BrAIn Lab.
Brain and Artificial Intelligence Lab.

Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

컴퓨터공학과
23100493 김지윤

0. Author

BrAIn Lab.
Brain and Artificial Intelligence Lab.

Ramprasaath R. Selvaraju

FOLLOW

Explainable AI Researcher
Verified email at apple.com

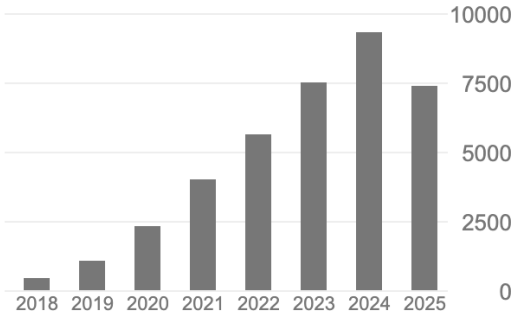
Explainable AI Computer Vision Machine Learning

GET MY OWN PROFILE

TITLE	CITED BY	YEAR
Grad-cam: Visual explanations from deep networks via gradient-based localization RR Selvaraju, M Cogswell, A Das, R Vedantam, D Parikh, D Batra Proceedings of the IEEE international conference on computer vision, 618-626	32768	2017
Align before fuse: Vision and language representation learning with momentum distillation J Li, R Selvaraju, A Gotmare, S Joty, C Xiong, SCH Hoi Advances in neural information processing systems 34, 9694-9705	2671	2021
Grad-CAM: Why did you say that? RR Selvaraju, A Das, R Vedantam, M Cogswell, D Parikh, D Batra arXiv preprint arXiv:1611.07450	924	2016
Proceedings of the IEEE international conference on computer vision RR Selvaraju, M Cogswell, A Das, R Vedantam, D Parikh, D Batra Proceedings of the IEEE international conference on computer vision [J]	334	2017
Taking a hint: Leveraging explanations to make vision and language models more grounded RR Selvaraju, S Lee, Y Shen, H Jin, S Ghosh, L Heck, D Batra, D Parikh Proceedings of the IEEE/CVF international conference on computer vision …	319	2019
Diverse beam search for improved description of complex scenes A Vijayakumar, M Cogswell, R Selvaraju, Q Sun, S Lee, D Crandall, ... Proceedings of the AAAI Conference on Artificial Intelligence 32 (1)	296	2018

Cited by

	All	Since 2020
Citations	38150	36294
h-index	15	15
i10-index	18	18



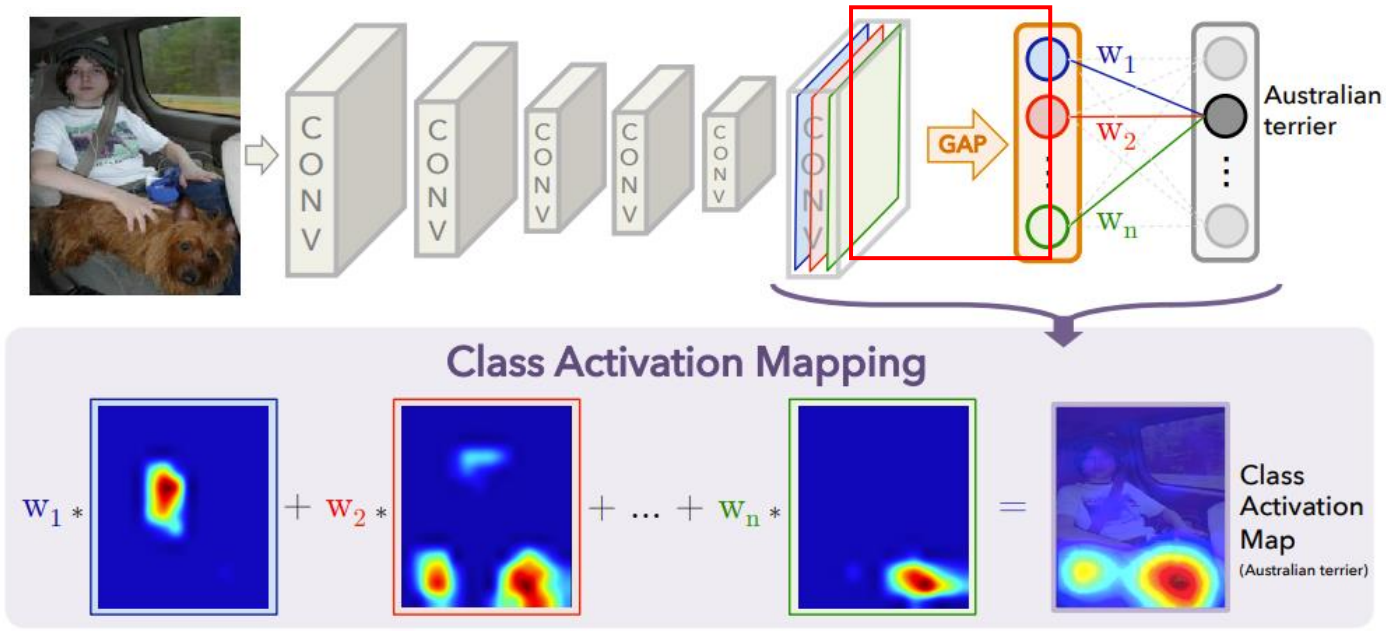
Public access

VIEW ALL

0 articles	7 articles
not available	available

Based on funding mandates

1. Background



- **Class Activation Map**

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

c: 특정 클래스 ex. 고양이

$f_k(x,y)$: k번째 Feature map

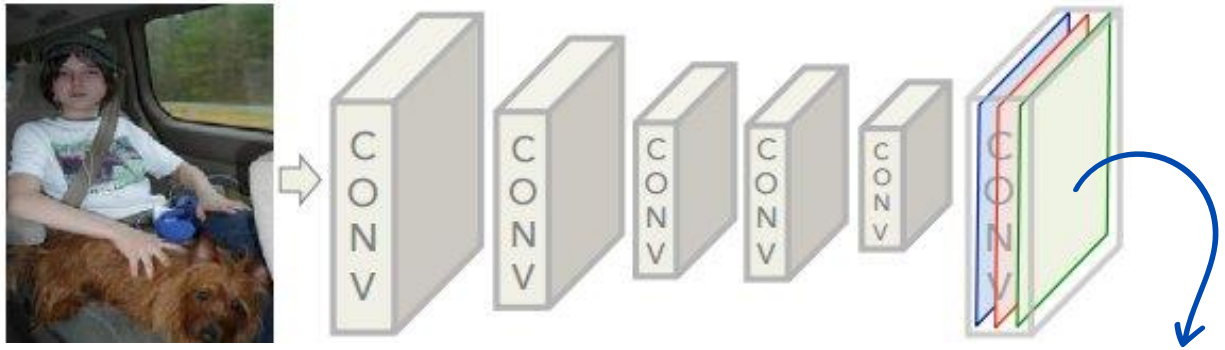
특정 클래스 c 최종 점수: $S_c = \sum_k w_k^c F_k$

➡ w_k^c: 각 Feature map의 중요도 점수

"왜 CAM은 GAP이 필요했는가?"

