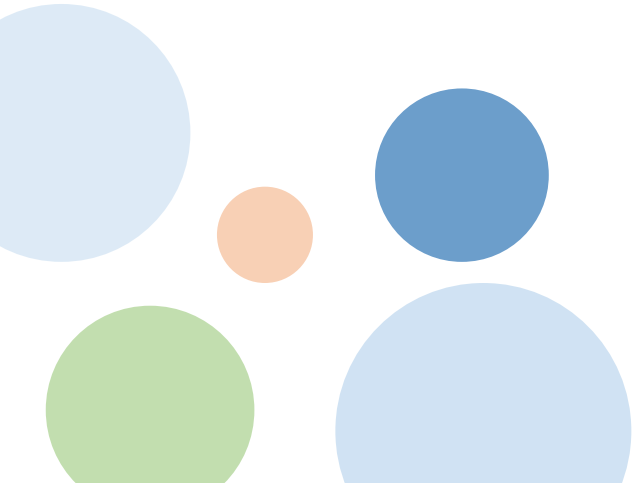
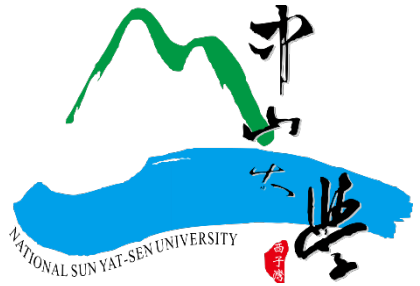
Applications in Computer Vision





Outlines

- image classification and object detection
- segmentation
- human pose estimation
- caption generation
- style transfer
- image enhancement
- facial expression recognition (FER)

Deep Learning Application Demo

- classification
  - ANN (digit recognition): https://www.youtube.com/watch?time_continue=12&v=aircAruvnKk
  - Training (MNIST, CIFAR-10): https://cs.stanford.edu/people/karpathy/convnetjs
  - Video (top-3): https://www.youtube.com/watch?v=qrzQ_AB1DZk
- detection
  - YOLOv3: https://pjreddie.com/darknet/yolo
- segmentation
  - image segmentation: http://mi.eng.cam.ac.uk/projects/segnet/demo.php#demo
  - instance segmentation and removal: https://www.youtube.com/watch?v=OAWCp7OXLnY
- pose estimation
  - dense human pose in the wild: https://www.youtube.com/watch?v=Dhkd_bAwwMc
- caption generation
  - http://dbs.cloudcv.org/captioning
- style transfer
  - artistic painting https://www.instapainting.com/assets

References and Credits

- Stanford CS231n, “Convolutional Neural Networks for Visual Recognition”
  - by Fei-Fei Li, Justin Johnson, and Serena Yeung
- MIT 6.S191, “Deep Learning”
  - by Alexander Amini, and Ava Soleimany
- UVA Deep Learning , Univ. of Amsterdam
  - by Efstratios Gavves
- CMSC 35264 Deep Learning, Univ. of Chicago
  - by Shubhendu Trivedi and Risi Kondor
- Deep Learning for Computer Vision
  - by 台大資工系 莊永裕 教授
- book: Deep Learning
  - by Ian Goodfellow, Yoshua Bengio, and Aaron Courville

Image Classification and Object Detection

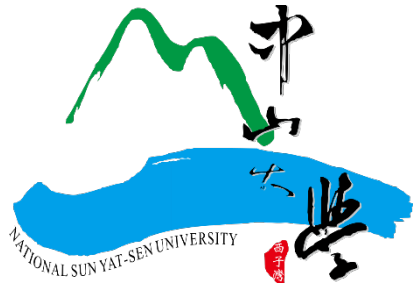
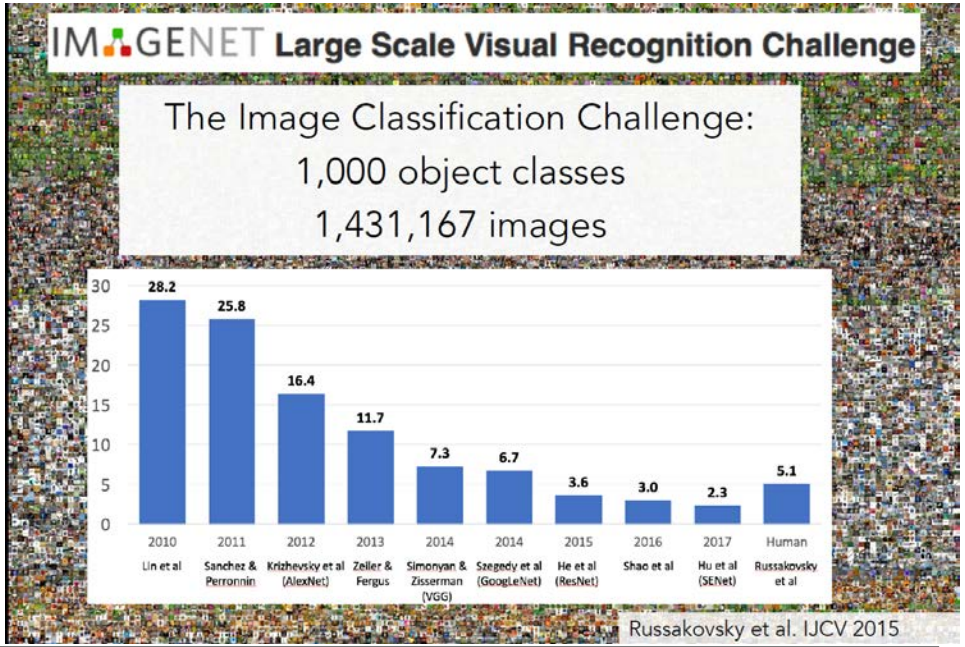


Image Classification in ILSVRC

- ImageNet dataset
  - 140M images
  - 22K categories
- ILSVRC (Image Large Scale Visual Recognition Challenge (ILSVRC)
  - 1000 object categories
  - top-1 and top-5 error rates



This image is CC0 public domain

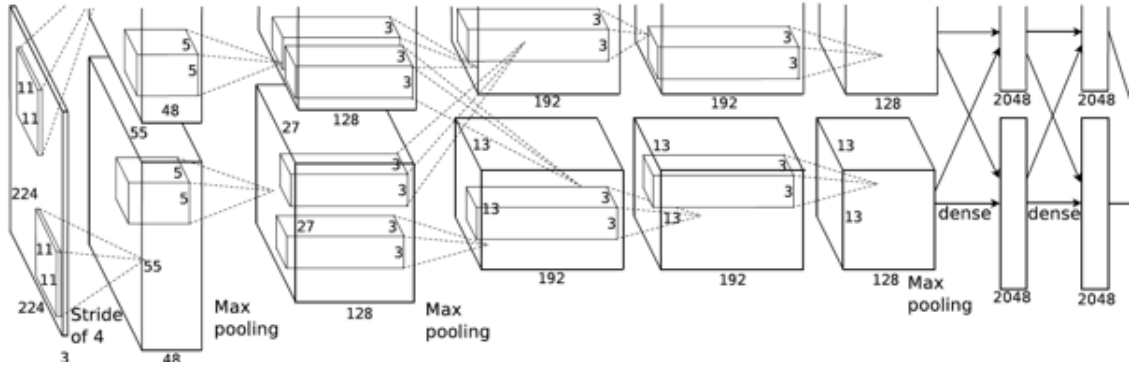


Figure copyright Alex Krizhevsky, ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission

Vectors: 4096

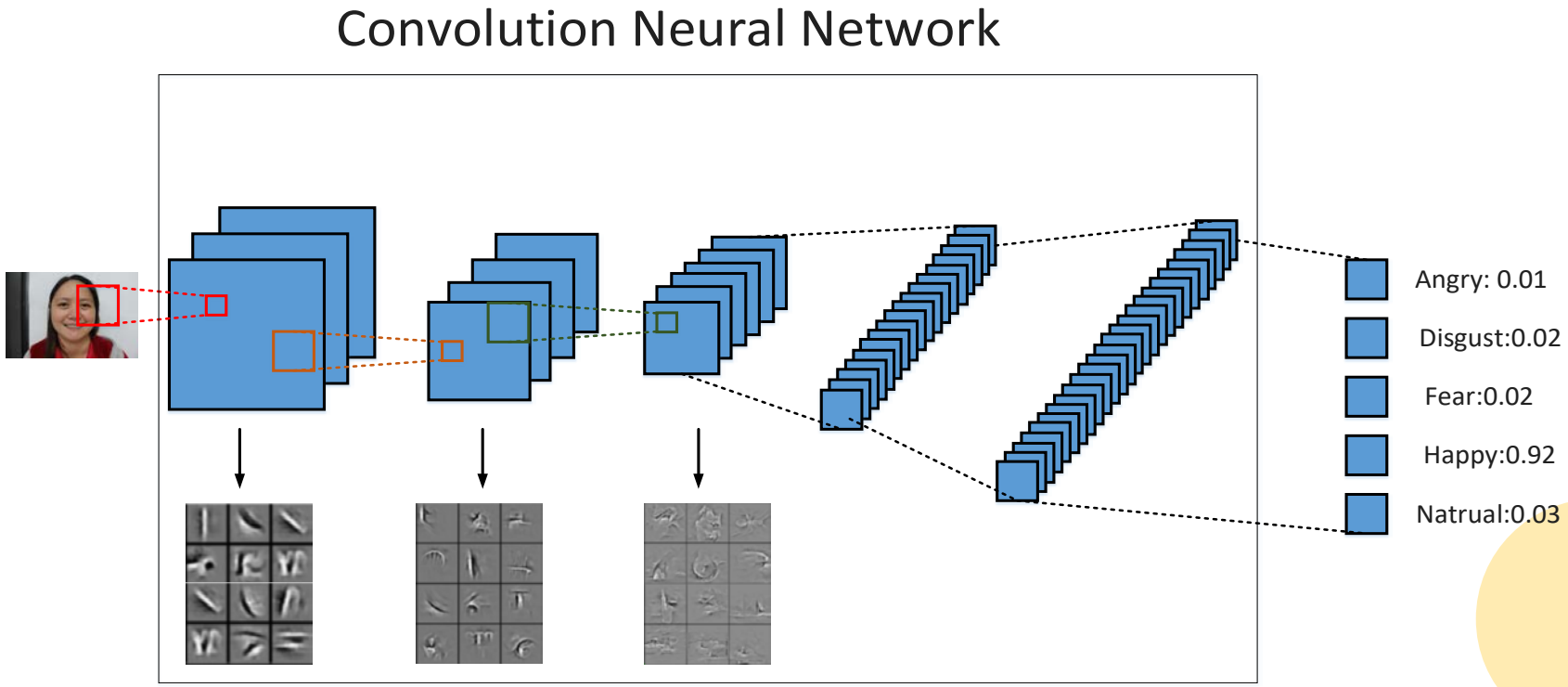
Fully-Connected: 4096 to 1000

Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01
…

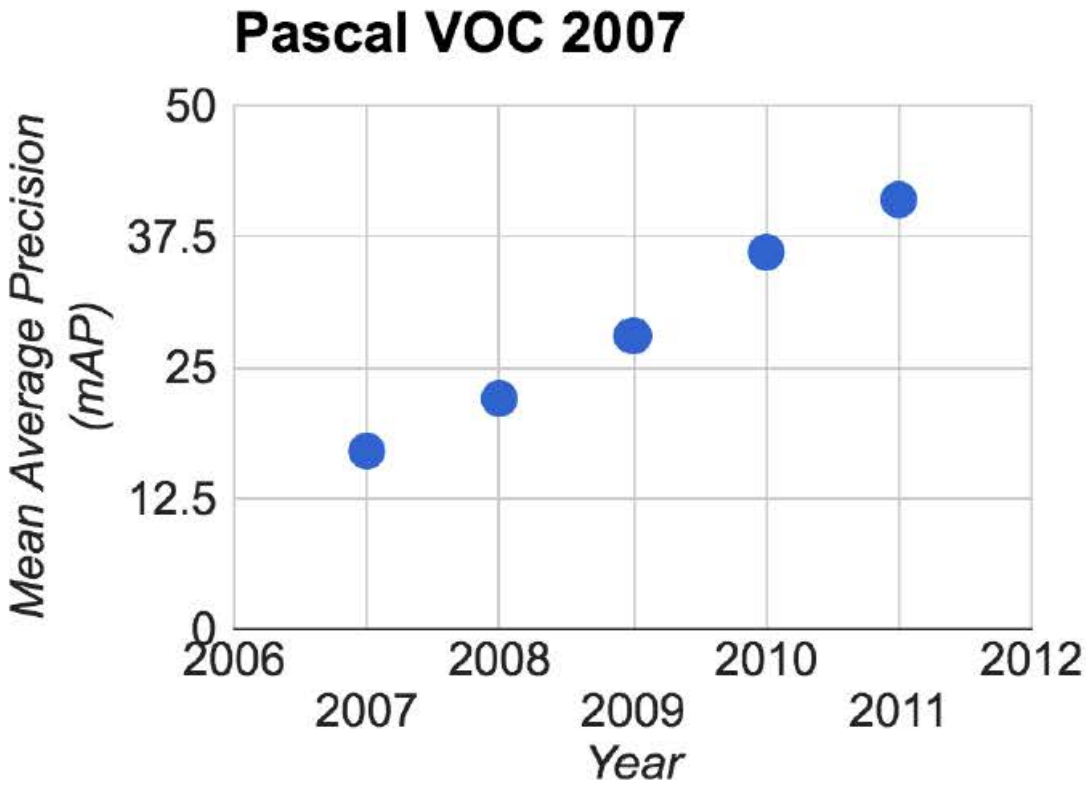
Image classification

- 由多層Convolution (Conv) layers 和 Fully connected (FC) layers所組成
  - 利用Convolution layers→進行圖片**特徵擷取**
  - 利用Fully connected layers→進行圖片的**分類**



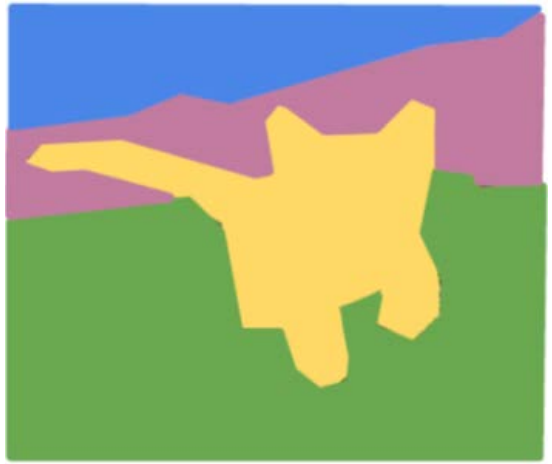
Object Detection in PASCAL VOC

- PASCAL Visual Object Challenge (VOC): 20 categories



Detection and Segmentation

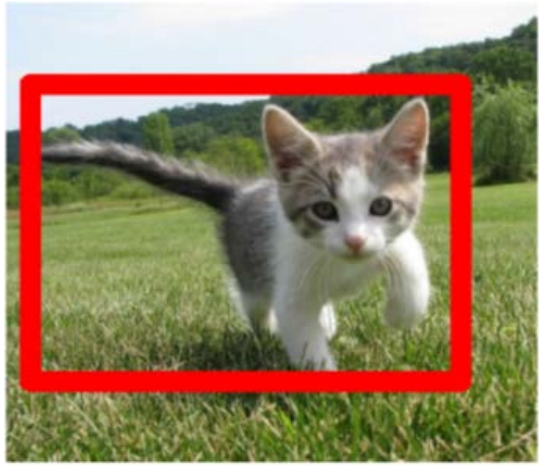
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Classification +Localization



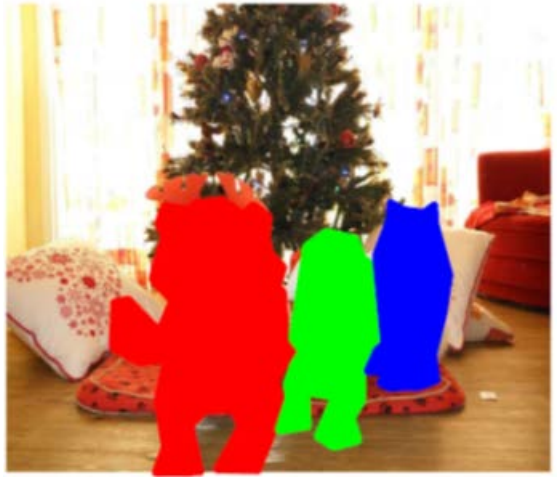
CAT

Single Object

Object Detection

DOG, DOG, CAT

Instance Segmentation



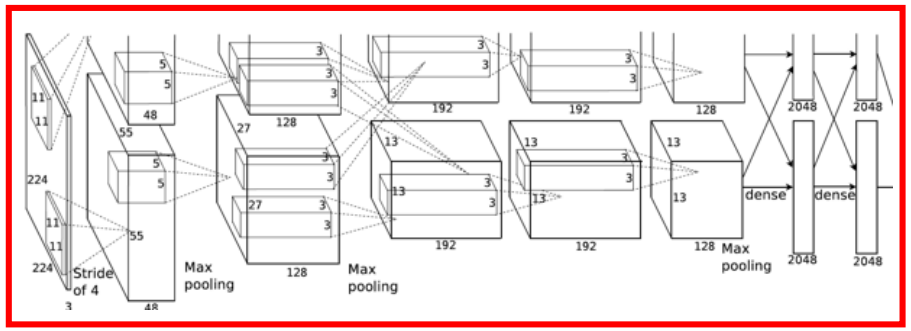
DOG, DOG, CAT

Multiple Object

Detection = Classification + Localization

- classification
  - find class scores
- localization
  - find bounding box coordinate

This image is CC0 public domain



Often pretrained on ImageNet
(Transfer learning)

Vectors: 4096

Fully Connected: 4096 to 1000

Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01
…

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

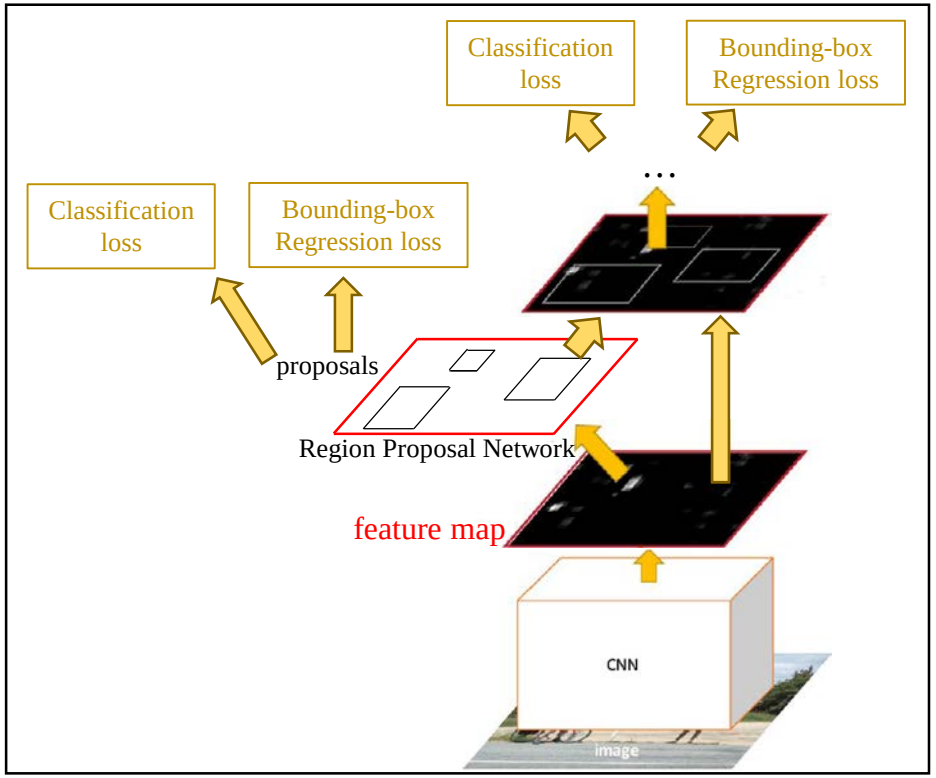
L2 Loss

+ → **Loss**

Correct box (x', y', w', h')

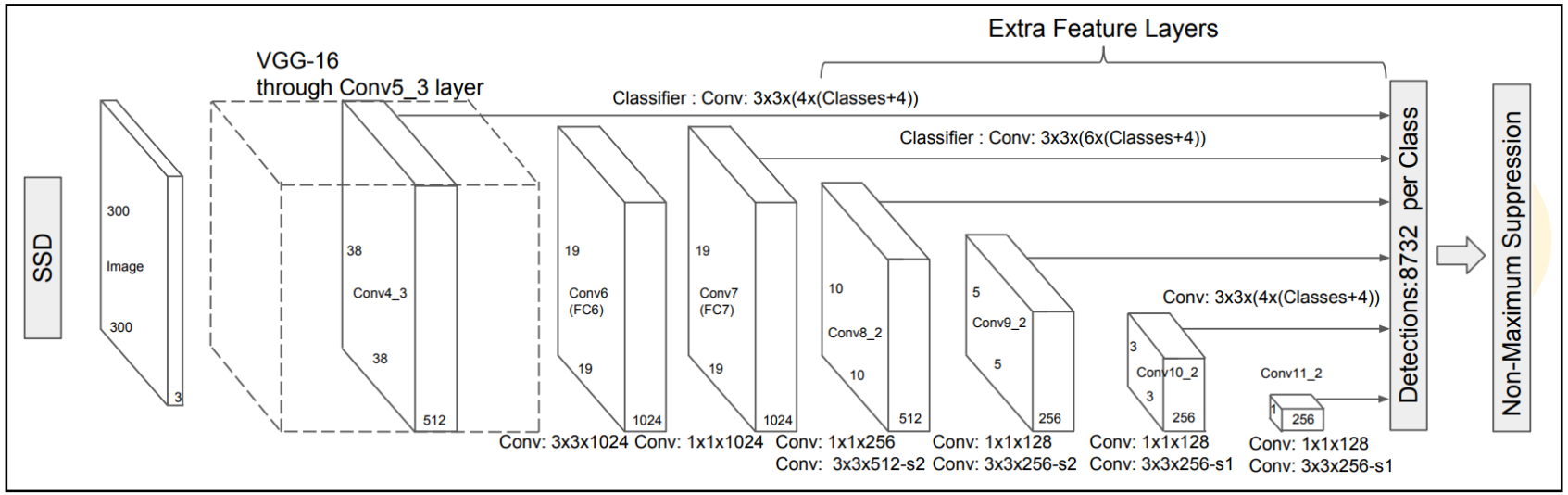
Object Detection: two-stage vs. one-stage

Two-Stage Detector: Faster RCNN (Region CNN)



影像特徵擷取 和 目標物件後選區 為兩個獨立網路
→運算量龐大
→運算速度不佳，無法達到即時的物件偵測

One-Stage Detector: Single Shot Detector (SSD)

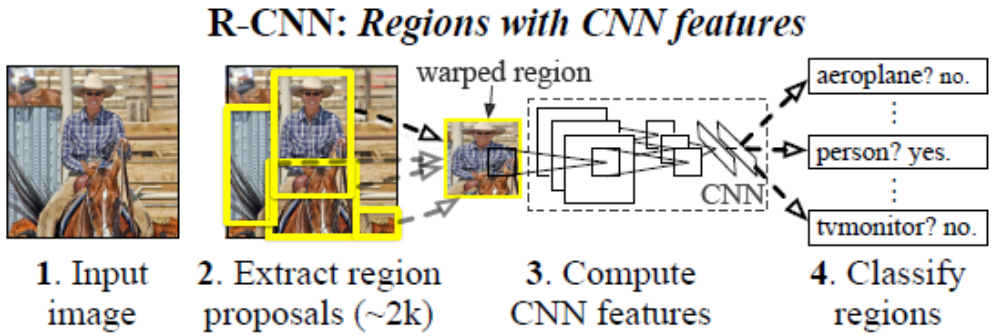


→運算速度提升
→可達到即時的物件偵測

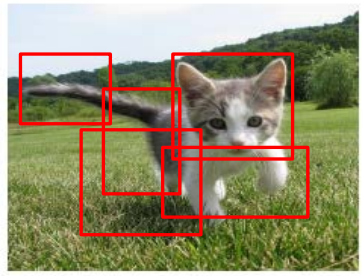
Two-stage Detector(1/2)

- R-CNN (2014 CVPR)
  - generate region proposals using other computer vision techniques (e.g. selective search)
  - use CNN to compute feature maps for each proposal
  - use SVM for classification

R-CNN



Region Proposals



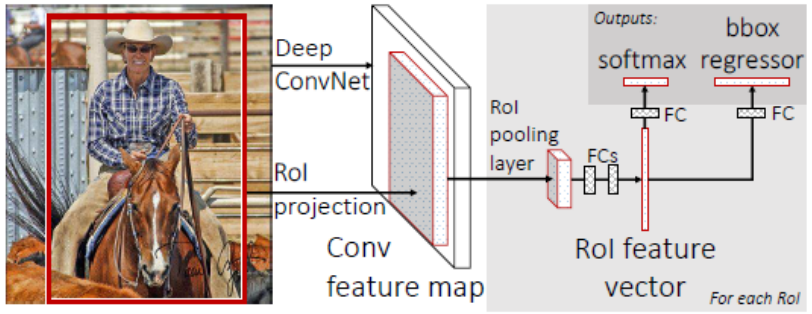
Selective Search、HOG、LBP

DNN

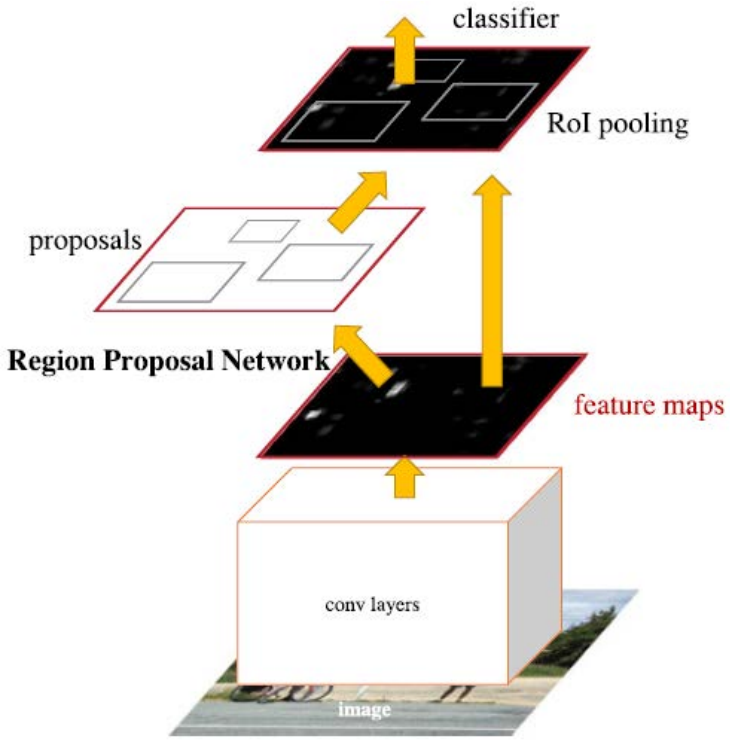
CAT

Two-stage Detector(2/2)

- Fast R-CNN (2015 ICCV)
  - ROI pooling to replace warpping in R-CNN
- Faster R-CNN (2015, 2017 TPAMI)
  - RPN to generate region proposals

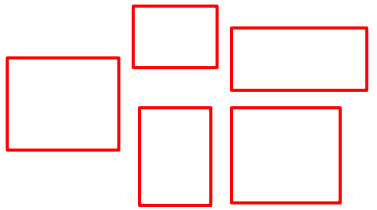


Fast R-CNN



Faster R-CNN

Region Proposal



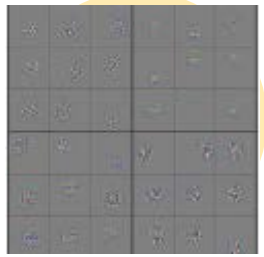
x, y, length, width

ROI Pooling

CAT

Feature extraction

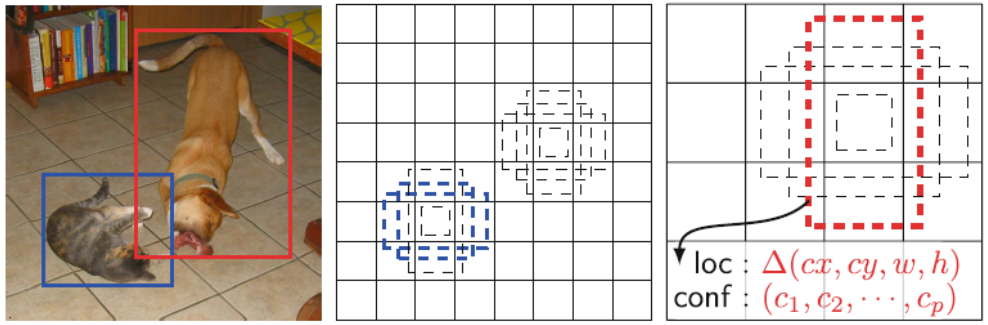
DNN



Feature Map

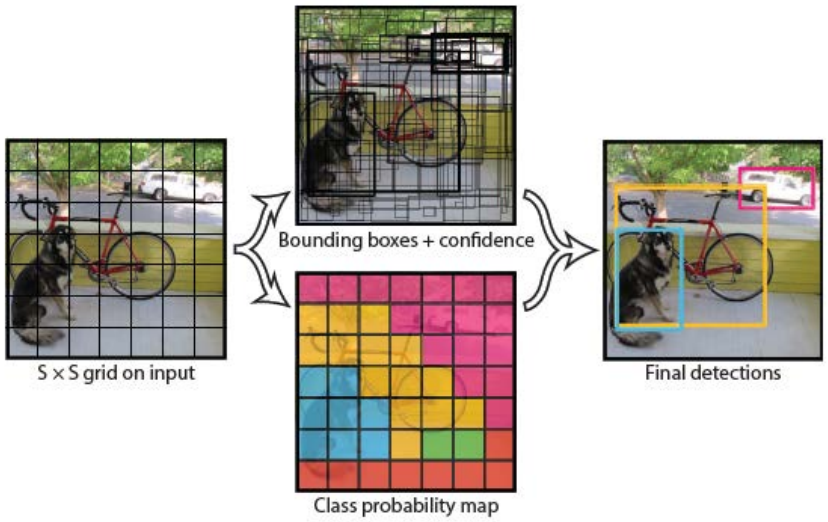
One-stage Detector

- SSD (2016 ECCV)
- YOLO v1~v3 (2016~2018)
- Same CNN for region proposal and classification



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

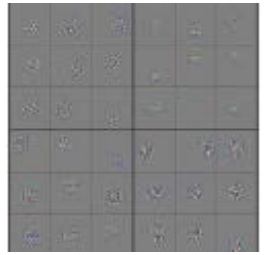
SSD



YOLO

Feature extraction

提取影像特徵



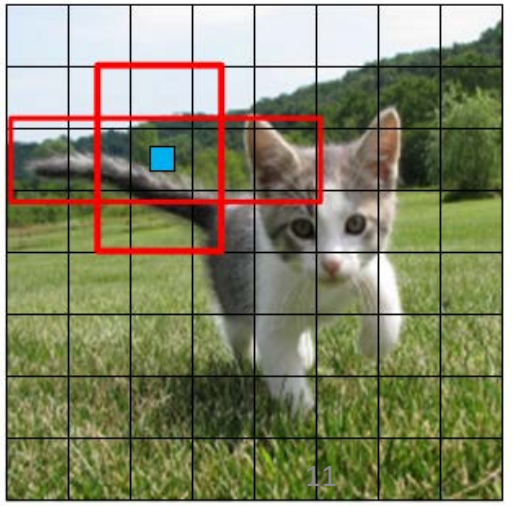
Feature Map

DNN

Detection

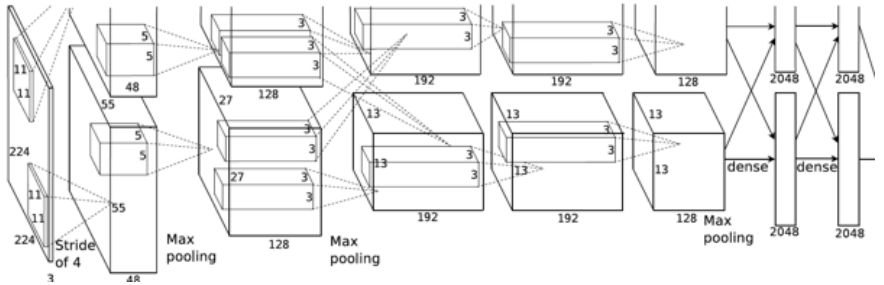
預測目標物件位置、種類

CAT

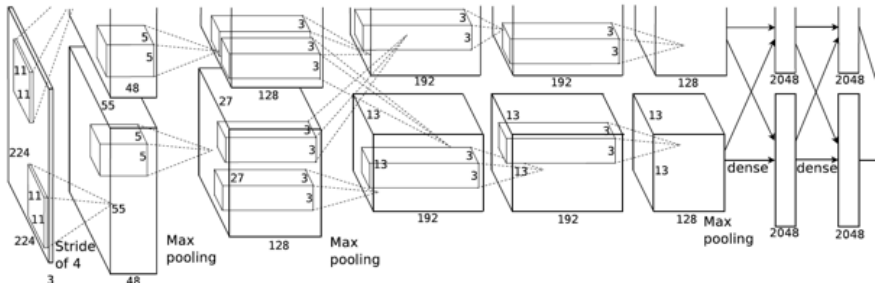


Object Detection as Regression ?

