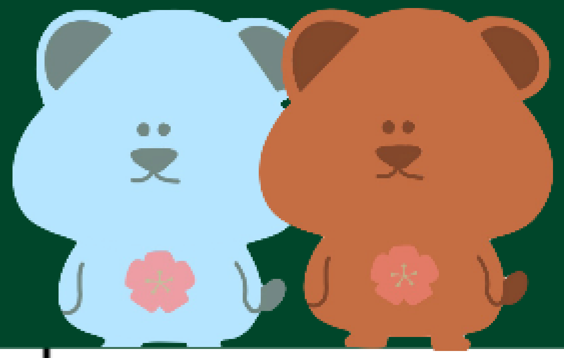
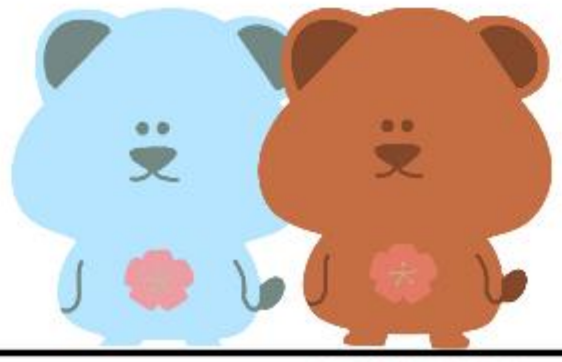
논문 스터디 2주차

Week18 최지우, 하수민, 민소연

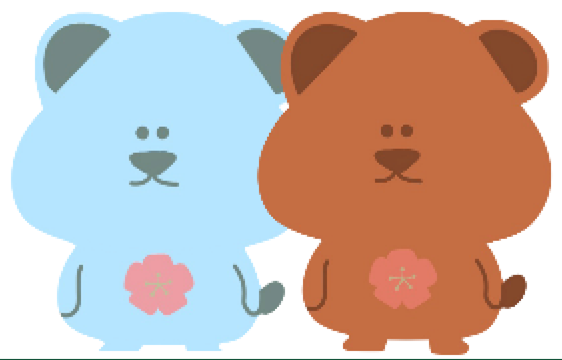


EWHA EURON

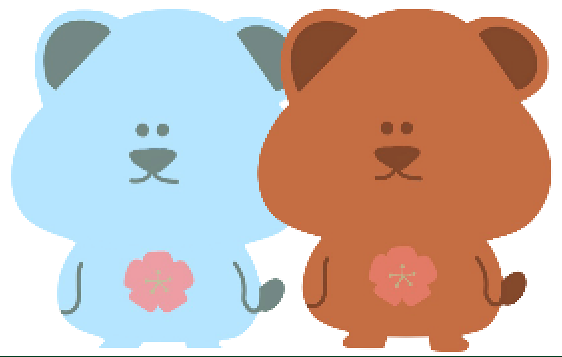
Index



Learning Deep Features for Discriminative Localization

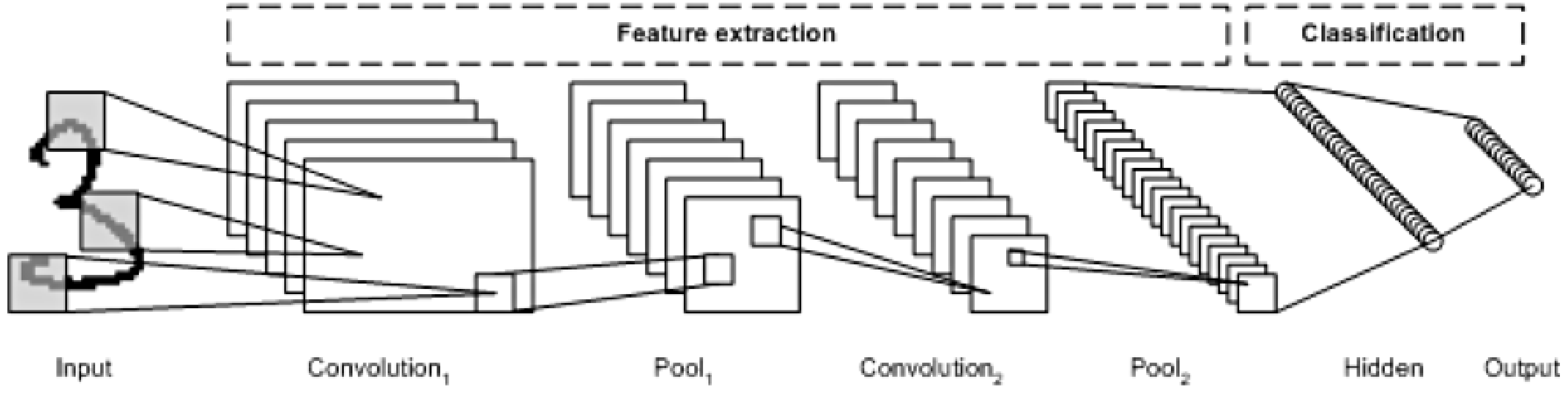


abstract



EWHA EURON

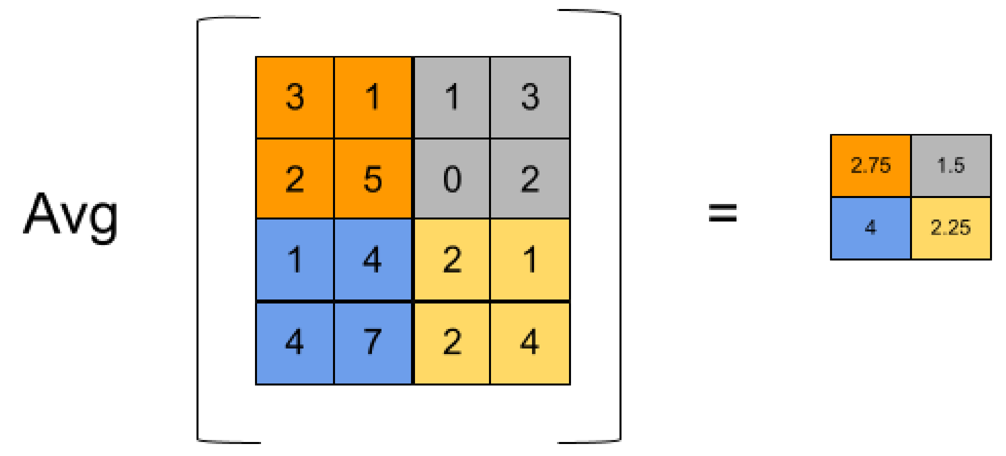
abstract

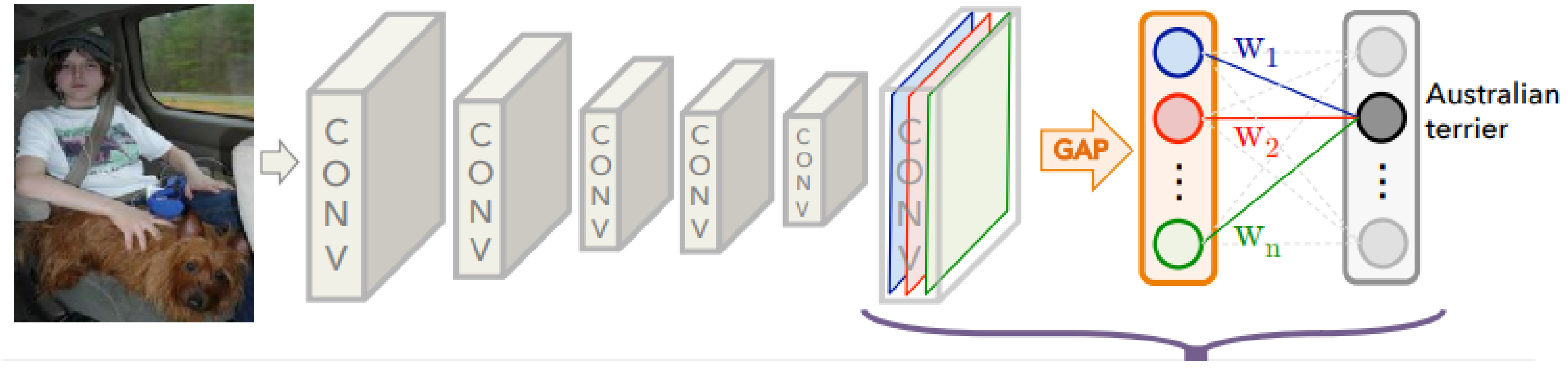


기존 CNN은 object를 localize하는 능력이 있지만, FC layer에서 손실되는 경우가 있었다.

Localization ability를 마지막 layer까지 유지하기 위해 Global Average Pooling 방식을 고안했다.

Abstract- Global Average Pooling





전체 object의 localization을 식별하고, overfitting을 방지하기 위한 regularization으로 이용. 평균값을 추출하여 Pooling.

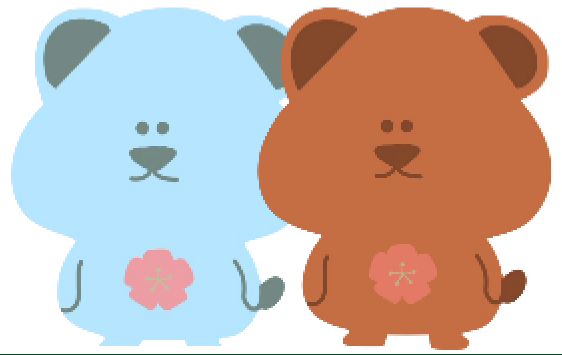
abstract

Our approach is trained end-to-end and can localize objects in a single forward pass.

Max pooling보다 average pooling 쓰는 이유는 the use of average pooling encourages the network to identify the complete extent of the object. the loss for average pooling benefits when the network identifies all discriminative regions of an object as compared to max pooling.

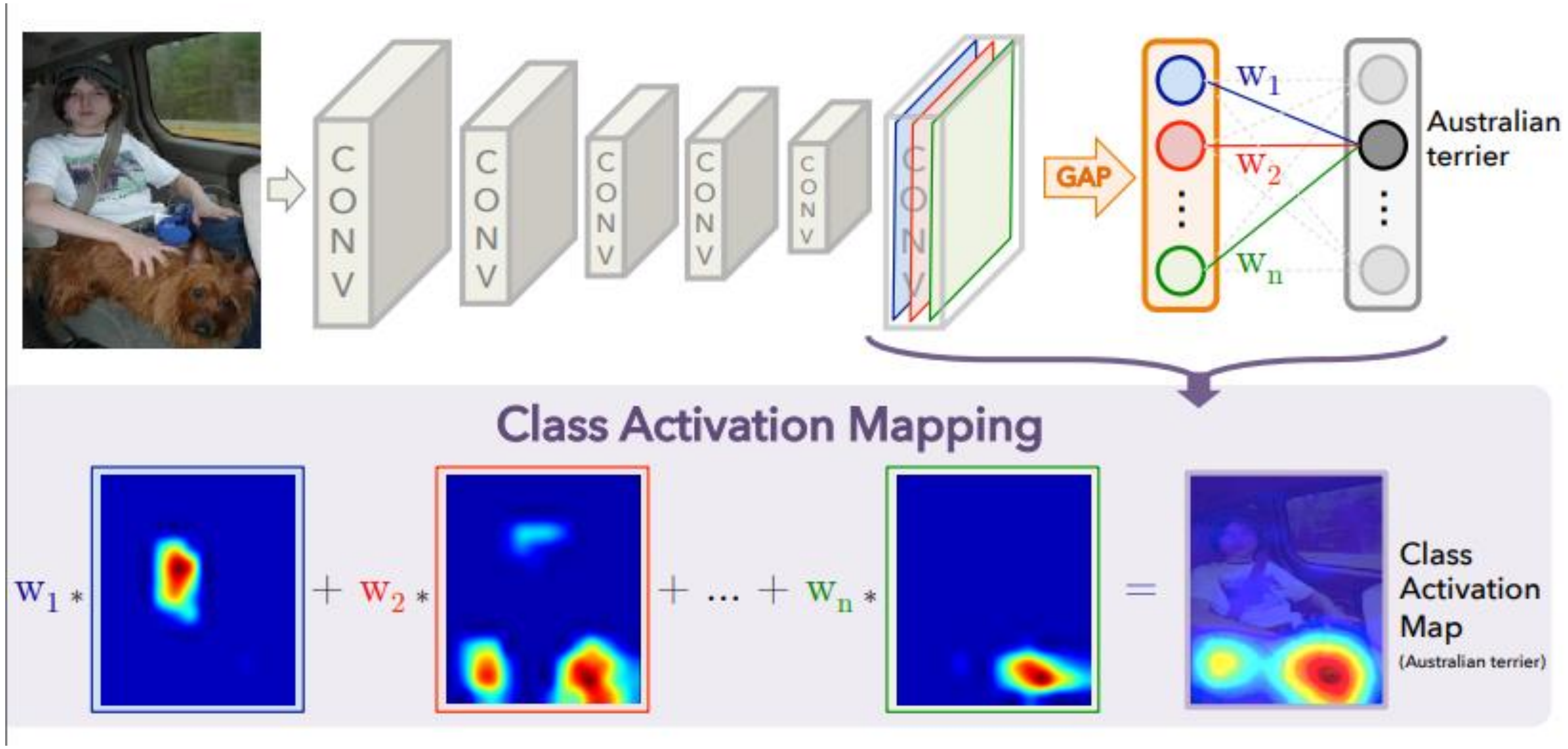
this localization ability is generic and can be observed even for problems that the network was not trained on.

contribution



EWHA EURON

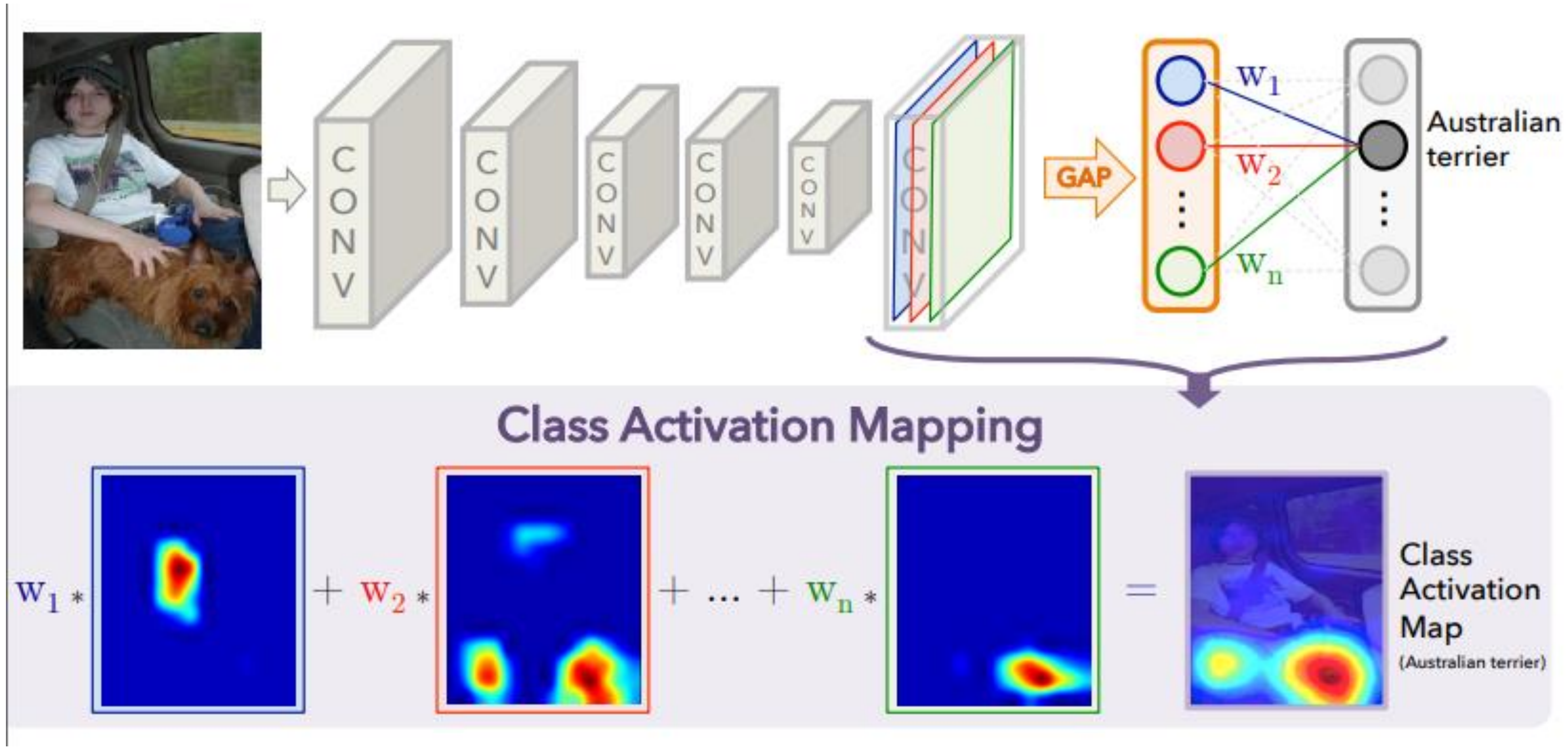
contribution – Class Activation Maps (CAM)



Convolution 층 + Global Average Pooling(GAP) + Softmax 모델 구조

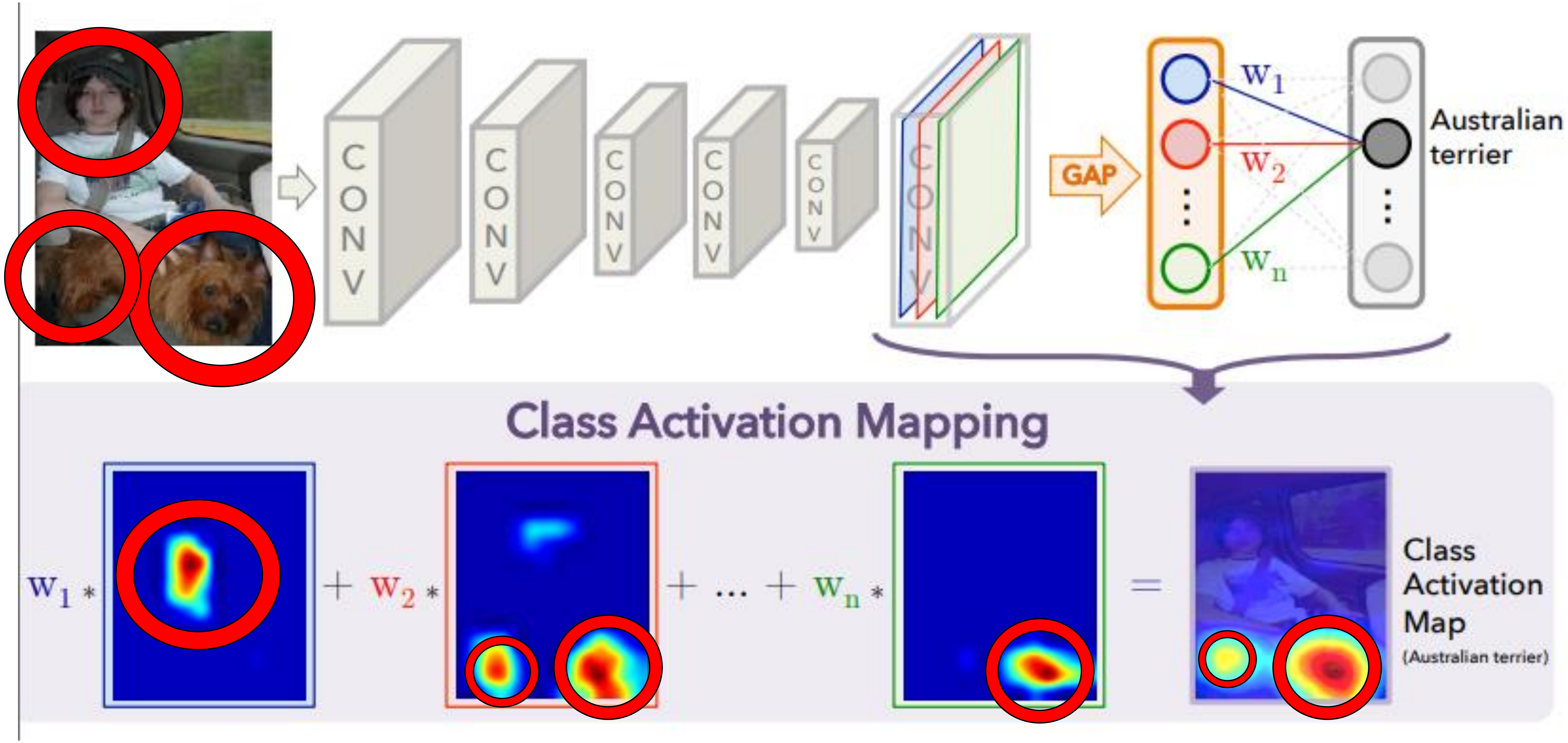
--> CAM을 통해 모델이 output을 낼 때 **집중한 영역을 확인**해볼 수 있다.

contribution – Class Activation Maps (CAM)

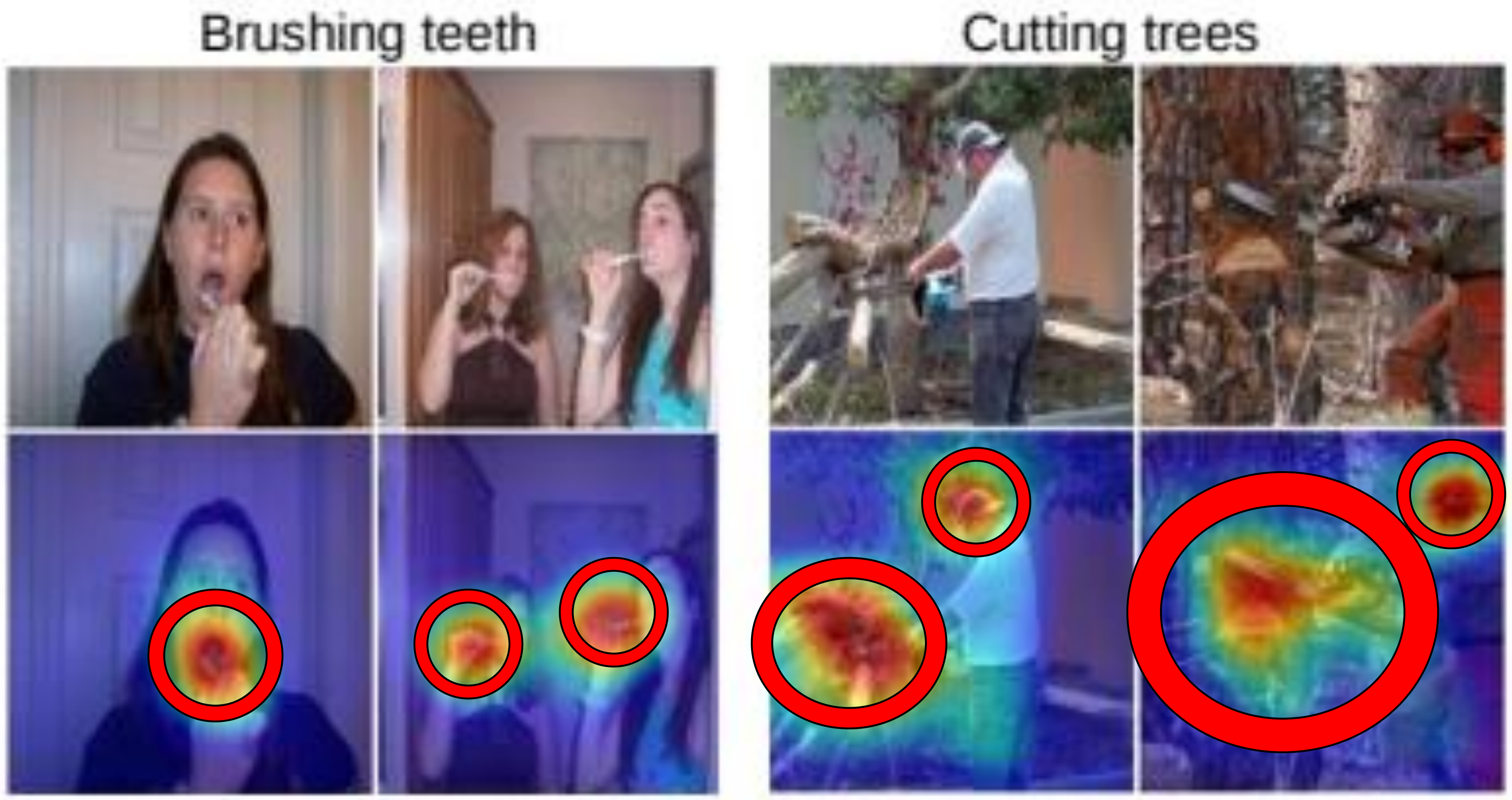


네트워크가 훈련되지 않아도 CAM을 통해 다양한 작업에 대해 the discriminative image regions in a single forward pass 쉽게 식별할 수 있다.