1. Background

BrAIn Lab.
Brain and Artificial Intelligence Lab.



- 모델이 이미지의 왼쪽 위에 귀가 있다고 학습했을 때,

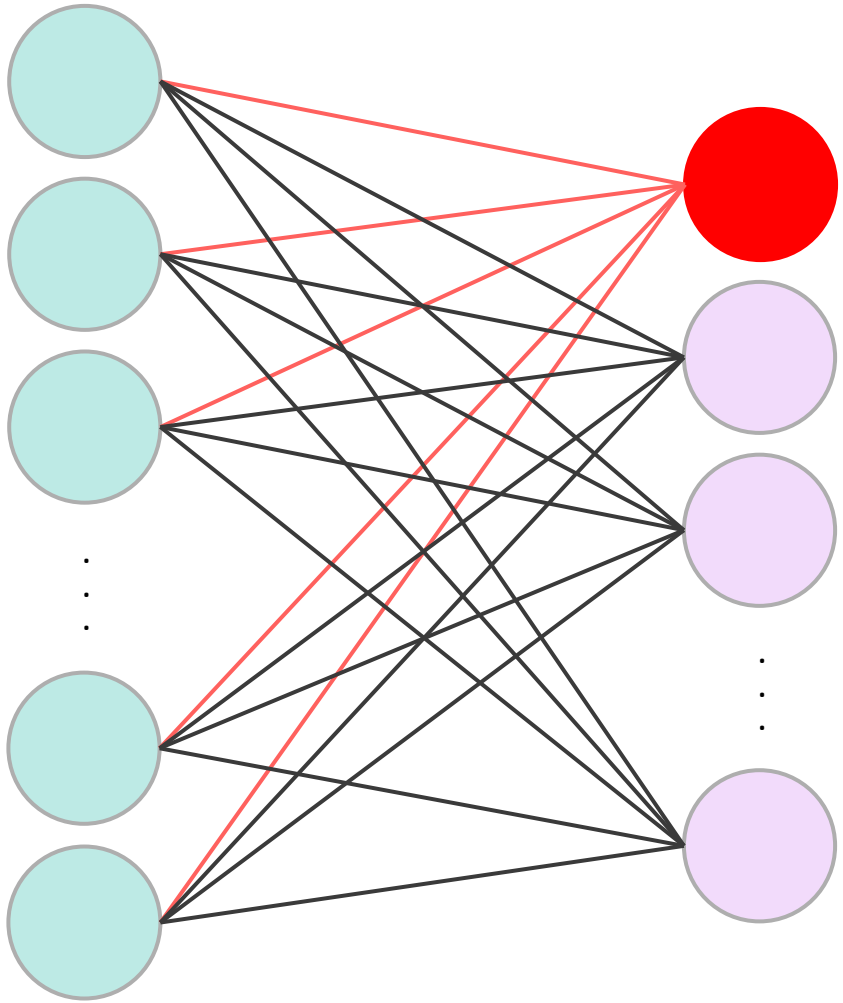
$\begin{pmatrix} 10 & 1 \\ 1 & 1 \end{pmatrix}$ → Flatten: $[10, 1, 1, 1]$

Fully Connected Layer: Dog Class Weight = [0.9, 0.1, 0.1, 0.1]

Dog Class's 최종 점수: $10 * 0.9 + 1 * 0.1 + 1 * 0.1 + 1 * 0.1 = 9.3$

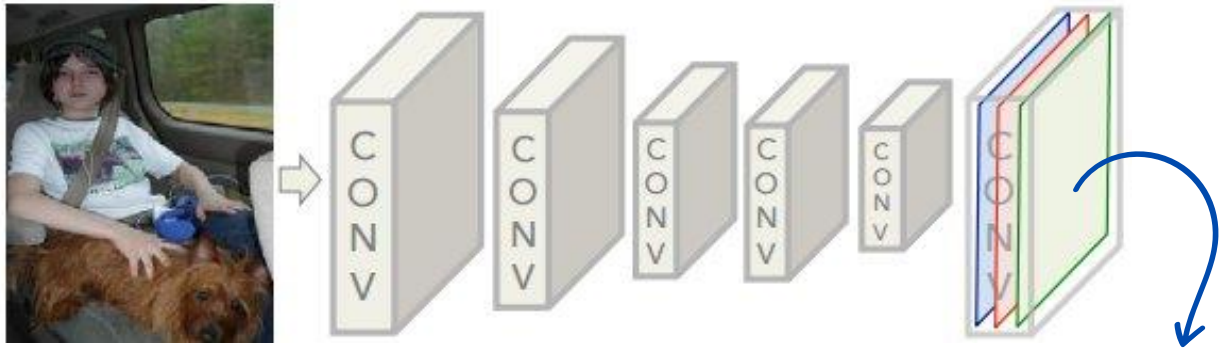
→ 매우 높은 점수로 "강아지"라 확신 및 분류한다.

FC Layer: Flatten(F_k) → 분류할 클래스



1. Background

BrAIn Lab.
Brain and Artificial Intelligence Lab.



- 예측을 진행할 새로운 이미지의 강아지 귀가 오른쪽 아래에 있을 때,

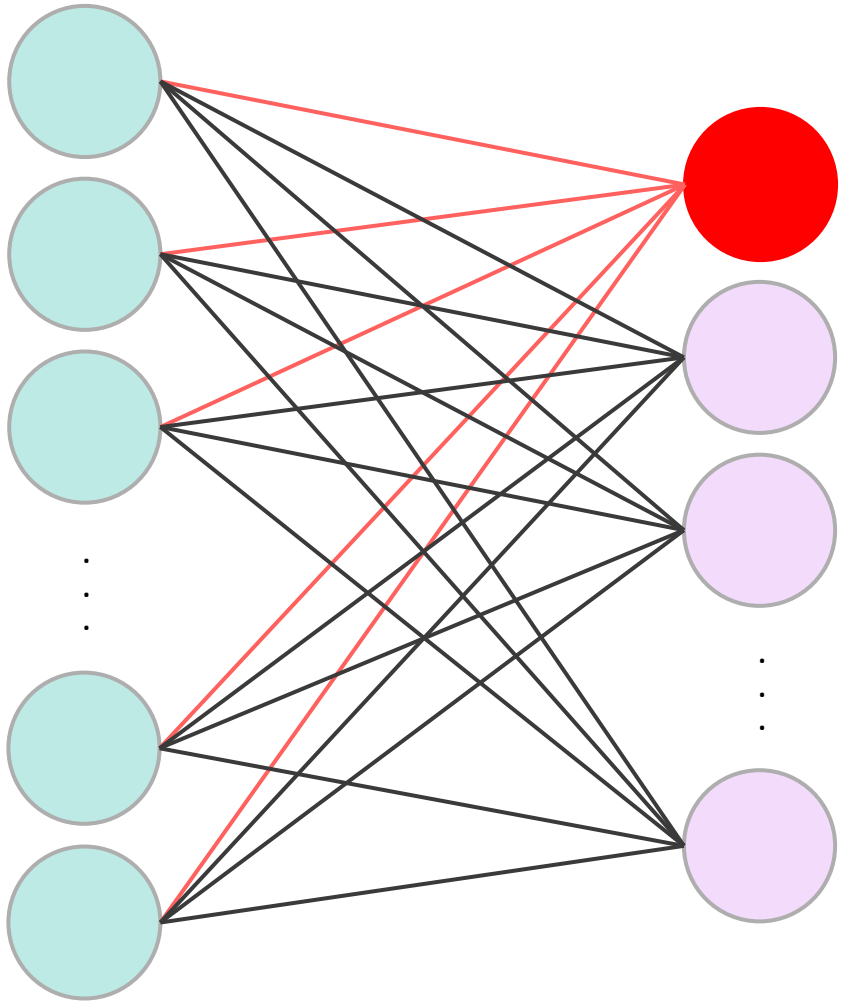
$\begin{pmatrix} 1 & 1 \\ 1 & 10 \end{pmatrix}$ → Flatten: $[1, 1, 1, 10]$