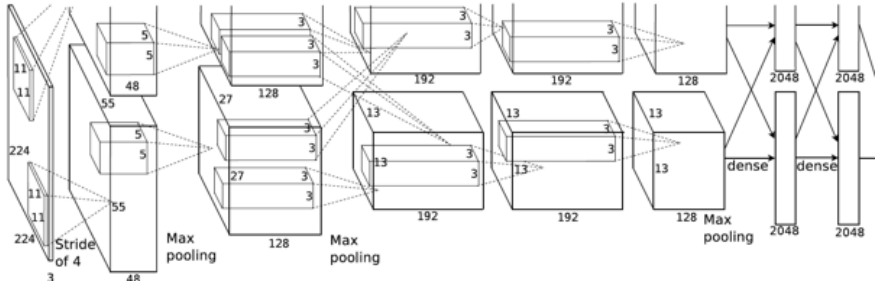
Each image needs a different number of outputs !



CAT: (x, y, w, h) 4 numbers

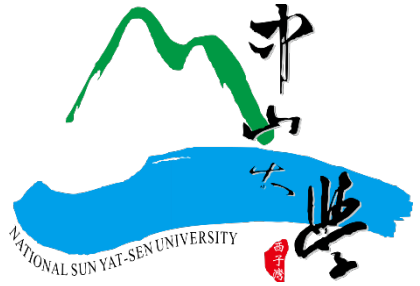


DOG: (x, y, w, h)
DOG: (x, y, w, h) 16 numbers
CAT : (x, y, w, h)



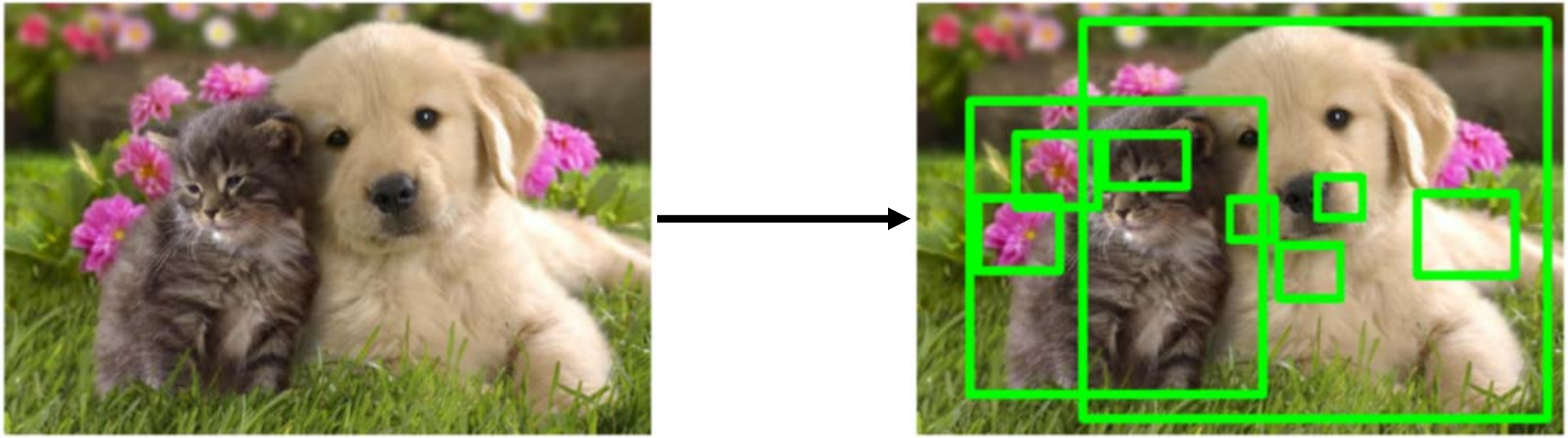
DUCK: (x, y, w, h) Many
DUCK: (x, y, w, h) numbers !
...

Two-Stage Detector



Region Proposals / Selective Search

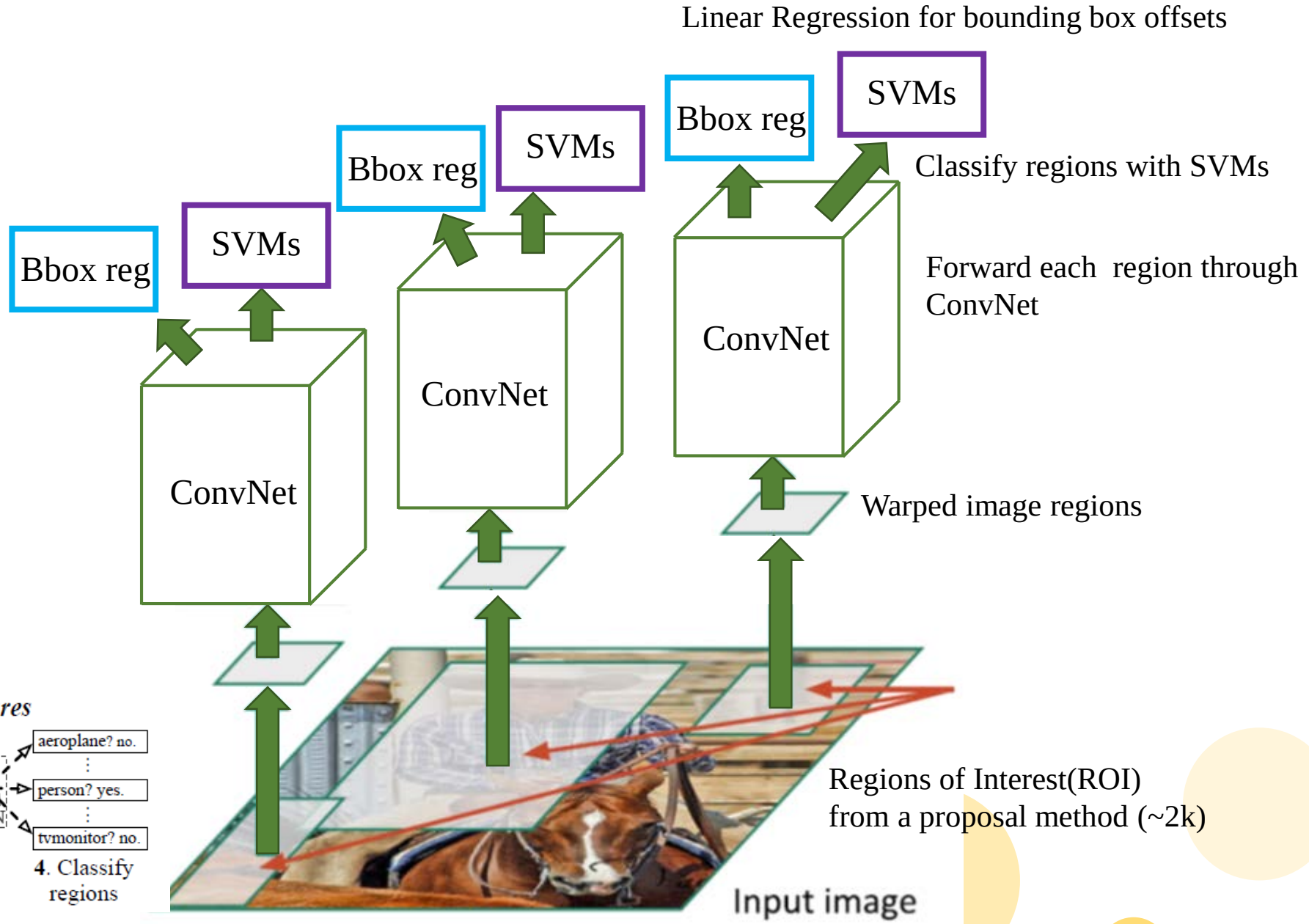
- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU

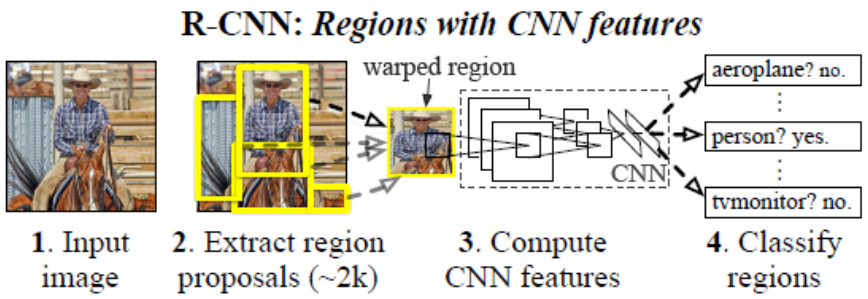


Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

R-CNN*

- generate region proposals using selective search
- use CNN to compute features for each proposal
- classify and localize

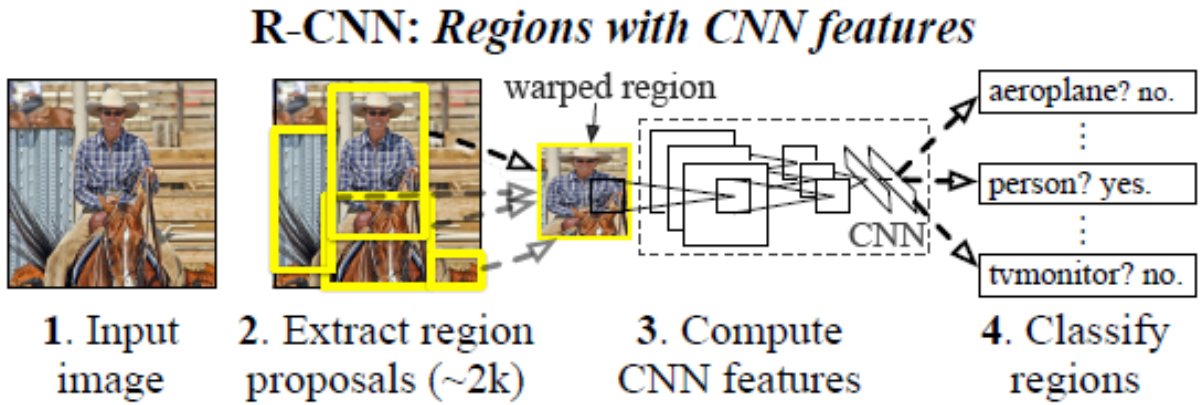




*R. Girshick, et al, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” CVPR, 2014.

R-CNN Results and Problems

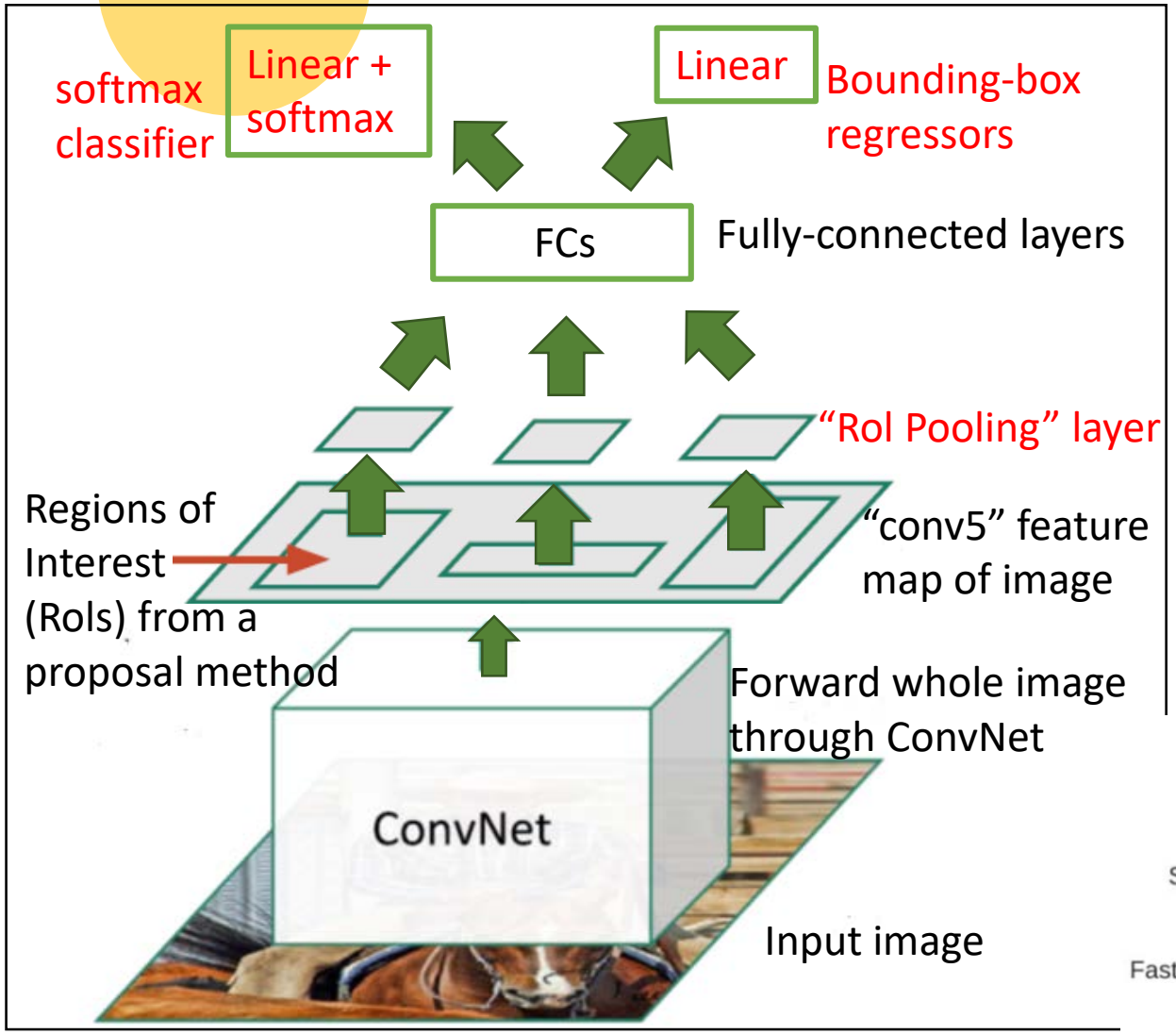
- Ad hoc training objectives
  - Fine-tune ConvNet with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16

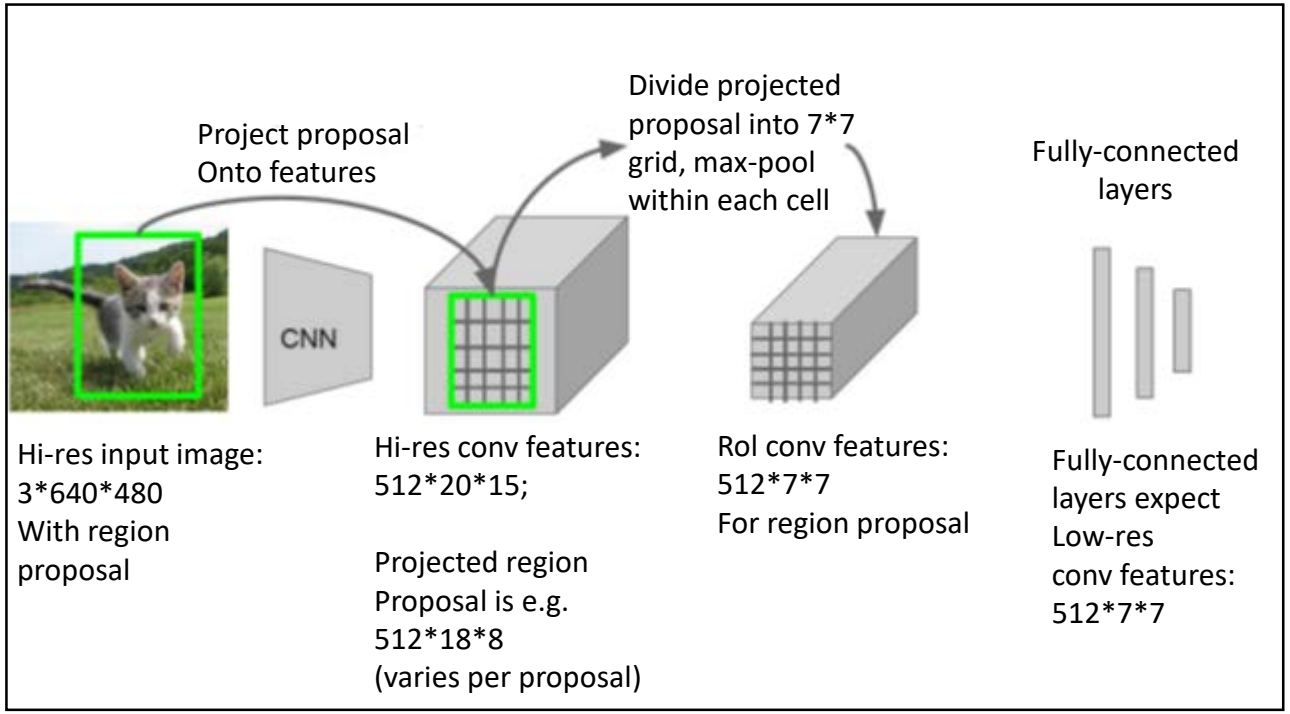


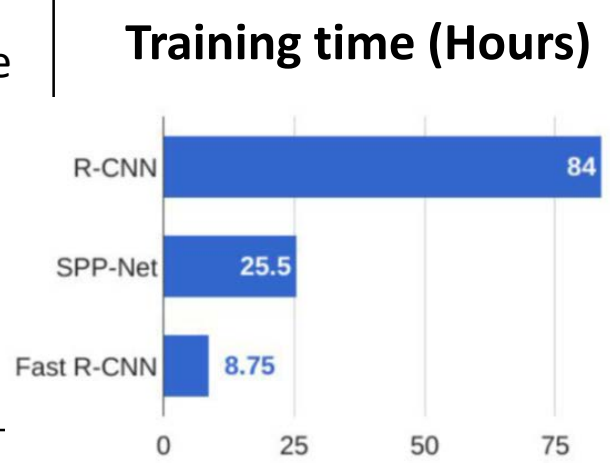
VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [17]†	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [32]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [35]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [15]†	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	**71.8**	**65.8**	**53.0**	**36.8**	**35.9**	**59.7**	**60.0**	**69.9**	**27.9**	**50.6**	**41.4**	**70.0**	**62.0**	**69.0**	**58.1**	**29.5**	**59.4**	**39.3**	**61.2**	**52.4**	**53.7**

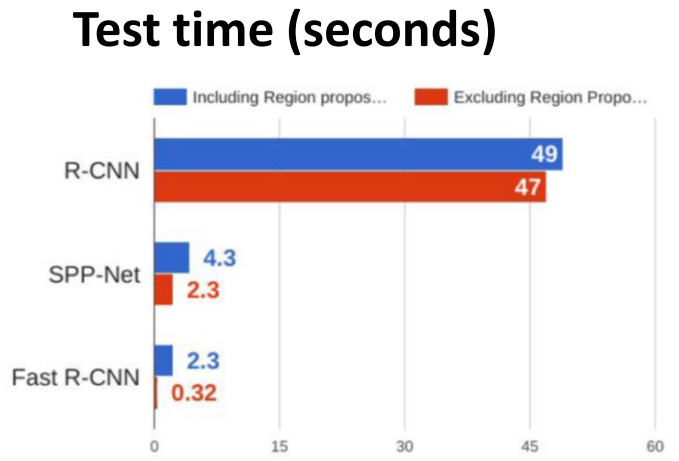
Fast R-CNN

- ROI Pooling





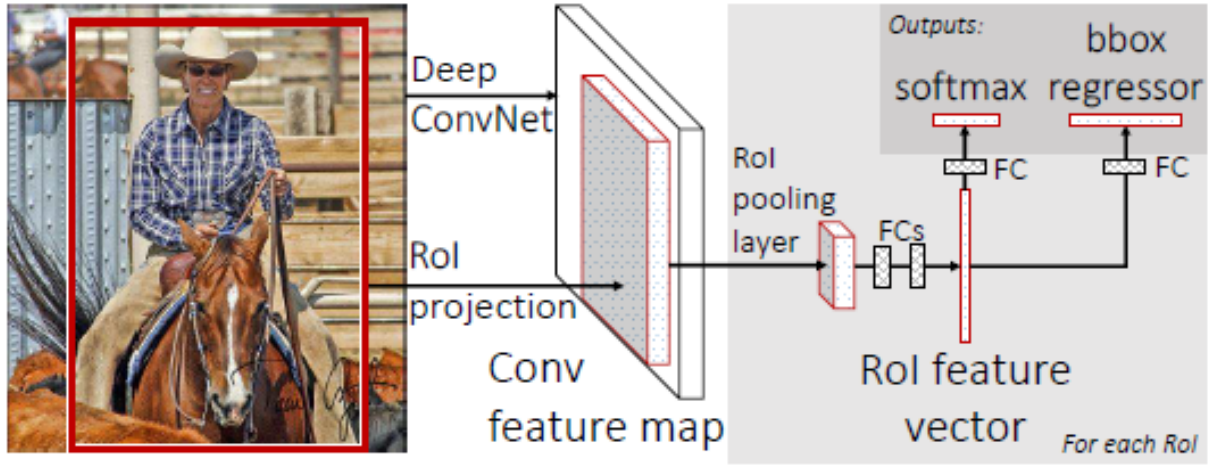




*R. Girshick, "Fast R-CNN," ICCV, 2015.

Fast R-CNN

- for VGG16-based object detector
  - 9x faster than R-CNN for training
  - 213x faster than R-CNN for testing

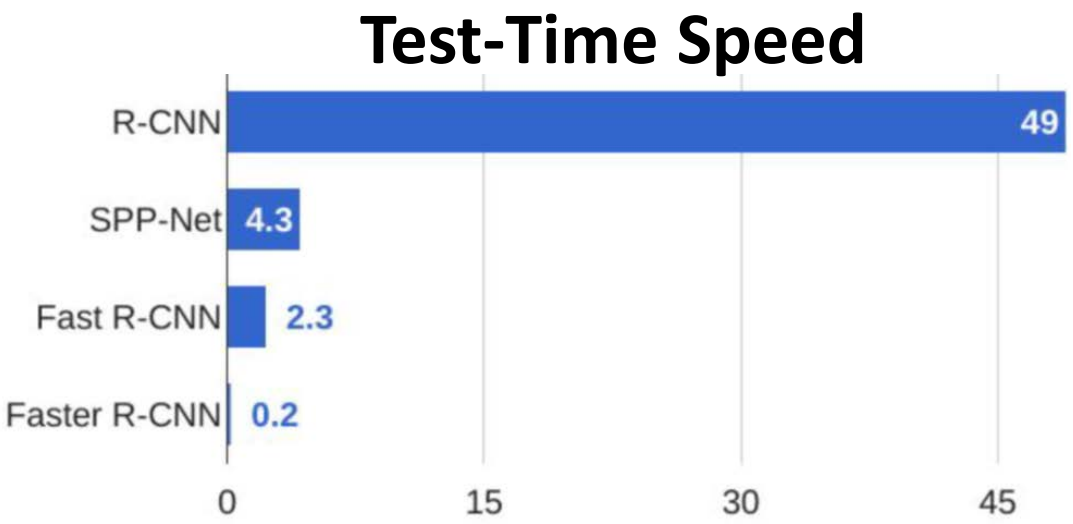


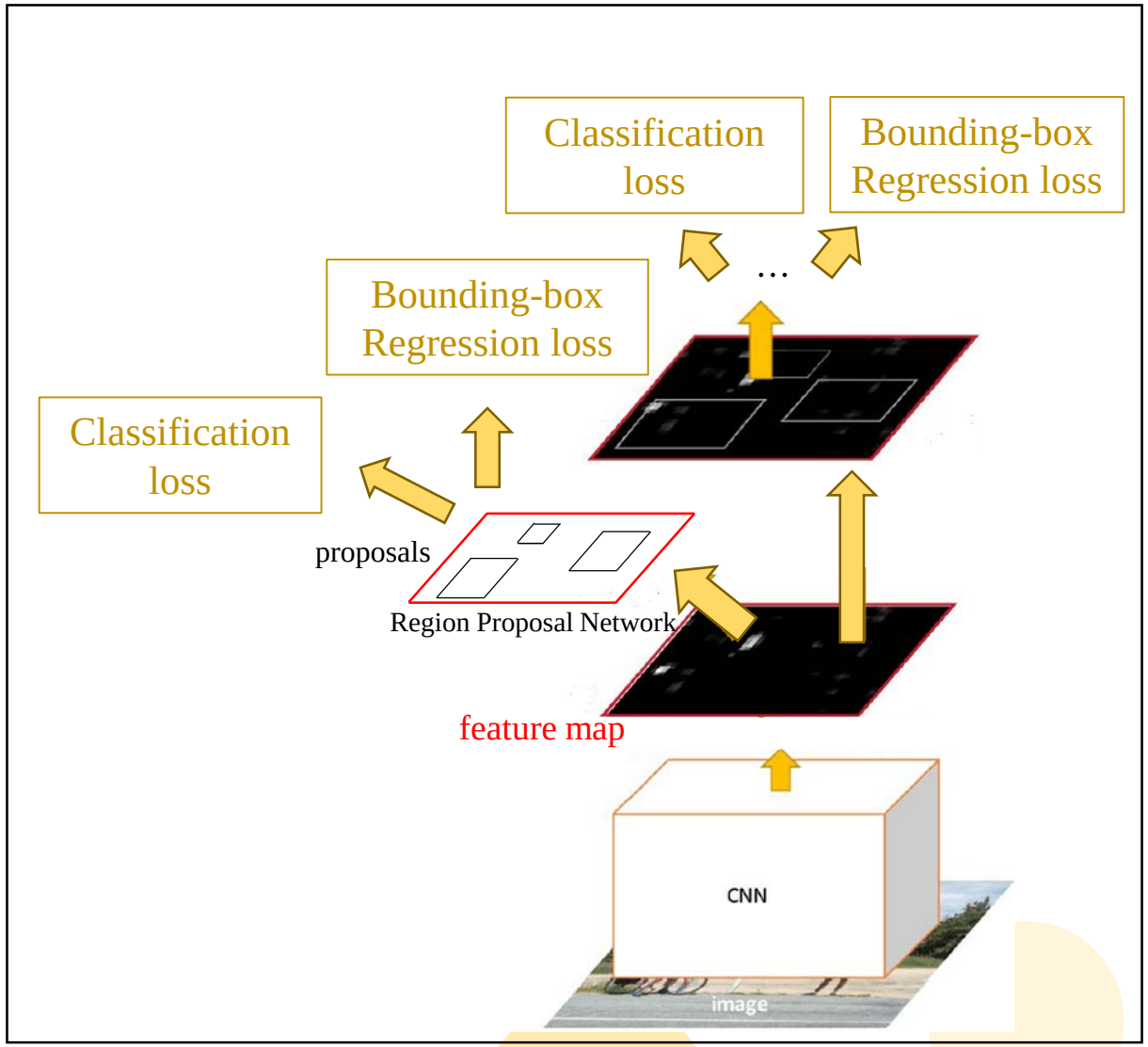
method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

Faster R-CNN

- Make CNN do proposals!
- Insert Region Proposal Network (RPN) to predict proposals from features
- Jointly train with 4 losses :
    1. RPN classify object / not object
    2. RPN regress box coordinates
    3. Final classification score (object classes)
    4. Final box coordinates

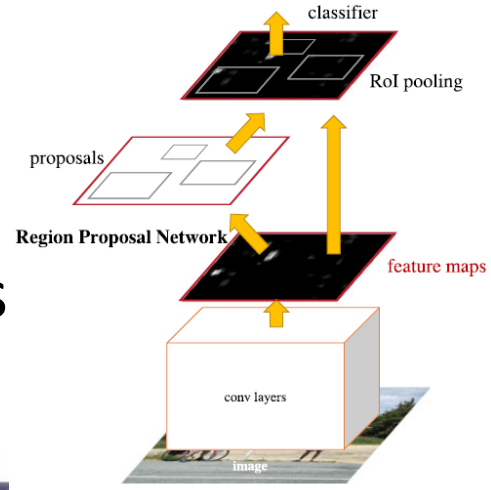


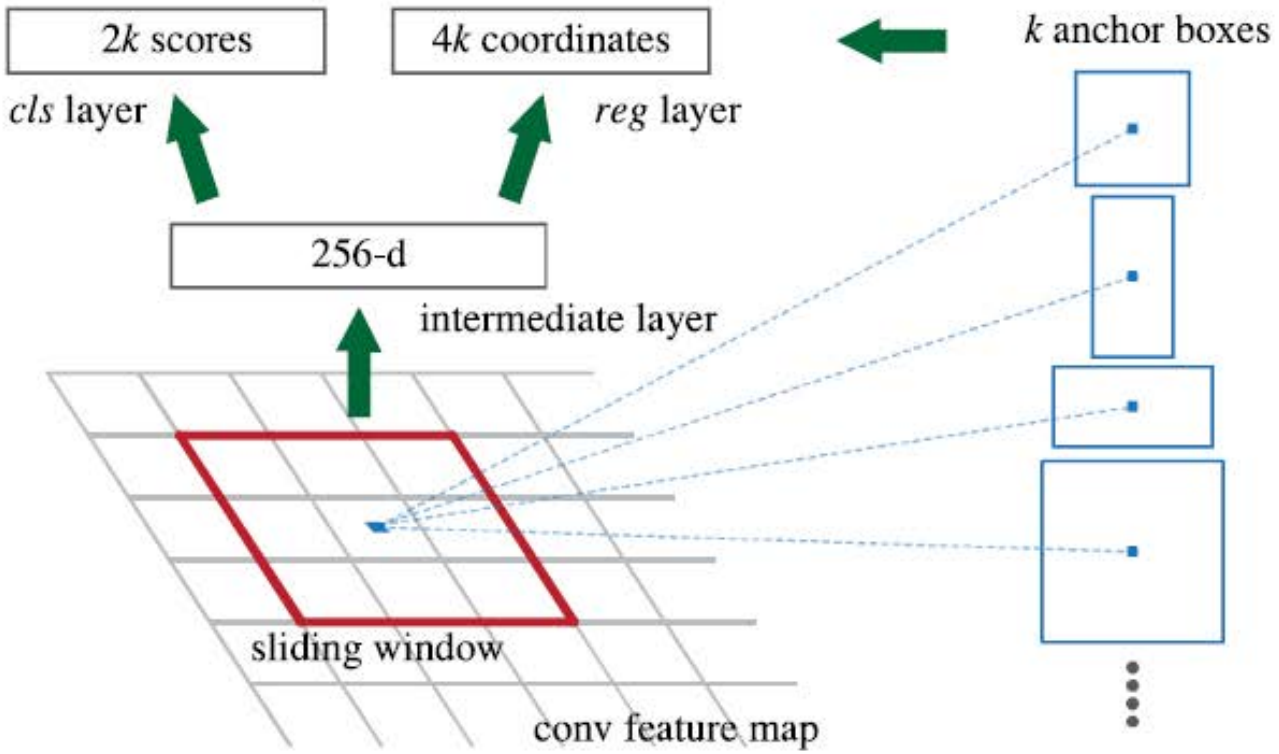


*S. Ren, et al., “Faster R-CNN: Toward Real-Time Object Detection with Region Proposal Network,” TPAMI, 2017.

