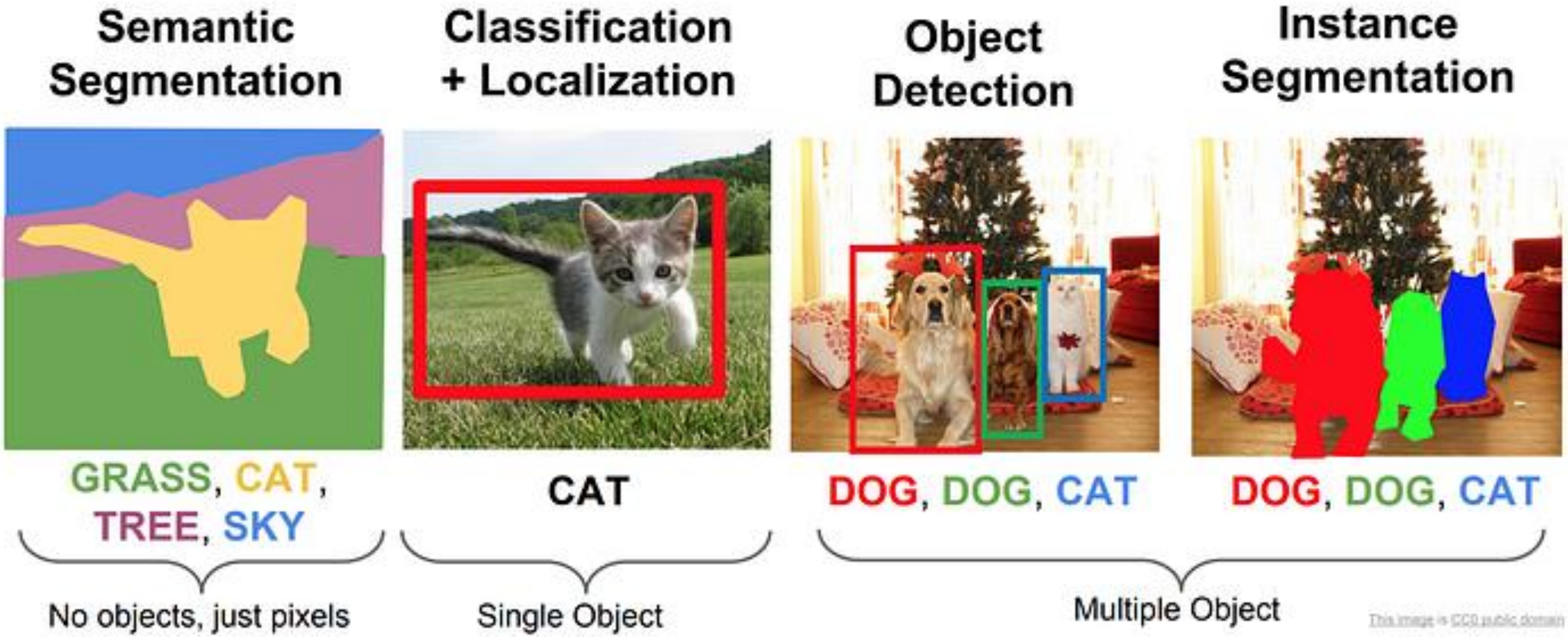
Image Classification / Object Detection

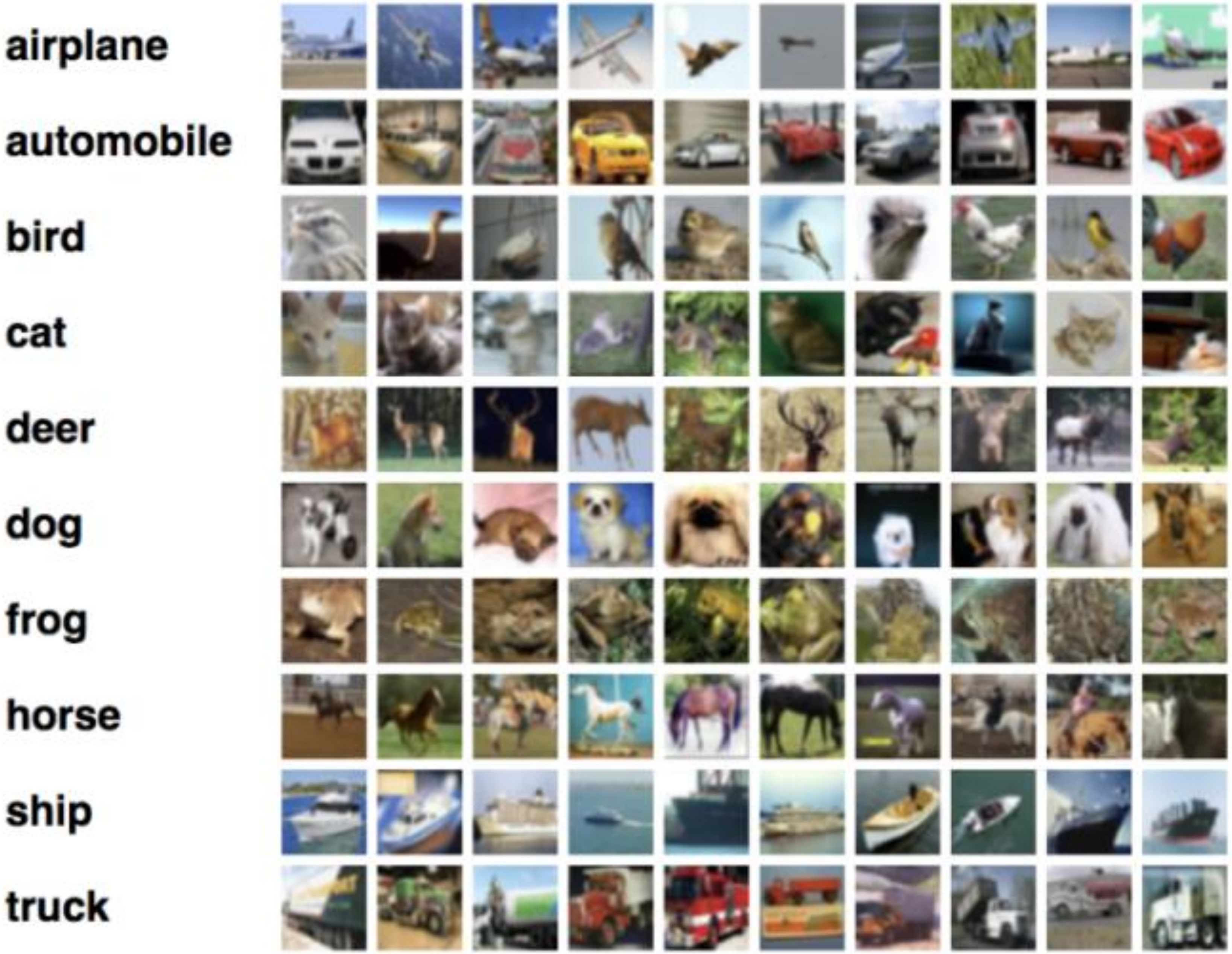
김남범 교수

Image Classification vs Object Detection



LSVRC

The Image Classification Challenge:
1,000 object classes
1,431,167 images

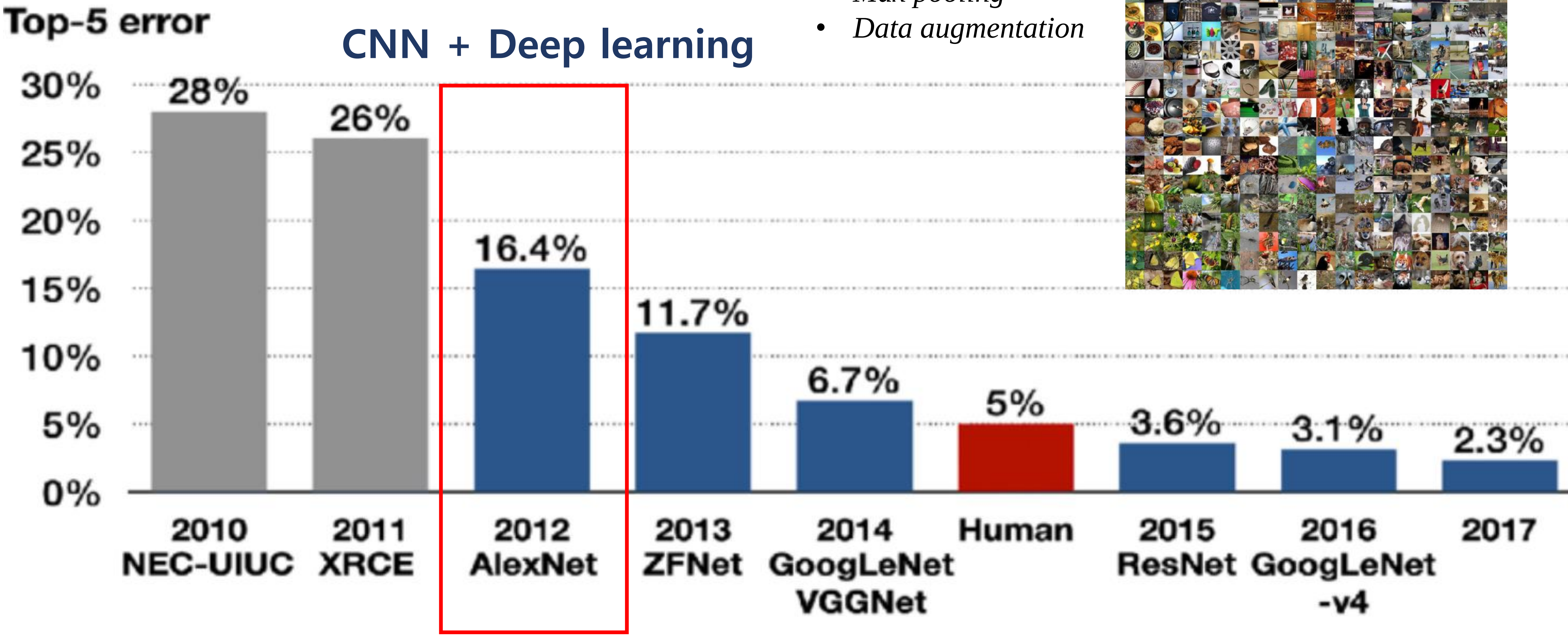


http://www.image-net.org/challenges/LSVRC/

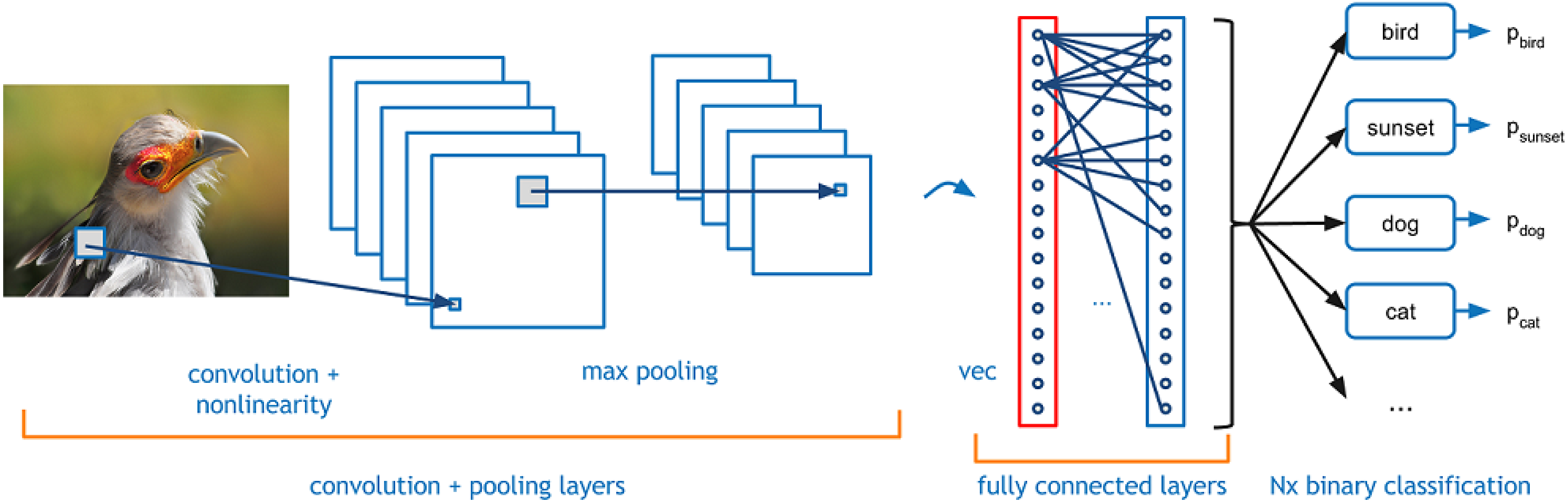
LSVRC (Imagenet)

