AiDA Lab.
Artificial Intelligence & Data Analysis Laboratory

2021 인공지능 소수전공

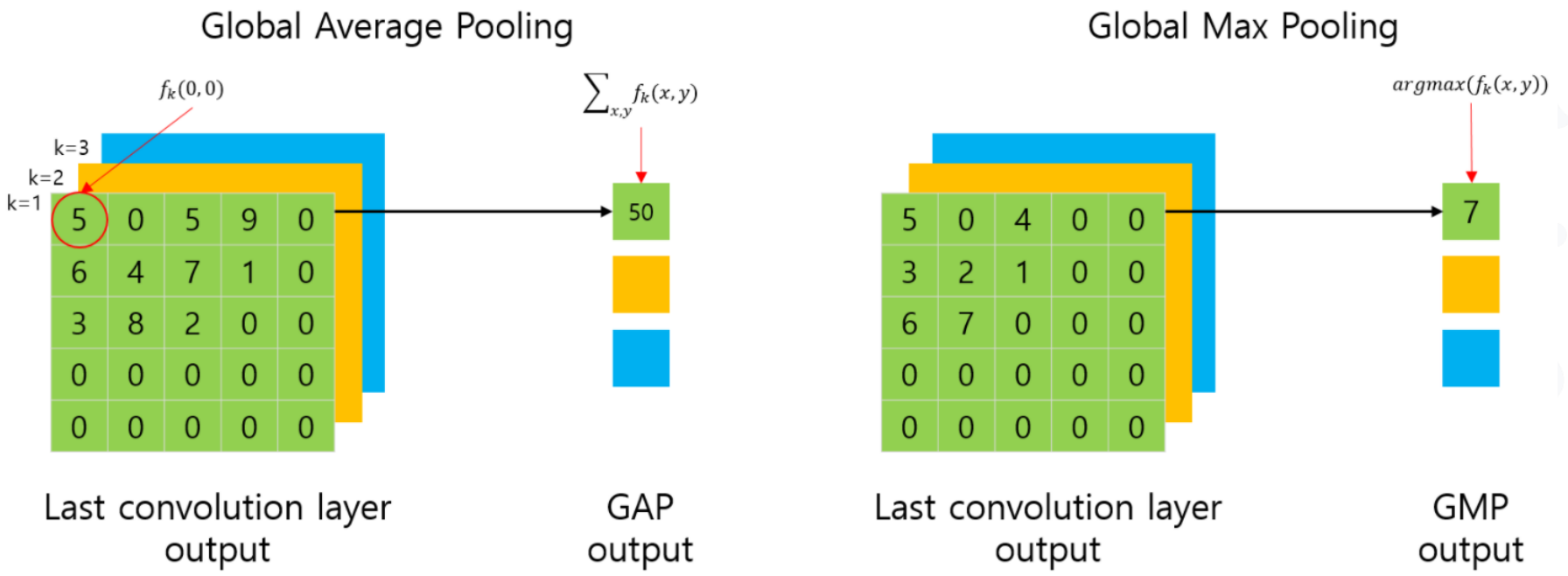
72차시: CAM & Grad-CAM

2021.08.09 18:30~19:15

Seokhwan Yang

GAP (Global Average Pooling)

- **GAP Layer**
  - **각각의 Feature Map의 값들을 평균을 취한 것**
  - **즉, Feature Map의 크기와 관계없이 channel이 k개라면 k개의 평균 값을 얻을 수 있음**



GAP (Global Average Pooling)

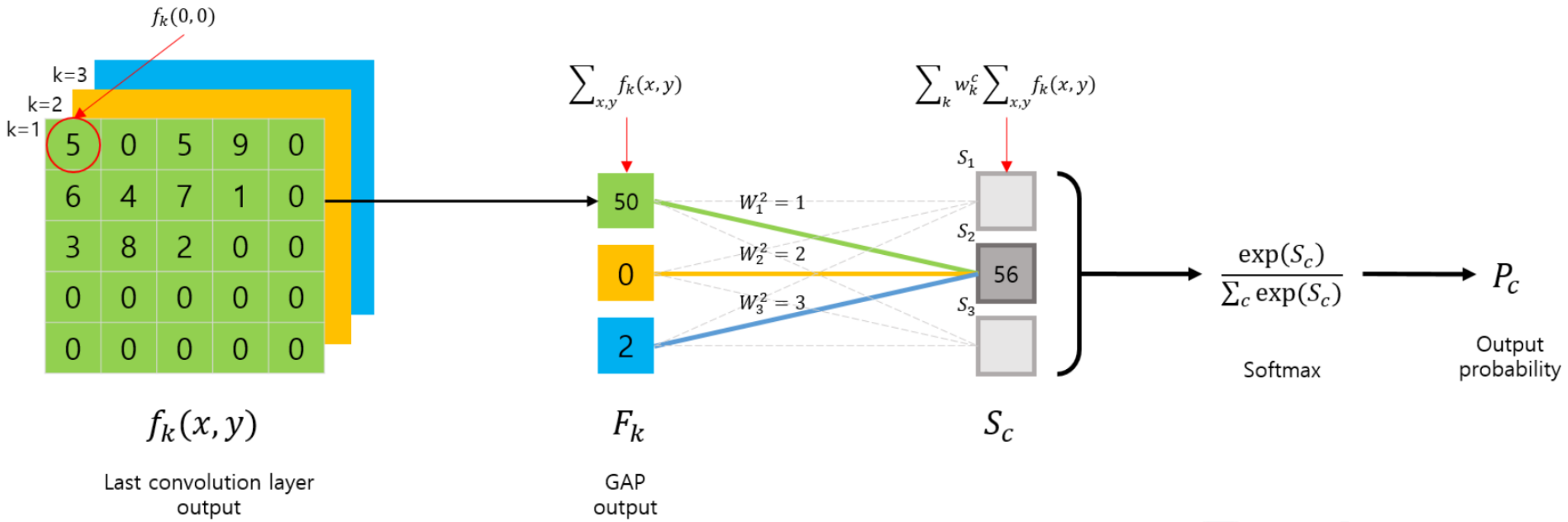
- GAP는 Fully Connected(FC) Layer와 달리 연산이 필요한 파라미터 수를 크게 줄일 수 있음

  → Regularization과 유사한 동작을 통해 overfitting을 방지할 수 있음

- FC layer는 Convolution layer에서 유지하던 위치정보가 손실되는 반면에, GAP layer는 위치정보를 담고 있기 때문에 localization에 유리함

CAM (Class Activation Map)

