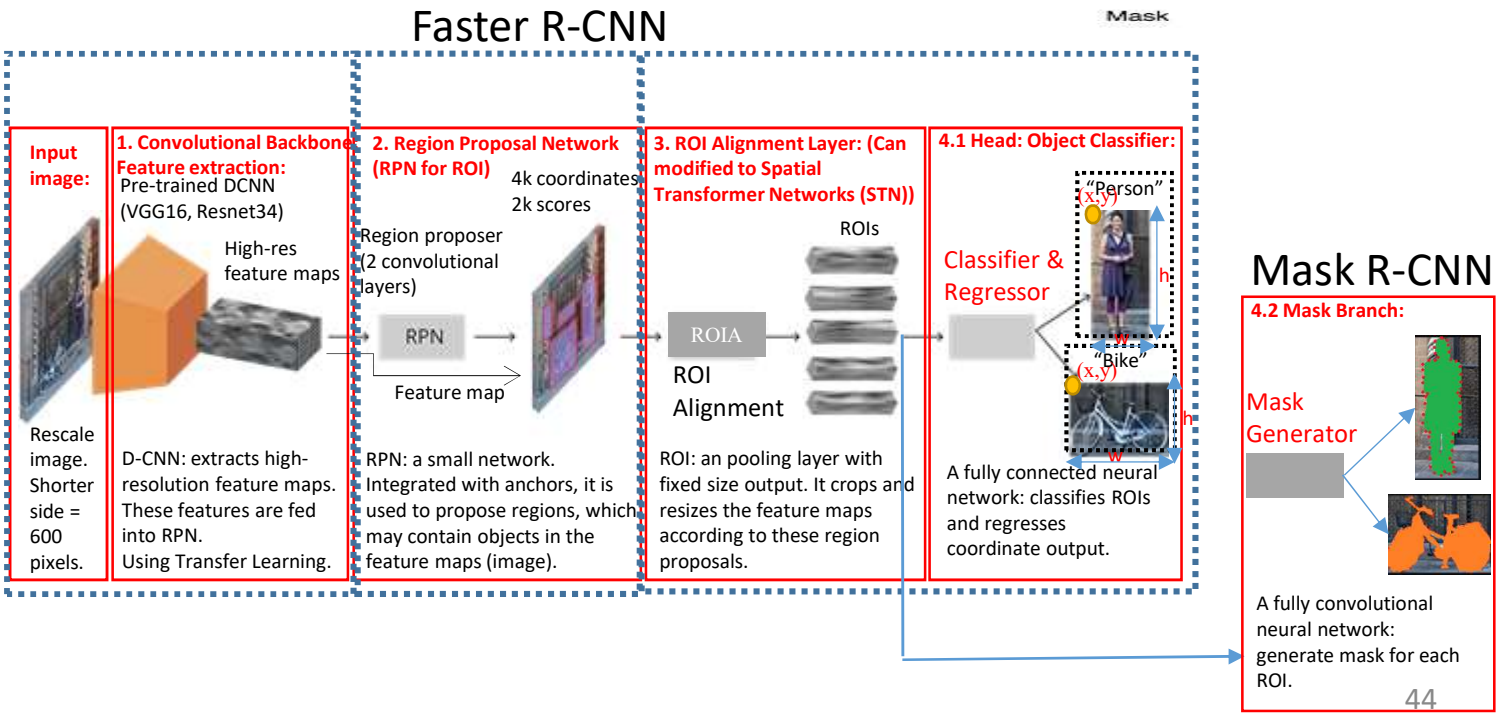


Fig. Mask R-CNN scheme.



4.3 One- Vs. Two-Stages: SSD, Yolo, RetinaNet

-RPN: Region Proposal Network

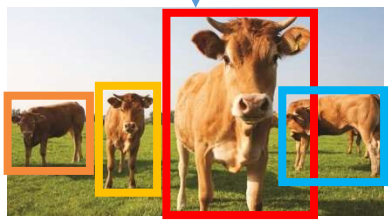
-SSD: Single Shot MultiBox Detector

One-stage

input image

CNN

Dense samples



Two-stage

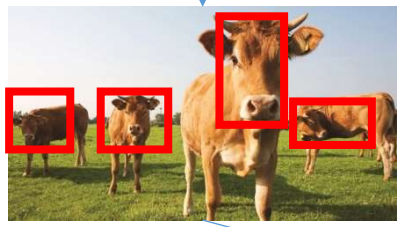
input image

CNN

(or Selective Search, etc…)

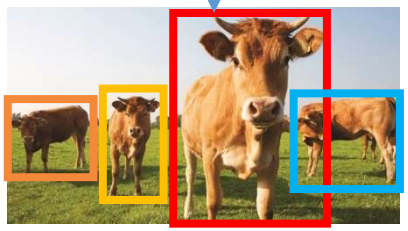
RPN

region proposals



Sparse samples after RPN selection, so this will reduce many background regions

classifier



In general,

	Speed	Accuracy	Examples
One-stage	faster	Lower ??	Yolo, SSD, RetinaNet
Two-stage	slower	higher	Fast R-CNN, Faster R-CNN

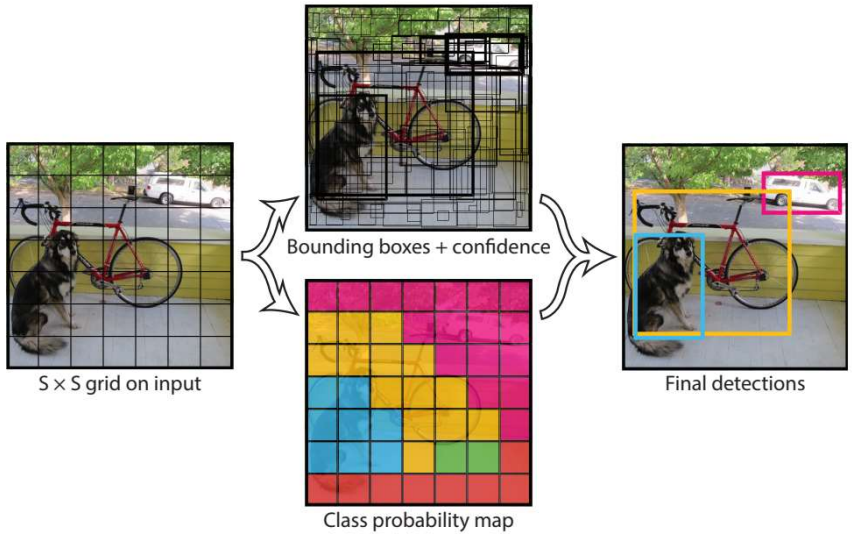
4.3 SSD, Yolo, RetinaNet: One-Stage Vs. Two-Stage

Ex: Background Vs. Foreground classes

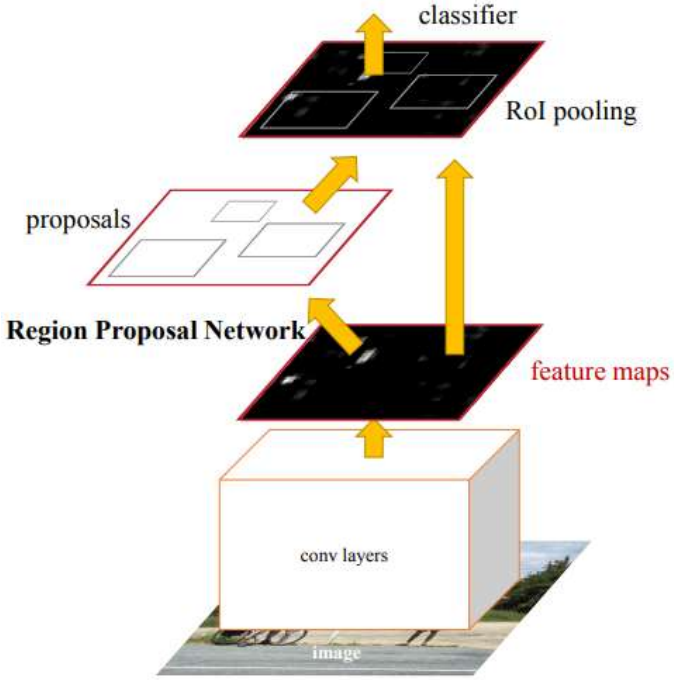
Drawbacks of class imbalance:
1) training is inefficient as most locations are easy negatives that contribute no useful learning signal
2) the easy negatives can overwhelm training and lead to degenerate models

J: Think about SVM or AdaBoost for weighting similar positive but negative samples.

- Why one-stage is less accurate? Class imbalance!
  1. Two-stage detector:
     1) Proposal (region) stage (ex: Selective Search, RPN): Filtering out most imbalance background (negative) samples (only 1-2k are left) as AdaBoost
     2) Classification stage: Fixed foreground-to-background ratio (1:3) or online hard example mining (OHEM) are used to maintain a manageable balance between foreground and background
  2. One-stage detector: Enumerating ~100k locations that densely cover spatial positions, scales, and aspect ratios (J: Anchors??) => inefficient



One-stage (Yolo)

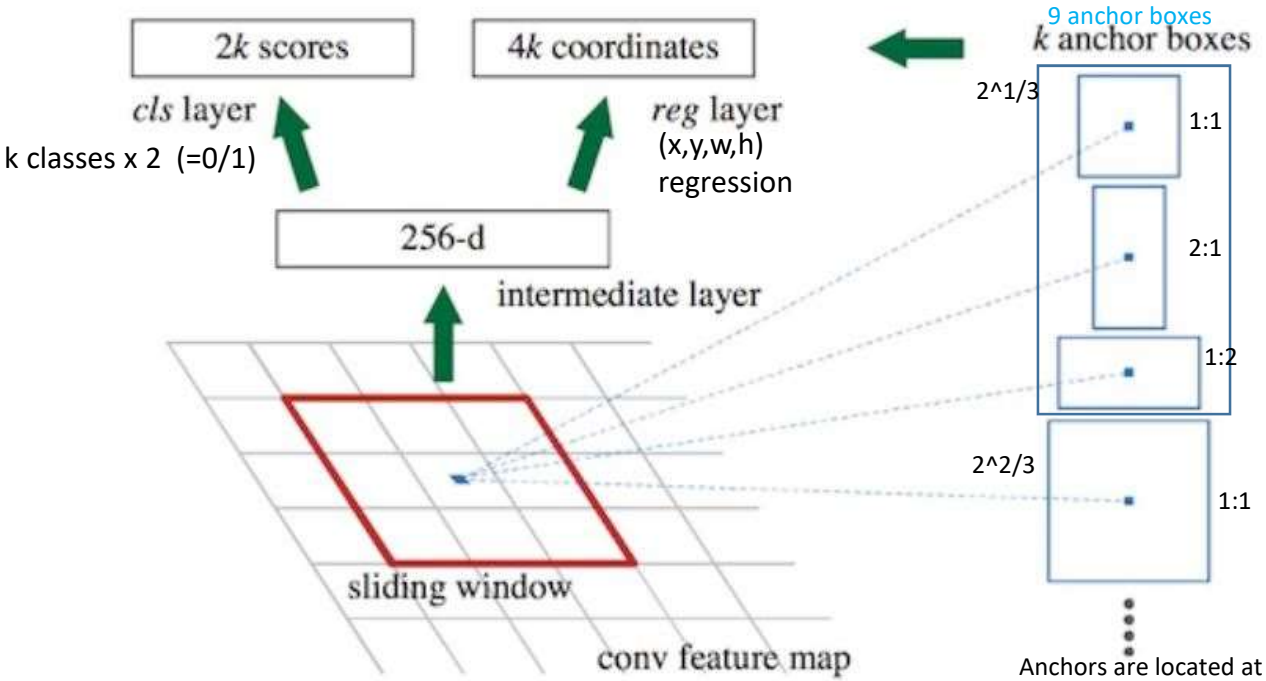


Two-stage (Faster R-CNN)

- Solution: down-weights the loss that is assigned to well-classified examples

4.3 SSD, Yolo, RetinaNet: RPN Vs. SSD

1) RPN: Region Proposal Network

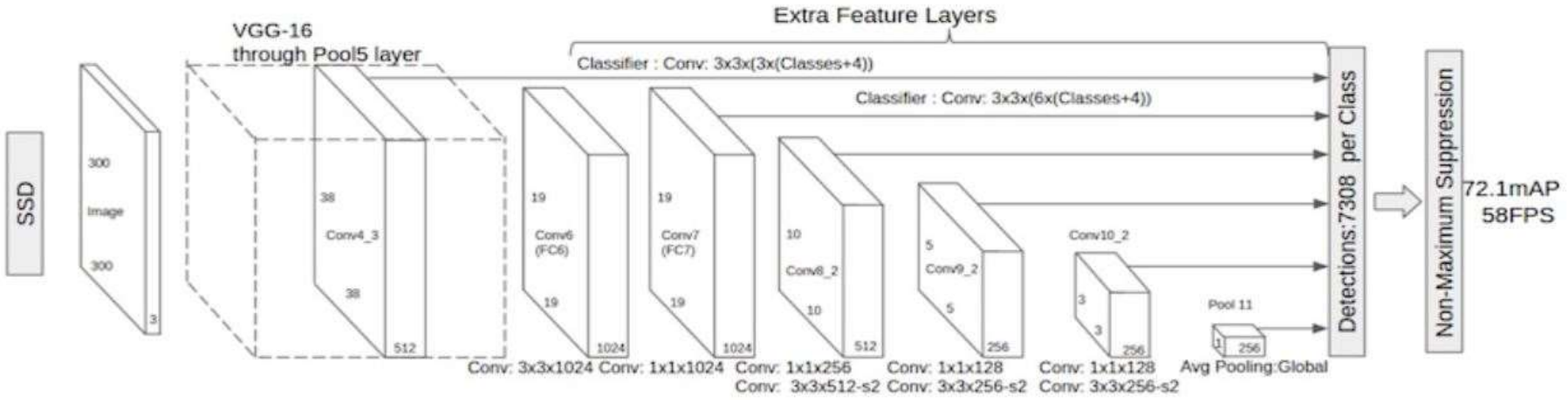


1) 3 Anchor area: $\{2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}\} * \{32^2, 64^2, 128^2, 256^2, 512^2\}$ on pyramid levels P_3 to P_7, respectively

2) 3 Anchor aspect ratio: {1:2, 1:1, 2:1}

=> In total, there are A = 9 (3 scales x 3 ratios) anchors per grid at each level l ?? \ (Yes)

2) SSD: Single Shot MultiBox Detector



4.3 RetinaNet

- RetinaNet is a single, unified network composed of:
  1. Backbone network: Feature extraction
  2. Two task-specific subnetworks BBox and Classification

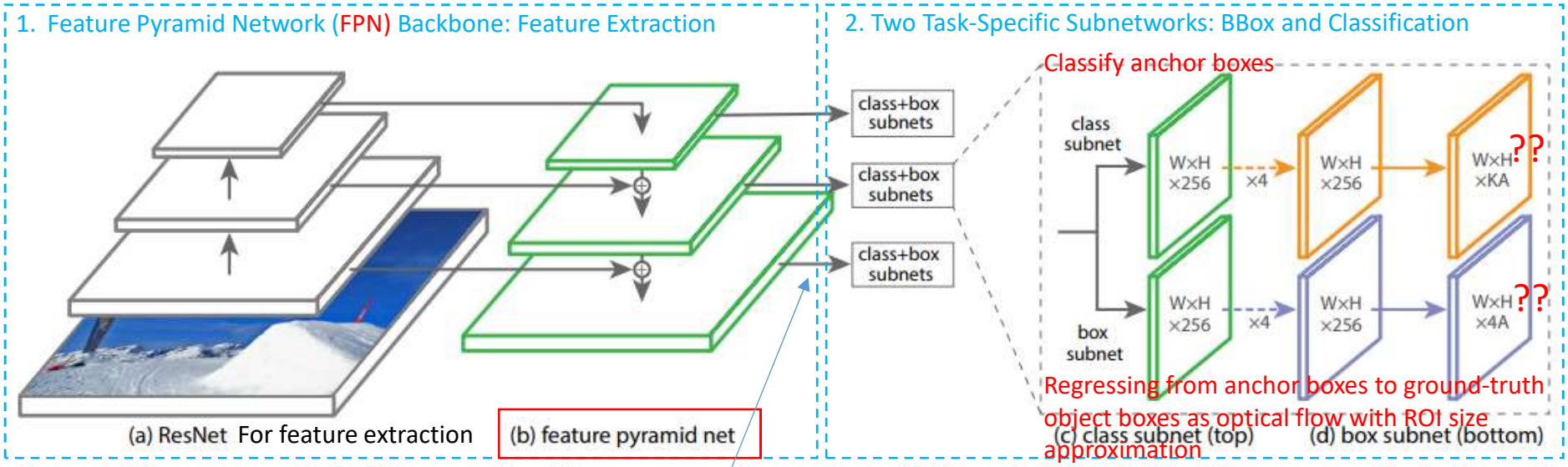


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [20] while running at faster speeds.