contribution – Class Activation Maps (CAM)

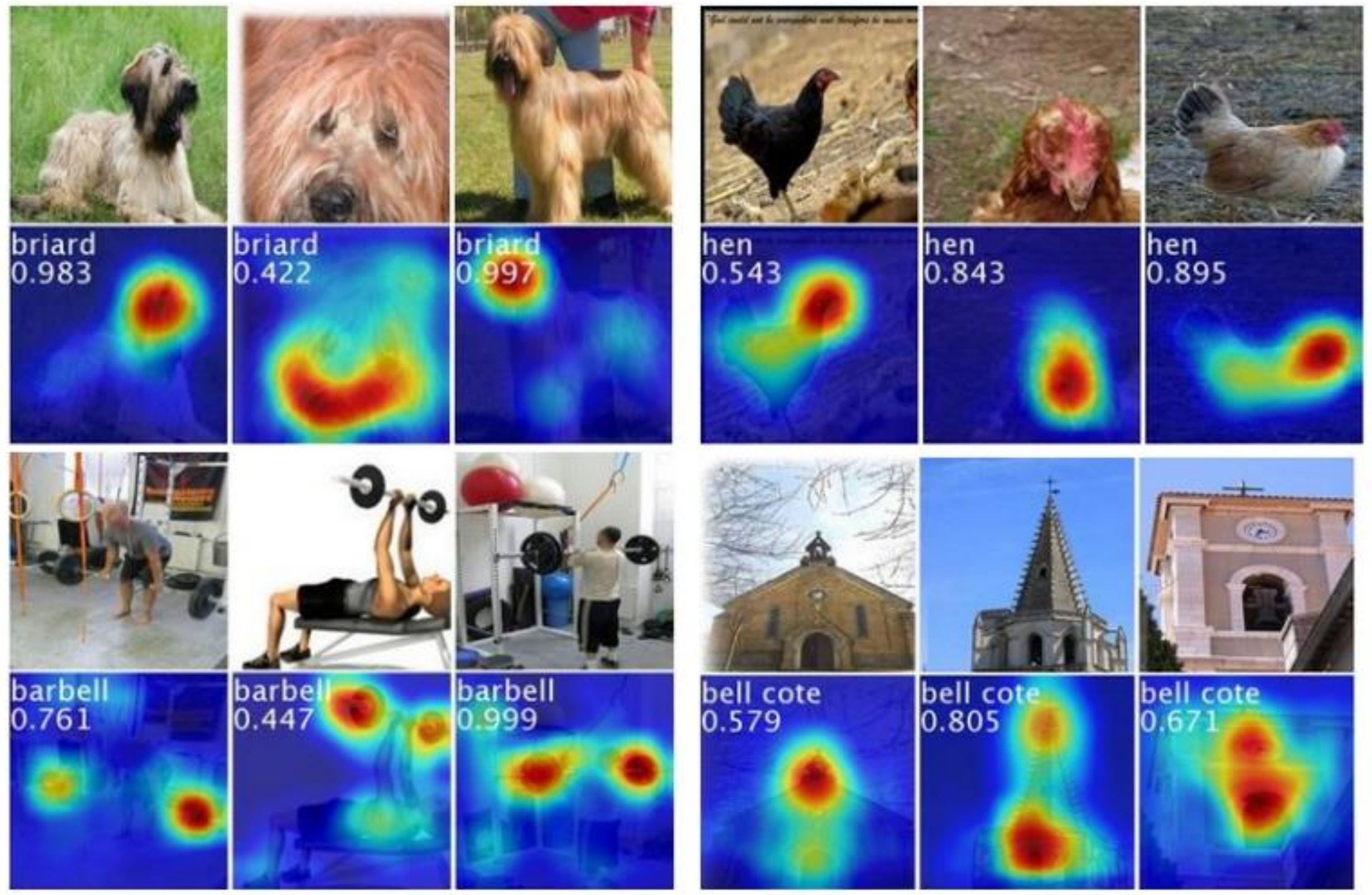


contribution – Class Activation Maps (CAM)



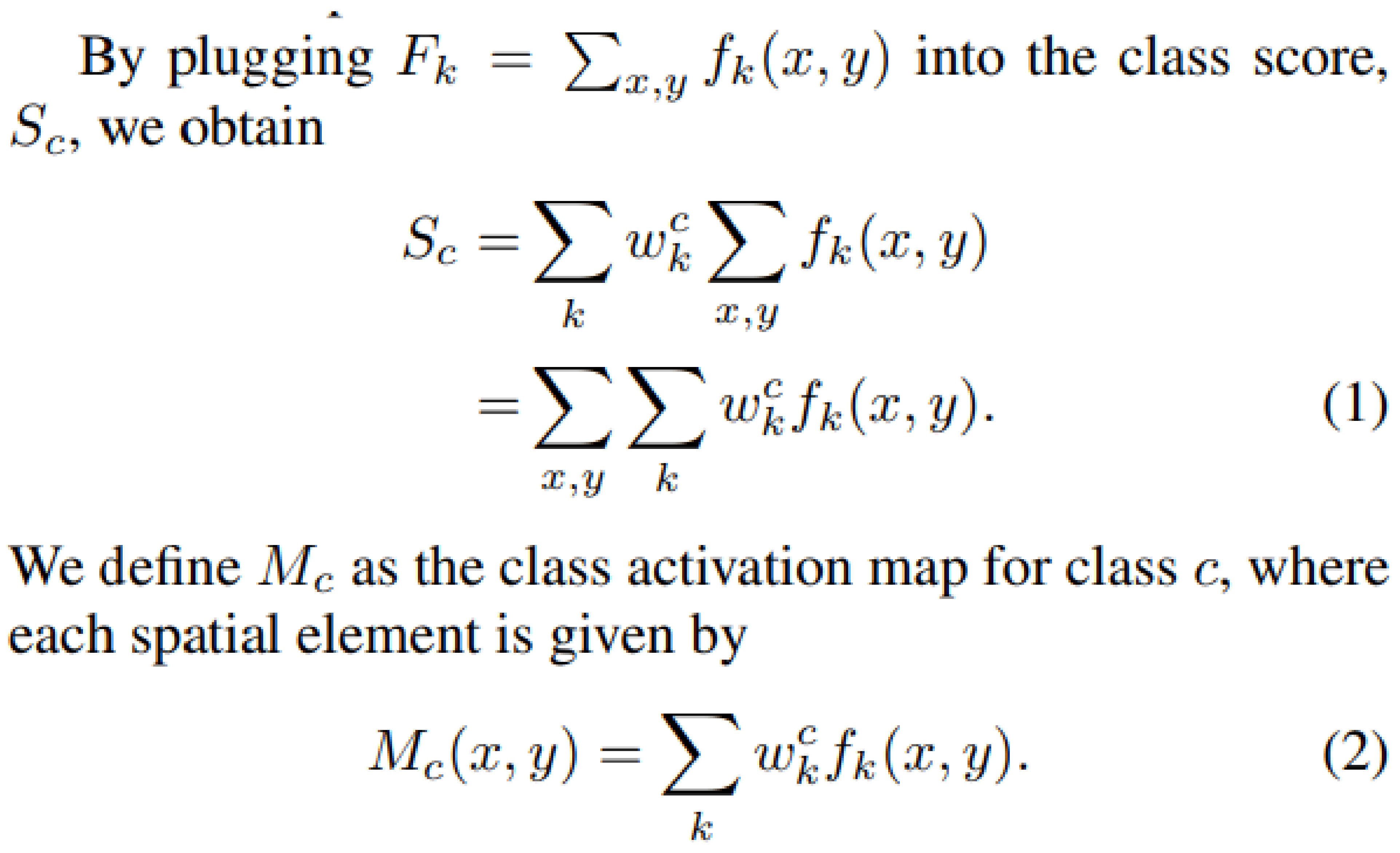
CAM+GAP을 통해 이미지를 분류하고 특정 부분을 localize한다.

contribution – Class Activation Maps (CAM)



A class activation map for a particular category indicates the discriminative image regions used by the CNN to identify that category

contribution – Class Activation Maps (CAM)



By plugging $F_k = \sum_{x,y} f_k(x,y)$ into the class score, S_c, we obtain

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y). \quad (1)$$

We define M_c as the class activation map for class c, where each spatial element is given by

$$M_c(x,y) = \sum_k w_k^c f_k(x,y). \quad (2)$$

fk(x,y) : 공간 벡터의 (x,y)의 마지막 컨볼루션 레이어의 활성화를 나타낸다.

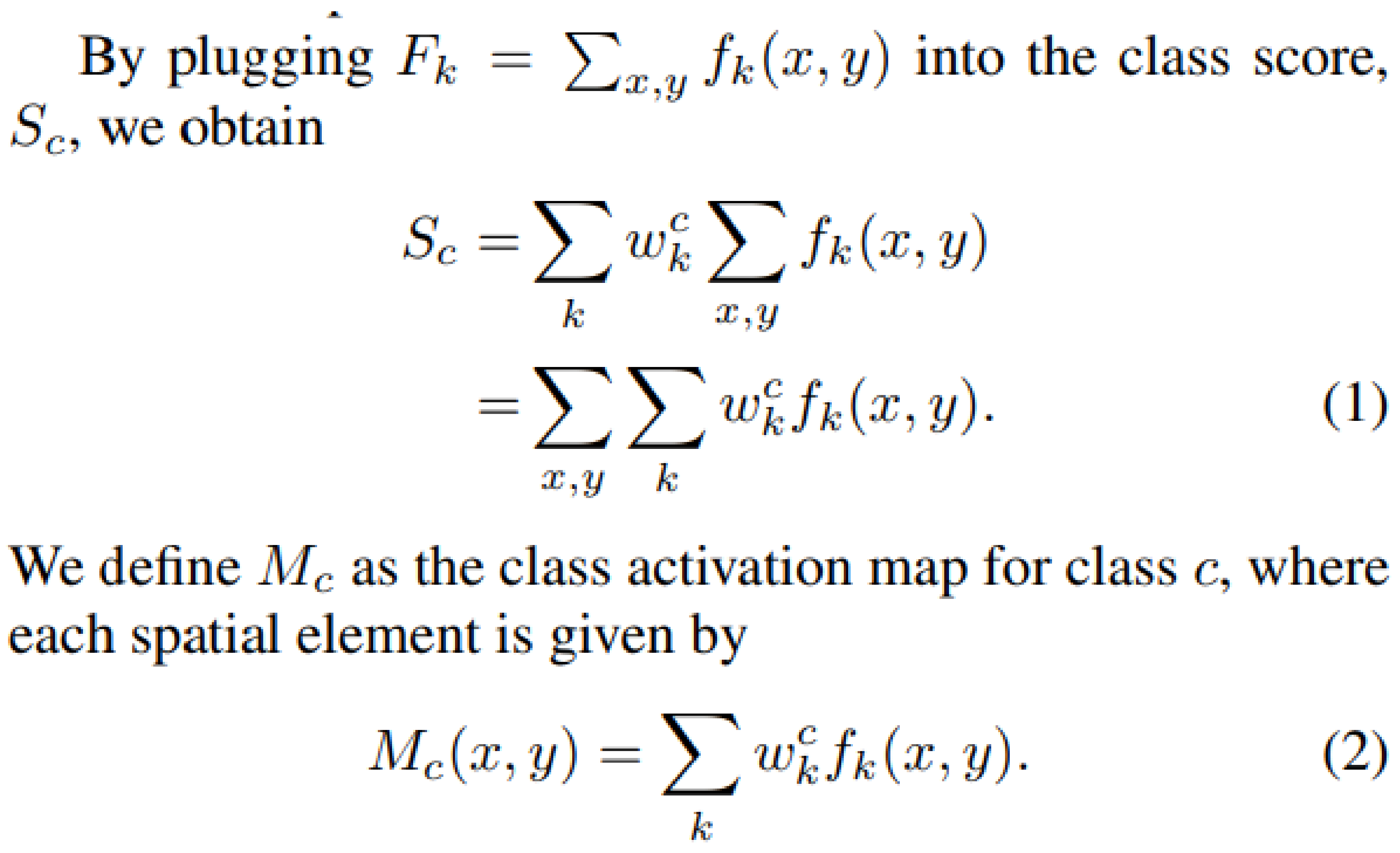
Fk = fk(x,y)의 합

Sc = Wkc * Fk 의 합, softmax의 input이 됨

Wkc = 클래스 c에 대응하는 가중치, 즉 Fk의 중요성을 나타냄(집중도)

Pc = softmax output

contribution – Class Activation Maps (CAM)



By plugging $F_k = \sum_{x,y} f_k(x,y)$ into the class score, S_c, we obtain

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y). \quad (1)$$

We define M_c as the class activation map for class c, where each spatial element is given by

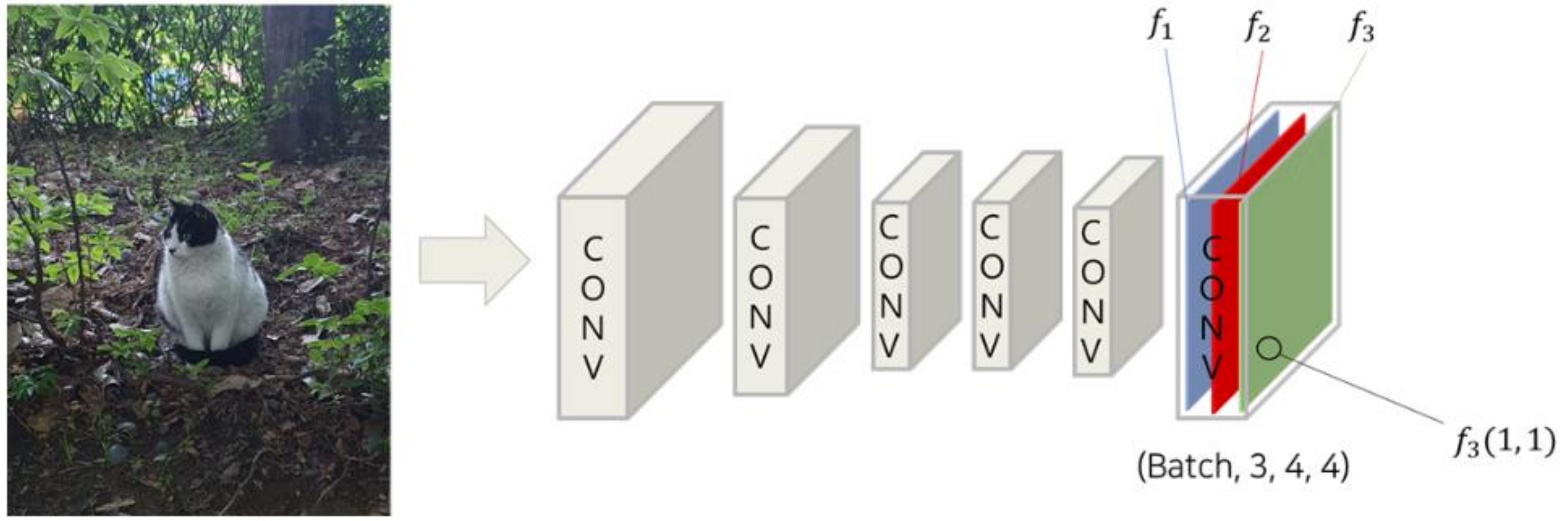
$$M_c(x,y) = \sum_k w_k^c f_k(x,y). \quad (2)$$

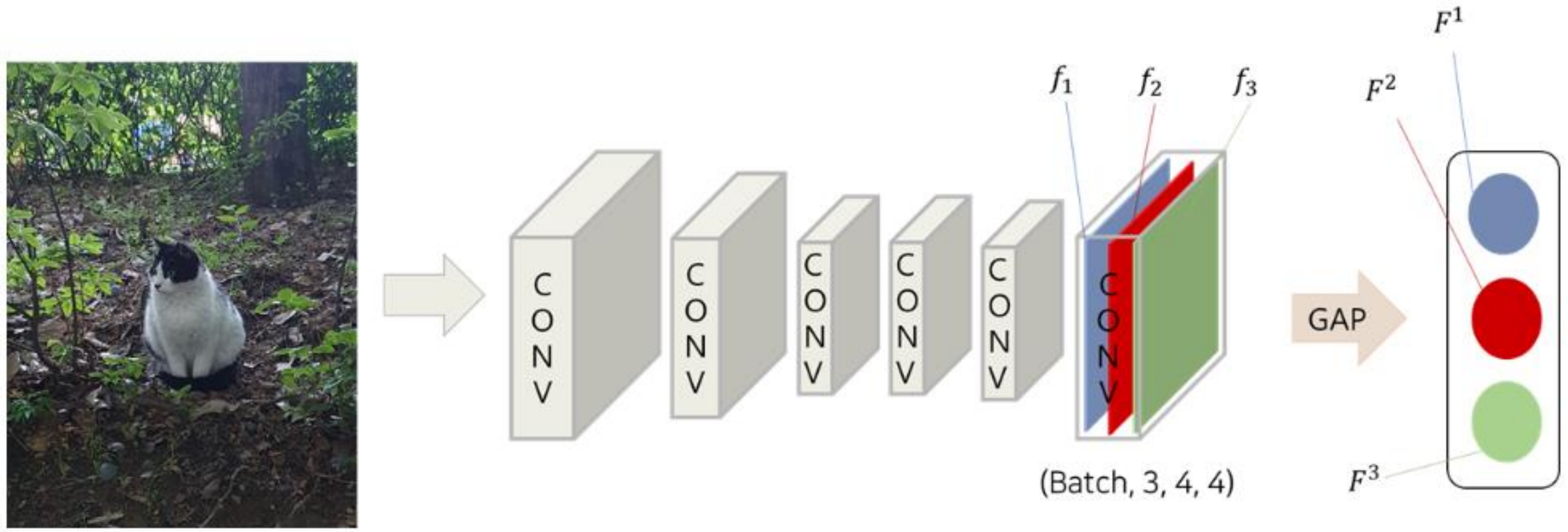
클래스 c의 이미지로 분류하는 공간 grid (x,y)에서 활성화의 중요성(집중도)를 의미

→ 집중한 영역 확인 가능

(we can identify the image regions most relevant to the particular category)

contribution – Class Activation Maps (CAM)





Last convolutional layer의 크기를 (Batch, 3, 4, 4)라고 가정.

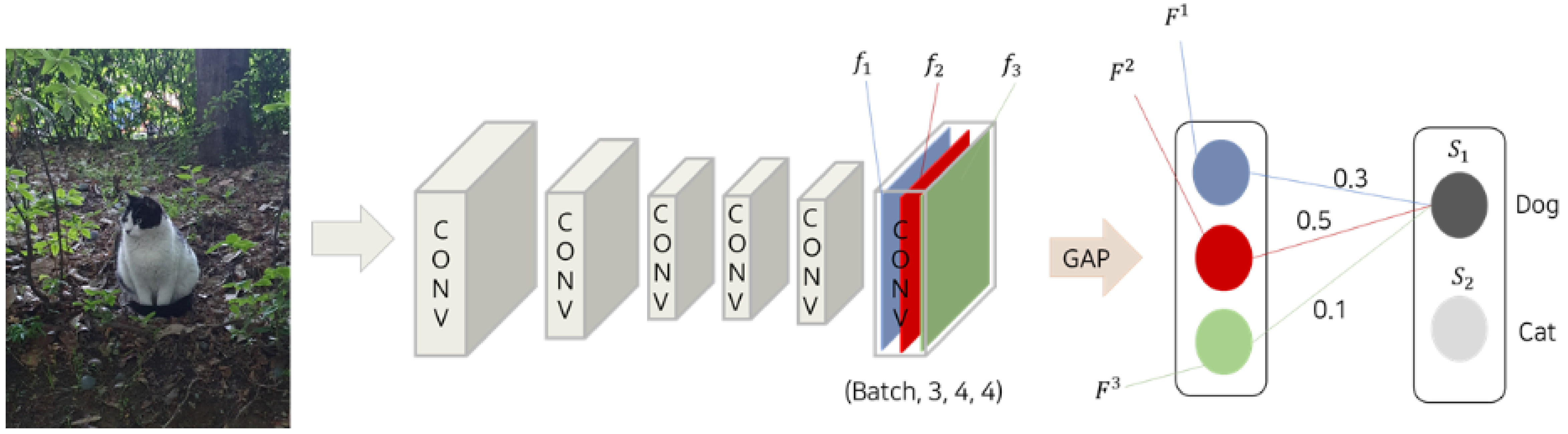
채널이 3개이므로 k=3

3번째 채널의 (1,1) 위치 = f3(1,1)

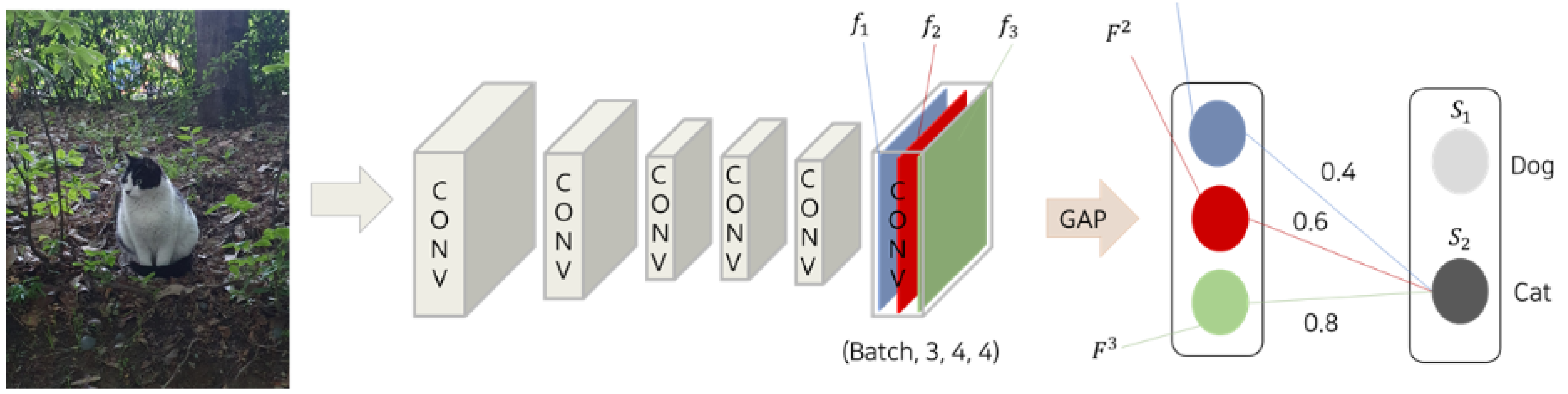
각 k에 대해 GAP수행한 결과 = Fk

채널의 평균 값으로 하나의 값이 되기 때문에 하나의 데이터로 표현이 됨.