Fully Connected Layer: Dog Class Weight = [0.9, 0.1, 0.1, 0.1]

→ 학습된 그대로 고정

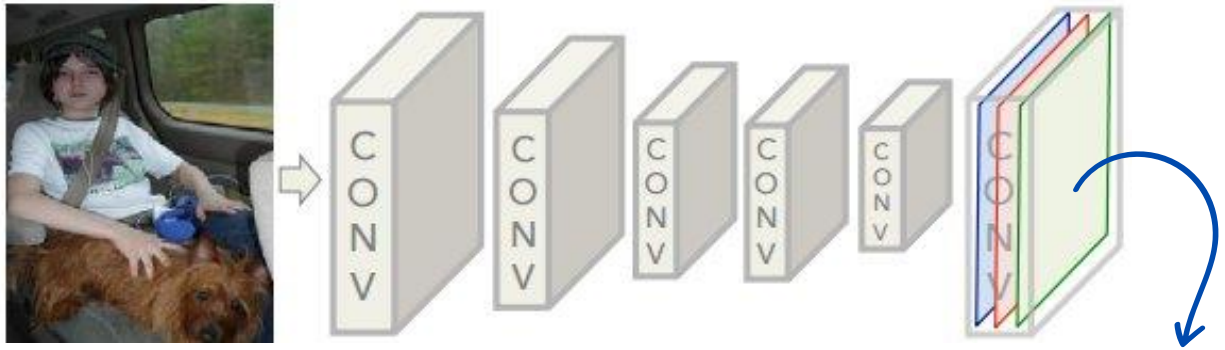
Dog Class's 최종 점수: $1 * 0.9 + 1 * 0.1 + 1 * 0.1 + 10 * 0.1 = 2.1$

FC Layer: Flatten(F_k) → 분류할 클래스



1. Background

BrAIn Lab.
Brain and Artificial Intelligence Lab.



- 모델이 이미지의 왼쪽 위에 귀가 있다고 학습했을 때,

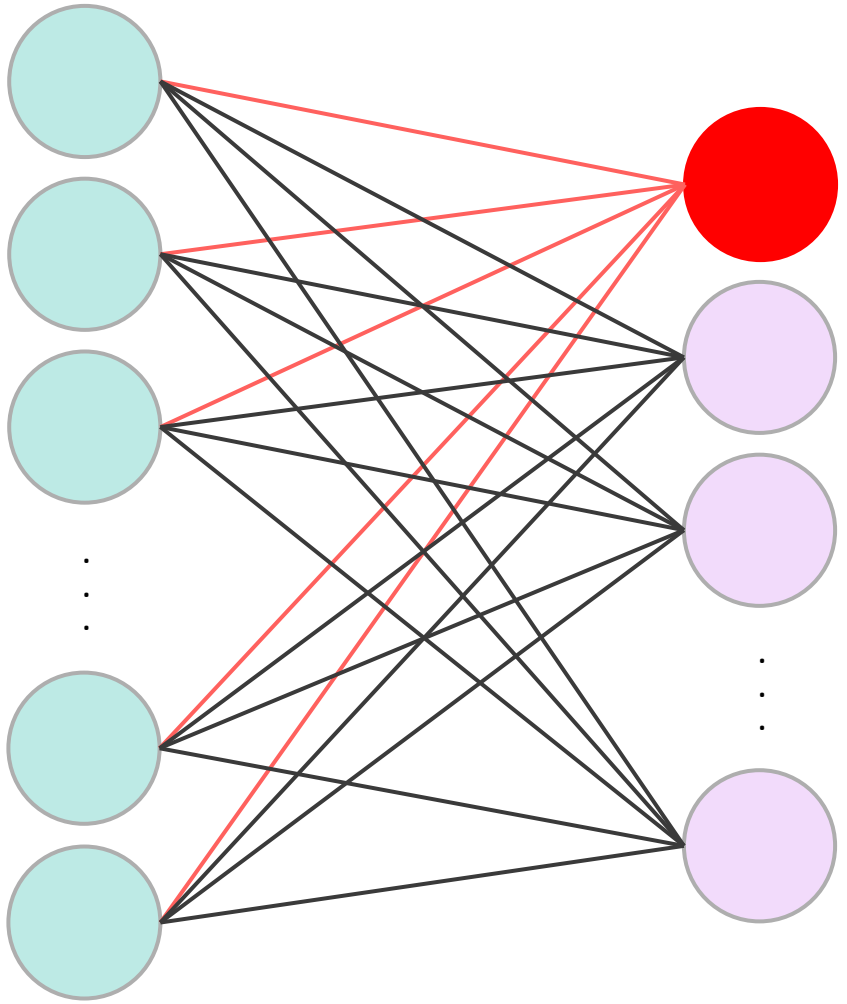
$\begin{pmatrix} 10 & 1 \\ 1 & 1 \end{pmatrix}$ → Global Average Pooling(GAP): $\frac{10+1+1+1}{4} = 3.25$

Fully Connected Layer: Dog Class Weight = [2.86]

Dog Class's 최종 점수: $3.25 * 2.86 \approx 9.3$

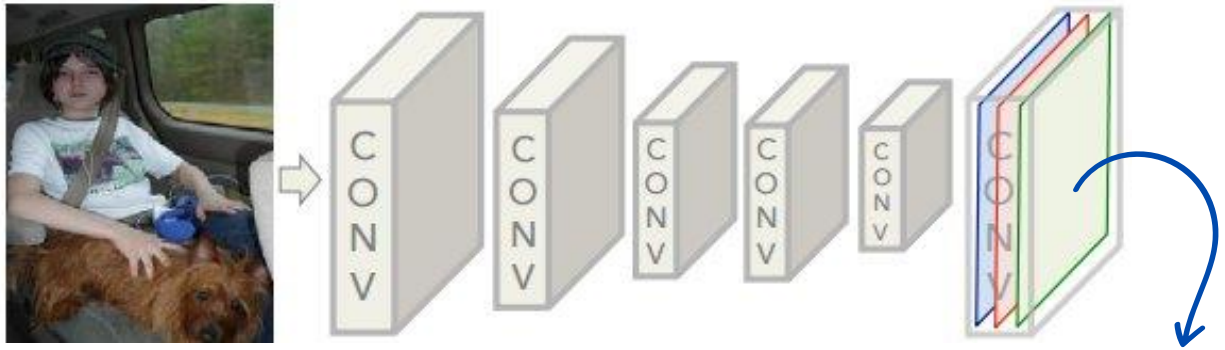
→ 매우 높은 점수로 "강아지"라 확신 및 분류한다.