RPN with Anchors

- 3x3 sliding window
- k=9 anchors to capture objects of various sizes and aspect ratios
  - 3 scales, each with 3 aspect ratios





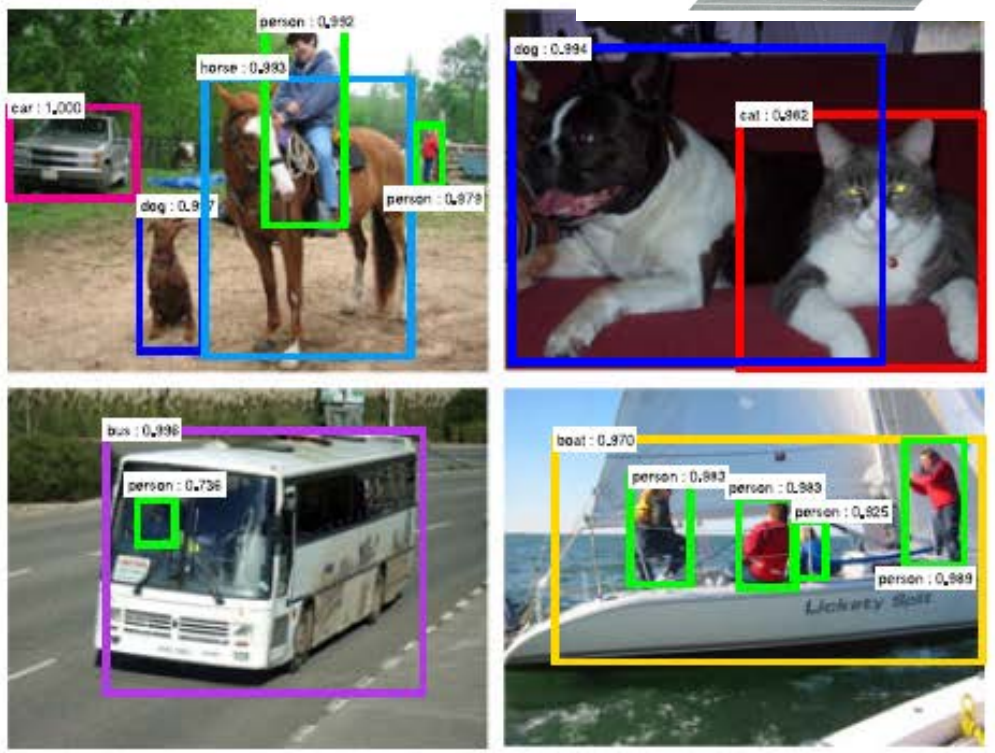


Fig. 3. **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

Faster R-CNN with RPN

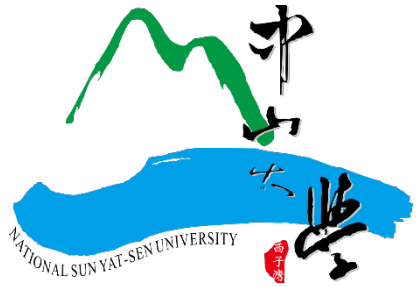
TABLE 8
Detection Results of Faster R-CNN on PASCAL VOC 2007 Test Set Using **Different Settings of Anchors**

settings	anchor scales	aspect ratios	mAP (%)
1 scale, 1 ratio	128^2	1:1	65.8
	256^2	1:1	66.7
1 scale, 3 ratios	128^2	{2:1, 1:1, 1:2}	68.8
	256^2	{2:1, 1:1, 1:2}	67.9
3 scales, 1 ratios	$\{128^2, 256^2, 512^2\}$	1:1	**69.8**
3 scales, 3 ratios	$\{128^2, 256^2, 512^2\}$	{2:1, 1:1, 1:2}	**69.9**

TABLE 6
Results on PASCAL VOC 2007 Test Set with Fast R-CNN Detectors and VGG-16

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2,000	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
SS	2,000	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
RPN*	300	07	68.5	74.1	77.2	67.7	53.9	51.0	75.1	79.2	78.9	50.7	78.0	61.1	79.1	81.9	72.2	75.9	37.2	71.4	62.5	77.4	66.4
RPN	300	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
RPN	300	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
RPN	300	COCO+07+12	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9

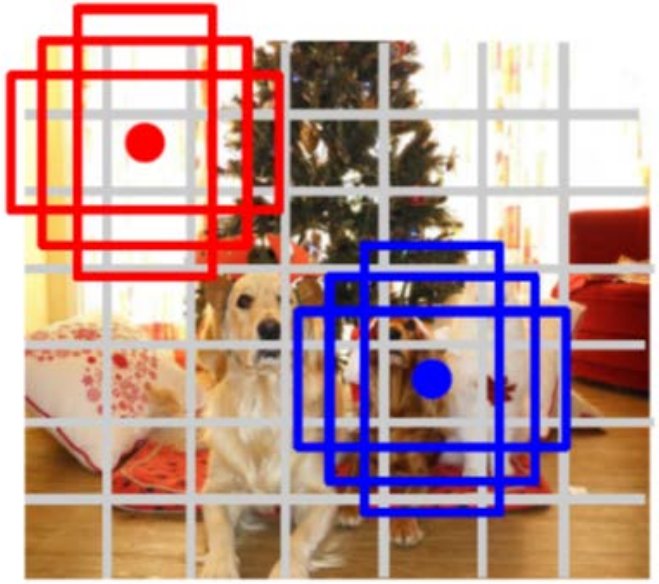
One stage Detector



Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutions network !

Input image
3*H*W



Divide image into grid
7*7

Image a set of **base boxes**
Centered at each grid cell
Here B = 3

Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes(including background as a class)

Output:
7*7*(5*B+C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

SSD (Single Shot Detection)

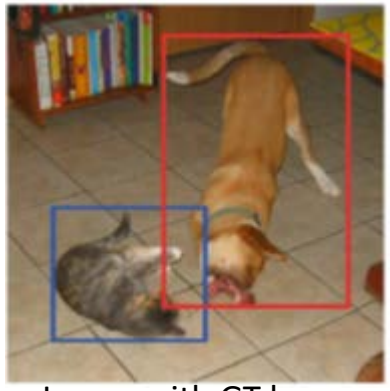
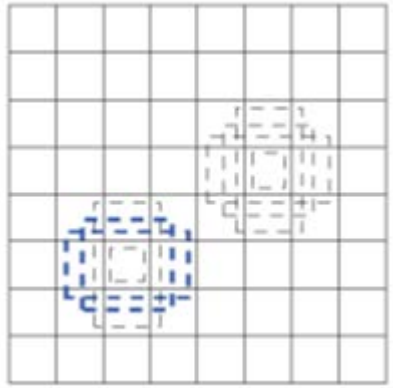
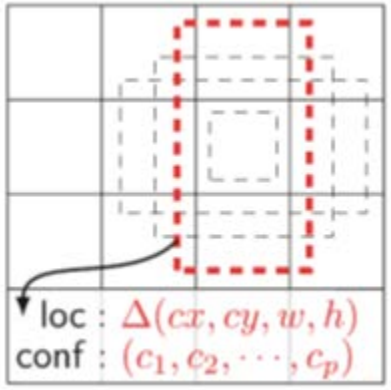


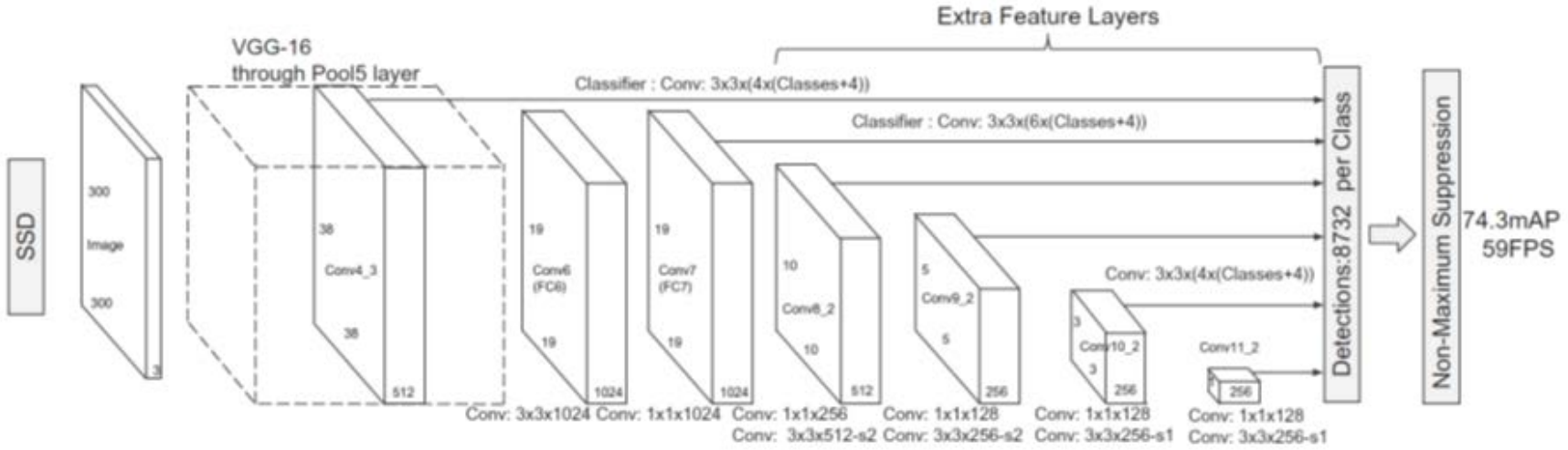
Image with GT boxes

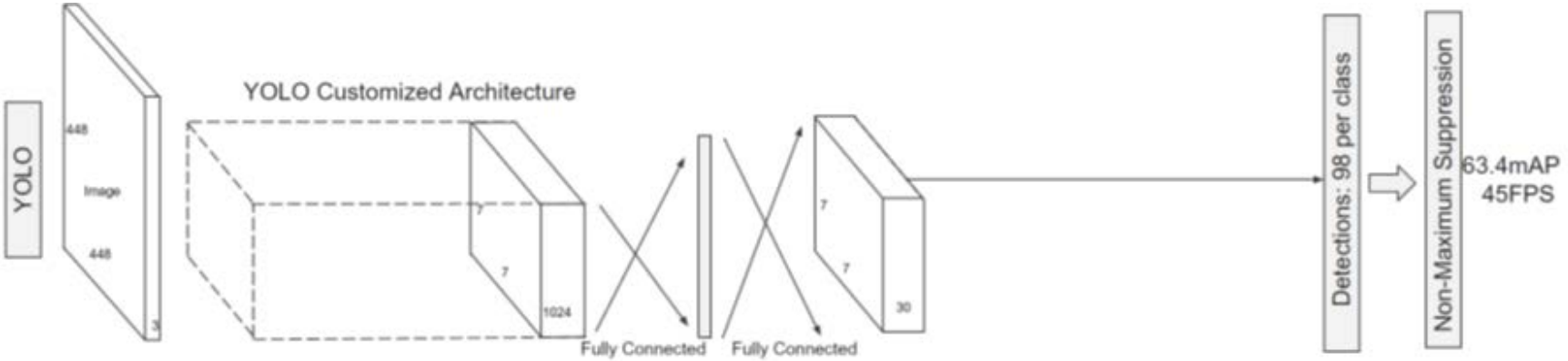


8*8 feature map



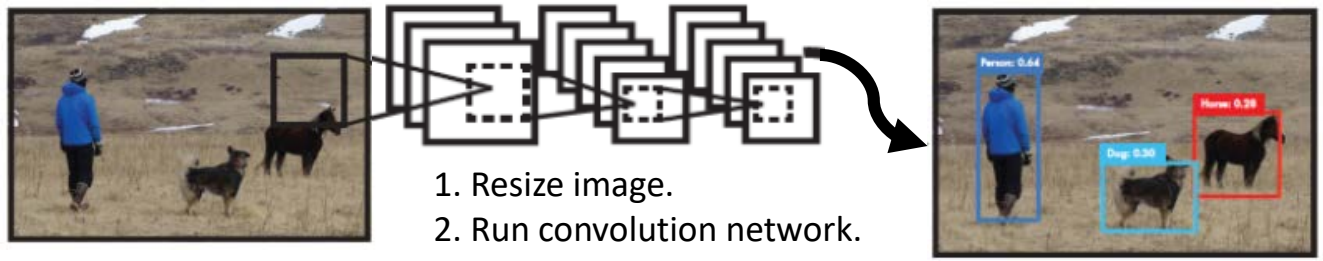
4*4 feature map

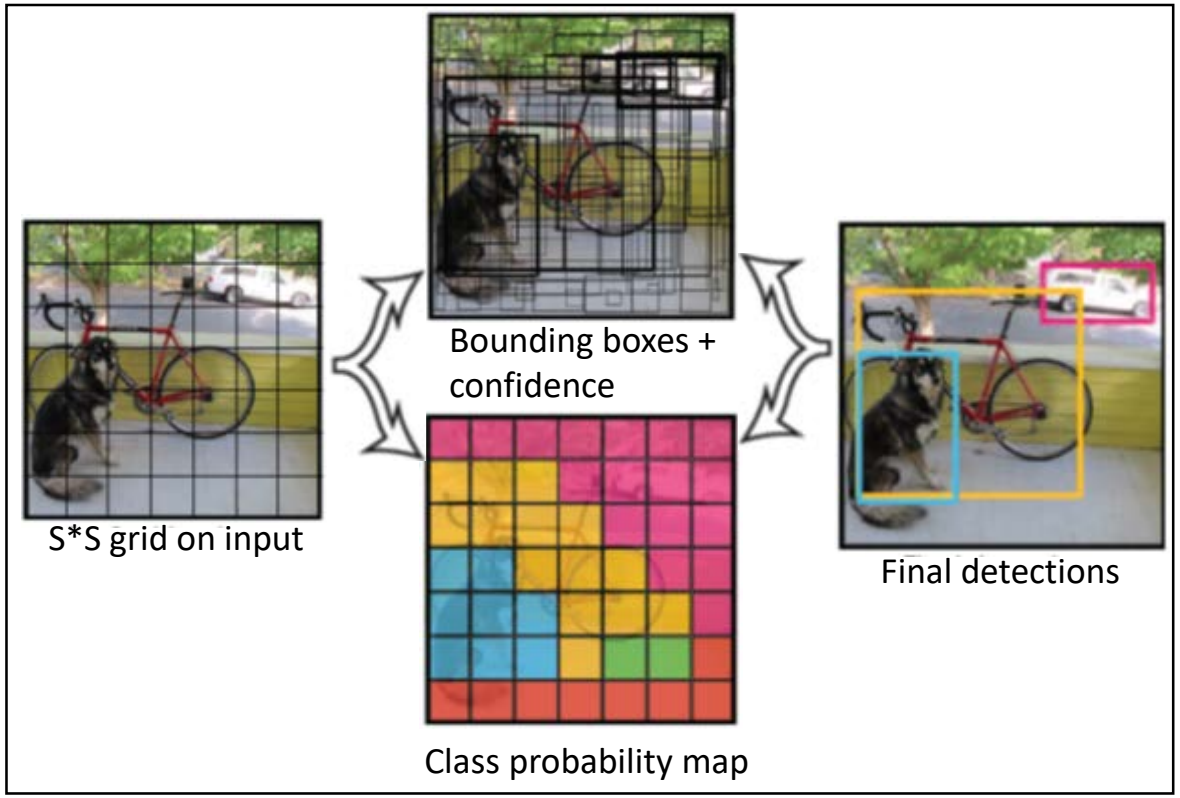


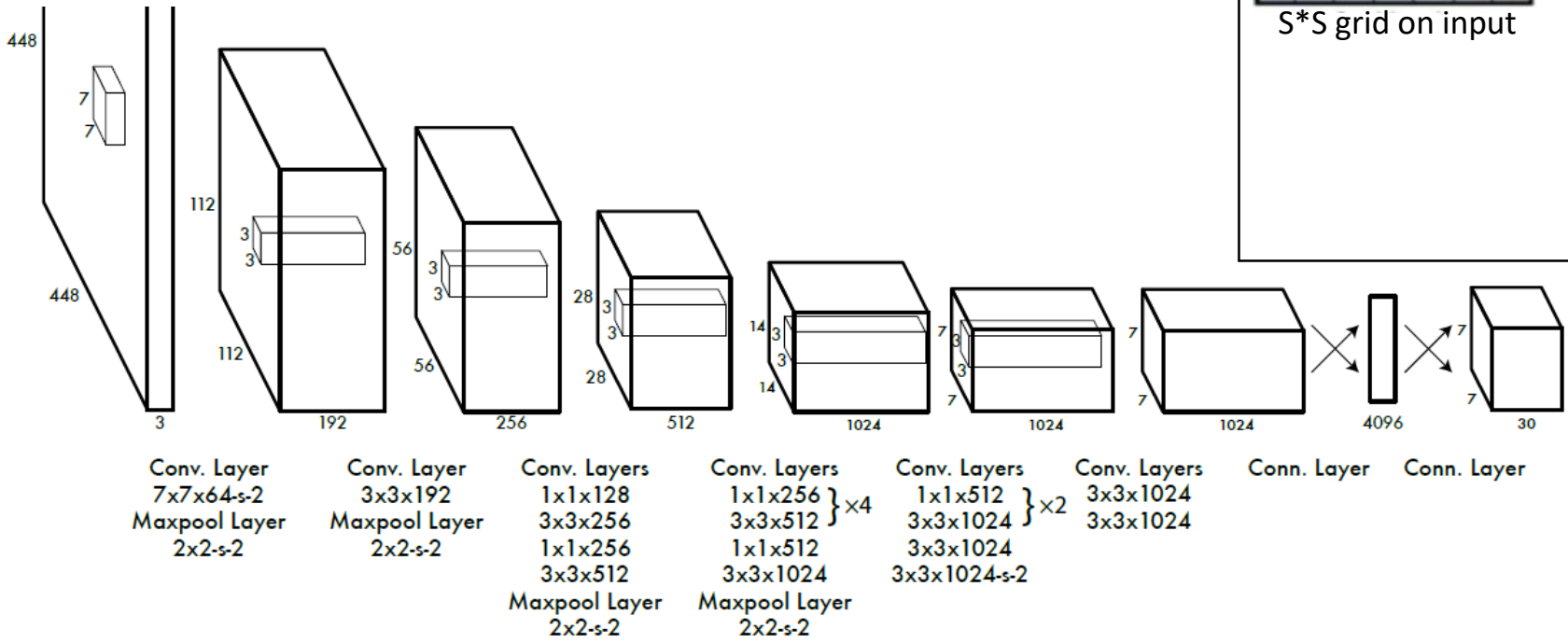


2016_SSD, Single Shot MultiBox Detector (ECCV)

YOLO (You Only Look Once)







2016_You Only Look Once, Unified, Real-Time Object Detection (CVPR)

YOLO

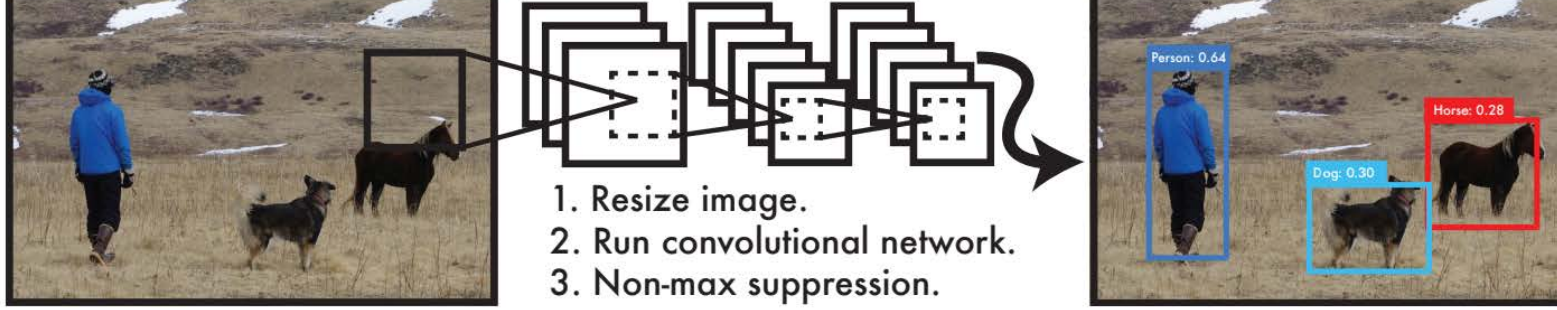
Divide Image into **S x S** grids

In each rid cell

→B: BBoxes and Confidence score

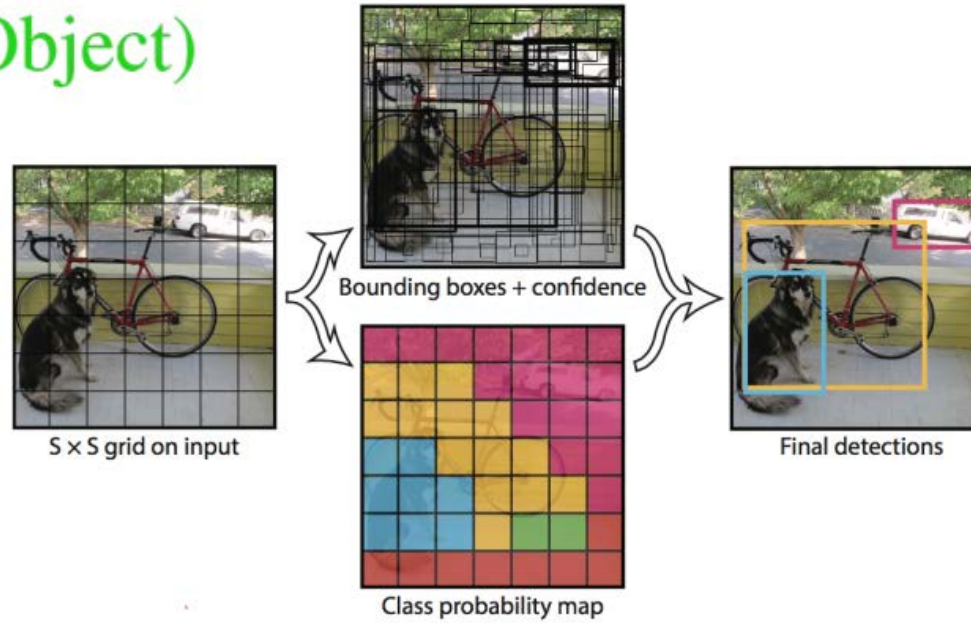
→ C: class probabilities w.r.t #classes

→ loss function



x, y, w, h, confidence → $\Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}}$

$\Pr(\text{Class}_i \mid \text{Object})$



$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

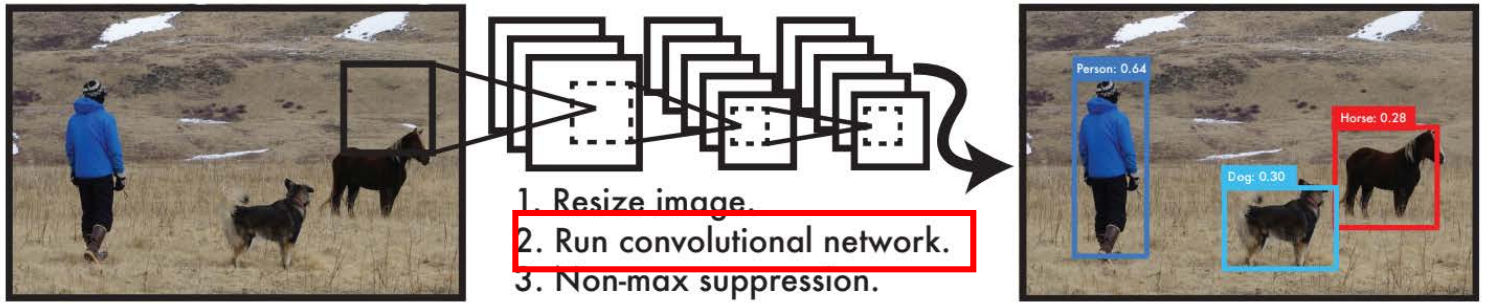
SxS grids

B bounding boxes for each grid

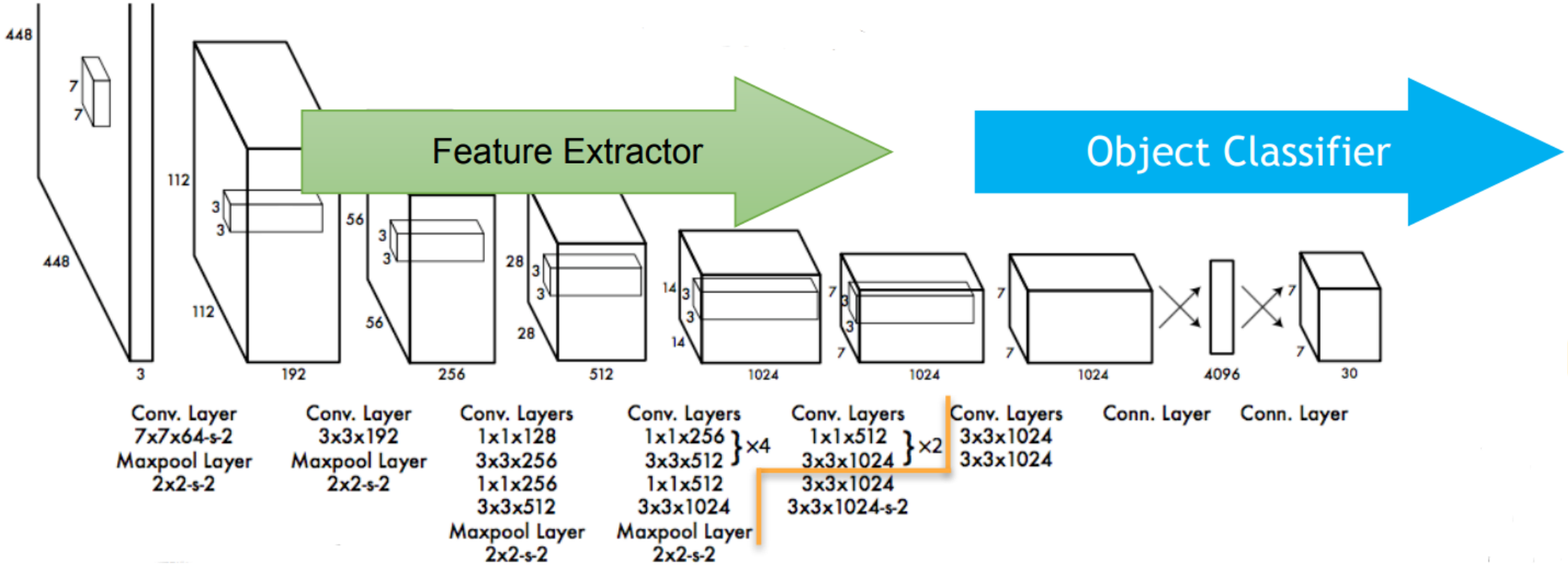
C class probabilities for each grid

=> an S*S*(5*B+C) tensor

YOLO

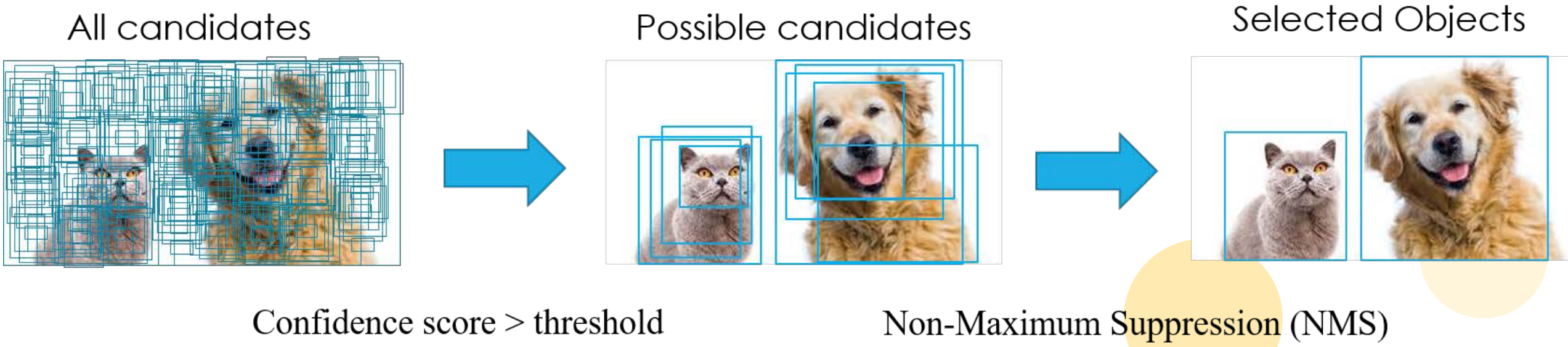


- Network Design
  - Modified GoogLeNet
  - 1x1 reduction layer ("Network in Network")
  - Pretrain with ImageNet 1000-class competition dataset for first 20 conv. layers
  - for detection, add 4 Conv layers + 2 FC layers
    - a total of 24 Conv layers + 2 FC layers

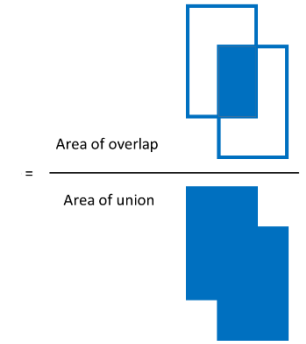


Non-Maximum Suppression (NMS)

- each bonding box (BBox) contains a confidence score for foreground objects
- confidence score threshold removes background BBox
- NMS remove redundant bounding boxes covering partial foreground



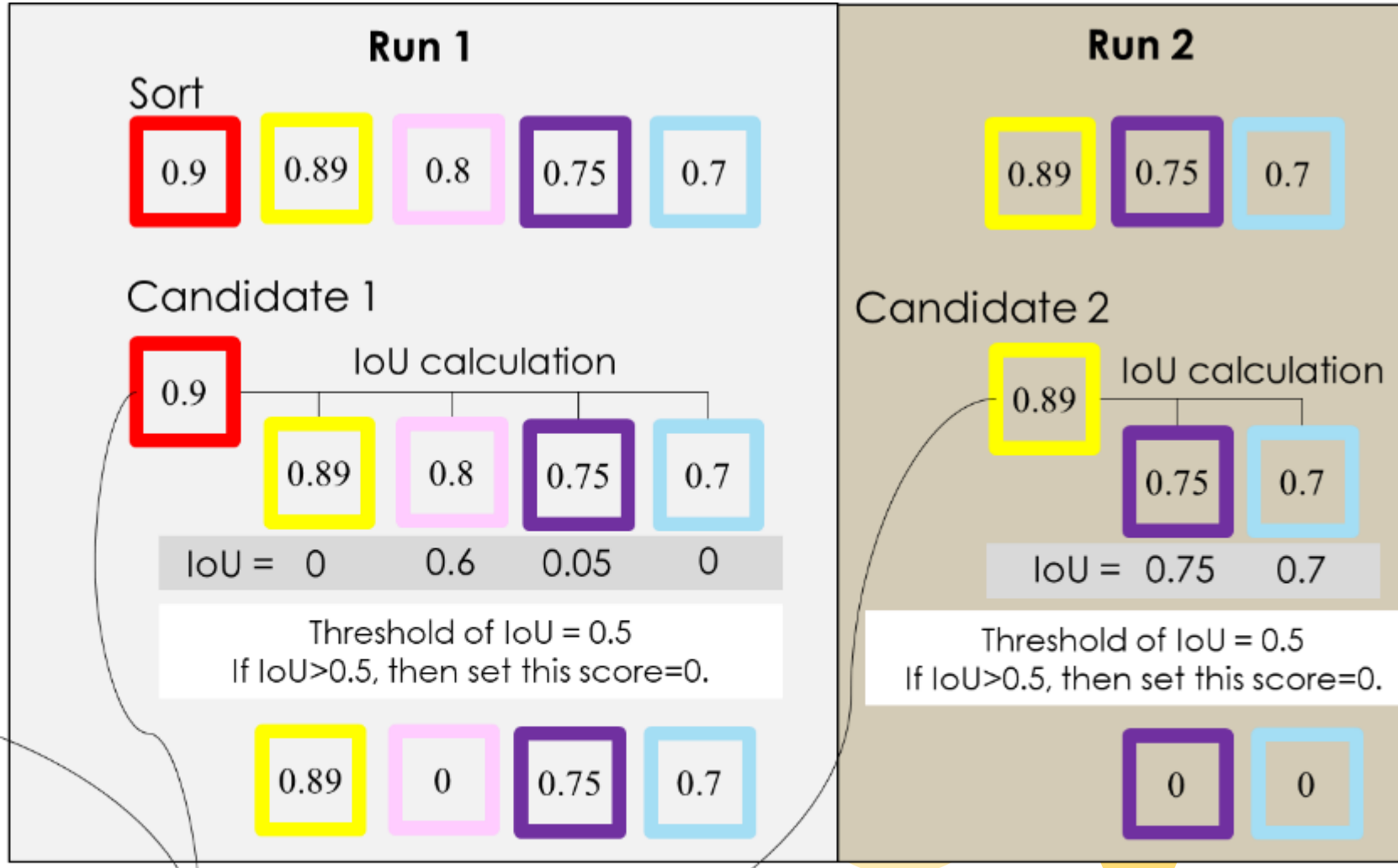
NMS Example 1 (IOU threshold=0.5)



