

DeepIntoDeep

Object Detection & Segmentation

발표자: 김민재

Object Detection

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

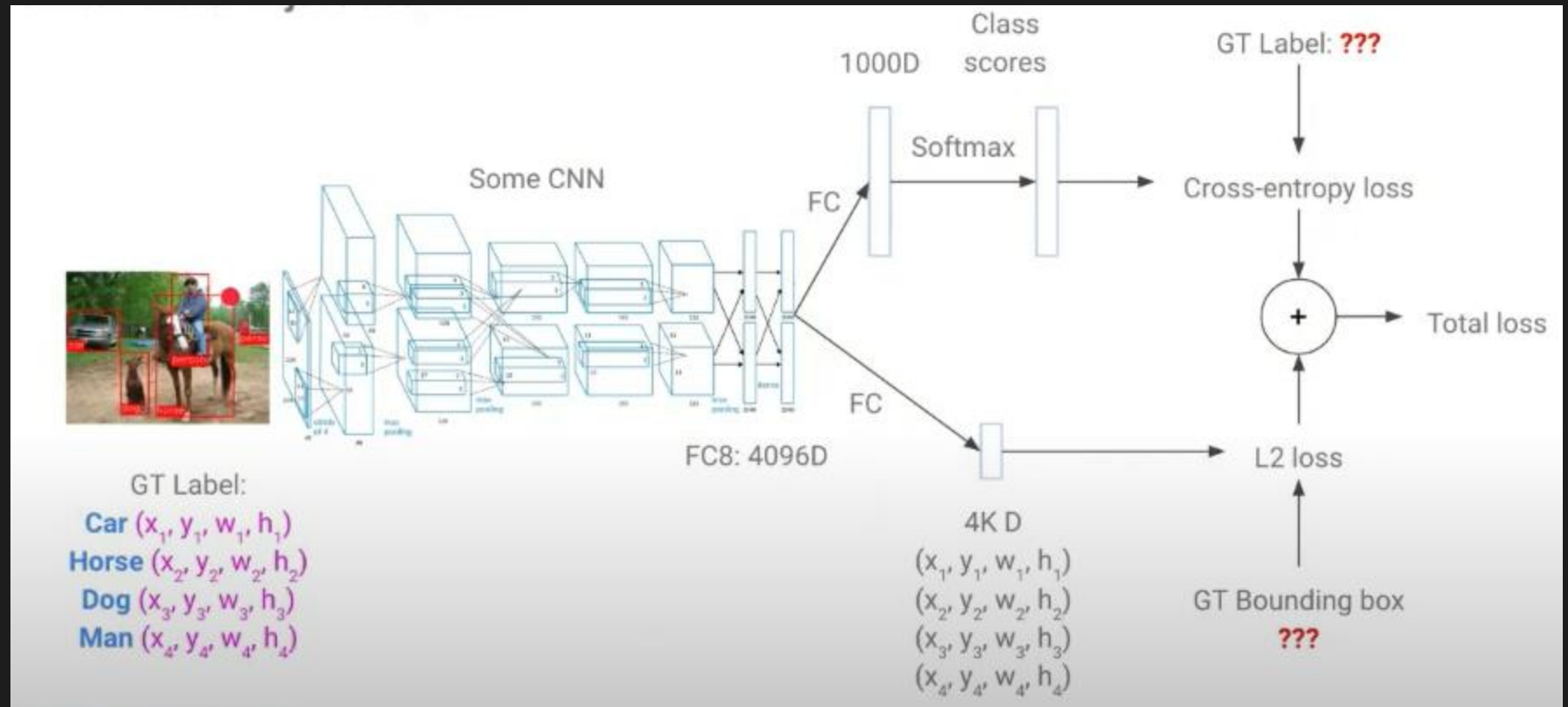
Object Detection

- Classification + Localization of objects in the image
- Expected outputs : a list of entities with
 - Class
 - Bounding box
 - Confidence score



First Idea for Object Detection


- Each image may have different number of objects.
- There is no natural order among multiple objects.

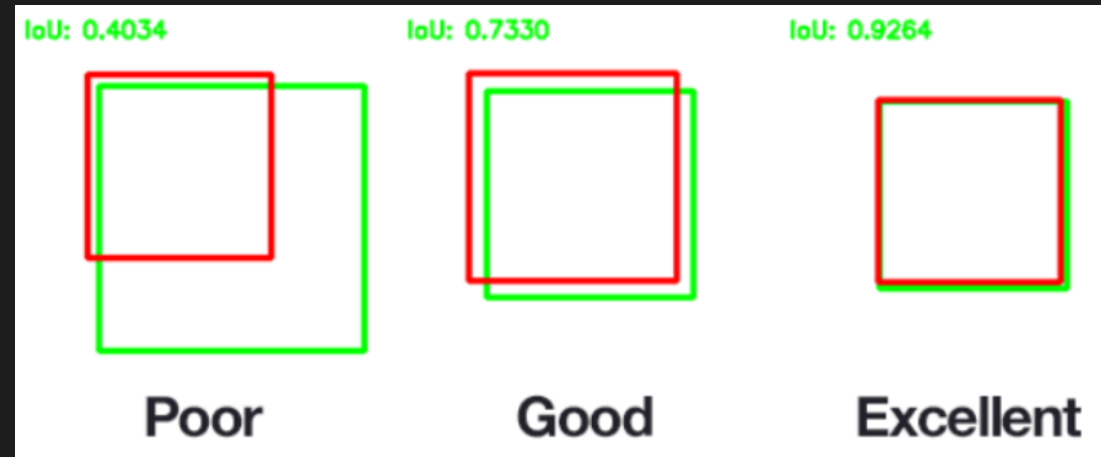


Object Detection Approaches

- Proposal-based models
 - Two-stage model
 - Consists of region proposal module and a recognition module
 - R-CNN (CVPR 2014), Fast R-CNN (ICCV 2015), Faster R-CNN (NIPS 2015), etc
- Proposal-free models
 - Single-stage model
 - Removes the proposal generating module and predicts object positions directly
 - YOLO (CVPR 2016), SSD (ECCV 2016), DETR (ECCV 2020), etc

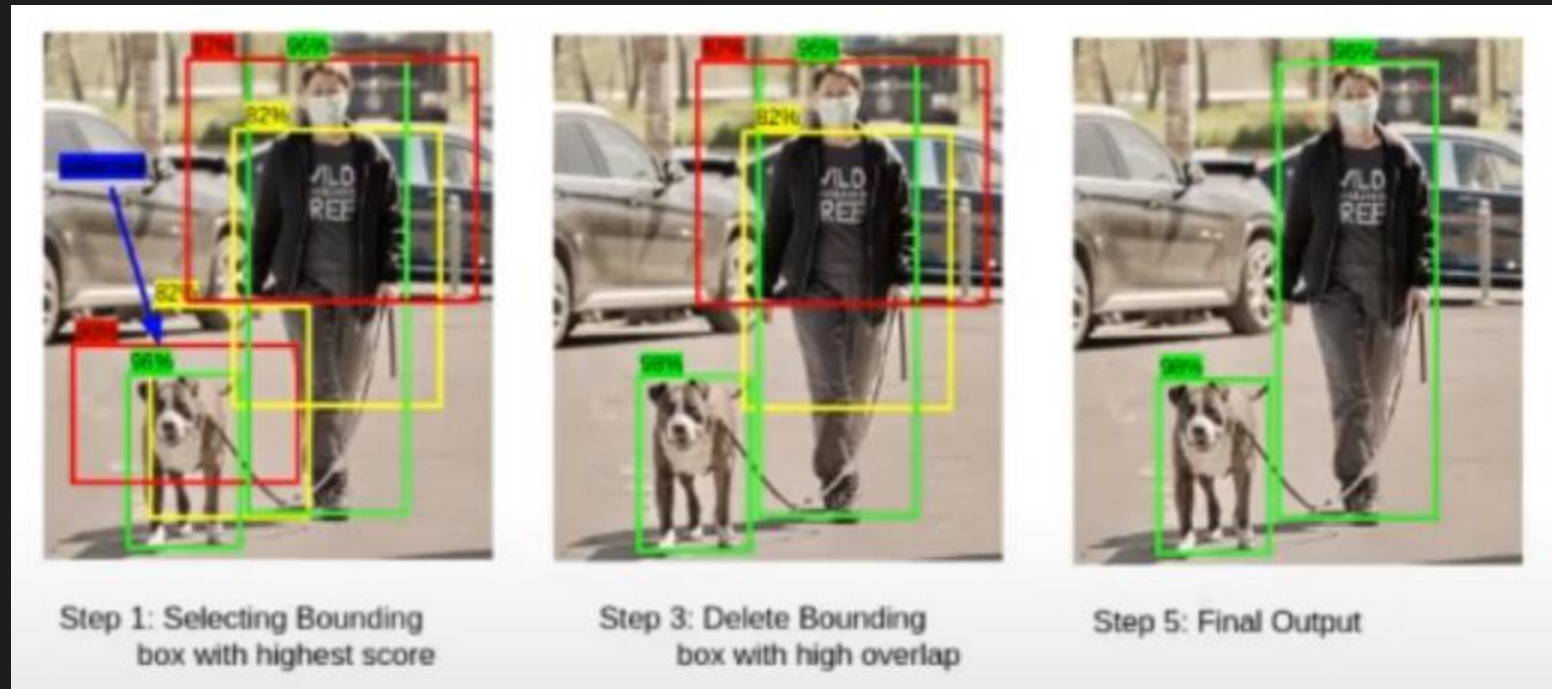
IoU : Intersection over Union

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$




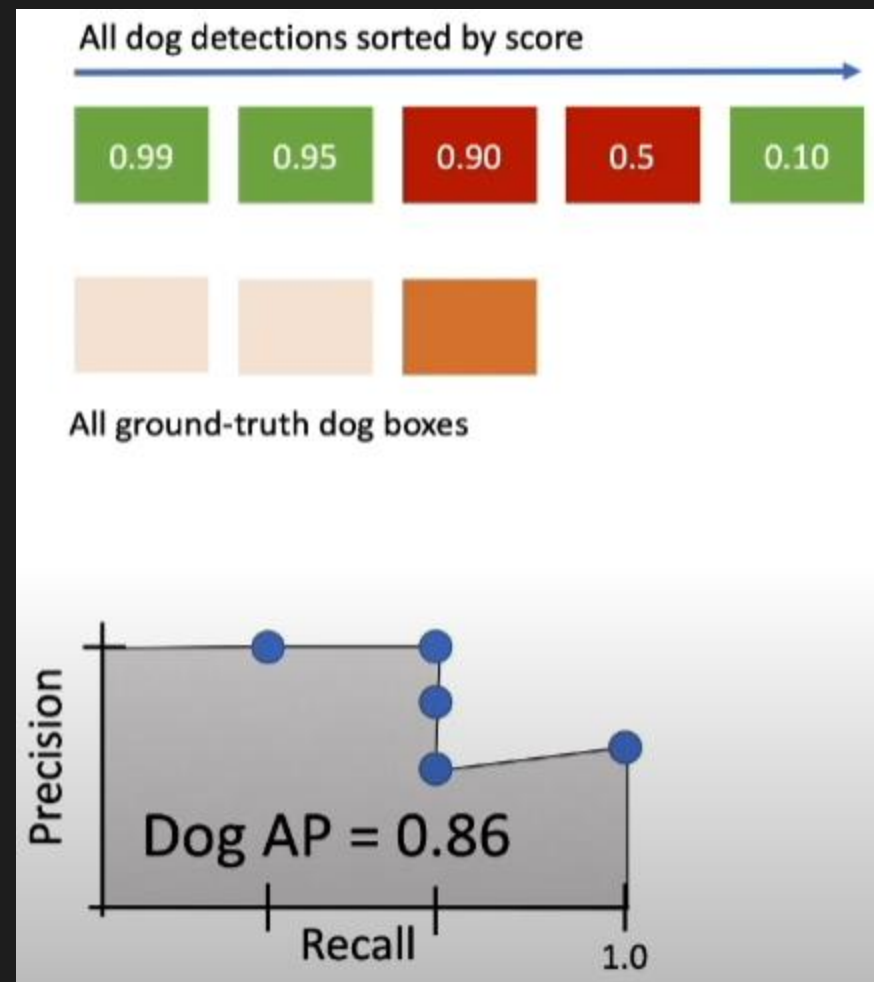
NMS : Non-Maximum Suppression

1. Select the box with the highest objectiveness score
2. Compute IoU between this box and all other boxes
3. Remove the bounding boxes with $\text{IoU} > \text{threshold}$
4. Move to the next highest objectiveness score
5. Repeat step 2-4



mAP : mean Average Precision

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision = area under Precision vs Recall Curve
 - For each detection (highest score to lowest)
 - If it matches some GT box with $\text{IoU} > \text{threshold}$, mark it as positive and eliminate the GT
 - Otherwise mark it as negative
 - Plot a point on PR Curve
 - Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category
4. For COCO mAP, compute mAP@thresh for each IoU threshold (0.5-0.95) and take average