FC Layer: Global Average Pooling(F_k) → 분류할 클래스

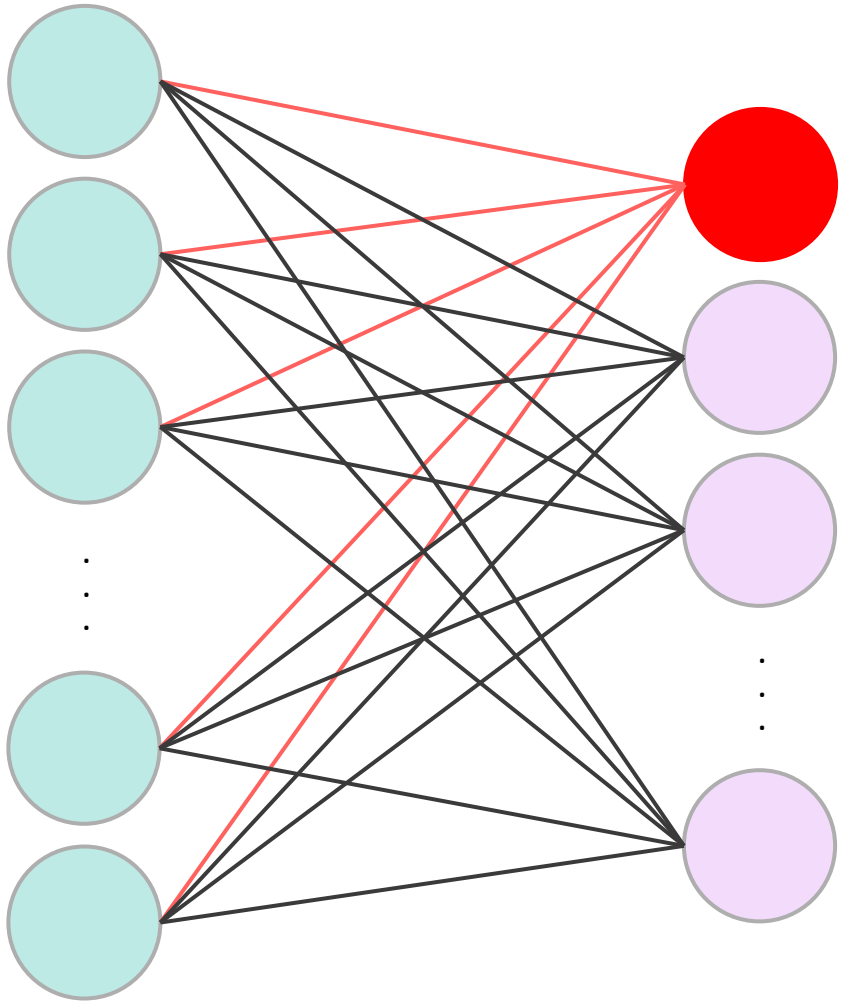


1. Background

BrAIn Lab.
Brain and Artificial Intelligence Lab.



FC Layer: Global Average Pooling(F_k) → 분류할 클래스



- 예측을 진행할 새로운 이미지의 강아지 귀가 오른쪽 아래에 있을 때,

$\begin{pmatrix} 1 & 1 \\ 1 & 10 \end{pmatrix}$ → Global Average Pooling(GAP): $\frac{10+1+1+1}{4} = 3.25$

Fully Connected Layer: Dog Class Weight = [2.86]

→ 학습된 그대로 고정

Dog Class's 최종 점수: $3.25 * 2.86 \approx 9.3$

1. Background

BrAIn Lab.
Brain and Artificial Intelligence Lab.

- **Class Activation Map**

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

c: 특정 클래스 ex. 고양이

$f_k(x,y)$: k번째 Feature map

특정 클래스 c 최종 점수: $S_c = \sum_k w_k^c F_k$

➡ w_k^c: 각 Feature map의 중요도 점수

- 모델의 마지막 Convolution Layer 4개의 Filter

$k_1 \rightarrow F_1$: "뾰족한 귀" 특징을 추출하는 Filter 및 결과 Feature map$_1$

$k_2 \rightarrow F_2$: "수염/코" 특징을 추출하는 Filter 및 결과 Feature map$_2$

$k_3 \rightarrow F_3$: "낼름거리는 혀" 특징을 추출하는 Filter 및 결과 Feature map$_3$

$k_4 \rightarrow F_3$: "바퀴 모양" 특징을 추출하는 Filter 및 결과 Feature map$_4$

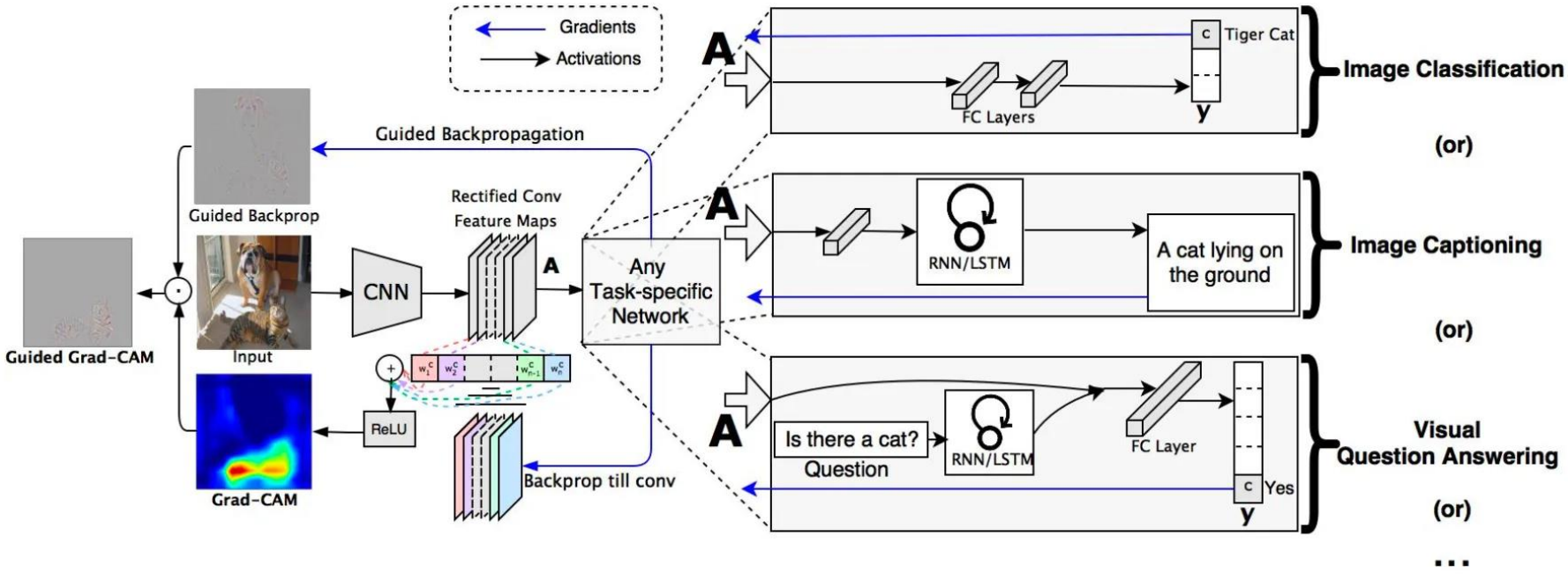
- 모델이 학습을 마친 후 최종 클래스 "고양이, 강아지"를 판단하기 위해 중요도 점수