- NMS with IOU threshold=0.5 removes all redundant BBox

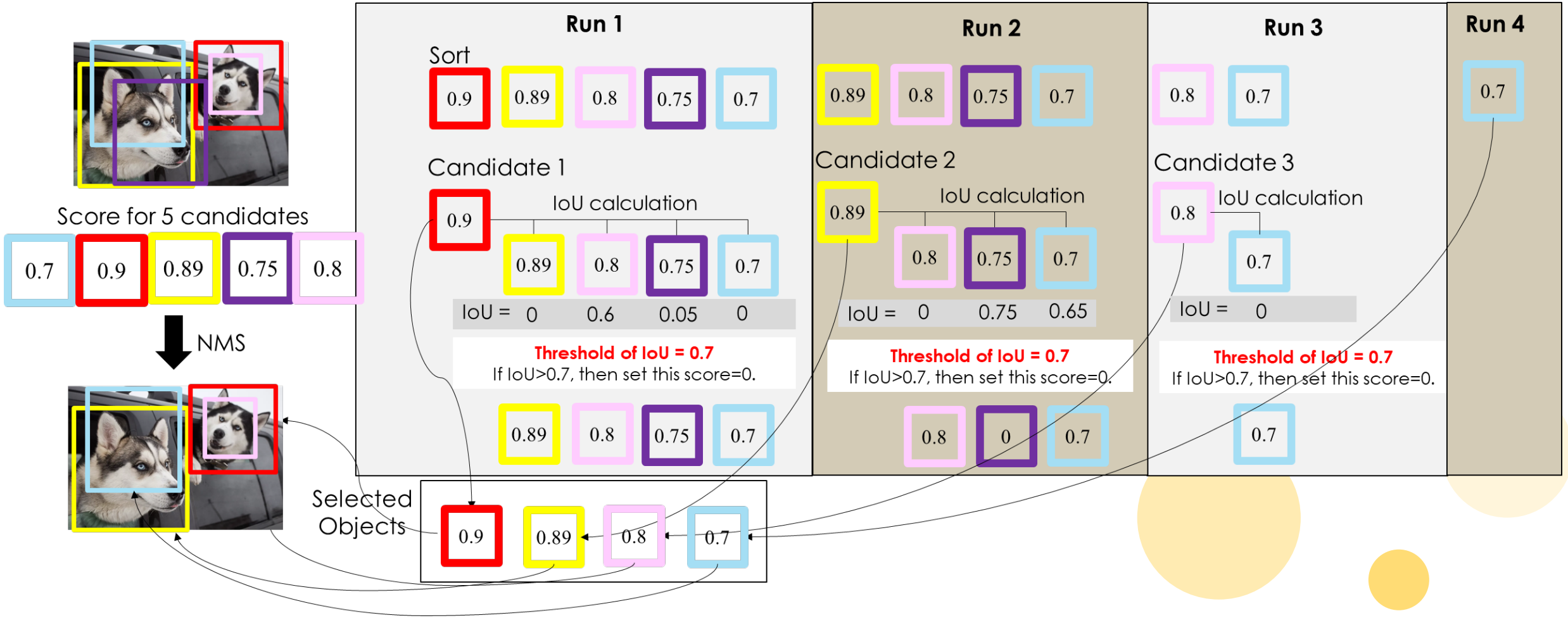
Score for 5 candidates

NMS



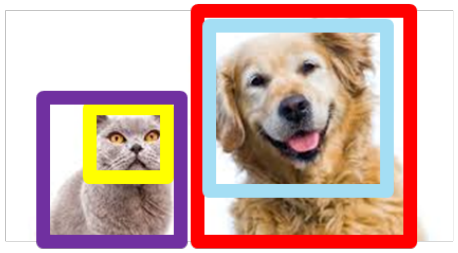
NMS Example 2 (IOU threshold=0.7

- too large IOU threshold results in more duplicate BBoxes
  - NMS with IOU threshold=0.7 only removes purple BBox

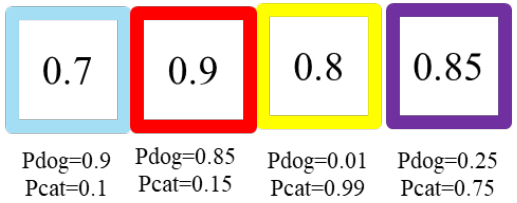


NMS Example 3 (IOU threshold=0.5)

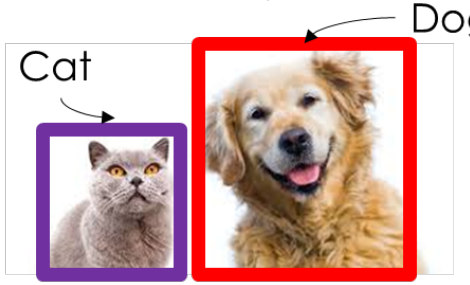
- different object classes in NMS
  - class probability for each class object

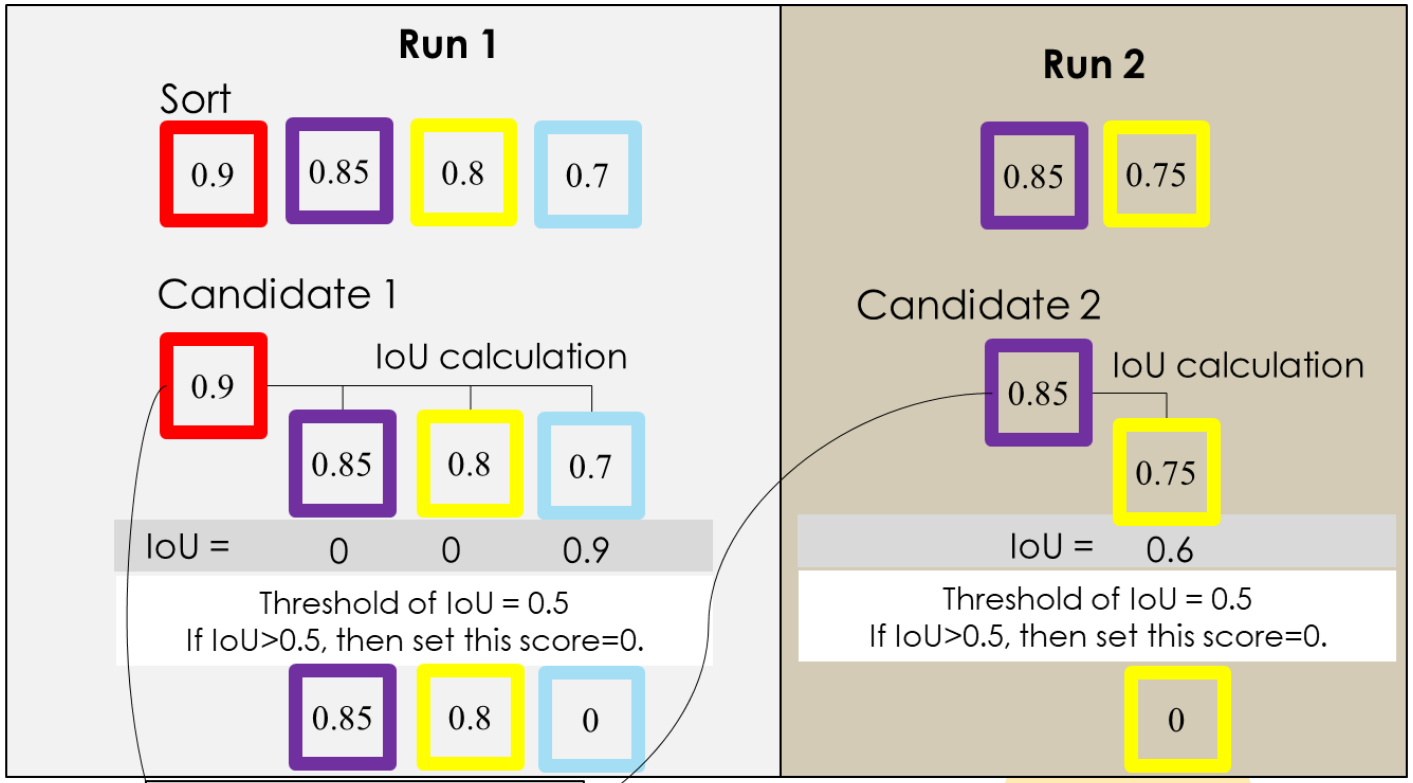


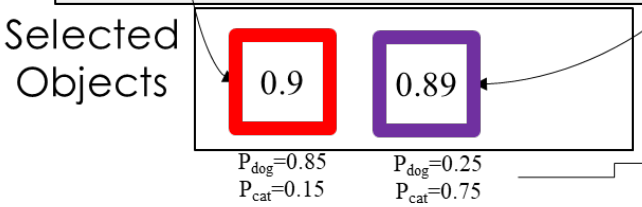
Score for 4 candidates



NMS







There are no objects, so NMS is done.

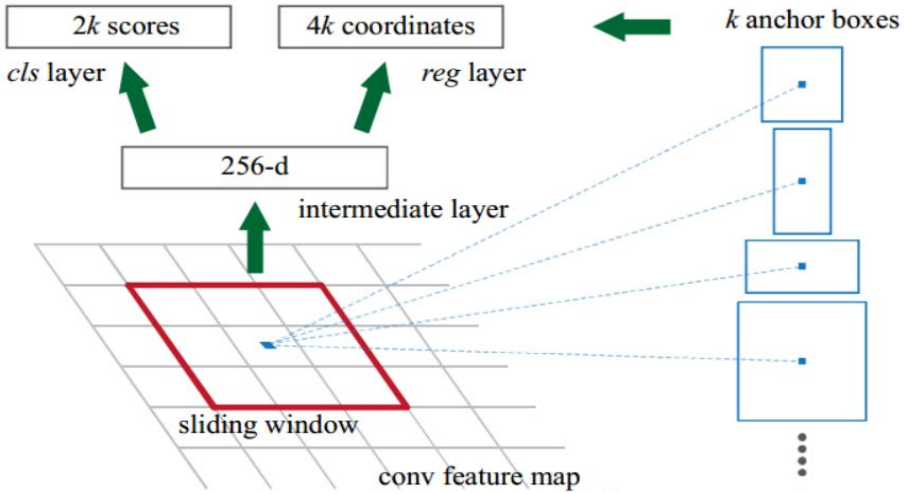
YOLOv2

CNN(darknet-19)

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

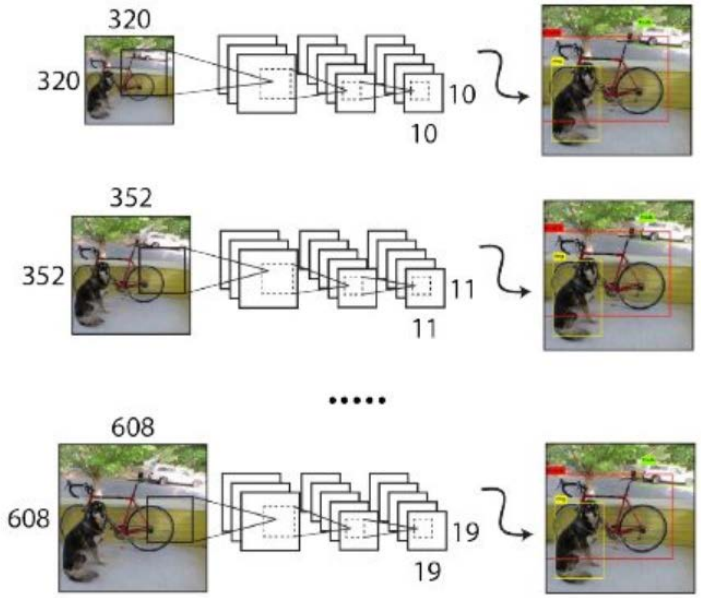
- **With Anchor Boxes :**

  引入Faster R-CNN中的anchor思維，並使用K-means聚類方法訓練，自動找到更好的boxes寬高維度，提高精準度。



- **Multi-Scale Training:**

  Average pooling Layer 使YOLOv2可以適應不同大小的輸入圖片。



Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

YOLO v2

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

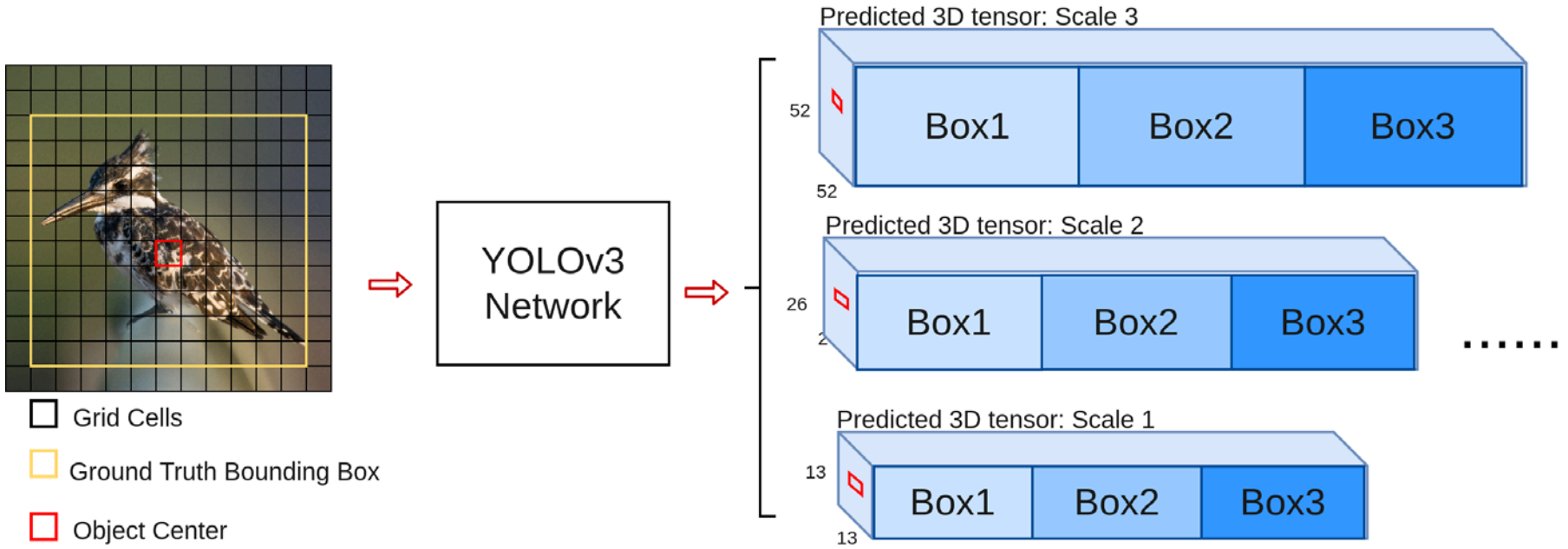
Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Darknet-19

2017_YOLO9000, Better, Faster, Stronger (CVPR)

YOLOv3

- DarkNet-53 cp. DarkNet-19 (YOLOv2) and GoogleNet (YOLOv1)
- different scales



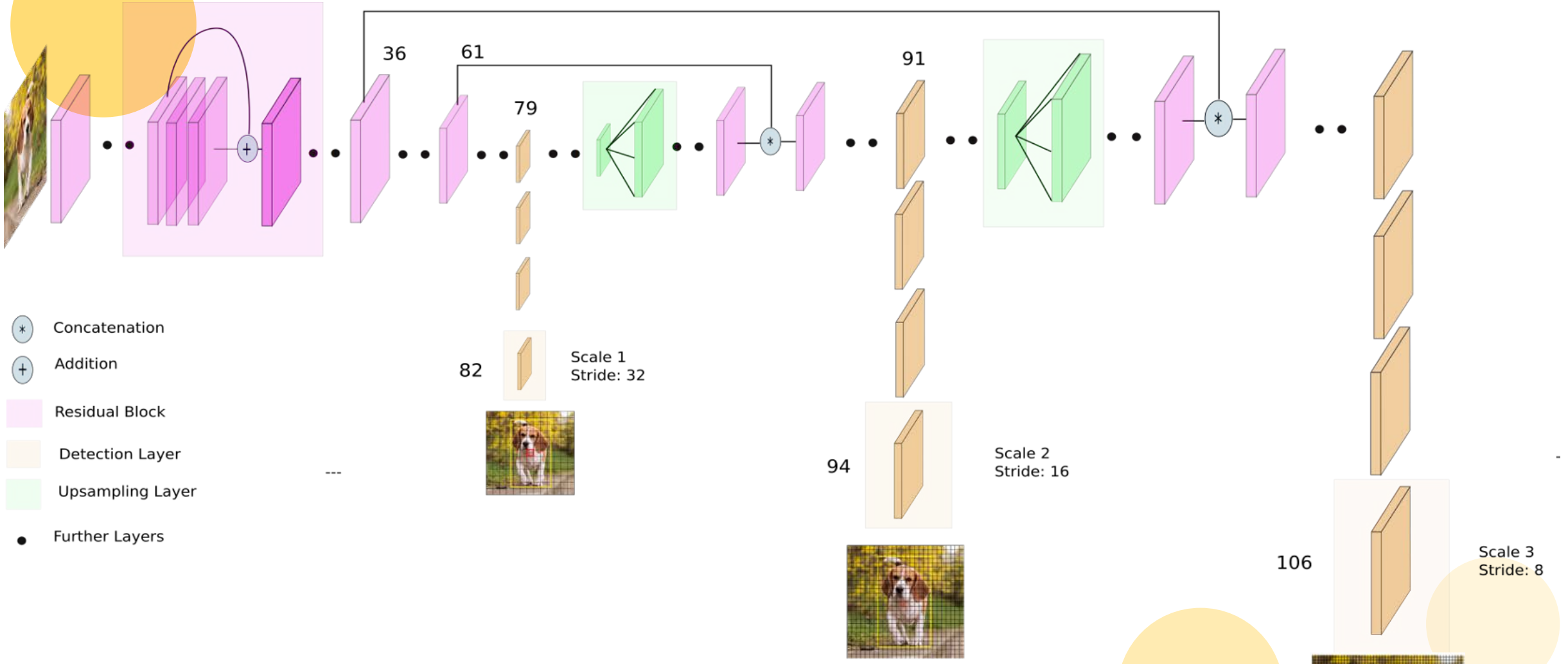
3D tensor Dimension:

N x N x [3 x (4 + 1 + 80)]

NxN: Scale size (i.e. 13x13)
3: # Boxes: (each grid predict 3 boxes)
4: Box Cordinate: (tx, ty, tw, th)
1: Objectness score: how likely it is an object
80: # classes

3 different scales

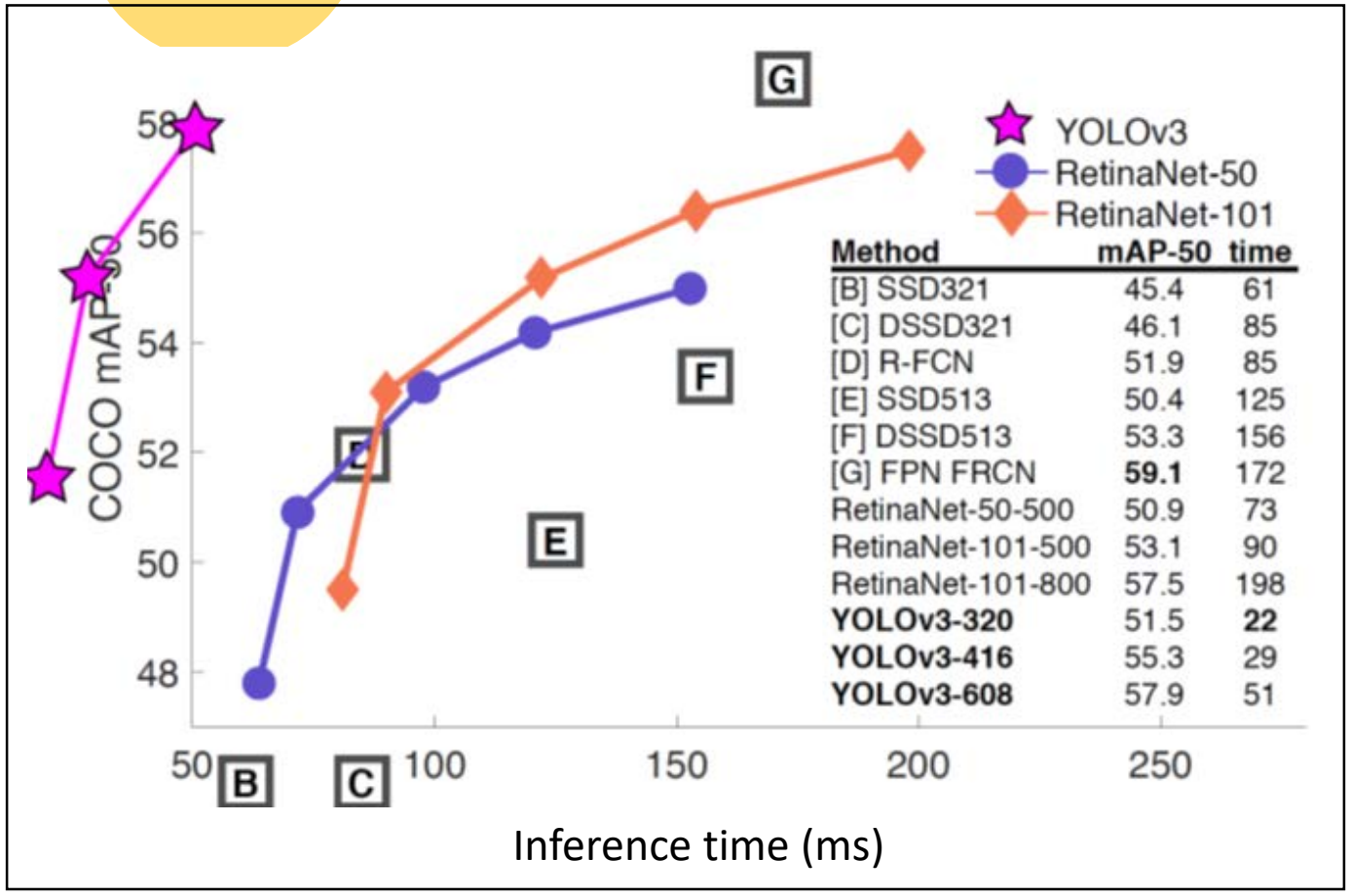
YOLOv3



Darknet-53

YOLOv3架構產出3種不同尺度(scale)，越精細的單元網格(grid cell) 可檢測出越精細的物體

YOLO v3



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

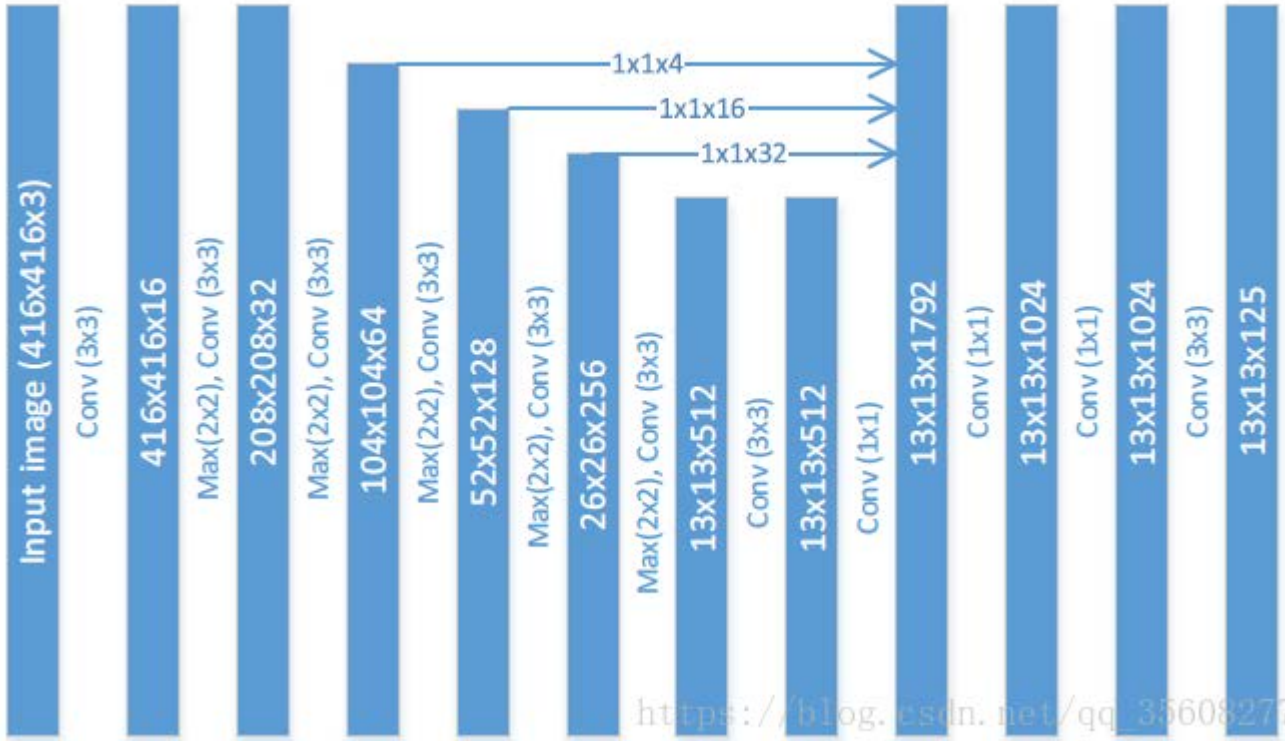
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

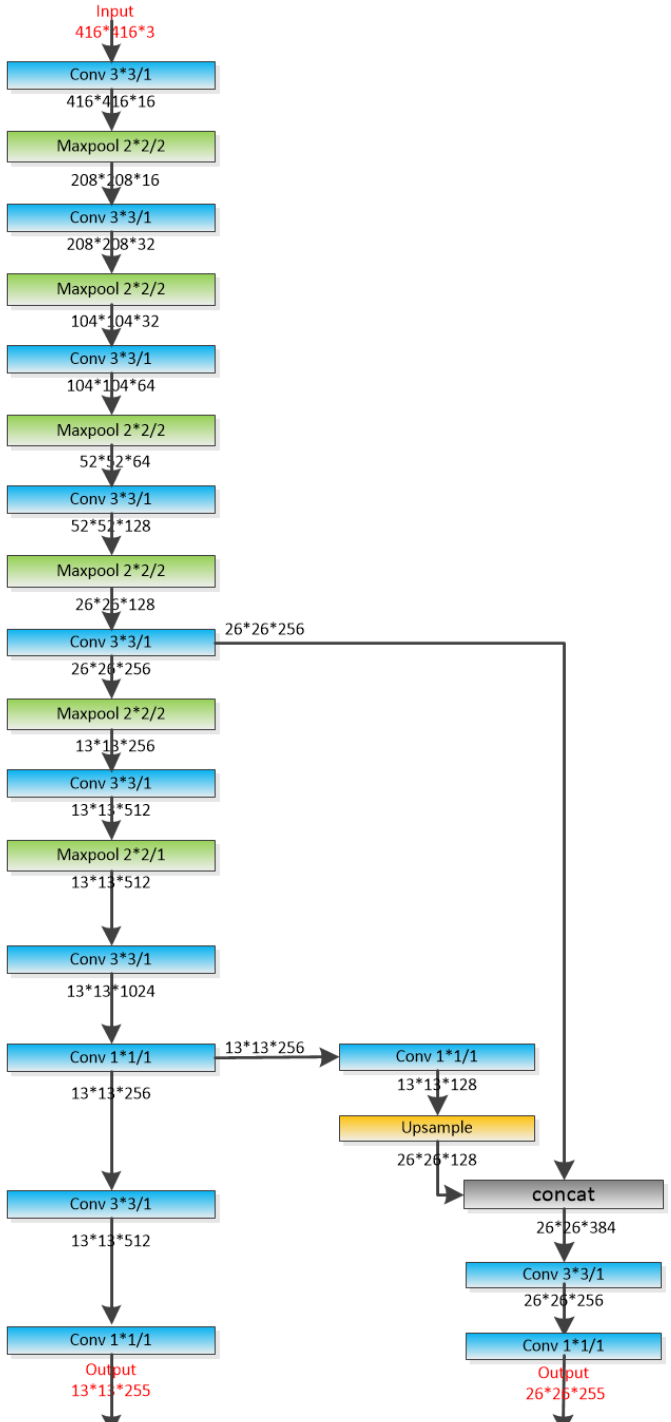
Darknet-53

2018_YOLOv3: An Incremental Improvement

Tiny YOLOv3

- around 10 layers (instead of 53 in YOLOv3)





Object Detection: Lots of variables…

Base Network :
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

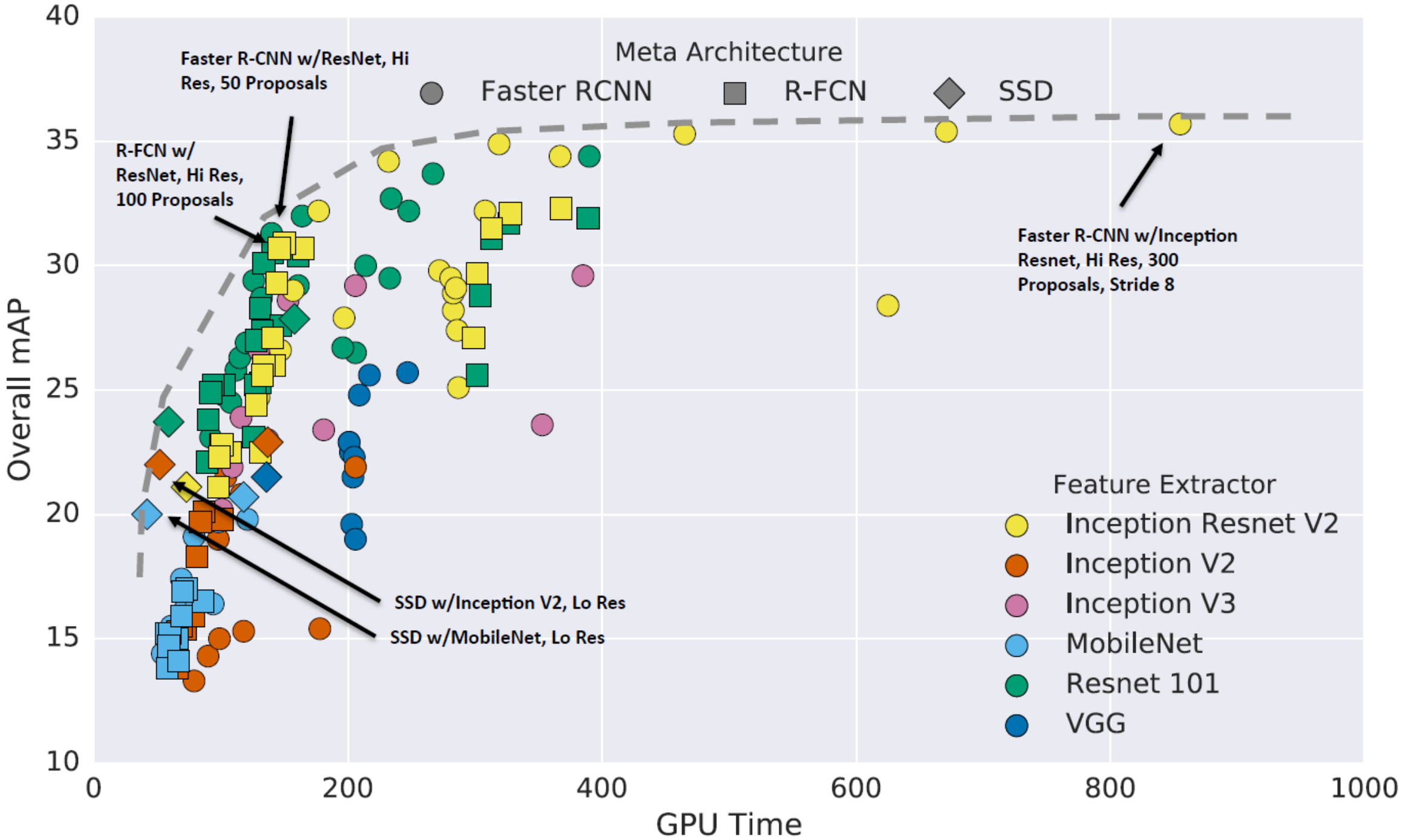
Object Detection architecture :
Faster R-CNN
R-FCN
SSD
YOLO