- **CAM**
  - **GAP layer를 사용하여 Classification을 진행하는 모델 구조**

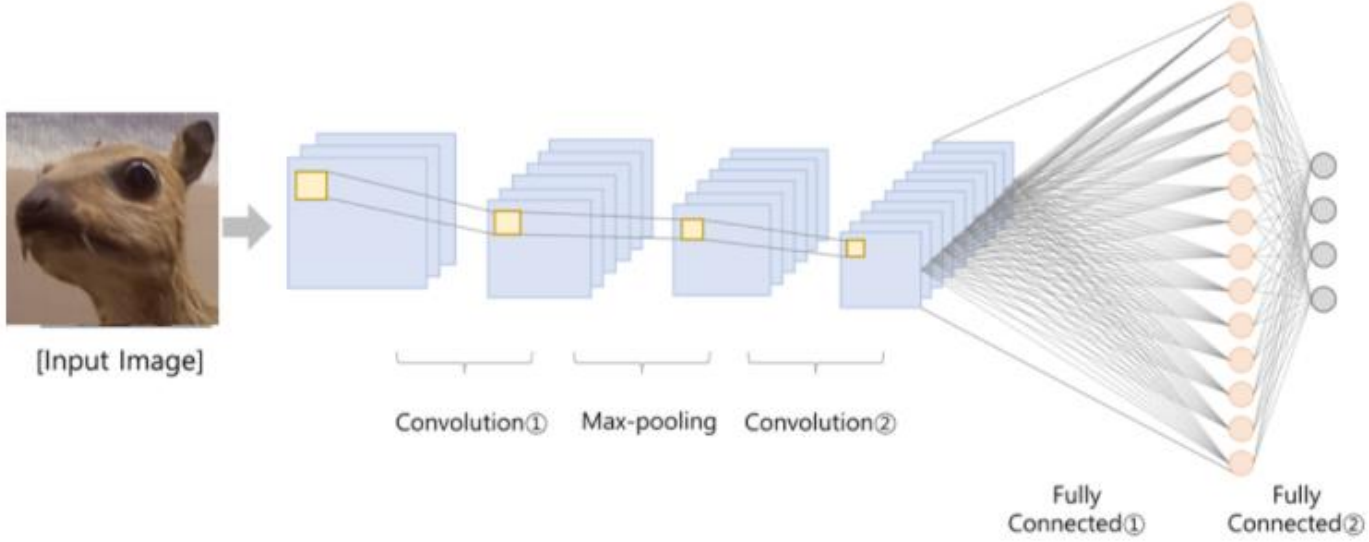


CAM (Class Activation Map)

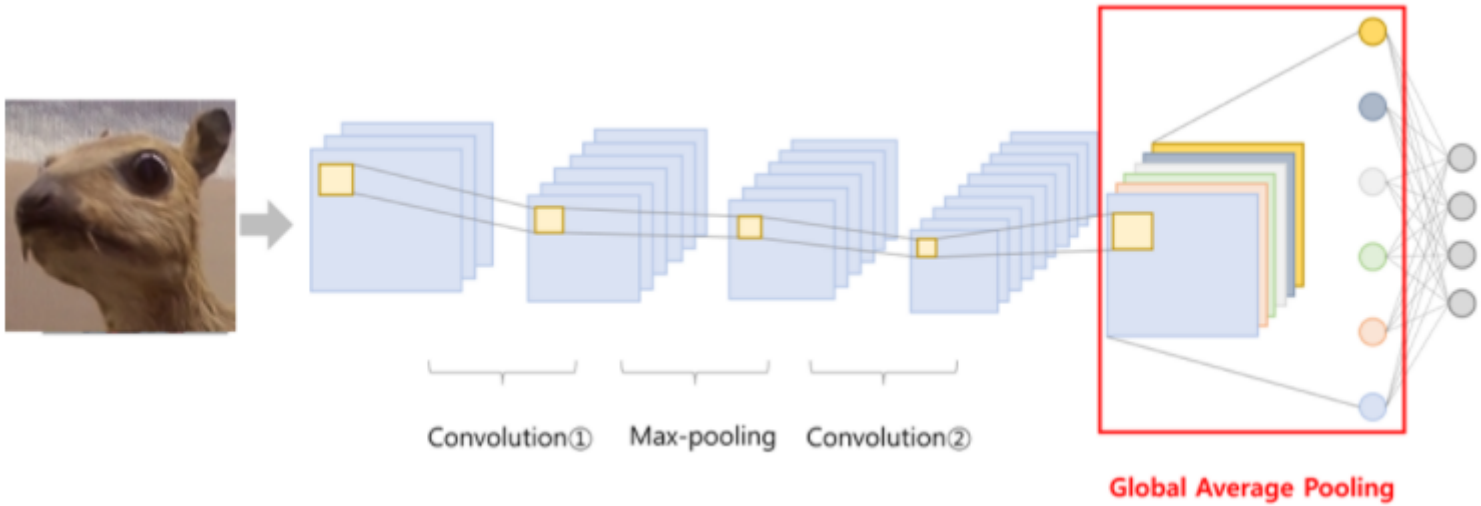
- CNN Layer를 거쳐 마지막에 k개의 Feature Map을 얻은 후,
- GAP를 적용하여 총 k개의 output을 얻을 수 있음
- 이 값을 선형결합한 후 여기에 softmax를 적용하여 최종 output probability가 출력됨

CAM (Class Activation Map)