- *ReLU*
- *Dropout*
- *Max pooling*
- *Data augmentation*

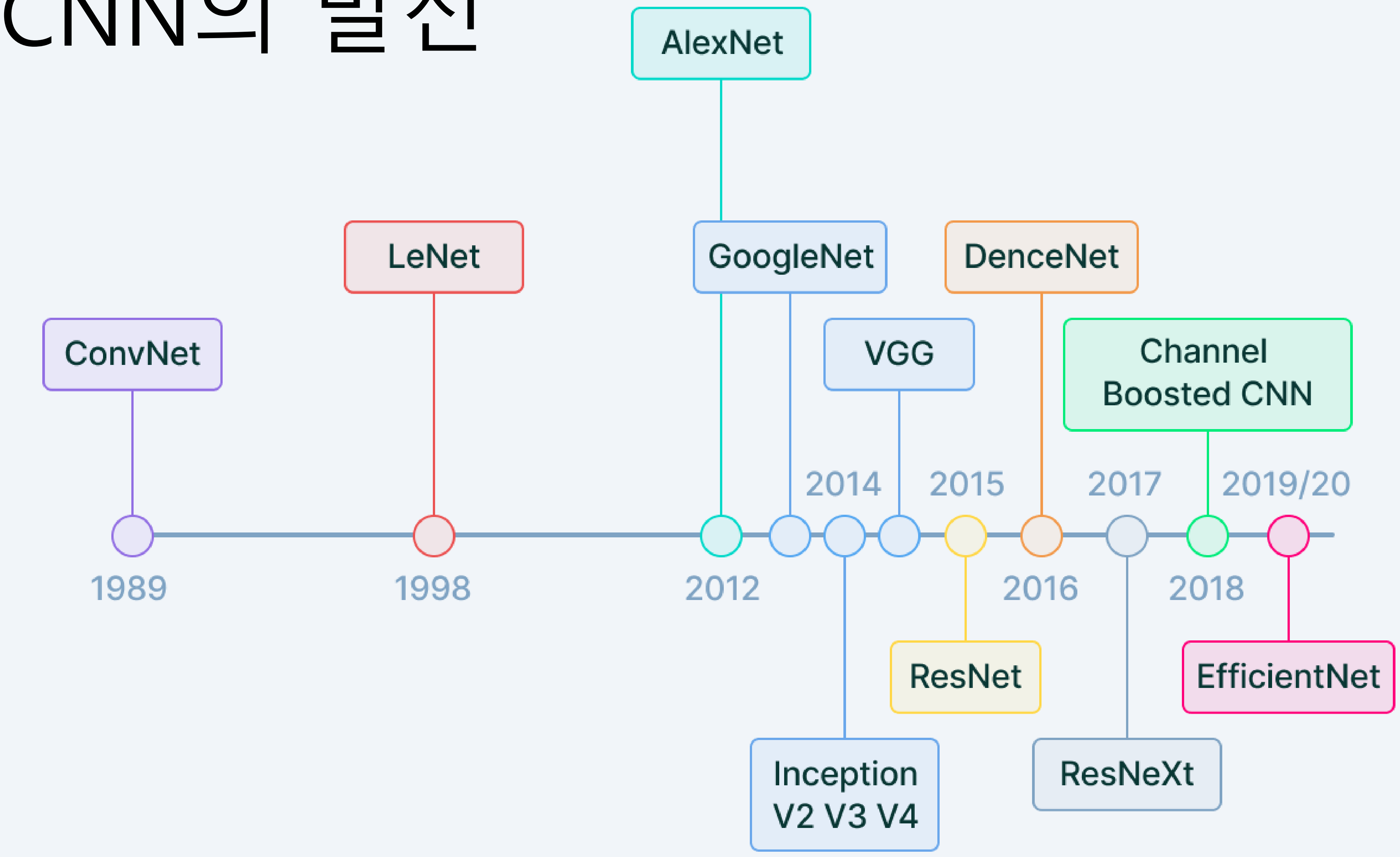
CNN + Deep learning



CNN Basic structure

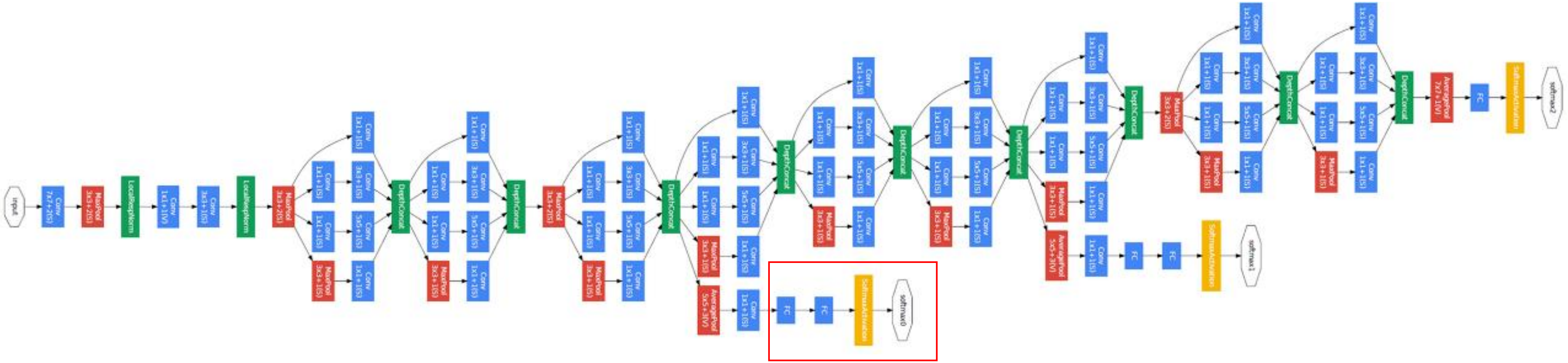


CNN의 발전

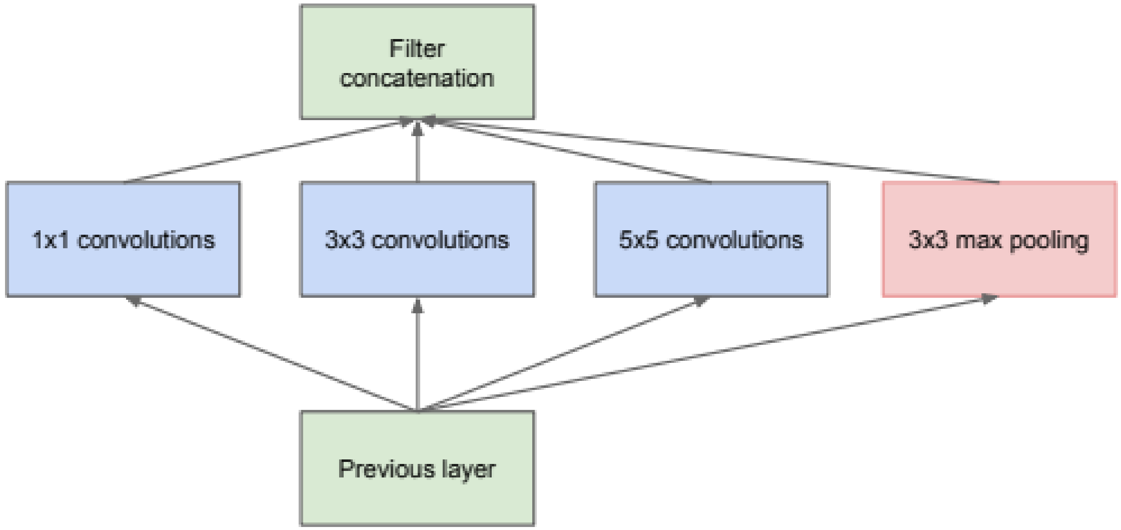


V7 Labs

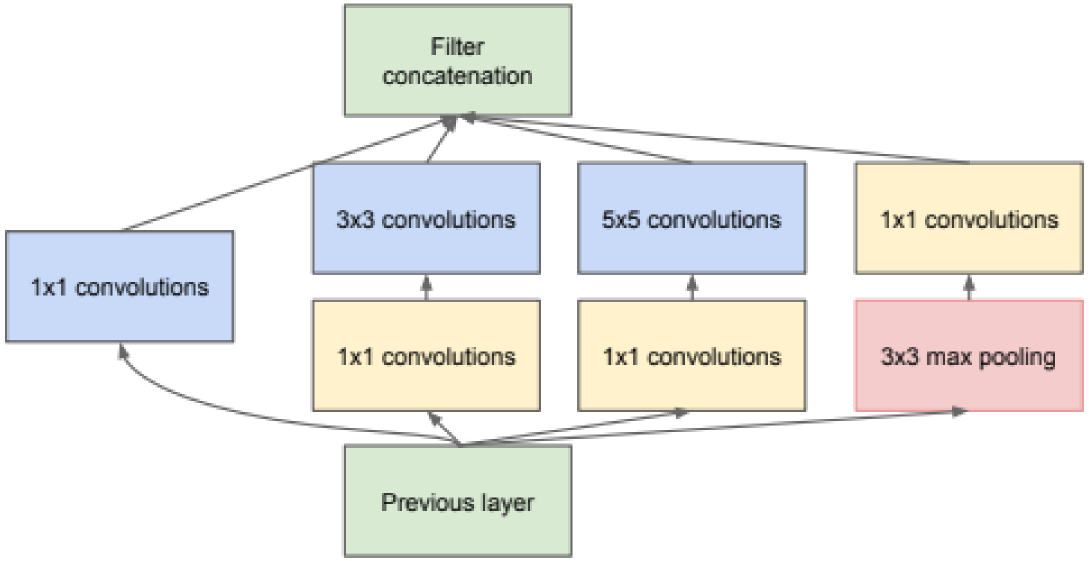
GoogleNet


input
Conv 7x7+2(S)
MaxPool 3x3+2(S)
LocalRespNorm
Conv 1x1+1(V)
Conv 3x3+1(S)
LocalRespNorm
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
AveragePool 5x5+3(V)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