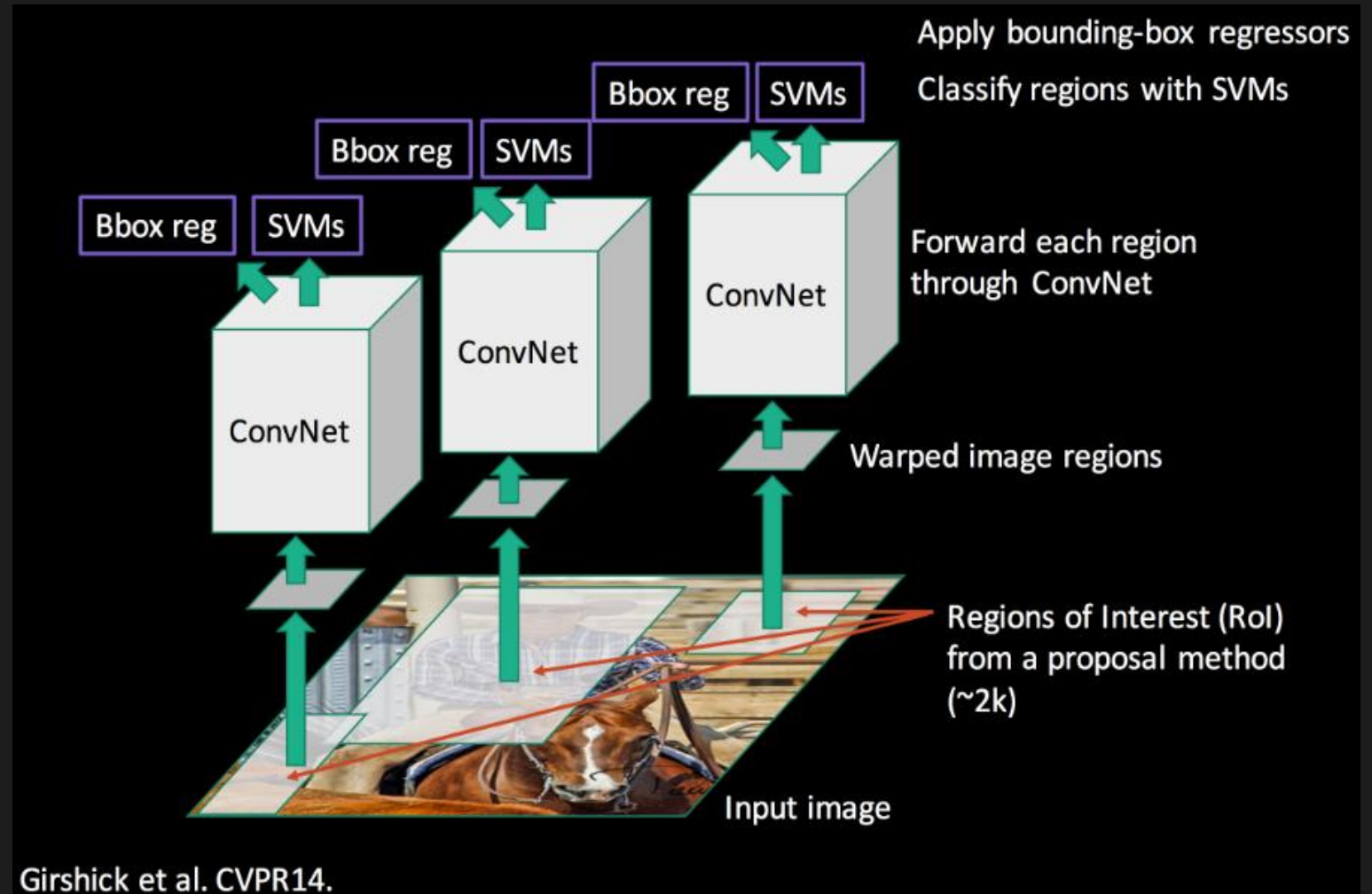


R-CNN

- Two-stage model : region proposal + recognition
- Region proposal
 - As region proposal wasn't that great at that time, they had to produce large number of proposals (~2000/image)
 - Selective search, EdgeBoxes, MCG...
- Object recognition
 - Any CNN, classifier can be used

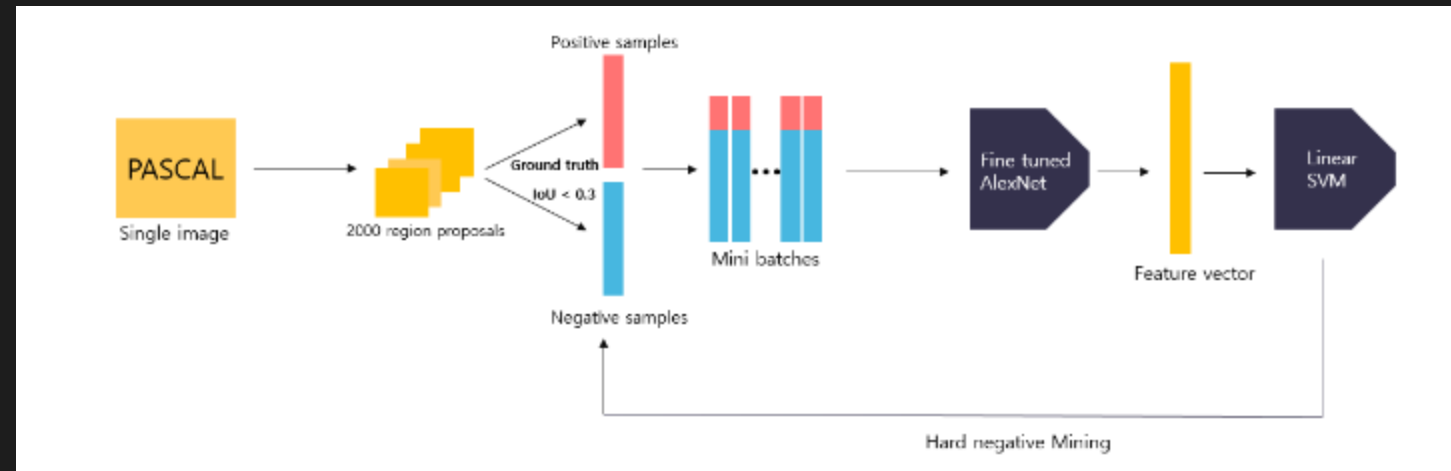
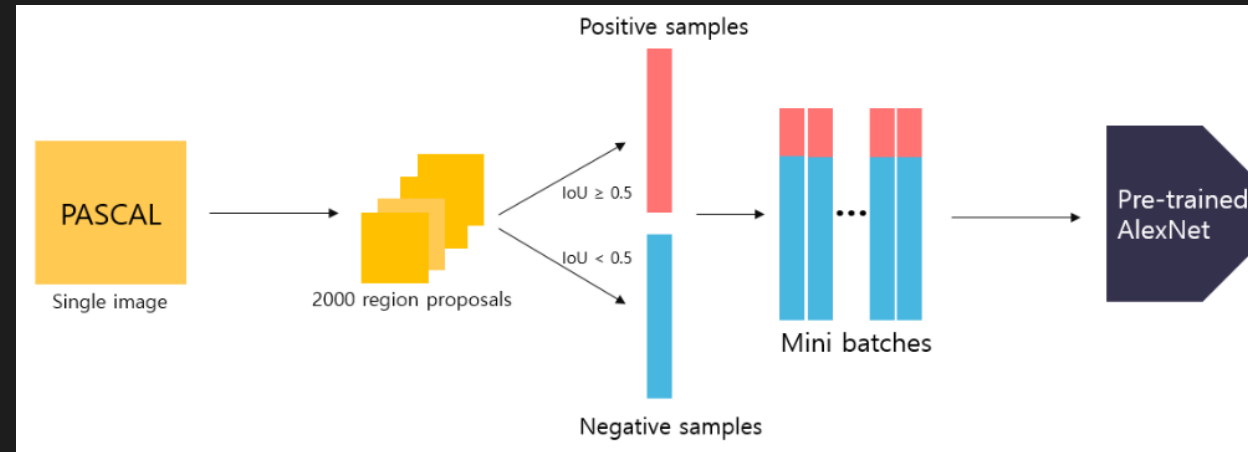
R-CNN : Architecture

1. Extract region proposals by selective search (~2k/image)
2. Resize regions
3. Extract CNN features
4. Classify & refine regions



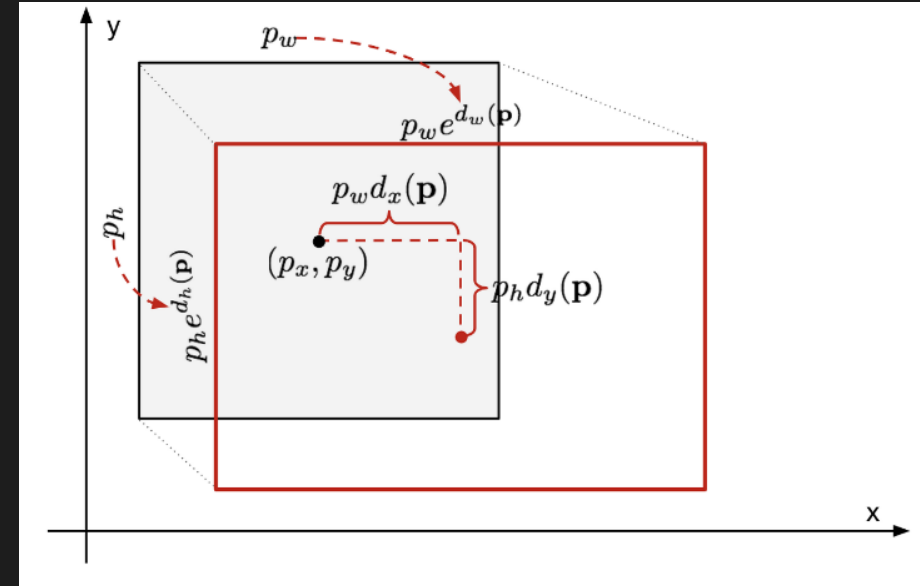
R-CNN : Fine-tuning & Classification

- Construct mini-batch with positive/negative examples and IoU threshold 0.5
- First, train N+1 independent linear SVM
- Apply Hard Negative Mining after 1 epoch



R-CNN : Bounding Box Regression

Given pool5 feature of a proposal \mathbf{p} , its bounding box coordinate \mathbf{P} and its corresponding GT box coordinate \mathbf{G} ,
the regressor learns **scale-invariant transformation** between two centers and **log-scale transformation** between width and heights.



$$\hat{G}_x = P_w d_x(P) + P_x \quad (1)$$

$$\hat{G}_y = P_h d_y(P) + P_y \quad (2)$$

$$\hat{G}_w = P_w \exp(d_w(P)) \quad (3)$$

$$\hat{G}_h = P_h \exp(d_h(P)). \quad (4)$$

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^T \phi_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2. \quad (5)$$

$$d_\star(P) = \hat{\mathbf{w}}_\star^T \phi_5(P)$$

$$t_x = (G_x - P_x) / P_w \quad (6)$$

$$t_y = (G_y - P_y) / P_h \quad (7)$$

$$t_w = \log(G_w / P_w) \quad (8)$$

$$t_h = \log(G_h / P_h). \quad (9)$$

R-CNN : Summary

Contribution

- First deep learning based object detection
- Significantly reduced computation from the brute-force method
- Assuming that the region proposal works well, detection performs okay

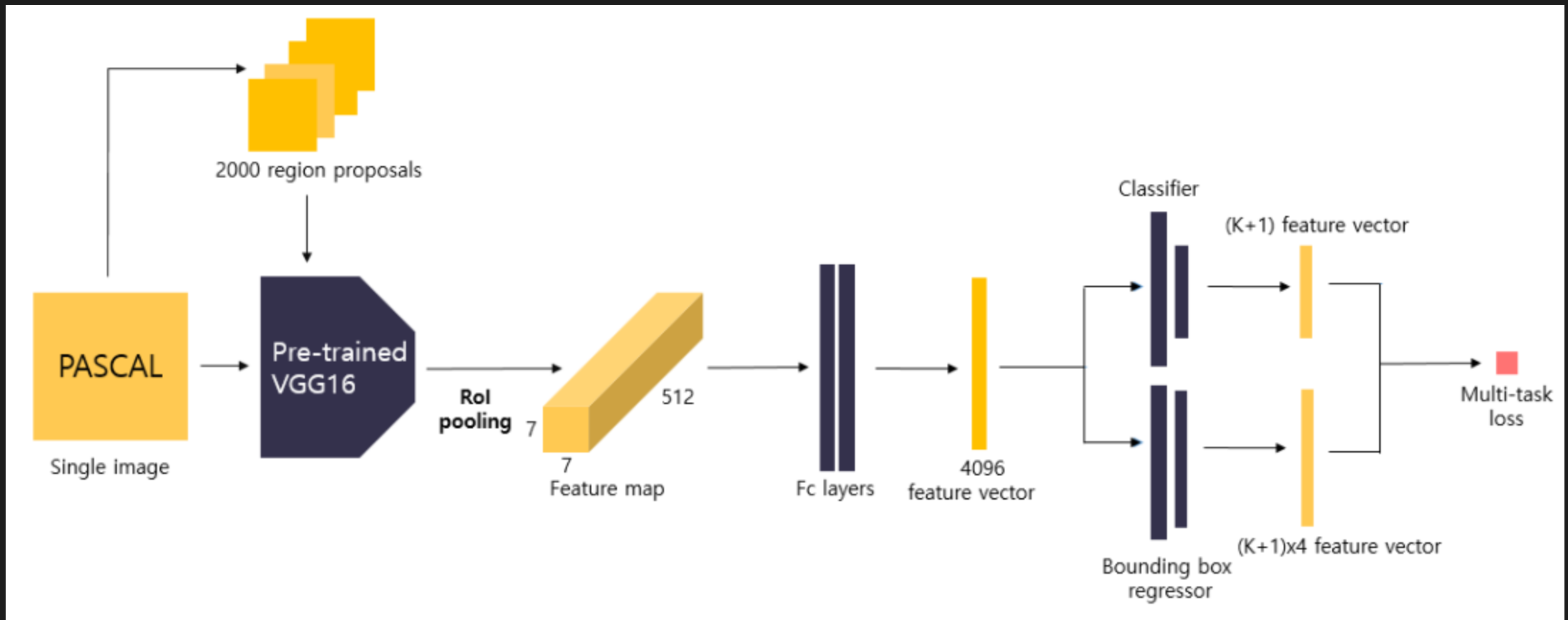
Limitation

- Computationally very expensive. (~2000 independent forward passes)
- Inefficient training process

Fast R-CNN

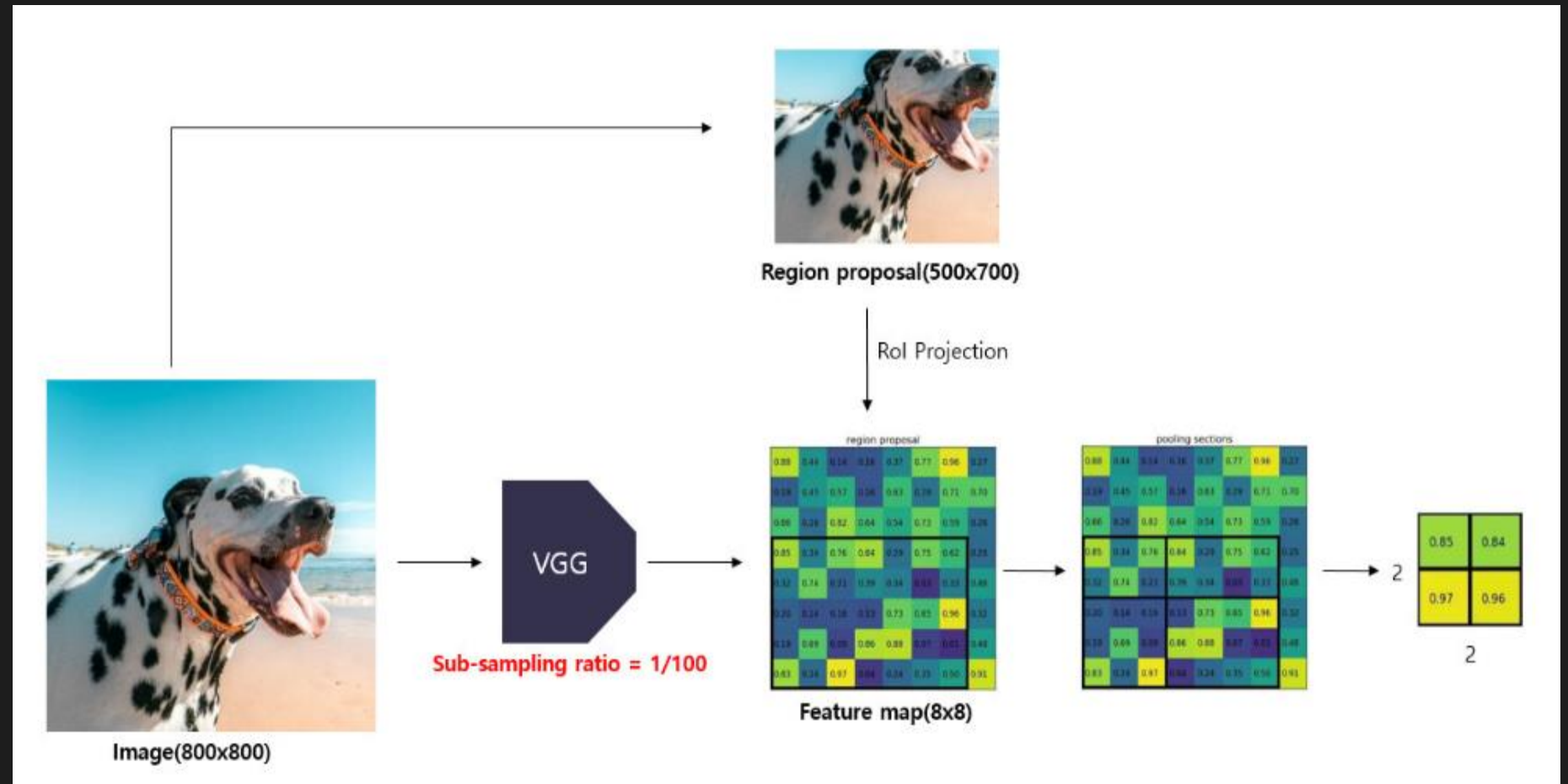
- Try to reduce the main bottleneck of R-CNN:
 - Instead of independently performing ~2000 forward passes, Fast R-CNN **extract CNN features only once** with the entire image, then **take the region features on the conv feature space**.
- Region proposal
 - Same as R-CNN
- Object recognition
 - Any CNN, classifier can be used
 - With **RoI pooling** and max pooling, make pooled region features to have same dimensionality