- 일반적인 CNN

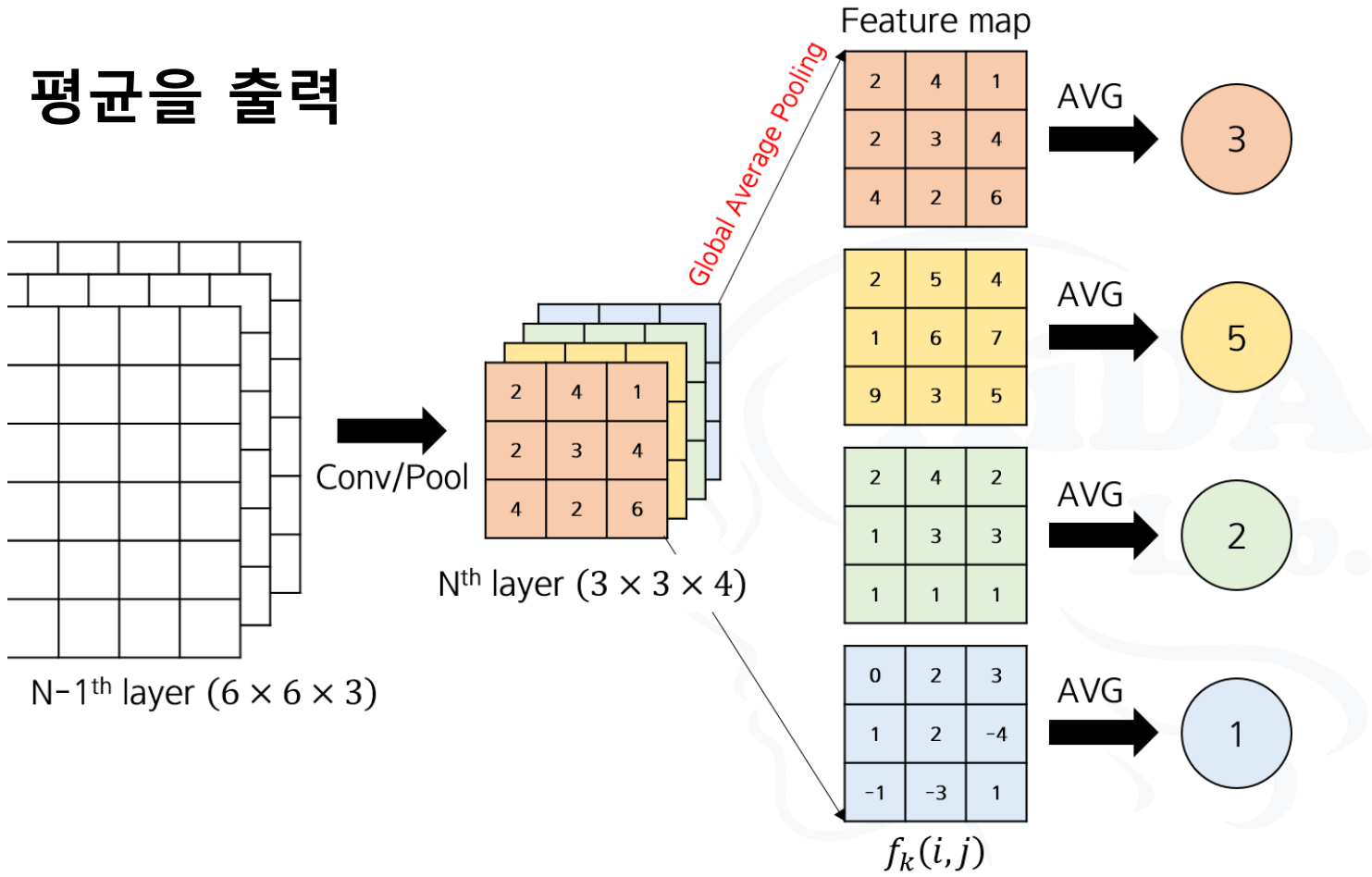


- Conv Layer 이후를 GAP로 교체



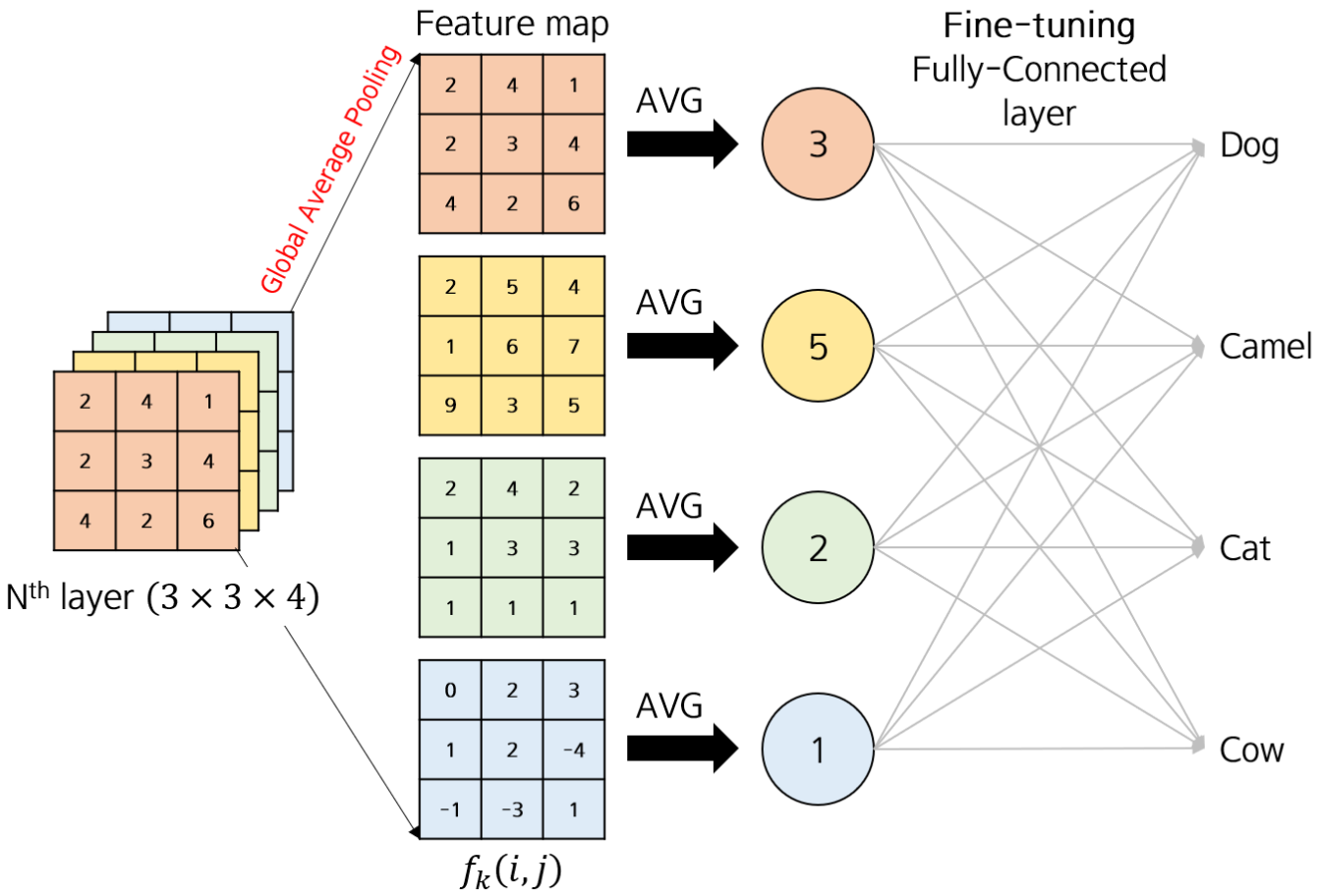
CAM (Class Activation Map)

- **GAP 적용**
  - **입력 이미지의 모든 값의 평균을 출력**



CAM (Class Activation Map)

- **Fine Tuning**



$$L^c_{CAM}(i,j) = \sum_k w^c_k f_k(i,j)$$

$L^c_{CAM}(i,j)$: Class c 에 대한 CAM

- $f_k(i,j)$: k번째 feature image (i, j는 x, y 축 좌표를 의미함)
- w^c_k: k번째 feature image $f_k(i,j)$에서 class c로 가는 weight

CAM (Class Activation Map)