J: 1. This way cannot keep original detail and clear feature because features from upsampling will be blur

4.3 Multi-Task: Application –
Tool Defect Detection Using RetinaNet

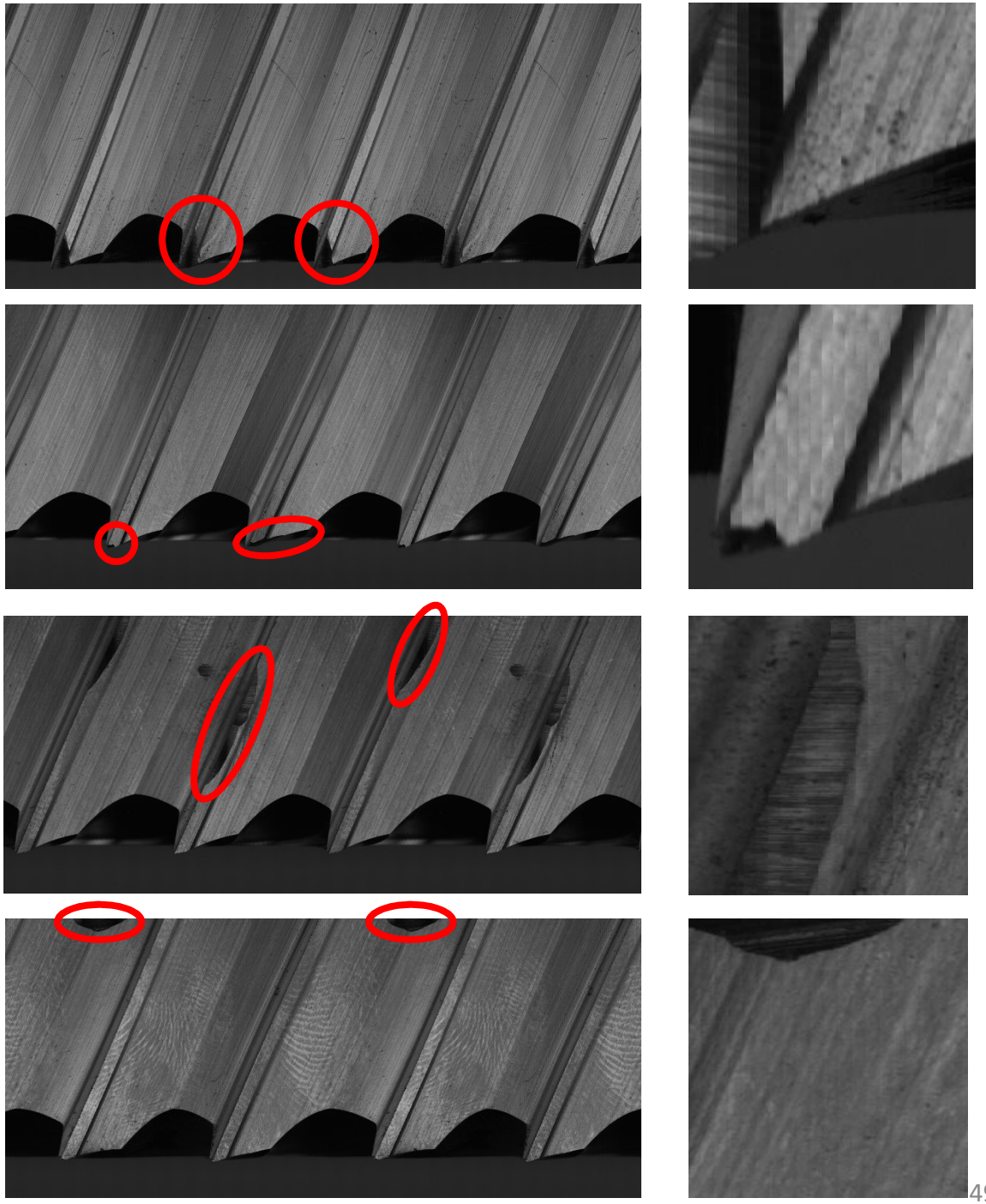
- 與中正/台大蔡孟勳教授及東台精機合作

Defect

1) Tip

2) Flank

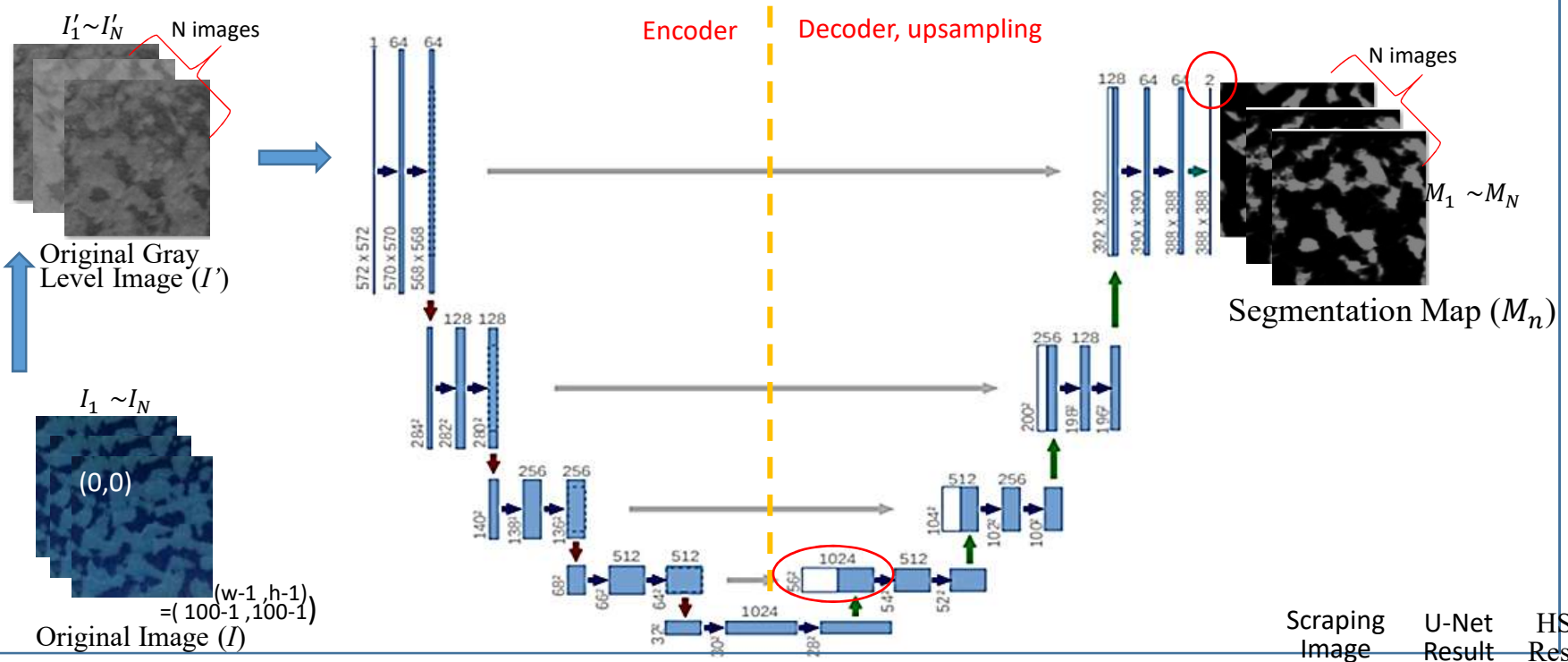
3) Plane

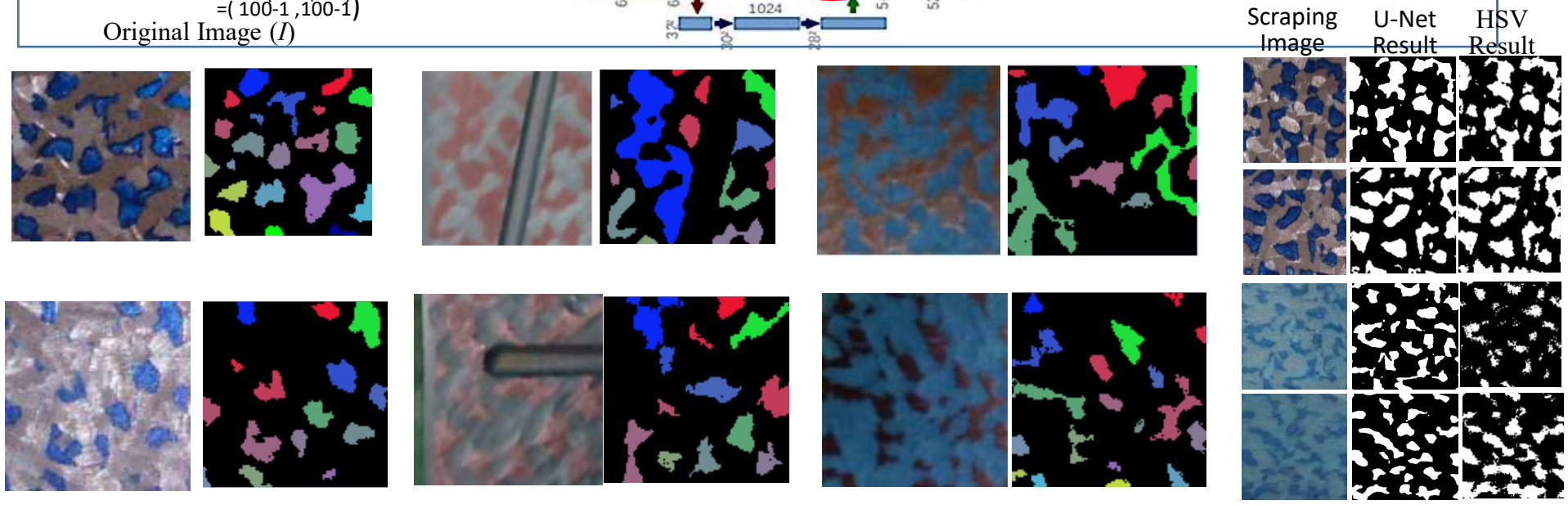


- Image Resolution: 2054 pixels * 4581 pixels
- Patch size: 224 pixels * 224 pixels

4.3 Multi-Task: Application – Segmentation Using U-Net

❑ Scrapping Segmentation Using U-Net (Vs. AutoEncoder)



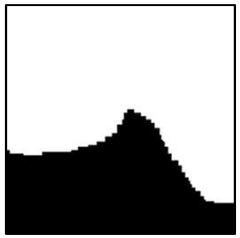
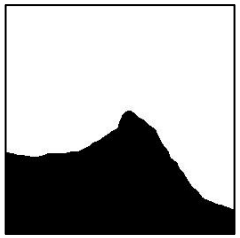
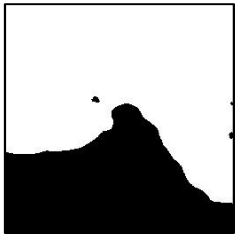


4.3 Segmentation of Oil Spill in Marine (油污) (1/4)

- 油水邊界清楚

Fg: Foreground (white)
Bg: Background (black)

map overlay = Original + Test

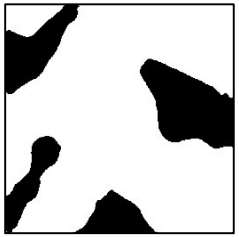
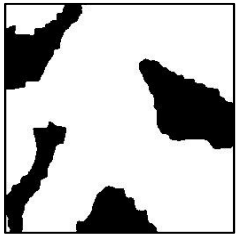
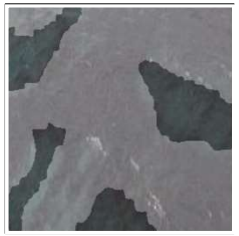
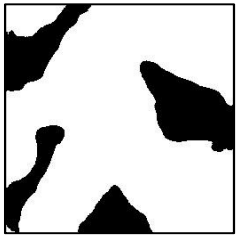
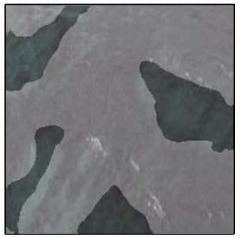
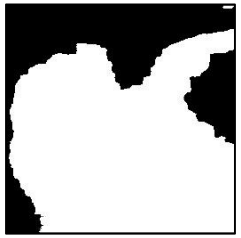
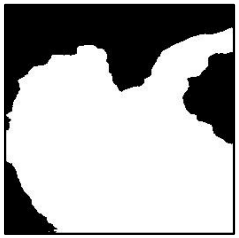
NO.	Original	GT	ReLu Test	ReLu map overlay	eLu Test	eLu map overlay
15						
17						

4.3 Segmentation of Oil Spill in Marine (油污) (2/4)

- 油水混雜

Fg: Foreground (white)
Bg: Background (black)

map overlay = Original + Test

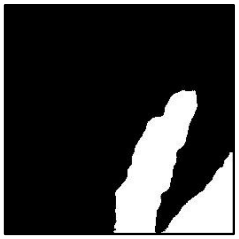
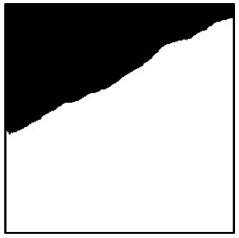
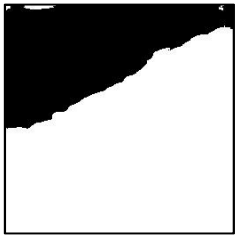
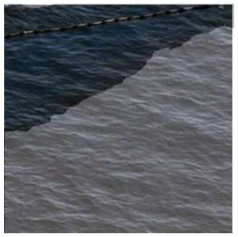
NO.	Original	GT	ReLu Test	ReLu map overlay	eLu Test	eLu map overlay
27						
91						

4.3 Segmentation of Oil Spill in Marine (油污) (3/4)

- 油水顏色相似

Fg: Foreground (white)
Bg: Background (black)

map overlay = Original + Test

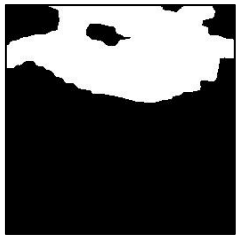
NO.	Original	GT	ReLu Test	ReLu map overlay	eLu Test	eLu map overlay
65						
46						

4.3 Segmentation of Oil Spill in Marine (油污) (4/4)

- 波紋

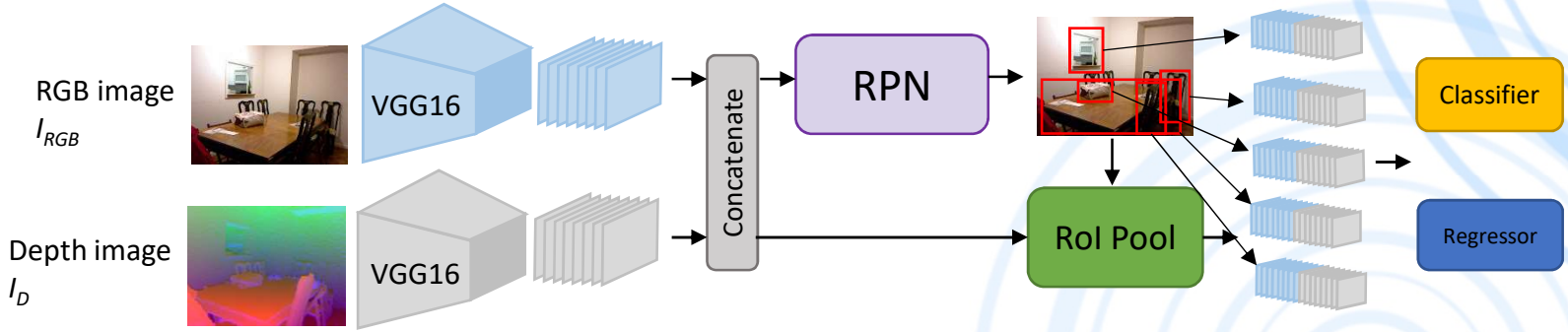
Fg: Foreground (white)
Bg: Background (black)