contribution – Class Activation Maps (CAM)



$$S_1 = F^1 \times 0.3(w_1^1) + F^2 \times 0.5(w_2^1) + F^3 \times 0.1(w_3^1)$$



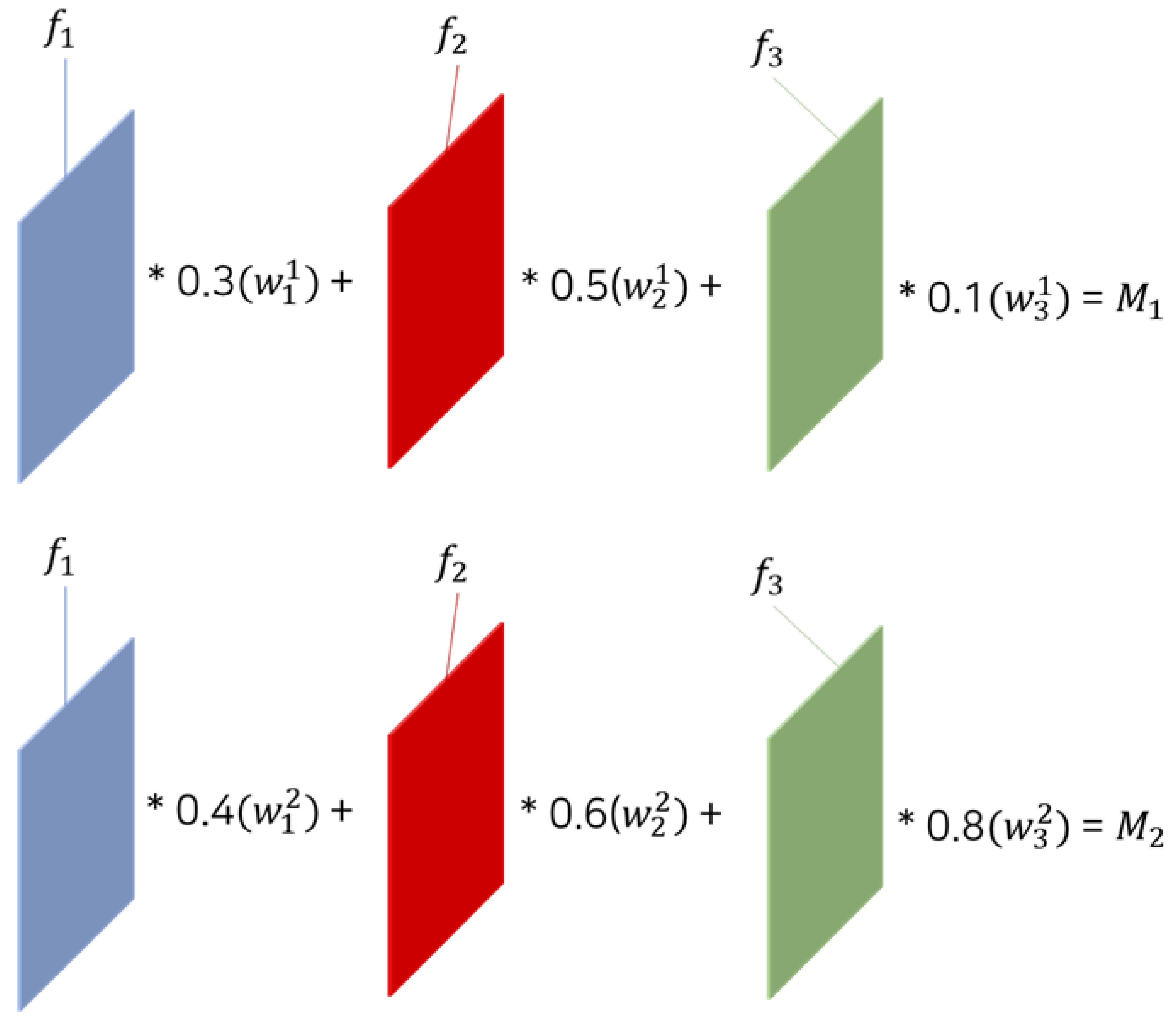
$$S_2 = F^1 \times 0.4(w_1^2) + F^2 \times 0.6(w_2^2) + F^3 \times 0.8(w_3^2)$$

Softmax input은 Fk*wk값으로 나타냄

wk은 class c에 대한 Fk의 importance를 나타냄

최종 class c에 대한 확률값 계산

contribution – Class Activation Maps (CAM)

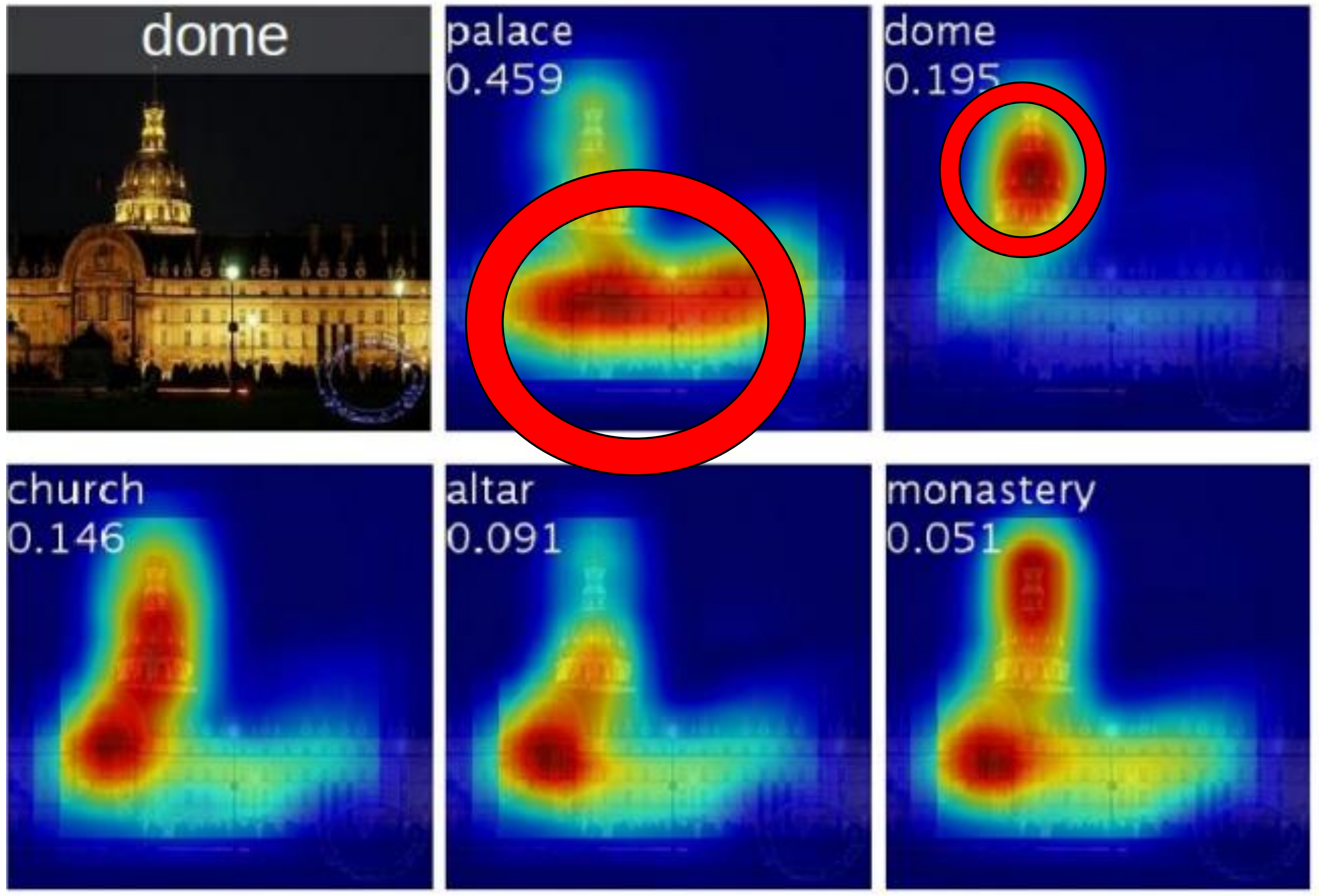


Sc가 Mc(class activation map)이며

Image의 분류가 class c가 되도록 하는 spatial grid (x,y)에서의 activation의 importance를 나타낸다.

→ 집중한 영역 확인 가능

contribution – Class Activation Maps (CAM)

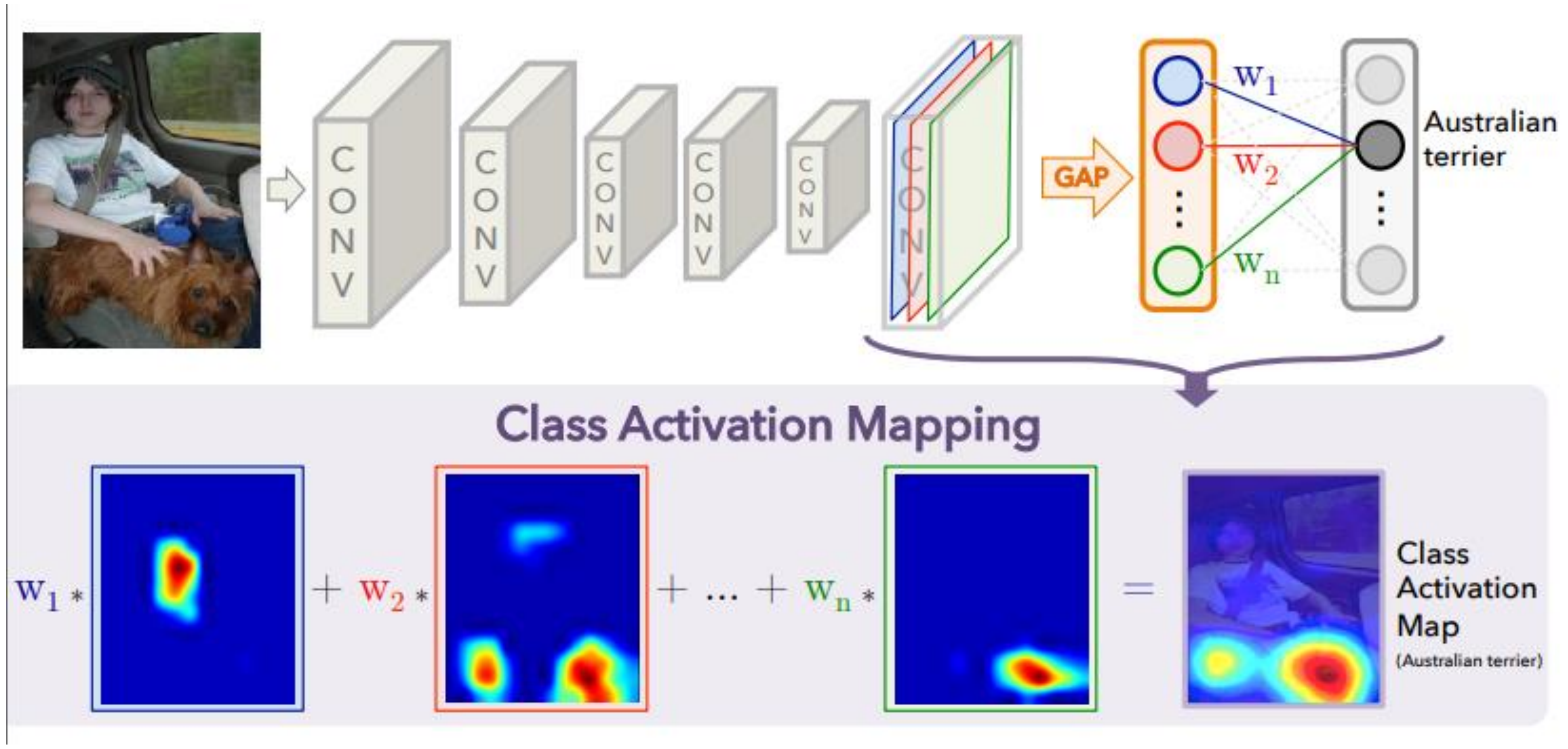


예측된 클래스와 해당 score은 각 클래스 CAM에 표시된다. 강조 표시된 영역이 예측된 클래스에 따라 다르다는 것을 알 수 있다.

예를 들어, 돔 class는 상부 둥근 부분을 활성화하고, palace class는 하부 평평한 부분을 활성화한다.

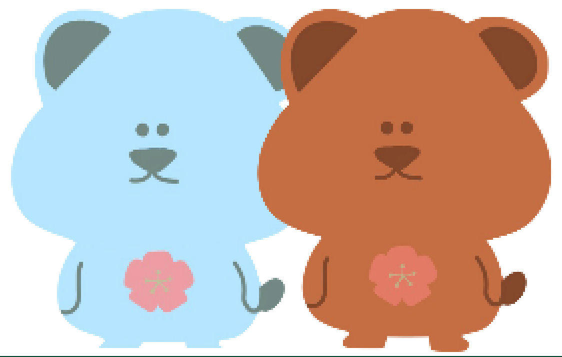
we highlight the differences in the CAMs for a single image when using different classes c to generate the maps.

contribution – Class Activation Maps (CAM)



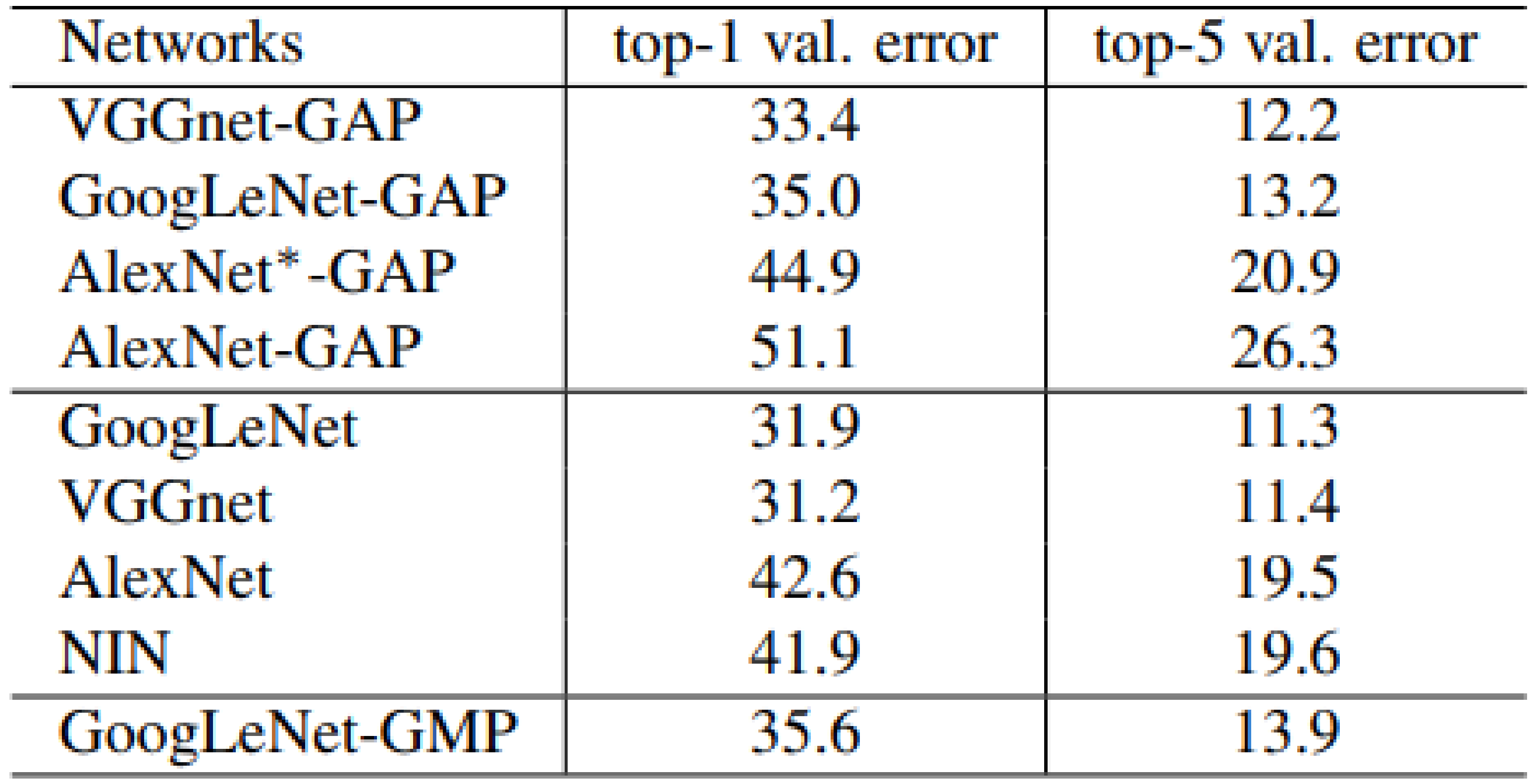
Australian terrier를 분류할 때, 사람의 얼굴에 집중한 첫 번째 activation map의 W1은 낮은 값일 것이고 개의 특징에 주목한 두 번째, n 번째 activation map에 연결된 W2,Wn은 높은 값을 가질 것으로 유추할 수 있다. 즉 학습된 이미지 위주로 가중치를 올려 집중하는 경향이 있다.

experiments



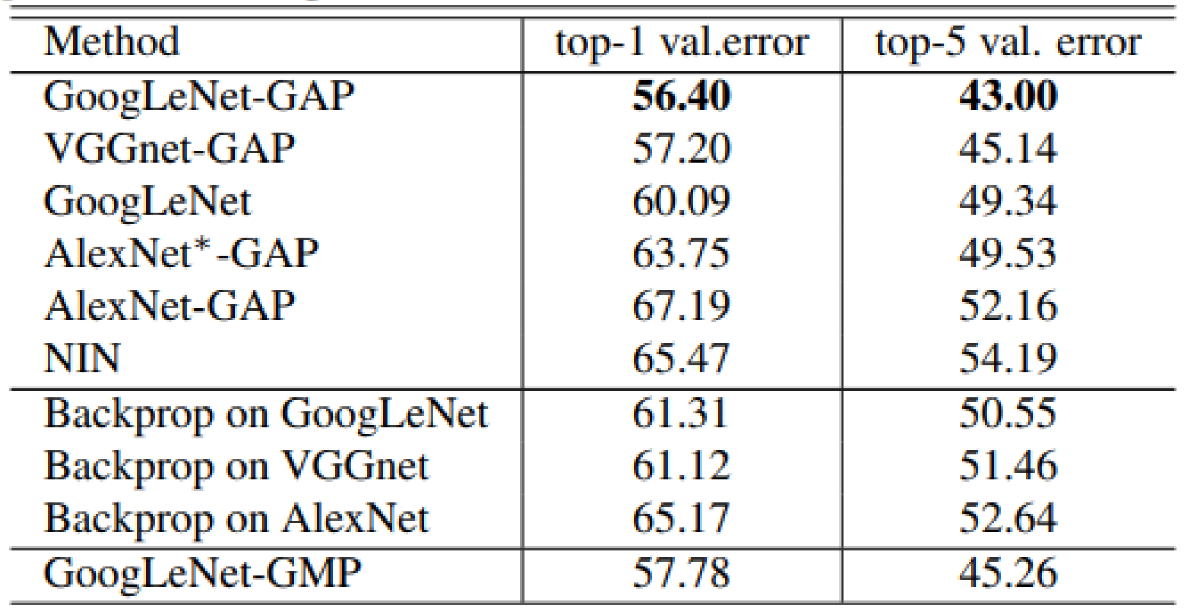
contribution – experiments

Table 1. Classification error on the ILSVRC validation set.



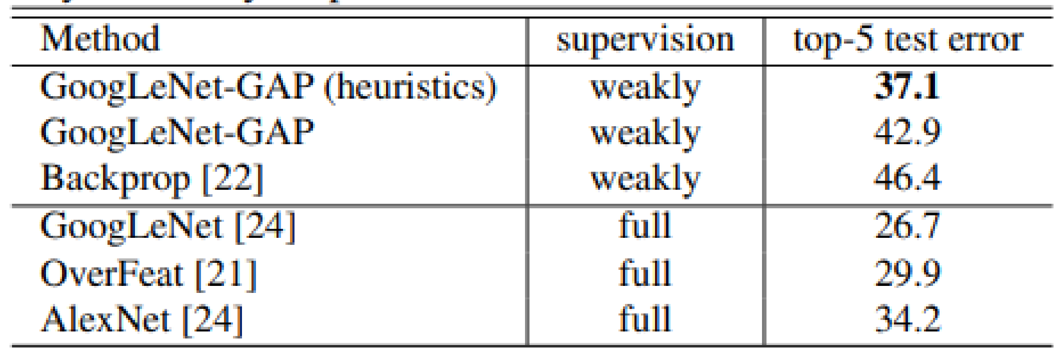
Networks	top-1 val. error	top-5 val. error
VGGnet-GAP	33.4	12.2
GoogLeNet-GAP	35.0	13.2
AlexNet*-GAP	44.9	20.9
AlexNet-GAP	51.1	26.3
GoogLeNet	31.9	11.3
VGGnet	31.2	11.4
AlexNet	42.6	19.5
NIN	41.9	19.6
GoogLeNet-GMP	35.6	13.9

Table 2. Localization error on the ILSVRC validation set. *Back prop* refers to using [22] for localization instead of CAM.



Method	top-1 val.error	top-5 val. error
GoogLeNet-GAP	**56.40**	**43.00**
VGGnet-GAP	57.20	45.14
GoogLeNet	60.09	49.34
AlexNet*-GAP	63.75	49.53
AlexNet-GAP	67.19	52.16
NIN	65.47	54.19
Backprop on GoogLeNet	61.31	50.55
Backprop on VGGnet	61.12	51.46
Backprop on AlexNet	65.17	52.64
GoogLeNet-GMP	57.78	45.26

Table 3. Localization error on the ILSVRC test set for various weakly- and fully- supervised methods.