$F_k = \sum_{x,y} f_k(x, y)$: k번째 Feature map의 GAP

$S_c = \sum_k w_k^c F_k$: softmax의 input

w_k^c: class c에 대한 F_k의 중요도

$P_c = \frac{\exp(S_c)}{\sum_c \exp(S_c)}$: class cc에 대한 softmax output

바이어스는 분류 성능에 영향을 미치지 않는다고 가정
→ bias=0 설정

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

Class c에 대한 Activation Map을 다음과 같이 Mc로 정의

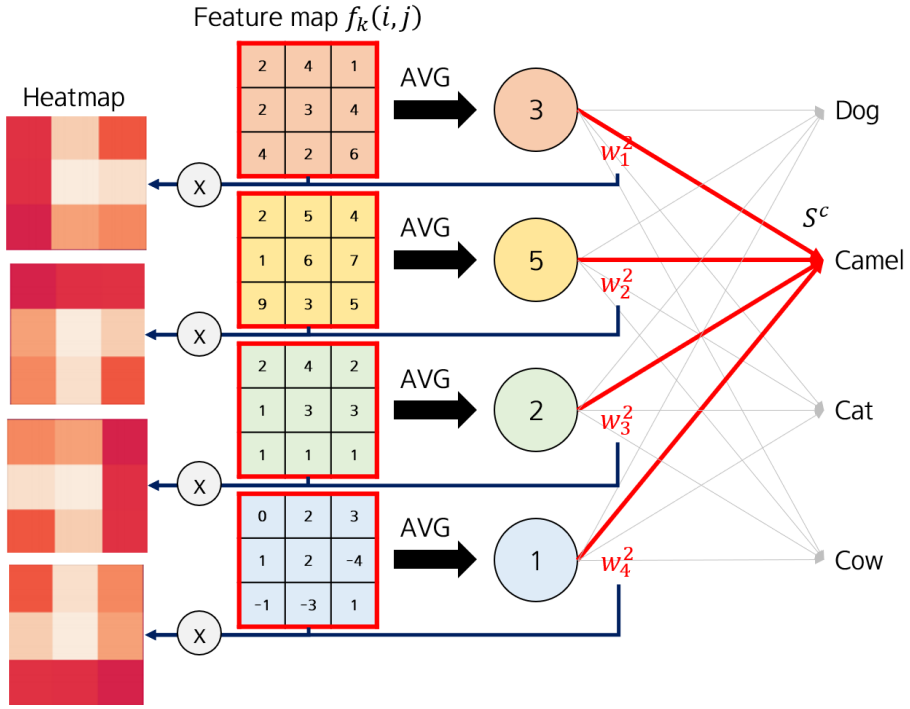
$$M_c(x, y) = \sum_k w_k^c f_k(x, y)$$

즉, $S_c = \sum_{x,y} M_c(x, y)$로 쓸 수 있고,

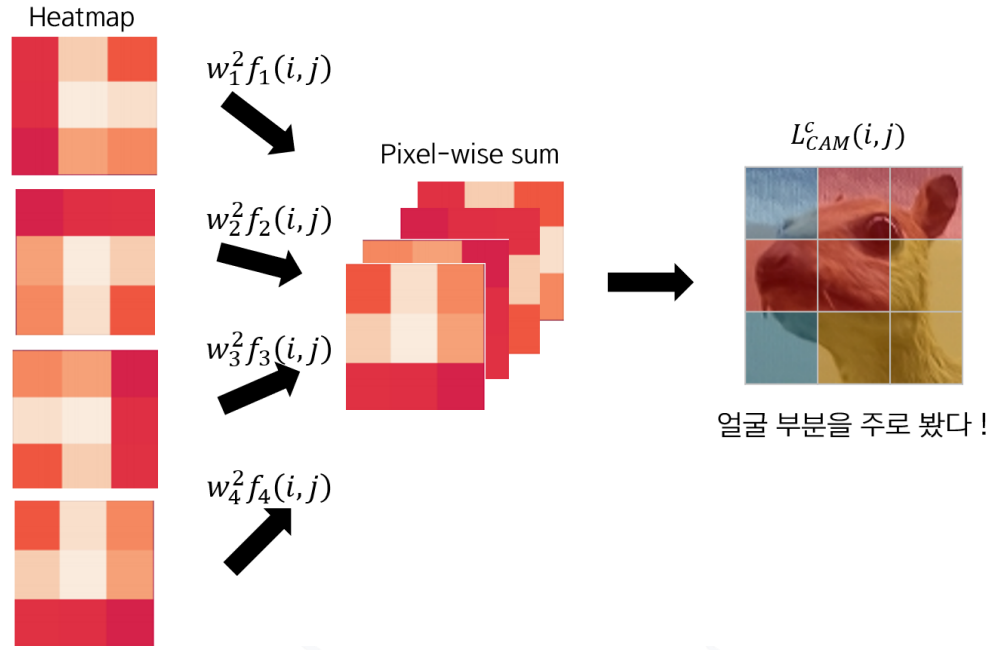
$M_c(x, y)$는 class c에 (x, y) 지점이

미치는 영향을 나타낸다.

CAM (Class Activation Map)



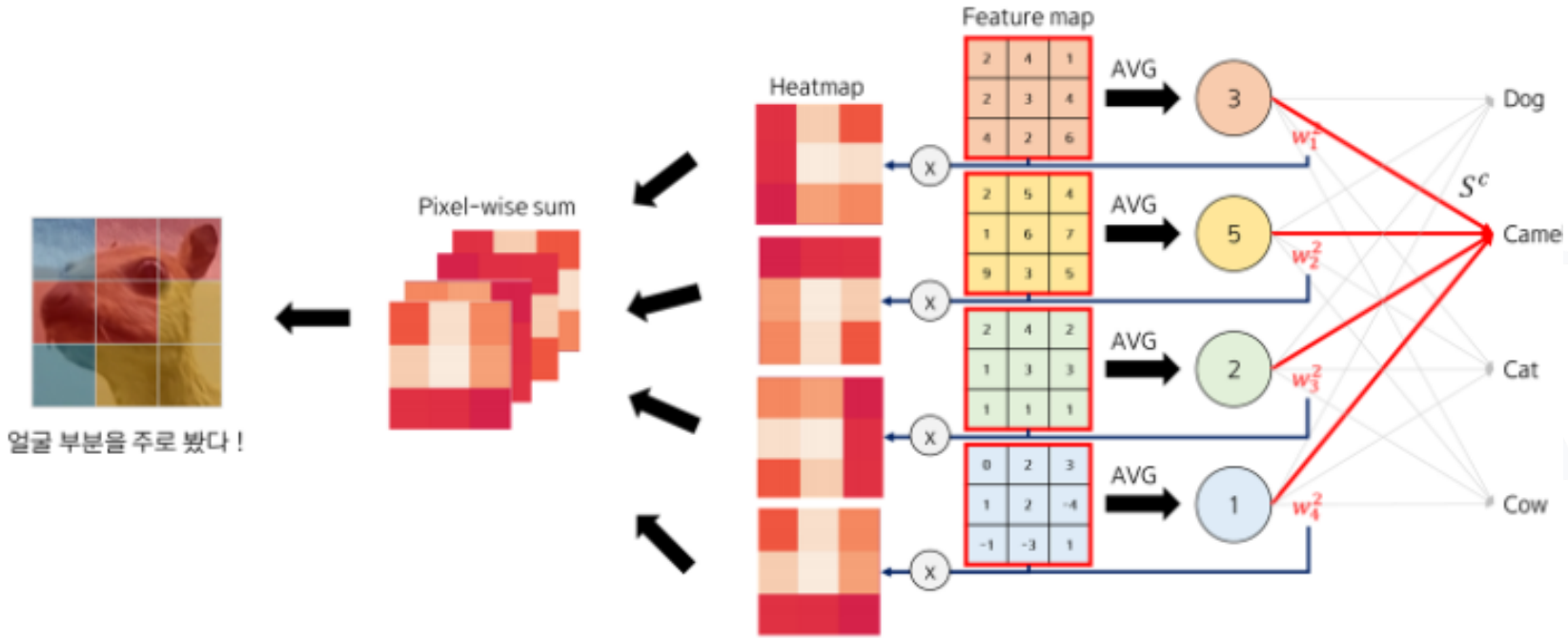
각 feature map $f_k(i,j)$에 각 class에 대한 가중치 w_k^c를 곱해주면 heatmap을 feature map 개수 k만큼 얻을 수 있습니다.