FC
FC
SoftmaxActivation
softmax0
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
AveragePool 5x5+3(V)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
FC
FC
SoftmaxActivation
softmax1
DepthConcat
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
AveragePool 7x7+1(V)
FC
SoftmaxActivation
softmax2

GoogleNet : Inception module

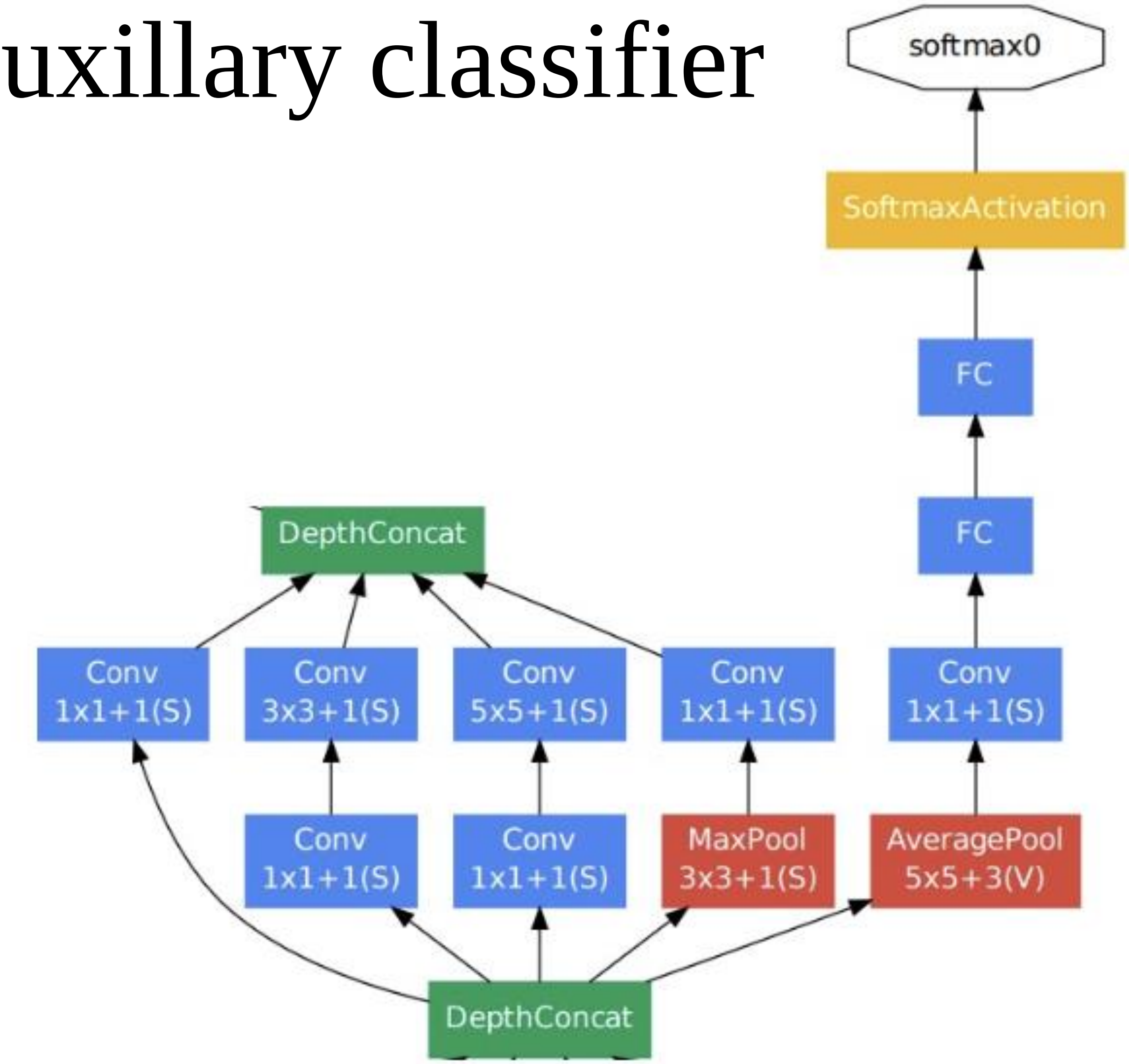


(a) Inception module, naïve version



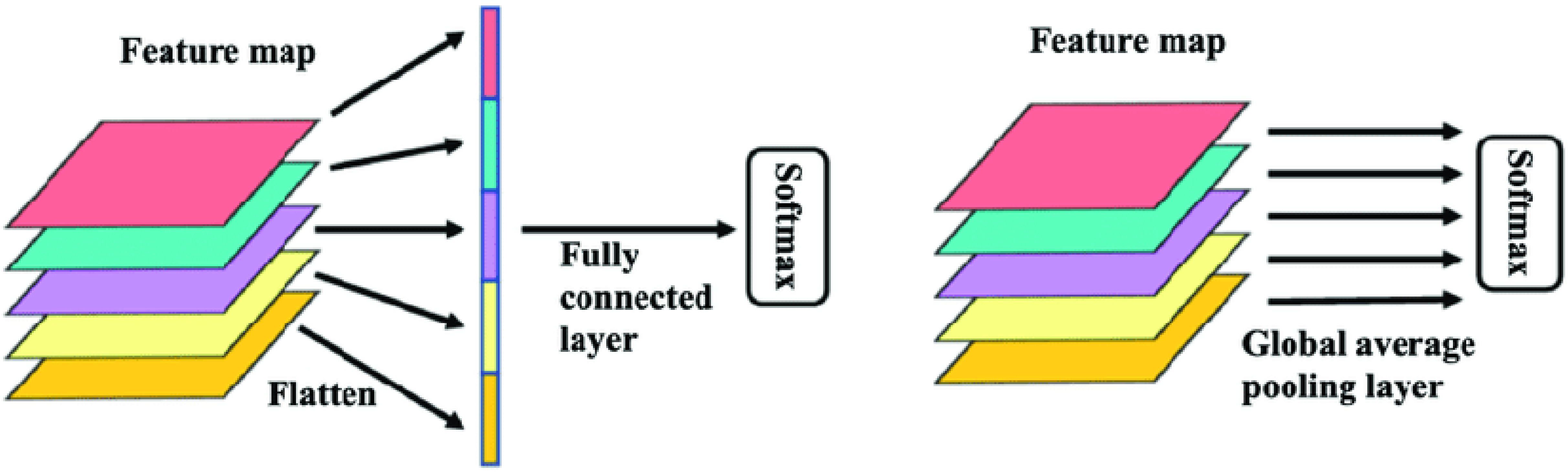
(b) Inception module with dimension reductions