Method	supervision	top-5 test error
GoogLeNet-GAP (heuristics)	weakly	**37.1**
GoogLeNet-GAP	weakly	42.9
Backprop [22]	weakly	46.4
GoogLeNet [24]	full	26.7
OverFeat [21]	full	29.9
AlexNet [24]	full	34.2

table 1, table 2를 비교했을 때 CAM을 사용한 Networks가 더 에러가 낮고 성능이 좋다.
특히 GMP(부분적 Pooling)일 때도 성능이 더 향상된 것을 볼 수 있다.

contribution – experiments

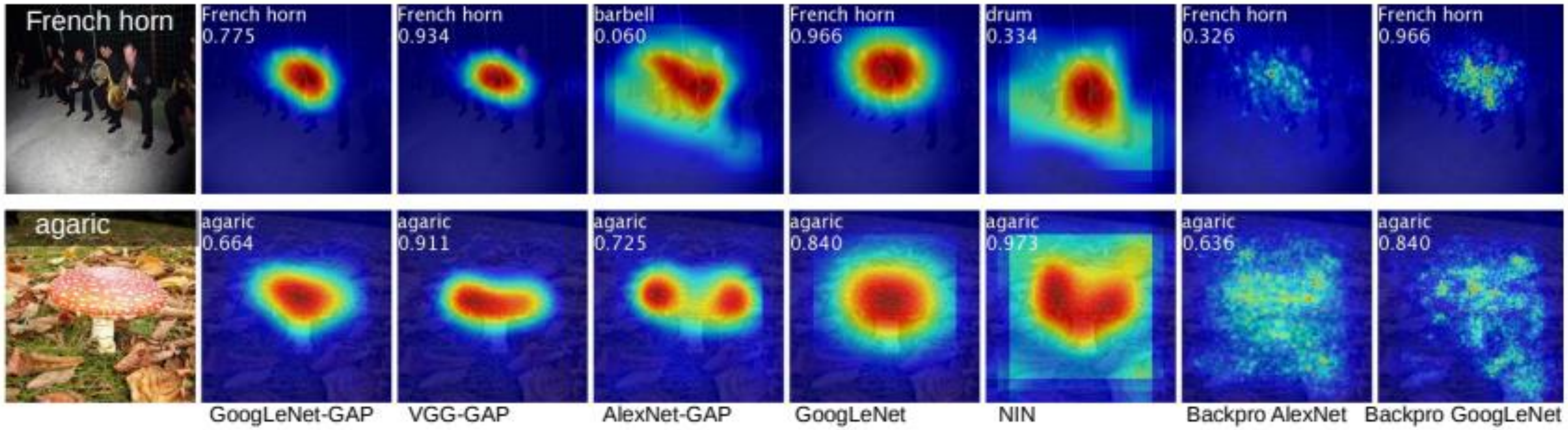
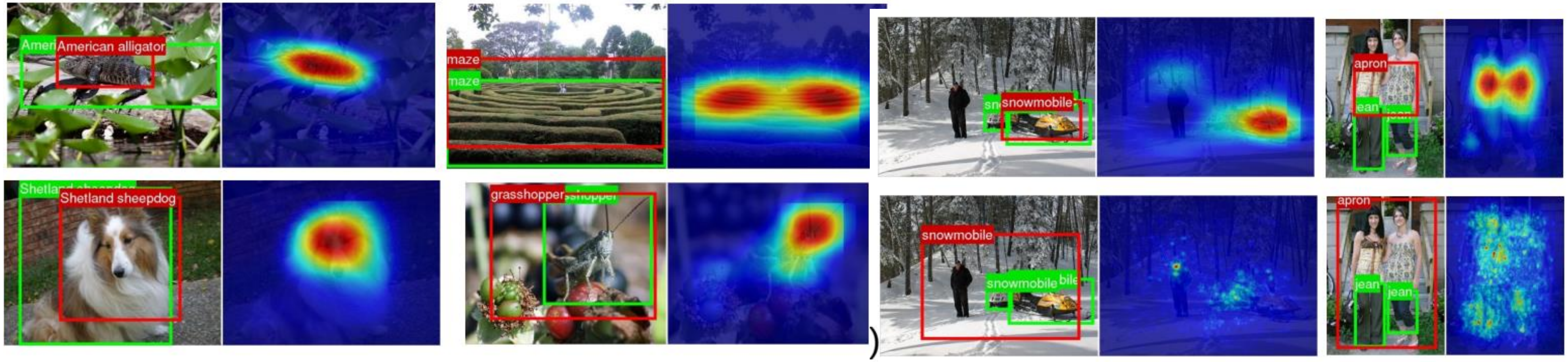


Figure 5. Class activation maps from CNN-GAPs and the class-specific saliency map from the backpropagation methods.

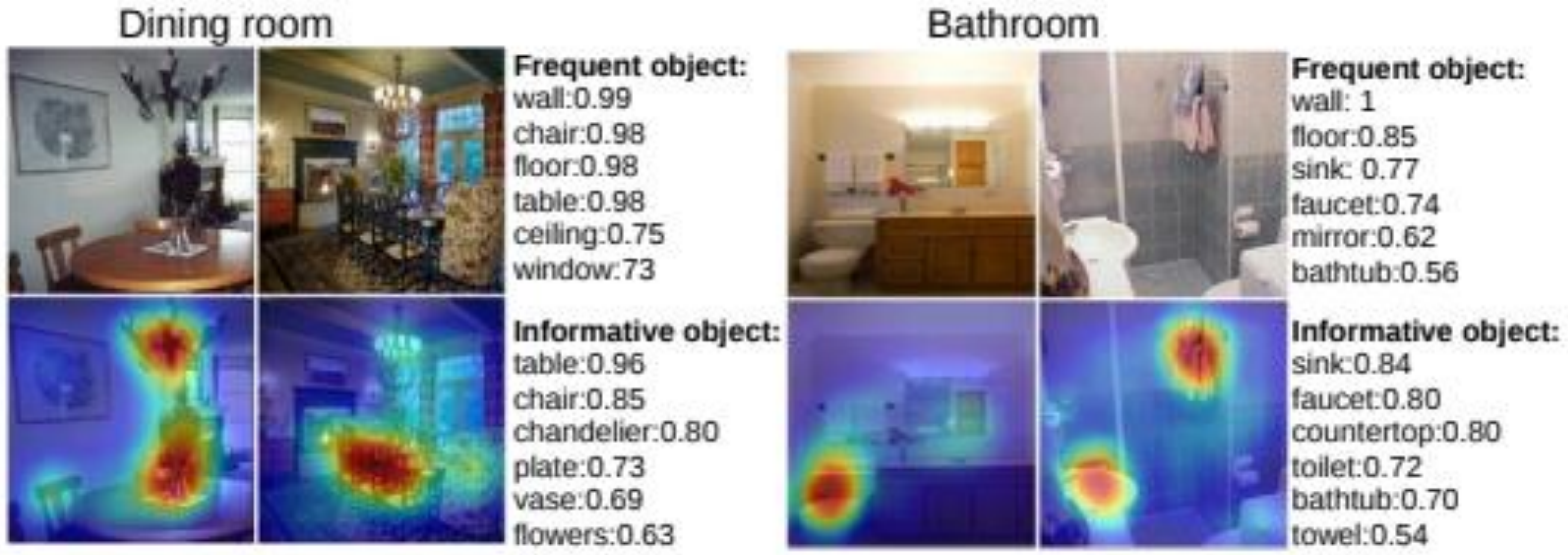
모델마다 집중한 영역을 확인해볼 수 있다. (CAM)

contribution – experiments



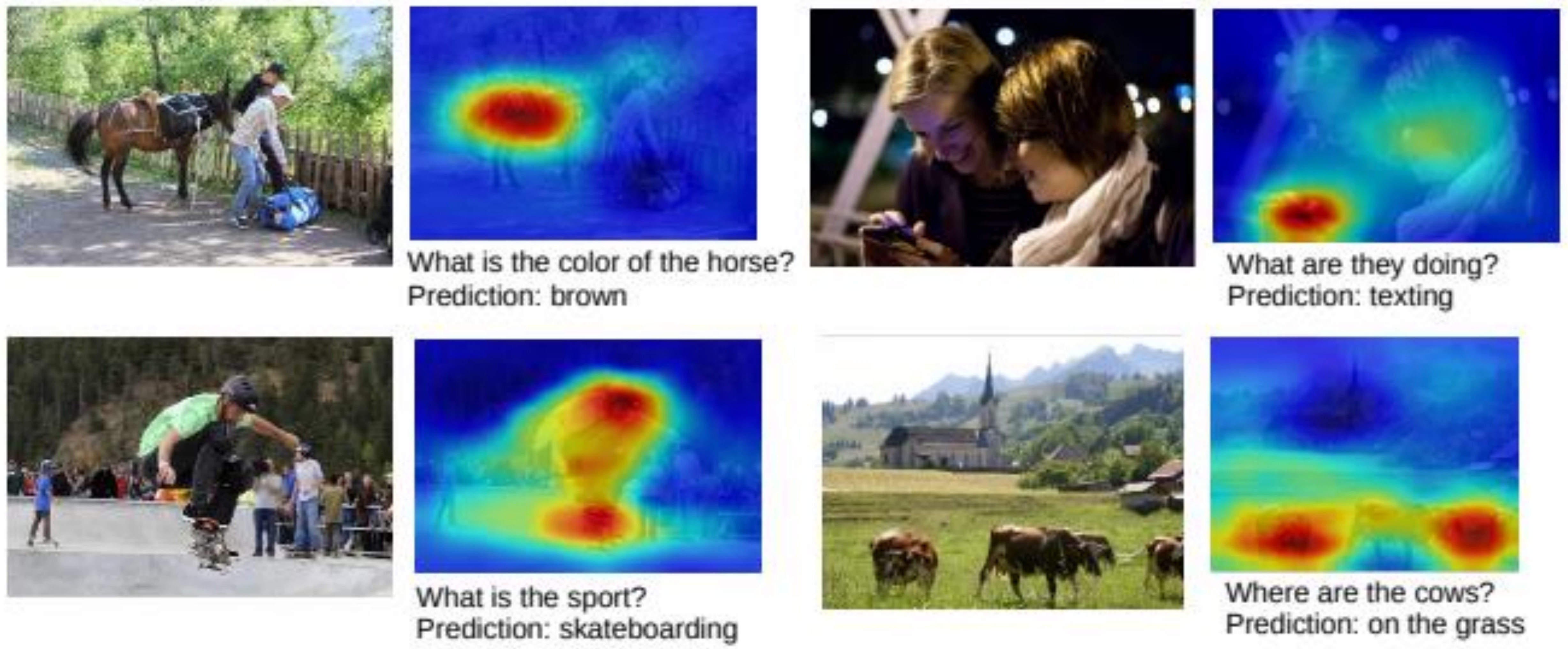
학습이 없이도 bounding box를 annotation한 방법이
backpropagation 방법보다 더 좋은 성능임을 확인할 수 있다.

contribution – experiments



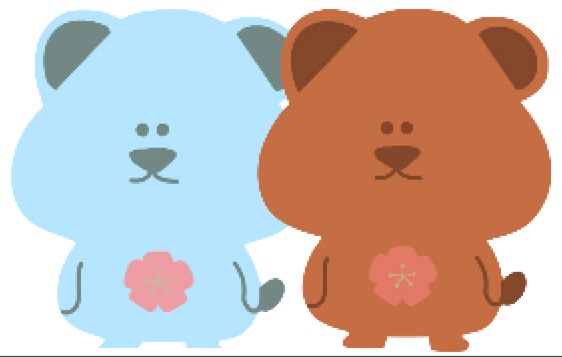
Informative한 objects를 인식하고 어떤 영역을 집중하고
판단했는지의 실험

contribution – experiments



Visual Question Answering 분야에서 predictor가 문제에 대한 답을 내릴 때 어느 영역을 집중하였는지 확인할 수 있는 실험

MediaPipe Hands: On-device Real-time Hand Tracking



#0. Abstract

Real-time on-device hand tracking solution
➡ predict a hand skeleton

Pipeline consisting of two models
1) A palm detector => a bounding box of a hand
2) A hand landmark model => predict the hand skeleton

https://mediapipe.dev

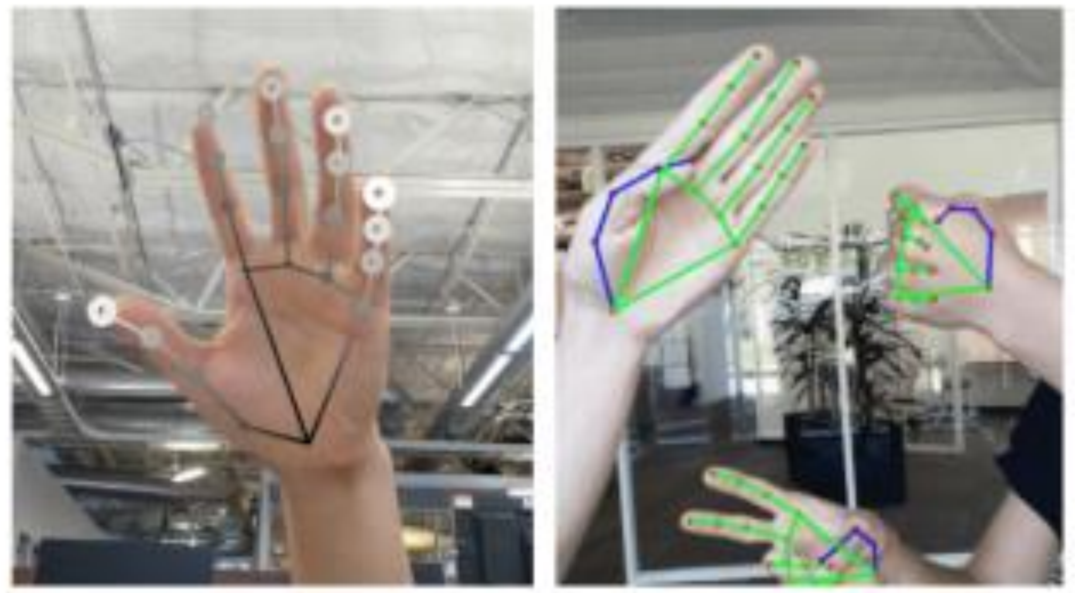


Figure 1: Rendered hand tracking result. (Left): Hand landmarks with relative depth presented in different shades. The lighter and larger the circle, the closer the landmark is towards the camera. (Right): Real-time multi-hand tracking on Pixel 3.

#1. Introduction

Previous work

1) Requires specialized hardware (depth sensors)

2) Not lightweight enough to run real-time on commodity mobile devices
   ➡ limited to platforms equipped with powerful processors

In this paper

1) No additional hardware

2) Performs in real-time on mobile devices

#1. Introduction

Main contributions

1) An efficient two-stage hand tracking pipeline that can <u>track multiple hands in real-time on mobile devices</u>

2) A hand pose estimation model that is capable of predicting 2.5D hand pose <u>with only RGB input</u>

3) <u>Open source</u> hand tracking pipeline as a ready-to-go solution on a variety of platforms, including Android, iOS, Web and desktop PCs

#2. Architecture

ML pipeline

1) A palm detector
    - On a full input image
    - Locate palms via an oriented hand bounding box

2) A hand landmark model
    - On the cropped hand bounding box
    - Return high-fidelity 2.5D landmarks

#3. Dataset and Annotation

1) **In-the-wild dataset**
- 6K images of large variety
- 👎 no complex articulation of hands

Great complements to each other to improve robustness

2) **In-house collected gesture dataset**
- 10K images
- Various angles of all physically possible hand gestures
- 👎only 30 people with limited variation in background

#3. Dataset and Annotation

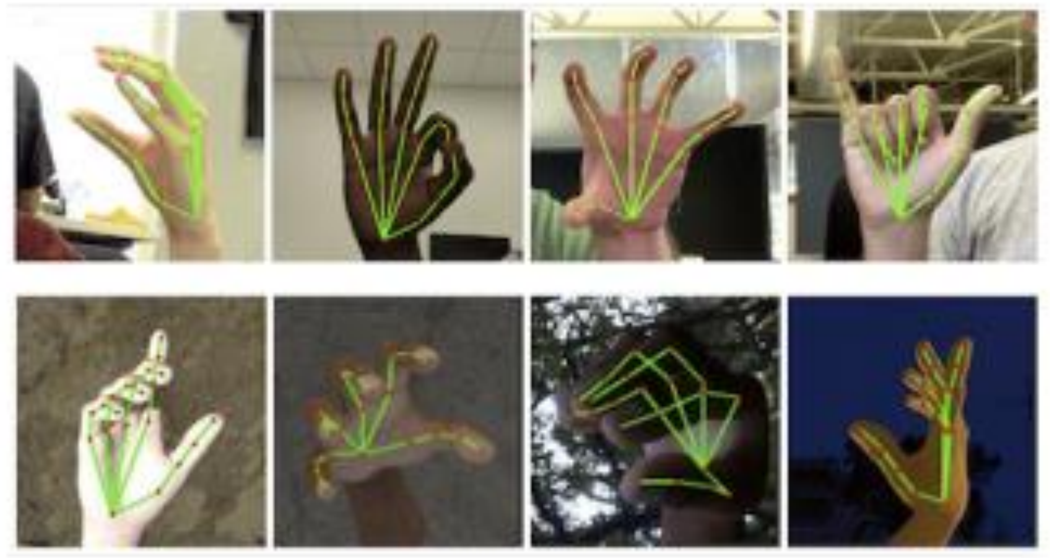


Figure 4: Examples of our datasets. (Top): Annotated real-world images. (Bottom): Rendered synthetic hand images with ground truth annotation. See Section 3 for details.

3) Synthetic dataset

a. render a high-quality synthetic hand model over <u>various backgrounds</u>
b. map it to the corresponding <u>3D coordinates</u>
➞ even better cover the possible hand poses and provide additional supervision for depth

- A commercial 3D hand model
  - 24 bones, 36 blendshapes – control fingers and palm thickness
  - 5 textures with different skin tones

#3. Dataset and Annotation

1) Palm detector

- Only "in-the-wild dataset"
  - Sufficient for localizing hands
  - Offer the highest variety

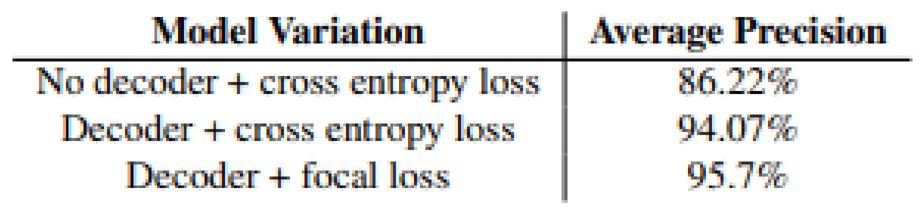
2) Landmark model

- All datasets

  => "In-the wild dataset", "In-house collected gesture dataset", "Synthetic dataset"
- Authors annotate the real-word images with 21 landmarks & use projected ground-truth 3D joints for synthetic images.

- For hand presence)
  - Select a subset of real-world images as positive examples
  - Sample on the region excluding annotated hand regions as negative examples
- For handedness)
  - Annotate a subset of real-world images with handedness to provide such data

#4. Results

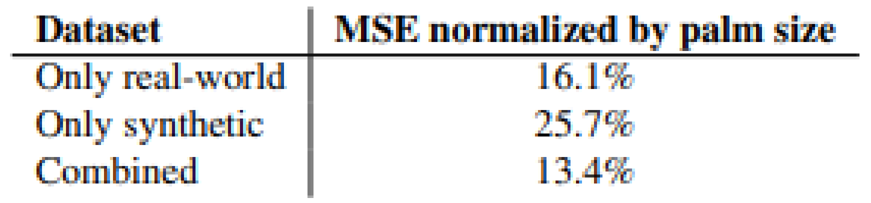


Model Variation	Average Precision
No decoder + cross entropy loss	86.22%
Decoder + cross entropy loss	94.07%
Decoder + focal loss	95.7%

Table 1: Ablation study of palm detector design elements of palm detector.

The hand landmark model

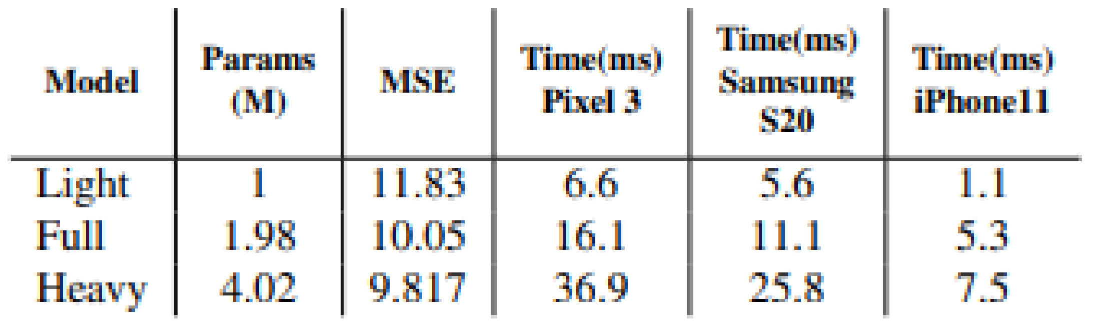
the best results
: the combination of real-world and synthetic datasets



Dataset	MSE normalized by palm size
Only real-world	16.1%
Only synthetic	25.7%
Combined	13.4%

Table 2: Results of our model trained from different datasets.

training with a large synthetic dataset
➡ less jitter
➡ authors' real-world dataset can be enlarged for better generalization



Model	Params (M)	MSE	Time(ms) Pixel 3	Time(ms) Samsung S20	Time(ms) iPhone11
Light	1	11.83	6.6	5.6	1.1
Full	1.98	10.05	16.1	11.1	5.3
Heavy	4.02	9.817	36.9	25.8	7.5

Table 3: Hand landmark model performance characteristics.

goal
: real-time performance on mobile devices
➡ experiments with different model sizes
➡ the "Full" model : good trade-off between quality and speed

#5. Conclusion

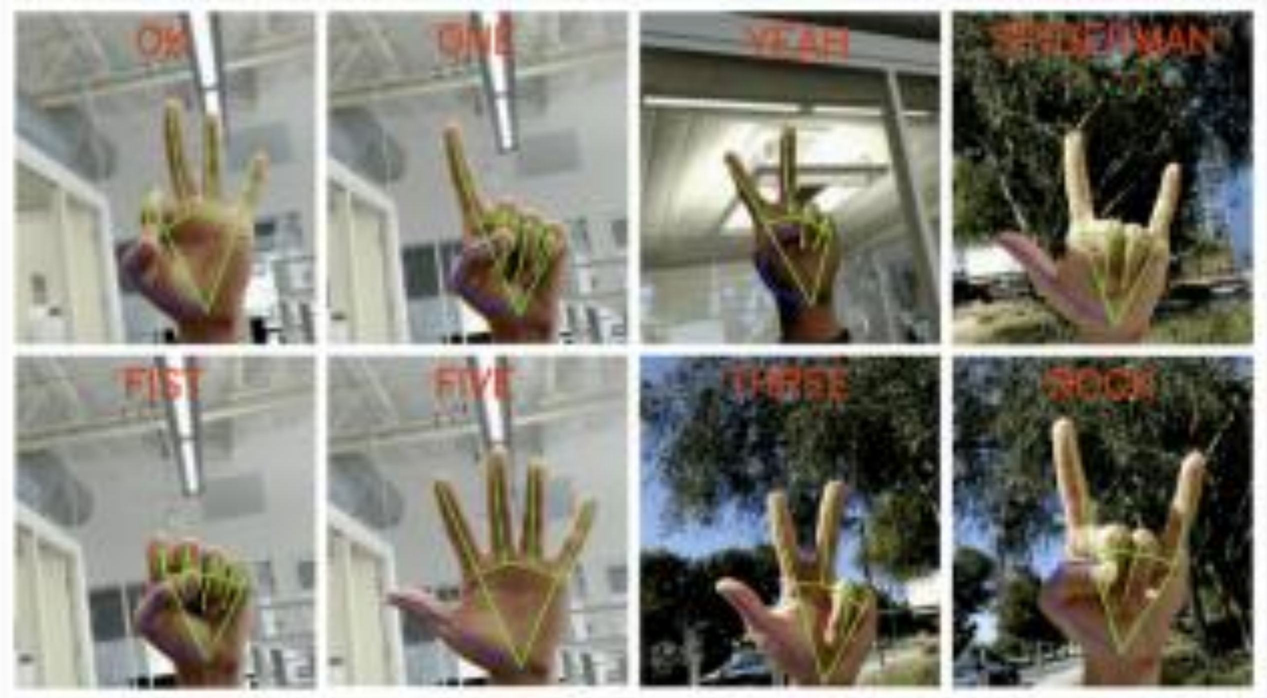
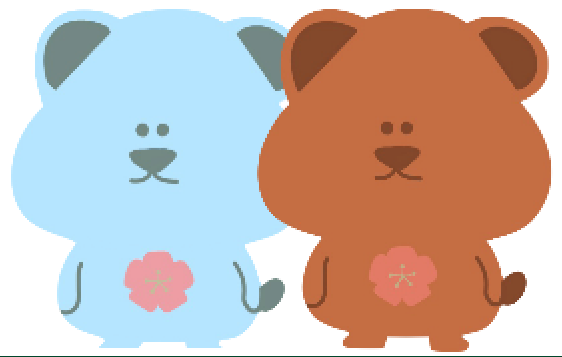


Figure 6: Screenshots of real-time gesture recognition. Semantics of gestures are rendered at top of the images.