Image Size
Region Proposals
…

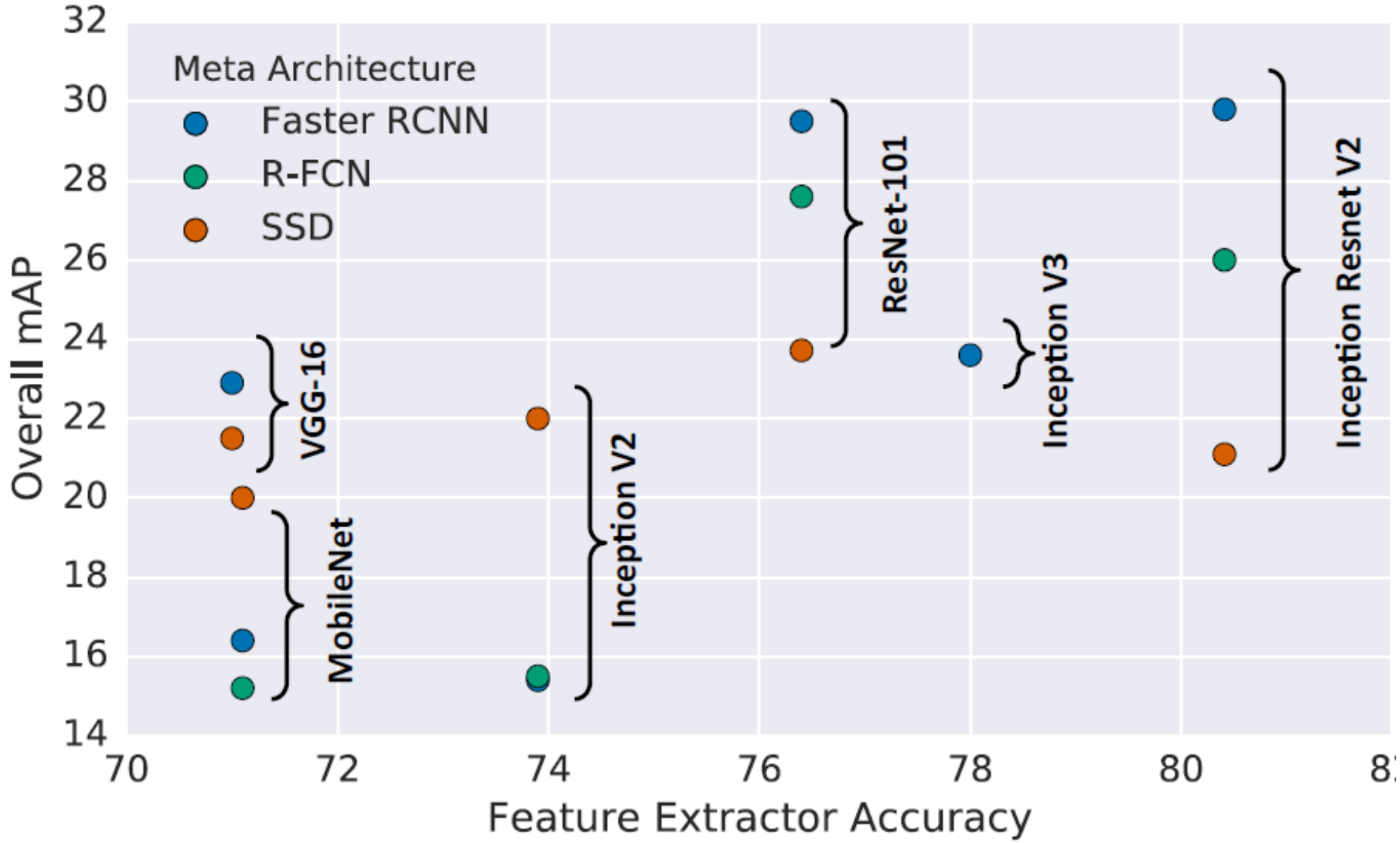
Comments :
Faster R-CNN is slower but more accurate

YOLO is much faster but not as accurate

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017
R-FCN: Dai et al, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", NIPS 2016
Inception-V2: Ioffe and Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015
Inception V3: Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", arXiv 2016
Inception ResNet: Szegedy et al, "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv 2016
MobileNet: Howard et al, "Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv 2017


Meta Architecture
Faster RCNN
R-FCN
SSD
Faster R-CNN w/ResNet, Hi Res, 50 Proposals
R-FCN w/ ResNet, Hi Res, 100 Proposals
Faster R-CNN w/Inception Resnet, Hi Res, 300 Proposals, Stride 8
SSD w/Inception V2, Lo Res
SSD w/MobileNet, Lo Res
Feature Extractor
Inception Resnet V2
Inception V2
Inception V3
MobileNet
Resnet 101
VGG
Overall mAP
GPU Time
40
35
30
25
20
15
10
0
200
400
600
800
1000

mAP (COCO) vs. Top-1 Accuracy



Object Detection: Impact of Deep Learning

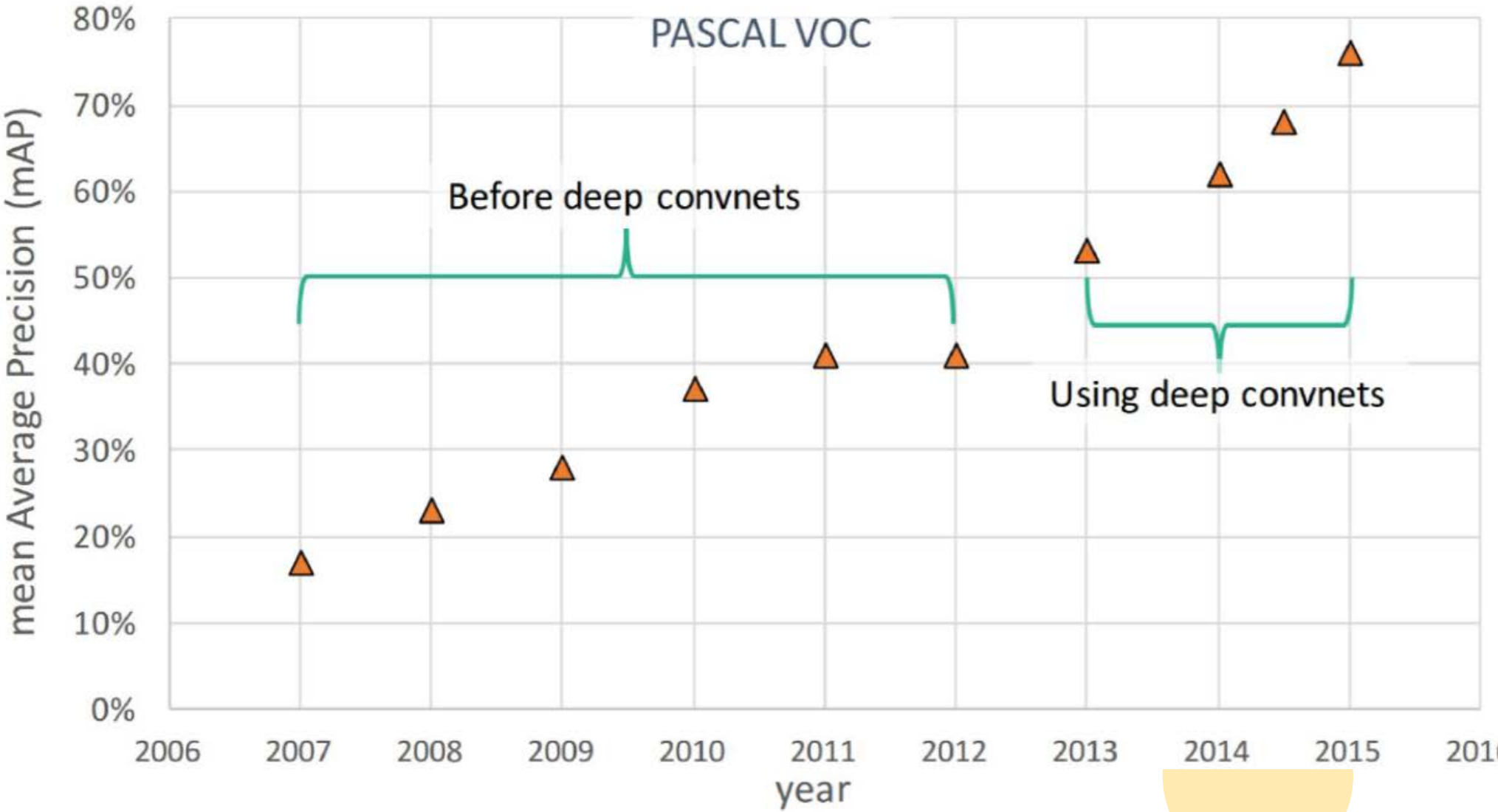
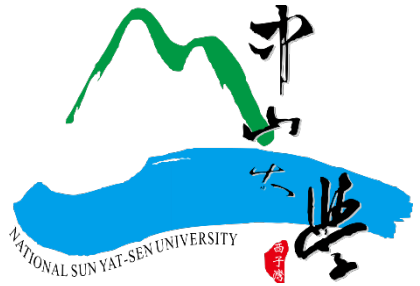
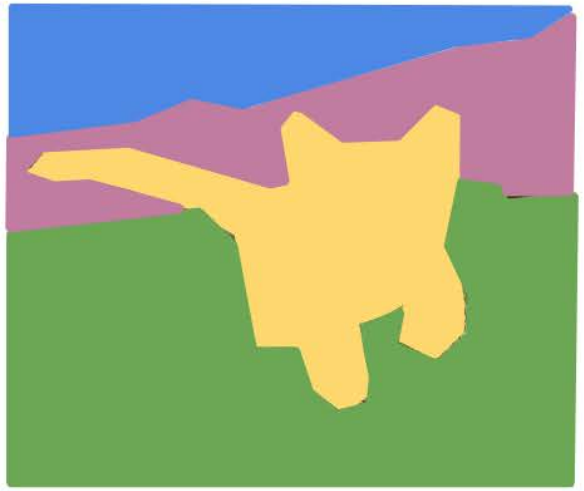


Image Segmentation



classification/detection/segmentation

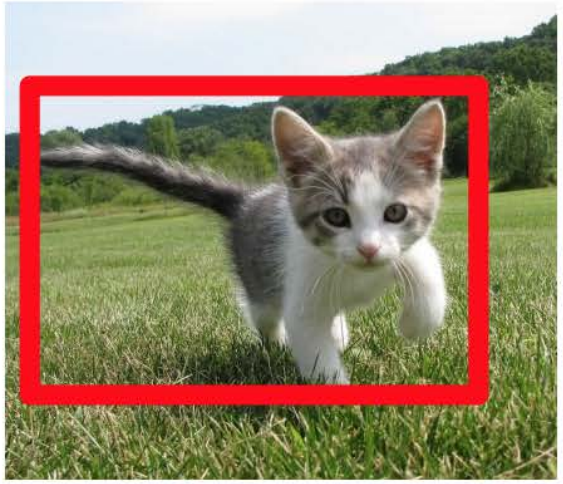
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Classification + Localization



CAT

Single Object

Object Detection

DOG, DOG, CAT

Instance Segmentation

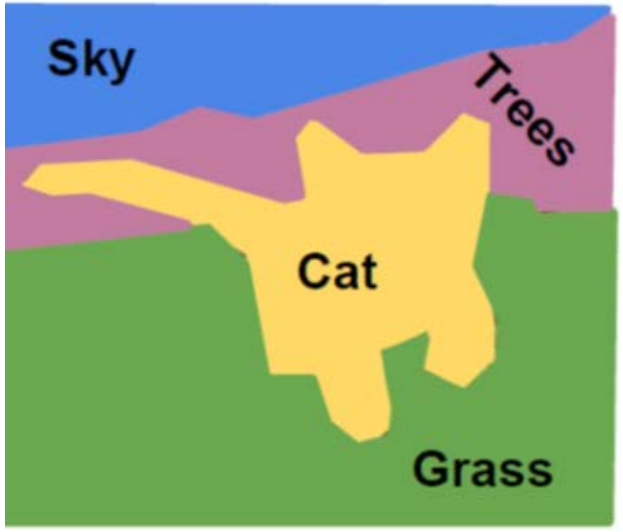
DOG, DOG, CAT

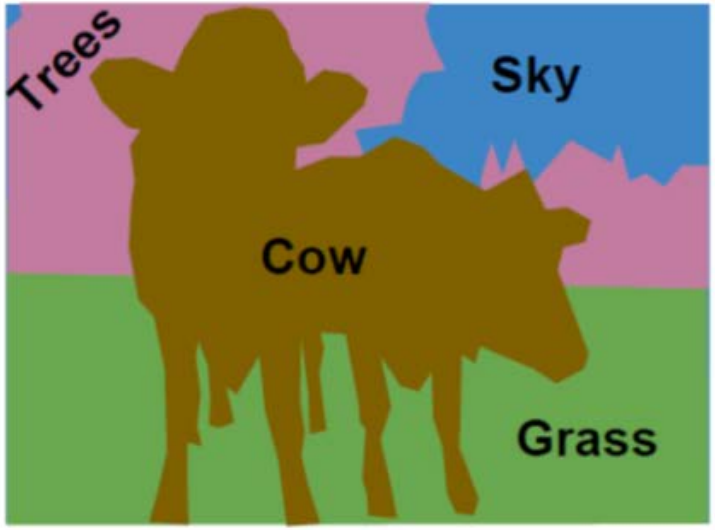
Multiple Object

Semantic Segmentation

Label each pixel in the image with a category label

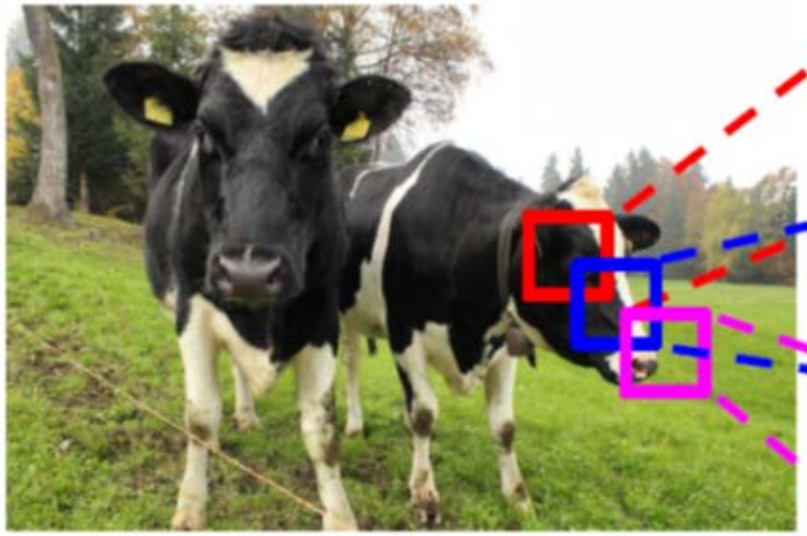
Don’t differentiate instances, only care about pixels





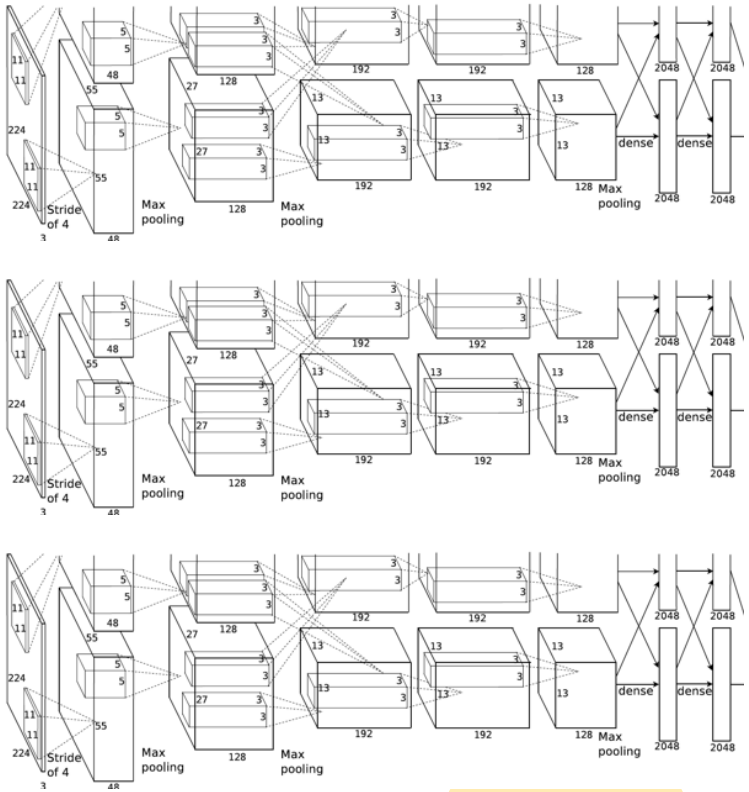
Semantic Segmentation Idea: Sliding Window

Full image



Extract patch

Classify center pixel with CNN



Cow

Cow

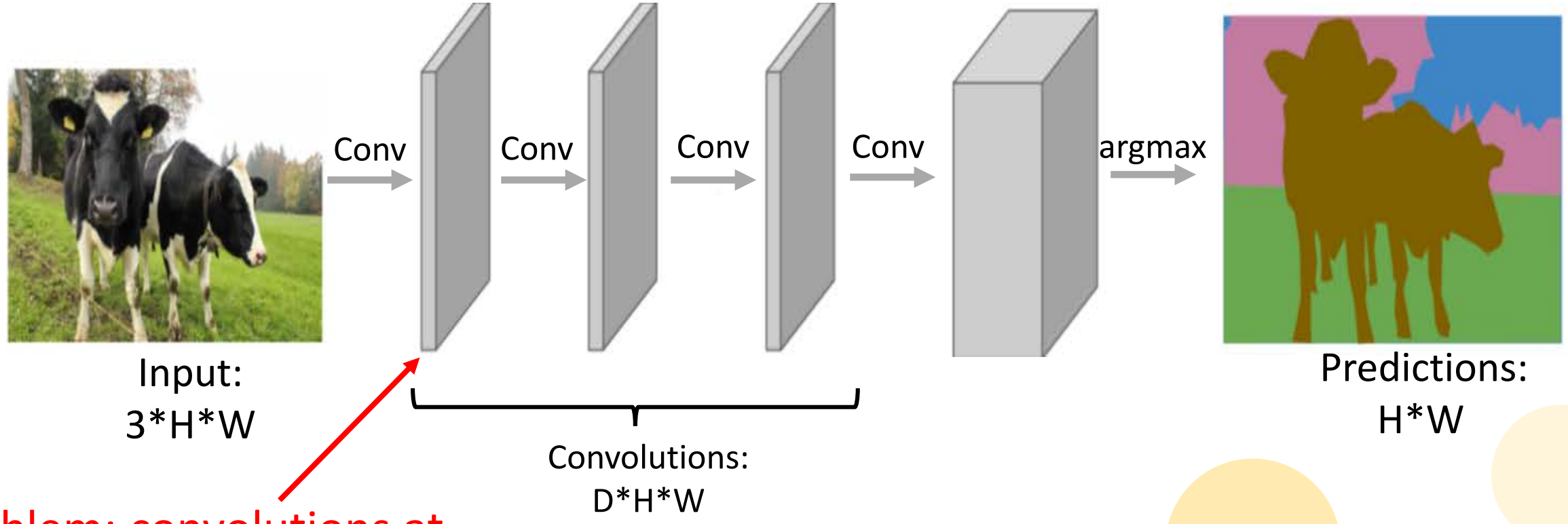
Grass

Problem: Very inefficient!
Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Fully Convolution

Design a network as a bunch of convolution layers to make predictions for pixels all at once!



Problem: convolutions at original image resolution will be very expensive …

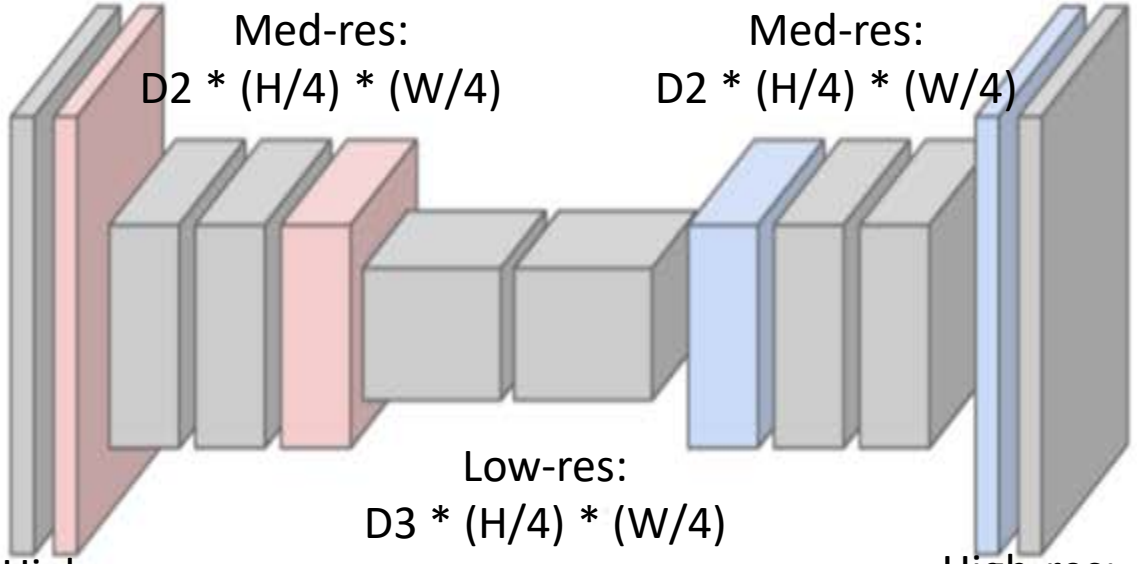
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with downsampling and unsampling inside the network !

Unsampling:
Unpooling or strided transpose convolution

Input:
3*H*W

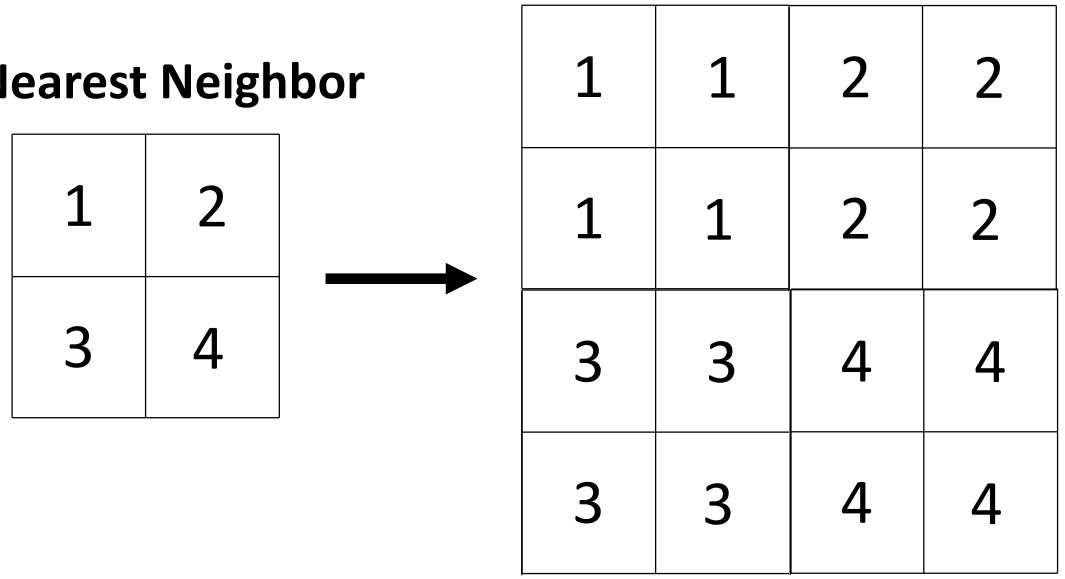


Predictions:
H*W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

In-Network upsampling: “Unpooling”

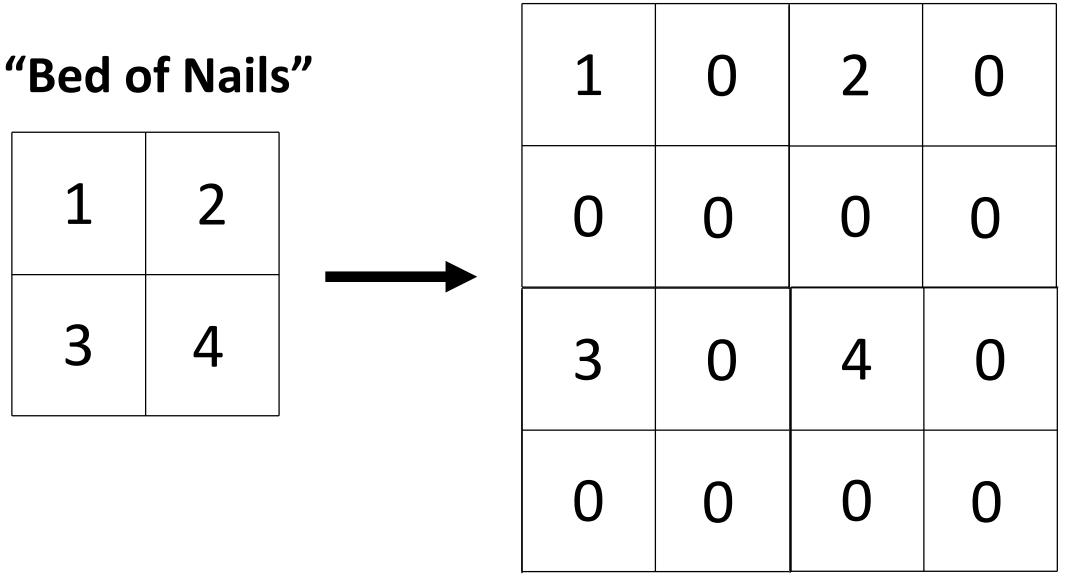
Nearest Neighbor



Input: 2 * 2

Output: 4 * 4

“Bed of Nails”



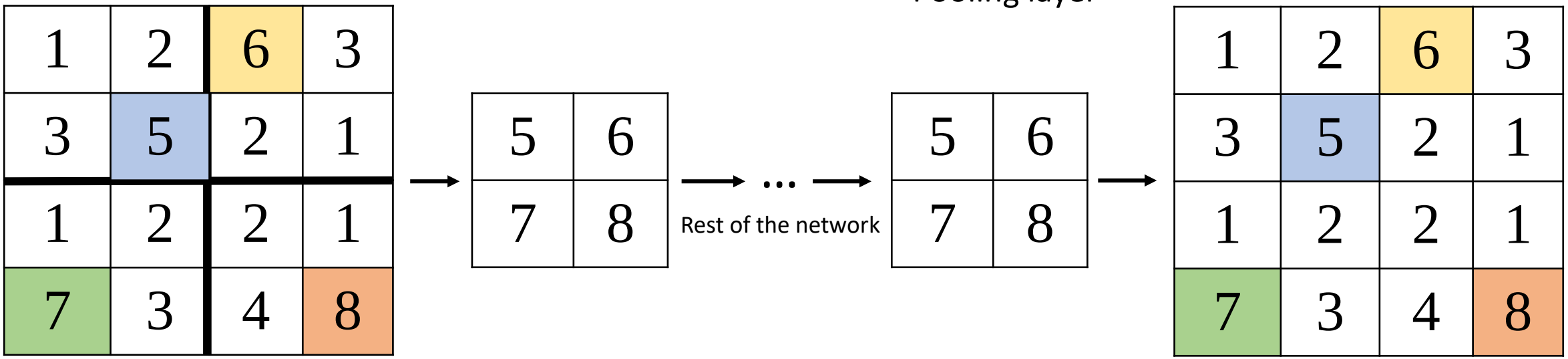
Input: 2 * 2

Output: 4 * 4

In-Network upsampling: “Max Unpooling”

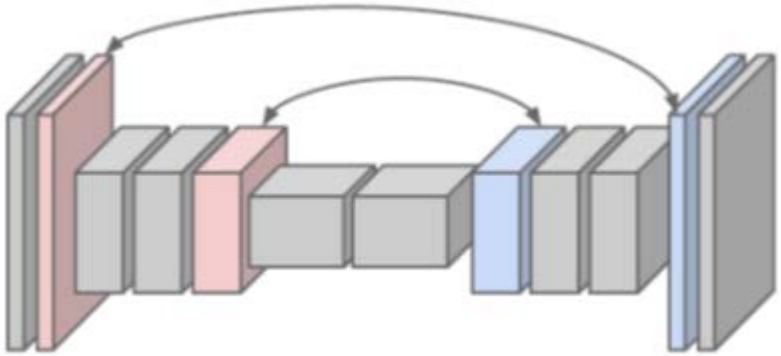
Max Pooling
Remember which element was max !