Fast R-CNN : Architecture



Fast R-CNN : RoI Pooling

- Project proposal onto features, then align to grid cells
- Extract fixed size region features by max pooling for each grid cell
- 7x7 or 14x14 grid in practice



Fast R-CNN : Multi-task loss

- Ignore background using index parameter u
- Use L1 loss to make model robust to outlier
- Multi-task loss results in 0.8-1.1% mAP improvement

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

$p = (p_0, \dots, p_k) : (K+1)$ 개의 class score

u : ground truth class score

$t^u = (t_x^u, t_y^u, t_w^u, t_h^u) :$ 예측한 bounding box 좌표를 조정하는 값

$v = (v_x, v_y, v_w, v_h) :$ 실제 bounding box의 좌표값

$L_{cls}(p, u) = -\log p_u :$ classification loss(**Log loss**)

$L_{loc}(t^u, v) = \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L_1}(t_i^u - v_i) :$ regression loss(**Smooth L1 loss**)

$$\text{smooth}_{L_1}(t_i^u - v_i) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases}$$

Fast R-CNN : Summary

Contribution

- Better accuracy than R-CNN
- 8-18x faster training time than R-CNN
- 80-213x faster inference time than R-CNN
- Much less memory space needed, since conv features are extracted only once

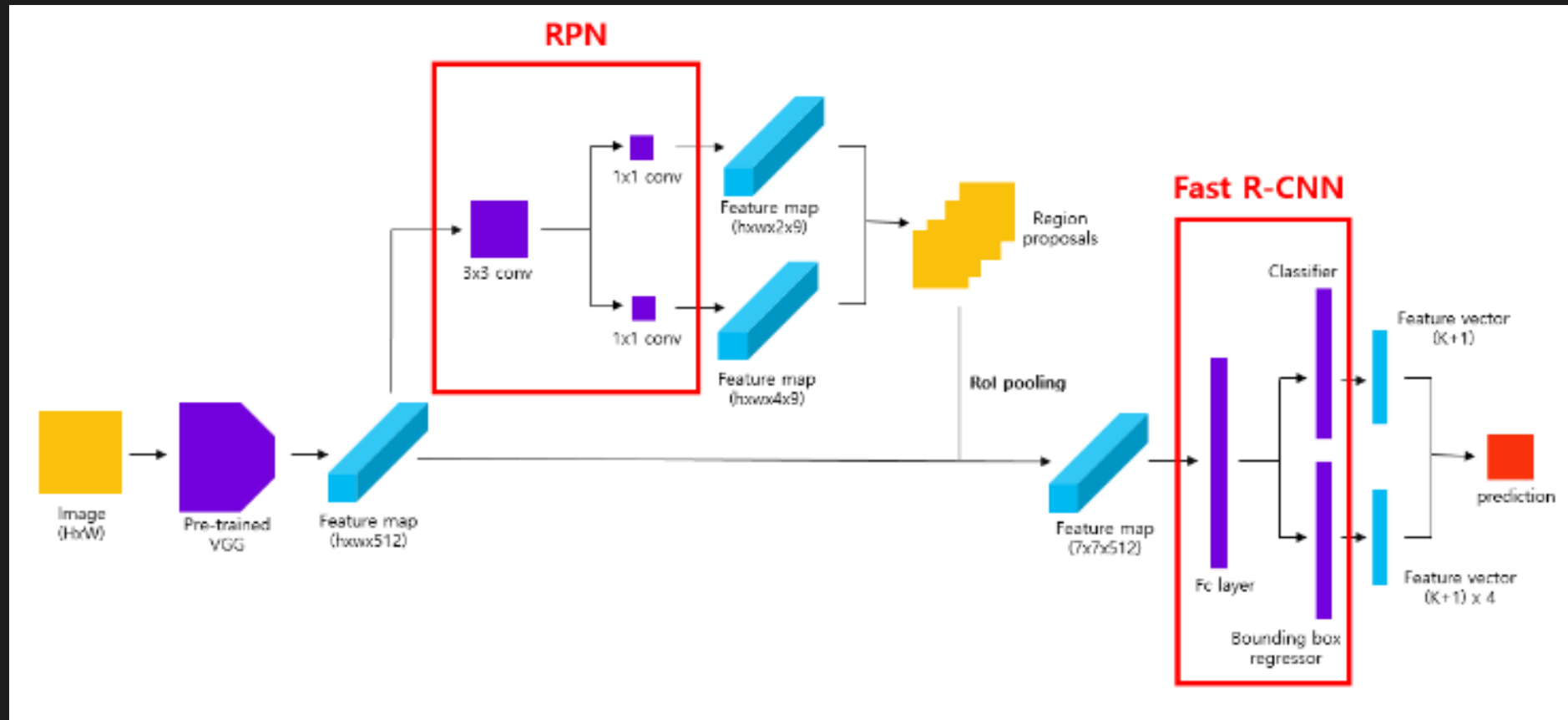
Limitation

- Still using computationally expensive region proposal

Faster R-CNN

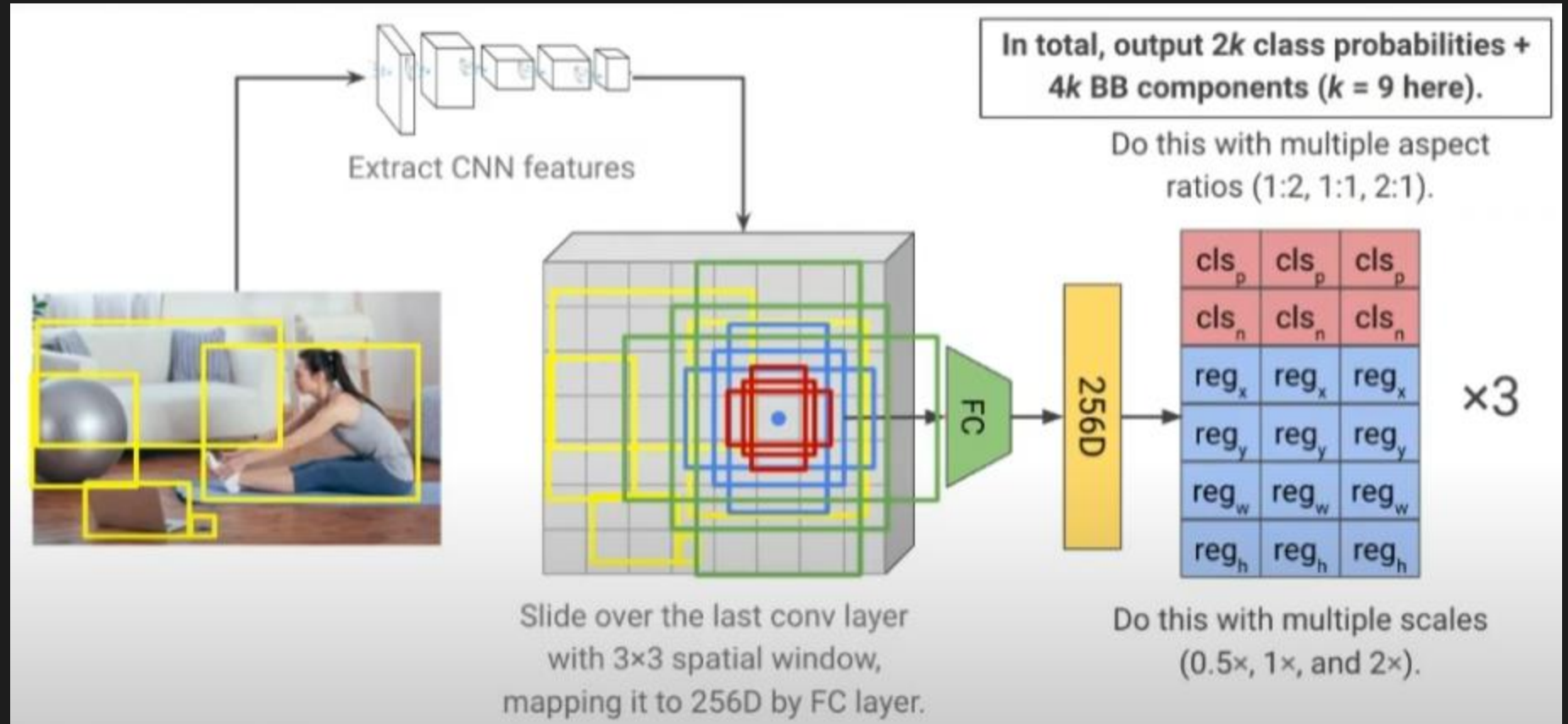
- Bring the **Region Proposal Network (RPN)** into the framework
- Stage 1 : Training the RPN
 - For each position in conv feature map, suppose a set of candidate bounding box (anchor)
 - Usually, 3 ratio and 3 sizes, total 9 anchors used.
 - RPN is trained to classify positive/negative for each anchor and to regress the bounding box if it is a positive.
- Stage 2 : Same as Fast R-CNN
 - Using the trained RPN, perform RoI pooling, classification and regression

Faster R-CNN : Architecture

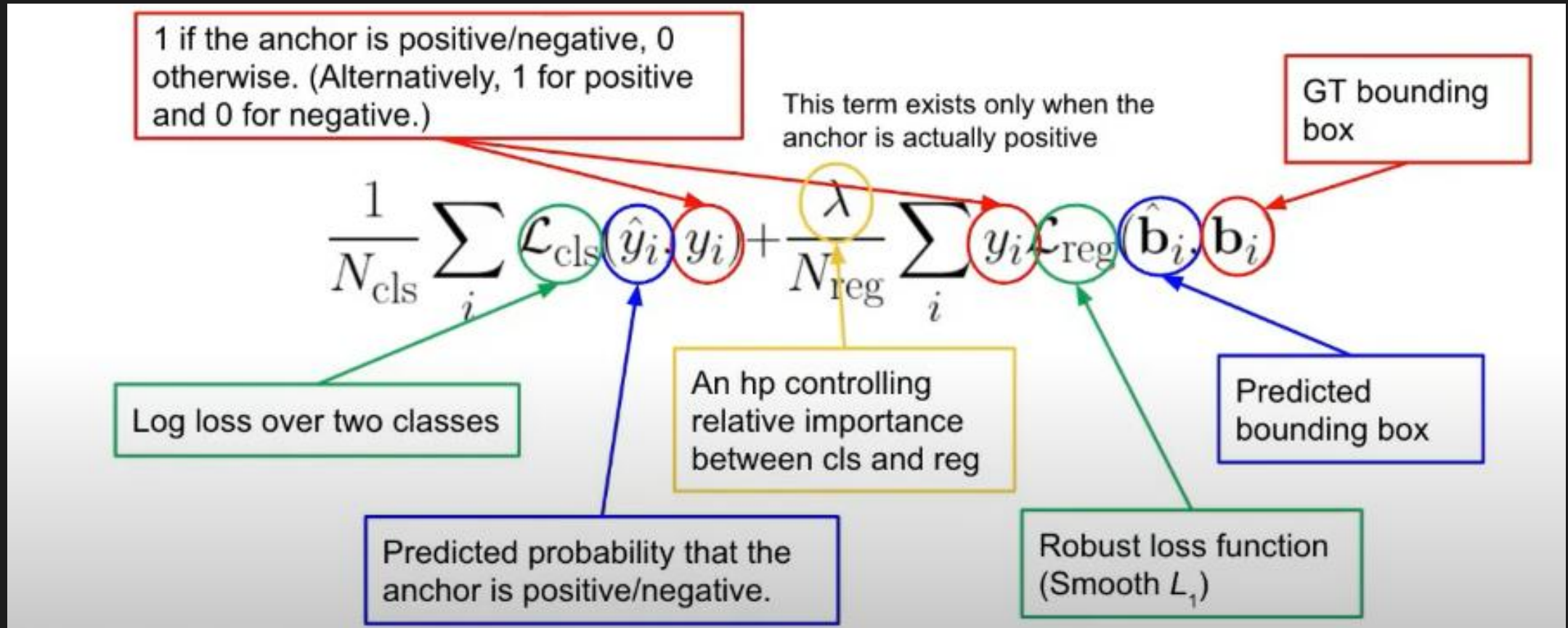


Faster R-CNN : RPN

- Predict the probability of the input region to be a **positive anchor** and to be a **negative anchor**
- Also possible to have output just positive score
- Regress 4 bounding box coordinates