얼굴 부분을 주로 봤다 !

이 heatmap 이미지를 모두 pixel-wise sum을 해주면, 하나의 heatmap을 얻을 수 있는데, 이게 바로 CAM 입니다.

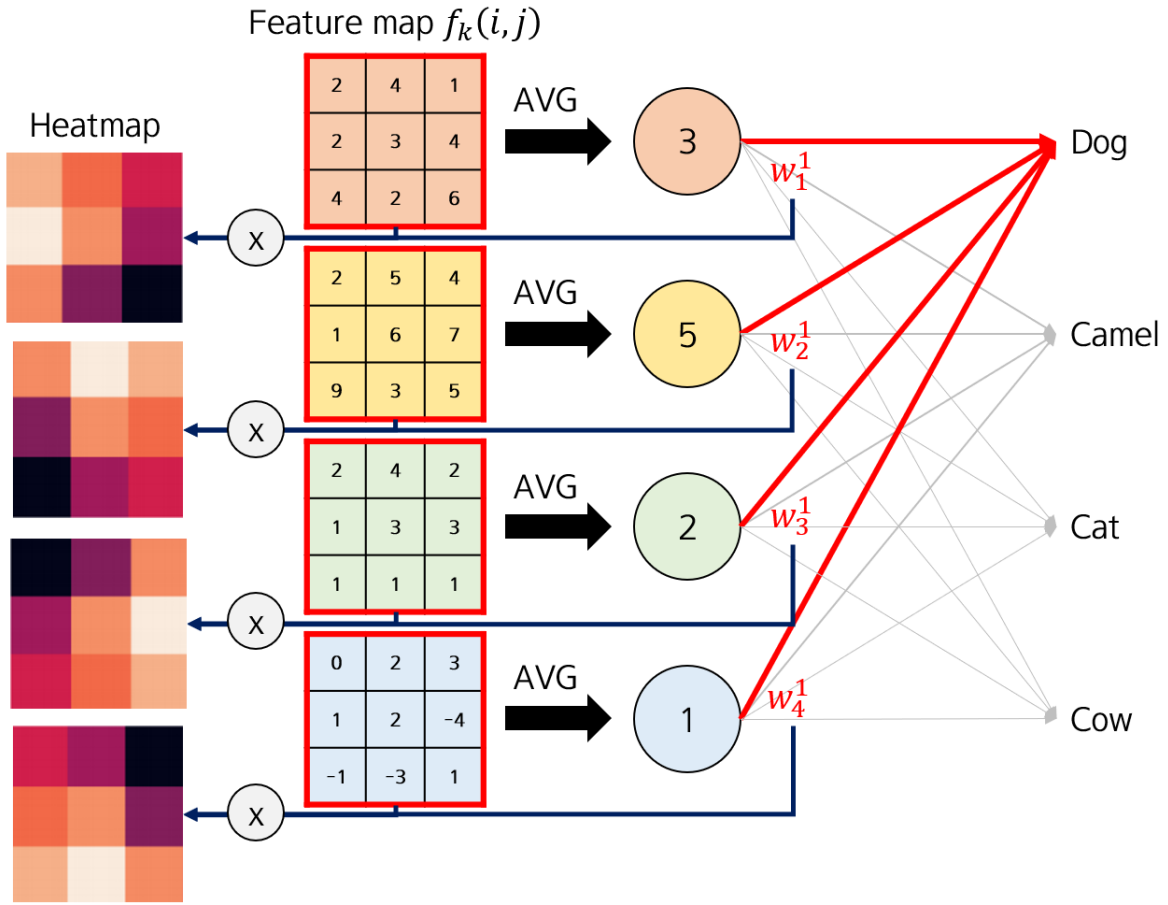
CAM (Class Activation Map)



$$L_{CAM}^{c}(i,j) = \sum_{k} w_k^c f_k(i,j)$$

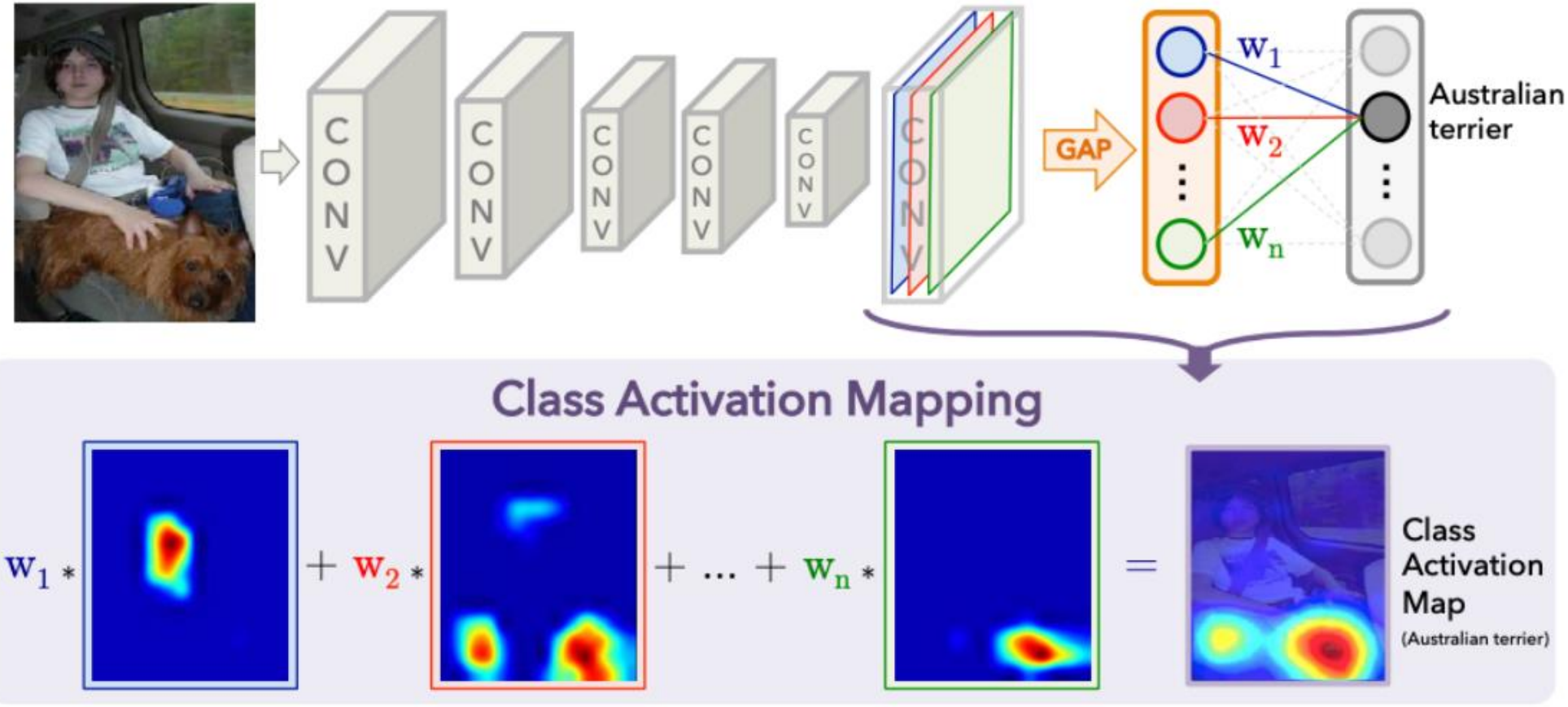
CAM (Class Activation Map)