GoogleNet : Auxillary classifier



Auxiliary classifier

GoogleNet : global average pooling



(a) Fully connected layer

(b) Global average pooling layer

Model Size vs. Accuracy Comparison

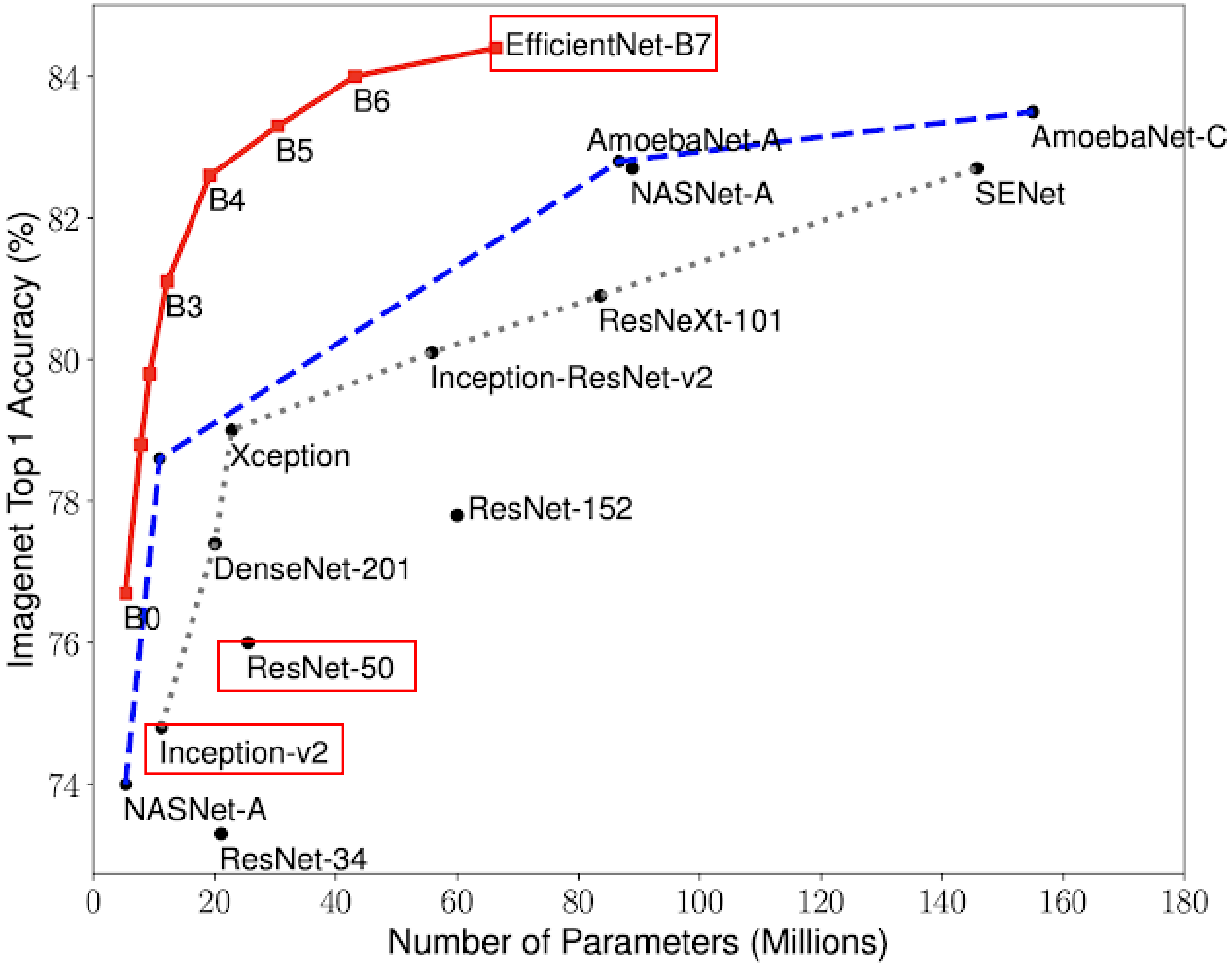
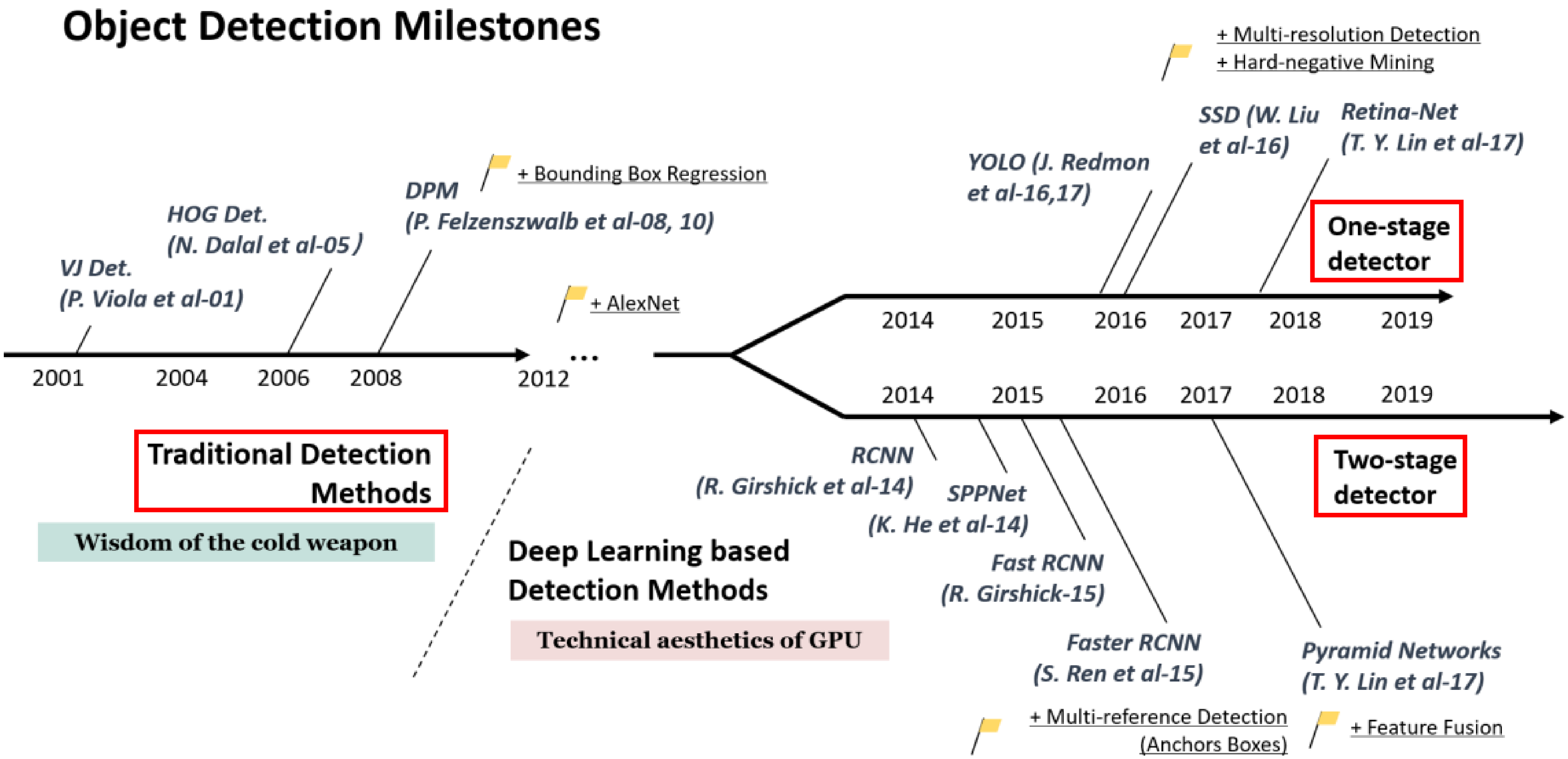