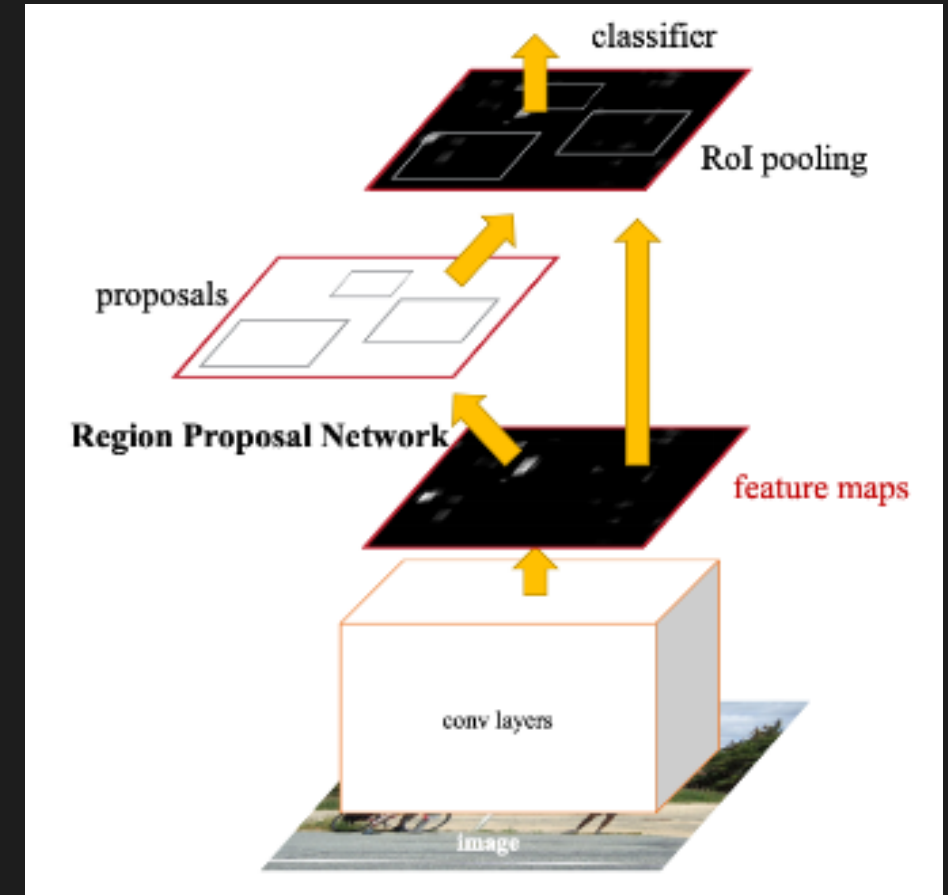
Faster R-CNN : Loss function for training RPN



Faster R-CNN : Training

Alternating Training

1. Train the RPN and pretrained CNN
2. Train a separate detection network by Fast R-CNN using the RPN, also pretrained CNN
3. Fine-tune the unique layers in RPN, freezing pretrained CNN
4. Fine-tune Fast R-CNN, freezing pretrained CNN



Faster R-CNN : Speed

Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

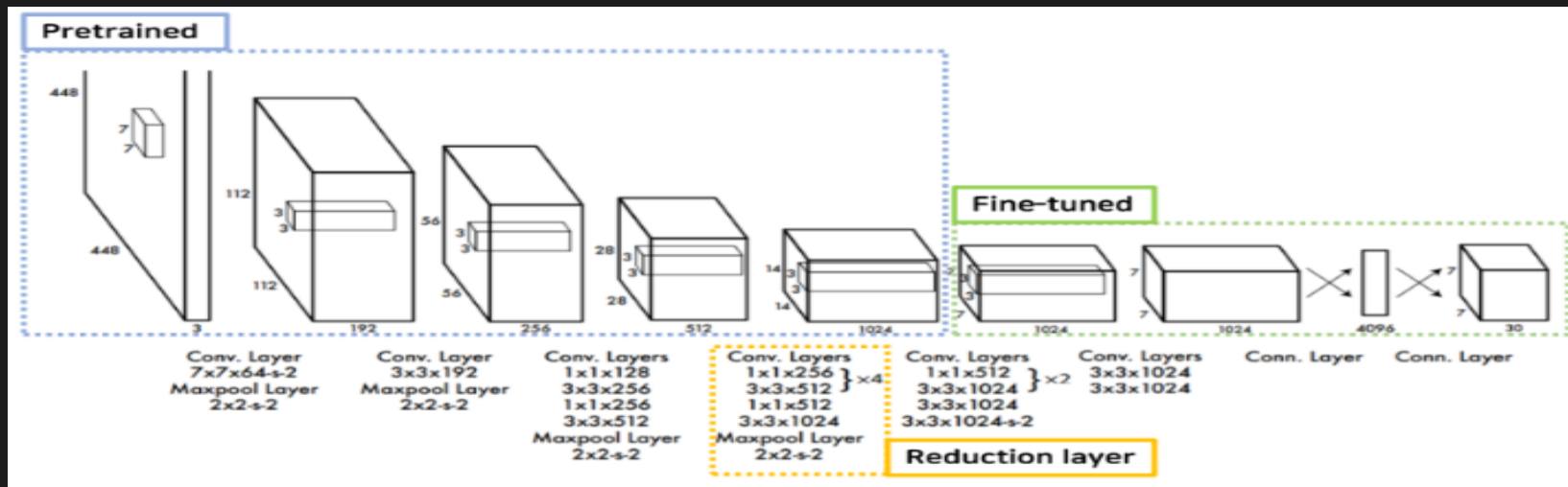
model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	10	47	198	5 fps
ZF	RPN + Fast R-CNN	31	3	25	59	17 fps

Fast R-CNN

Faster R-CNN

YOLO

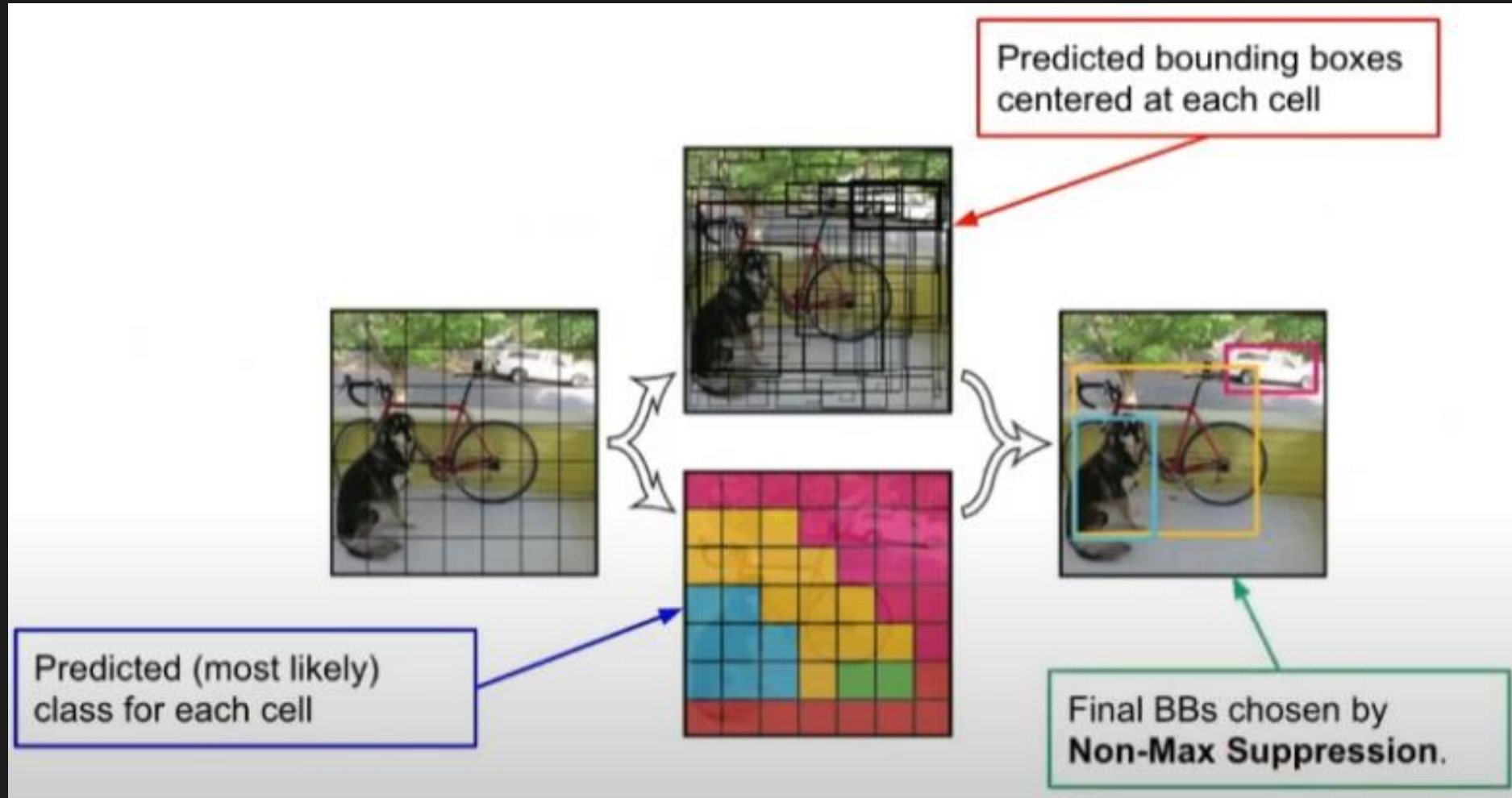
- A **single neural network** to predict bounding boxes and class probabilities from an image in a **single evaluation**
- Image detection as a **regression** problem
 - Model is similar to AlexNet, except for the output tensor



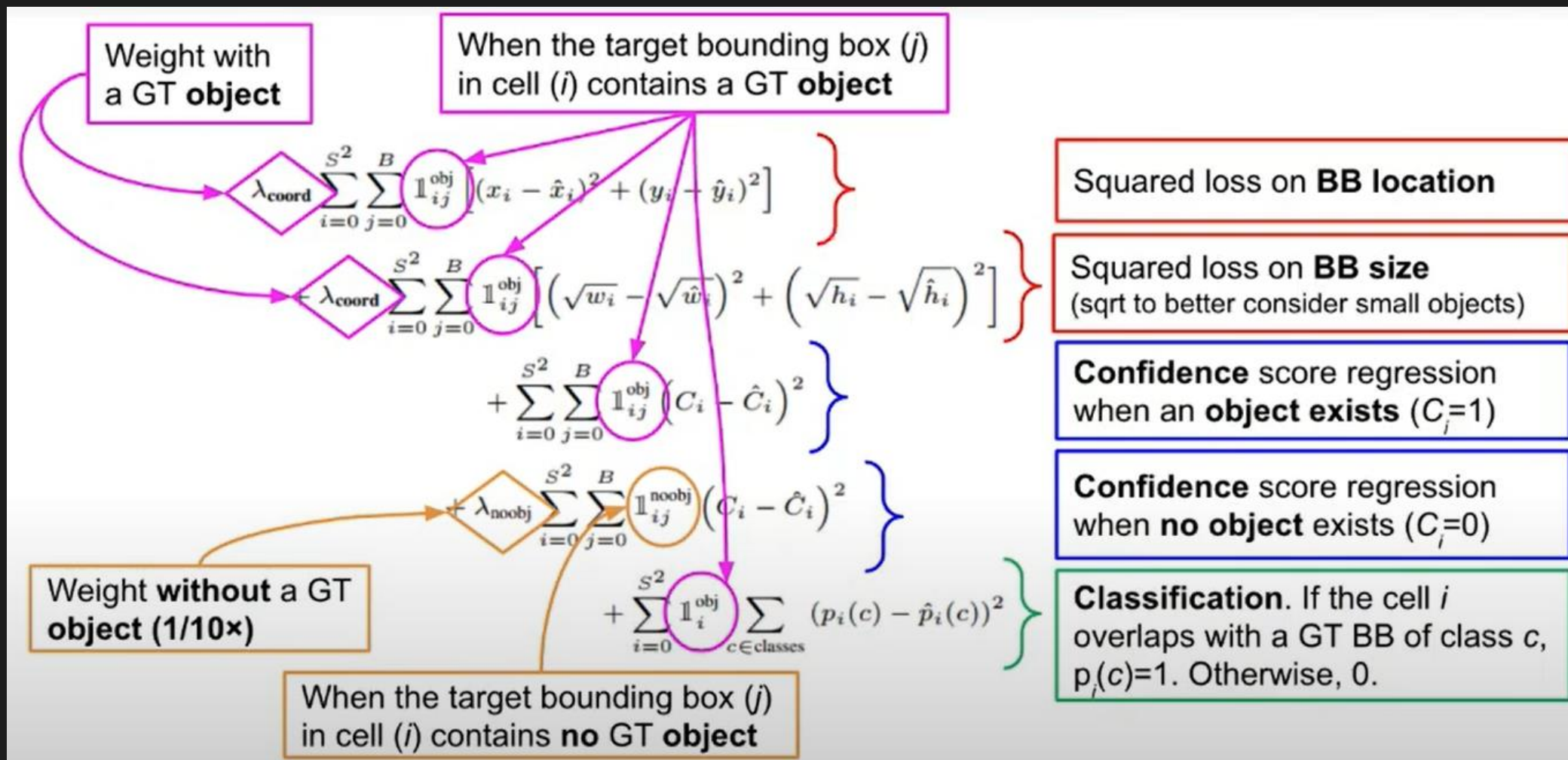
YOLO : Algorithm

- Divides the input image into a **S x S grid** (S=7)
 - If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.
- Each grid cell predicts **B bounding boxes** and confidence scores for those boxes (B=2)
 - B infers that up to B objects may coincide in the same cell.
 - Each bounding box consists of 5 predictions : x, y, w, h, and confidence.
 - Confidence = $P(\text{object}) \times \text{IoU}(\text{pred}, \text{GT})$
- Each grid cell also predicts **C conditional class probabilities**.
 - In total, output tensor is in shape $S \times S \times (5B+C)$.