- 전체 과정을 요약하면



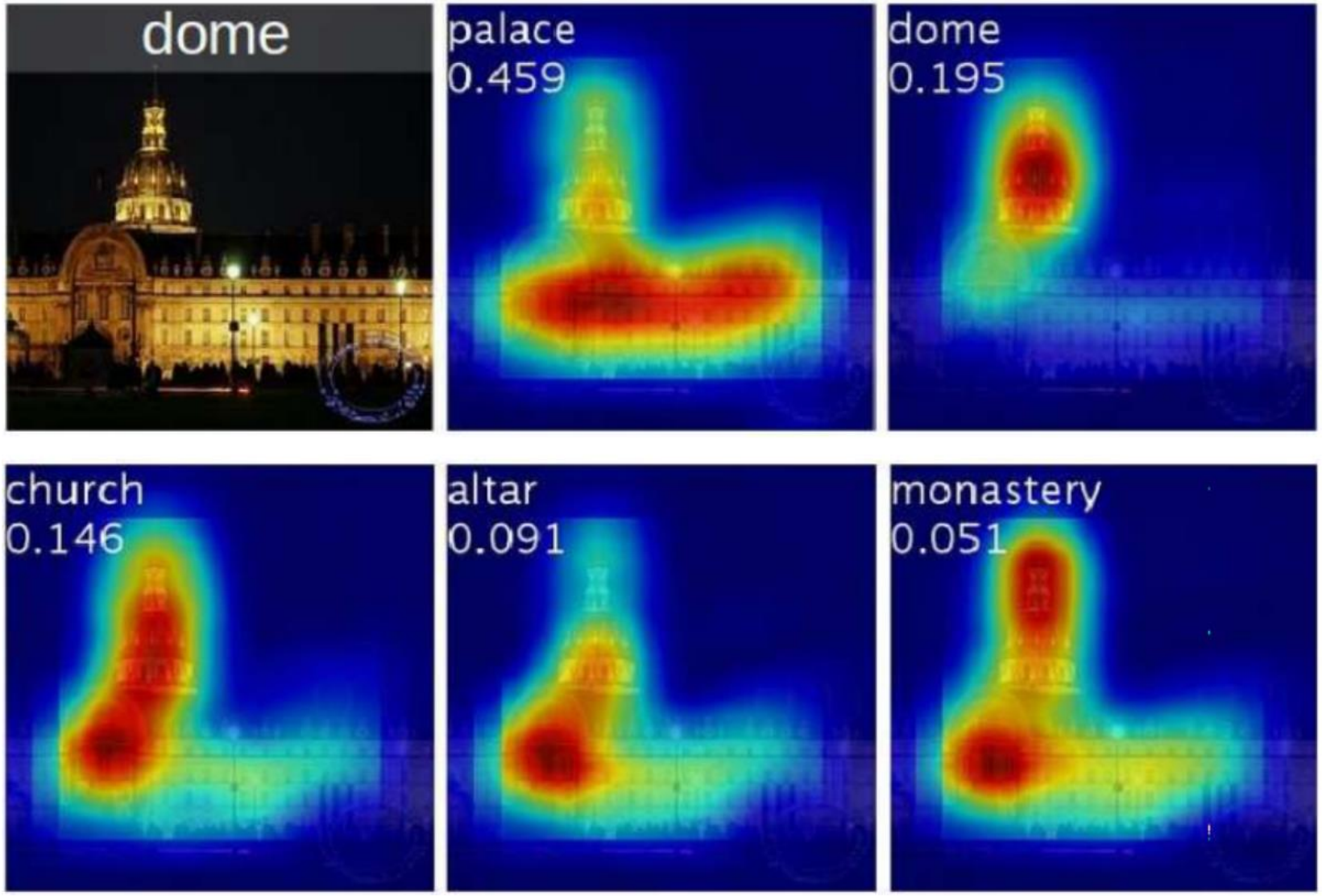
- CAM의 특징은 Class 마다 계산할 수 있다는 것
- 즉, Dog class로 CAM을 계산할 경우에는, 다른 heatmap을 얻게 됨

CAM (Class Activation Map)



- CAM을 입력 이미지 크기로 Up-Sampling하면 특정 Class c와 관련된 부분을 찾을 수 있음

CAM (Class Activation Map)



CAM (Class Activation Map)

- CAM의 장점
  - CAM은 CNN의 내부를 열어볼 수 있는 아주 유용한 도구
  - 기존 모델의 Convolution Layer 뒤에 붙은 FC Layer를 GAP로 교체하고 fine-tuning함으로써, 뉴럴 네트워크가 이미지의 어떤 부분을 보고 특정 레이블로 판단을 내리는지에 대해 알 수 있음
  - 기존 접근 방식들에 비해 end-to-end로 한번에 (single forward pass) CAM을 얻을 수 있다는 점에서 매우 매력적인 방법

CAM (Class Activation Map)

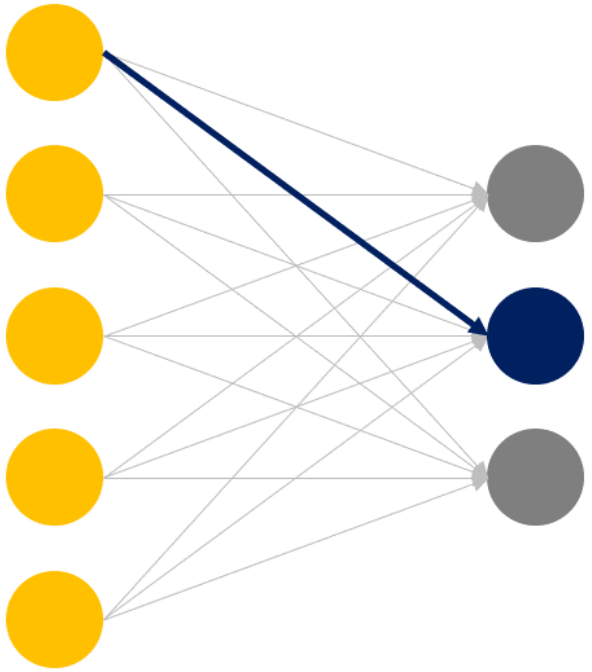
- **CAM의 한계**
  - **CAM은 그 방식으로 인한 태생적인 단점을 가짐**
    - FC를 GAP로 대체해야 한다
    - GAP 직전의 Conv layer만 쓸 수 있다
    - GAP 뒷단의 Dense Layer의 weight 정보가 필요하므로 Fine-Tuning 이나 Re-Training의 과정을 거쳐야 한다
  - **이러한 문제로 인해 Object Detection 외에 Visual Question Answer나 Captioning처럼 다양한 목적을 수행하는 CNN에는 CAM을 적용하기 어려움**

Grad-CAM (Gradient-weighted CAM)

- **Grad-CAM**
  - **CAM을 일반화한 버전**
  - **CAM의 한계점을 Gradient를 이용하여 해결함**
  - **Gradient의 특징은?**
    - Gradient는 Class C에 대하여 Input K가 주는 영향력이다.