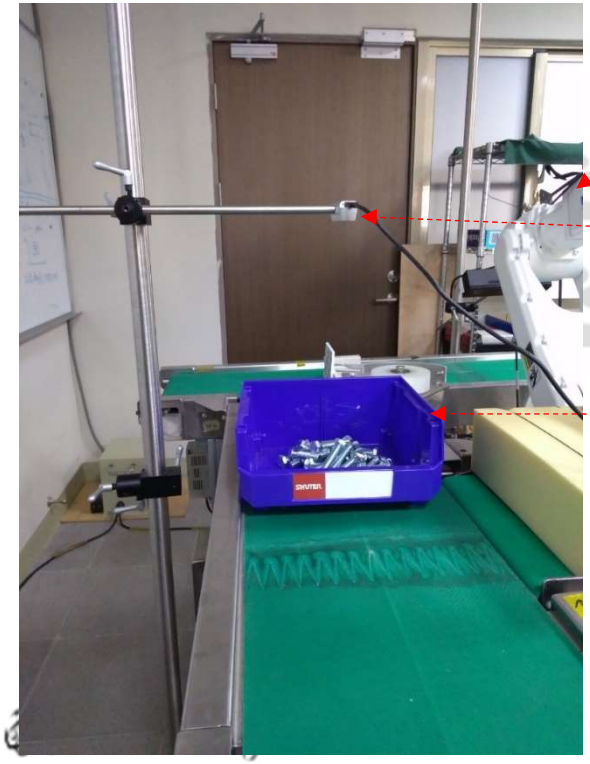
map overlay = Original + Test

NO.	Original	GT	ReLu Test	ReLu map overlay	eLu Test	eLu map overlay
96						
90						

4.4 Multi-Input: Segmentation Using VGG-16 in RGB-D (1/3)

G: 2019/11/21

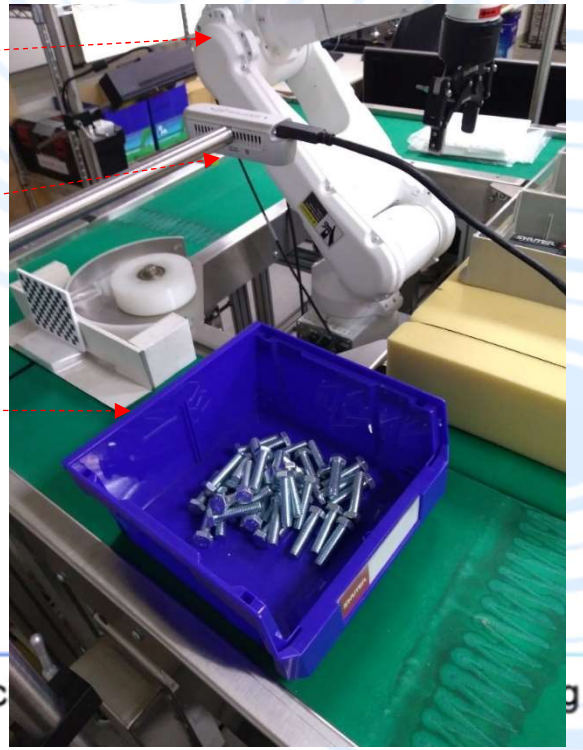




機械手臂

RGB-D攝影機

待夾堆疊物體

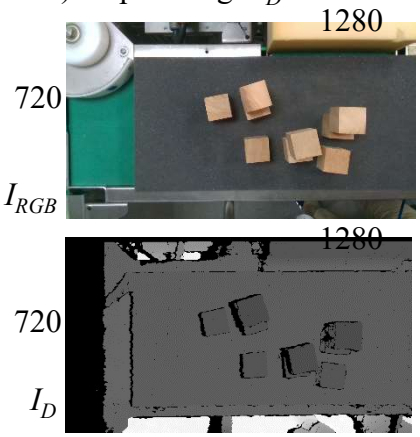


4.4 Multi-Input: Segmentation Using VGG-16 in RGB-D (2/3)

1. Objective: Detect and picking piled object with RGB-D images.
2. Global Framework:

0. Input RGB-D Image Pair

1) RGB image I_{RGB}
2) Depth image I_D



I_{RGB}, I_D

1. Detect Highest Object *B(x1, y1, x2, y2, z)* using Faster RCNN in RGB-D Space

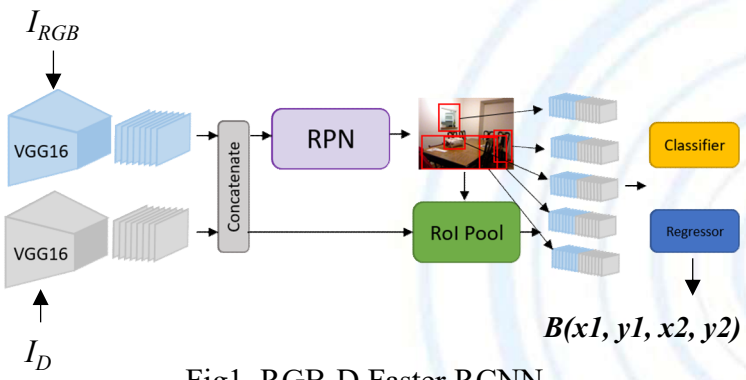
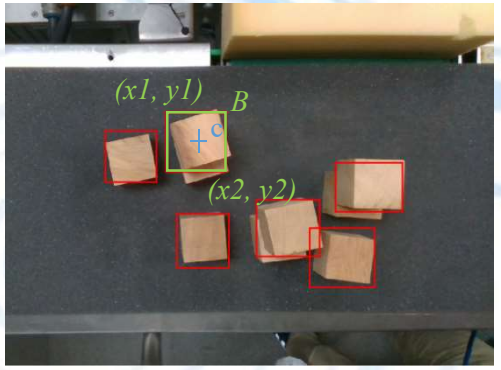


Fig1. RGB-D Faster RCNN

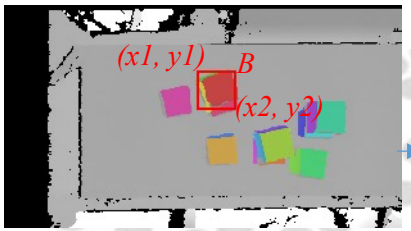


Box center: c(x, y, z)

I_D

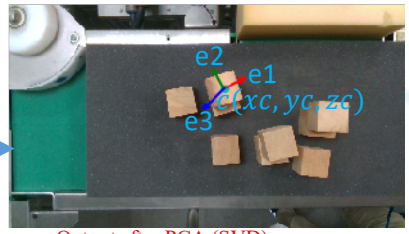
$B(x1, y1, x2, y2, z)$

2. Facet Segmentation and Normal Direction Estimation using Mask R-CNN



Segment all facets using Mask R-CNN + mask *B(x1, y1, x2, y2,z)*

Highest (or topest) Facet in *B(x1, y1, x2, y2,z)*, $I_{top}(x, y, z)$



Output after PCA (SVD):
1.1) Facet center point: $c\ (x_c, y_c, z_c)$
1.2) 夾爪張開方向: e2
1.3) Facet normal direction: e3

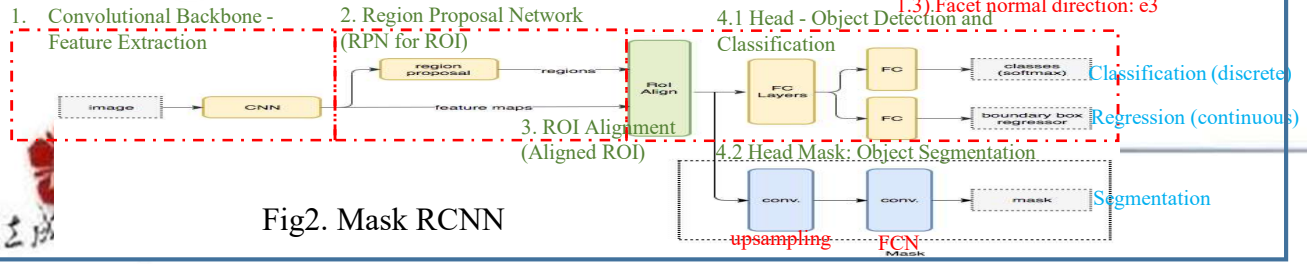
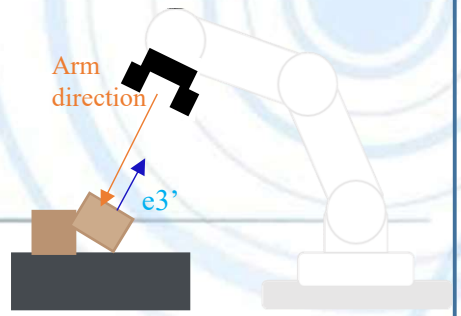


Fig2. Mask RCNN

c、e2、e3

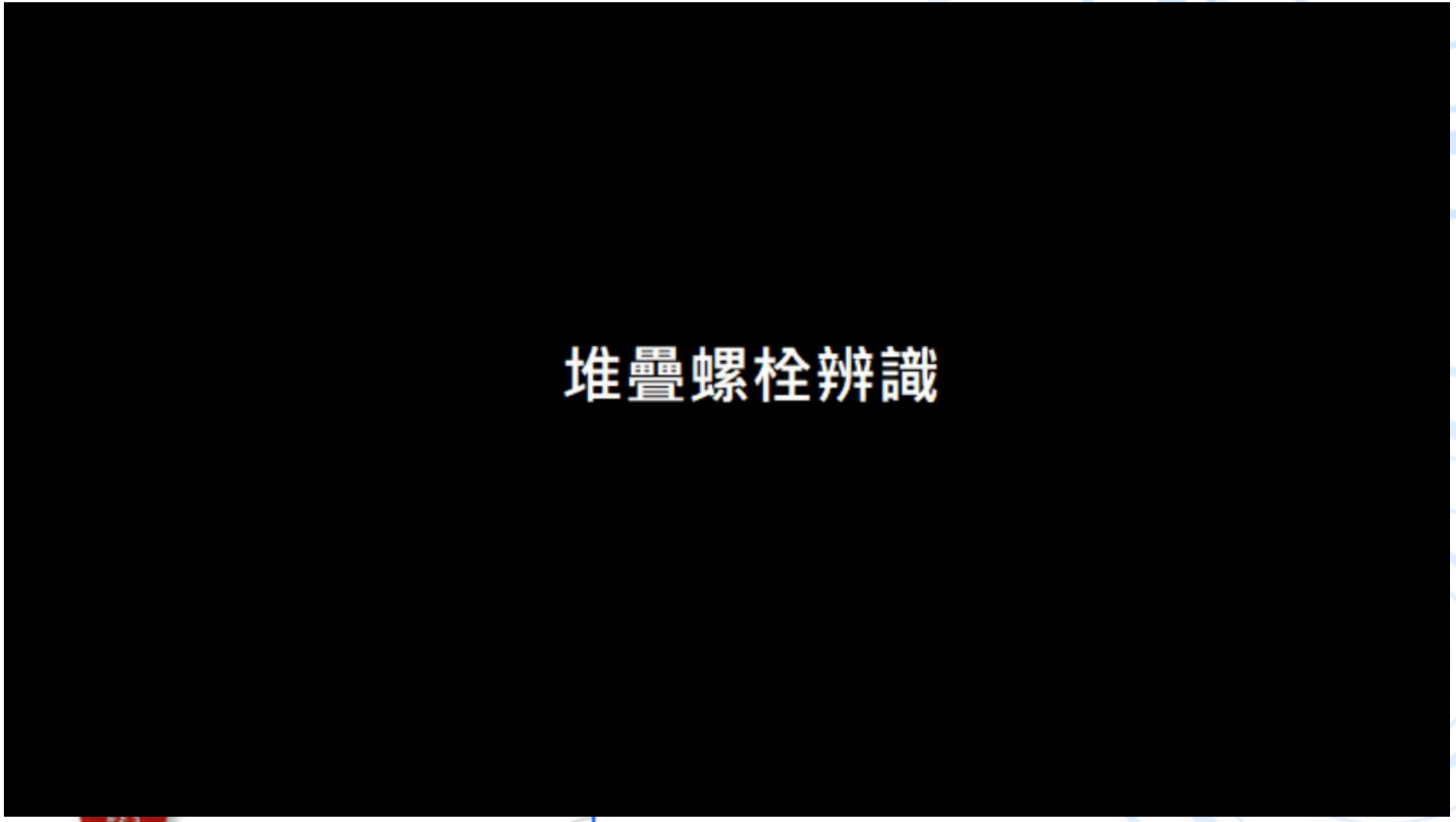
3. Picking Target Object

1) Transform depth image coordinate c, e2, e3 to robot arm coordinate c', e2', e3'
2) Grasping object along e3' direction

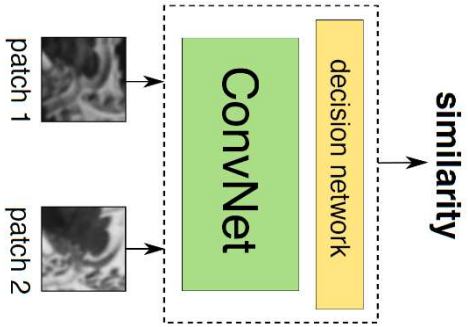


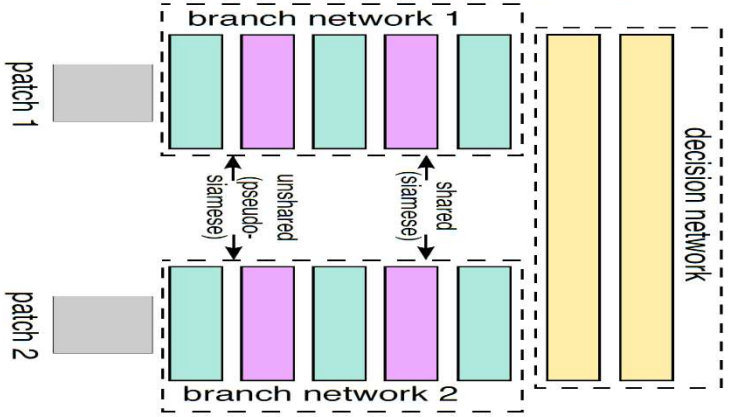
4.4 Multi-Task: Visual-Guided Robot Arm Using VGG-16 - Pile of Objects (3/3) 02:42:28