YOLO : Algorithm

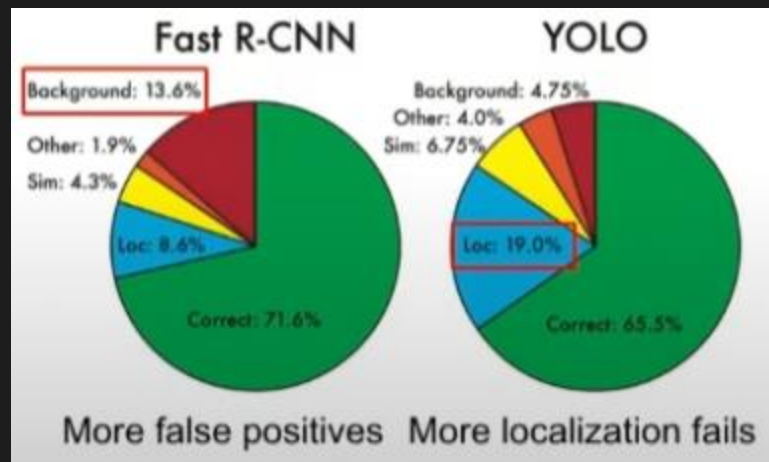


YOLO : Loss Function



YOLO : Result

- Real-time fast (45 frames/sec)
- Context-aware : Consider the entire image for object detection
- In overall accuracy, YOLO does not outperform Fast/Fast R-CNN
 - Tend to make different mistakes



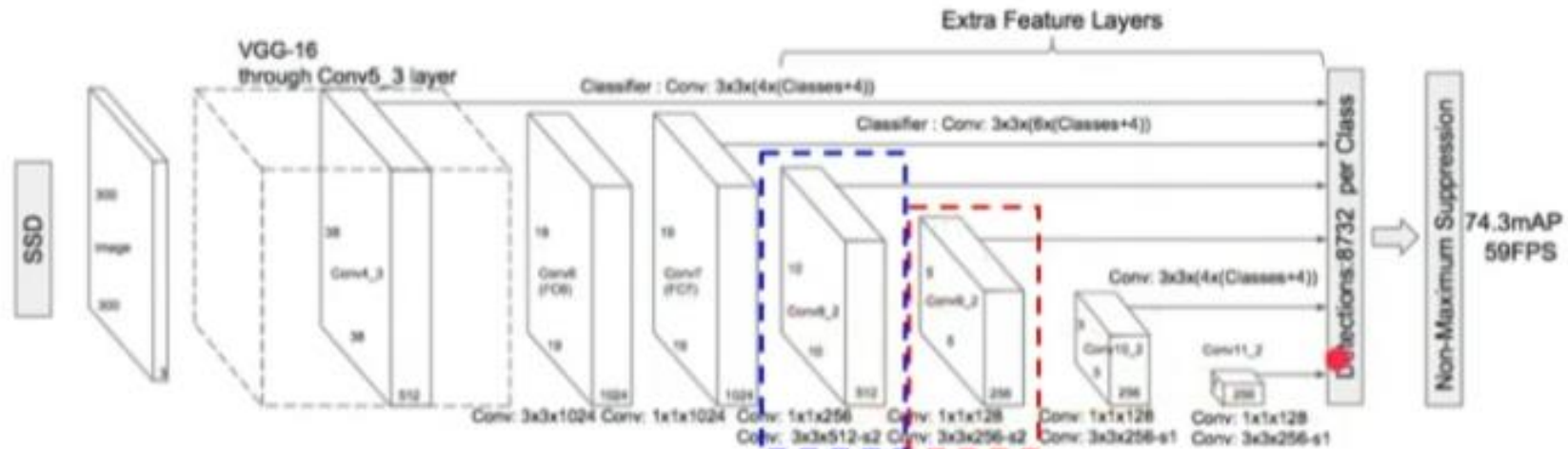
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16 [27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

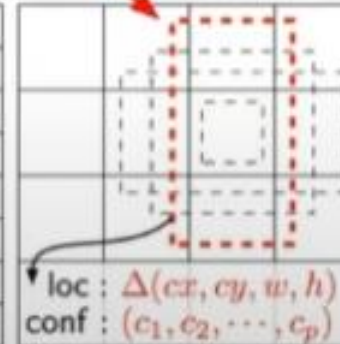
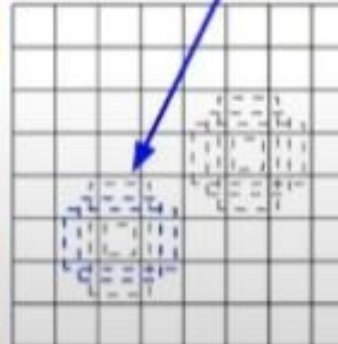
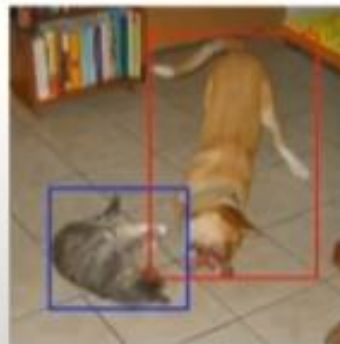
SSD : Single Shot MultiBox Detector

- On top of conv features (VGG-16), stack more conv layers with reduced feature map size.
 - Receptive field gets larger in the later layers.
 - Use multiple level of conv layers to detect objects of different size.
- Removed FC layers at the end.
 - Convolutional classifier : each 3 x 3 patch -> k boxes (C class scores + 4 for bbox)

SSD : Main Idea



Earlier conv layer:
Each cell looks **narrow**
range of the image.
→ Detect **small** objects.



Later conv layer:
Each cell looks **wider**
range of the image.
→ Detect **large** objects.

loc : $\Delta(cx, cy, w, h)$
conf : (c_1, c_2, \dots, c_p)

SSD : Loss Function

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

Loss on localization

$$L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x_i^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx}) / d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy}) / d_i^h$$

$$\hat{g}_j^w = \log \left(\frac{g_j^w}{d_i^w} \right) \quad \hat{g}_j^h = \log \left(\frac{g_j^h}{d_i^h} \right)$$

Similar to other
detection models

Loss on classification confidence

$$L_{conf}(x, c) = - \sum_{i \in Pos} x_{ij}^p \log(\tilde{c}_i^p) - \sum_{i \in Neg} \log(\tilde{c}_i^0) \quad \text{where} \quad \tilde{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

Maximize predicted score
for the correct class

For negative boxes, maximize
the background score

SSD : Result

Method	data	mAP	Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
					0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast [6]	07	66.9	Fast [6]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast [6]	07+12	70.0	Fast [23]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [2]	07	69.9	Faster [2]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
Faster [2]	07+12	73.2	ION [23]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster [2]	07+12+COCO	78.8	Faster [24]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	07	68.0	SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD300	07+12	74.3	SSD512	trainval35k	26.8	46.5	27.8	9.0	28.9	41.9	24.8	37.5	39.8	14.0	43.5	59.0
SSD300	07+12+COCO	79.6														
SSD512	07	71.6														
SSD512	07+12	76.8														
SSD512	07+12+COCO	81.6														

Accuracy on COCO 2015: SSD wins!

**Accuracy on PASCAL VOC
2007: SSD wins!**

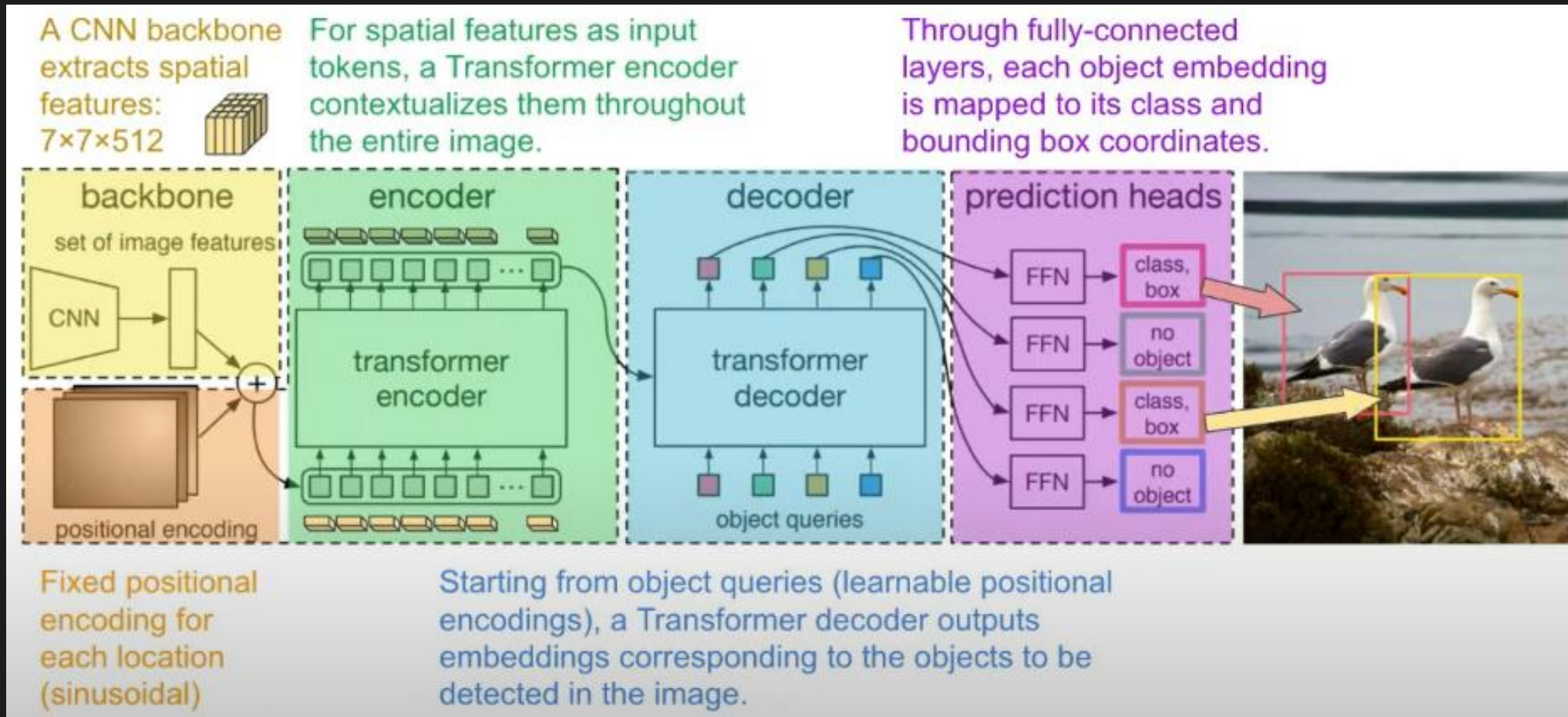
Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Speed: Faster R-CNN < SSD < YOLO

DETR : Detection Transformer

- Main idea
 - Adopting the **encoder-decoder Transformer** architecture, learns the relations of the objects and the global context.
 - A set-based global loss that forces unique predictions via **bipartite matching**.
 - Removing the need for hand-designed components like NMS.

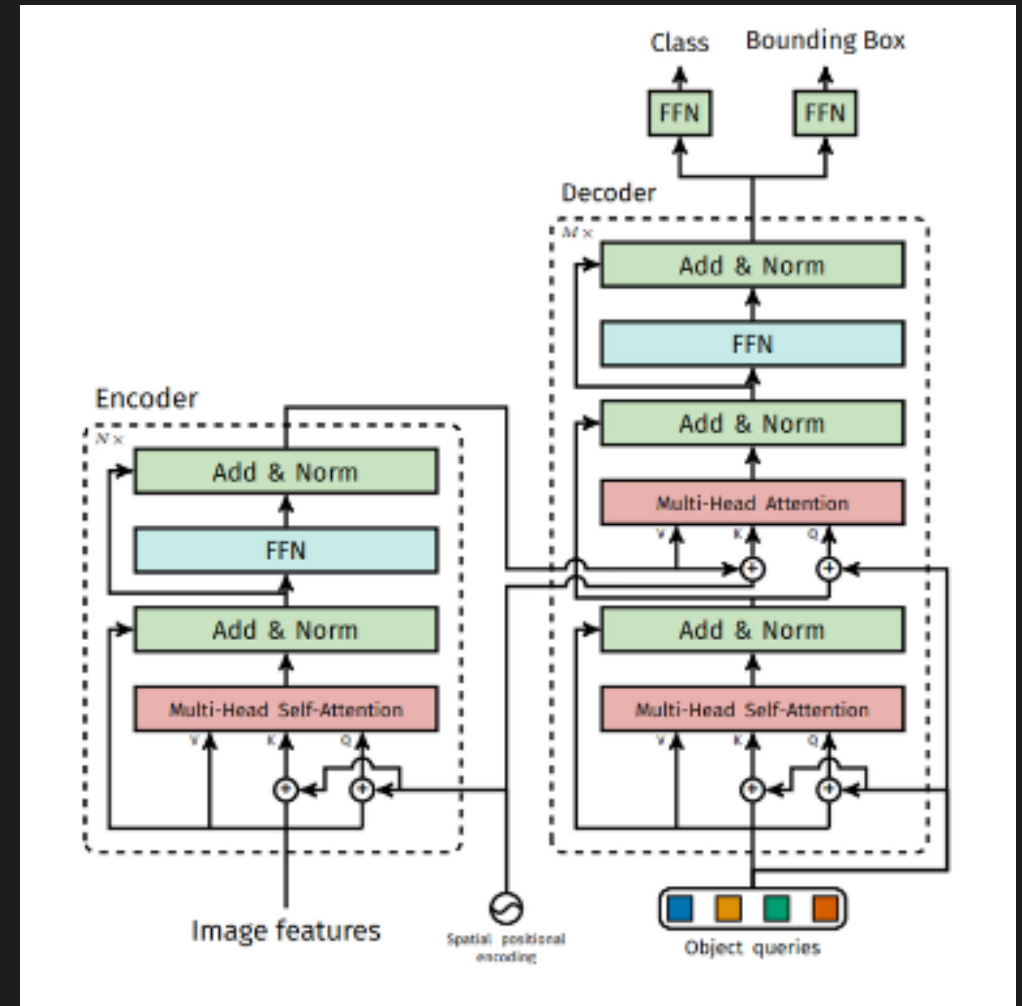
DETR : Architecture



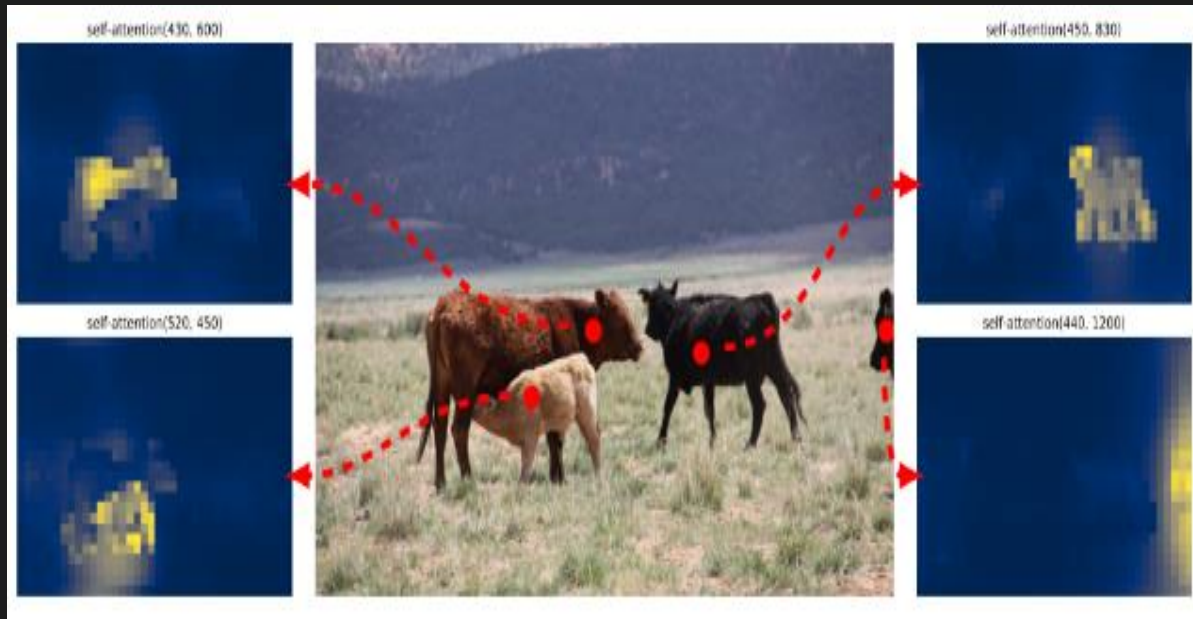
DETR : Architecture

Difference from the original Transformer :

- Positional encoding is applied only to the queries and keys.
- Positional encoding is added at every layer. (No proof but experimental evidence)
- Outputs are produced in parallel, multi head after decoder.