Mediapipe Hands
: an end-to-end hand tracking solution that achieves real-time performance on multiple platforms

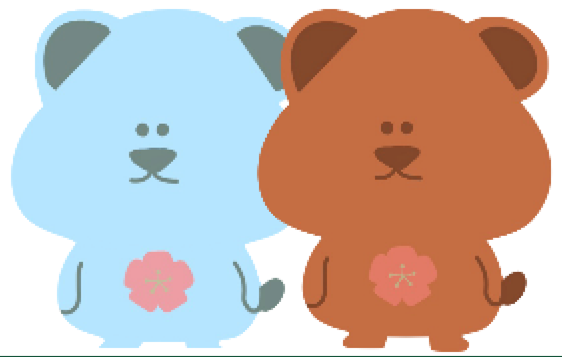
The pipeline predicts 2.5D landmarks without any specialized hardware and thus, can be easily deployed to commodity devices.

Mask R-CNN



EWHA EURON

Preview



01 Instance Segmentation

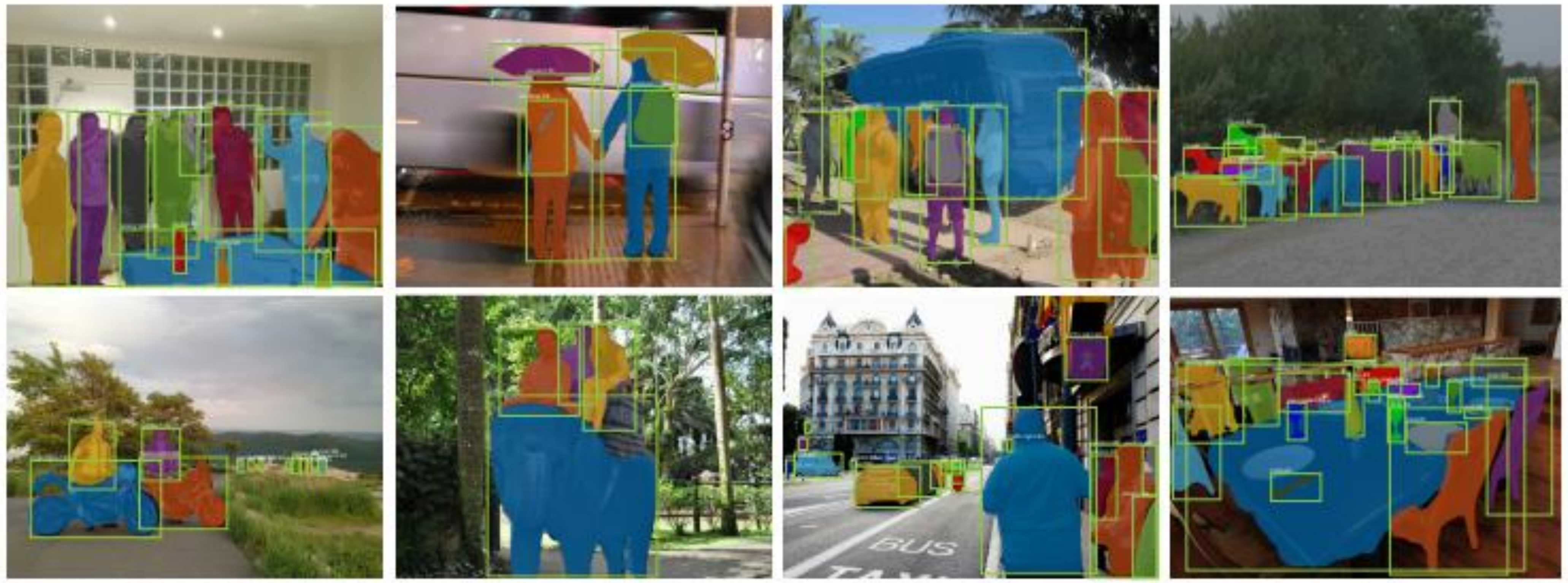
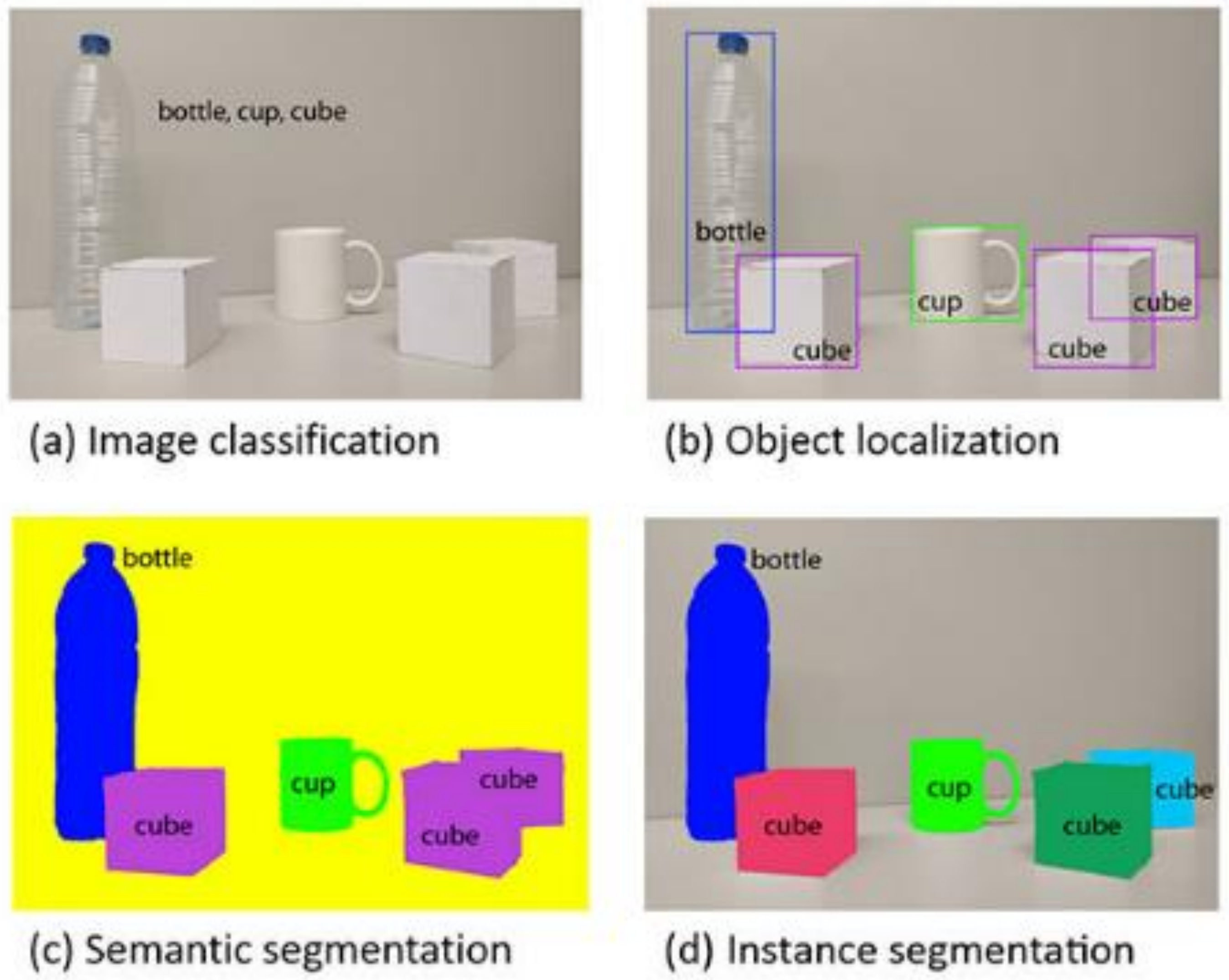
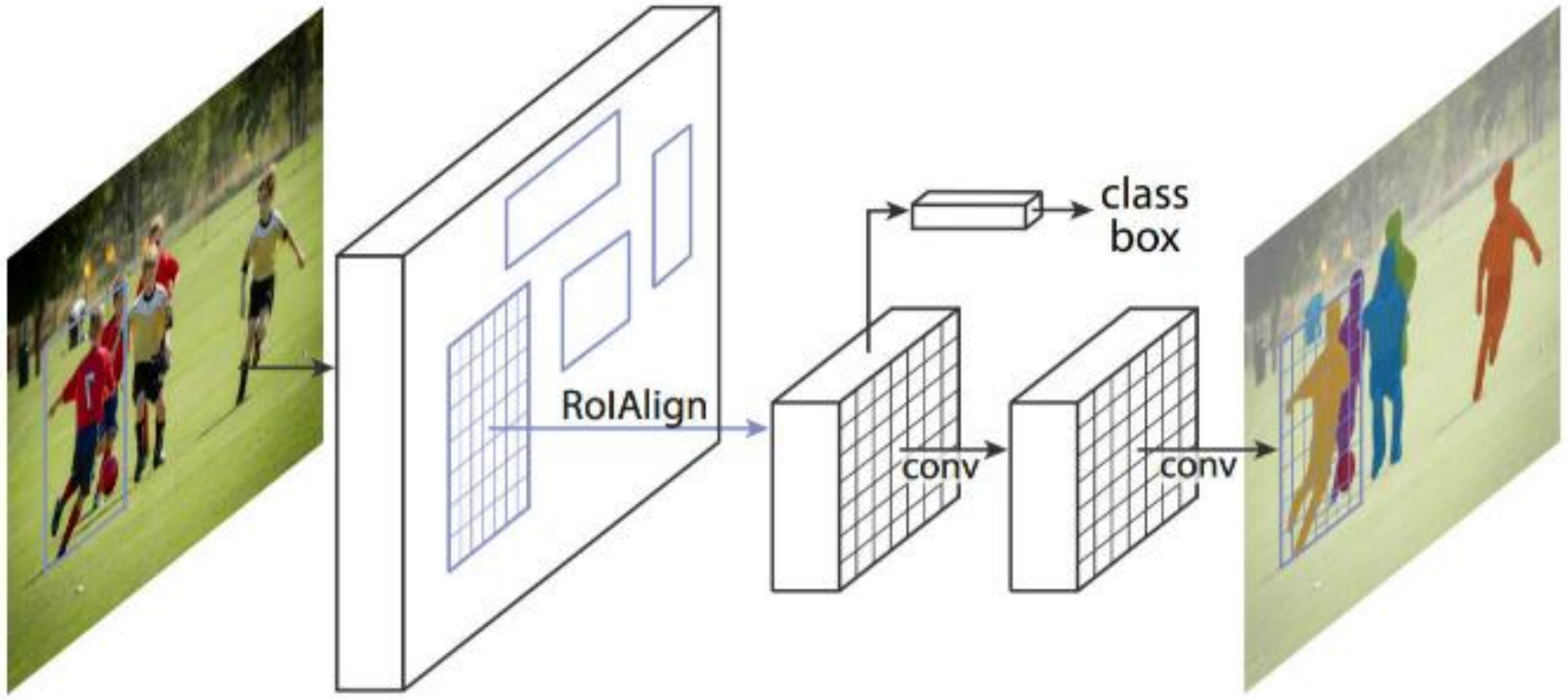


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.