$$W_{cat} = [0.9, 0.7, 0.3, -0.1],\ W_{dog} = [0.2, 0.8, 0.9, -0.1]$$

- CAM 시각화

$$M_{cat} = f_1(x,y) * 0.9 + f_2(x,y) * 0.7 + f_3(x,y) * 0.3 + f_4(x,y) * (-0.1)$$

2. Grad-CAM



$$\alpha_k^c = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \rightarrow L_{Grad-CAM}^c = ReLU\left(\sum_k \alpha_k^c A^k\right)$$

2. Grad-CAM

- 모델 구조:

마지막 Convolution Layer → Global Average Pooling → Fully-Connected Layer → 최종 점수

- 마지막 Convolutional Layer의 출력 특징 맵 A

각 채널별 특징 맵 A^1, A^2, A^3

$$A^1 = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix},\ A^2 = \begin{pmatrix} 2 & 1 \\ 1 & 5 \end{pmatrix},\ A^3 = \begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix}$$

- Fully-Connected Layer의 가중치 W

클래스는 "고양이"와 "개" 두 가지가 있다고 가정할 때, "고양이" 클래스 예측을 위한 가중치 벡터 w^{cat}

$$w^{cat} = \begin{pmatrix} w_1^{cat} \\ w_2^{cat} \\ w_3^{cat} \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.8 \\ -0.2 \end{pmatrix} \quad \text{참고 : } w^{dog} = \begin{pmatrix} -0.3 \\ 0.1 \\ 0.9 \end{pmatrix}$$

2. Grad-CAM

$$f^1 = Average(A^1) = \frac{1+3+4+2}{4} = 2.5,\ f^2 = Average(A^2) = 2.25,$$

$$f^3 = Average(A^3) = 1.5$$

$$y^{cat} = w_1^{cat} f^1 + w_2^{cat} f^2 + w_3^{cat} f^3 = 2.5 \cdot 0.5 + 2.25 \cdot 0.8 + 1.5 \cdot (-0.2) = 2.75$$

$$y^{dog} = 1.1,\ y^{car} = -0.5$$

→ 고양이의 점수인 2.75가 가장 높기 때문에 모델은 이 이미지를 '고양이'일 가능성이 가장 높다고 판단

$[y^{cat}, y^{dog}, y^{car}]$ = [2.75, 1.1, -0.5]

$$P(class_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

$$P(cat) = \frac{e^{2.75}}{e^{2.75}+e^{1.1}+e^{-0.5}} \approx 0.83$$

$$P(dog) = \frac{e^{1.1}}{e^{2.75}+e^{1.1}+e^{-0.5}} \approx 0.16$$

$$P(car) = \frac{e^{-0.5}}{e^{2.75}+e^{1.1}+e^{-0.5}} \approx 0.01$$

2. Grad-CAM

BrAIn Lab.
Brain and Artificial Intelligence Lab.

"고양이" 클래스 점수 y^{cat}에 대한 마지막 합성곱 레이어의 각 픽셀별 Gradient, 즉 Gradient map $\frac{\partial y^{cat}}{\partial A_{ij}^k}$를

- 연쇄 법칙을 이용하여 계산하기

Gradient map → $\frac{\partial y^{cat}}{\partial A_{ij}^k} = \frac{\partial y^{cat}}{\partial f^k} \times \frac{\partial f^k}{\partial A_{ij}^k}$

$\frac{\partial y^{cat}}{\partial f^1} = \frac{\partial}{\partial f^1}(0.5f^1 + 0.8f^2 - 0.2f^3) = 0.5,\ \frac{\partial y^{cat}}{\partial f^2} = 0.8,\ \frac{\partial y^{cat}}{\partial f^3} = -0.2$

- "특징 맵의 특정 픽셀 A_{ij}^k 값이 변할 때, 그 채널의 평균값 f^k이 얼마나 변하는가?"를 계산

$f^k = \frac{1}{z}\sum_i \sum_j A_{ij}^k = \frac{1}{4}(A_{11}^k + A_{12}^k + A_{21}^k + A_{22}^k) \rightarrow \frac{\partial f^k}{\partial A_{ij}^k} = \frac{1}{4} = 0.25$

→ 채널 내의 모든 픽셀에 대해 동일

$\frac{\partial y^{cat}}{\partial A_{ij}^1} = 0.5 \times 0.25 = 0.125,\ \frac{\partial y^{cat}}{\partial A_{ij}^2} = 0.8 \times 0.25 = 0.2,\ \frac{\partial y^{cat}}{\partial A_{ij}^3} = -0.2 \times 0.25 = -0.05$

채널 1의 Gradient map → $\nabla A^1 = \begin{pmatrix} 0.125 & 0.125 \\ 0.125 & 0.125 \end{pmatrix},\ \nabla A^2 = \begin{pmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{pmatrix}$

→ 해당 값은 각 특징 맵의 픽셀 활성화 값이 "고양이" 점수에 얼마나 큰 영향을 미치는지를 나타낸다.

2. Grad-CAM

BrAIn Lab.
Brain and Artificial Intelligence Lab.