Max UnPooling
Use positions from
Pooling layer



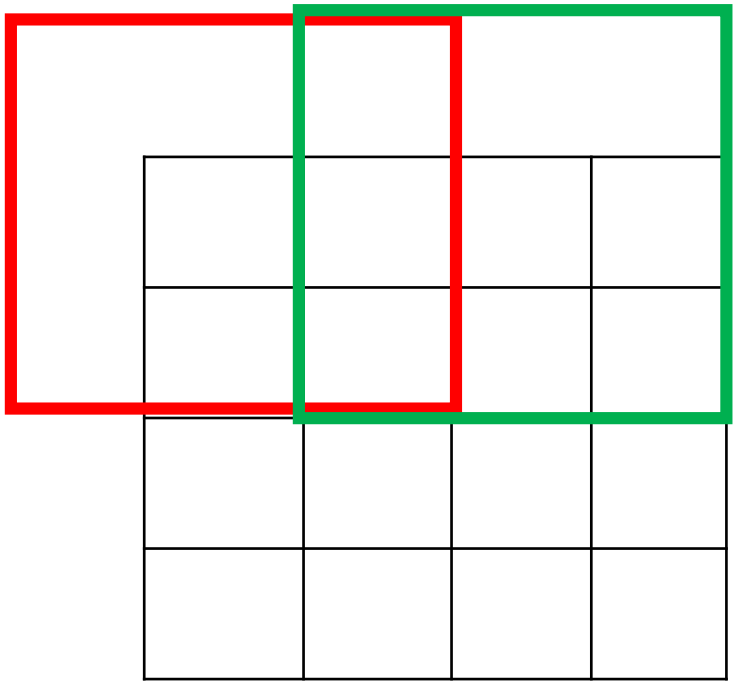
Input: 4 * 4 Output: 2 * 2 Input: 2 * 2 Output: 4 * 4

Corresponding pairs of
downsampling and
unsampling layers



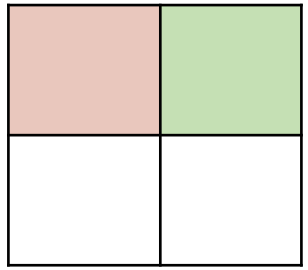
3x3 Convolution, Stride=2, padding=1

Recall: Normal 3*3 convolution, stride 2 pad 1



Input: 4 * 4

Dot product between filter and input



Output: 2 * 2

Filter moves 2 pixels in the **input** for every one pixel int the **output**

Stride gives ratio between movement in input and output

3x3 Transpose Convolution (DeConvolution)

3 * 3 transpose convolution, stride 2 & pad 1

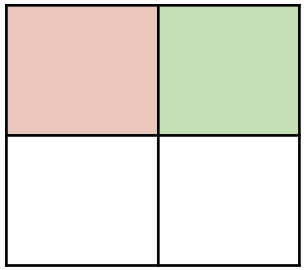
Other names:

-Deconvolution

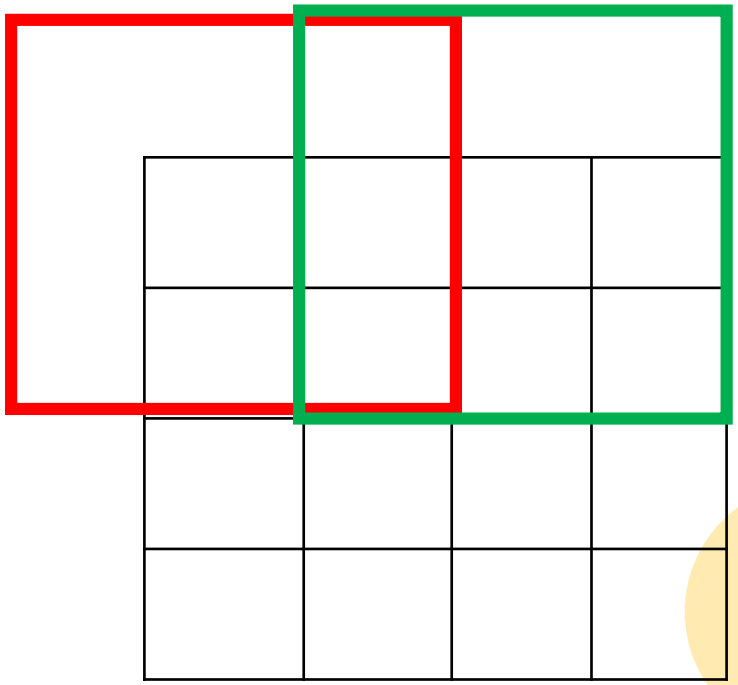
-Upconvolution

-Fractionally strided convolution

-Backward strided convolution



Input gives weight for filter



Input: 2 * 2

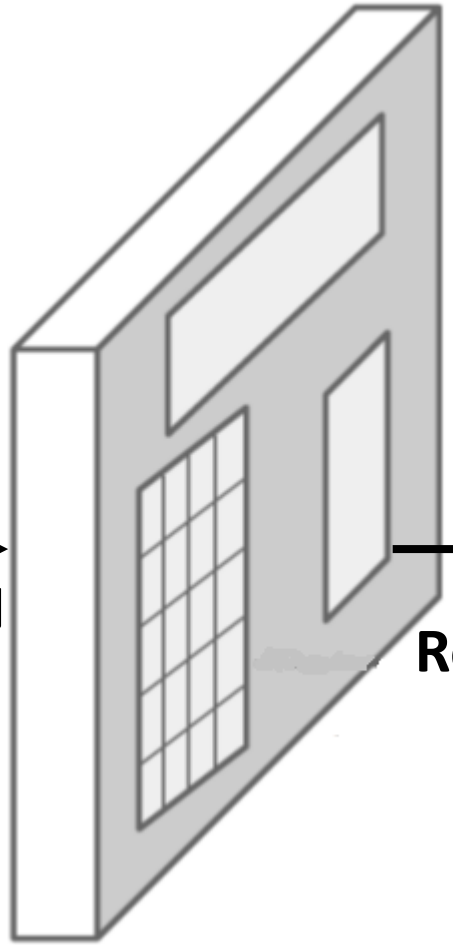
Output: 4 * 4

Filter moves 2 pixels in the **output** for every one pixel int the **input**

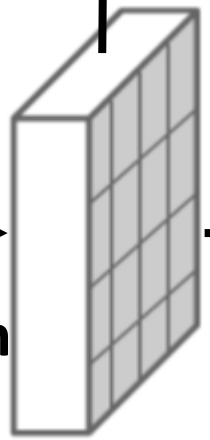
Stride gives ratio between movement in output and input

Mask R-CNN (detection + segmentation)

CNN

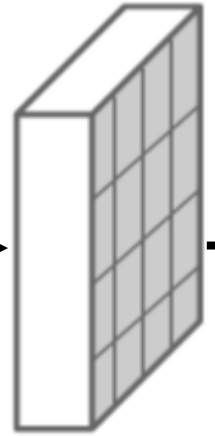


Rol Align



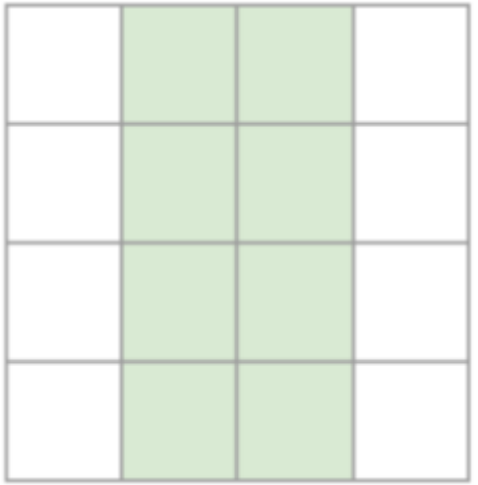
256*14*14

Conv



256*14*14

Conv



Predict a mask for each of C classes

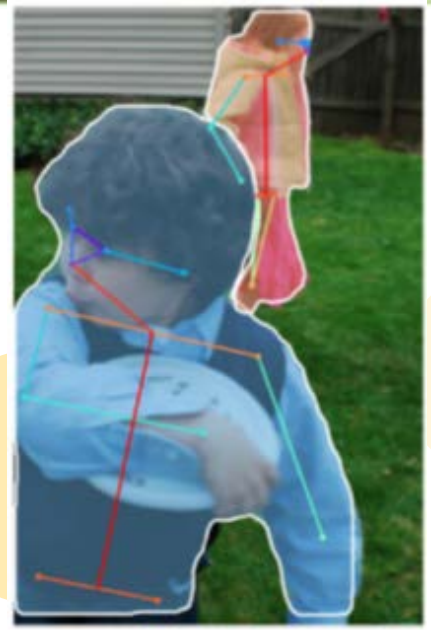
C*14*14

Classification Scores: C

Box coordinates (per class): 4*C

*K. He, et al., “Mask R-CNN,” ICCV, 2017.

Mask R-CNN Results

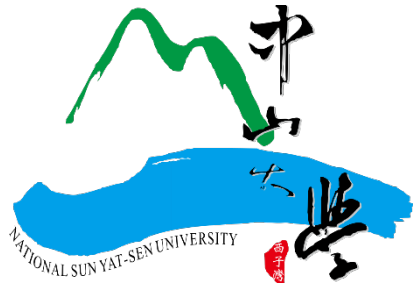


Mask R-CNN

- on COCO dataset

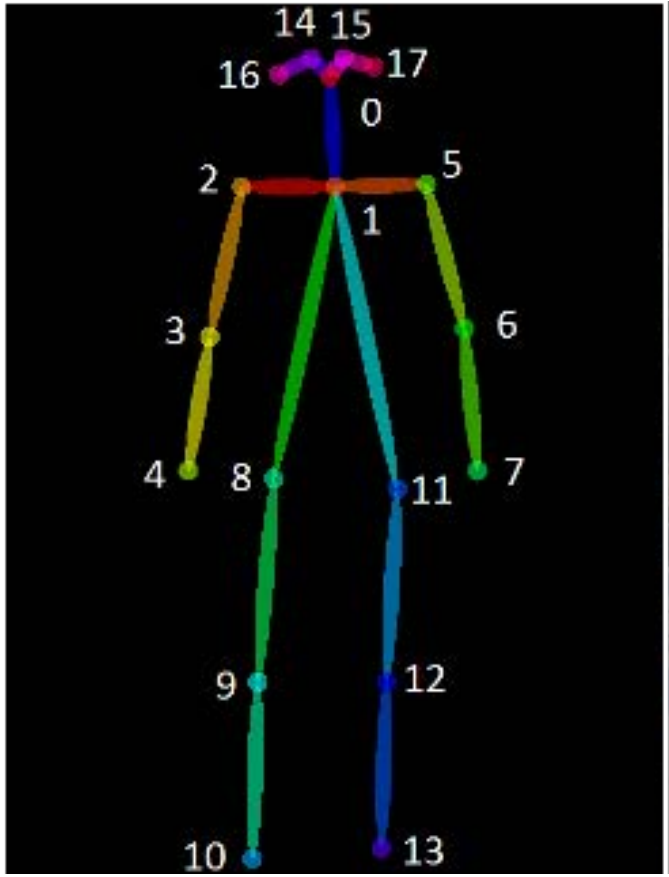
	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [22]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [32]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [31]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

Human Pose Estimation



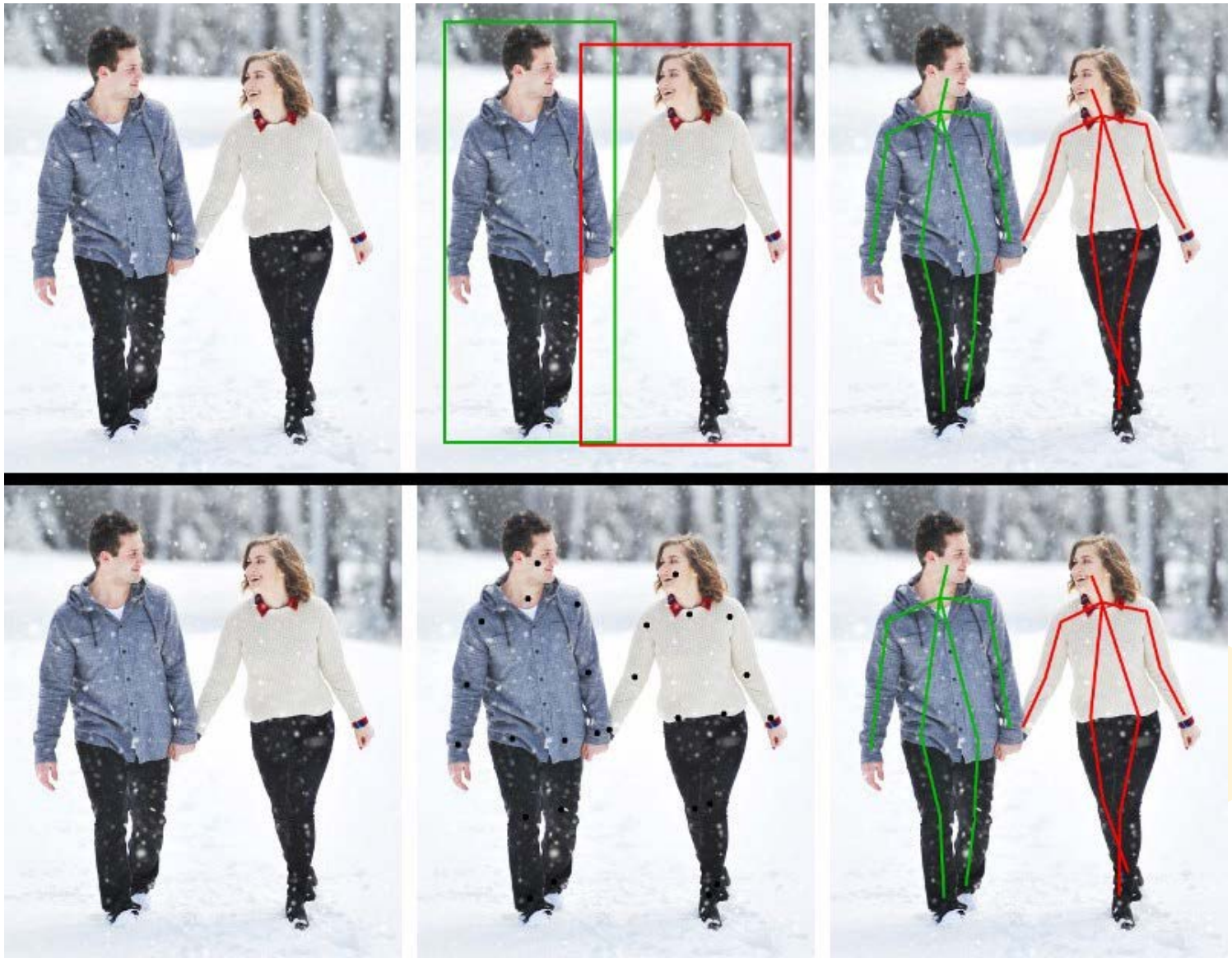
Part (Keypoint) and Pair (Limb)

- Each co-ordinate in the skeleton is known as a part (or a joint, or a keypoint)
- A valid connection between two parts is known as a pair (or a limb)



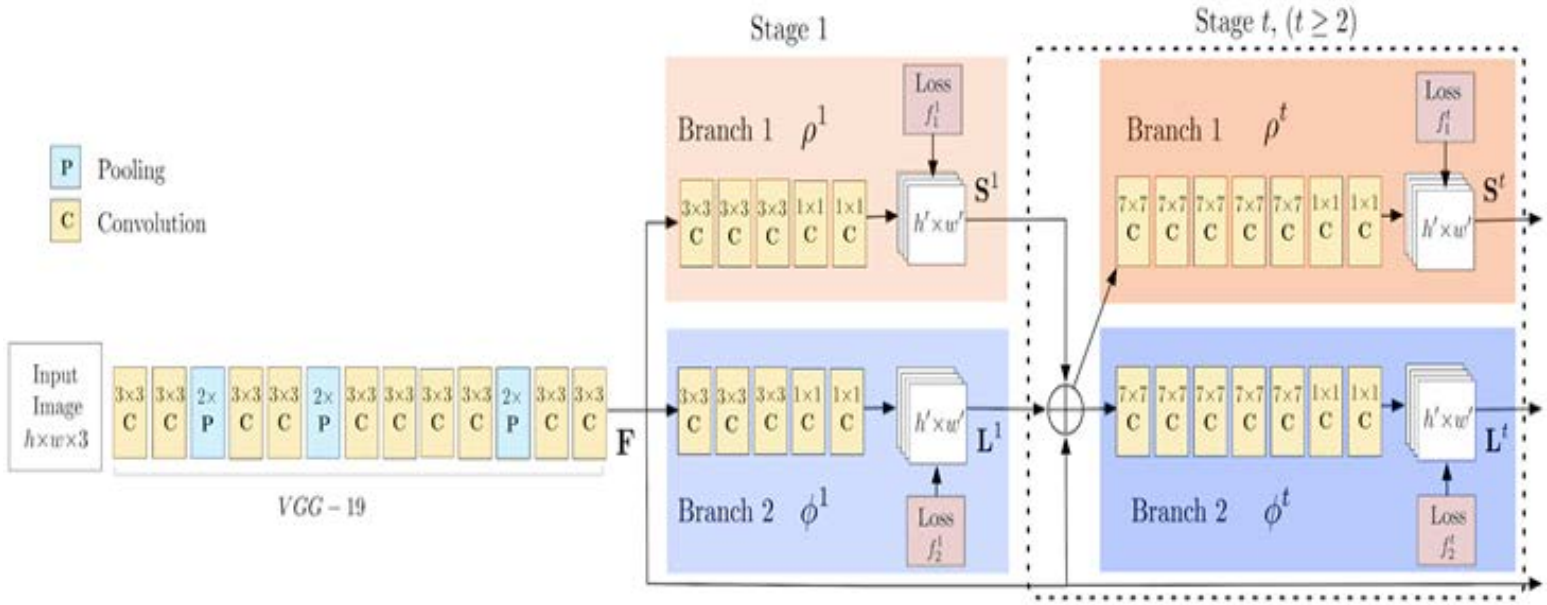
Top-down vs. Bottom-up Approaches

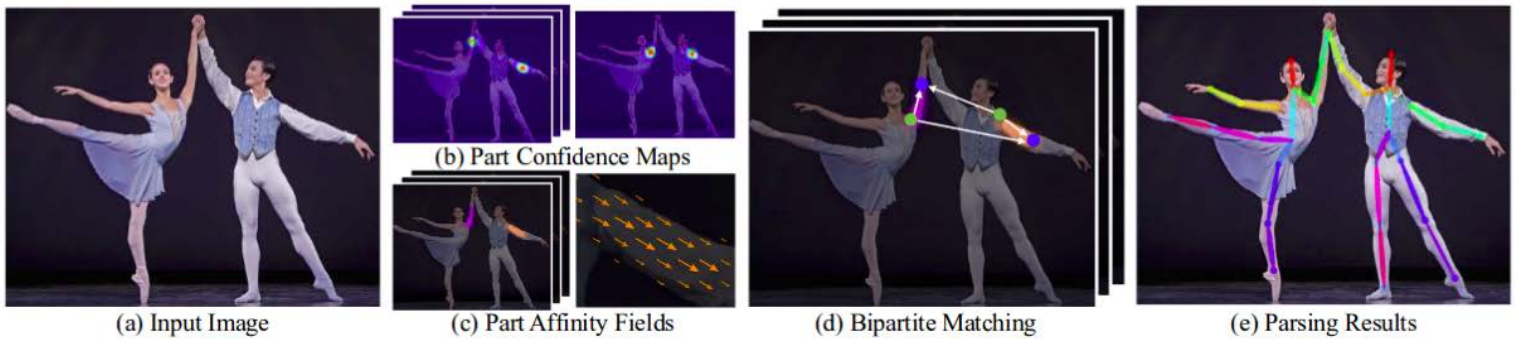
- top-down
  - incorporate a person detector first,
  - followed by estimating the parts
  - and then calculating the pose for each person
  - easier
- bottom-up
  - detect all parts in the image (i.e. parts of every person)
  - followed by associating/grouping parts belonging to distinct persons.



OpenPose

- first detects parts belonging to every, followed by assigning parts to distinct individuals
- first extracts features using the first few layers (VGG-19 in flowchart).
- The features are then fed into two parallel branches of convolutional layers
  - The first branch predicts a set of 18 confidence maps, with each map representing a particular part of the human pose skeleton
  - The second branch predicts a set of 38 Part Affinity Fields (PAFs) which represents the degree of association between parts
  - Successive stages are used to refine the predictions made by each branch.
  - Using the part confidence maps, bipartite graphs are formed between pairs of parts
  - Using the PAF values, weaker links in the bipartite graphs are pruned.
- bottom-up approach



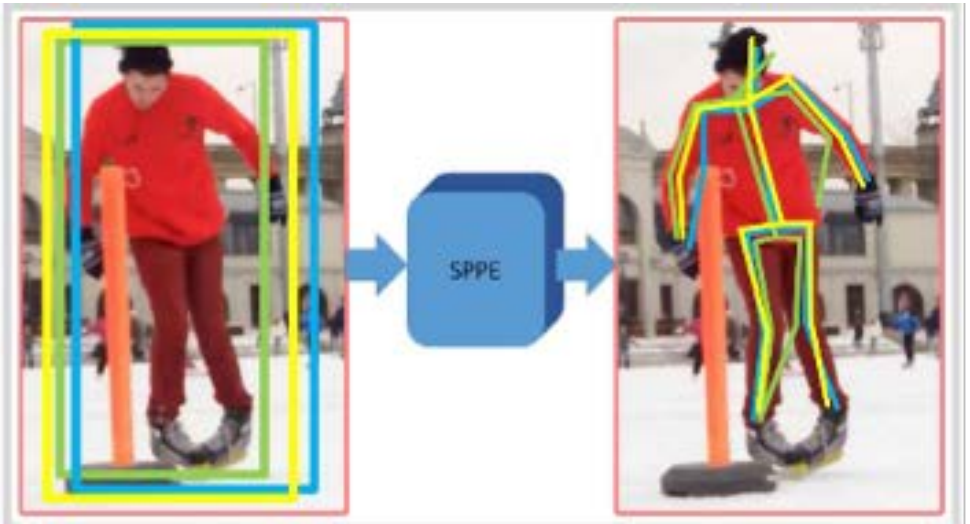


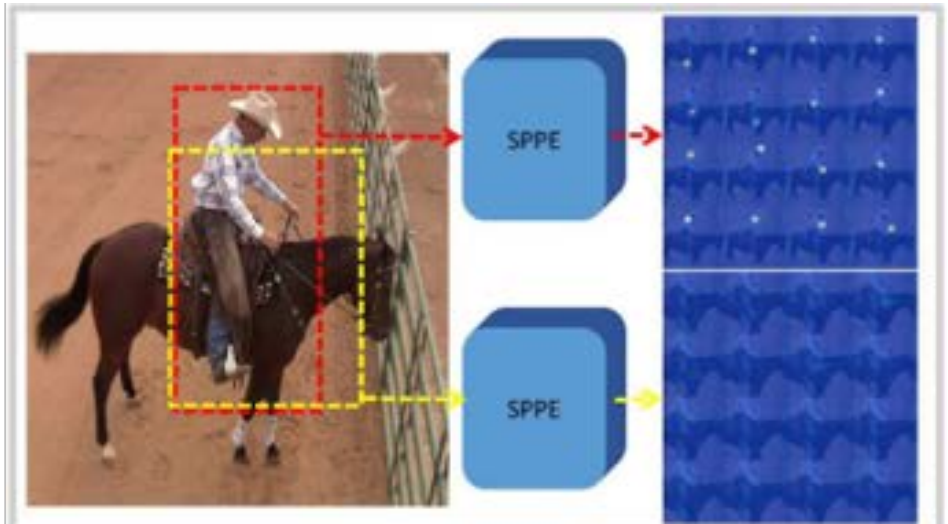
DeepCut

- Produce a set of D body part candidates
  - represents all possible locations of body parts for every person
- Select a subset of body parts from the above set of body part candidates.
- Label each selected body part with one of C body part classes
  - represent the types of parts, such as “arm”, “leg”, “torso” etc.
- Partition body parts that belong to the same person
- bottom-up approach

RMPE (AlphaPose)

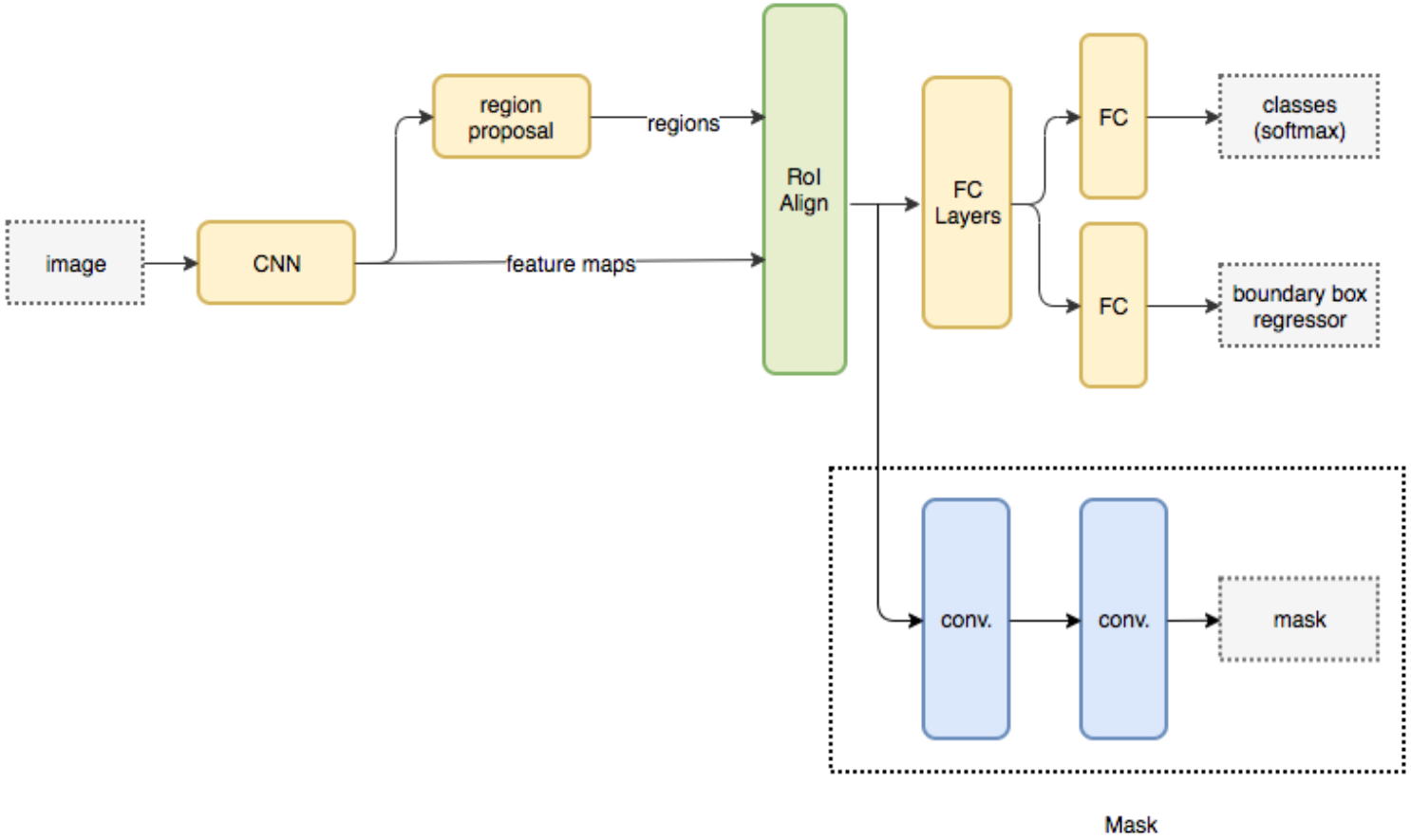
- top-down approach
  - pose estimation is performed on the region where the person is located
  - Hence, errors in localization and duplicate bounding box predictions can cause the pose extraction algorithm to perform sub-optimally
- use Symmetric Spatial Transformer Network (SSTN) to extract a high-quality single person region from an inaccurate bounding box
- A Single Person Pose Estimator (SPPE) is used to estimate the human pose skeleton for that person.
- A Spatial De-Transformer Network (SDTN) is used to remap the estimated human pose back to the original image coordinate system.
- Finally, a parametric pose Non-Maximum Suppression (NMS) technique is used to handle the issue of redundant pose deductions.



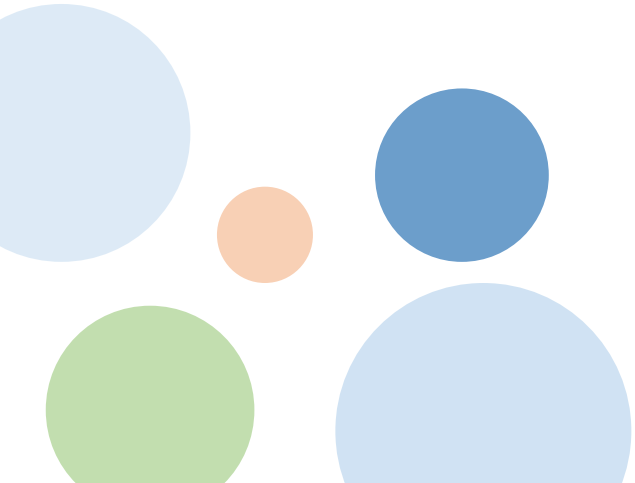


Mask RCNN

- Mask RCNN parallelly predicts both the bounding box locations of the various objects and a mask that semantically segments the object
- first extracts feature maps using a CNN
- use Region Proposal Network (RPN) to get bounding box candidates for the presence of objects.
- Since the bounding box candidates can be of various sizes, a layer called RoIAlign is used to reduce the size of the extracted feature such that they are all of the uniform size.
- this extracted feature is passed into the parallel branches of CNNs for final prediction of the bounding boxes and the segmentation masks.
- By combining the information of the location of the person as well as their set of keypoints, we obtain the human pose skeleton



Caption Generation



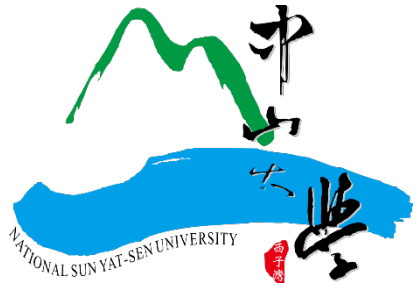
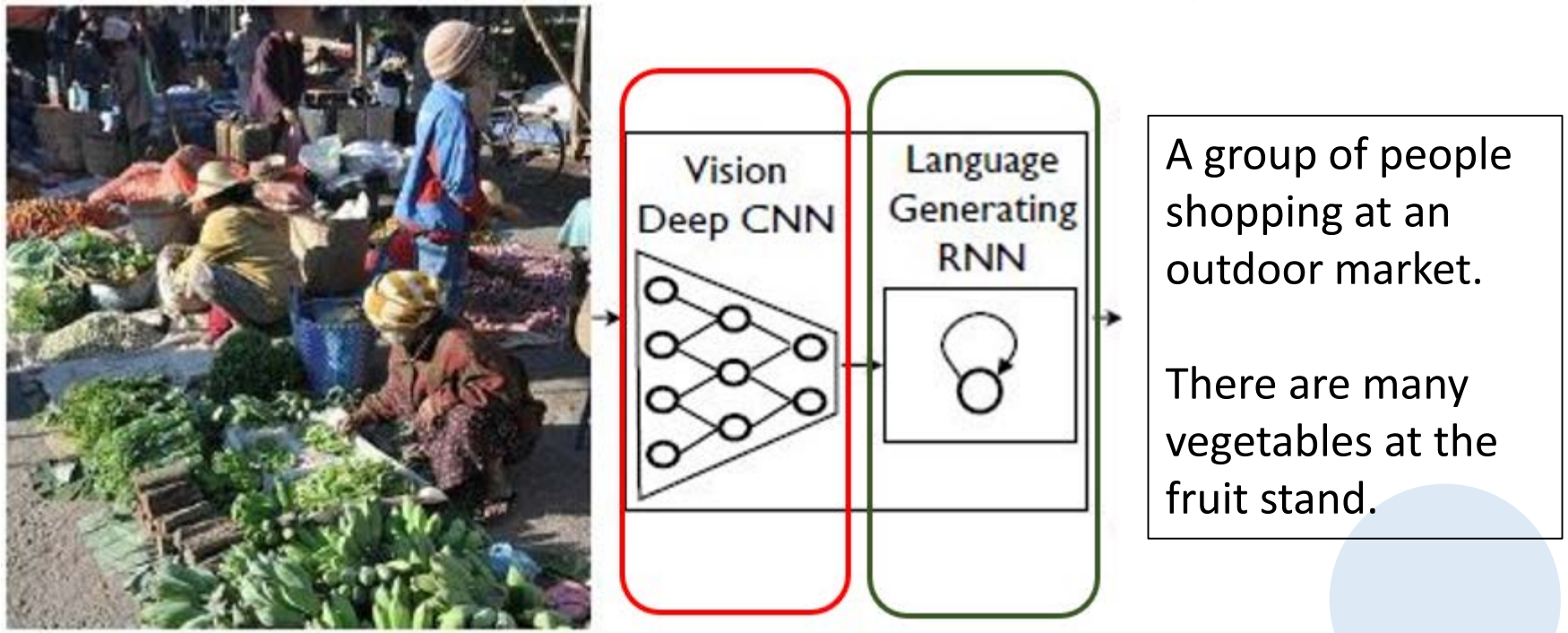


Image Caption

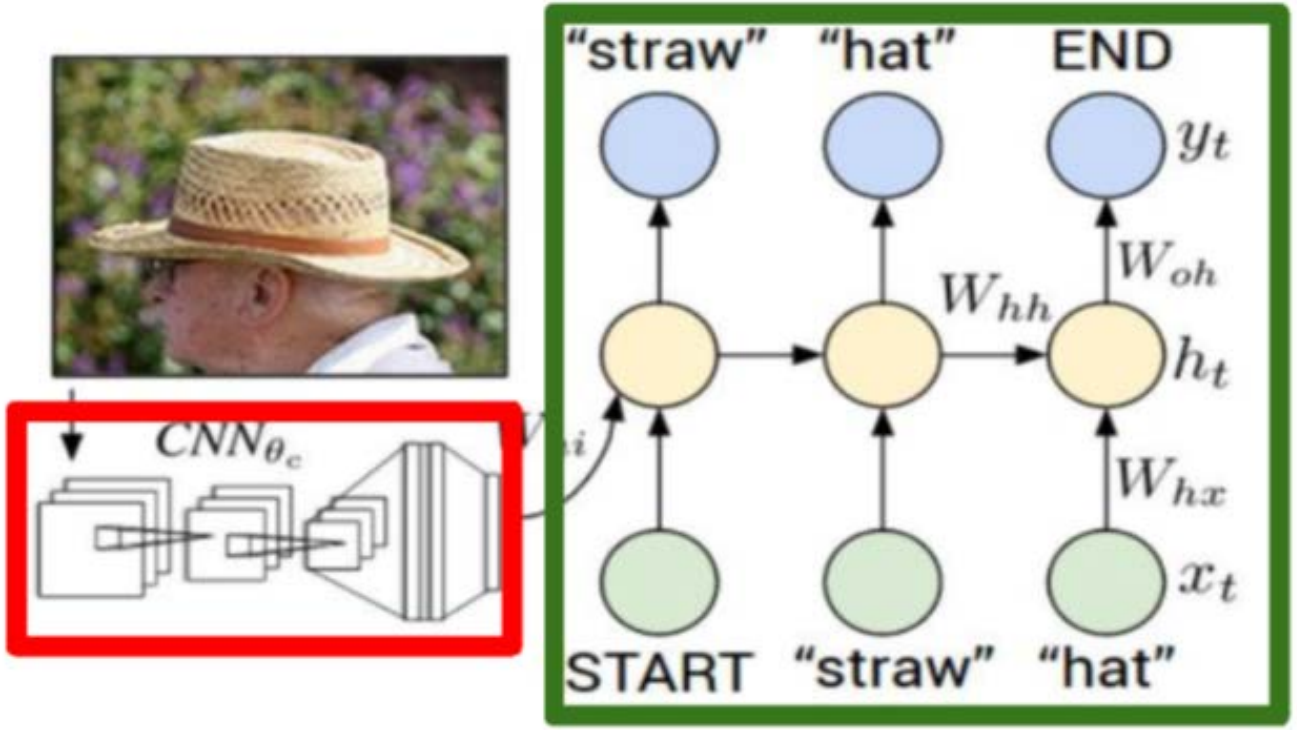
- 主要是 Visual to Language 的問題

RNN:產生文字



CNN:特徵擷取

Captioning with RNN



Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Flow

image

conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096

Base model (CNN)
→特徵擷取

straw
Hat
<End>
Finish !

y0
h0
X0
<start>
h0
straw
y2
h2
Hat

Results

A large white **bird** standing in a forest.