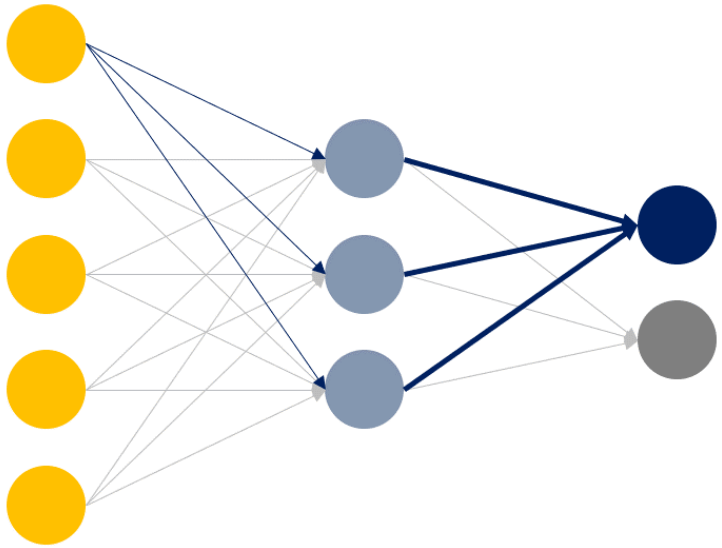
Grad-CAM (Gradient-weighted CAM)

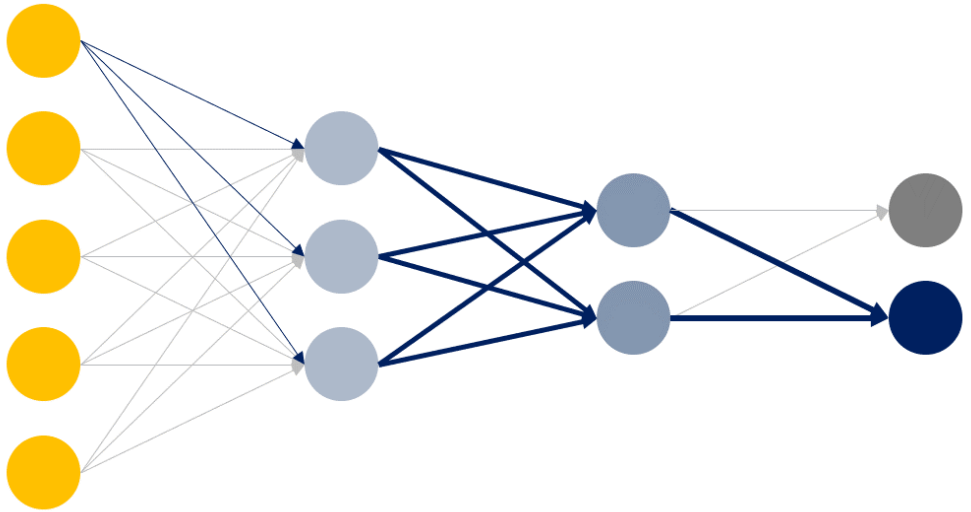
- **Gradient의 특징**
  - **hidden layer가 없는 경우, 각 input 노드는 output 노드에 하나의 weight 를 통해 영향을 주며, 또한 이 path를 통해 Gradient 가 역전파됨**



Grad-CAM (Gradient-weighted CAM)

- hidden layer가 추가되면, 각 input 노드는 모든 hidden layer의 노드를 거쳐 output 노드에 영향을 주며,
- Gradient 또한 모든 hidden layer를 거쳐 역전파됨





Gradient
[Total amount of effect of input K on output class C]

Grad-CAM (Gradient-weighted CAM)

- **Grad-CAM에서는**
  - **별도의 CNN 구조를 따를 필요가 없음 → GAP Layer를 쓰지 않아도 됨**
  - **CAM, Grad-CAM 수식 비교**