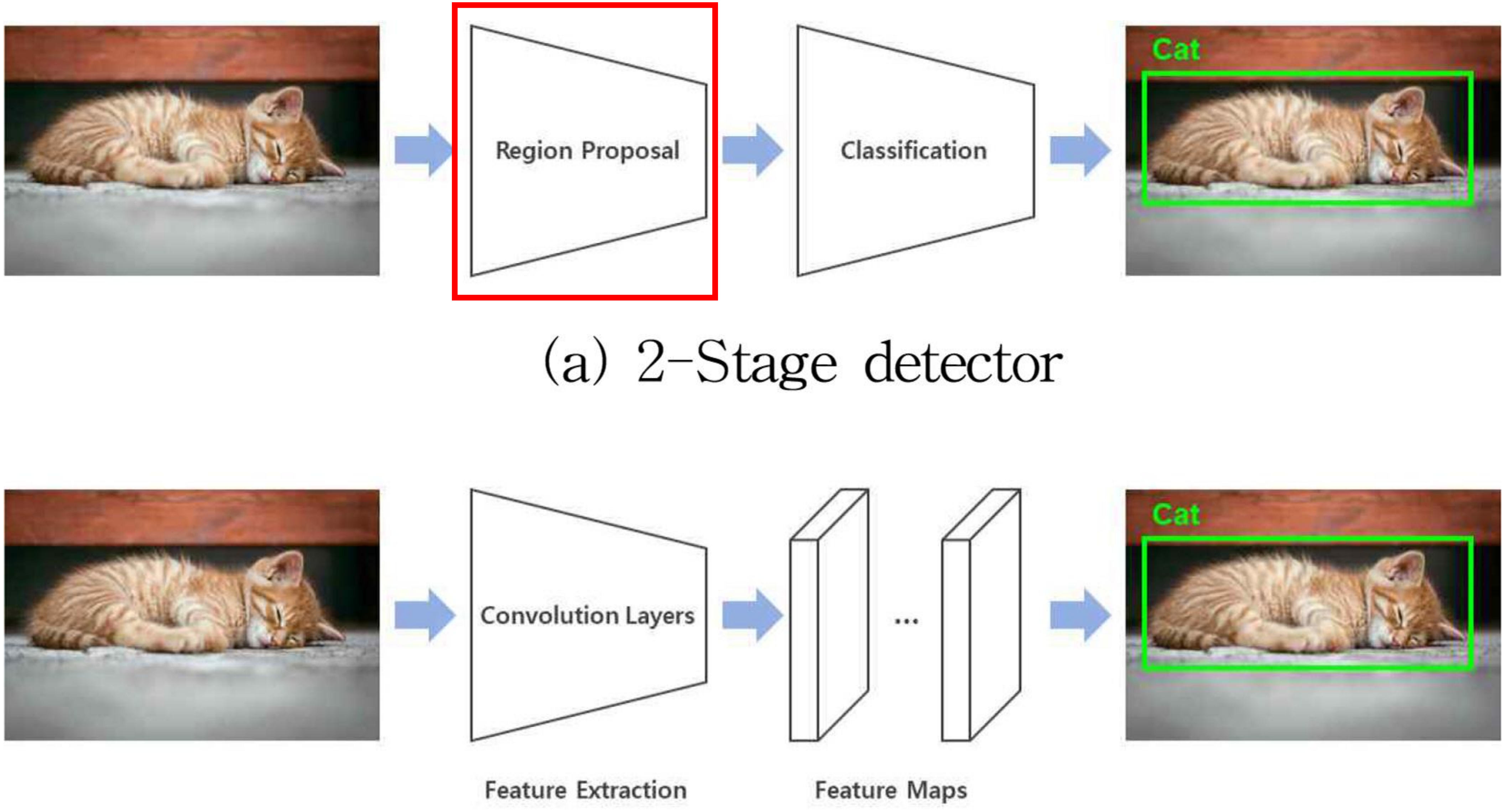
Image Source: Google AI Blog

Pytorch 실습

Object detection Milestones



Zou et al. 2019. Object Detection in 20 Years: A Survey



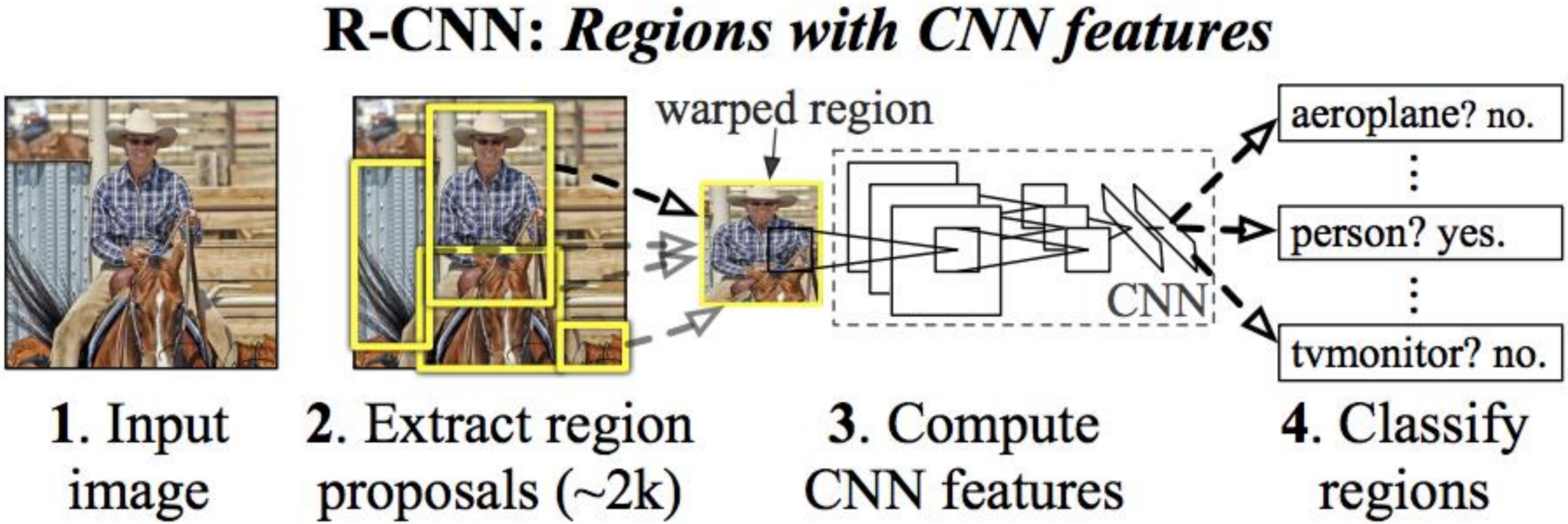
(a) 2-Stage detector

(b) 1-Stage detector

Stage에 따른 모델분류

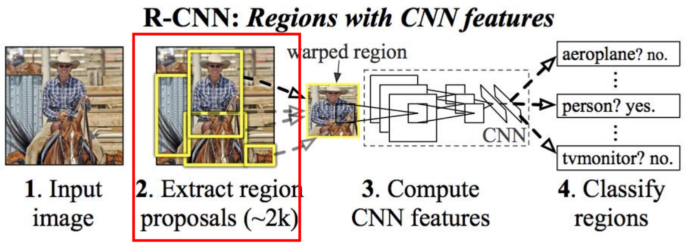
- Two stage detector
  - R-CNN (2014)
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN

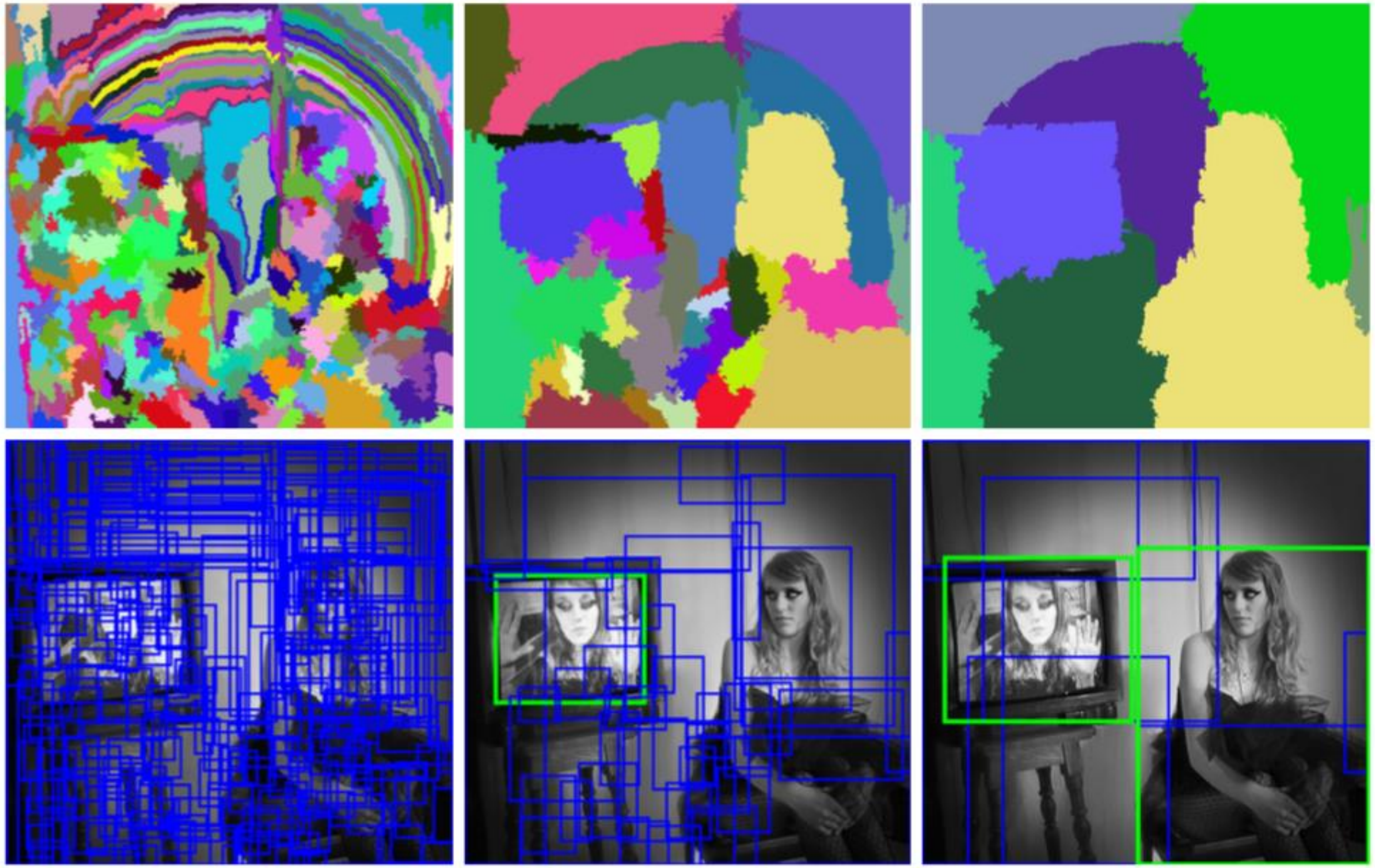
- One stage detector
  - Yolo series
    - Yolo 1 – Yolo9
  - SSD series
    - SSD
    - RetinaNet


R-CNN: *Regions with CNN features*
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

[Paper] Rich feature hierarchies for accurate object detection and semantic segmentation

R-CNN: Region proposal





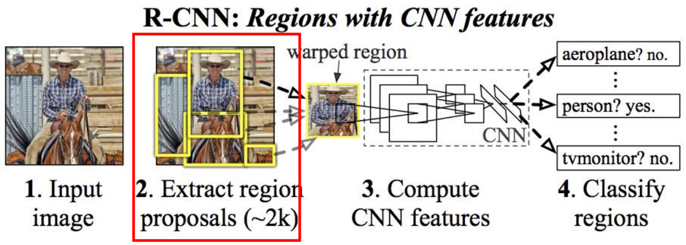
Selective search:

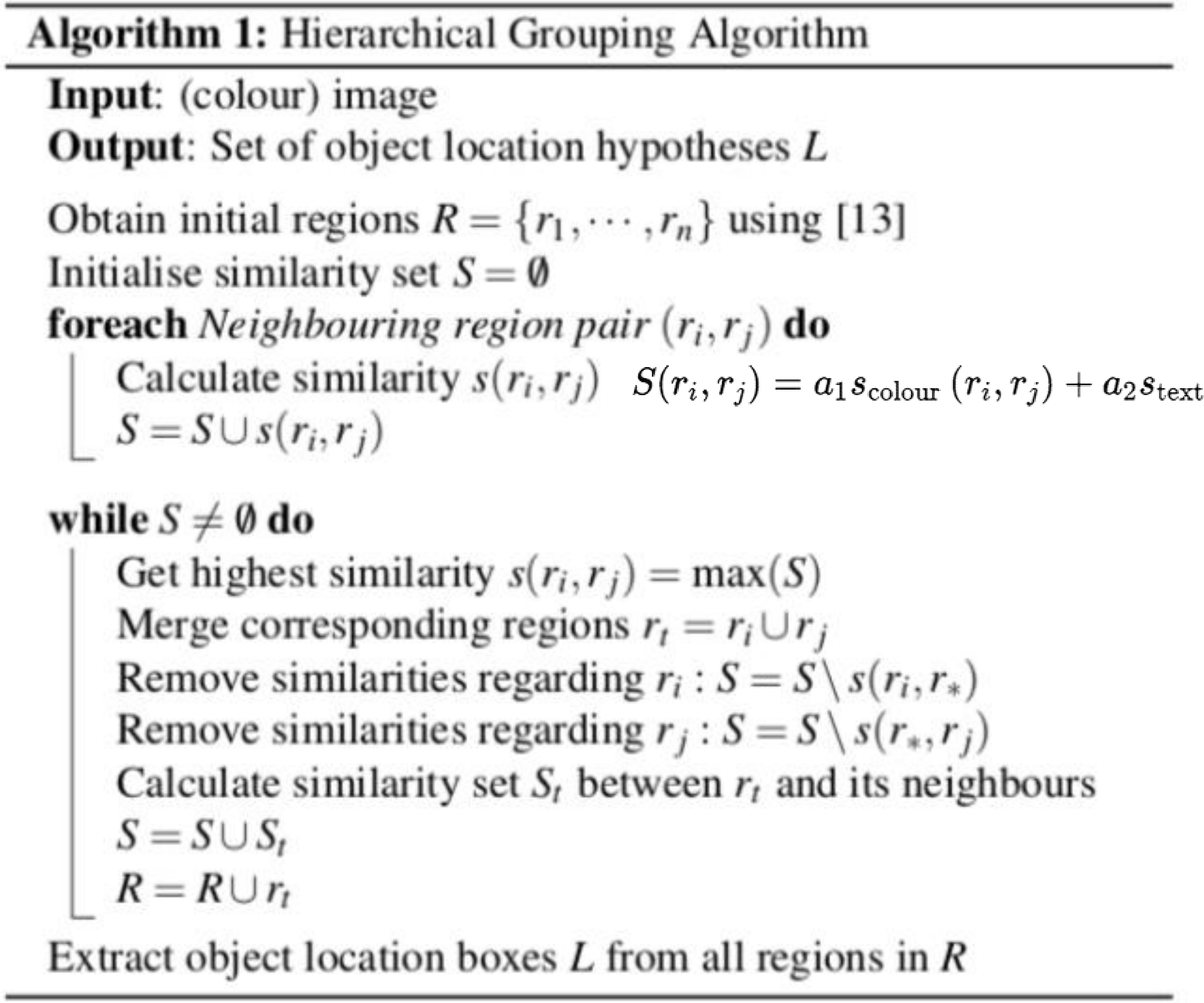
1. Image segmentation
   - Graph-based segmentation