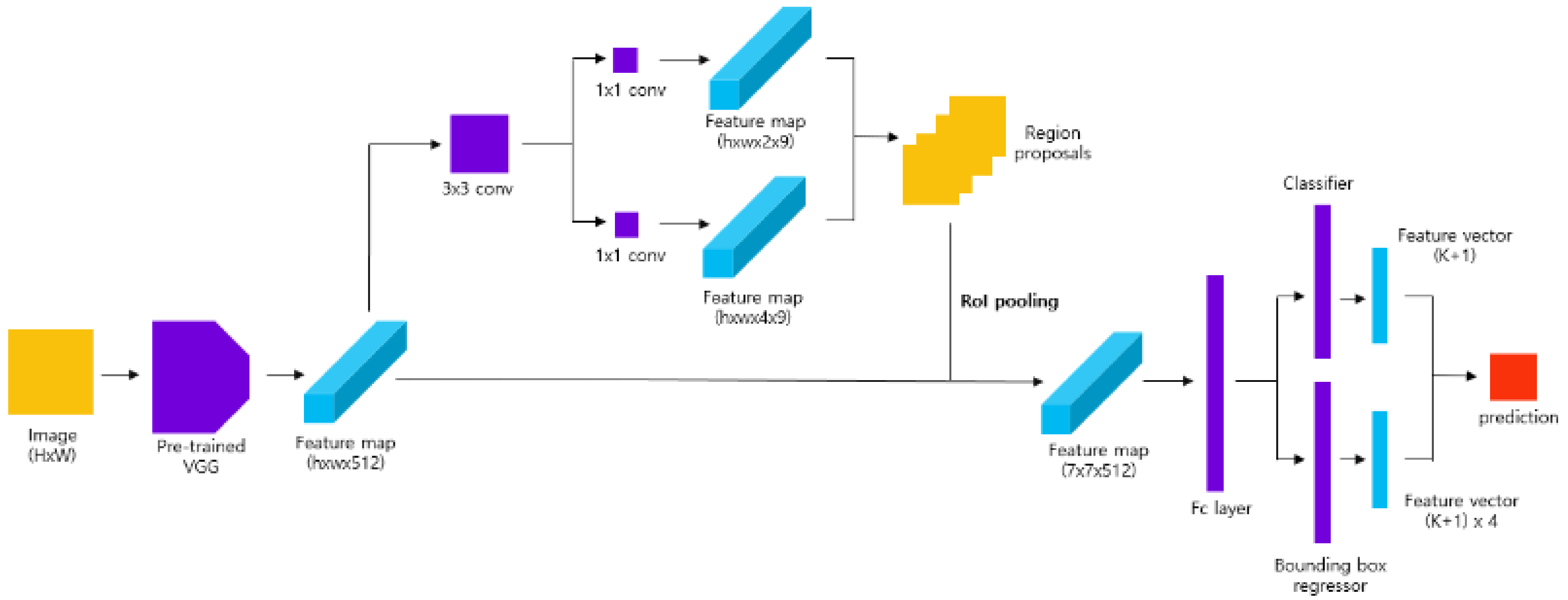
Instance segmentation

02 Architecture



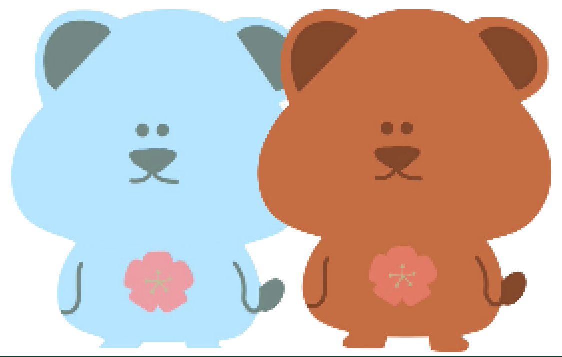
Mask R-CNN architecture

03 Faster R-CNN



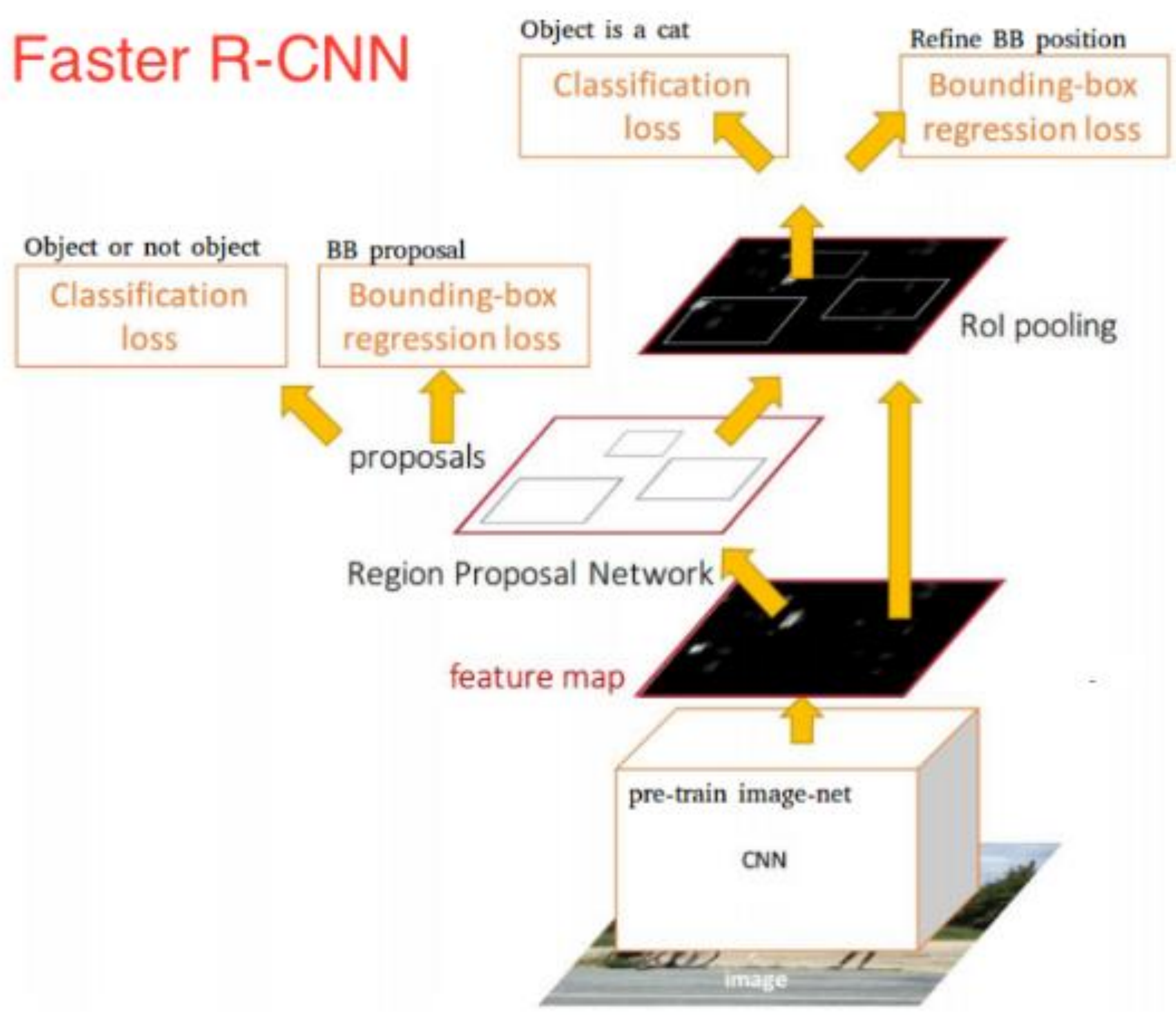
Faster R-CNN architecture

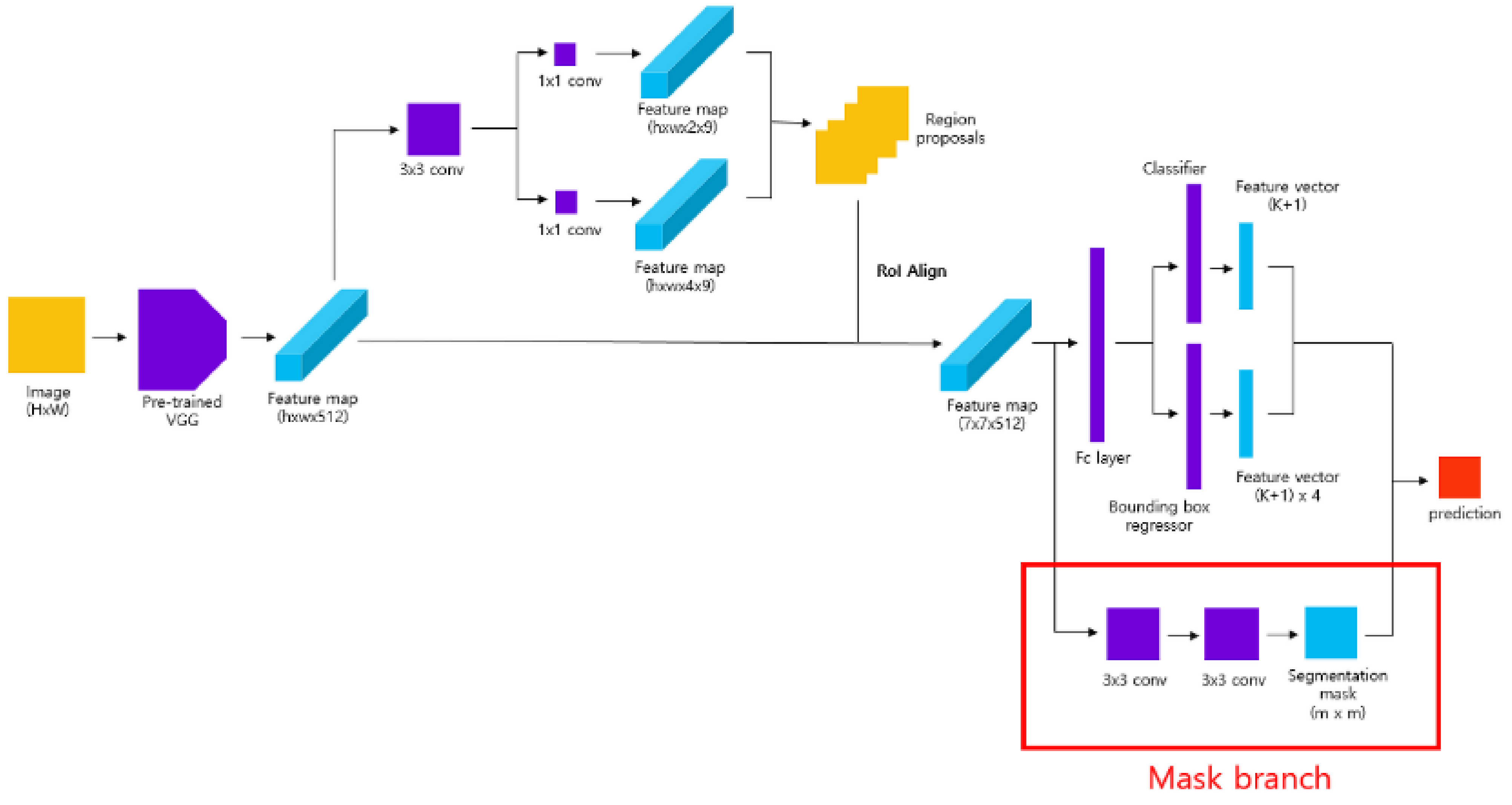
Main Idea



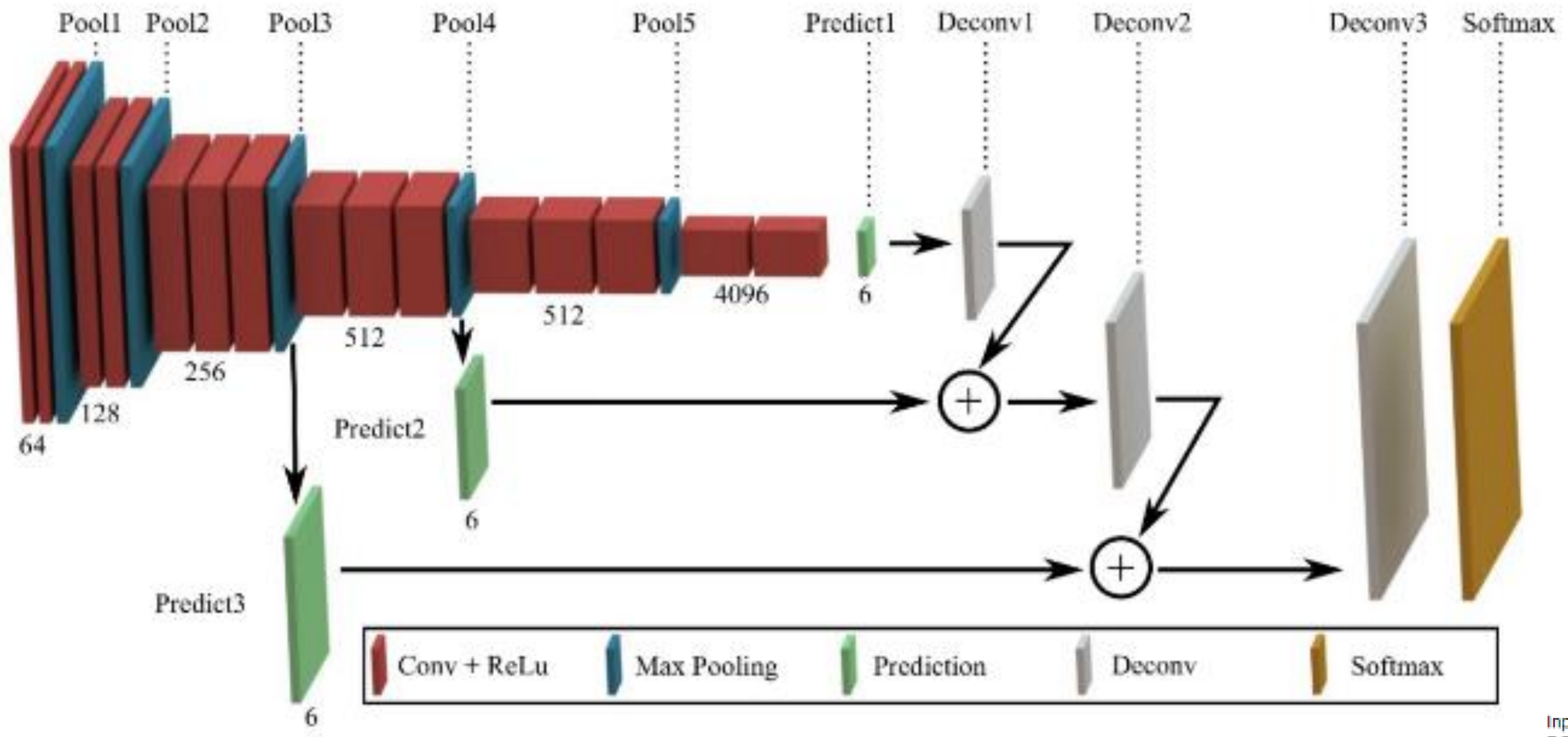
EWHA EURON

01 Mask branch

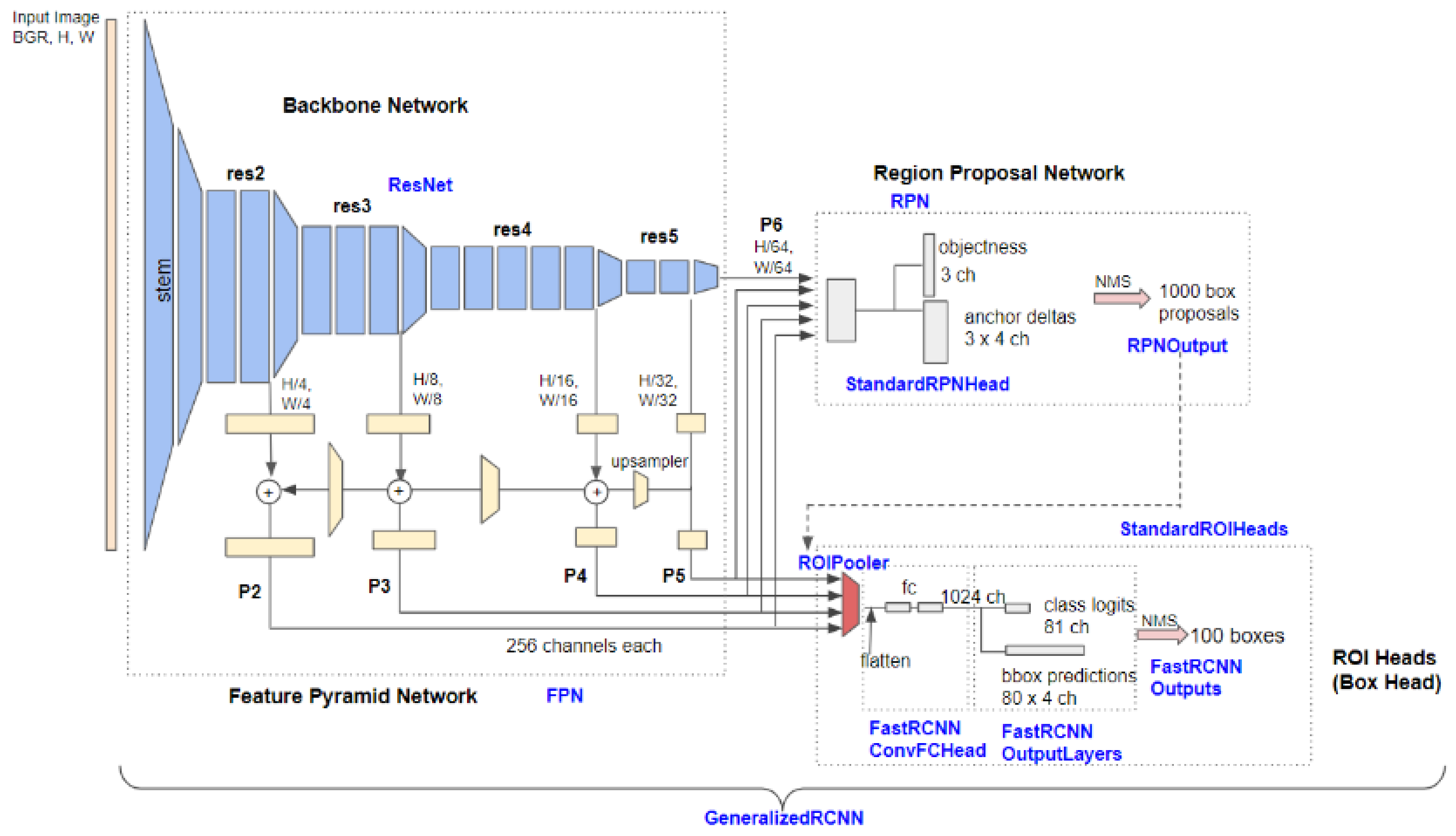




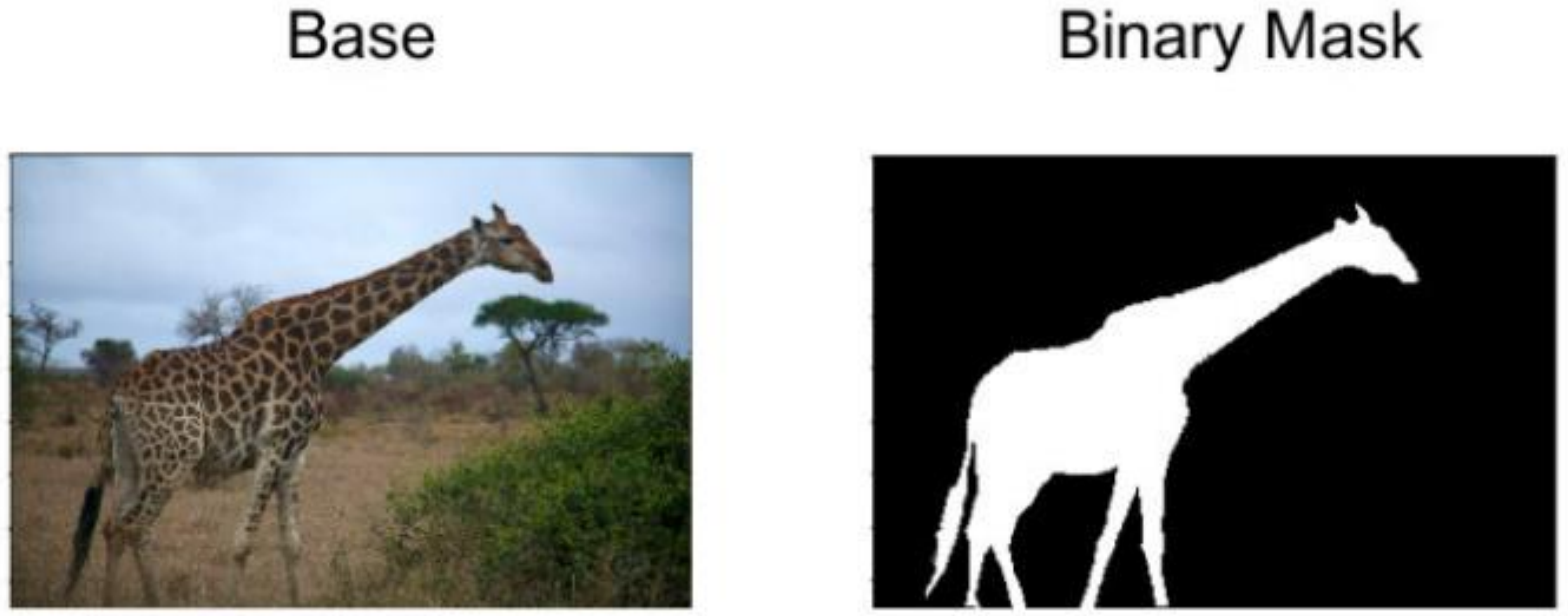
01 Mask branch



Original Skip Layer Architecture

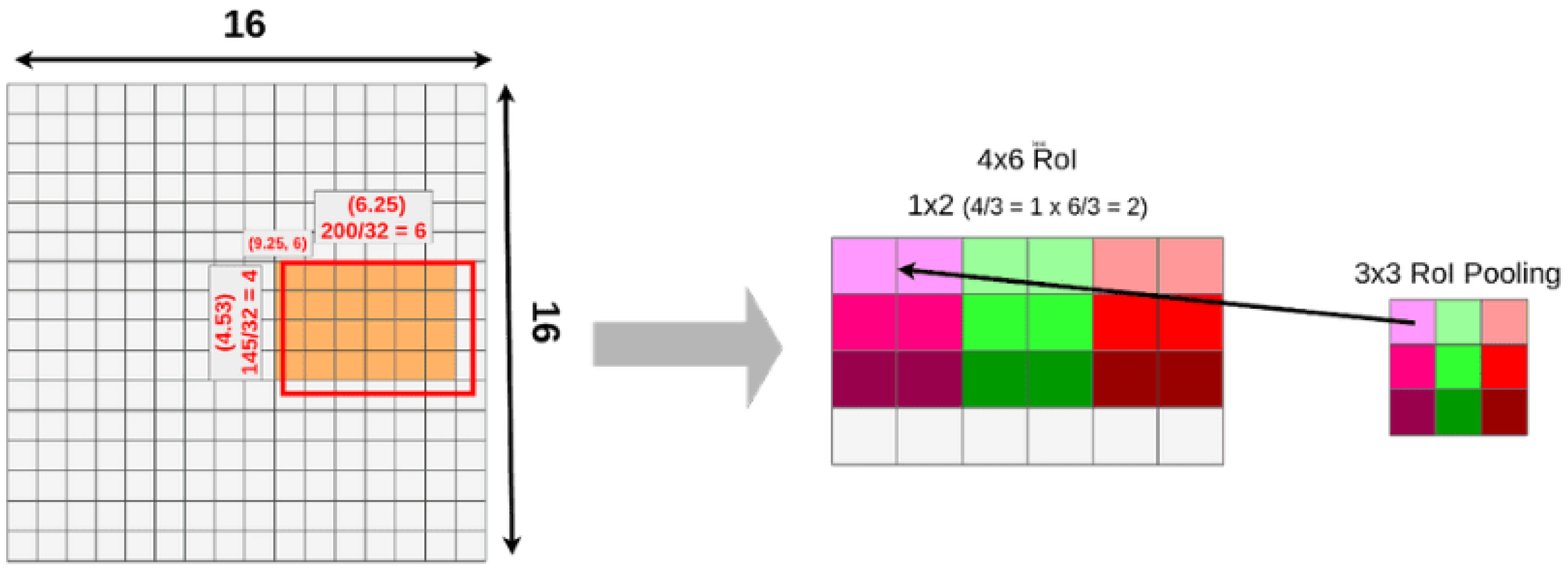


01 Mask branch



Binary mask

02 RoI Align



RoI pooling

02 RoI Align

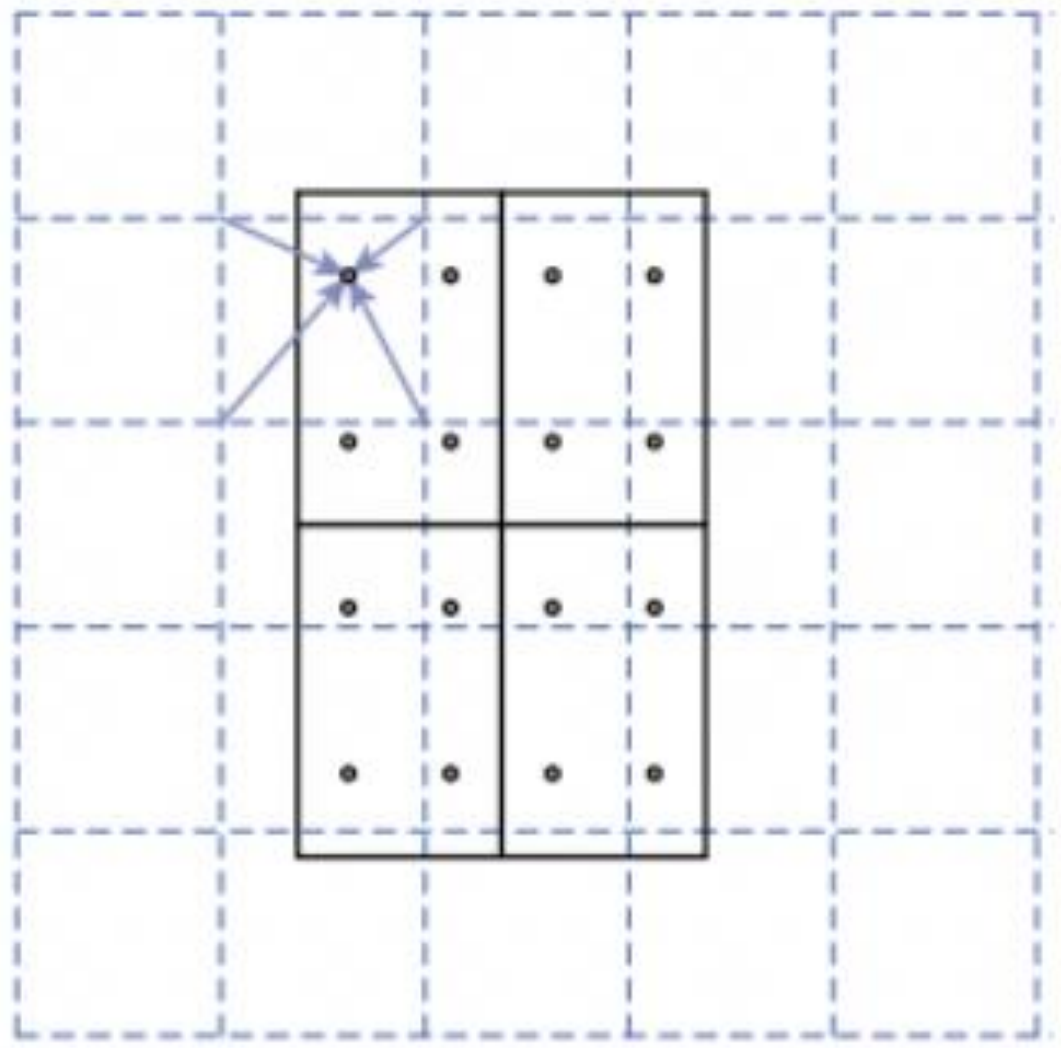
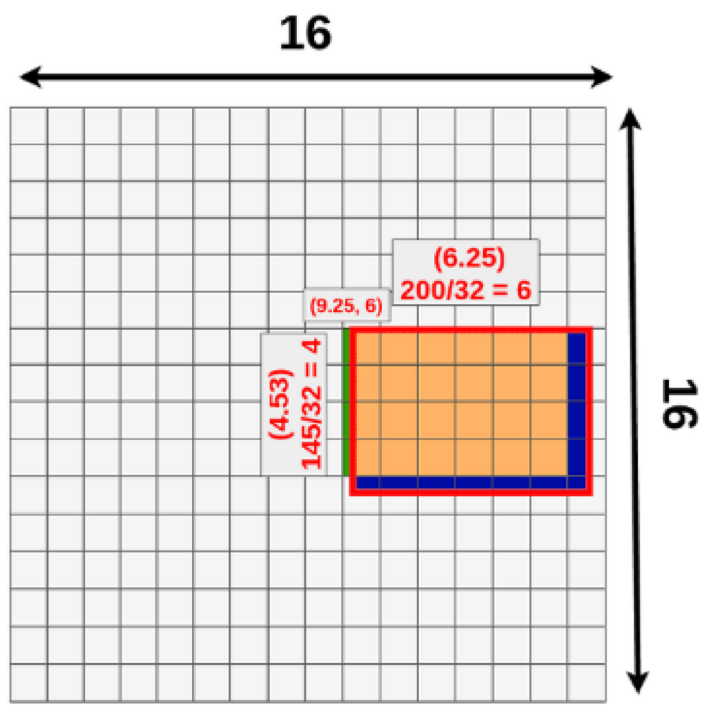


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

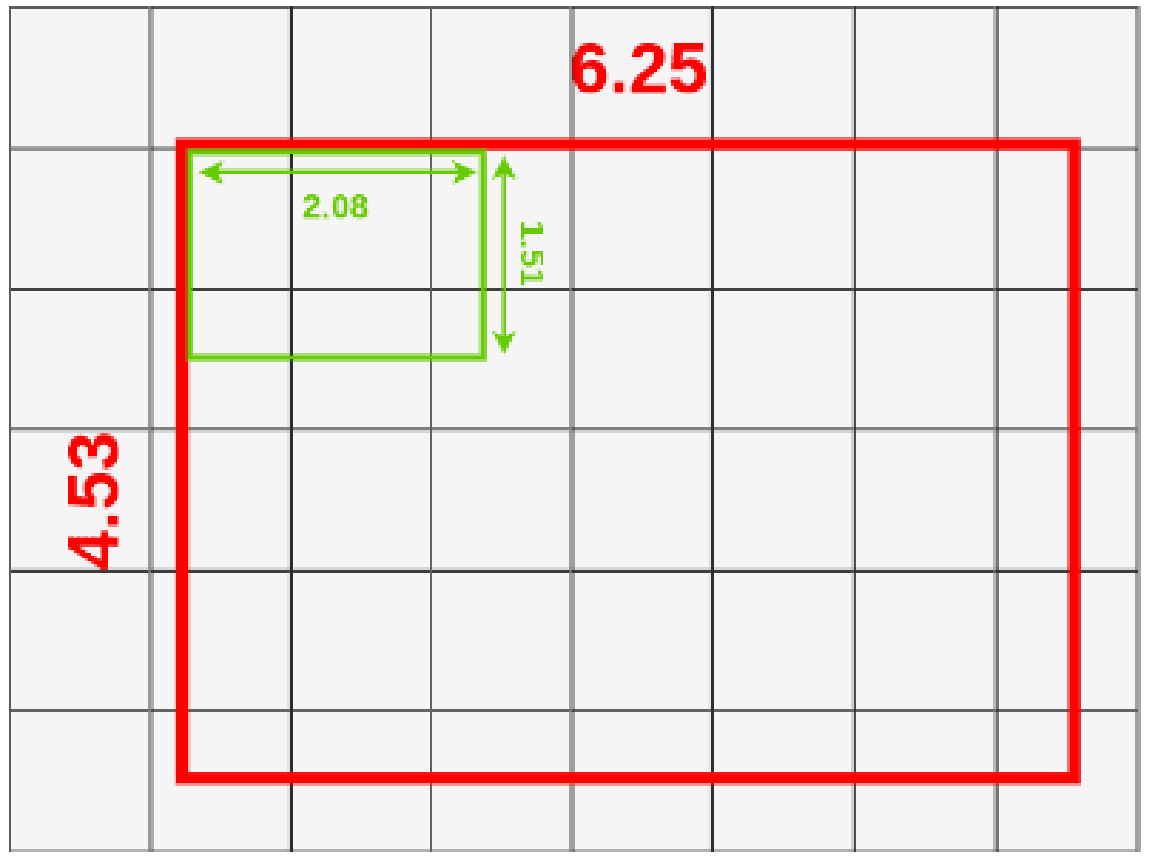
02 RoI Align

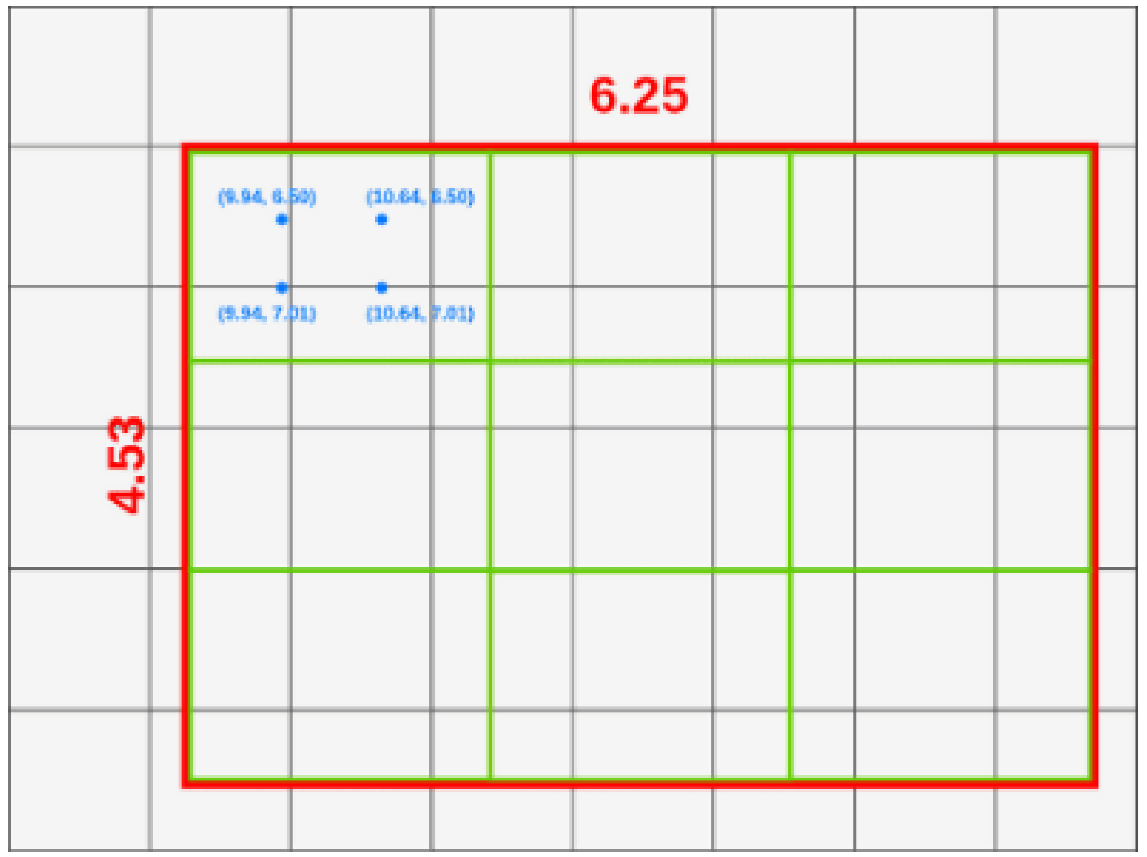


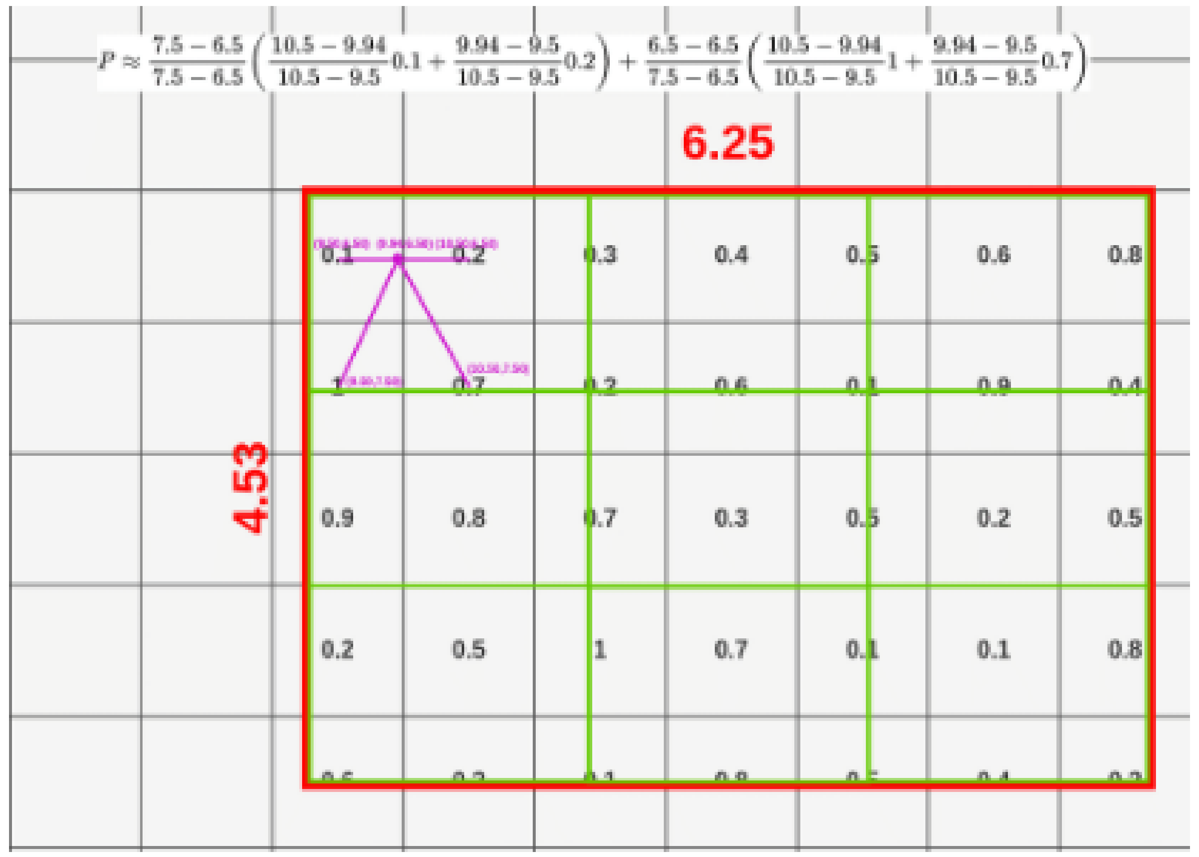
4x6 RoI (Lost Data)