DETR : Results



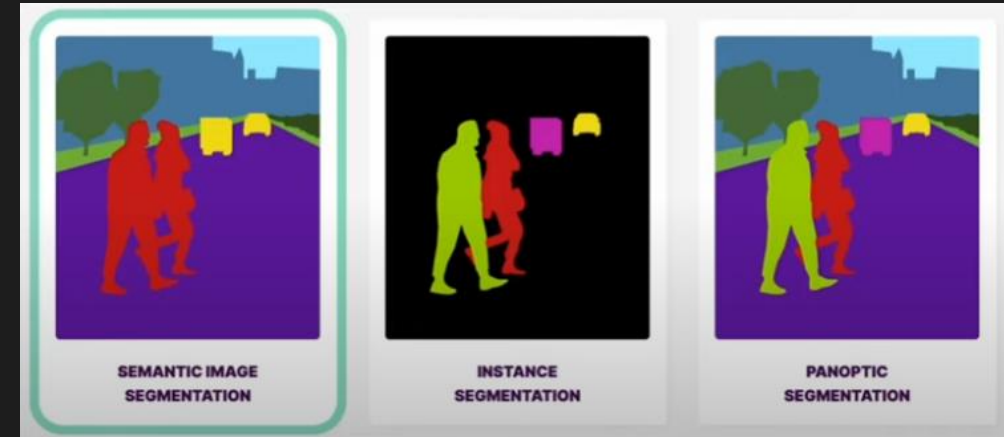
Segmentation

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

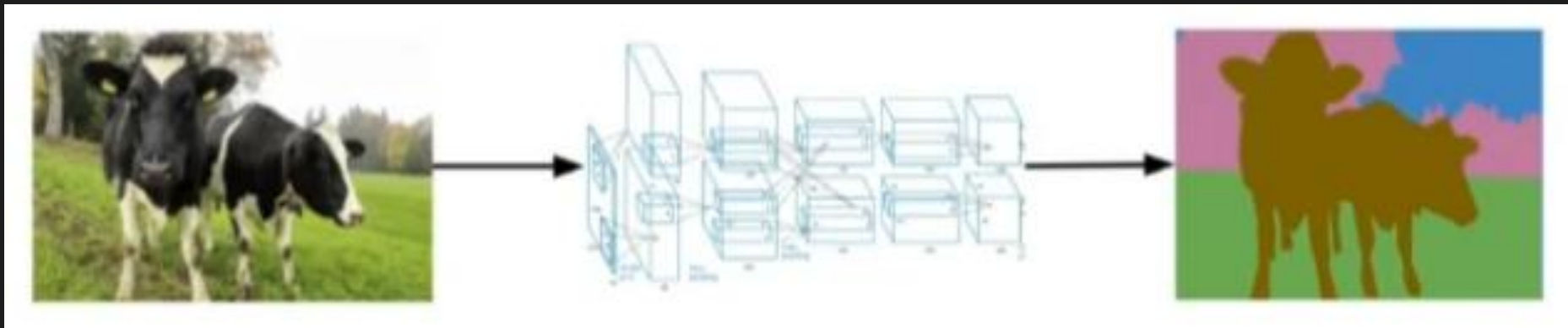
Segmentation

- Semantic Segmentation
 - Each pixel is labeled with a semantic category.
 - Don't differentiate instances, only care about pixels.
- Instance Segmentation
 - Combination of object detection and semantic segmentation.
 - Detect all objects of all classes, and segmentize them pixel-wise (instead of bounding boxes)
- Panoptic Segmentation
 - Combination of semantic and instance segmentation



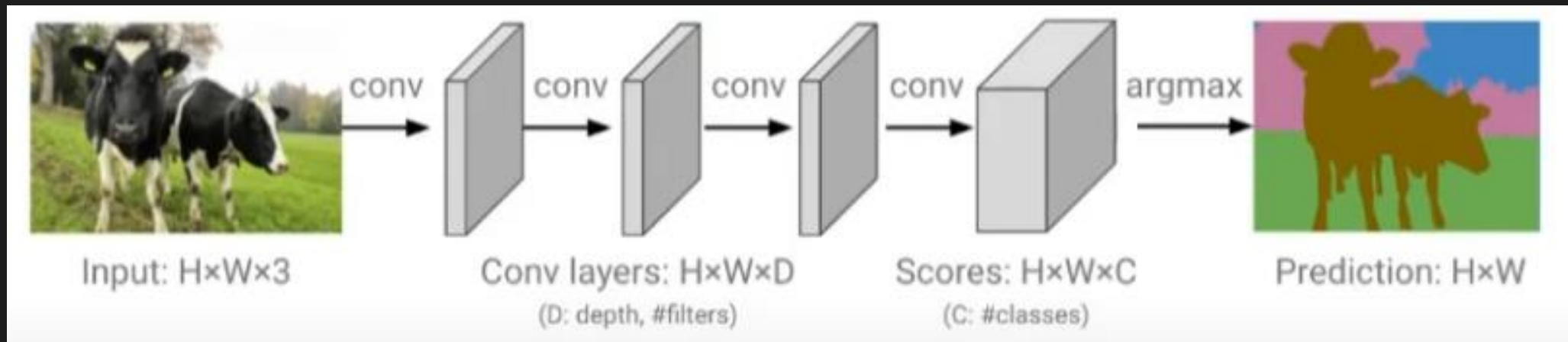
First idea for Semantic Segmentation

- We need to consider context nearby the target pixel.
 - For each pixel, we may take a small image patch centered on the target pixel, then classify it. -> It would be very inefficient!
- Encode the entire image with a conv net, then do semantic segmentation on top.
 - We do require the output feature map size to be same as input size.



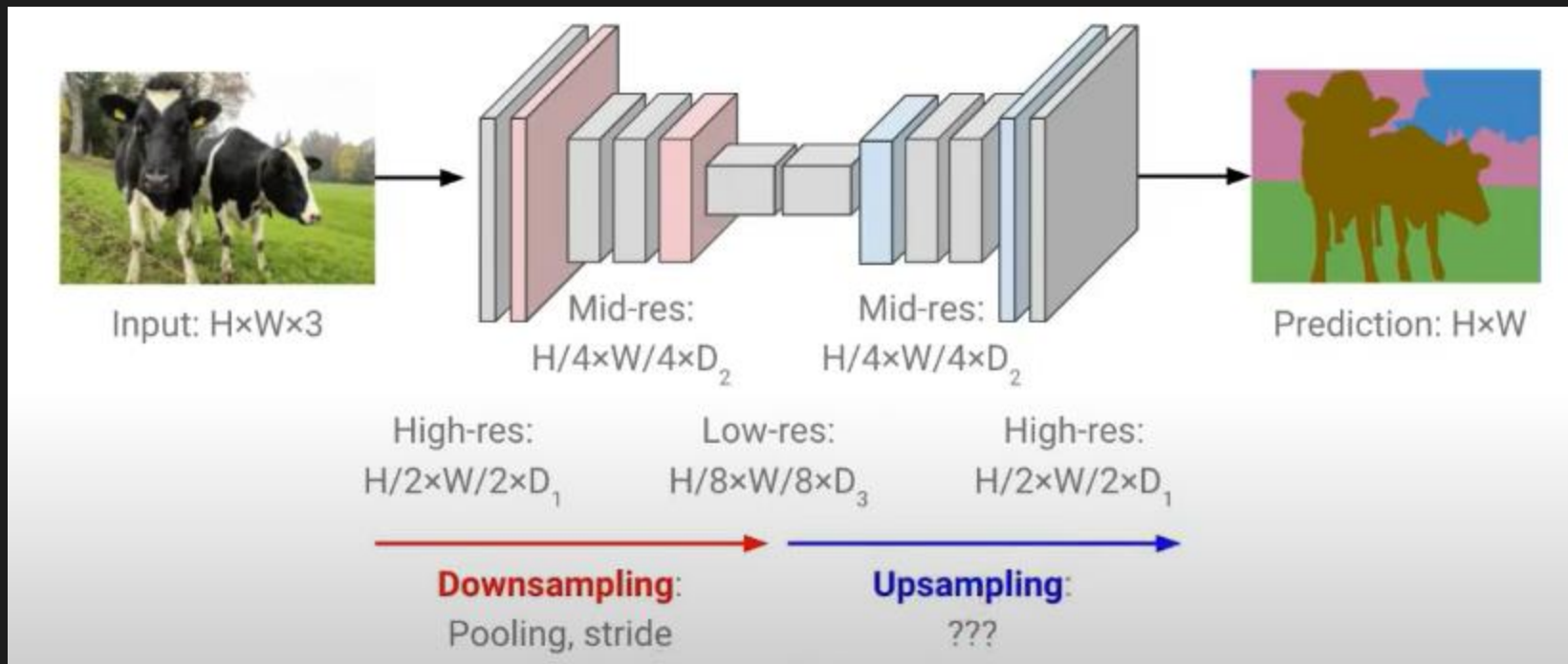
Fully Convolutional Solution

- Design a network with only conv layers without downsampling operators to make prediction for pixels all at once
 - Backpropagation to the earlier stages are very expensive.



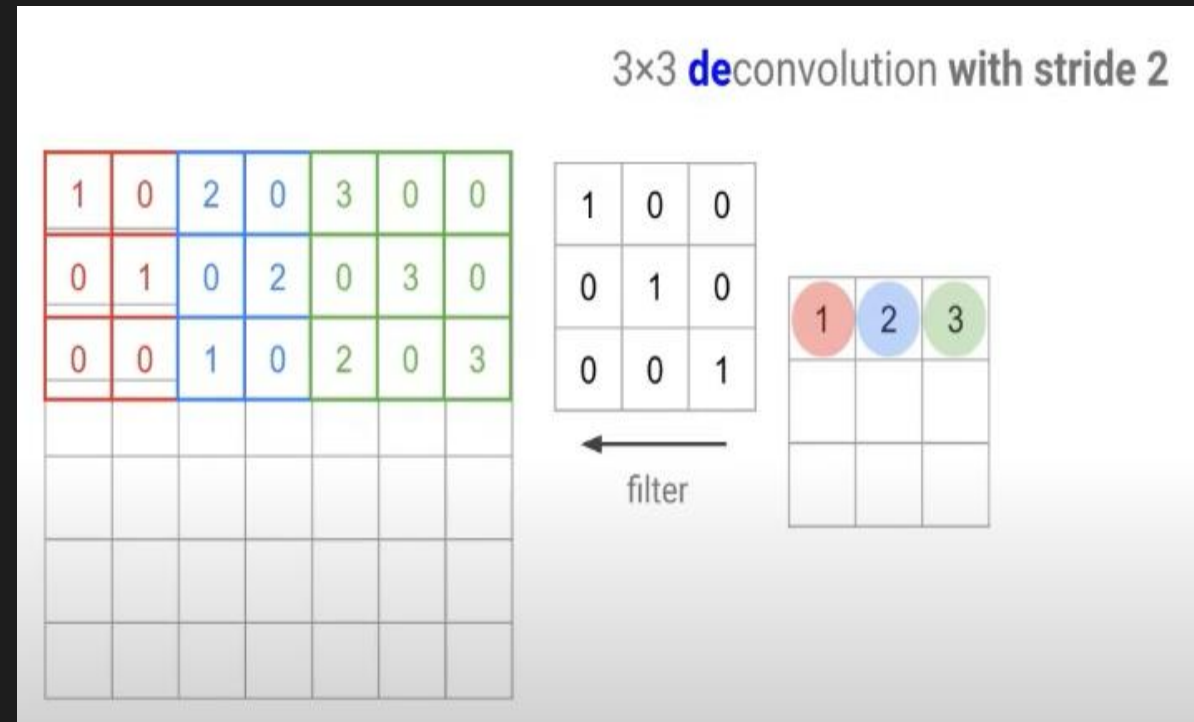
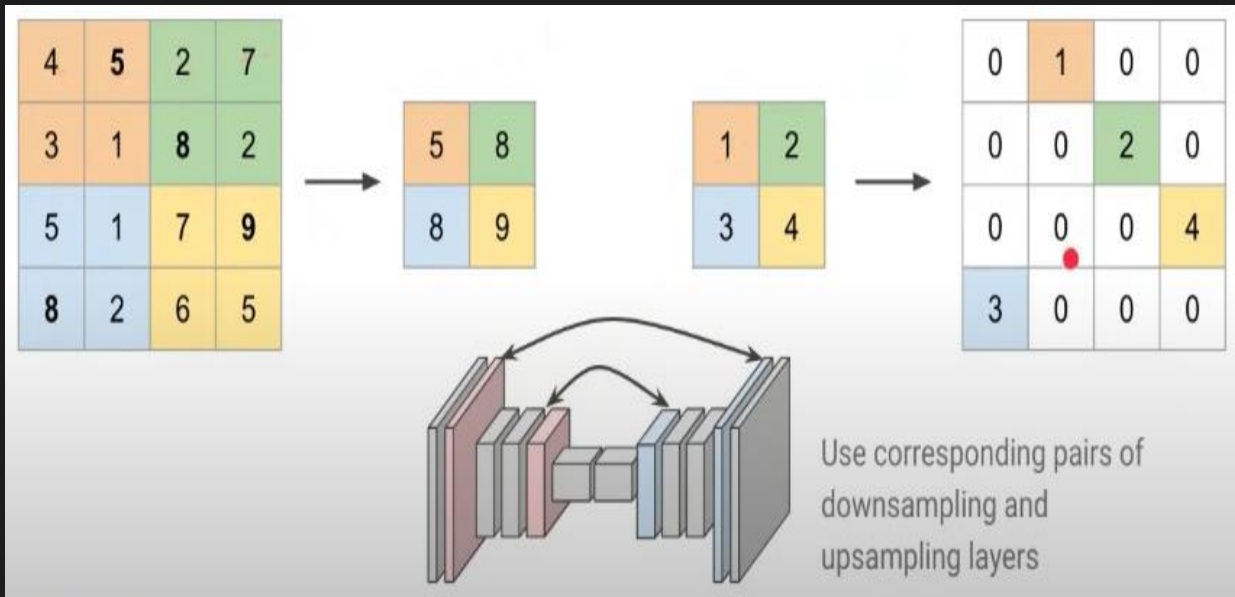
Deconvolution Networks

- Design a network with only conv layers with downsampling and upsampling inside the network



Deconvolution Networks : Upsampling

- Take advantage of in-network information
- Max Pooling : remember which element was max.
- Max Unpooling : use positions from previous pooling layer