4.4 Multi-Task: Visual-Guided Robot Arm Using Deep Learning - Pile of Objects



4.5 Multi-Input: Multi-Focus Image Fusion Using Siamese



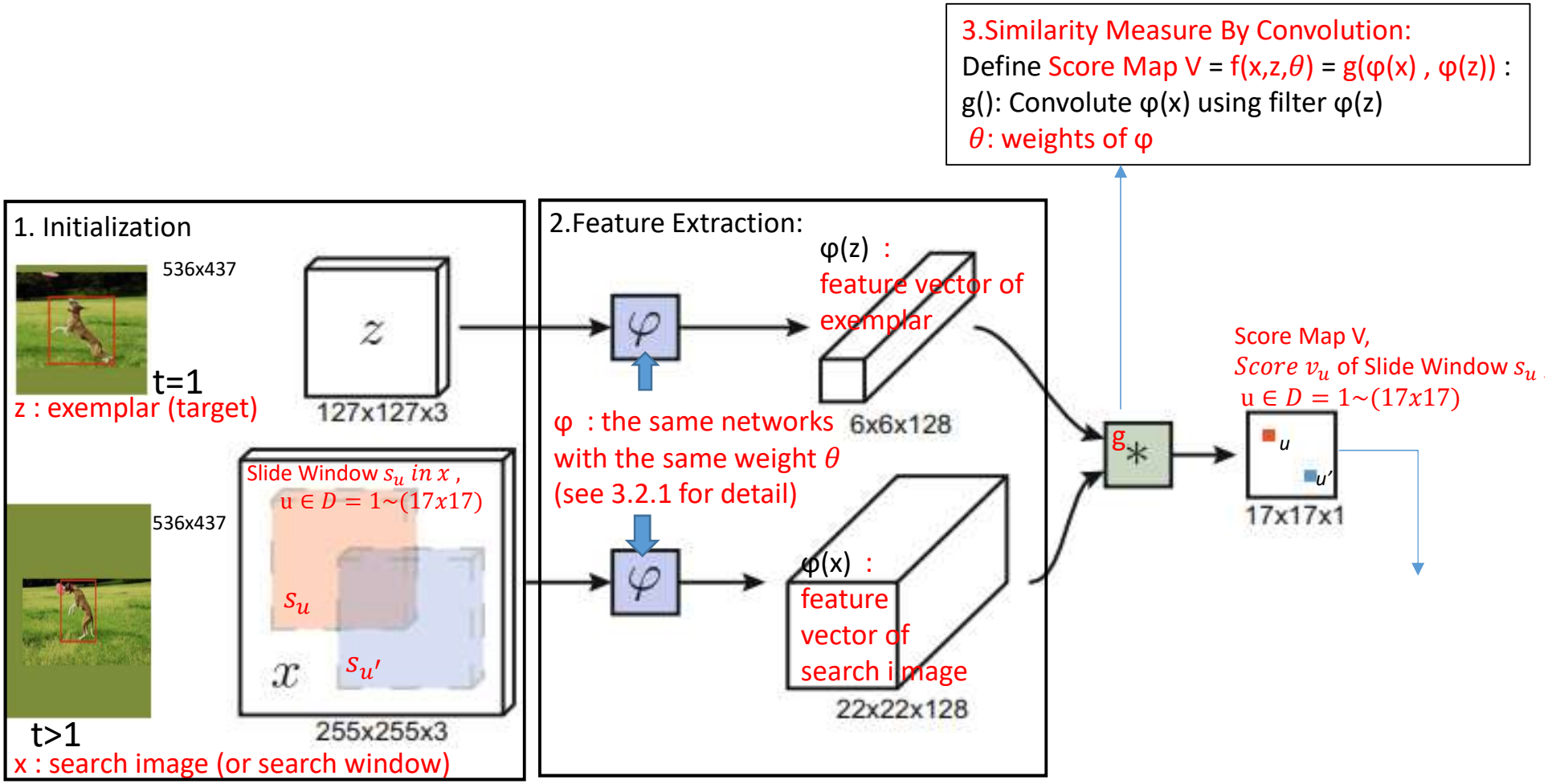


A: FG Clear, BG Blur

B: FG Blur, BG Clear

A+B: Fusion FG Clear, BG Clear

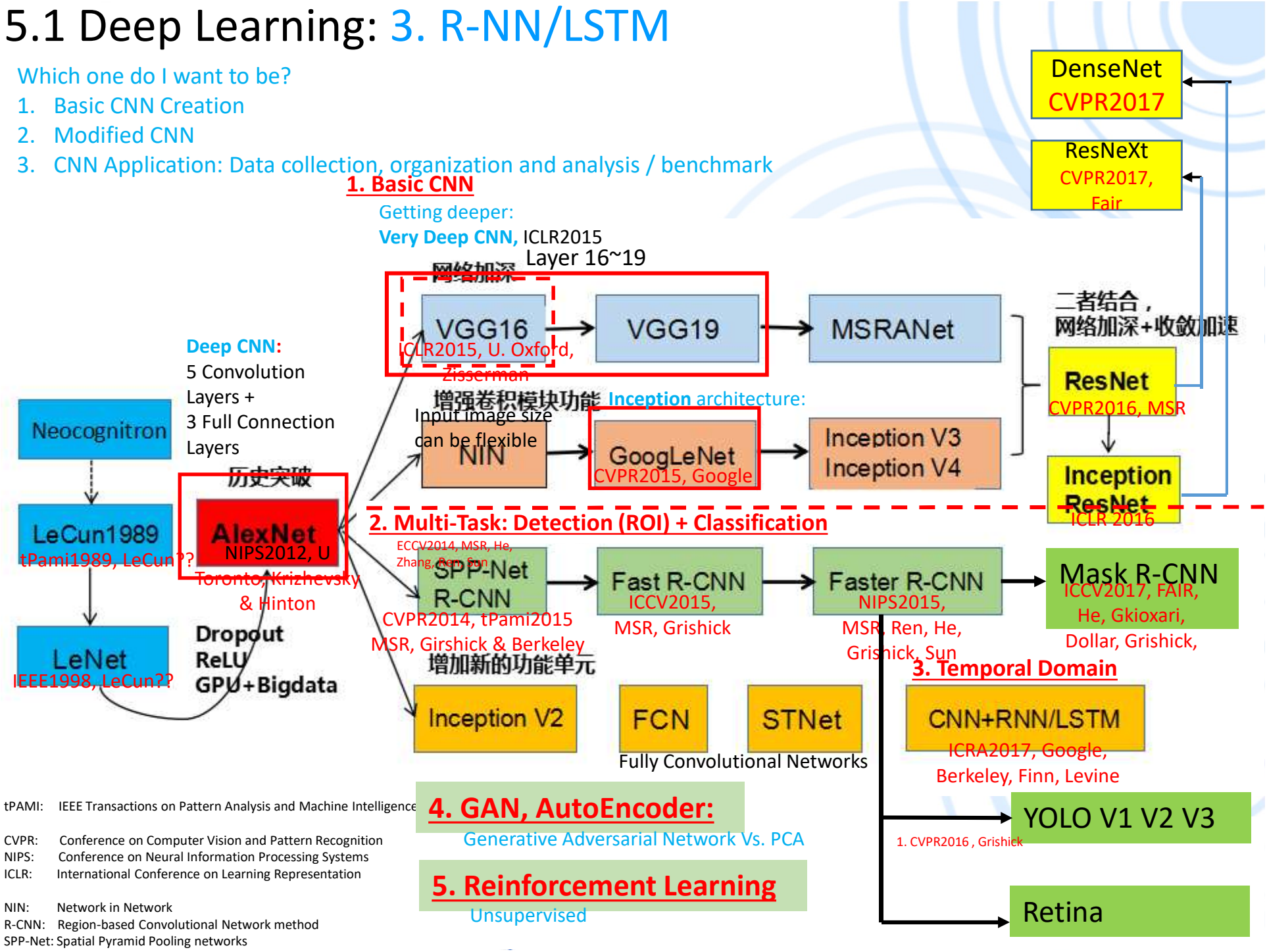
4.5 Multi-Input: Tracking / Stereo Using Siamese



5.1 Deep Learning: 3. R-NN/LSTM

Which one do I want to be?

1. Basic CNN Creation
2. Modified CNN
3. CNN Application: Data collection, organization and analysis / benchmark

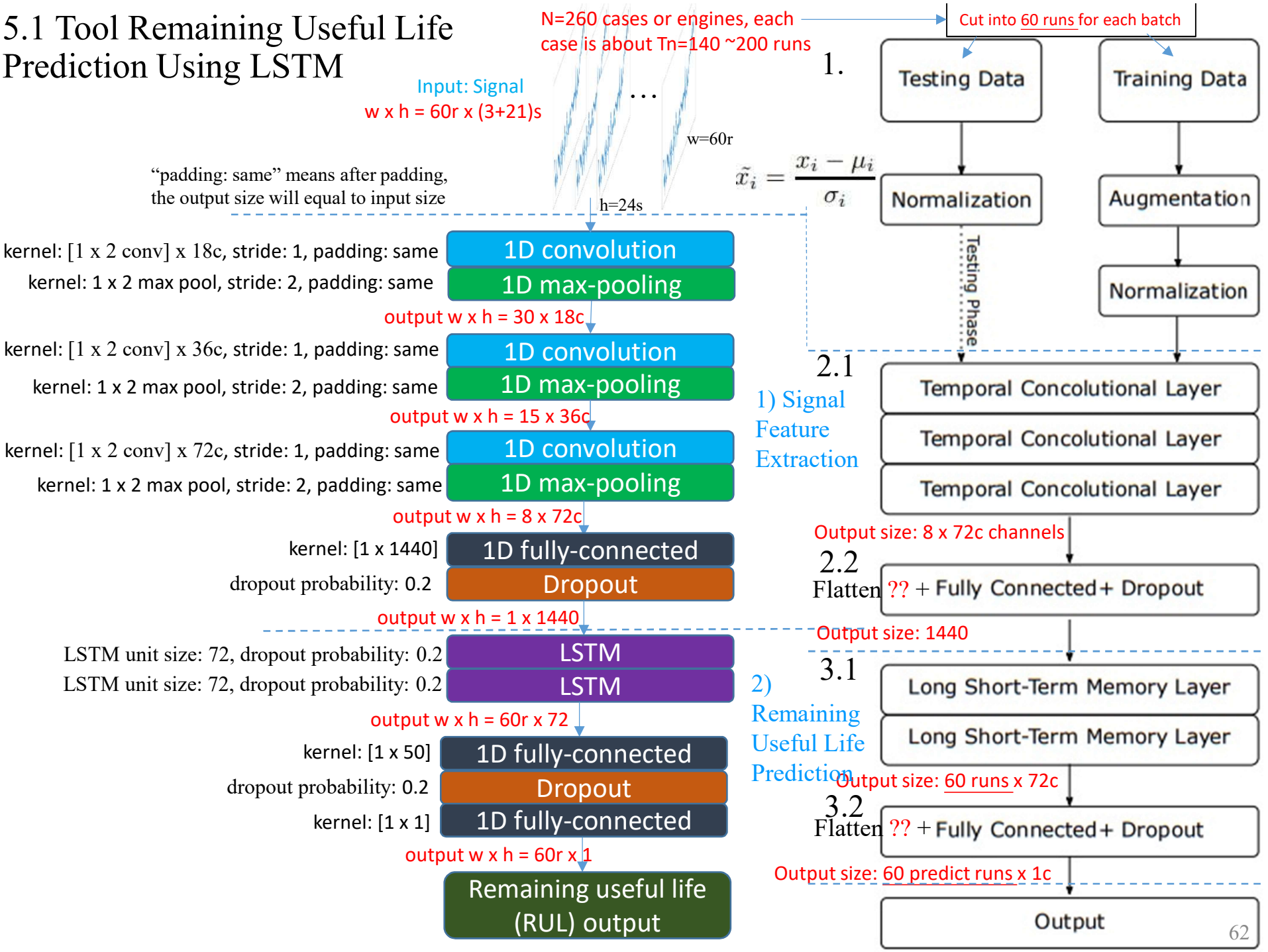


tPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence

CVPR: Conference on Computer Vision and Pattern Recognition
NIPS: Conference on Neural Information Processing Systems
ICLR: International Conference on Learning Representation

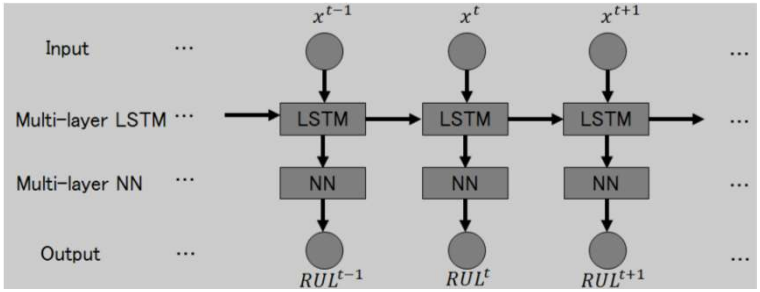
NIN: Network in Network
R-CNN: Region-based Convolutional Network method
SPP-Net: Spatial Pyramid Pooling networks

5.1 Tool Remaining Useful Life Prediction Using LSTM



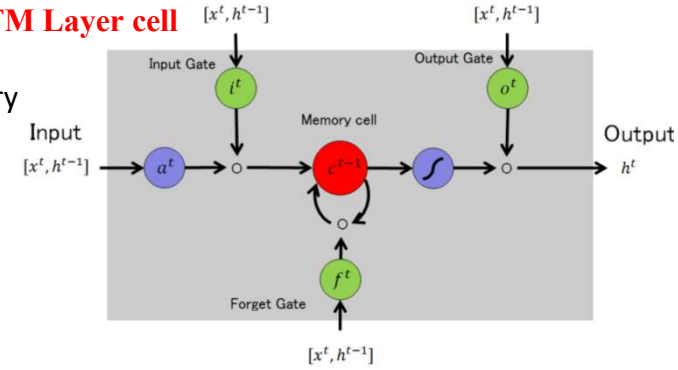
5.1 LSTM: Tool Life Signal Prediction Using LSTM 02:12.31

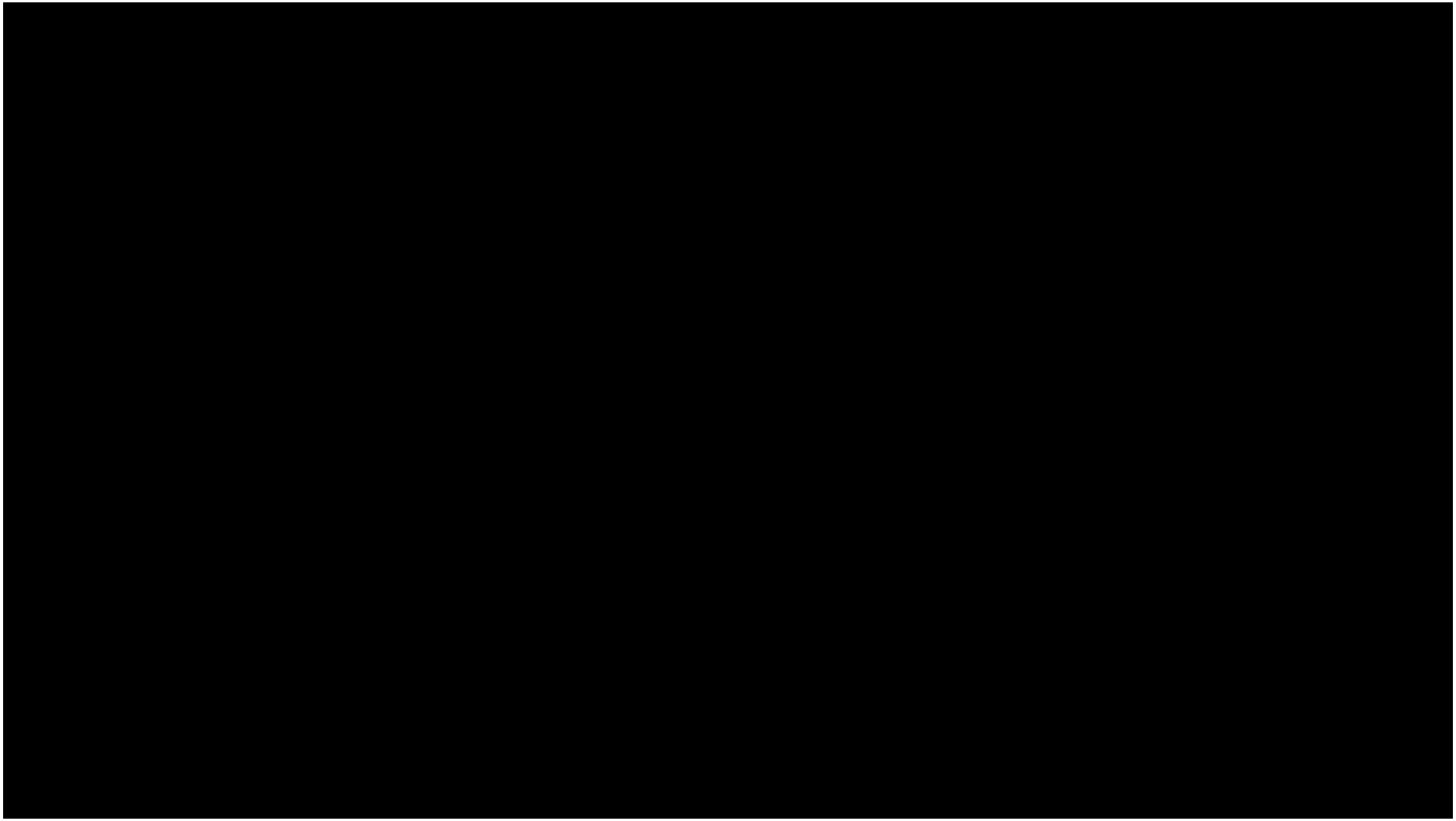
LSTM Model -x_i : i-th sensor data ($x_i = [x_i^1, \ldots, x_i^t, \ldots, x_i^{Tn}]$