0.1	0.2	0.3	0.4	0.5	0.6
1	0.7	0.2	0.6	0.1	0.9
0.9	0.8	0.7	0.3	0.5	0.2
0.2	0.5	1	0.7	0.1	0.1

Quantization losses

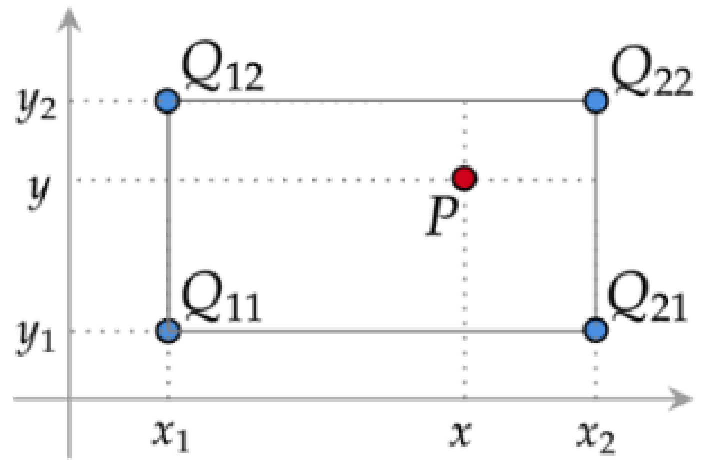






$$P \approx \frac{7.5-6.5}{7.5-6.5}\left(\frac{10.5-9.94}{10.5-9.5}0.1+\frac{9.94-9.5}{10.5-9.5}0.2\right)+\frac{6.5-6.5}{7.5-6.5}\left(\frac{10.5-9.94}{10.5-9.5}1+\frac{9.94-9.5}{10.5-9.5}0.7\right)$$

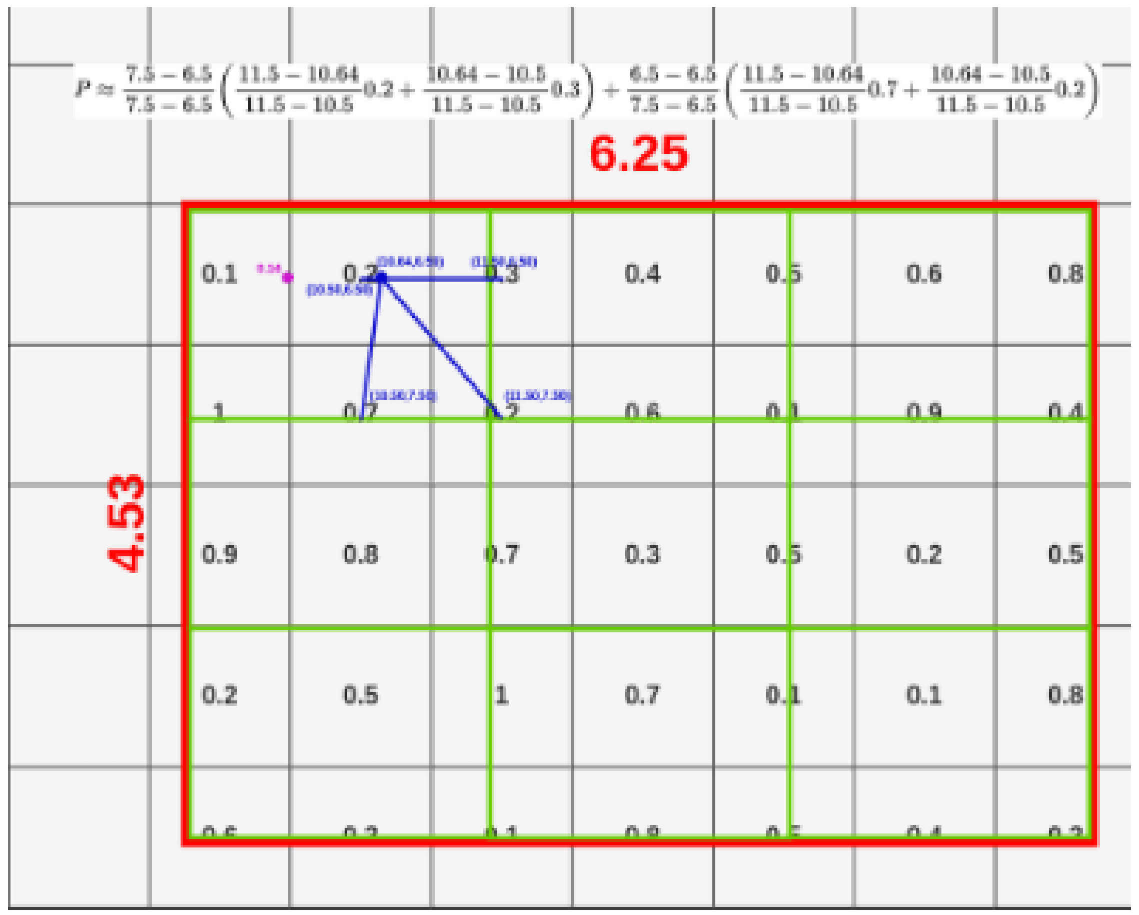
02 RoI Align



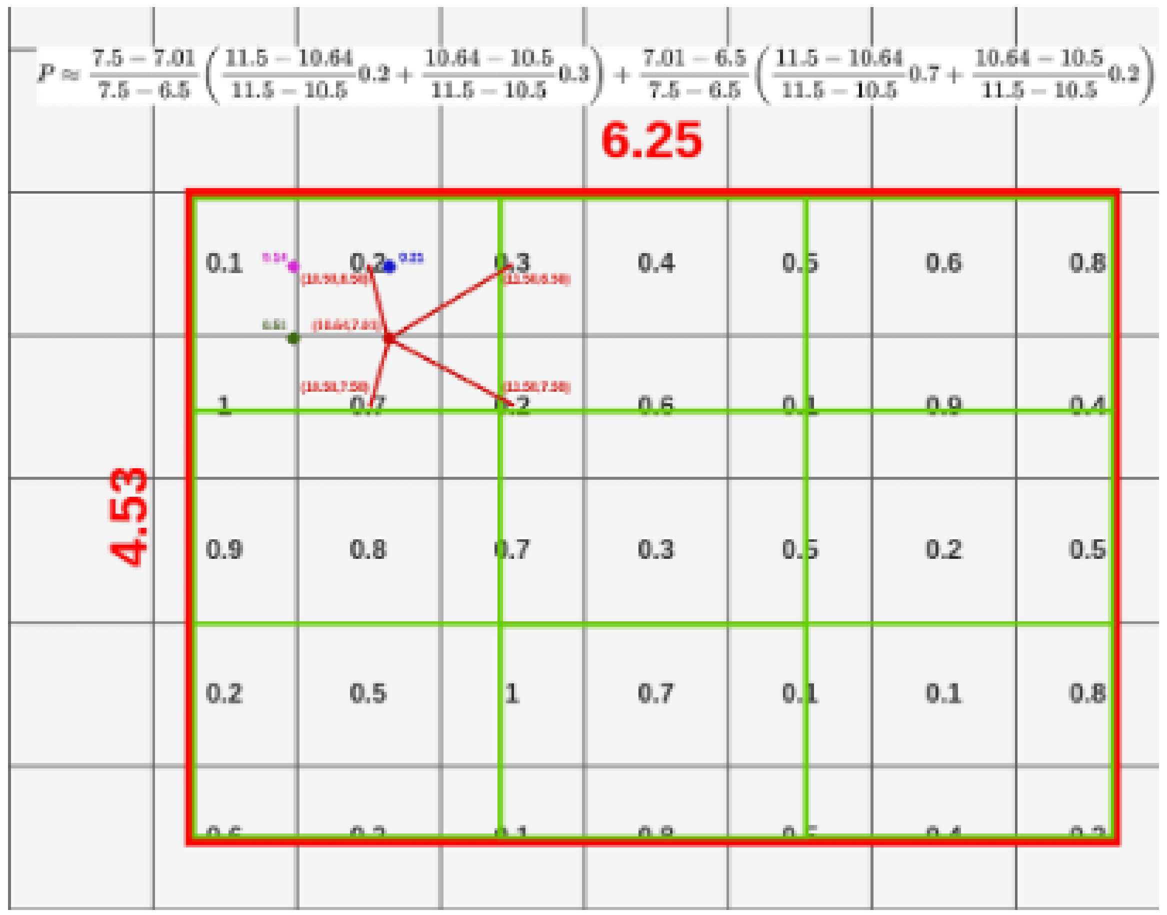
	x_1	x	x_2
y_1	Q_{11}		Q_{21}
y		P	
y_2	Q_{12}		Q_{22}

Bilinear interpolation

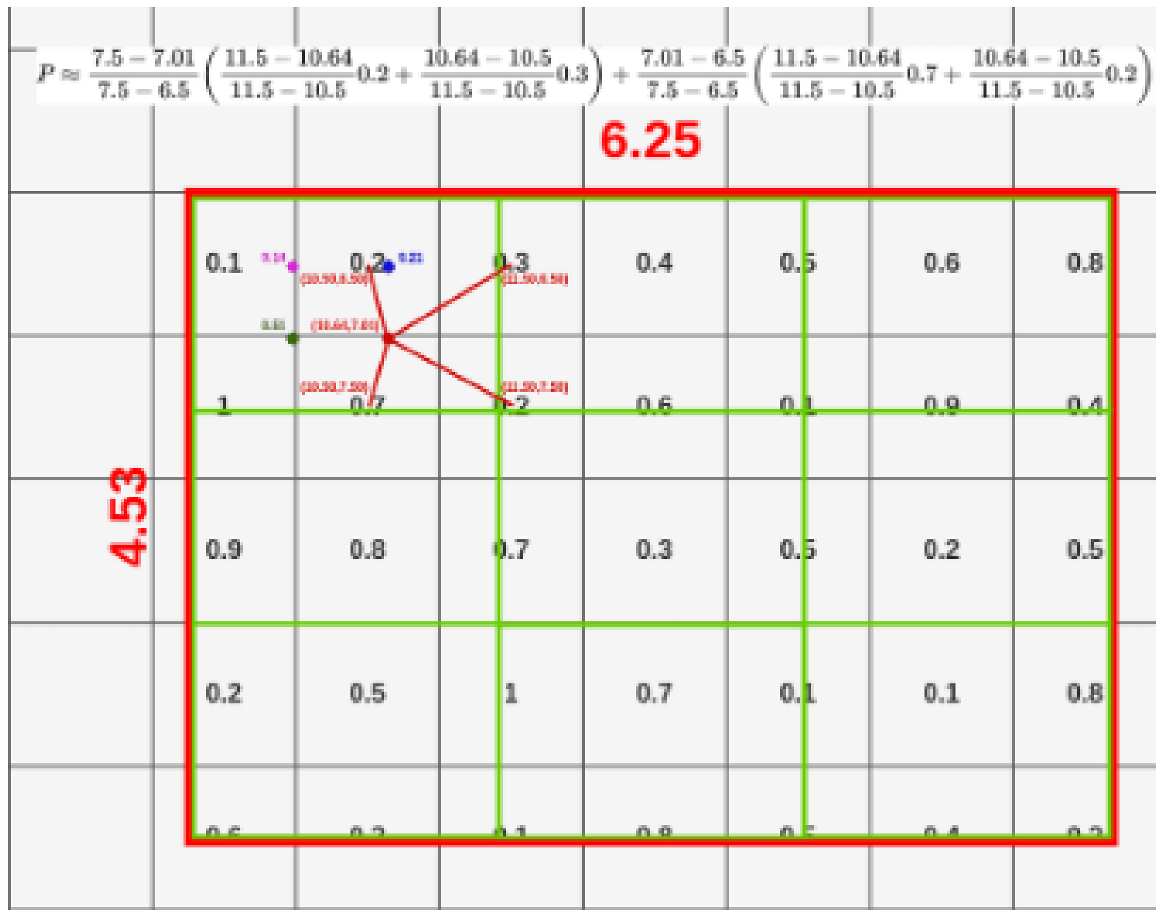
$$P \approx \frac{y_2 - y}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}Q_{11} + \frac{x - x_1}{x_2 - x_1}Q_{21}\right) + \frac{y - y_1}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}Q_{12} + \frac{x - x_1}{x_2 - x_1}Q_{22}\right)$$



$$P \approx \frac{7.5 - 6.5}{7.5 - 6.5}\left(\frac{11.5 - 10.64}{11.5 - 10.5}0.2 + \frac{10.64 - 10.5}{11.5 - 10.5}0.3\right) + \frac{6.5 - 6.5}{7.5 - 6.5}\left(\frac{11.5 - 10.64}{11.5 - 10.5}0.7 + \frac{10.64 - 10.5}{11.5 - 10.5}0.2\right)$$

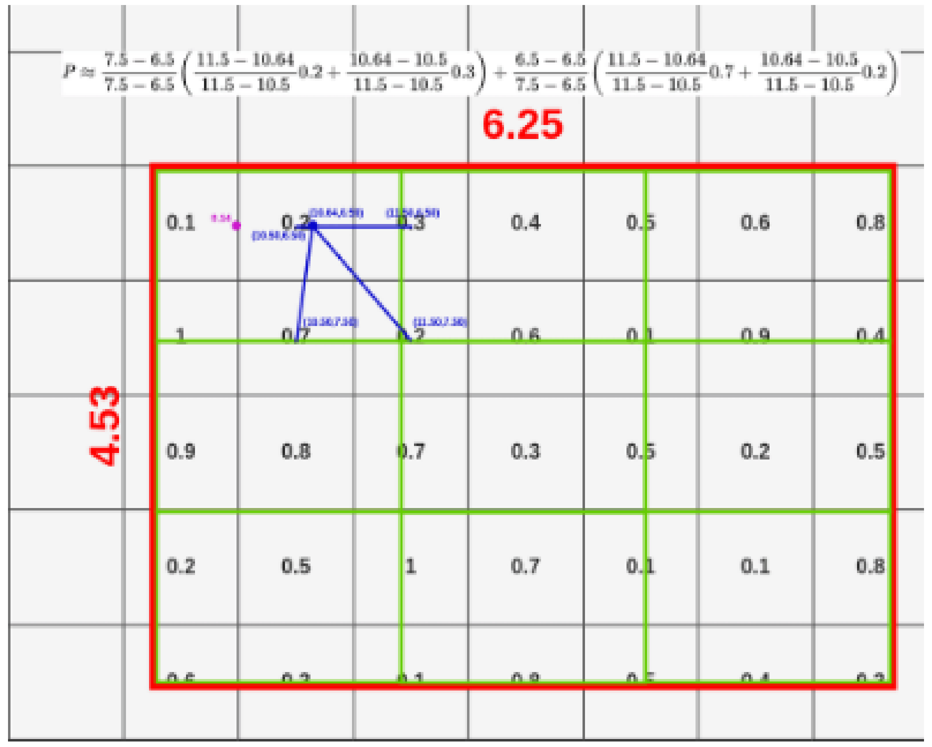


$$P \approx \frac{7.5 - 7.01}{7.5 - 6.5}\left(\frac{11.5 - 10.64}{11.5 - 10.5}0.2 + \frac{10.64 - 10.5}{11.5 - 10.5}0.3\right) + \frac{7.01 - 6.5}{7.5 - 6.5}\left(\frac{11.5 - 10.64}{11.5 - 10.5}0.7 + \frac{10.64 - 10.5}{11.5 - 10.5}0.2\right)$$

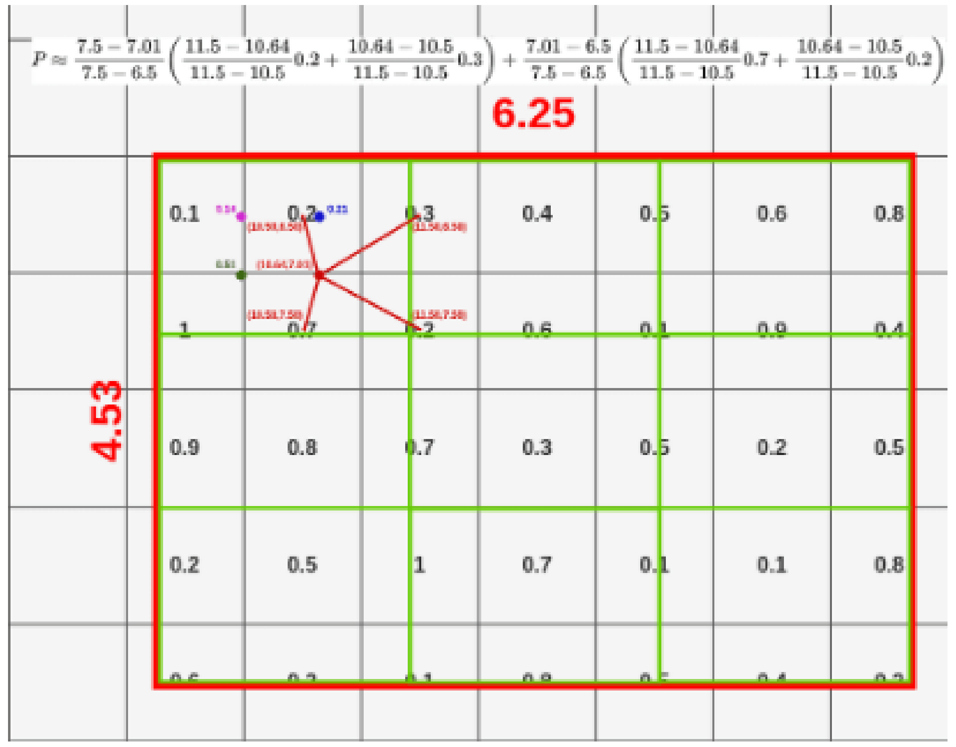


$$P \approx \frac{7.5 - 7.01}{7.5 - 6.5}\left(\frac{11.5 - 10.64}{11.5 - 10.5}0.2 + \frac{10.64 - 10.5}{11.5 - 10.5}0.3\right) + \frac{7.01 - 6.5}{7.5 - 6.5}\left(\frac{11.5 - 10.64}{11.5 - 10.5}0.7 + \frac{10.64 - 10.5}{11.5 - 10.5}0.2\right)$$