U-Net

- Originally proposed for biomedical segmentation task: cells and membranes

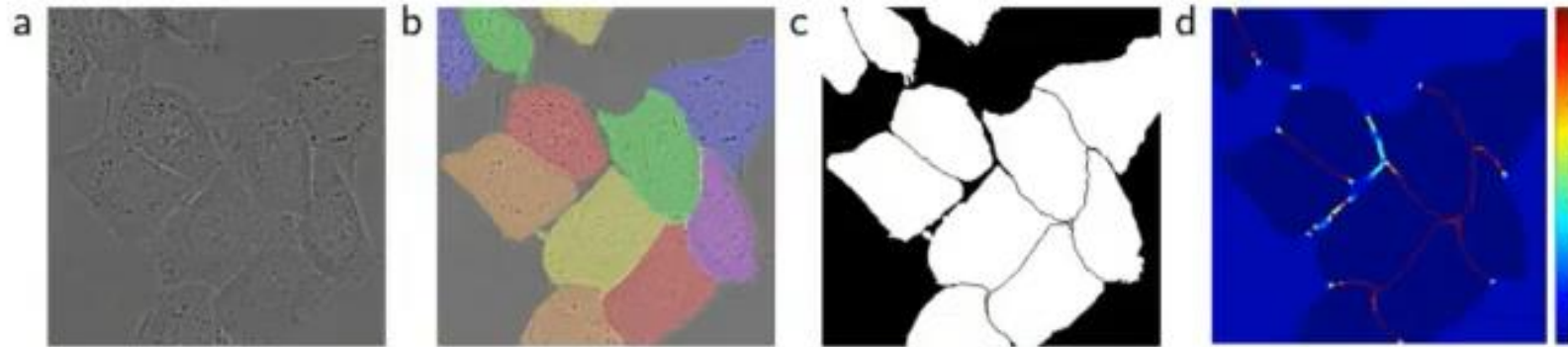
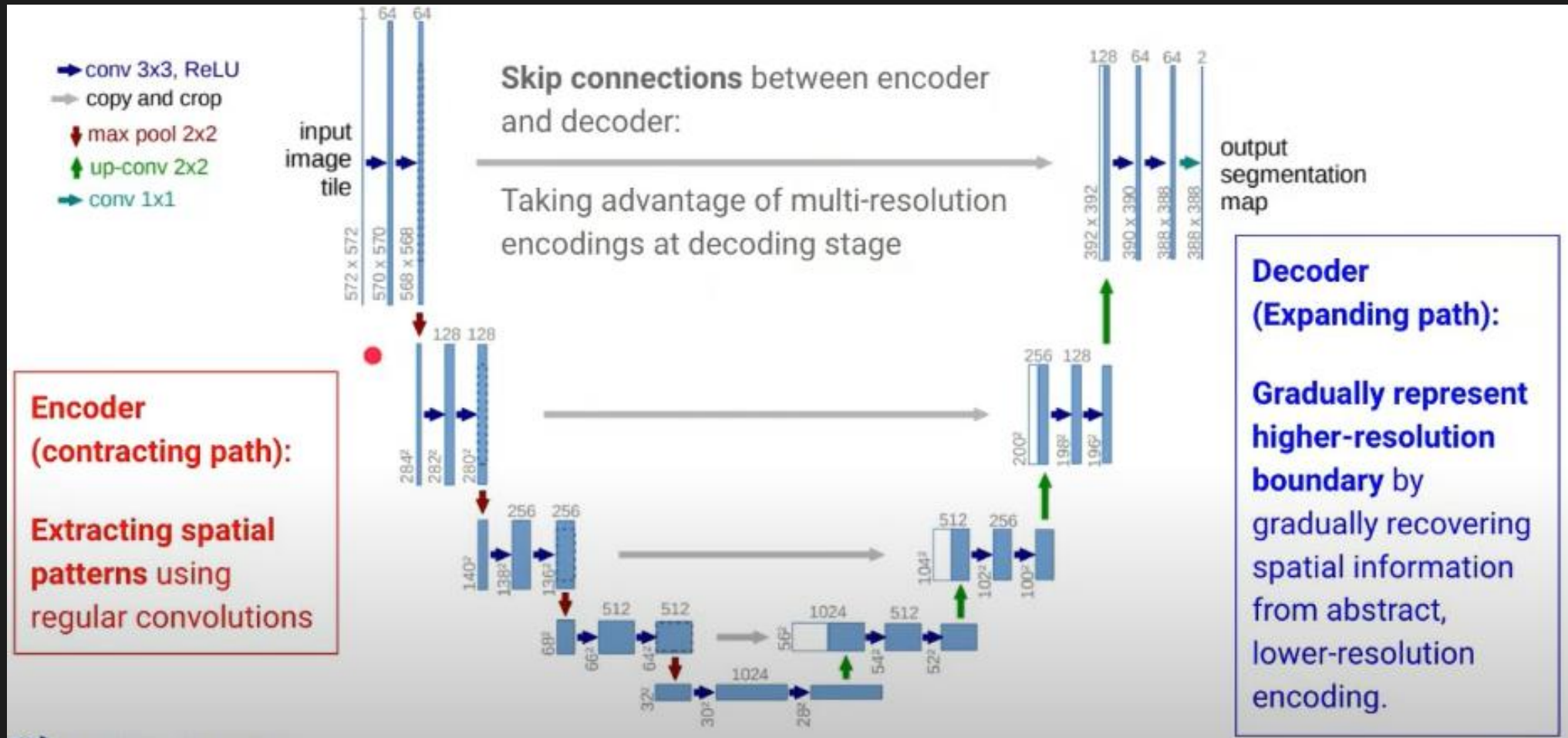


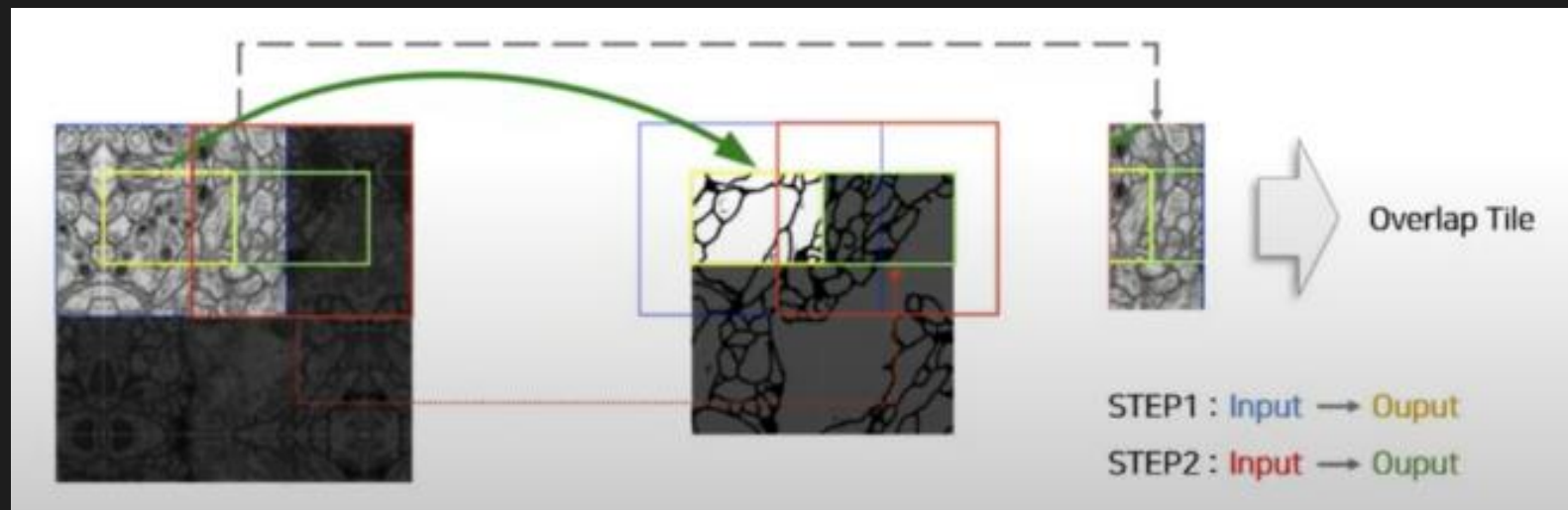
Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (a) raw image. (b) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (c) generated segmentation mask (white: foreground, black: background). (d) map with a pixel-wise loss weight to force the network to learn the border pixels.

U-Net : Architecture



U-Net : Detailed Ideas

- Different input and output size
 - Due to the contracting nature of this network, we **feed larger image** as input.
 - **Mirror extrapolation** : extrapolate the missing context by **mirroring the input image** instead of zero-padding.
 - **Overlap-tile strategy** : generate seamless segmentation that only pixels for which the full context is available from the input image.



U-Net : Loss Function

- **Pixel-wise softmax** over the final feature map + cross entropy loss
- Weighted cross-entropy
 - Ordinary cross-entropy function combined with weight map
 - Weight map (pre-computed)
 - Focus more on exact classification of pixels adjacent to cell border

$$E = \sum_{x \in \Omega} w(x) \log(p_{l(x)}(x))$$

Distance to the first and second nearest borders

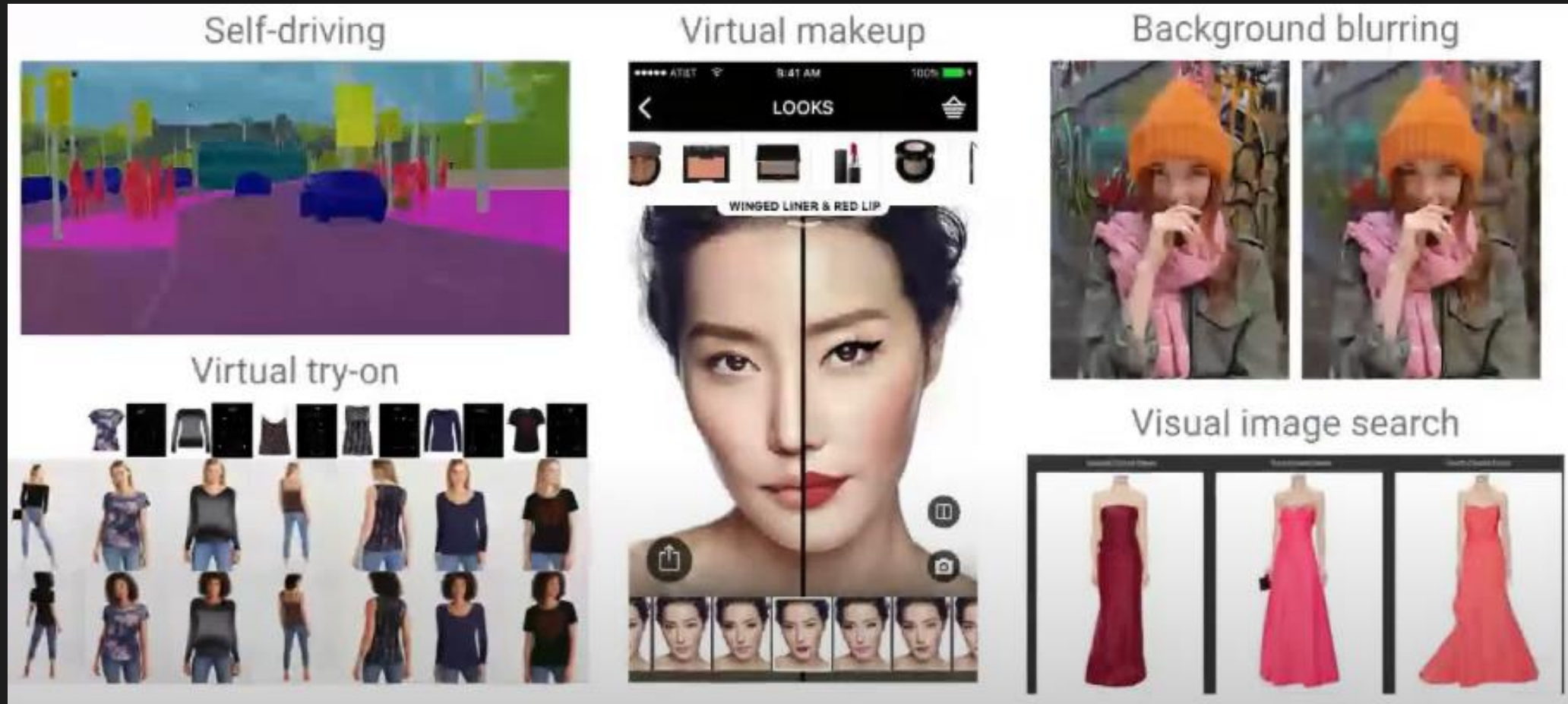
$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$

$p_k(x)$: probability that pixel x belongs to class k

$a_k(x)$: logit that pixel x belongs to class k (model output)

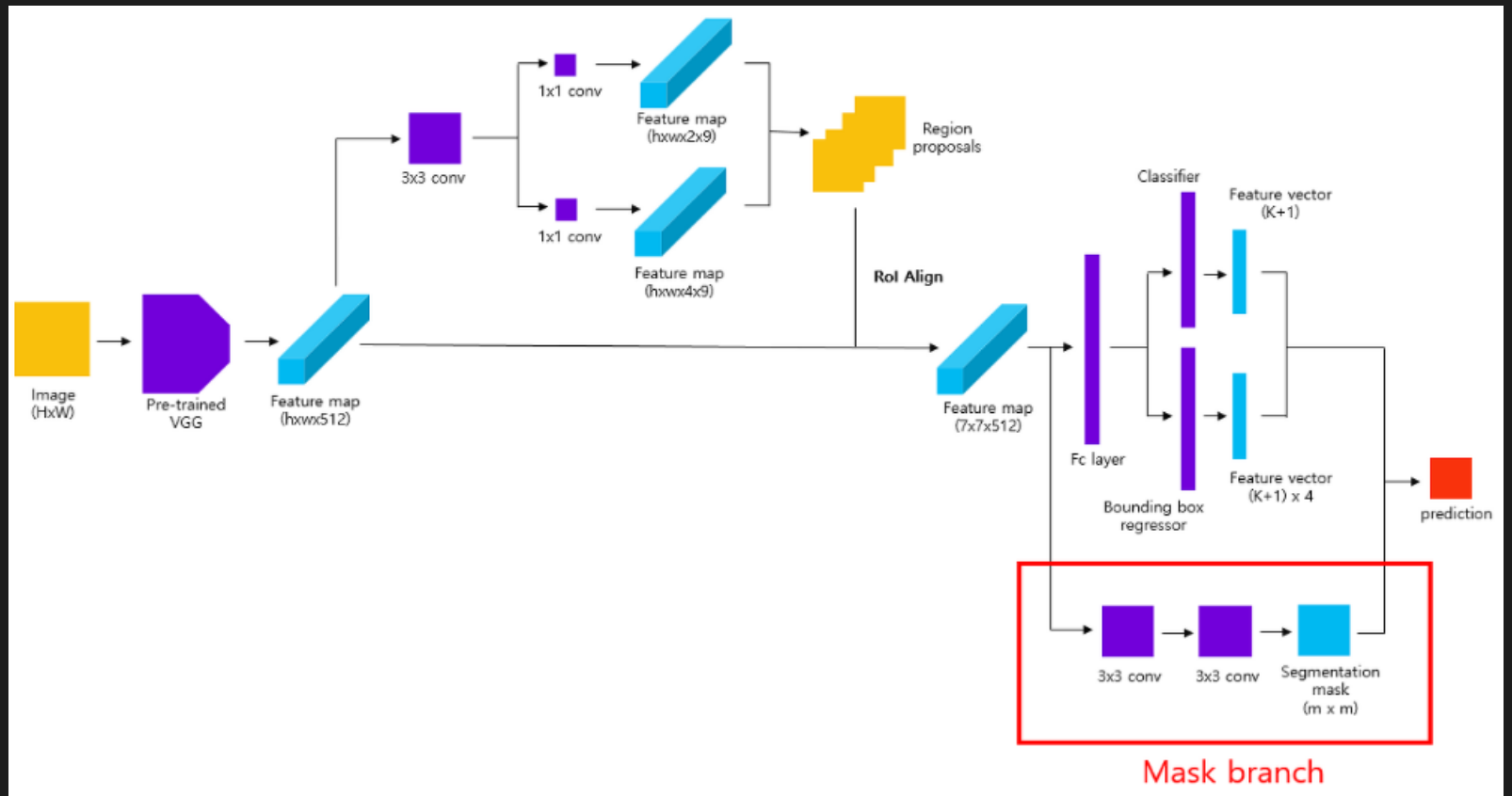
$$p_k(x) = \exp(a_k(x)) / (\sum_{k'=1}^K \exp(a_{k'}(x)))$$

Semantic Segmentation : Applications



Mask R-CNN

- Extends Faster R-CNN by adding the mask prediction tower.
- Mask network operates on each RoI and predicts a $m \times m$ binary mask, in parallel with other existing losses.