LSTM:
Long Short-Term Memory

LSTM Layer cell

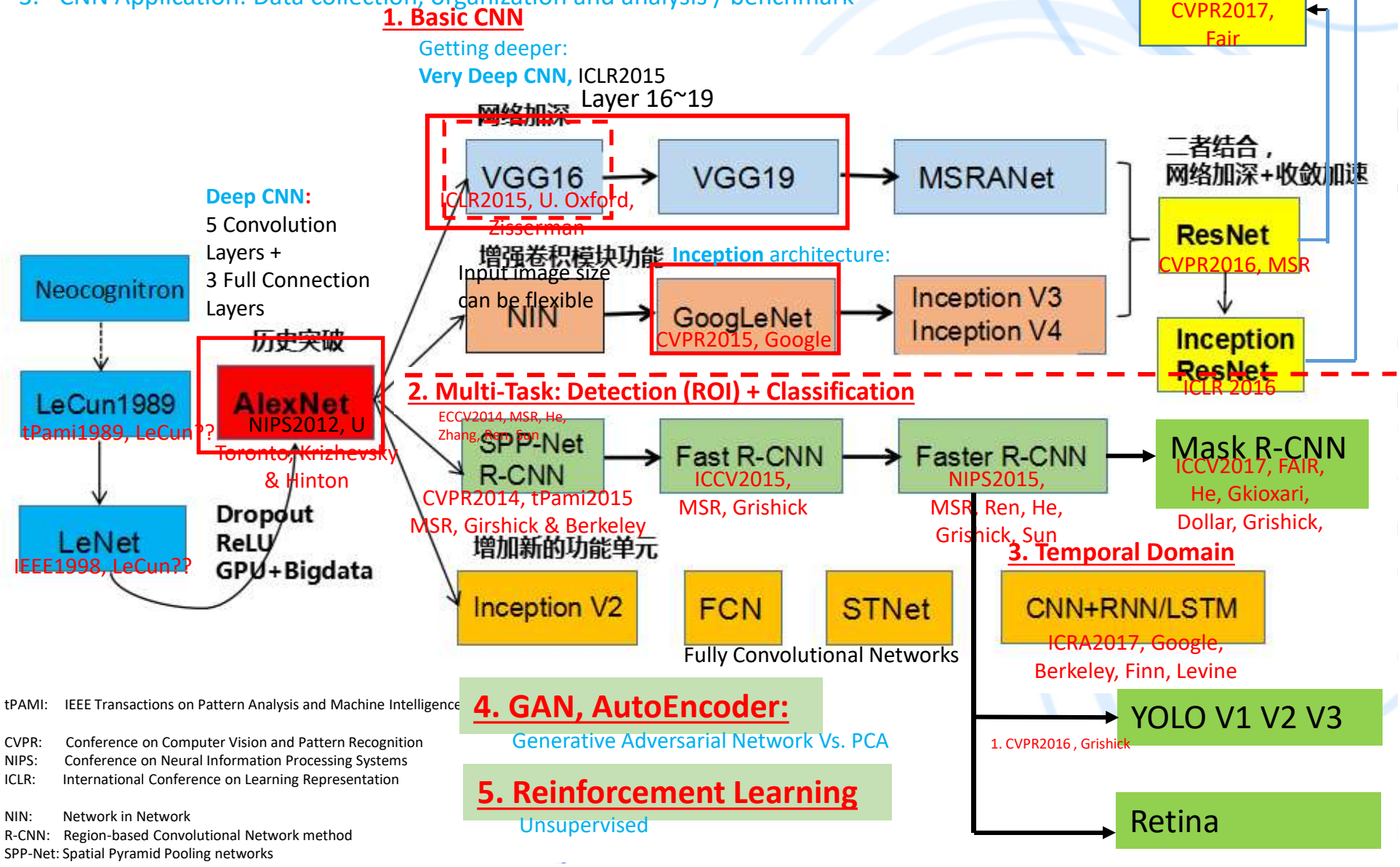




6.0 Deep Learning: 4. GAN, AutoEncoder

Which one do I want to be?

1. Basic CNN Creation
2. Modified CNN
3. CNN Application: Data collection, organization and analysis / benchmark



tPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence

CVPR: Conference on Computer Vision and Pattern Recognition
NIPS: Conference on Neural Information Processing Systems
ICLR: International Conference on Learning Representation

NIN: Network in Network
R-CNN: Region-based Convolutional Network method
SPP-Net: Spatial Pyramid Pooling networks

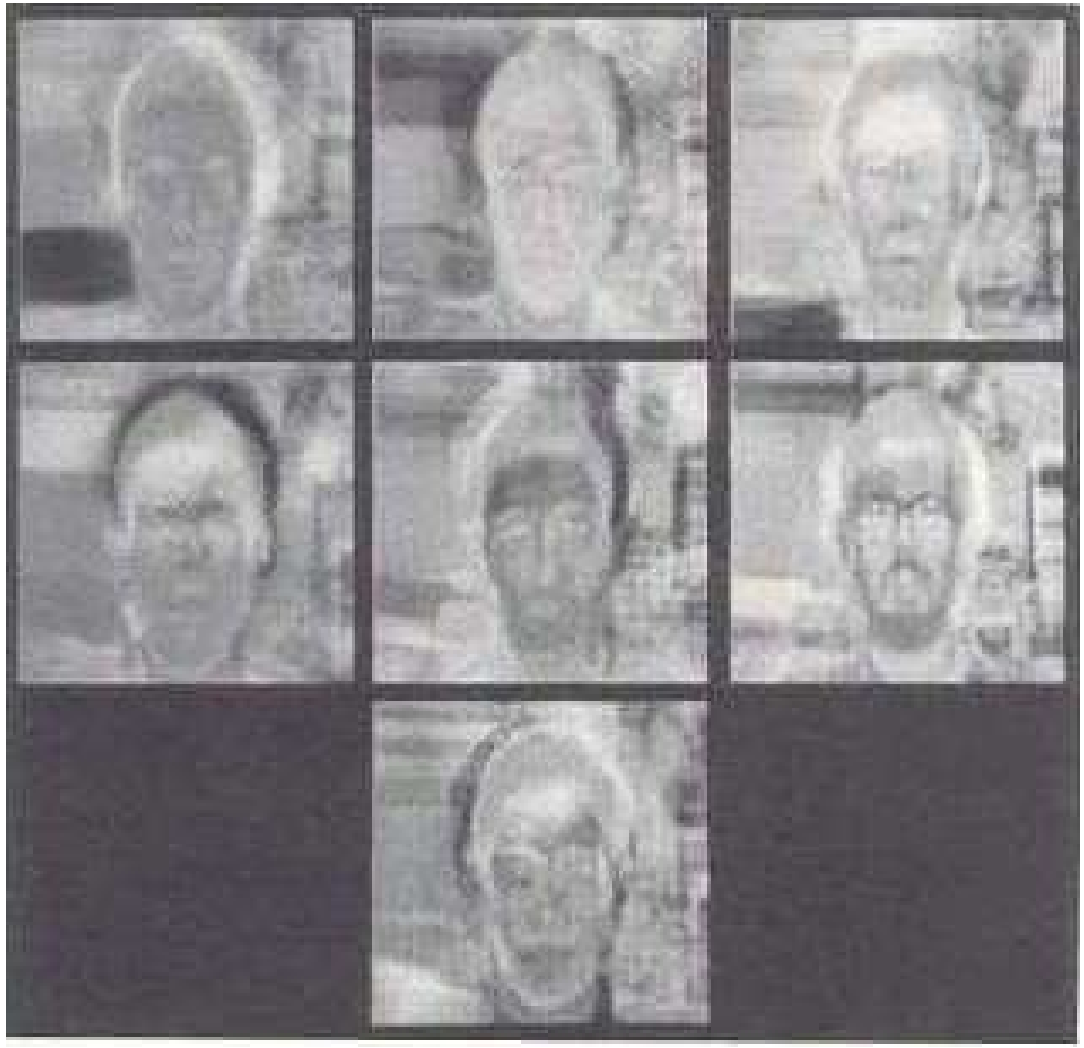
6.0 GAN (Generative Adversarial Networks) Vs. PCA：PCA（生成對抗網路）

❑ PCA

- Face Recognition: Input/output gray values

6.0 GAN (Generative Adversarial Networks) Vs. PCA: PCA

- PCA – Eigenspace encoding: Eigenvectors



6.0 GAN (Generative Adversarial Networks) Vs. PCA: PCA

- PCA – Eigenspace decoding: Reconstruction

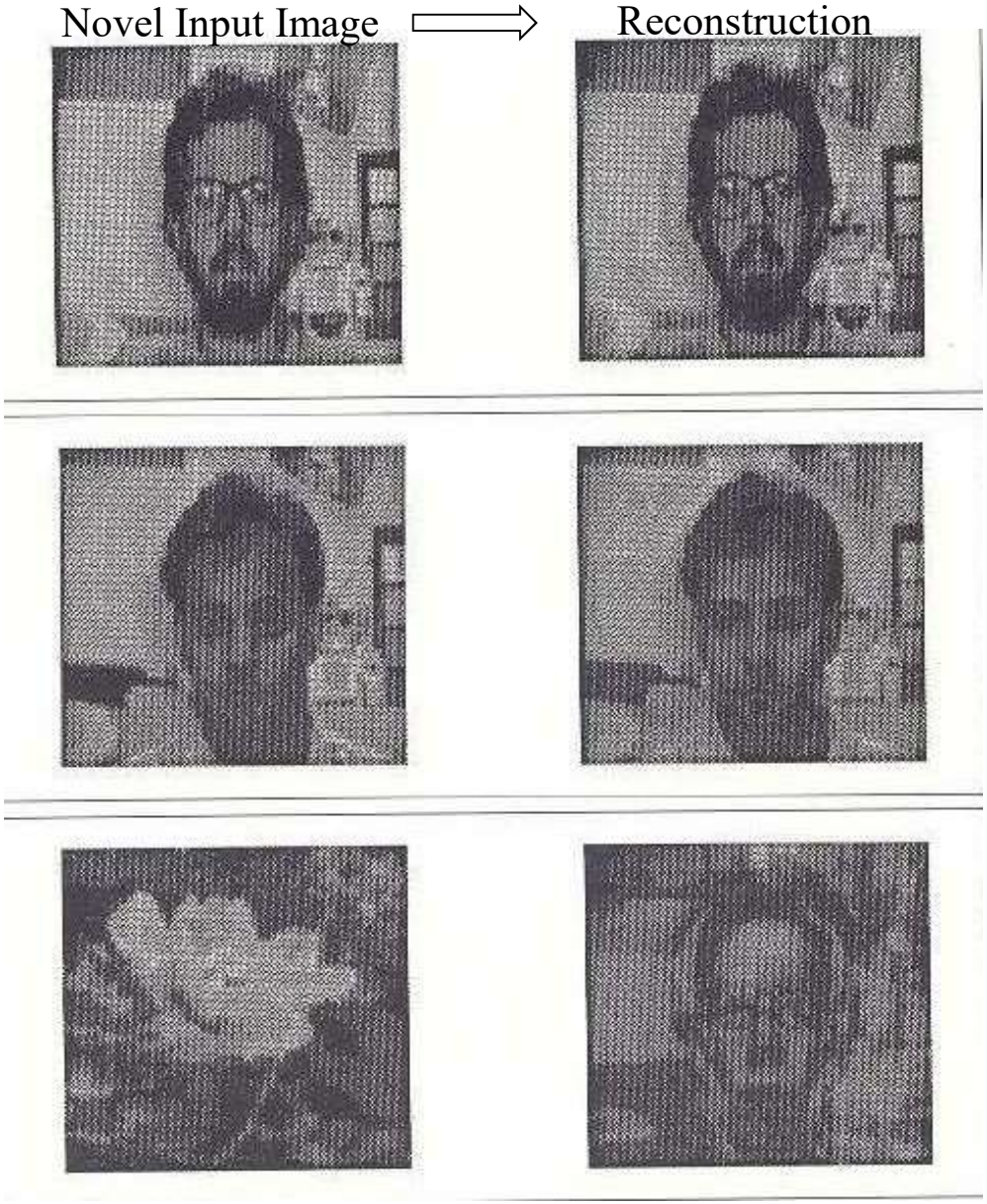
- Decoding: Reconstruction but Lost details

- Decoding: Recovery

PCA

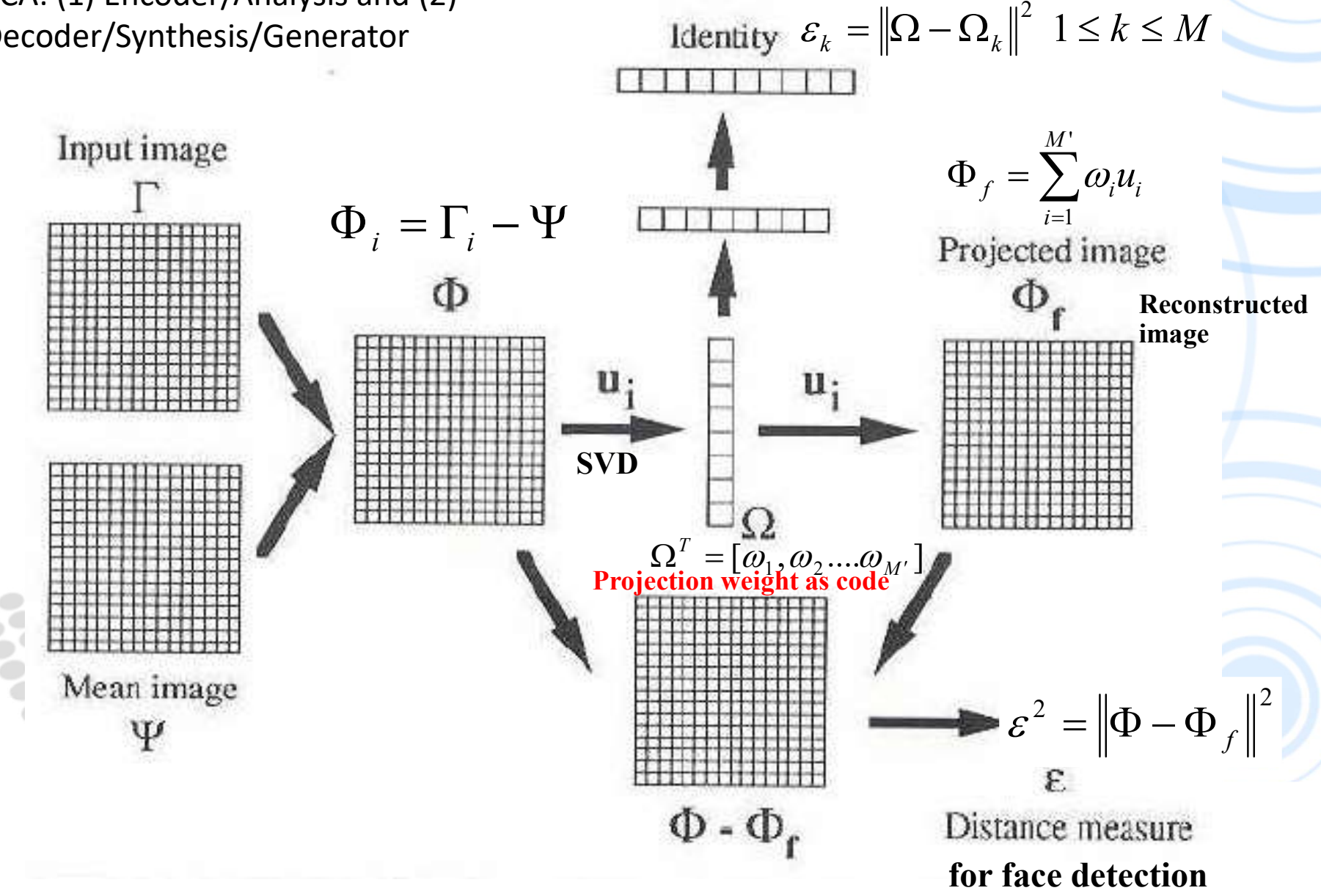
6.0 GAN (Generative Adversarial Networks) Vs. PCA: PCA

- PCA – Eigenspace decoding: Reconstruction



6.0 GAN (Generative Adversarial Networks) Vs. PCA: PCA for face recognition

- PCA: (1) Encoder/Analysis and (2) Decoder/Synthesis/Generator



6.1 GAN Vs. PCA: Auto-encoder and Decoder

- GAN: (1) Encoder/Analysis and (2) Decoder/Synthesis/Generator

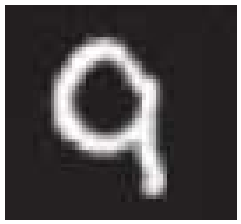
High dimension to low dimension

J: Trainable Encoder/Decoder. A set of basis functions as
1) PCA (linear) eigenvector,
2) Filter: Edge filter, wavelets (cos, sin)…
3) NN (non-linear)

J: Trainable Encoder/Decoder: As PCA eigenvectors or filter coefficients

J: Code: This low dimensional vector is as projection weights or feature extraction result

(1) Analysis: Computer Vision



28 X 28 = 784

NN Encoder

Low dimension

code

Compact representation of the input object

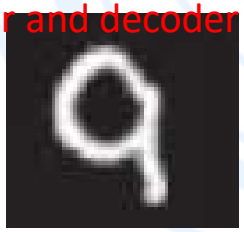
GAN: NN para. Learn together but different Eigenvectors. That is, Encoder coeff. And Decoder coeff are different

PCA: Encoder and decoder have the same para.