1. 필터 중요도 가중치 α_l^{cat} 계산:

Grad-CAM은 각 Gradient 맵에 대해 GAP을 다시 적용하여 채널별 최종 중요도 가중치 α_k^{cat} 구한다.

$$\alpha_1^{cat} = Average(\nabla A^1) = 0.125,\ \alpha_2^{cat} = Average(\nabla A^2) = 0.2,$$

$$\alpha_3^{cat} = Average(\nabla A^3) = -0.05$$

2. 히트맵 생성:

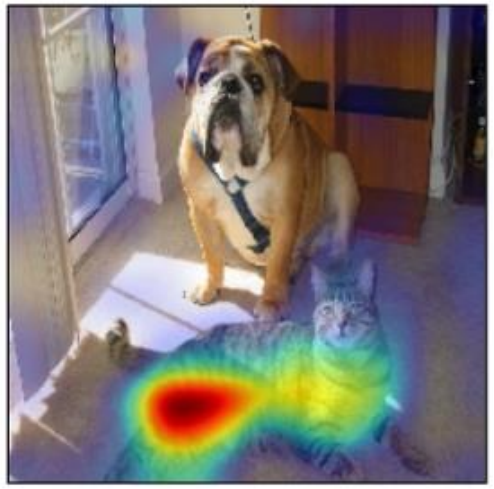
이 가중치들을 원본 특정 맵 A^k에 곱하여 더한 후, ReLU 함수를 적용하여 최종 Grad-CAM 히트맵을 만든다.

$$L_{Grad-CAM}^{cat} = ReLU(\sum_k \alpha_k^{cat} A^k) = ReLU(0.125 \cdot A^1 + 0.2 \cdot A^2 - 0.05 \cdot A^3)$$

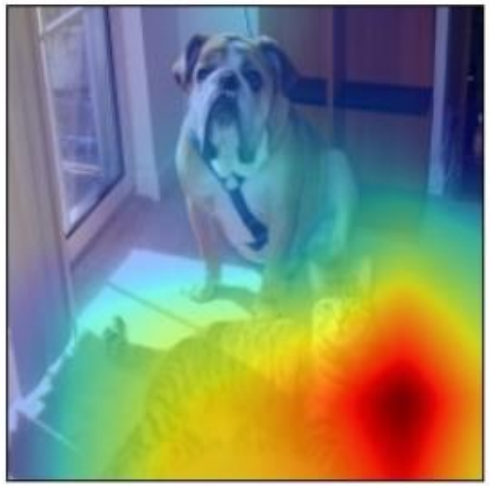
ReLU 함수: "고양이"일 확률을 높이는 증거들을 보고 싶은 것이지, "고양이"일 확률을 낮추는 증거에는 관심 X

3. Result

(a) Original Image



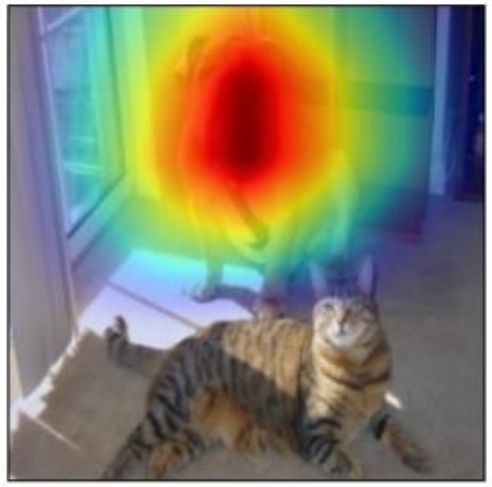
(c) Grad-CAM 'Cat'



(f) ResNet Grad-CAM 'Cat'

(g) Original Image

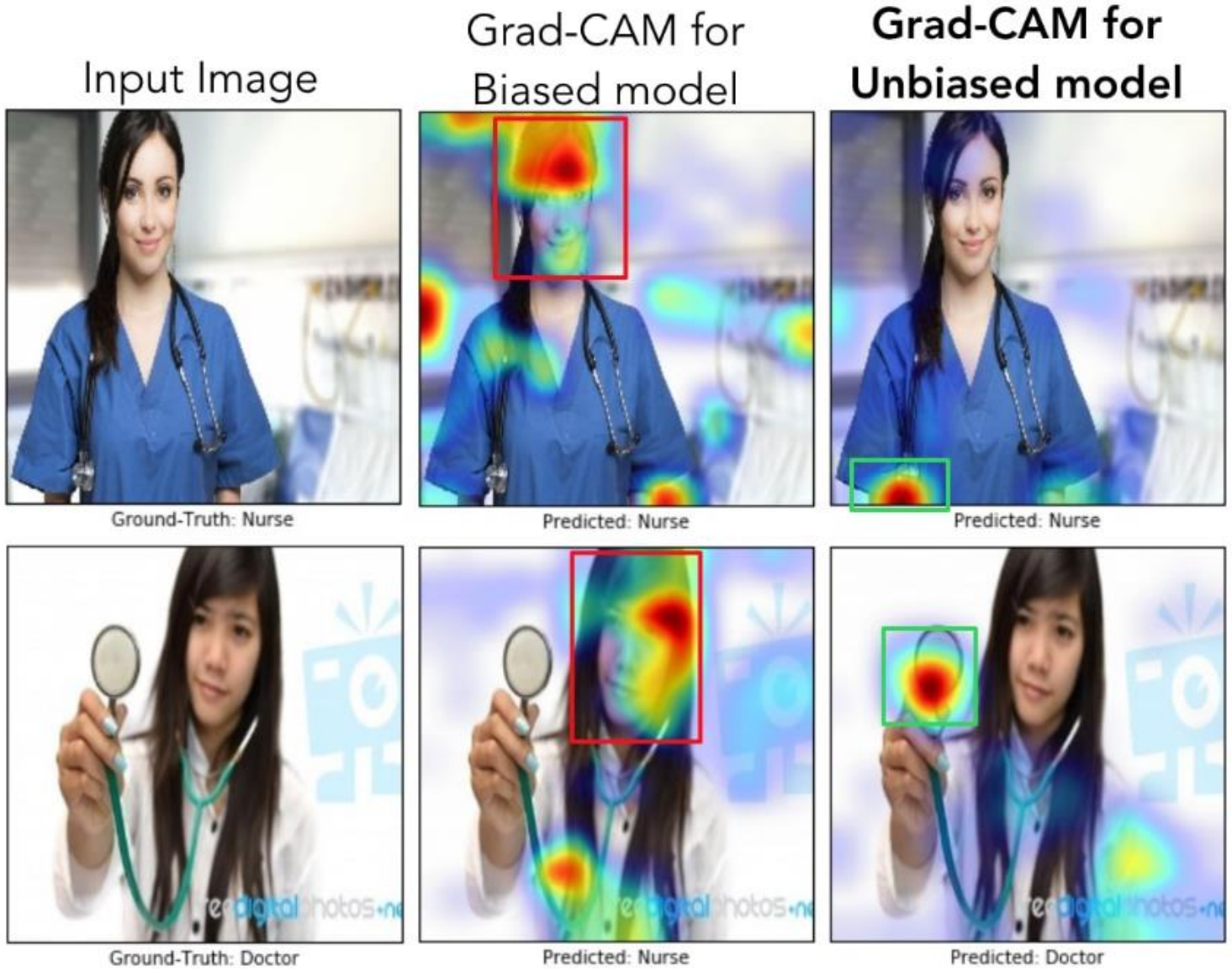
(i) Grad-CAM 'Dog'



(l)ResNet Grad-CAM 'Dog'