Mask R-CNN : RoI Align

- Pixel-to-pixel nature of segmentation problem requires the RoI feature maps to be well aligned to **smoothly preserve** the per-pixel spatial correspondence.
- Instead of RoI pooling, use **bilinear interpolation** to better estimate feature values at each cell.

0.8	1.1	2.1	3.9	0.4
2.2	4.3	4.2	1.8	6.3
0.2	5.5	4.8	7.7	9.5
1.1	0.8	2.2	0.8	5.2
5.0	1.8	1.9	2.0	4.1

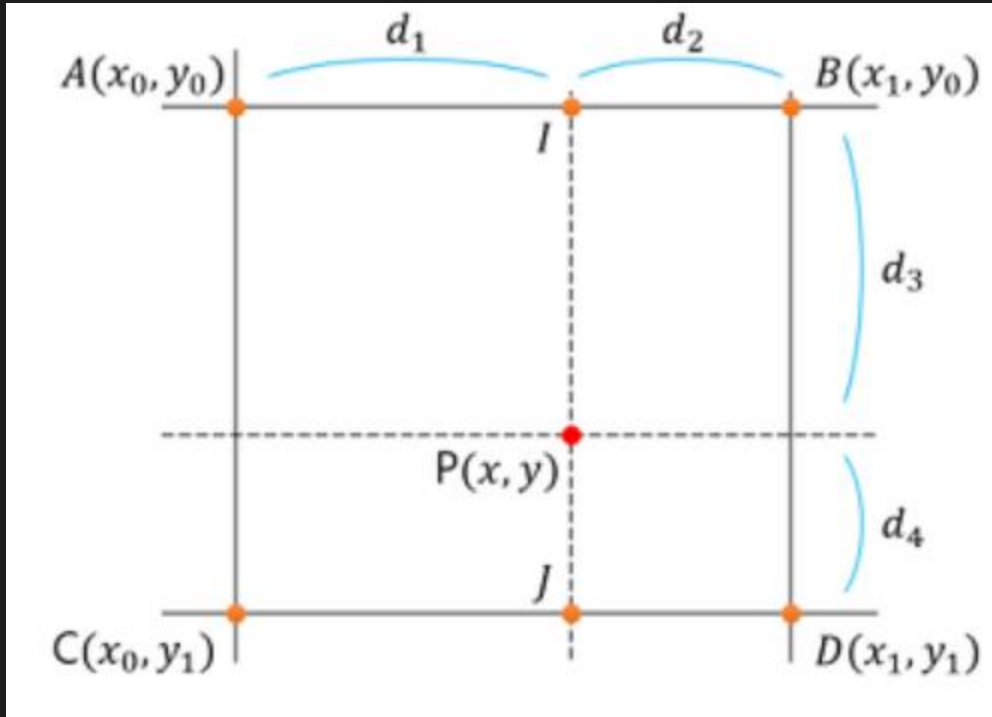
5 x 5 Feature Map
Red Dotted Line = RoI



0.8	1.1	2.1	3.9	0.4
2.2	4.3	4.2	1.8	6.3
0.2	5.5	4.8	7.7	9.5
1.1	0.8	2.2	0.8	5.2
5.0	1.8	1.9	2.0	4.1

Misalignment 발생
Segmentation 성능 저하

Mask R-CNN : BiLinear Interpolation



$$f(I) = \frac{d_1}{d_1 + d_2} f(B) + \frac{d_2}{d_1 + d_2} f(A)$$

$$f(J) = \frac{d_1}{d_1 + d_2} f(D) + \frac{d_2}{d_1 + d_2} f(C)$$

$$f(P) = \frac{d_3}{d_3 + d_4} f(J) + \frac{d_4}{d_3 + d_4} f(I)$$

$$= \frac{1}{(d_1 + d_2)(d_3 + d_4)} \{d_1 d_3 f(D) + d_2 d_3 f(C) + d_1 d_4 f(B) + d_2 d_4 f(A)\}$$

$$= \frac{1}{(d_1 + d_2)(d_3 + d_4)} \{ (y_1 - y)(x_1 - x)f(A) + (y_1 - y)(x - x_0)f(B) + (y - y_0)(x_1 - x)f(C) + (y - y_0)(x - x_0)f(D) \}$$

Mask R-CNN : RoI Align

1. Create $m \times m$ grid
2. Create $(n-1) \times (n-1)$ cross points, splitting specific grid as $n \times n$ area
3. For each cross point, estimate approximated value by bilinear interpolation
4. Max pool for $(n-1) \times (n-1)$ cross points
5. Repeat 2-4 for other grid

0.8	1.1	2.1	0.8	1.1	2.1	3.9	0.4
2.2	4.3	4.2	2.2	4.3	4.2	1.8	6.3
0.2	5.5	4.8	0.2	5.5	4.8	7.7	9.5
0.8	1.1	2.1	0.8	1.1	2.1	3.9	0.4
2.2	4.3	4.2	2.2	4.3	4.2	1.8	6.3
0.2	5.5	4.8	0.2	5.5	4.8	7.7	9.5
1.1	0.8	2.2	1.1	0.8	2.2	0.8	5.2
5.0	1.8	1.9	5.0	1.8	1.9	2.0	4.1

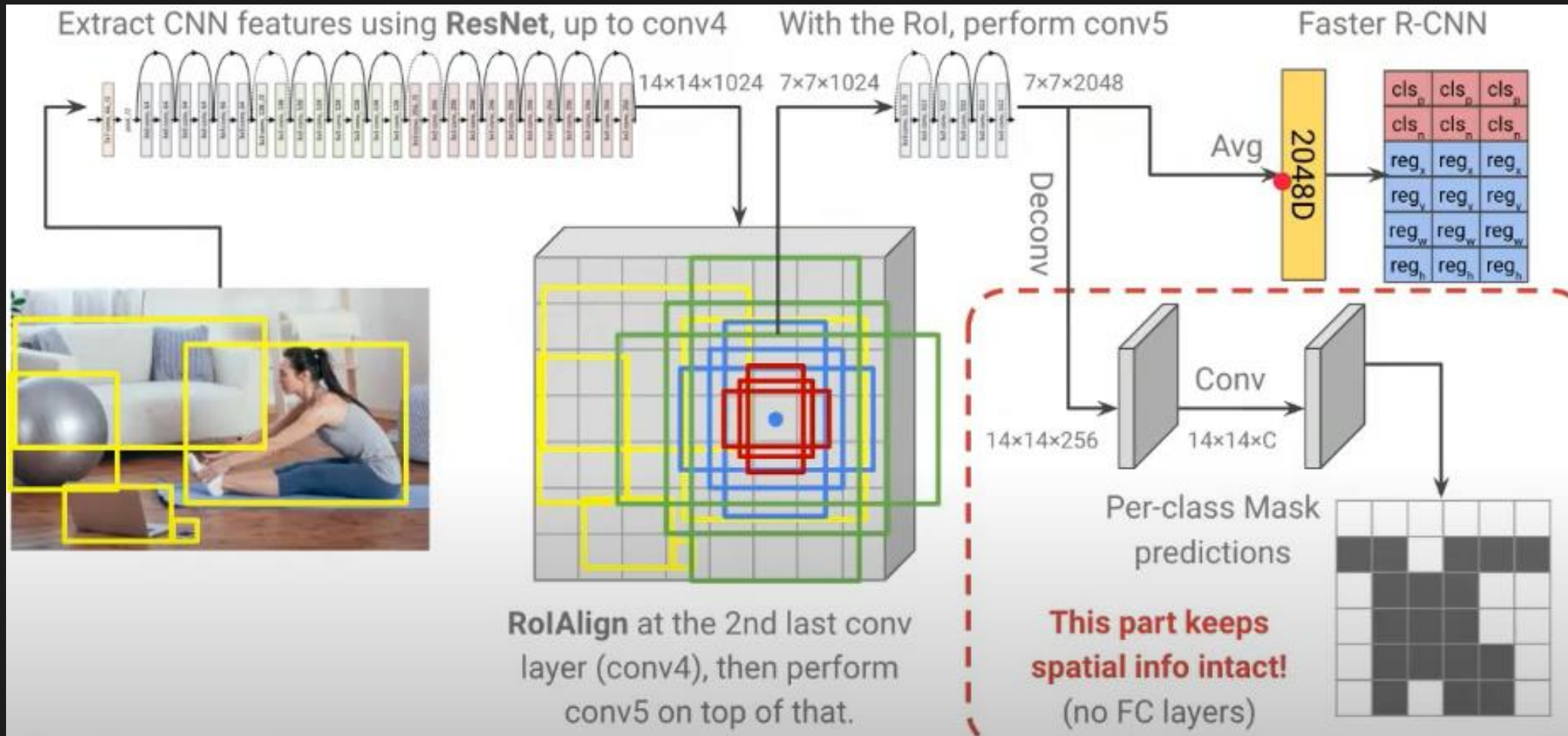
8 x 8 Feature Map
Red Dotted Line = RoI



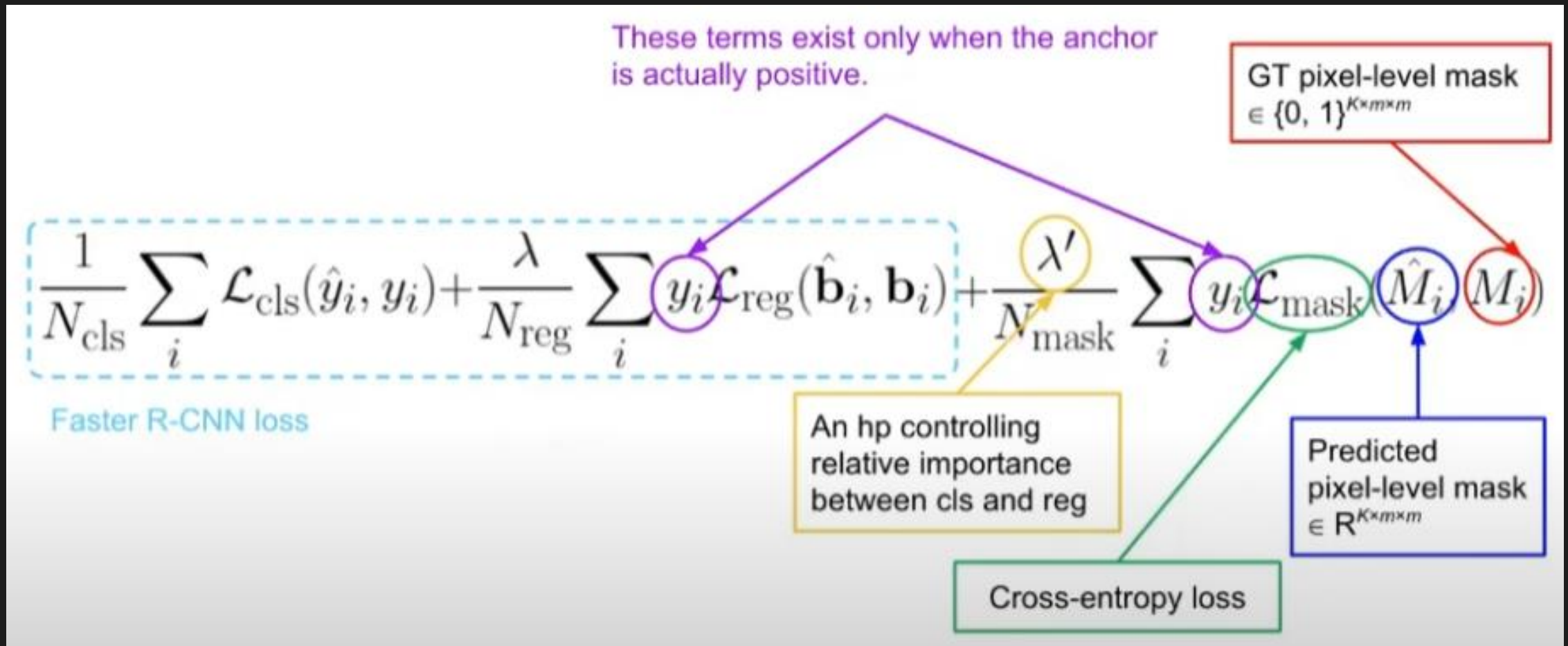
3.9	4.2
3.6	3.7

2 x 2 RoI
Fixed Size

Mask R-CNN : Mask Prediction Network



Mask R-CNN : Loss Function



Discussion

- 개인 과제

- Segmentation task에도 Transformer architecture을 활용한 많은 모델들이 있습니다. Semantic segmentation 또는 instance segmentation task에 대한 transformer based model 한 개를 찾아보고, 어떤 방식으로 segmentation을 수행하는지, transformer architecture을 어떻게 활용하는지 간략히 소개해주세요.

- 팀 과제

- IoU의 장단점은 무엇일까요? IoU의 단점을 해결할 수 있는 새로운 metric을 생각해봅시다. (IoU의 단점은 ReLU activation function와 유사한 맥락으로 생각해볼 수 있습니다.)

참고자료

자료 출처

- cs231n
- Machine Learning for Visual Understanding (Fall 2021) – Joonseok Lee

추가 자료

본 PPT는 고려대학교 딥러닝학회 AIKU의 정기세미나 및 기타 활동 내용을 바탕으로 하고 있습니다.

무단 도용 및 활용을 금합니다.

관련한 문의는 @aiku_.official로 DM부탁드립니다.

감사합니다.