2. Merge regions
   - Greedy algorithm

3. Candidate object location

R-CNN: Selective search


R-CNN: *Regions with CNN features*
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

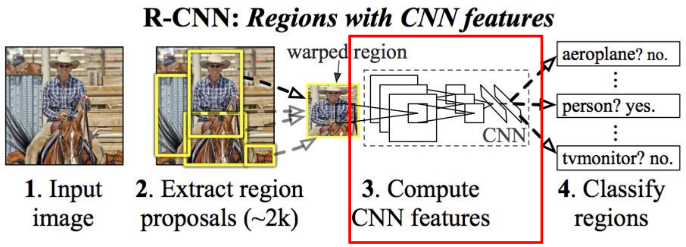


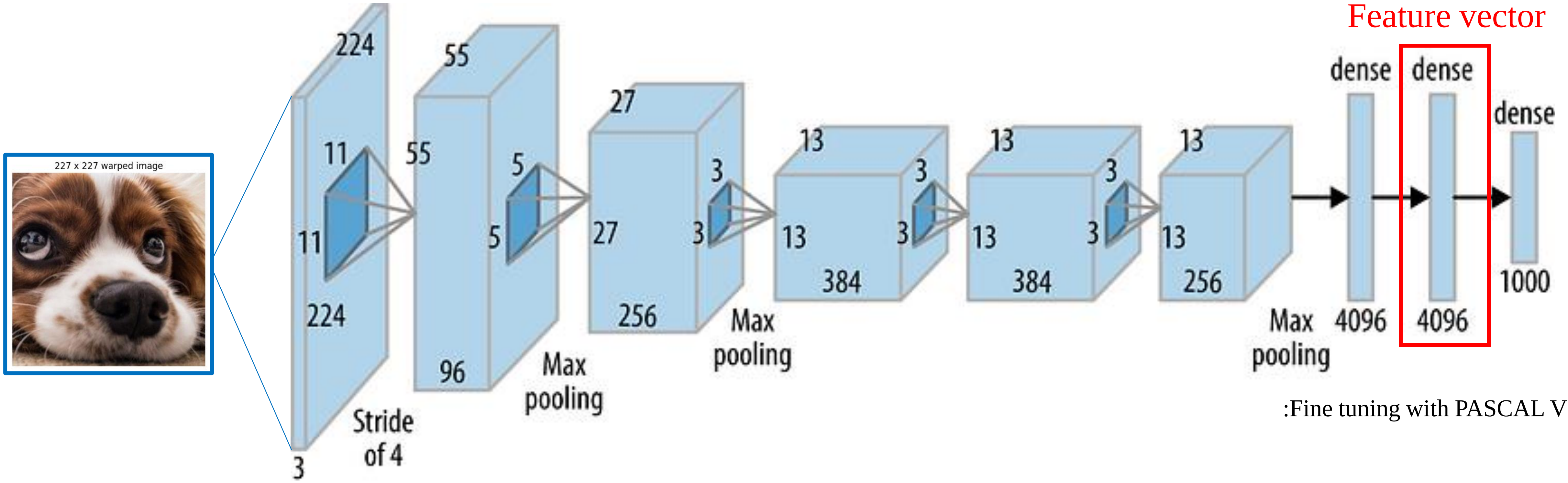
Algorithm 1: Hierarchical Grouping Algorithm

```
Input: (colour) image
Output: Set of object location hypotheses $L$

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach Neighbouring region pair $(r_i, r_j)$ do
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ do
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set $S_t$ between $r_t$ and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

Extract object location boxes $L$ from all regions in $R$
```

$$S(r_i, r_j) = a_1 s_{\text{colour}}(r_i, r_j) + a_2 s_{\text{texture}}(r_i, r_j) + a_3 s_{\text{size}}(r_i, r_j) + a_4 s_{\text{fill}}(r_i, r_j)$$

R-CNN: Feature extraction


R-CNN: *Regions with CNN features*
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

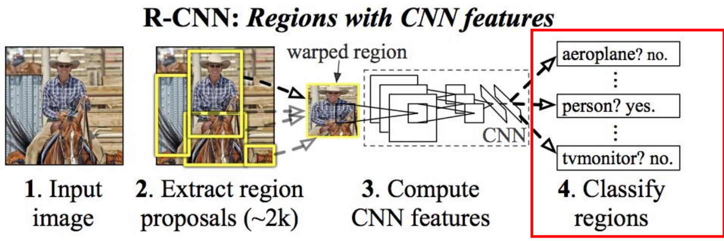


Feature vector

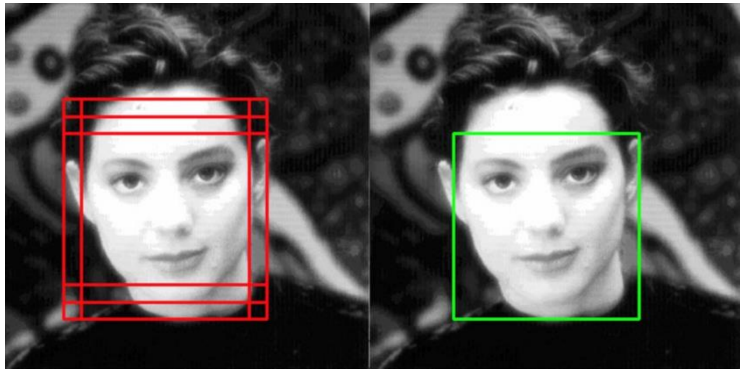
:Fine tuning with PASCAL VOC

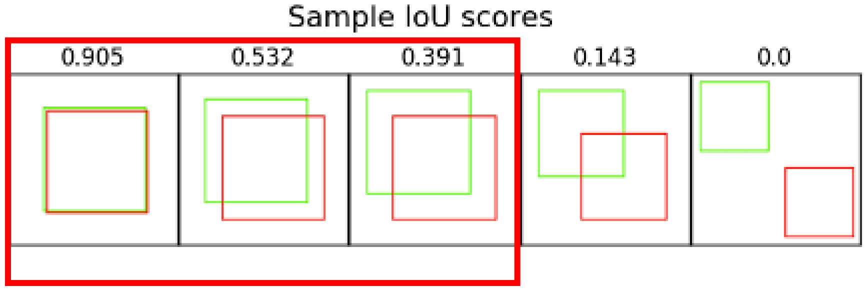
Alexnet (2012)

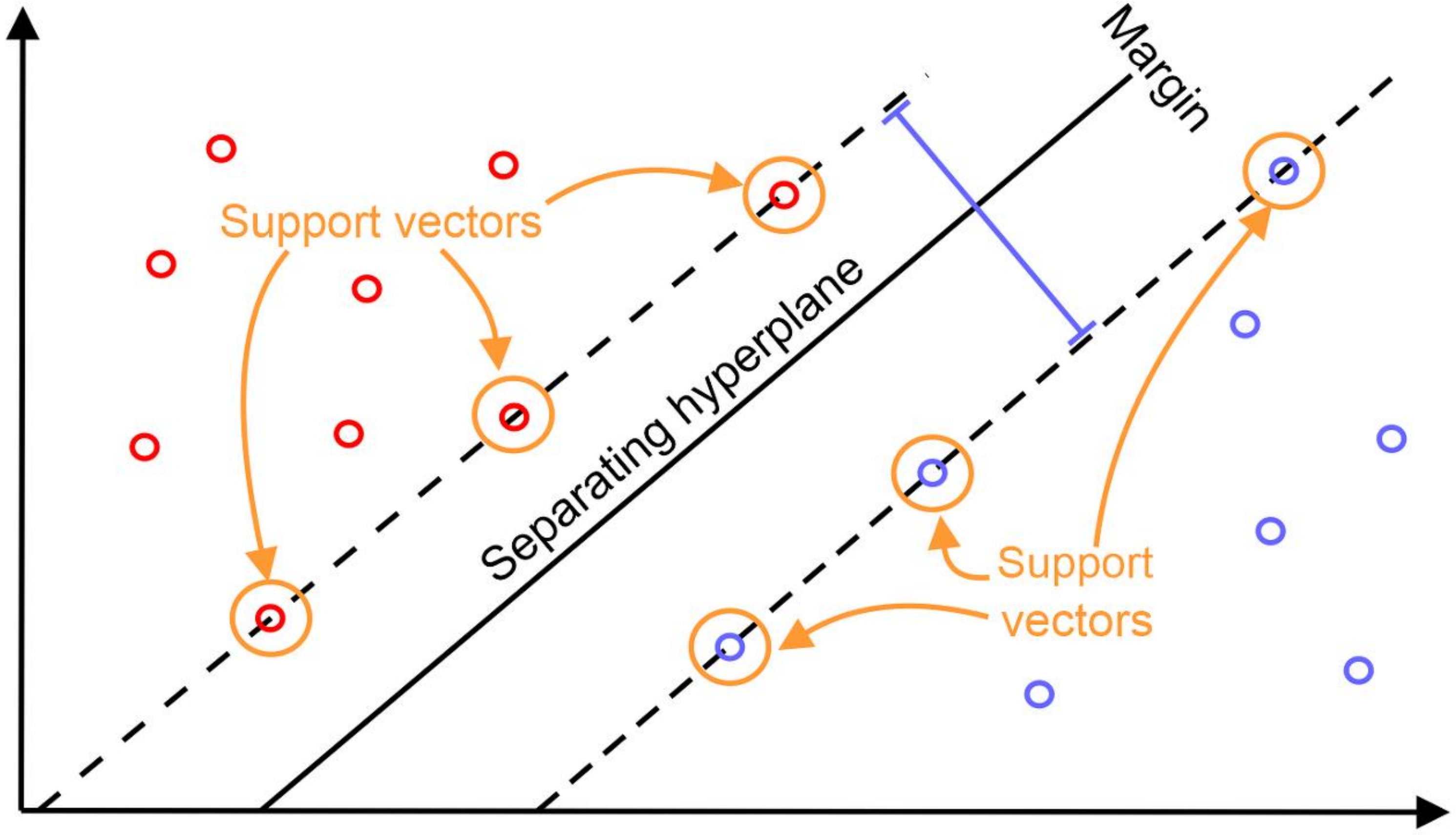
R-CNN: SVM Classification


R-CNN: *Regions with CNN features*
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