(2) Synthesis / Generator: Computer Graphics

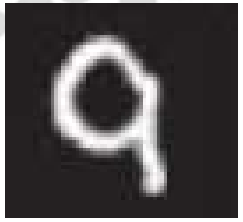
code

NN Decoder



Can reconstruct the original object

Trainable



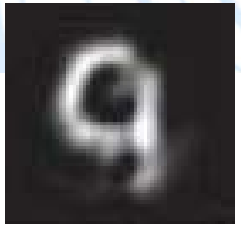
NN Encoder

(1) Analysis

c

NN Decoder

(2) Synthesis / Generator



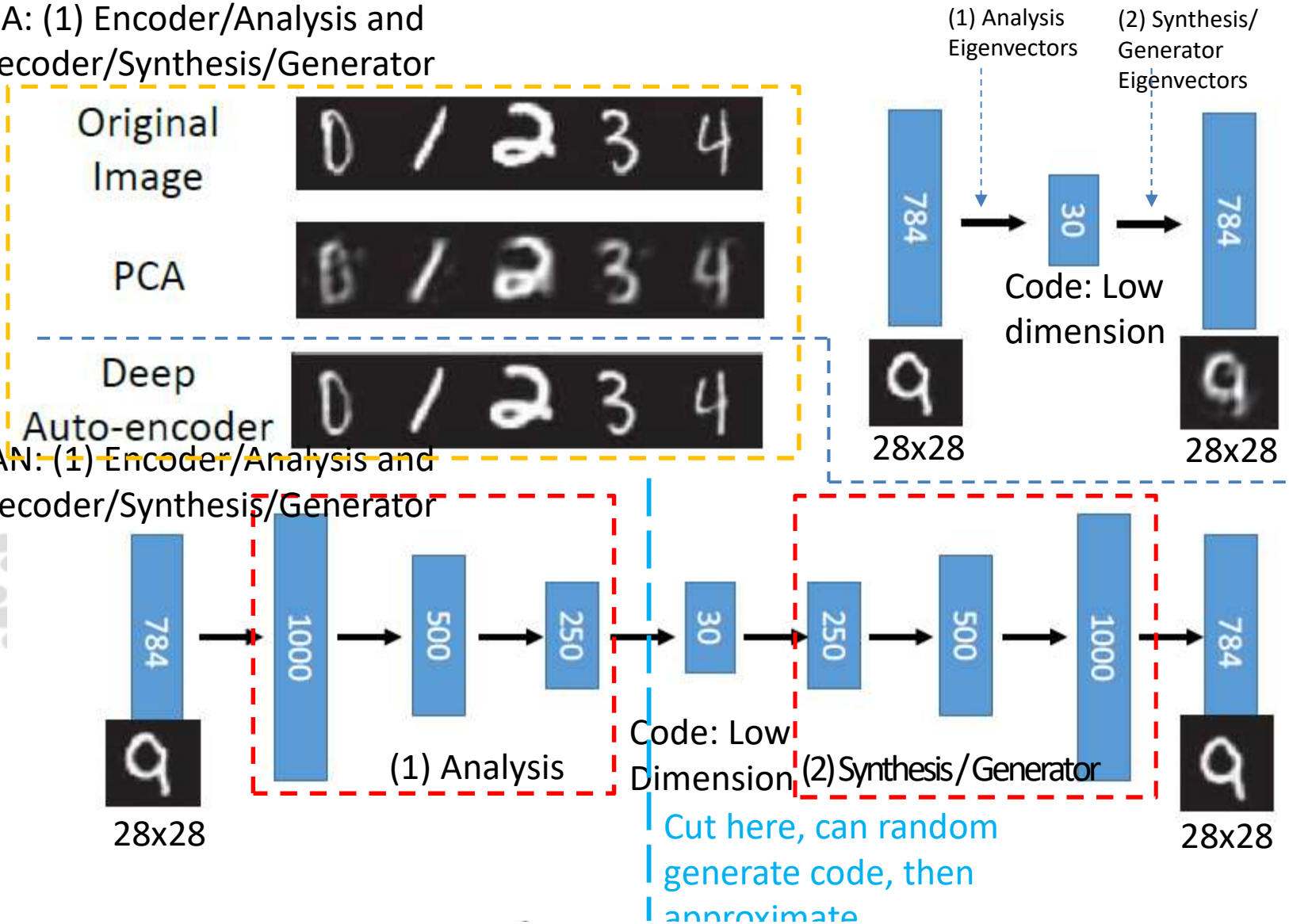
Need to keep processing / improving….

6.1 GAN (Generative Adversarial Networks) Vs. PCA

Jenny: difference of the result of PCA and Deep auto-encoder

Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

2) PCA: (1) Encoder/Analysis and
(2) Decoder/Synthesis/Generator



6.1 GAN (Generative Adversarial Networks) Vs. PCA

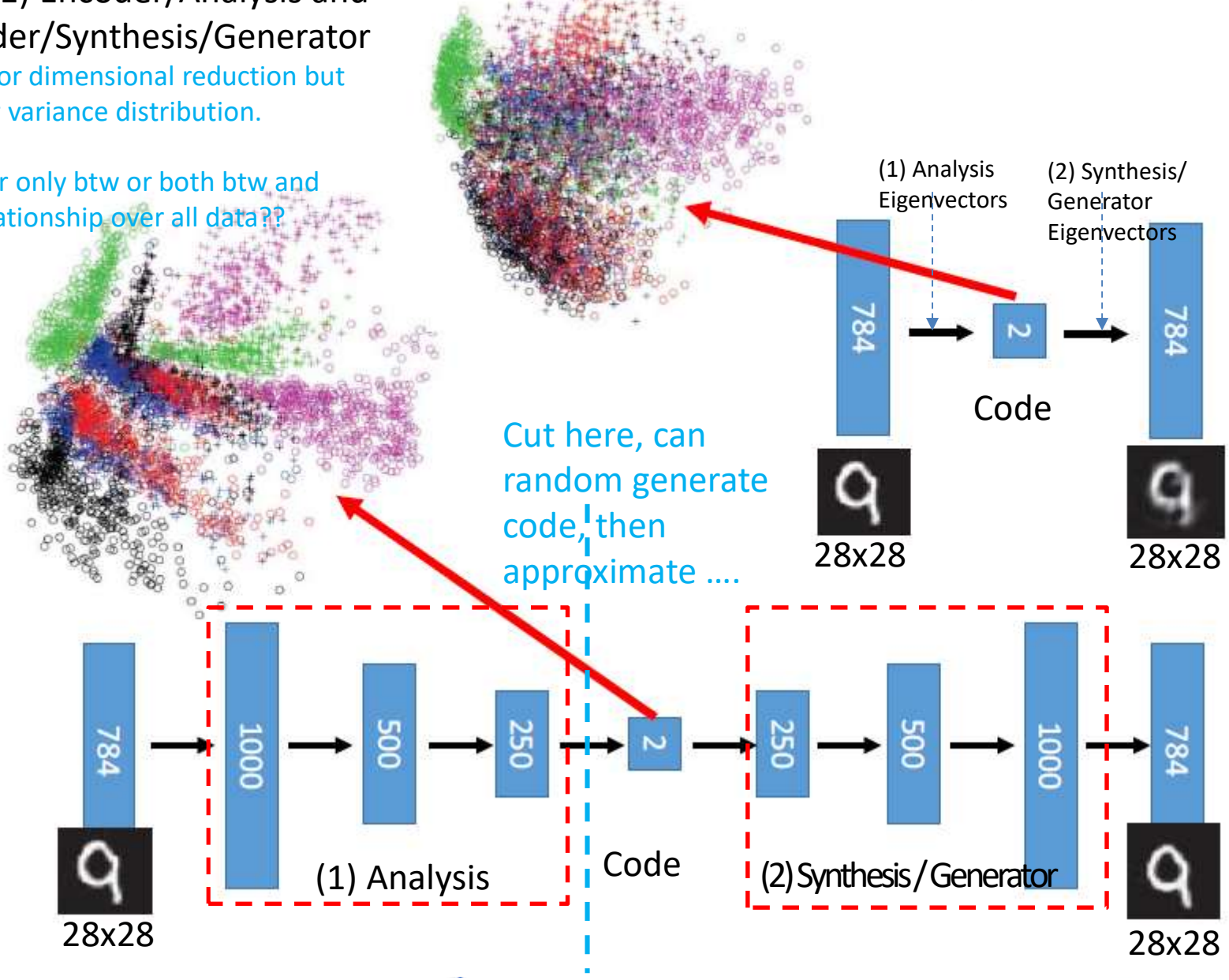
Jenny: difference of the result of PCA and Deep auto-encoder

1) GAN: (1) Encoder/Analysis and (2) Decoder/Synthesis/Generator

Not only for dimensional reduction but has better variance distribution.

J: Consider only btw or both btw and within relationship over all data??

2) PCA: (1) Encoder/Analysis and (2) Decoder/Synthesis/Generator



6.1 Basic Idea of GAN: Adversarial - Basic Idea of GAN: Adversarial - Relationship btw Generator and Discriminator

- GAN: 2 Models. Example

2) Generator model as a butterfly: changing to fit the brown-color leaf background in order not to be found.

3) Discriminator model as a bird: who wants to eat butterfly. Getting smarter to find Generator model target.

Improving better and better till cannot find the Generator Model butterfly.

(2) Generator

Becomes brown

veins

Butterflies are not brown, easy to be found

Butterflies becomes brown color as background but don't have veins, Still easy to be found

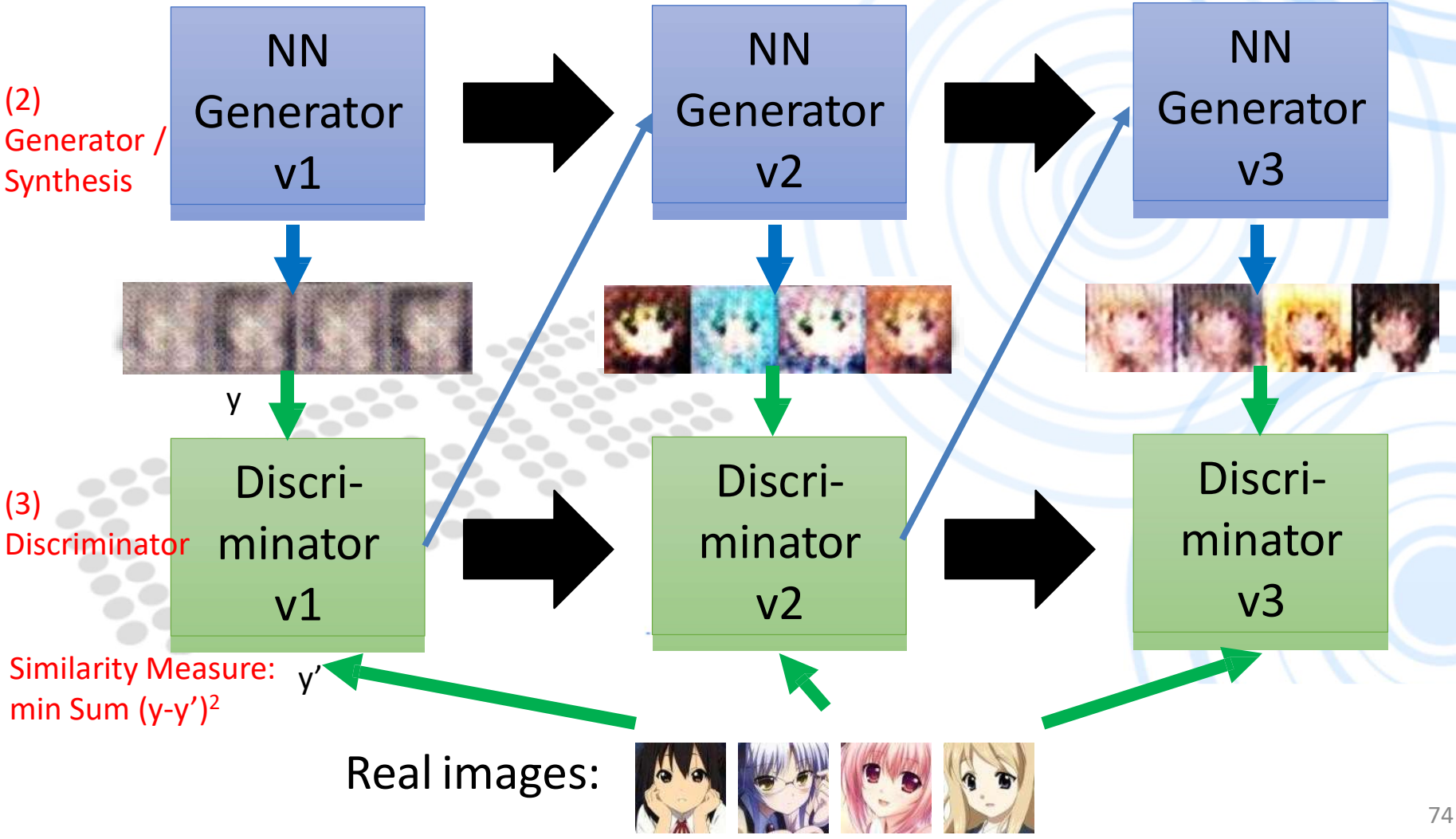
Where is the butterfly? Cannot be found

(3) Discriminator

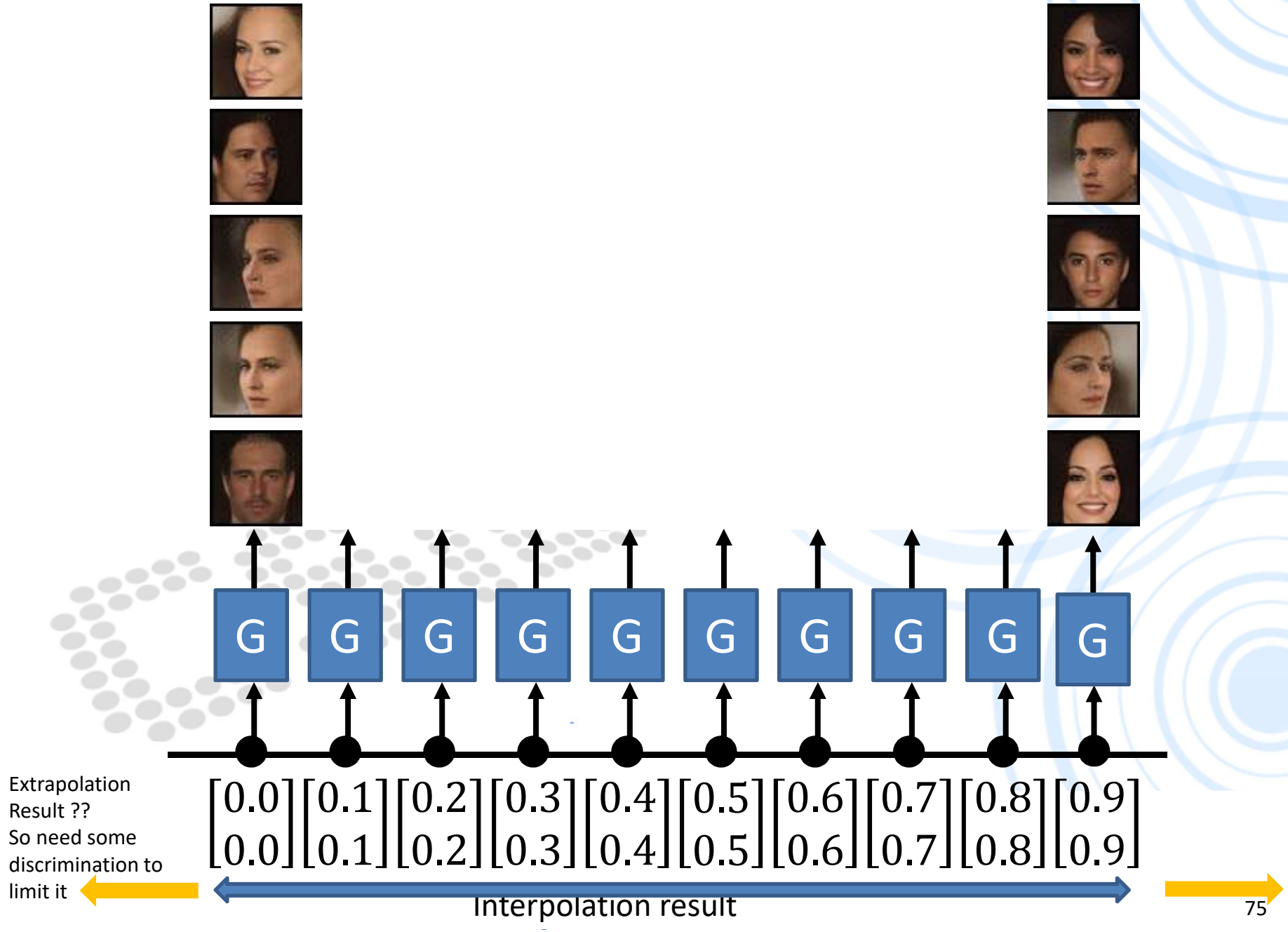
6.1 Basic Idea of GAN: Adversarial

This is where the term "***adversarial***" comes from.