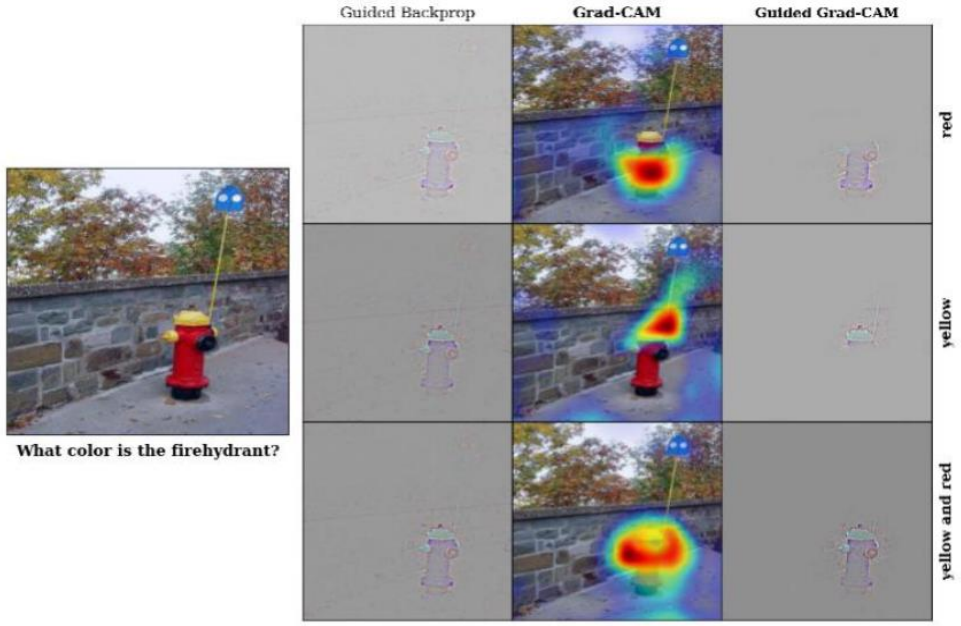
3. Result

BrAIn Lab.
Brain and Artificial Intelligence Lab.



3. Result

(a) Image captioning explanations



(a) Visualizing VQA model from [38]

4. Code

BrAIn Lab.
Brain and Artificial Intelligence Lab.

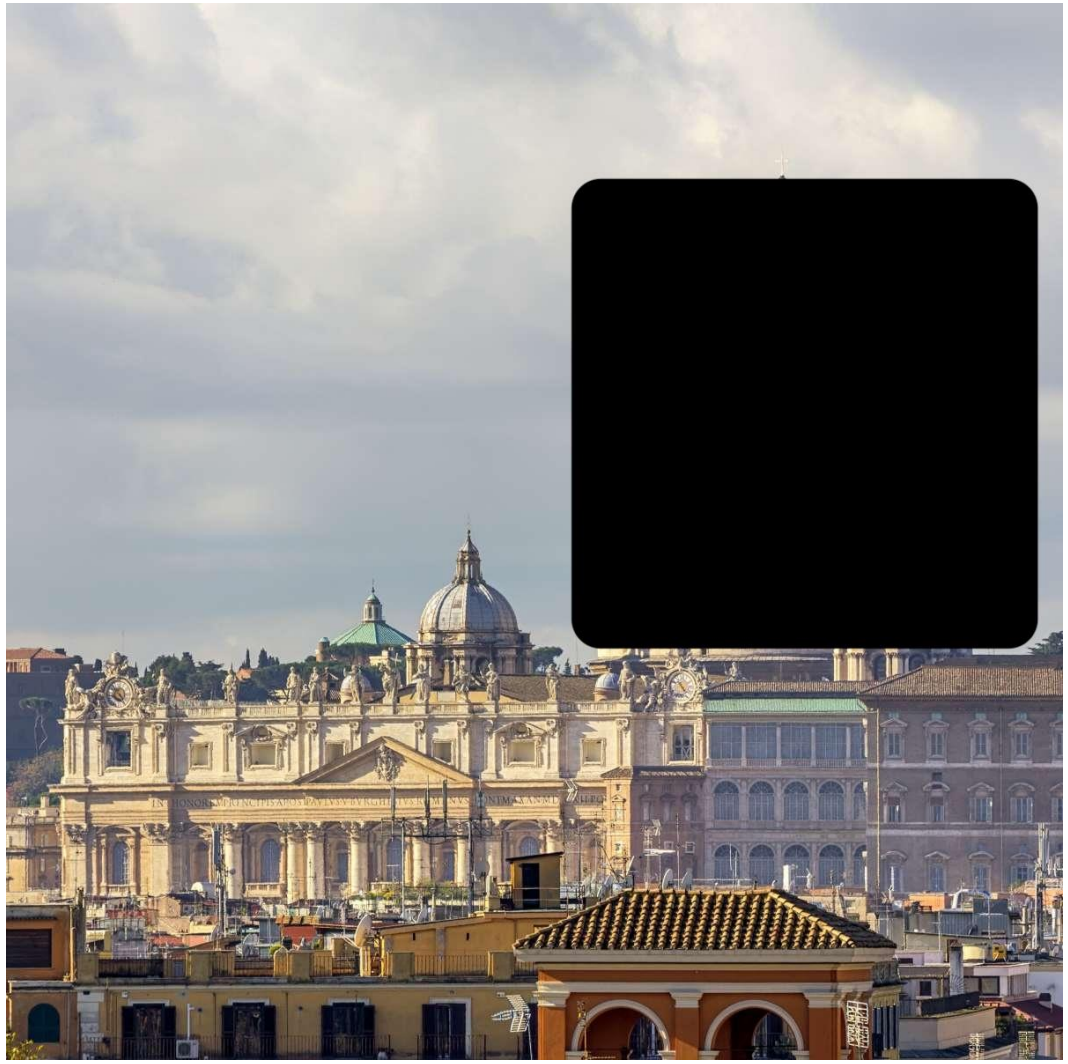
```python
# Grad-CAM 구현: 특정 레이어의 Feature Map(Activation)과 Gradient를 모두 얻어오기
class GradCAM:
    def __init__(self, model, target_layer):
        self.model = model
        self.target_layer = target_layer
        self.feature_map = None
        self.gradient = None

        self.target_layer.register_forward_hook(self.save_feature_map) #
        self.target_layer.register_full_backward_hook(self.save_gradient) #

    def save_feature_map(self, module, input, output):
        self.feature_map = output.detach()

    def save_gradient(self, module, gradient_input, gradient_output):
        self.gradient = gradient_output[0].detach()

    def generate_grad_cam(self, x, class_idx):
        # 목표 class 점수를 각 Feature Map으로 미분한 값을 모든 Feature map에 대해 구하는 과정 -> Grad-CAM을 그리기 위해
        self.model.zero_grad() # 이전 계산에서 남아있을 수 있는 그래디언트를 모두 0으로 초기화
        output = self.model(x)

        one_hot = torch.zeros_like(output) # output 텐서와 완전히 동일한 모양(shape)을 가지지만, 모든 요소가 0으로 채워진 새로운 텐서 생성 ex. [1, 1000] (배치크기 1, 클래스 1000개)
        one_hot[0][class_idx] = 1
        output.backward(gradient=one_hot, retain_graph=True) # 클래스 c에 대해 각 feature map으로 미분한 것이 self.gradient에 저장 -> Gradient map

        # GAP: gradient shape = [1, C, H, W] -> [1, C, 1, 1]
        weights = F.adaptive_avg_pool2d(self.gradient, 1)

        # Grad-CAM 생성
        # weights * self.feature_map: 각 채널에 해당하는 가중치가 곱해져서 [1, C, 1, 1] * [1, C, H, W] = [1, C, H, W]
        grad_cam = torch.sum(weights * self.feature_map, dim=1, keepdim=True) # C개의 모든 Feature Map을 더해서 하나의 Feature Map으로 압축: [1, C, H, W] -> [1, 1, H, W]
        grad_cam = F.relu(grad_cam)

        # 히트맵 시각화를 위해 크기 조정 및 정규화
        # 텐서에서 크기가 1인 차원을 모두 제거: [1, 1, H, W] -> [H, W] 후 텐서가 GPU(CUDA) 메모리에 올라가 있다면, CPU 메모리로 이동 및 pytorch 텐서를 numpy 배열로 변환한다.
        grad_cam = grad_cam.squeeze().cpu().numpy()
        # (x.shape[3], x.shape[2])는 크기를 조절할 목표 해상도로, OpenCV는 (너비, 높이) 순서로 인자를 받으며, 이를 통해 이미지 크기 조절
        grad_cam = cv2.resize(grad_cam, (x.shape[3], x.shape[2]))
        # 히트맵의 모든 픽셀 값을 0과 1 사이의 값으로 정규화(Normalization), Min-Max 정규화 적용
        # 0~1 사이로 값을 통일하면, 가장 낮은 활성화 영역은 0(파란색 계열), 가장 높은 활성화 영역은 1(붉은색 계열)로 일관되게 매핑 가능
        grad_cam = (grad_cam - np.min(grad_cam)) / (np.max(grad_cam) - np.min(grad_cam))

        return grad_cam
```