A woman holding a **clock** in her hand.

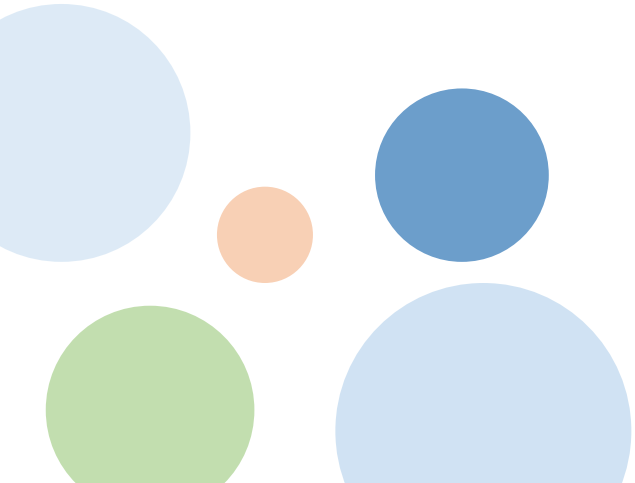
A man wearing a hat and
a hat on a **skateboard**.

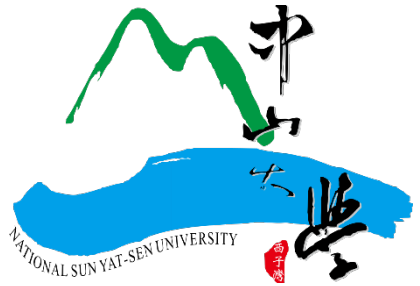
A person is standing on a beach
with a **surfboard**.

A woman is sitting at a table with
a large **pizza**.

A man is talking on his cell **phone**
while another man watches.

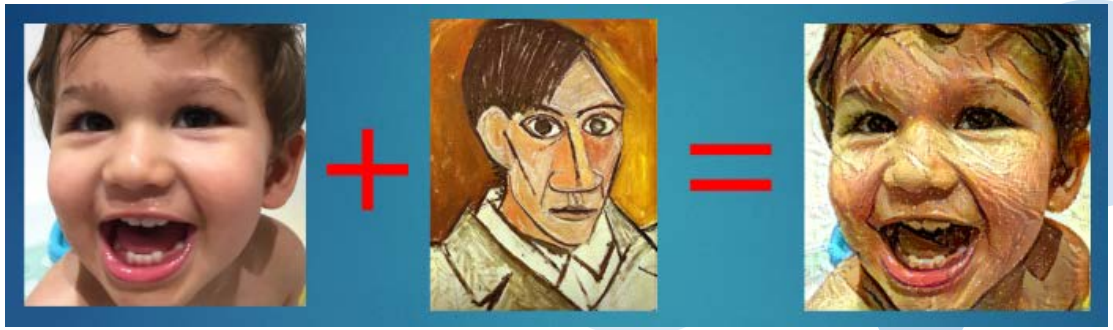
Style Transfer





Style-transfer

- 將一幅藝術作品的圖像風格 應用到 另一張圖片上
- 主要模擬出「顏色」、「紋理」就可以是某種風格
- 利用CNN 網路進行特徵擷取
- CNN 網路: 前面幾層特徵→ 包含較多的圖像視覺特徵 (顏色、紋理 等)
  後面幾層特徵→包含較多的圖像的內容描述



P 是原圖內容的特徵、F 是合成圖內容的特徵
A 是原圖風格的特徵、G 是合成圖風格的特徵

Style-transfer* (CNN-based)

1.初始化 Synthetic Image(合成圖片)
2.使用CNN架構 ，分別將 Content Image(原始圖片)、 Style Image(風格圖片)進行特徵擷取

3.把 前面幾層的特徵圖 進行風格相似性比較
(風格圖 和 合成圖的特徵差距)→計算 Style Loss

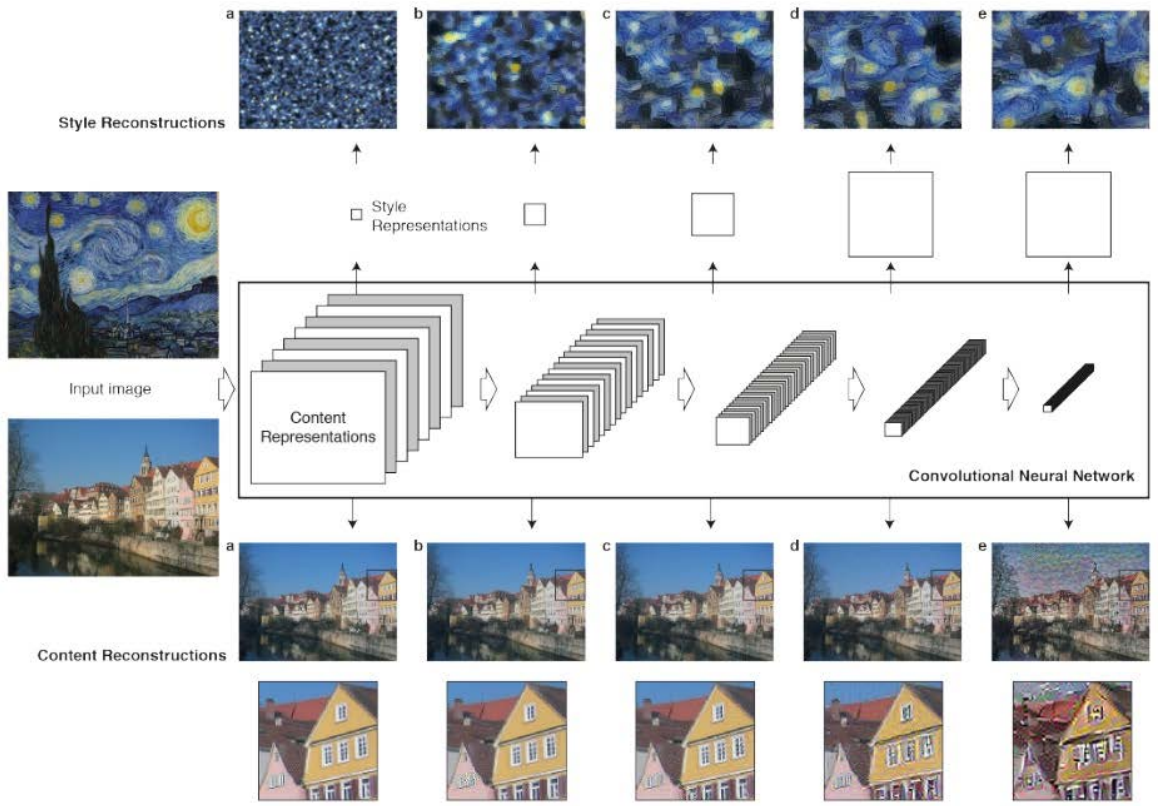
$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

把後面幾層的特徵圖 進行內容相似性比較
(原圖 和 合成圖的特徵差距)→計算Content Loss

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2$$

4. Total loss = Style Loss + Content Loss



5. 利用梯度下降法 →最小化 Total loss →更新合成圖信息，得到最終Synthetic Image

*L.A. Gatys, “Image Style Transfer Using Convolutional Neural Networks,” CVPR, 2016.

Results

A

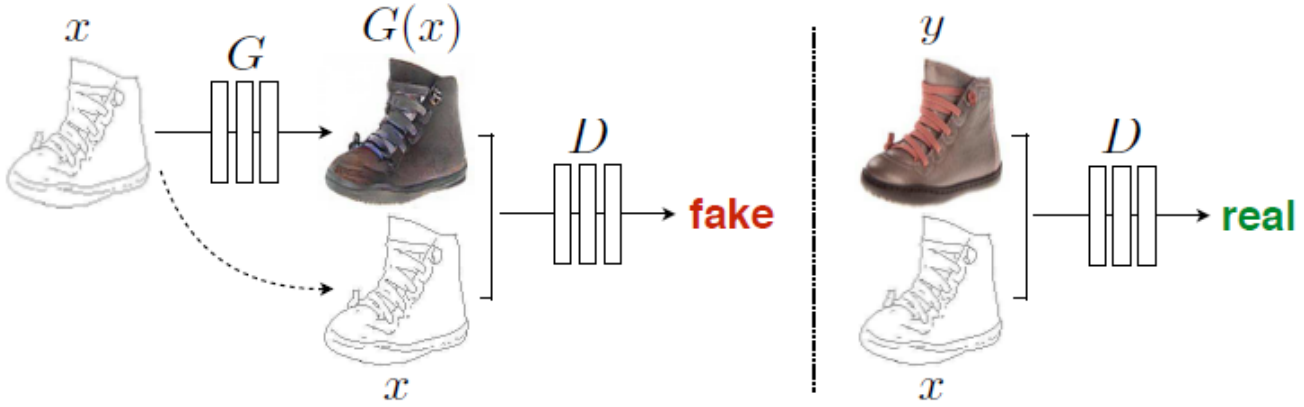
B

C

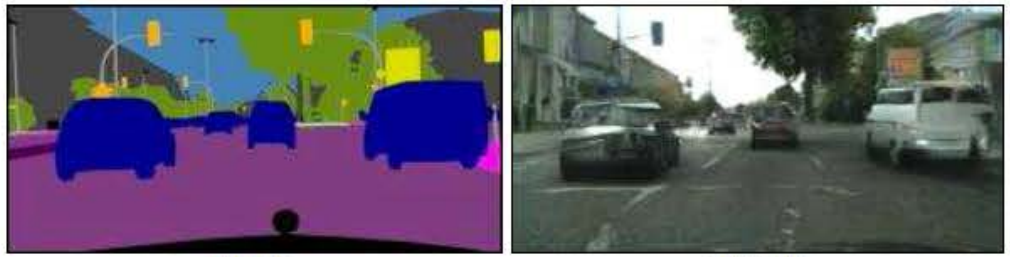
D

Image Translation (cGAN-based)

- conditional GAN with **paired** images

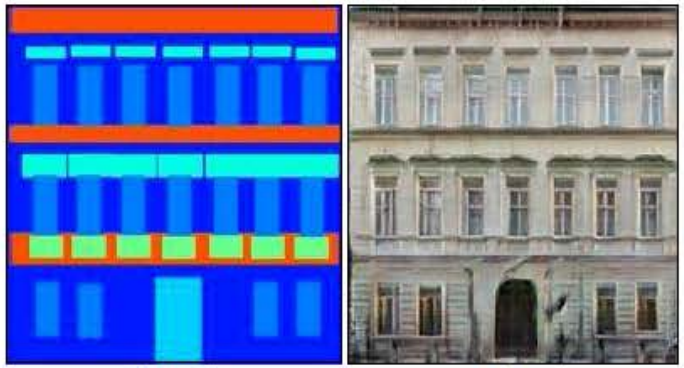


Labels to Street Scene



input output

Labels to Facade



input output

BW to Color

input output

Aerial to Map

input output

Day to Night

input output

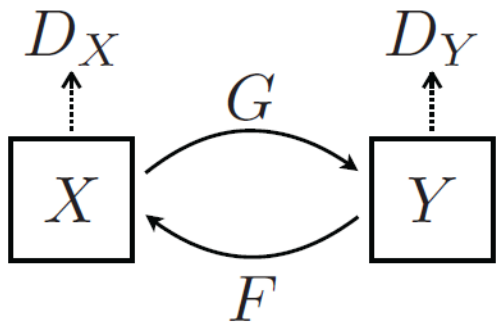
Edges to Photo

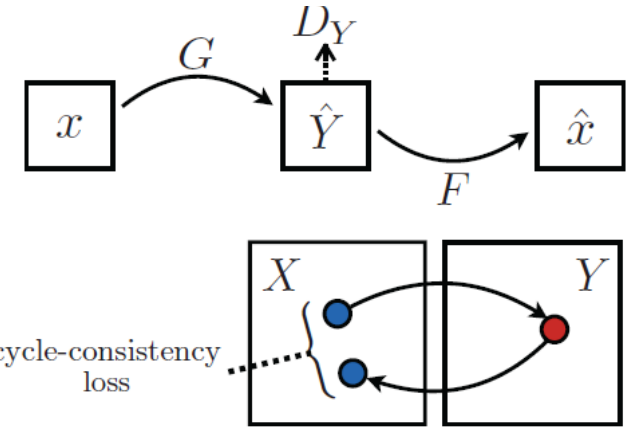
input output

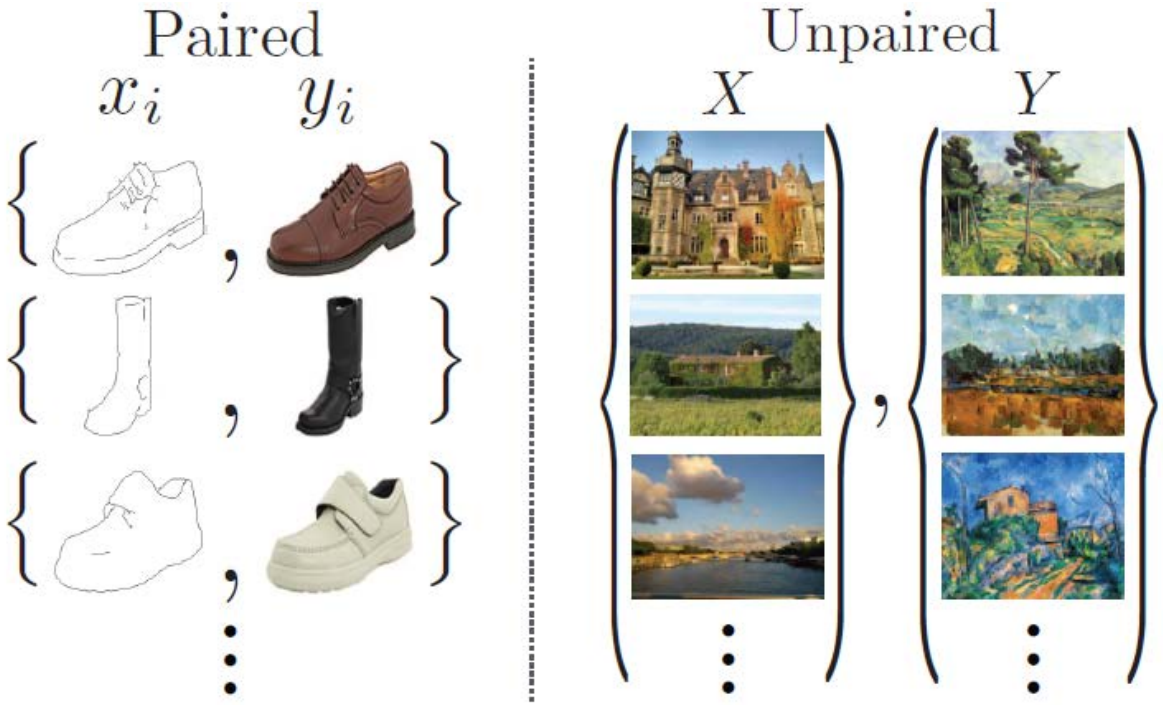
*P. Isola, et al., ”Image-to-Image Translation with Conditional Adversarial Networks’”, CVPR, 2017

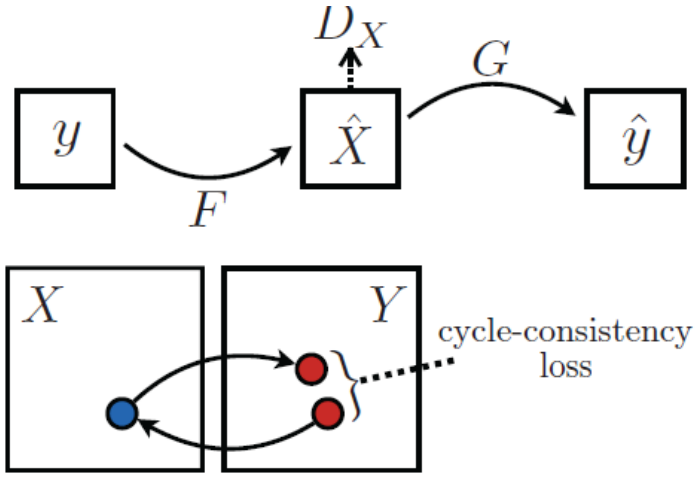
Image Translation (cycle GAN-based)

- cycle-GAN for unpaired image translation









*J.-Y Zhu, et al., ”Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks’”, ICCV, 2017

Results (Image Translation)

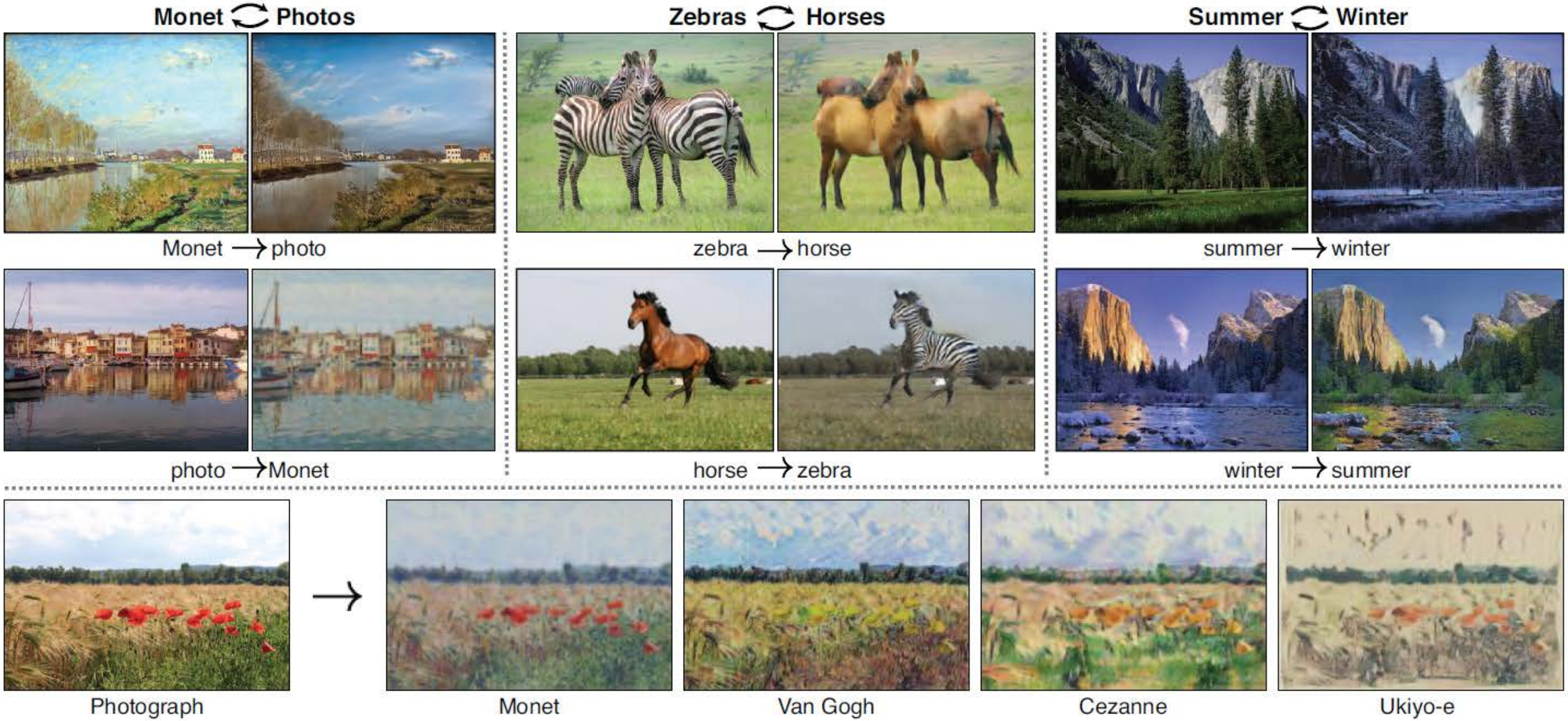
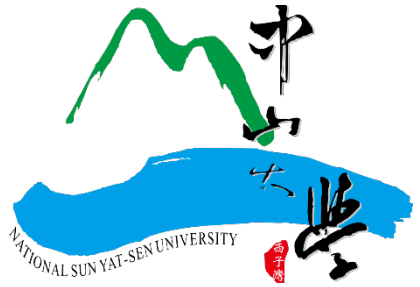
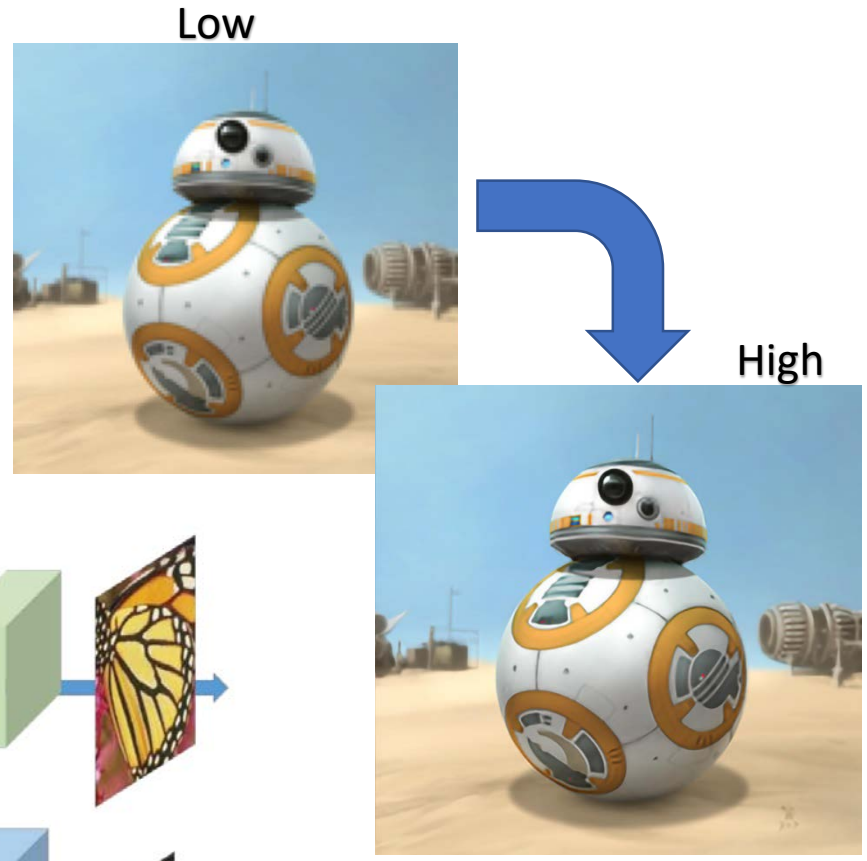


Image Enhancement

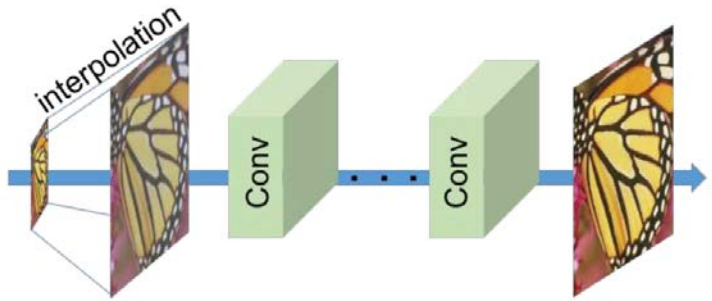


Super-Resolution

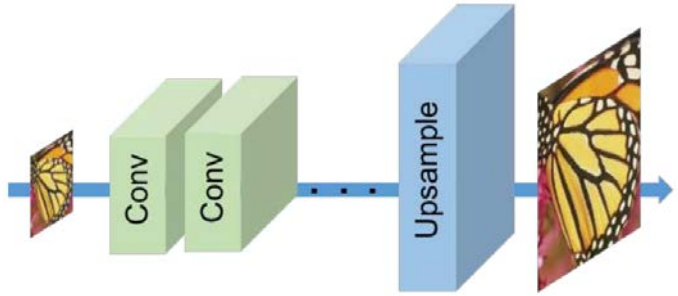
- 將 Image 從 **低解析度** 轉化成**高解析度**
- 大致上分成 3 種架構:



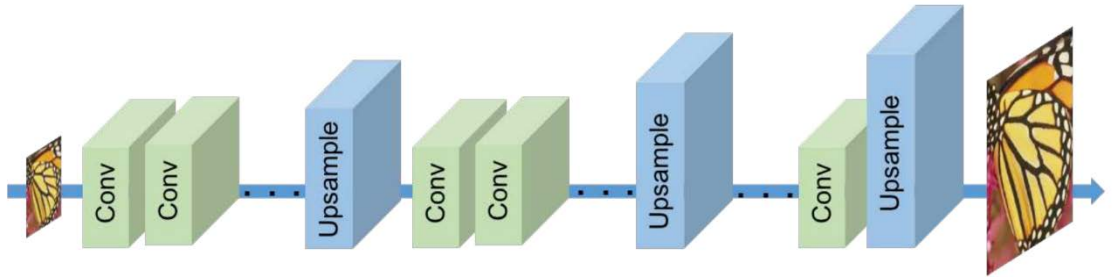
1. Predefined upsampling



2. Single upsampling

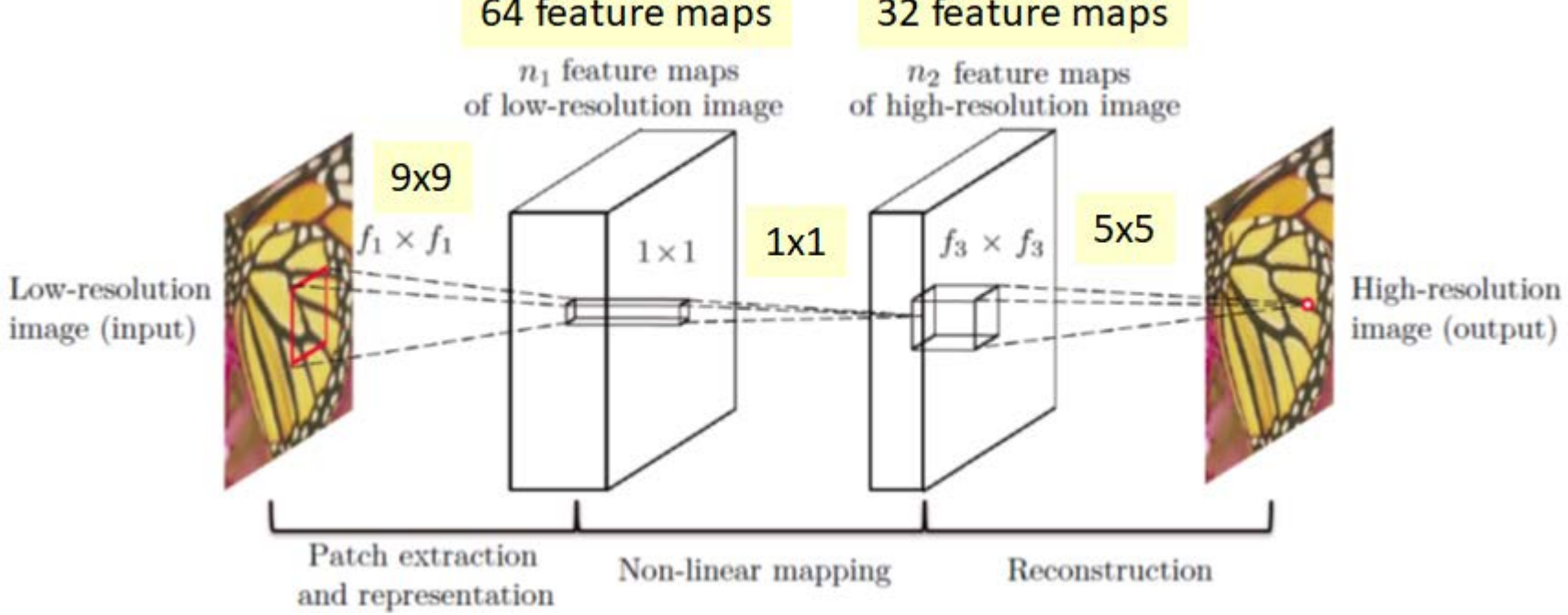


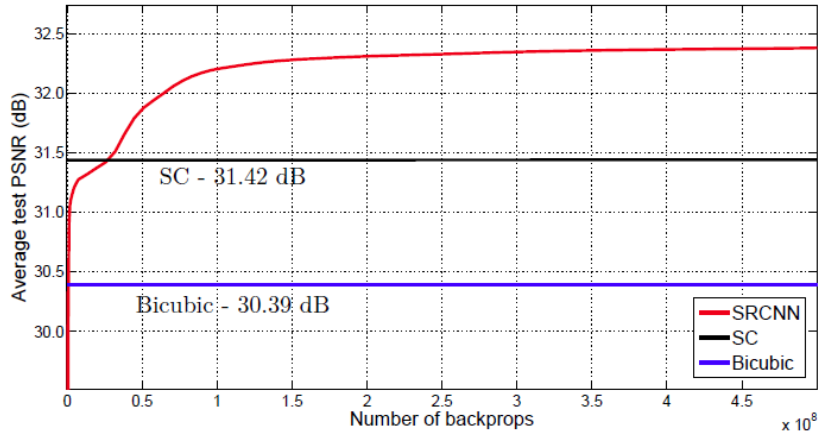
3. Progressive upsampling



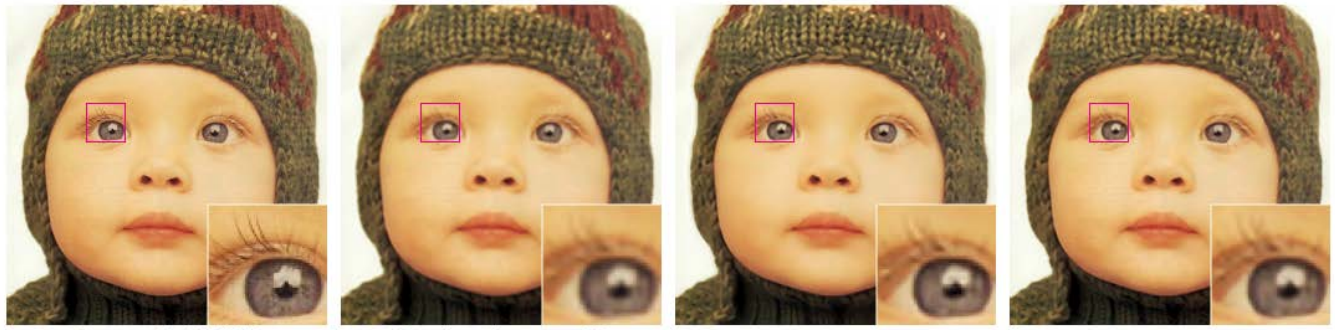
SRCNN

2014 ECCV, 2016 TPAMI

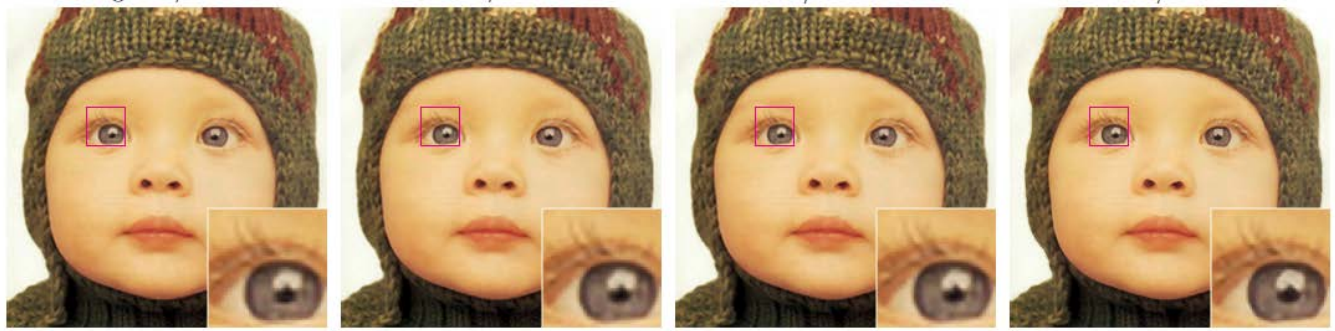




Original / PSNR | Bicubic / 24.04 dB | SC / 25.58 dB | SRCNN / 27.58 dB

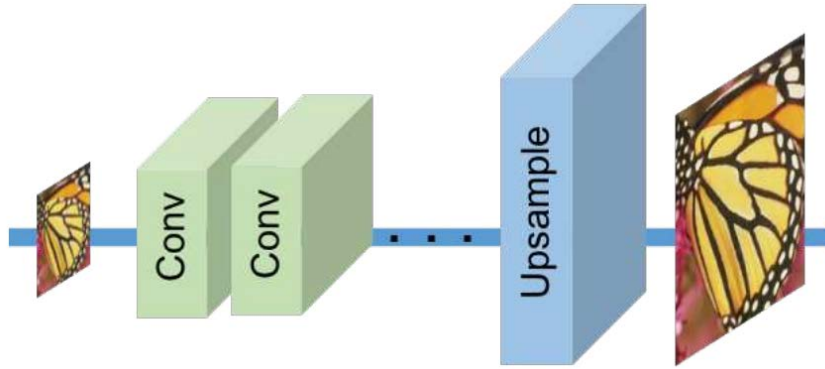


Original / PSNR | Bicubic / 33.91 dB | SC / 34.29 dB | K-SVD / 35.08 dB

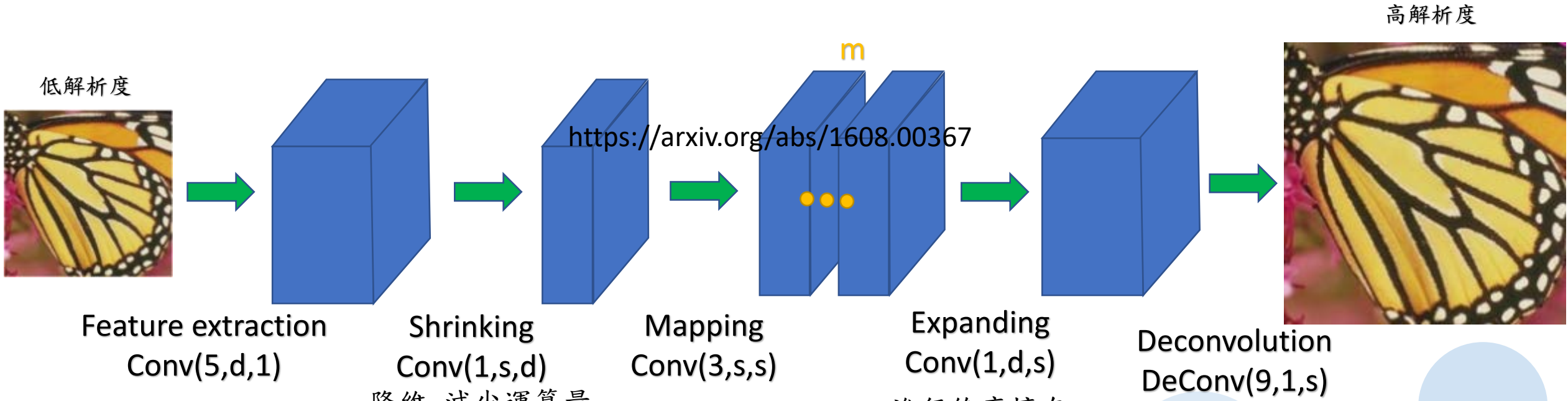


NE+NNLS / 34.77 dB | NE+LLE / 35.06 dB | ANR / **35.13** dB | SRCNN / 35.01 dB

Super-Resolution : FSRCNN



- 將 Image 從 **低解析度**轉化成**高解析度**



FSRCNN = (d,s,m)=(56,12,4)

Conv (f,n,c)→ f = the filter size

n = the number of filters

c = the number of channels

C. Dong, C. C. Loy, and X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network", 2016 ECCV.

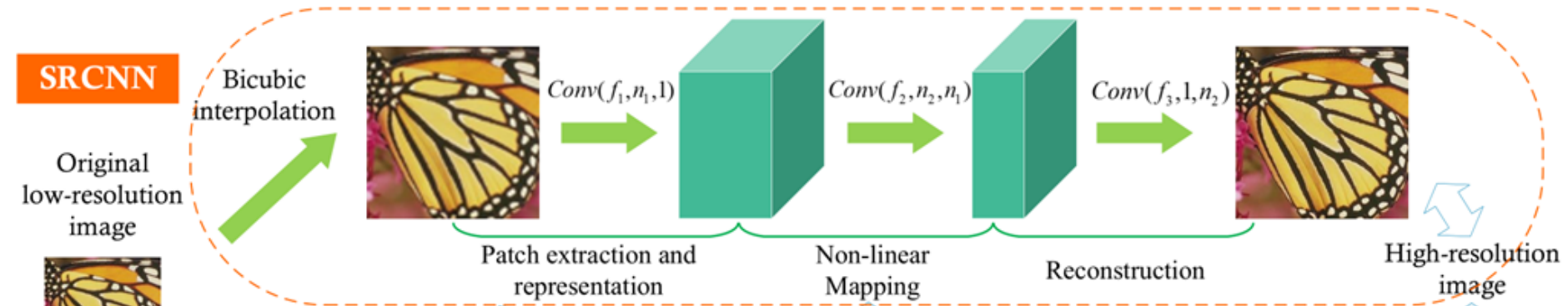
$O\{(f_1 n_1 + n_1 f_2 n_2 + n_2 f_3) S_{HR}\}$

SRCNN vs. FSRCNN

SRCNN (9-1-5)

9×9 | 64 Feature Maps | 1×1 | 32 Feature Maps | 5×5

$O\{(f_1^2 n_1 + n_1 f_2^2 n_2 + n_2 f_3^2) S_{HR}\}$



FSRCNN (d,s,m)

5×5 | d Feature Maps | 1×1 | s Feature Maps | 3×3 | s Feature Maps | 1×1 | d Feature Maps | 9×9

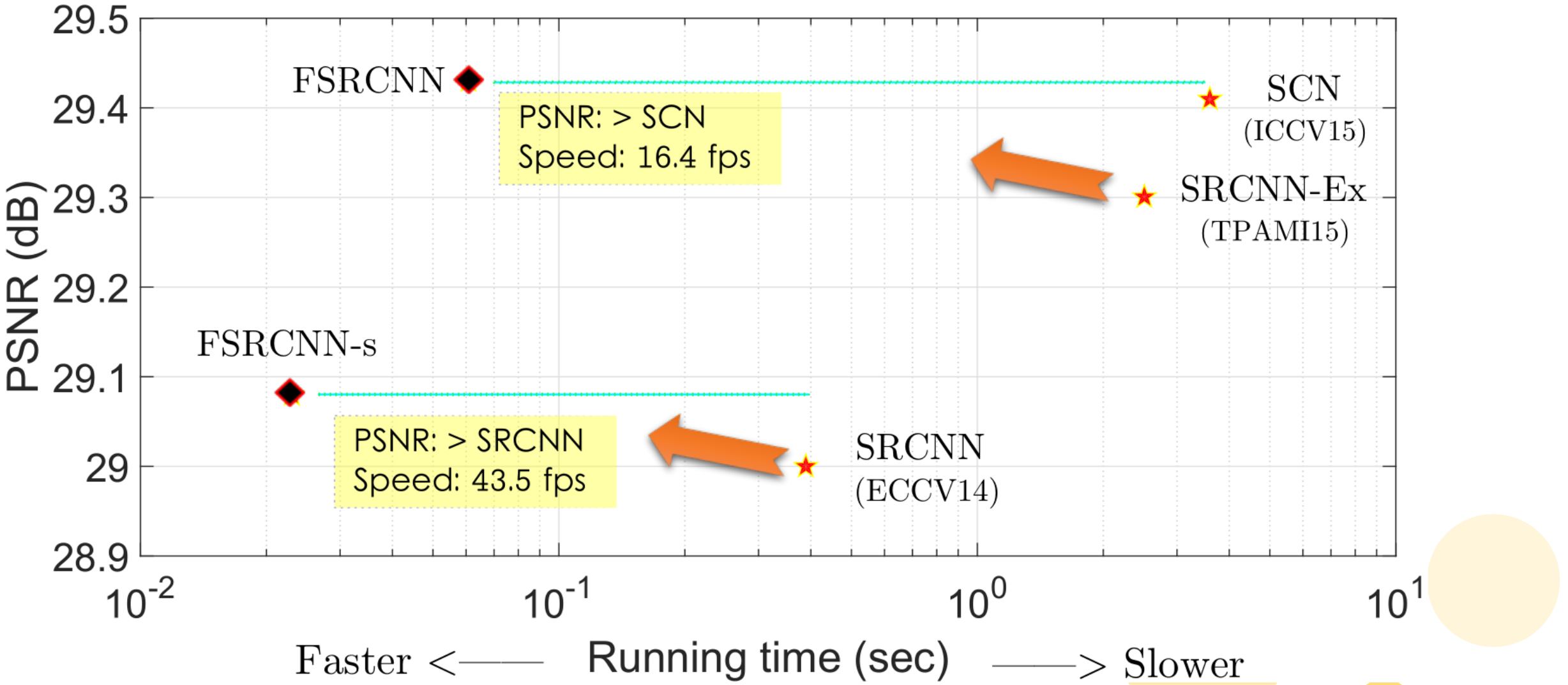
m times

PReLU

$O\{(25d + sd + 9ms^2 + ds + 81d) S_{LR}\}$

$= O\{(9ms^2 + 2sd + 106d) S_{LR}\}$

PSNR vs. Speed

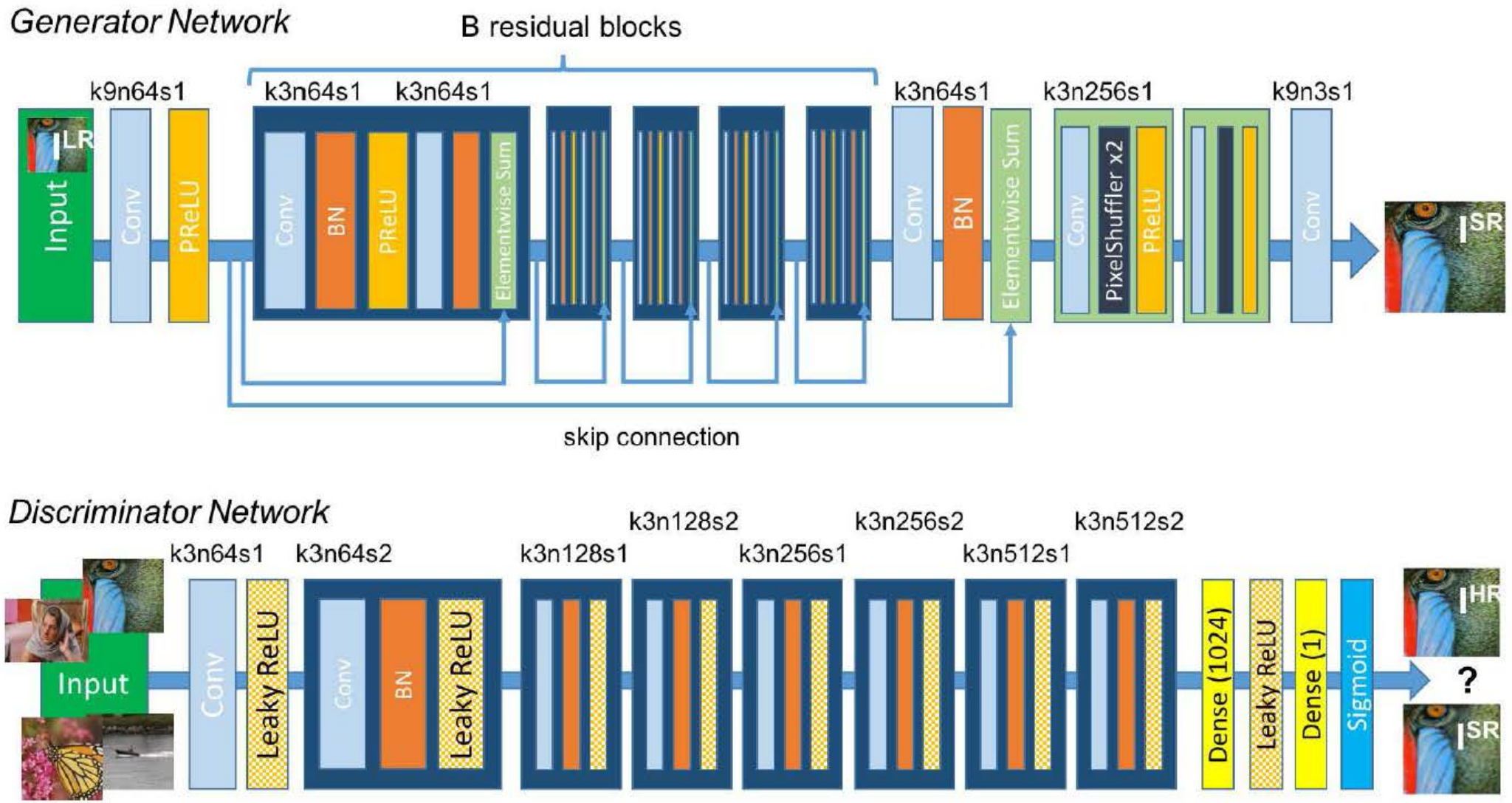


Results (FSRCNN)

Original

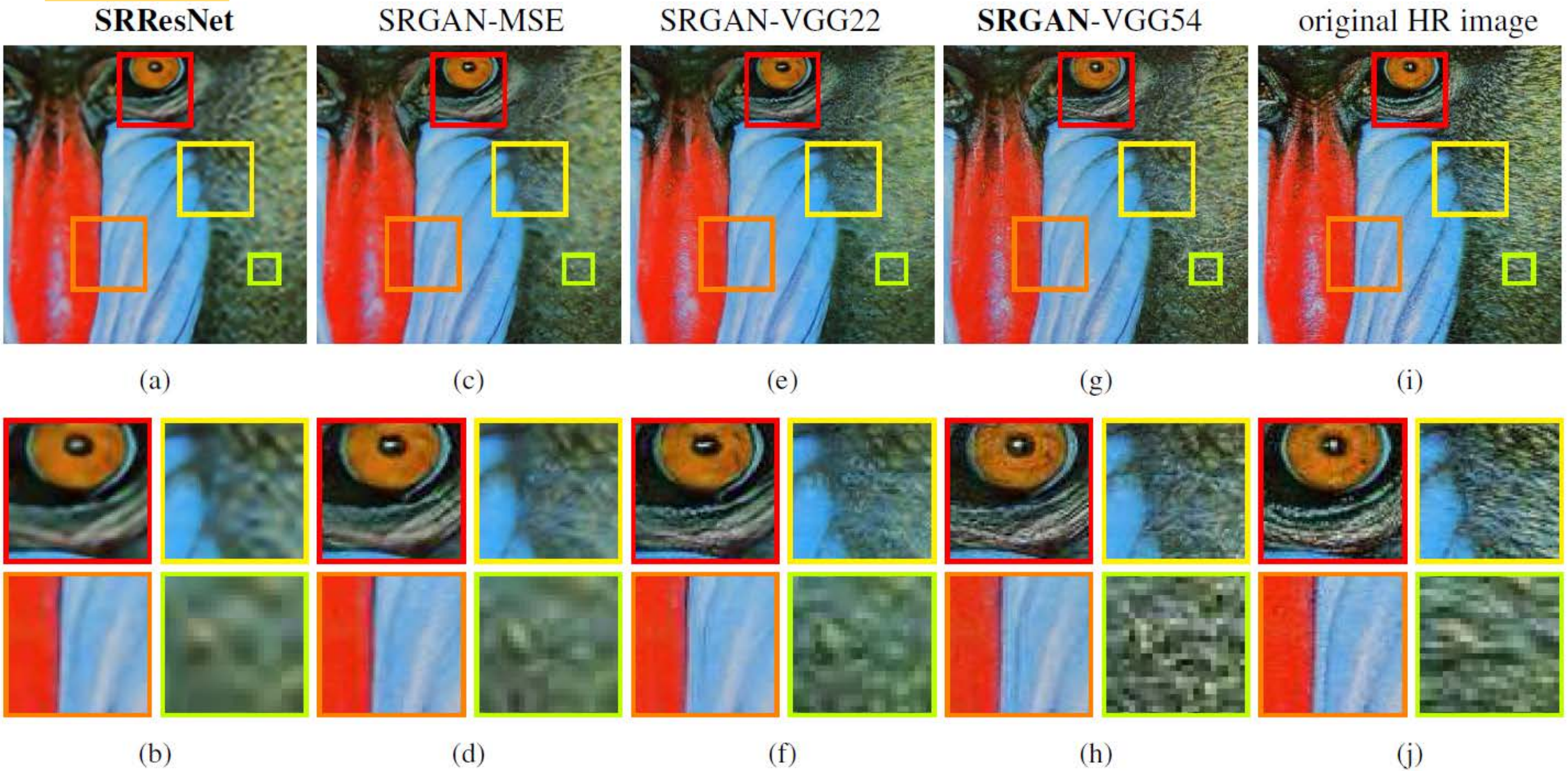
Result

GAN Architecture*

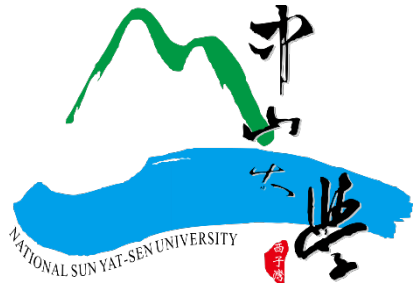


*C. Ledig, et al., "'Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", CVPR, 2017

Results (GAN-based Super-Resolution)



Facial Expression Recognition (FER)



Facial Expression Recognition

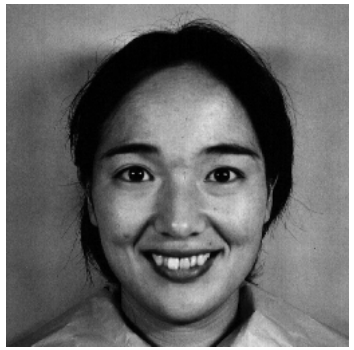
- FER→ Image Classification
- Powerful, Natural and Immediate means for human to communicate their emotions

- Input : Image / Video(Sequential Image)
- Output : Emotion category (Happy, Sad, Neutral, Disgust, Angry …)

Happy Sad Neutral Disgust Angry

Common Architecture

- Convolution Neural Network



Feature Extraction

Emotion Classification

Happy
Sad
Neutral
Disgust
Angry

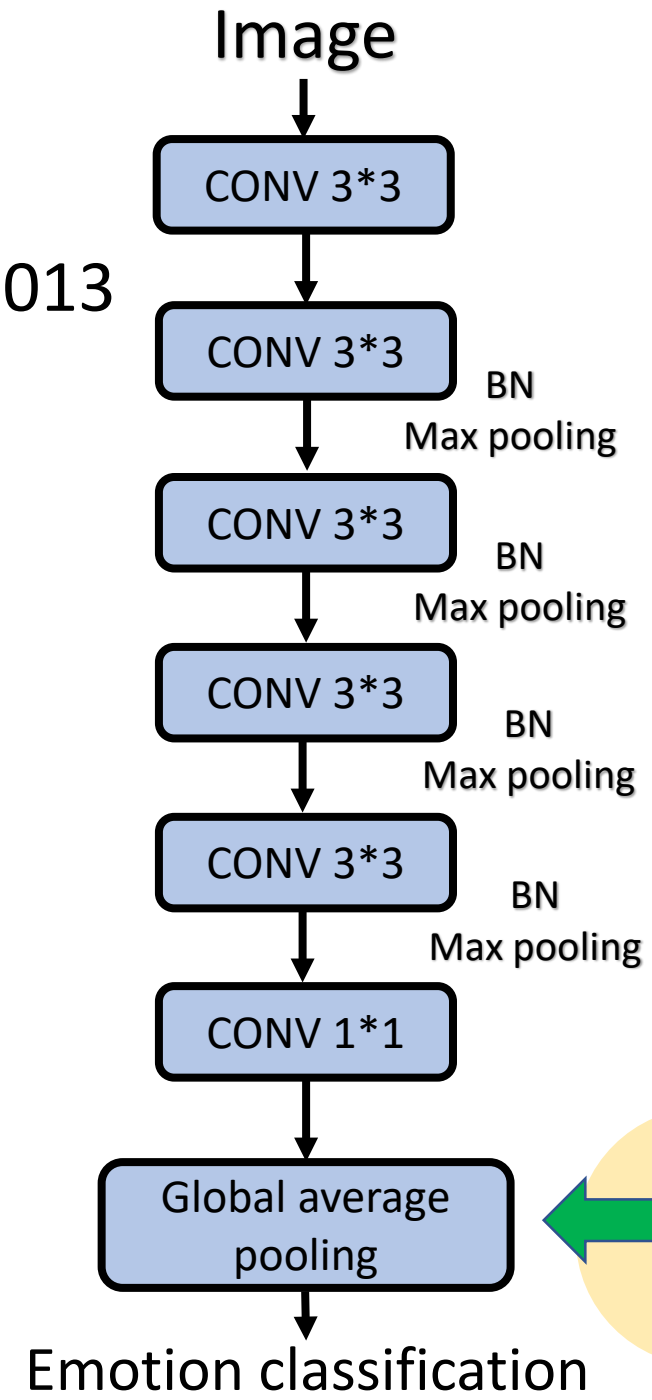
Convolution layer
Pooling layer
Activation
Batch Normalization

Fully connected layer

Result

- Dataset: Kaggle Challenge →FER2013
- Architecture : CNN

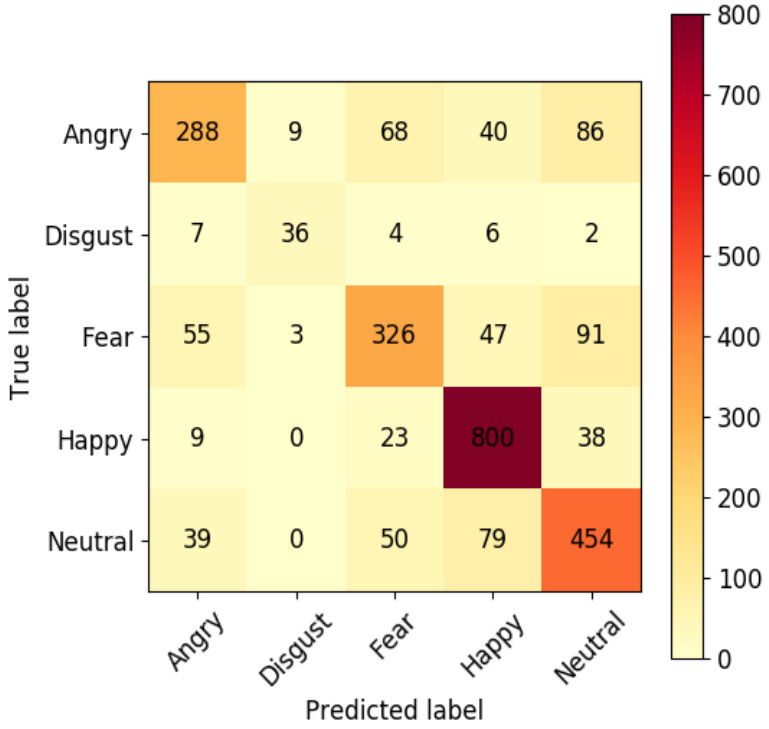
Emotion labels	Numbers
Angry	4,593
Disgust	547
Fear	5,121
Happy	8,989
Neutral	6,198



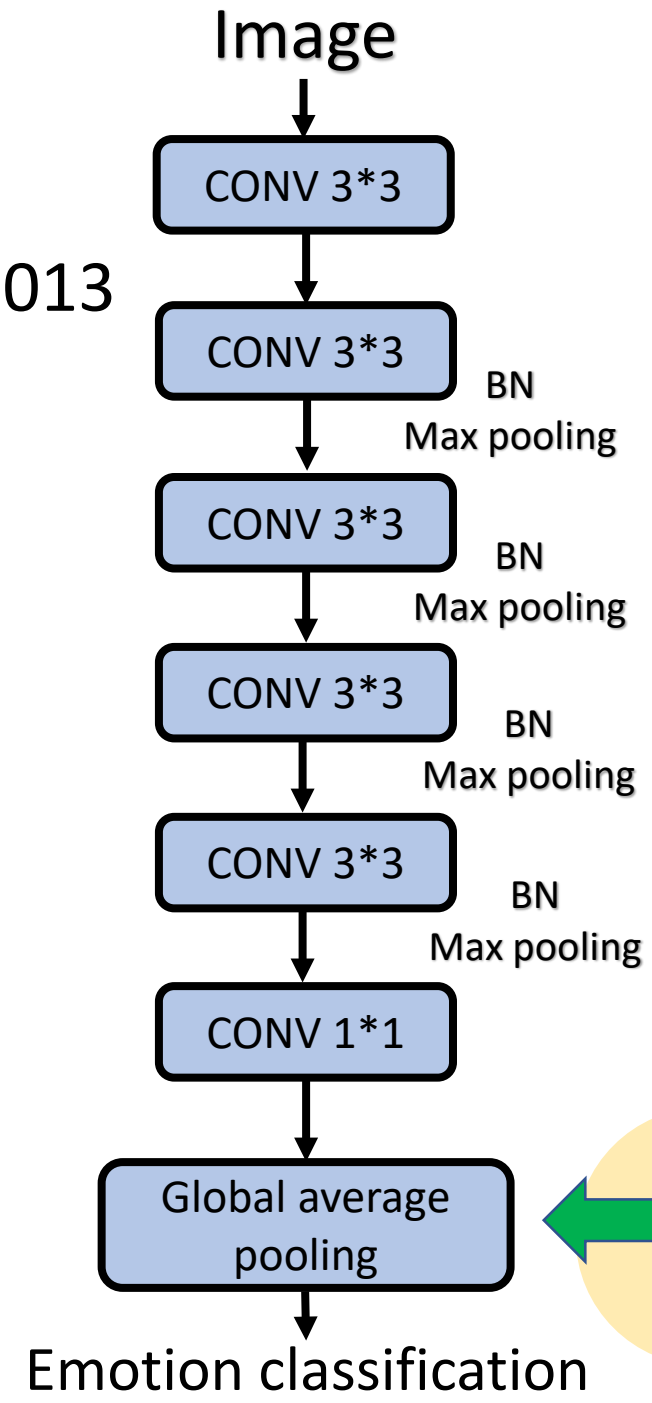
利用GAP 取代FC
使整體架構參數量下降

Result

- Dataset: Kaggle Challenge →FER2013
- Architecture : CNN



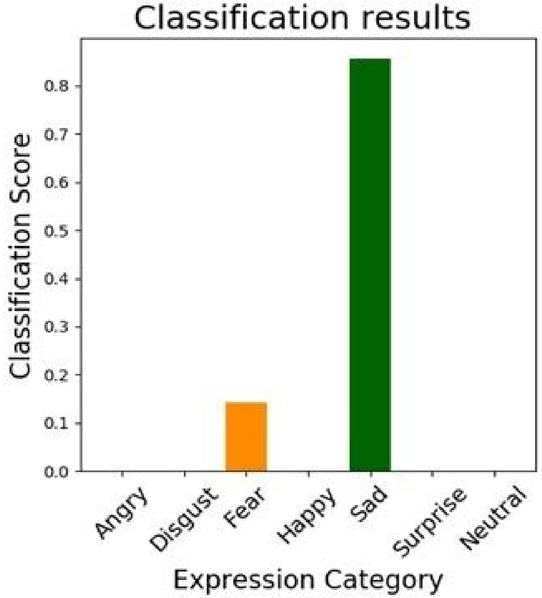
	Accuracy	UA
Val	73.1	67.3
Test	74.3	70.3



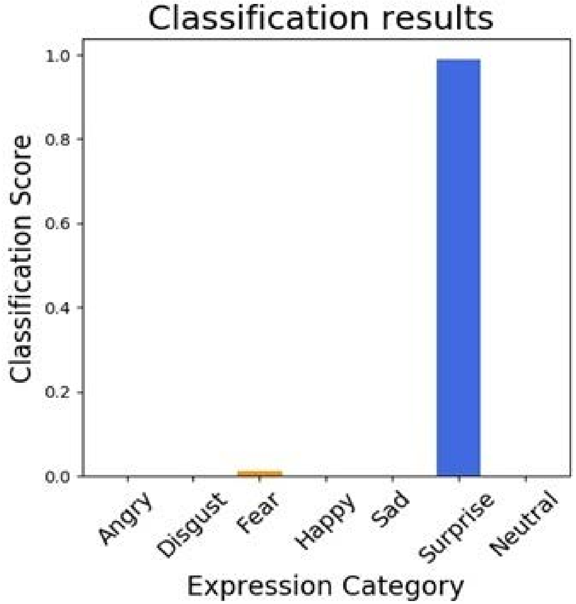
利用GAP 取代FC
使整體架構參數量下降

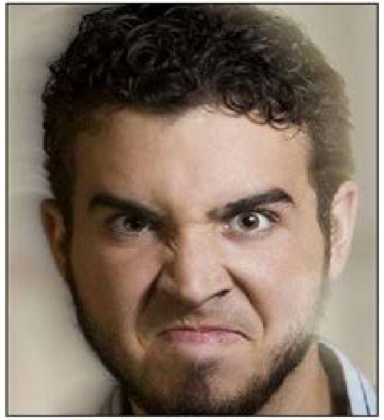
Result

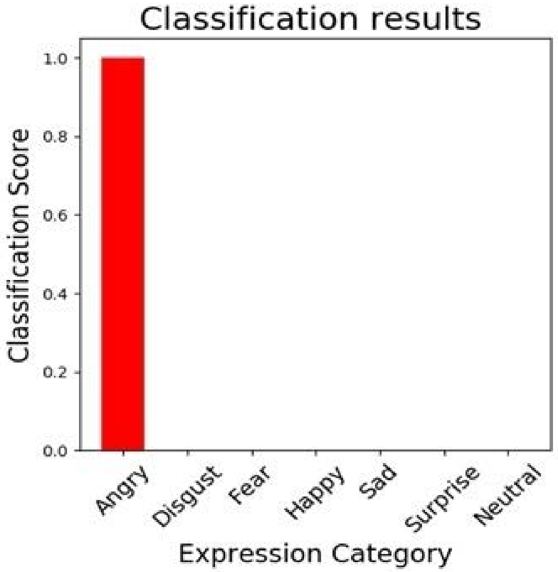

Input Image




Input Image




Input Image




Input Image