You can explain the process in different ways…….



6.1 GAN: Interpolation Vs. Extrapolation



6.2 GAN: Augmented Data

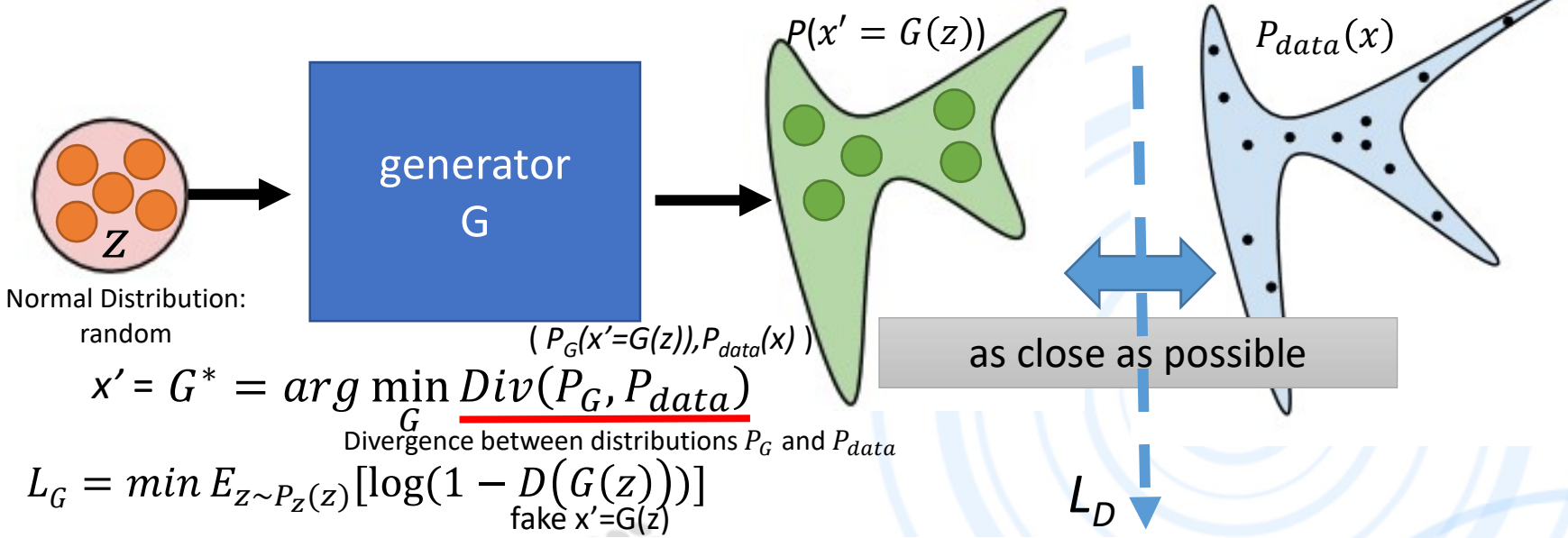
GAN: Generative Adversarial Networks

x: A real data image

x': Fake, generated data

$P_{data}(x)$: A high-dimensional distribution vector for real data image x

$P_G(x')$: In high dim. distribution space for generated data x'



($P_G(x'=G(z)), P_{data}(x)$)

$$x' = G^* = arg \min_G Div(P_G, P_{data})$$

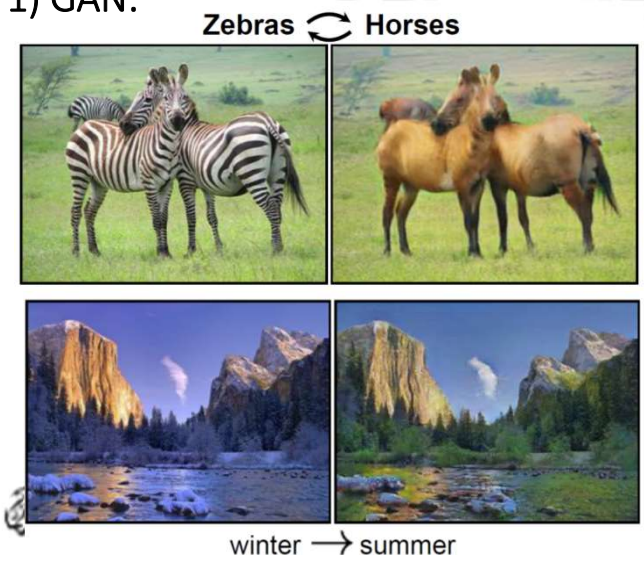
Divergence between distributions P_G and P_{data}

$$L_G = min\, E_{z\sim P_z(z)}[\log(1 - D(G(z)))]$$

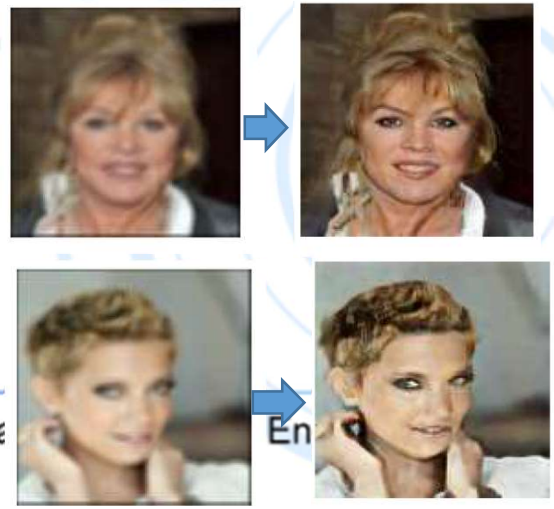
fake x'=G(z)

$$L_D = \max E_{x\sim P_{data}(x)}[logD(x)] + E_{z\sim P_z(z)}[\log(1 - D(G(z)))]$$

1) GAN:



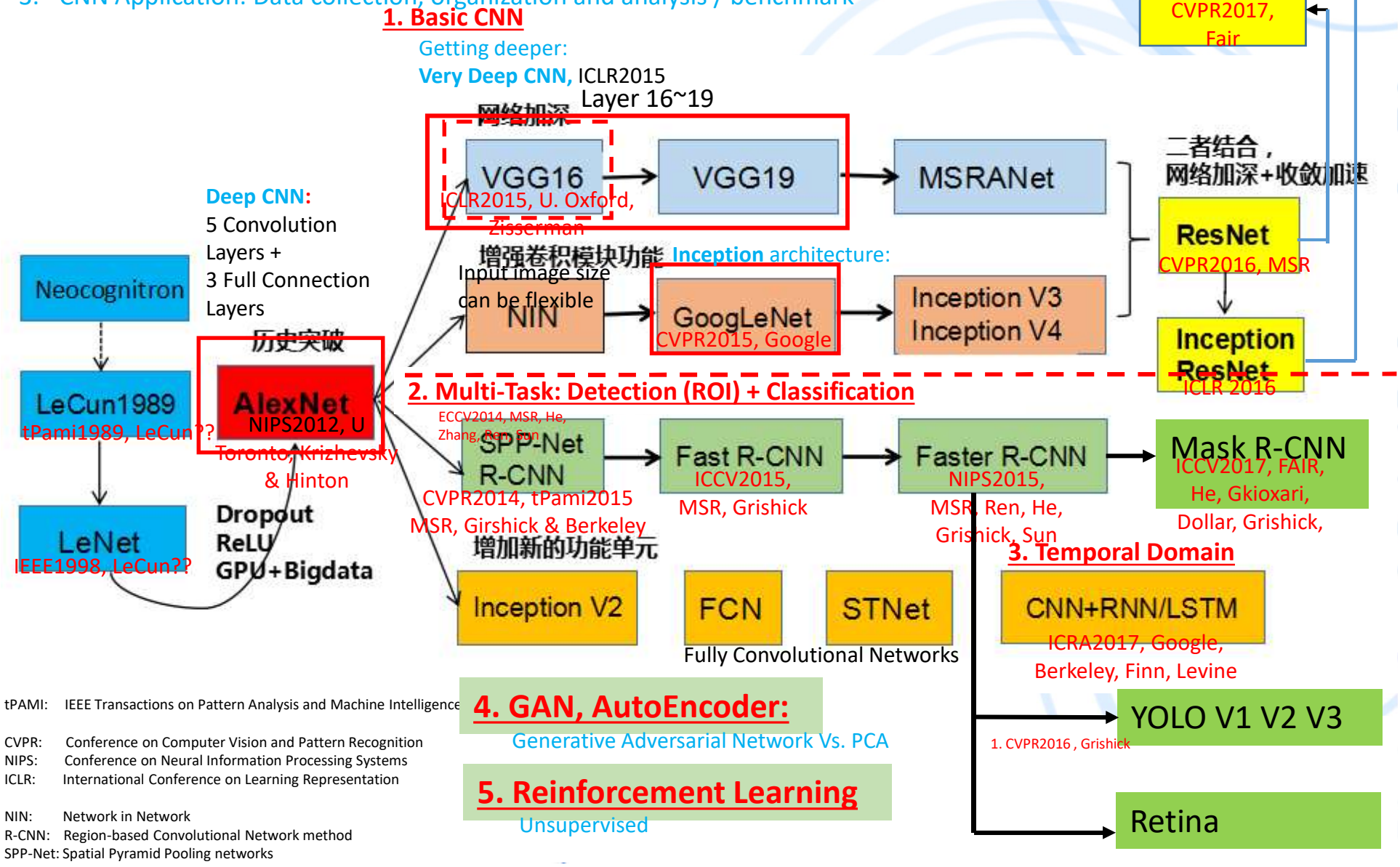
2) GAN: Deblurring



7.0 Deep Learning: 5. Reinforcement Learning

Which one do I want to be?

1. Basic CNN Creation
2. Modified CNN
3. CNN Application: Data collection, organization and analysis / benchmark



tPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence

CVPR: Conference on Computer Vision and Pattern Recognition
NIPS: Conference on Neural Information Processing Systems
ICLR: International Conference on Learning Representation

NIN: Network in Network
R-CNN: Region-based Convolutional Network method
SPP-Net: Spatial Pyramid Pooling networks

7.0 Reinforcement: Neural network learning category

1) Supervised Learning: PCA, LDA

- 給定 input 和 label (target) 的學習方式。

2) Unsupervised Learning: VQ

- 只給定input，學習 input data 間的關係，可用來分群。

3) Semi-Supervised Learning

- 給定少部分資料有 label，大部分資料沒 label，使沒label的資料得到對應的 label。

4) Reinforcement Learning: Self-Supervised

- 在學習過程中設立 reward (with environment)，藉由 reward 機制自我學習,
- need policy (brain of agent) as rule for agent to learn。

7.0.1 Reinforcement Learning

(One Episode, 一局) **MDP (Markov Decision Process)**

T: termination

Trajectory $\tau = \{s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T\}$

s_0 joint determined

Reward:
1) Delay reward as GO (long-term reward)
2) Immediate reward

-Sometimes, reward can't be received immediately (long-term reward).
-The reward is delay, such as, until game is over, then we can understand which one is the winner (reward=1) or loser (reward=0 or -1)

Learning to paly Go

Agent learns to take actions to maximize expected reward.

As rule for agent's brain

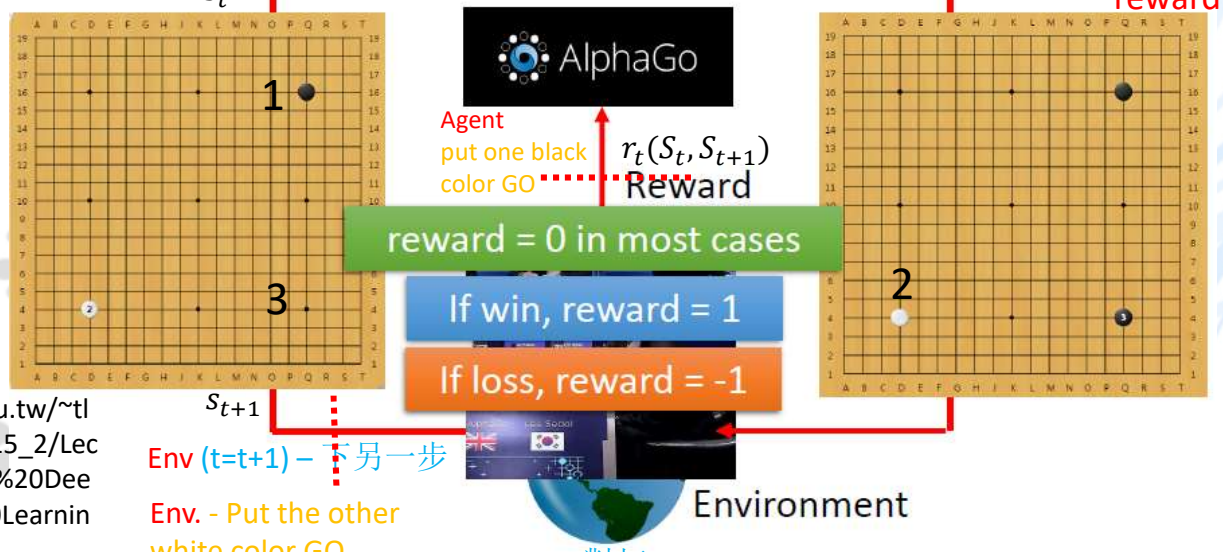
Actor/Policy : π

$\pi(a_t|s_t)$

State /Observation s_t

Action a_t

Long-term reward



Env (t=t+1) – 下另一步

Env. - Put the other white color GO
Now, there are 2 white and 2 black

對打:
Agent AlphaGo Action (t) –下一步 black 3 GO
Env – white 下另一步 4

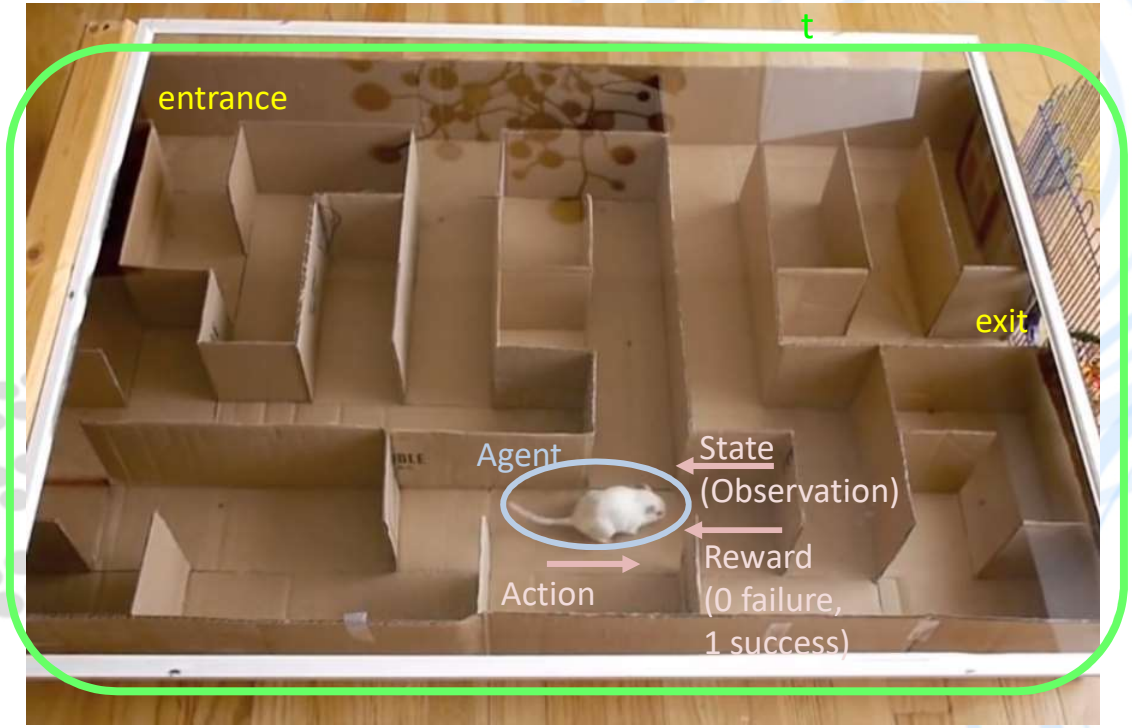
截自：
http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Introduction%20to%20Deep%20Reinforcement%20Learning.pdf