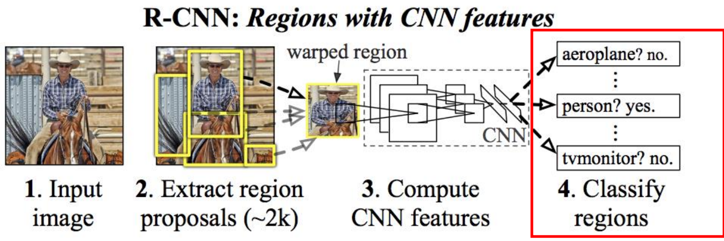
Non Maximum suppression







R-CNN: Bounding box regression



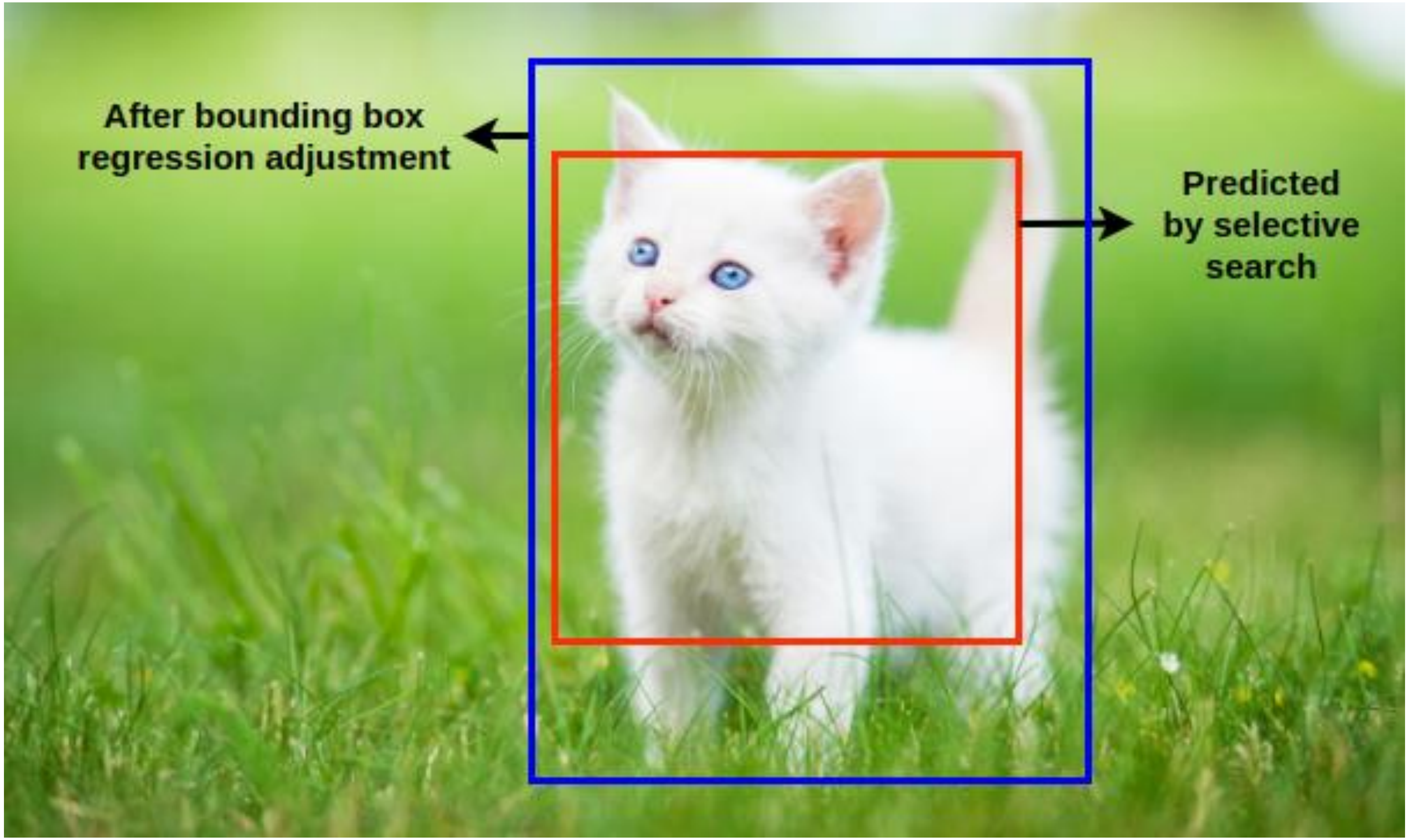
R-CNN: *Regions with CNN features*

1. Input image

2. Extract region proposals (~2k)

3. Compute CNN features

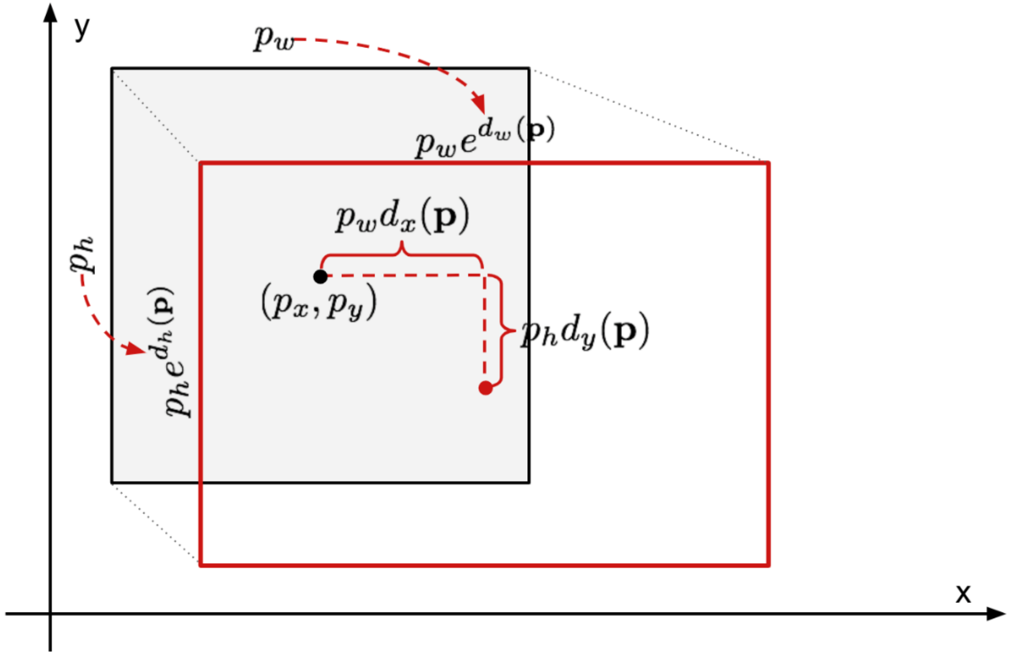
4. Classify regions



After bounding box regression adjustment

Predicted by selective search

R-CNN: Bounding box regression

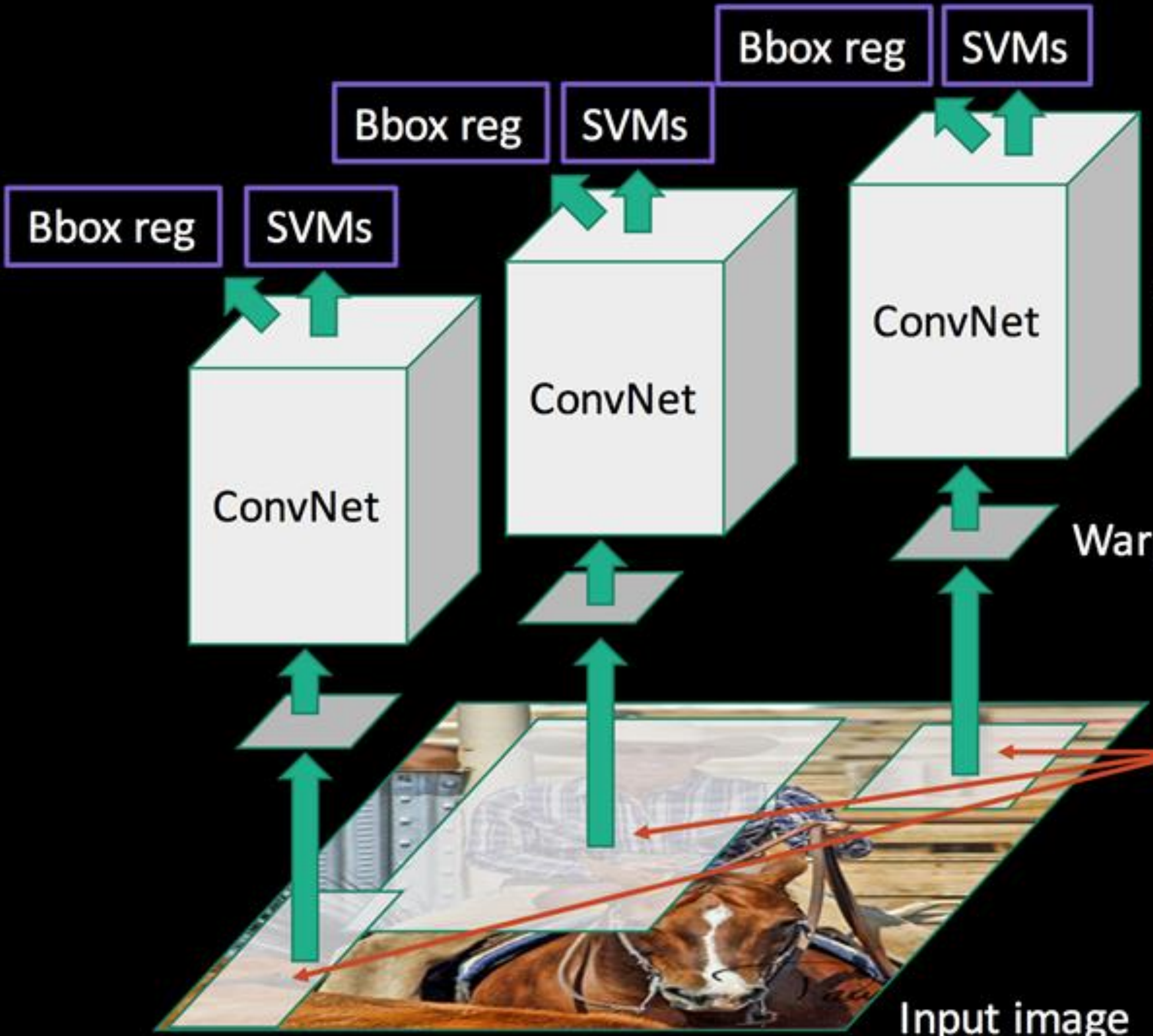


$$\hat{g}_x = p_w d_x(\mathbf{p}) + p_x$$
$$\hat{g}_y = p_h d_y(\mathbf{p}) + p_y$$
$$\hat{g}_w = p_w \exp(d_w(\mathbf{p}))$$
$$\hat{g}_h = p_h \exp(d_h(\mathbf{p}))$$

$$t_x = (g_x - p_x)/p_w$$
$$t_y = (g_y - p_y)/p_h$$
$$t_w = \log(g_w/p_w)$$
$$t_h = \log(g_h/p_h)$$

Minimizing SSE loss =