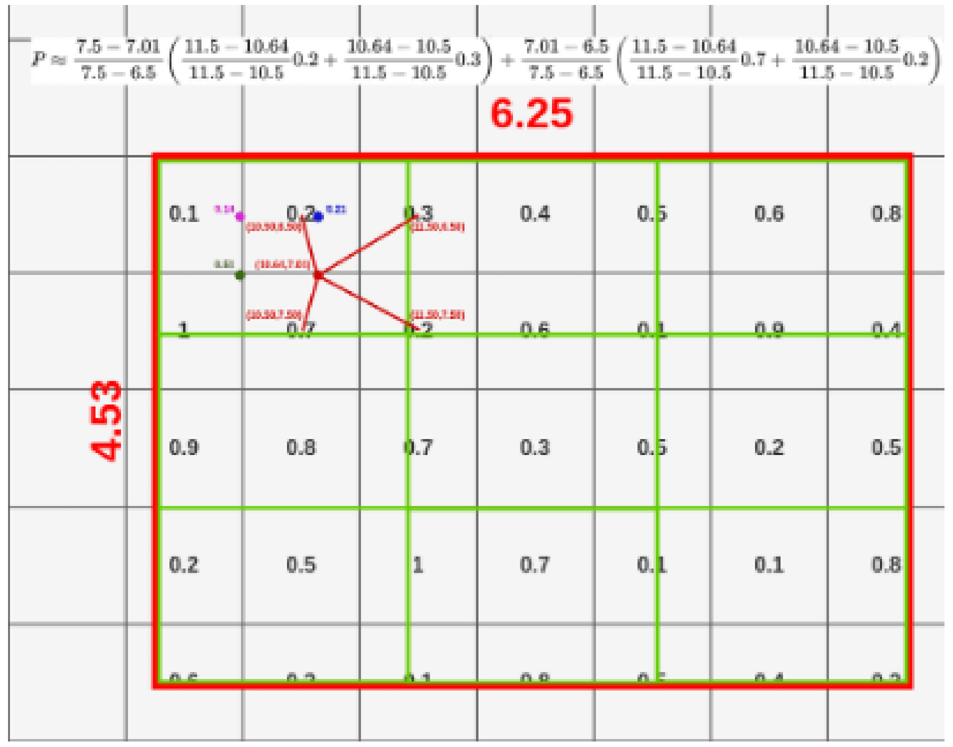
02 RoI Align



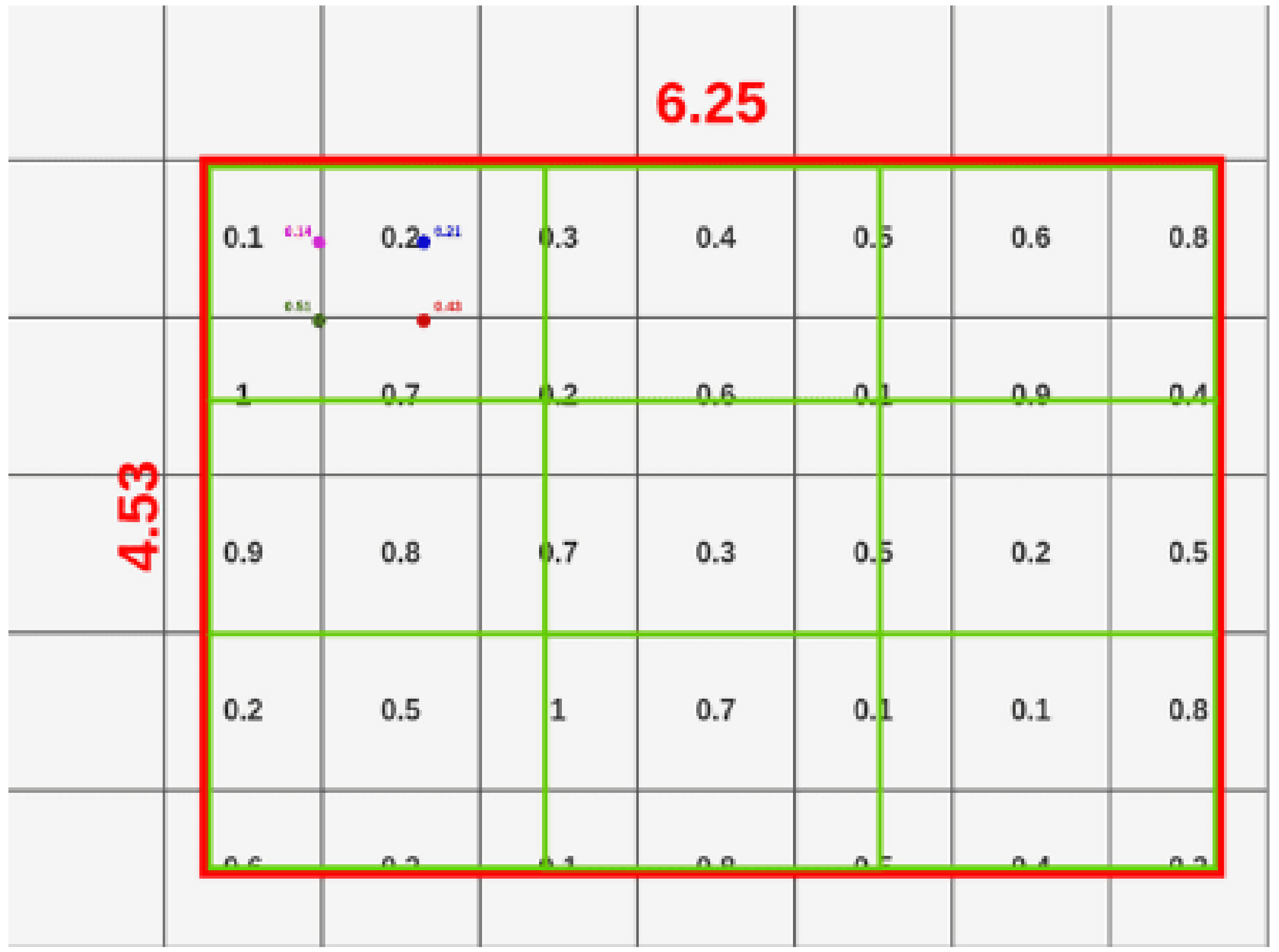
$P \approx \frac{7.5-6.5}{7.5-6.5}\left(\frac{11.5-10.64}{11.5-10.5}0.2+\frac{10.64-10.5}{11.5-10.5}0.3\right)+\frac{6.5-6.5}{7.5-6.5}\left(\frac{11.5-10.64}{11.5-10.5}0.7+\frac{10.64-10.5}{11.5-10.5}0.2\right)$
6.25
4.53



$P \approx \frac{7.5-7.01}{7.5-6.5}\left(\frac{11.5-10.64}{11.5-10.5}0.2+\frac{10.64-10.5}{11.5-10.5}0.3\right)+\frac{7.01-6.5}{7.5-6.5}\left(\frac{11.5-10.64}{11.5-10.5}0.7+\frac{10.64-10.5}{11.5-10.5}0.2\right)$
6.25
4.53



$P \approx \frac{7.5-7.01}{7.5-6.5}\left(\frac{11.5-10.64}{11.5-10.5}0.2+\frac{10.64-10.5}{11.5-10.5}0.3\right)+\frac{7.01-6.5}{7.5-6.5}\left(\frac{11.5-10.64}{11.5-10.5}0.7+\frac{10.64-10.5}{11.5-10.5}0.2\right)$
6.25
4.53



1x1 = MAX(0.14, 0.21, 0.51, 0.43) = 0.51

3x3 RoIAlign

0.51		

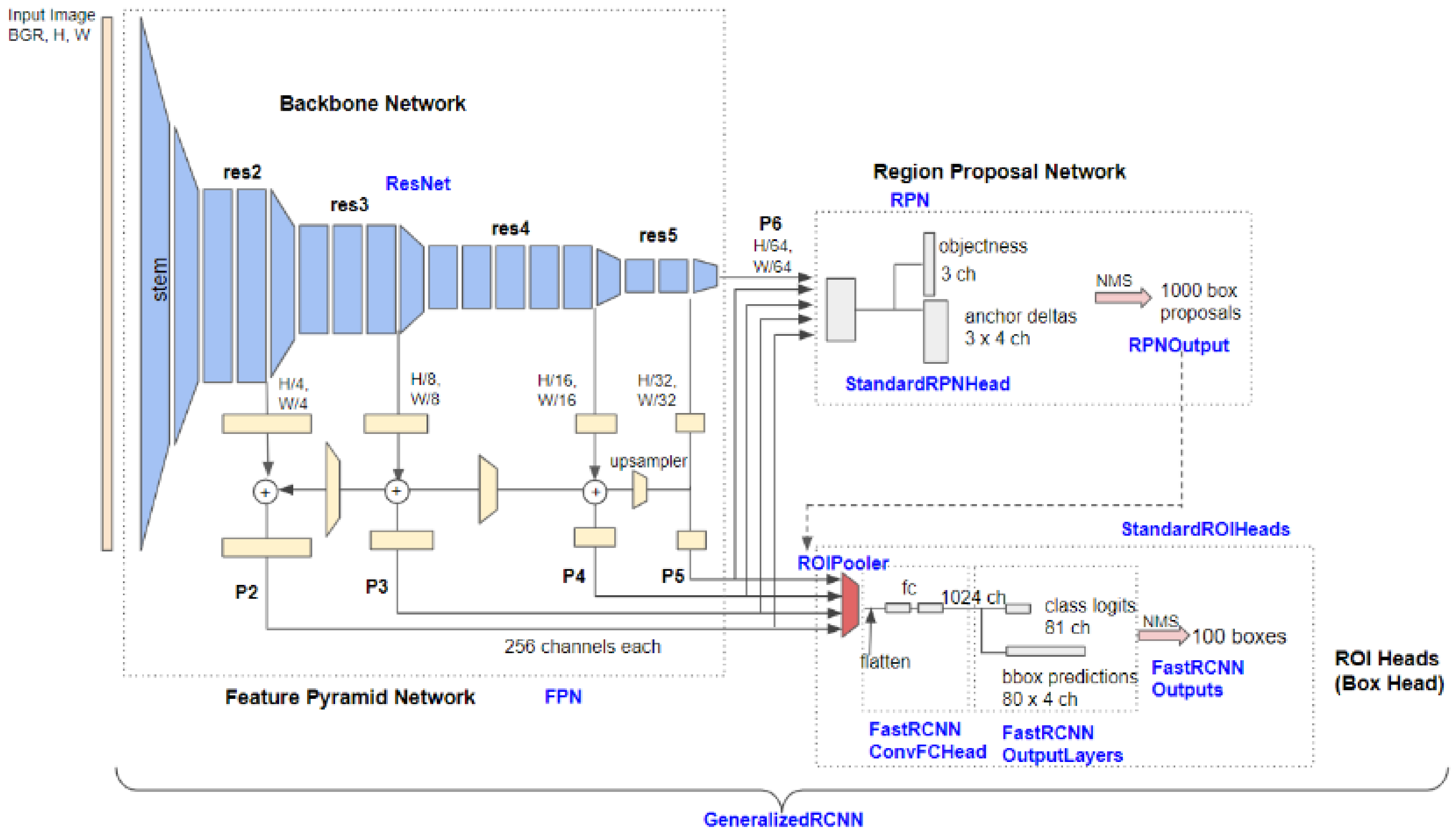
03 Besides that

1. Loss Function

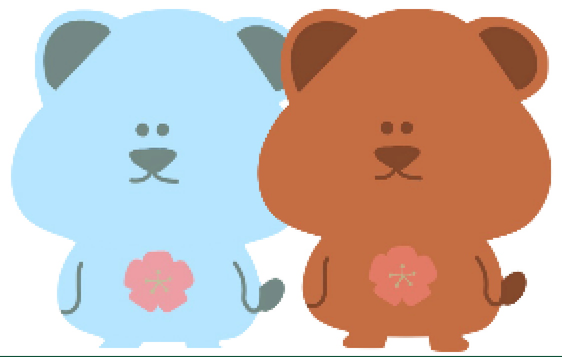
$$L = L_{cls} + L_{box} + L_{mask}$$

2. Backbone network

ResNet-FPN



Training Mask R-CNN



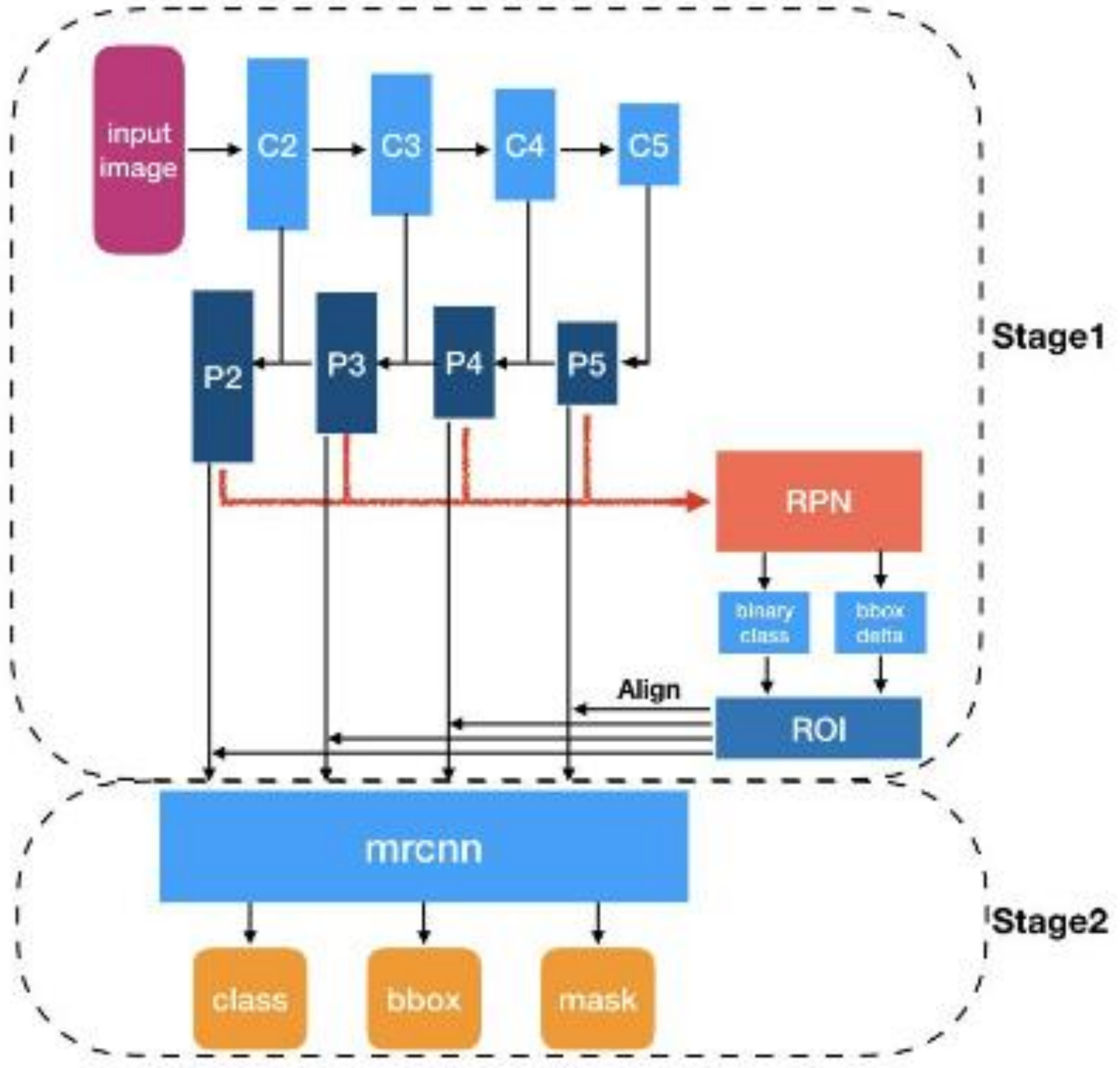
01 Training process

01 Input image Pre-processing

- Input: image
- Process: image pre-processing
- Output: resized image

02 Feature pyramid by backbone network

- Input: resized image
- Process: constructing feature pyramid
- Output: feature pyramid {P2, P3, P4, P5, P6}



ResNet-FPN

01 Training process

03 Region proposal by RPN

- Input: feature pyramid {P2, P3, P4, P5, P6}
- Process: Region proposal
- Output: Region proposals with objectness score and bbox regressor per feature pyramid {P2, P3, P4, P5, P6}

04 Select best RoI by Proposal layer

- Input: Region proposals
- Process: selecting top-N RoIs
- Output: top-N RoIs

1. objectness score가 높은 top-n개의 anchor 선정
2. Bbox regressor에 따라 anchor box 크기 조정
3. 이미지 경계를 벗어나는 anchor box 제거
4. Threshold = 0.7로 지정해 Non maximum suppression을 수행
5. Feature pyramid level 별로 수행 이전까지 얻은 모든 feature pyramid level의 anchor box에 대한 정보를 결합
6. 마지막으로 결합된 모든 anchor box에 대해 objectness score에 따라 top-N개의 anchor box를 선정

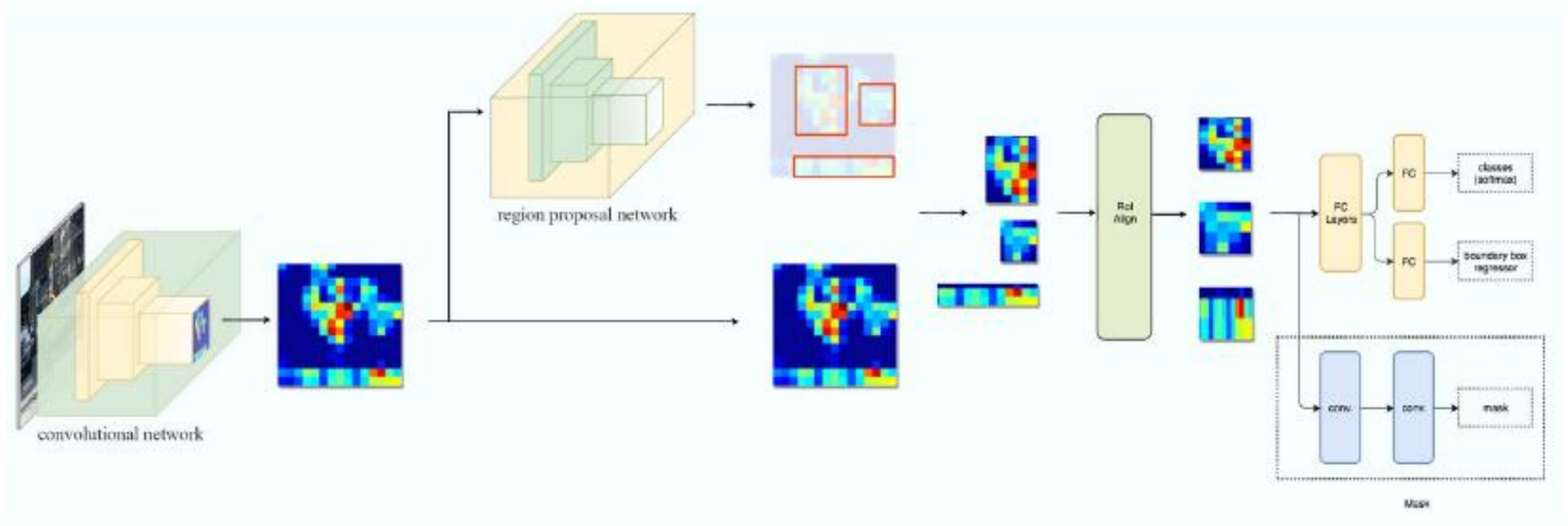
01 Training process

05 Feature map by RoI Align layer

$$k = [k_0 + log_2(\sqrt{wh}/224)]$$

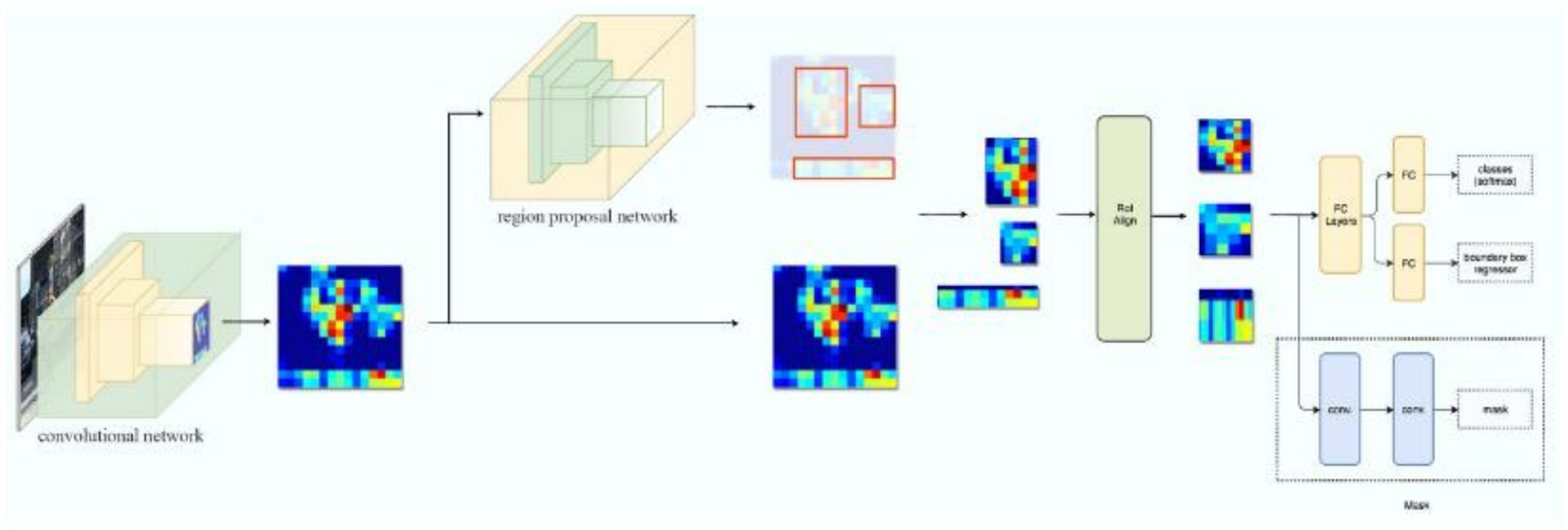
- Input: feature pyramid and RoIs
- Process: RoI Align
- Output: 7x7 sized feature map

06 Classification and Bounding box regression by Fast R-CNN



01 Training process

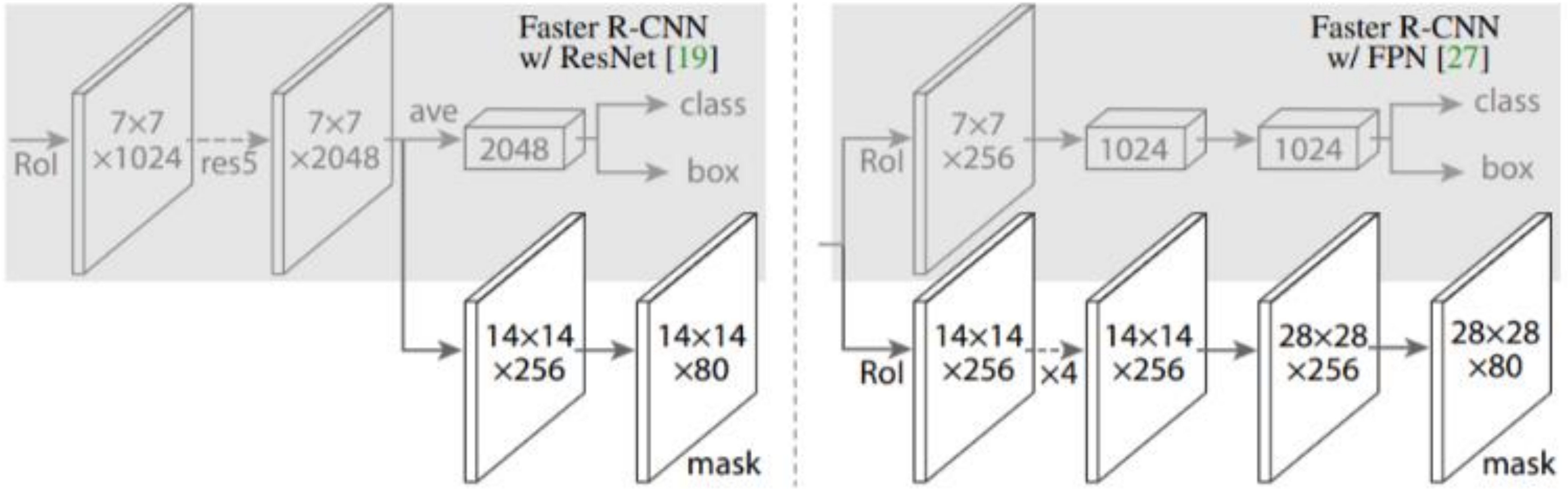
06 Classification and Bounding box regression by Fast R-CNN



- Input: 7x7 sized feature map
- Process: Classification by classification branch, bbox regressor by bbox regression branch
- Output: class scores and bbox regressors

01 Training process

07 Mask segment by Mask branch



- Input: 7x7 sized feature map
- Process: mask segment by mask branch
- Output: 14x14 sized feature map

01 Training process

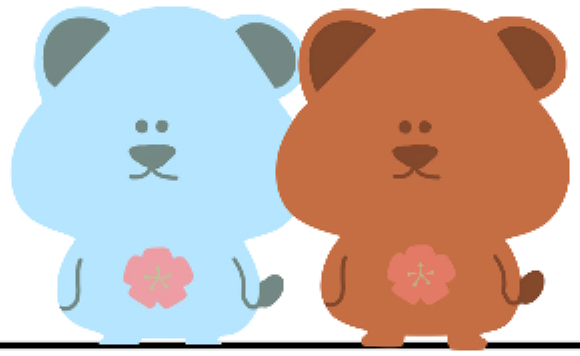
08 Post-processing of masks

- Input: 14x14 sized feature map
- Process: rescale and apply mask threshold
- Output: mask segment

09 Train Mask R-CNN by multi-task loss

- Mask R-CNN 네트워크를 multi-task loss function을 사용해 학습시킨다.

THANK YOU



EWHA EURON