7.0.2 Reinforcement Learning

補充

Reinforcement Learning – Example 1

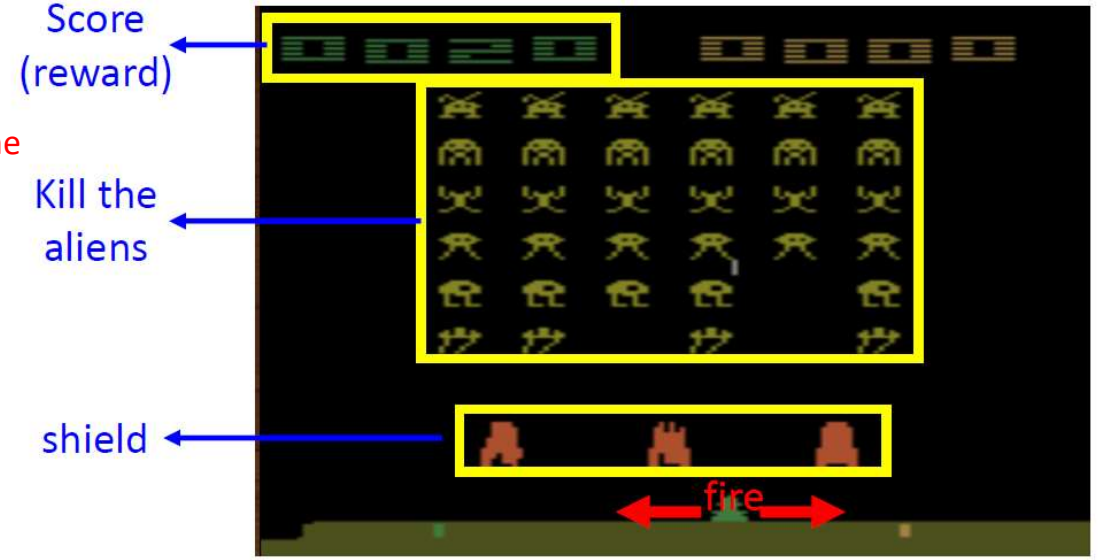
If we define
walk toward exit,
reward is 1,
Otherwise reward is 0
the reward is delay



7.0.3 Reinforcement Learning

- Space invader

Termination: all the aliens are killed, or your spaceship is destroyed.



Agent learns to take actions to maximize expected reward.

Observation: current frame
Action : left, right, fire
Reward : Score
Episode : 一局遊戲
Termination: ….

Long-term reward

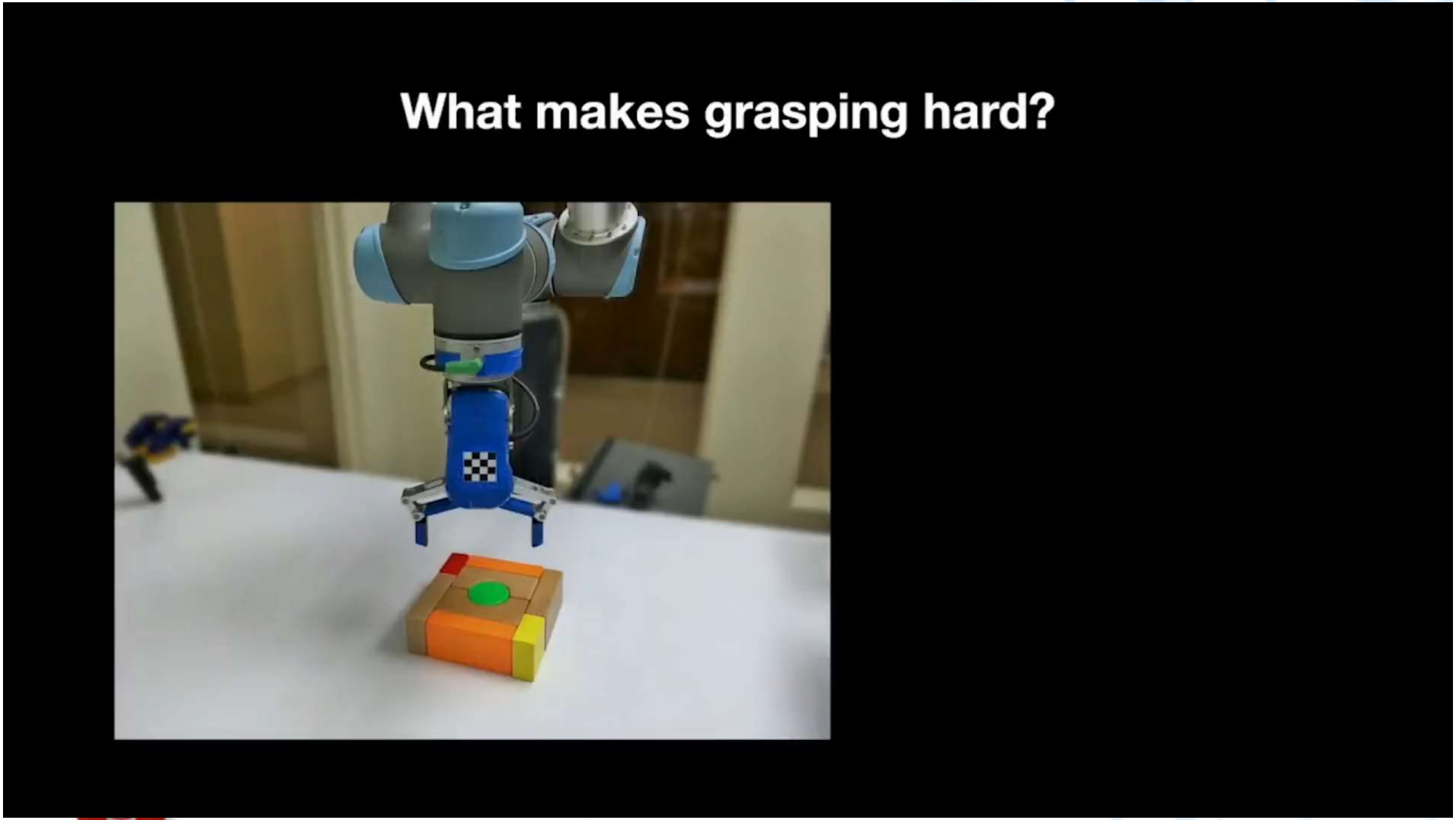
If agent doesn't maximize the long-term reward, the agent can only take fire action

Agent need long term planning

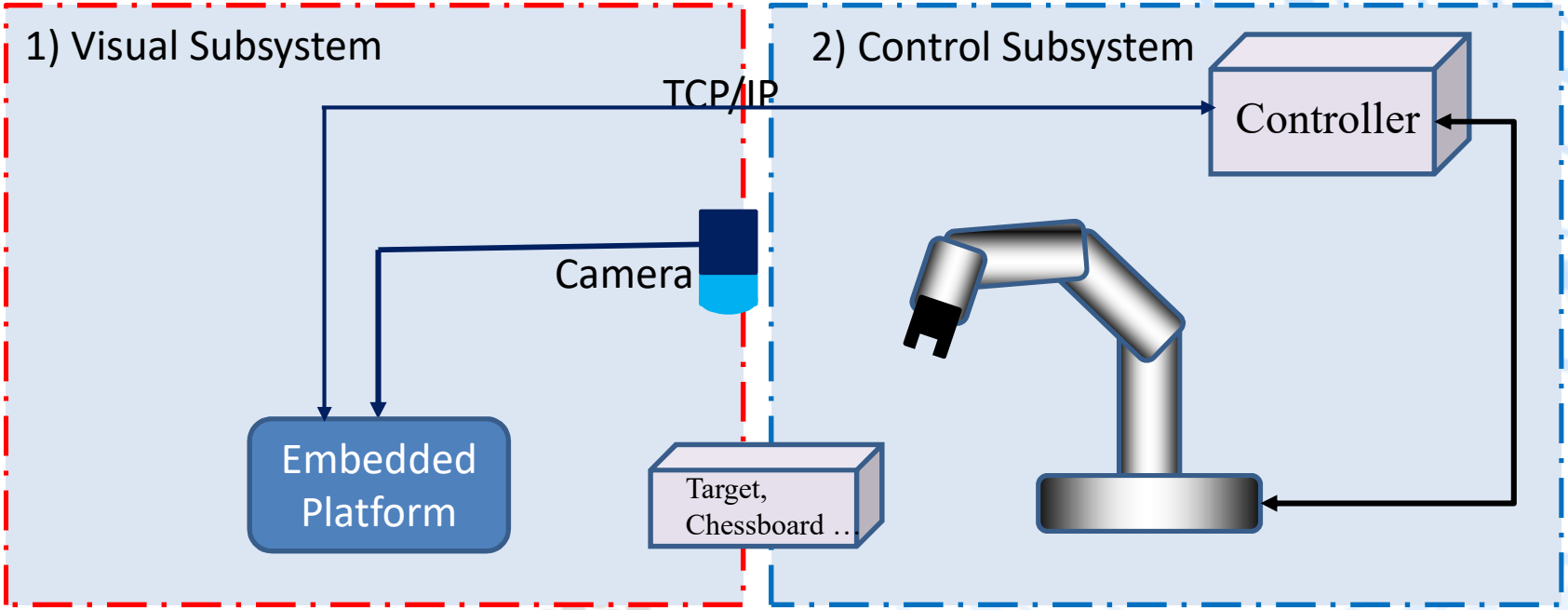
截自：
http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Introduction%20to%20Deep%20Reinforcement%20Learning.pdf

7.1 Reinforcement Learning: Robot Arm Grasp ~~or Push~~, and Place – Detection, Classification, and Angle Estimation ~~or Push~~ -(1/2) 03:22.44

7.1 Reinforcement Learning: Robot Arm Grasp or Push, and Place – Detection, Classification, and Angle Estimation or Push (2/2) 02:30.83



8.0 Embedded Deep Learning by Nvidia Jetson TX2

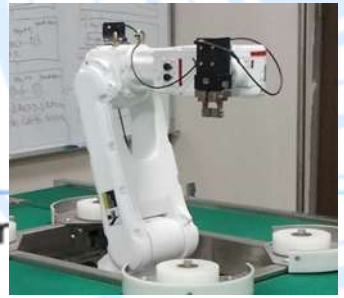


1.1) Camera

1.2) Jetson TX2

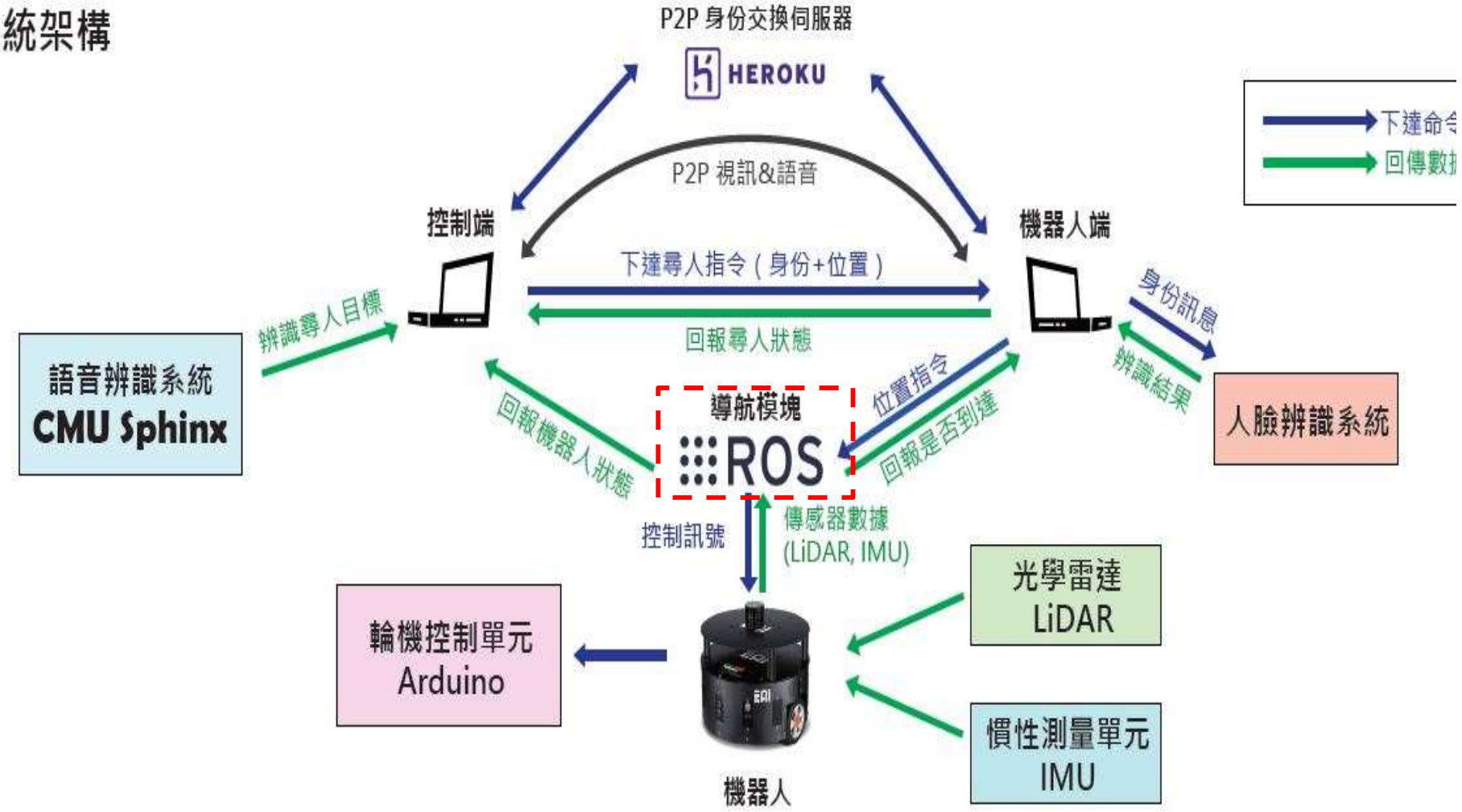
2.1) Controller

2.2) Robot Arm



9.0 AGV (Automatic Guided Vehicle) + ROS + Sensor

系統架構



9.1 AGV (Automatic Guided Vehicle): 北科大 合作夥伴