$$\mathcal{L}_{\text{reg}} = \sum_{i \in \{x,y,w,h\}} (t_i - d_i(\mathbf{p}))^2 + \lambda \|\mathbf{w}\|^2$$



Apply bounding-box regressors

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Input image

Girshick et al. CVPR14.

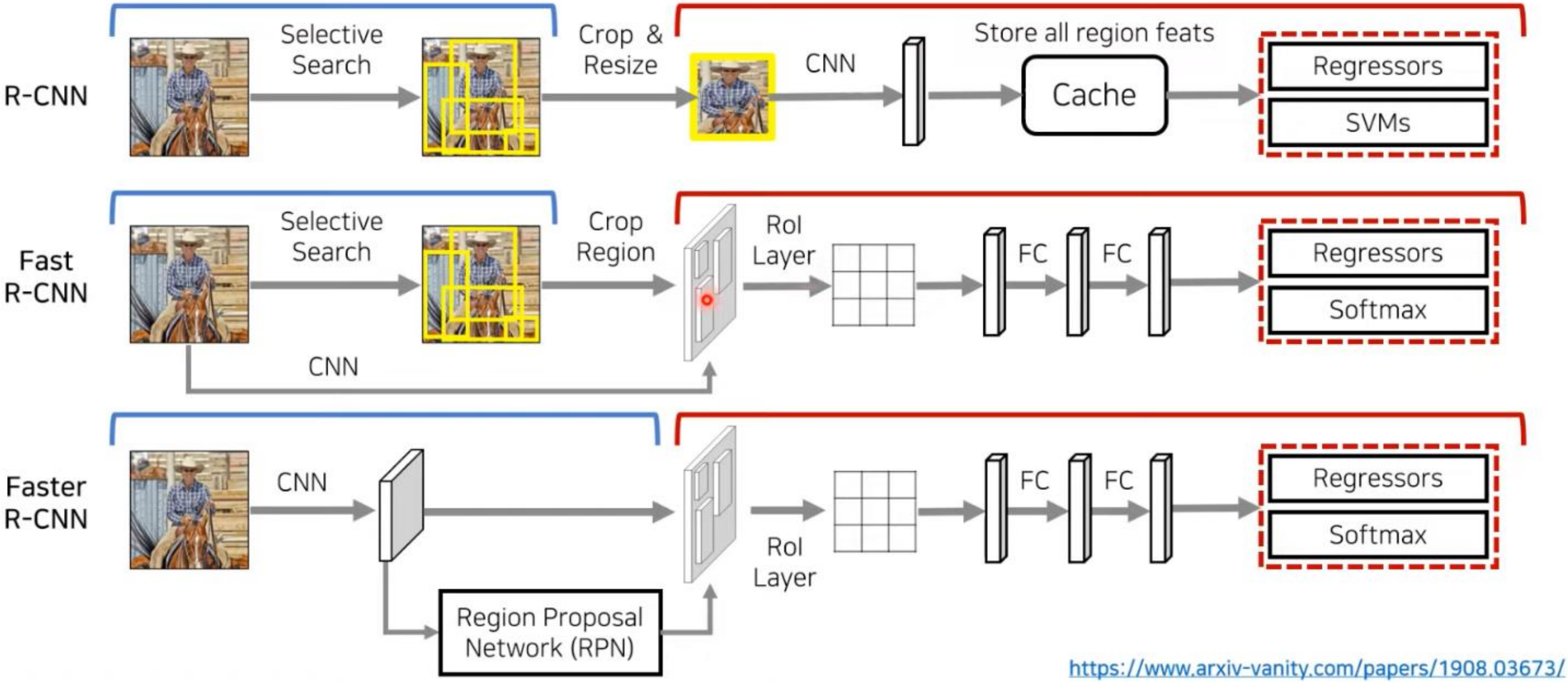
R-CNN의 장단점

- 높은 Detection 정확도

- 2000개씩 생성된 region 이미지를 CNN Feature map 생성
  - 너무 느림
  - 이미지 한 장당 50초 소요

- End-to-End 학습이 안됨
  - CNN Feature Extractor, SVM, Bounding box regressor

R-CNN, Fast R-CNN, Faster R-CNN



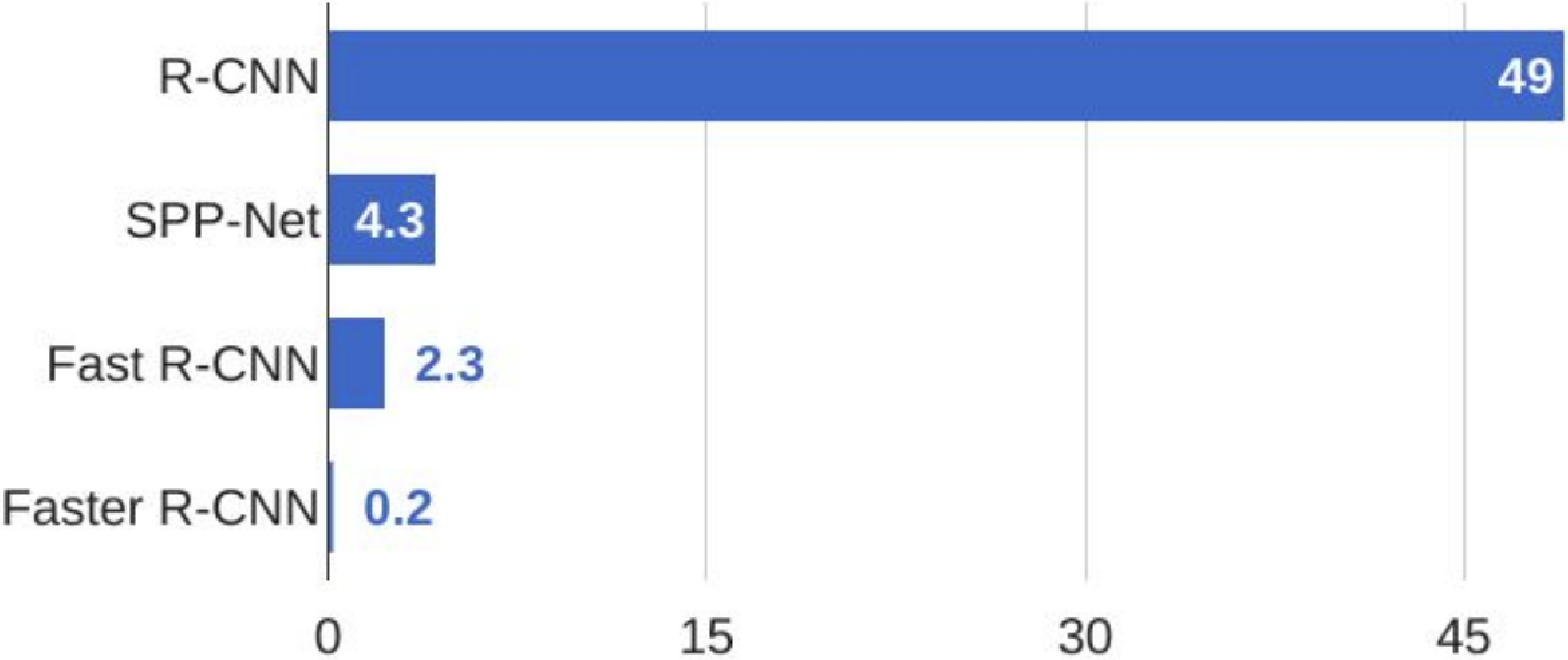
https://www.arxiv-vanity.com/papers/1908.03673/

R-CNN 계열 성능비교

System	Time	07 data	07 + 12 data
R-CNN	~ 50s	66.0	-
Fast R-CNN	~ 2s	66.9	70.0
Faster R-CNN	**~ 198ms**	**69.9**	**73.2**

Detection mAP on PASCAL VOC 2007 and 2012, with VGG-16 pre-trained on ImageNet Dataset

R-CNN Test-Time Speed


R-CNN
SPP-Net
Fast R-CNN
Faster R-CNN