CAM

$$L^c_{CAM}(i,j) = \sum_k w^c_k f_k(i,j)$$

Grad-CAM

$$L^c_{Grad-CAM}(i,j) = ReLU(\sum_k a^c_k f_k(i,j))$$

$$a^c_k = \frac{1}{Z}\sum_i\sum_j \frac{\partial S^c}{\partial f_k(i,j)}$$

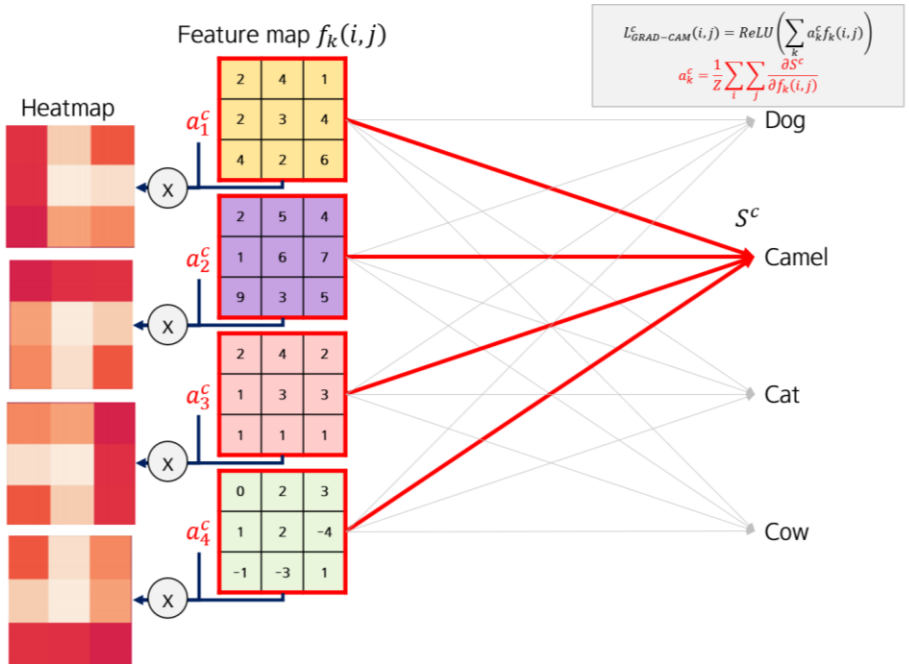
a^c_k의 수식

k번째 feature map $f_k(i,j)$의 각 원소 i,j가 matmul값 S^c에 주는 영향력의 평균

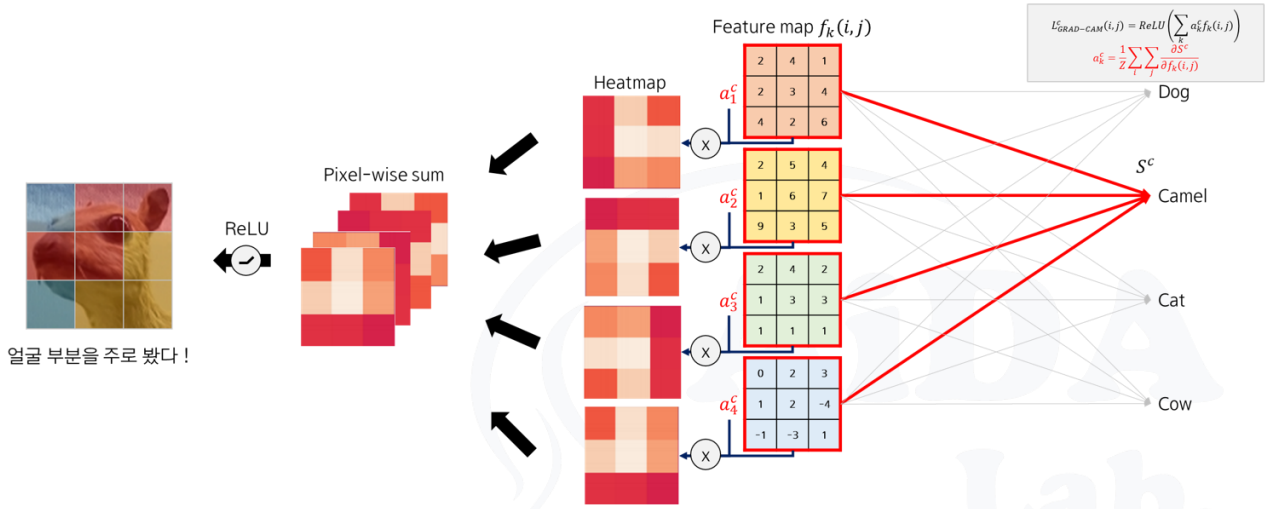
  - **즉, CAM에서는 weight으로 주었던 각 feature map의 가중치를, Gradient로 대신 주었다고 볼 수 있음**

Grad-CAM (Gradient-weighted CAM)

- 그림으로 정리



Gradient의 픽셀별 평균값인 a_k^c를
각 feature map $f_k(i,j)$에 곱해
heatmap을 만듦



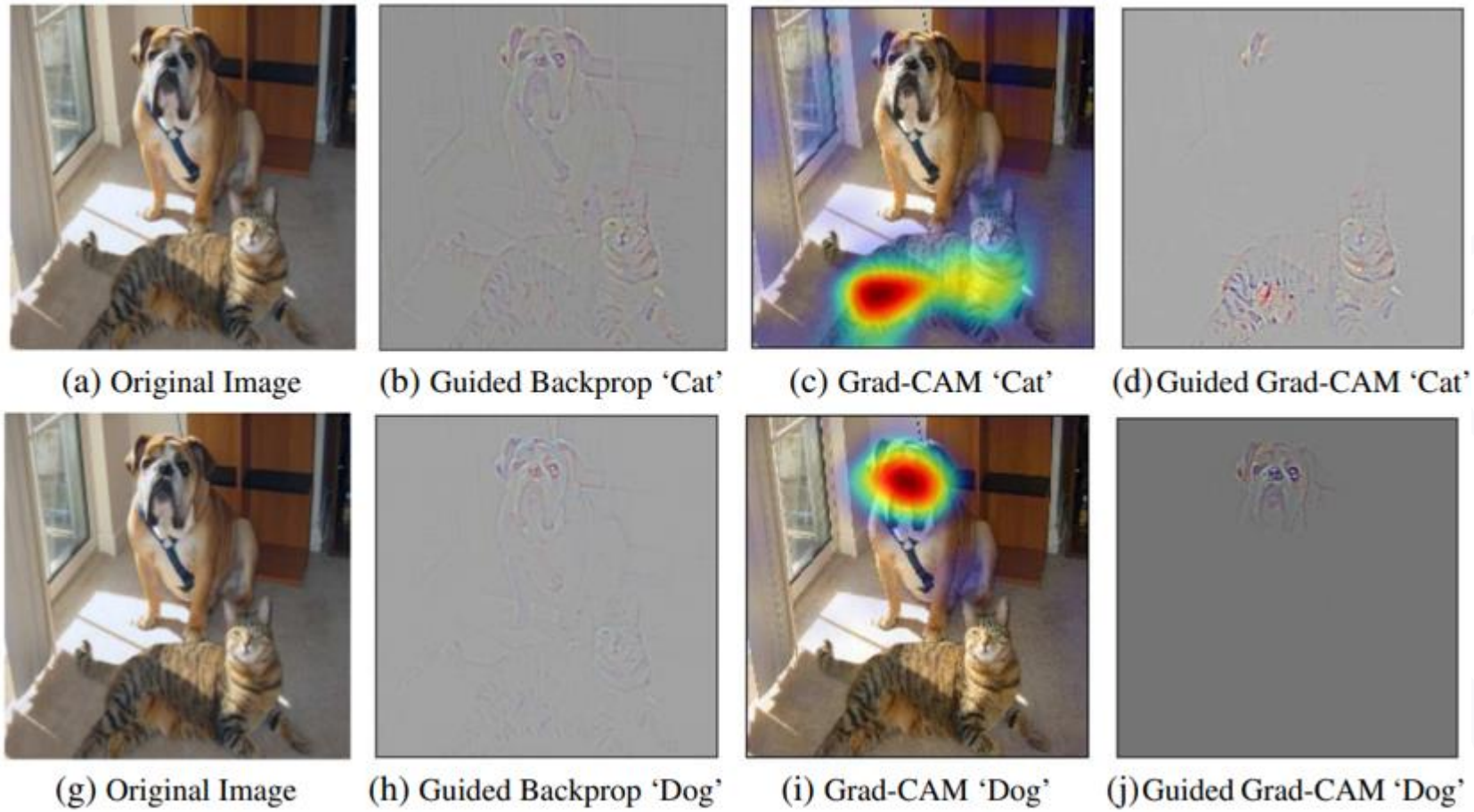
$$L^c_{GRAD-CAM}(i,j) = ReLU\left(\sum_k a_k^c f_k(i,j)\right)$$

$$a_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial S^c}{\partial f_k(i,j)}$$

pixel-wise sum을 한 후, ReLU 함수를 적용해 양의 가중치를 갖는
(중요하게 여기는) 부분을 골라내면 Grad-CAM이 됨

Grad-CAM (Gradient-weighted CAM)

- **Grad-CAM의 결과**



- (b): Guided backpropagation
  specific한 곳을 visualization 해주는 방법

- local 한 특성을 보여주는 Grad-CAM과
- Specific한 특성을 보여주는 Guided backpropagation을
- pixel-wise multiplication하게되면,
- Local+Specific 특징을 모두 갖는
- Guided Grad-CAM을 얻을 수 있음

Grad-CAM (Gradient-weighted CAM)

- **Reference**
  - [CNN visualization: CAM and Grad-CAM 설명 - tyami's study blog (https://tyami.github.io/deep%20learning/CNN-visualization-Grad-CAM/)]