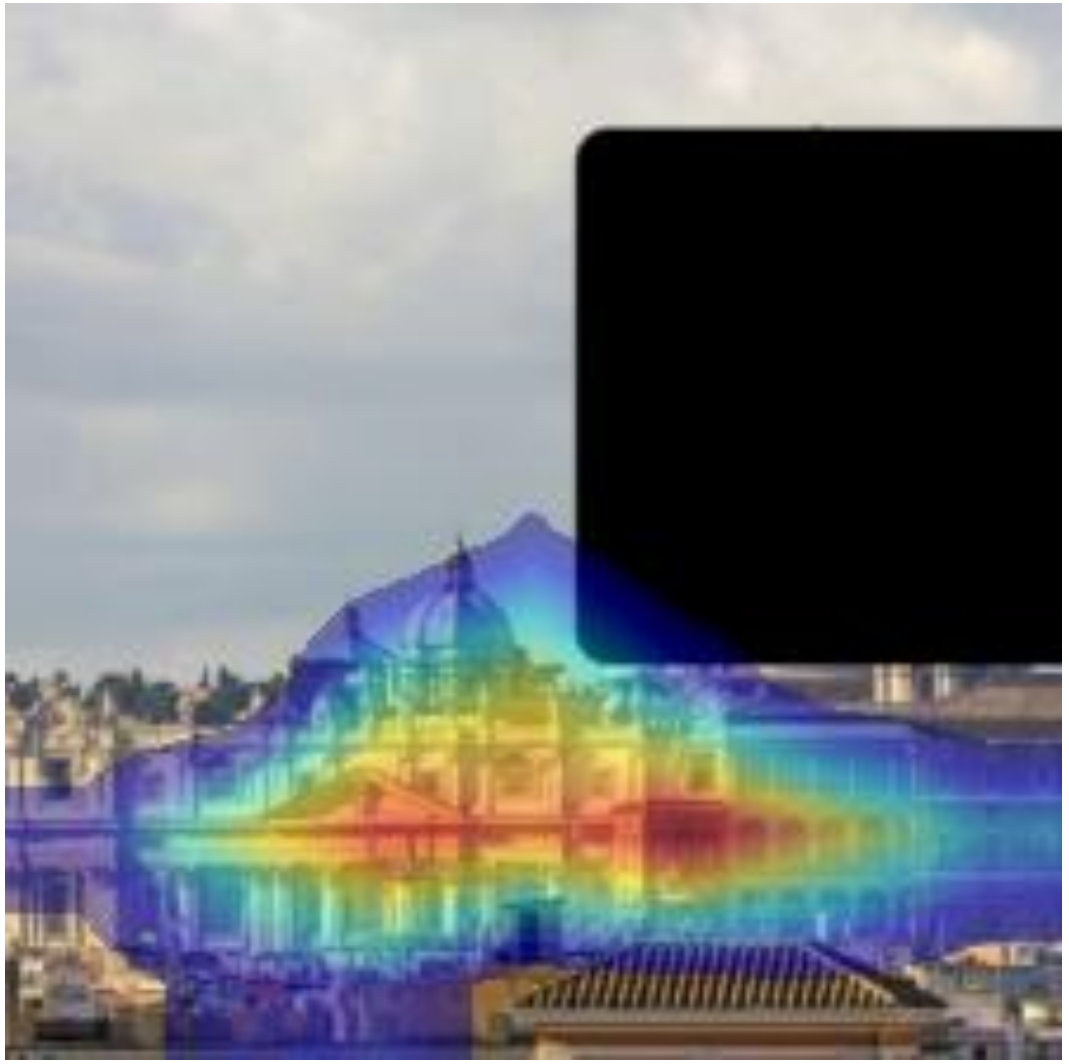
5. Code Results

BrAIn Lab.
Brain and Artificial Intelligence Lab.

5. Code Results

BrAIn Lab.
Brain and Artificial Intelligence Lab.



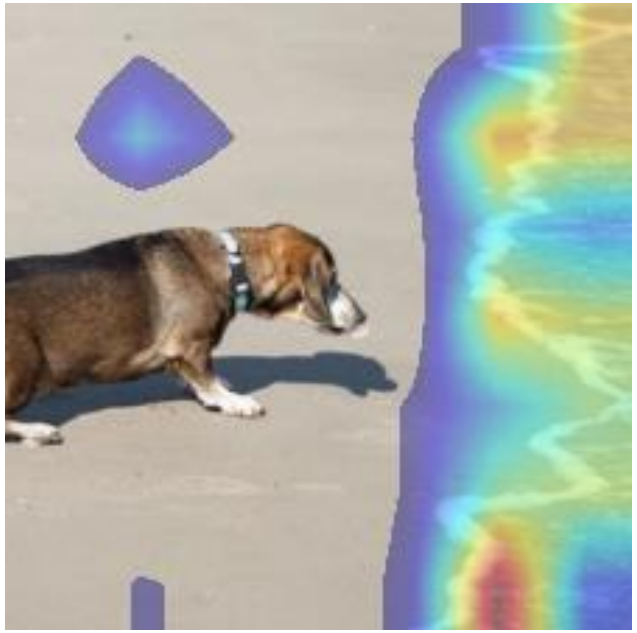


5. Code Results

BrAIn Lab.
Brain and Artificial Intelligence Lab.

5. Code Results

BrAIn Lab.
Brain and Artificial Intelligence Lab.

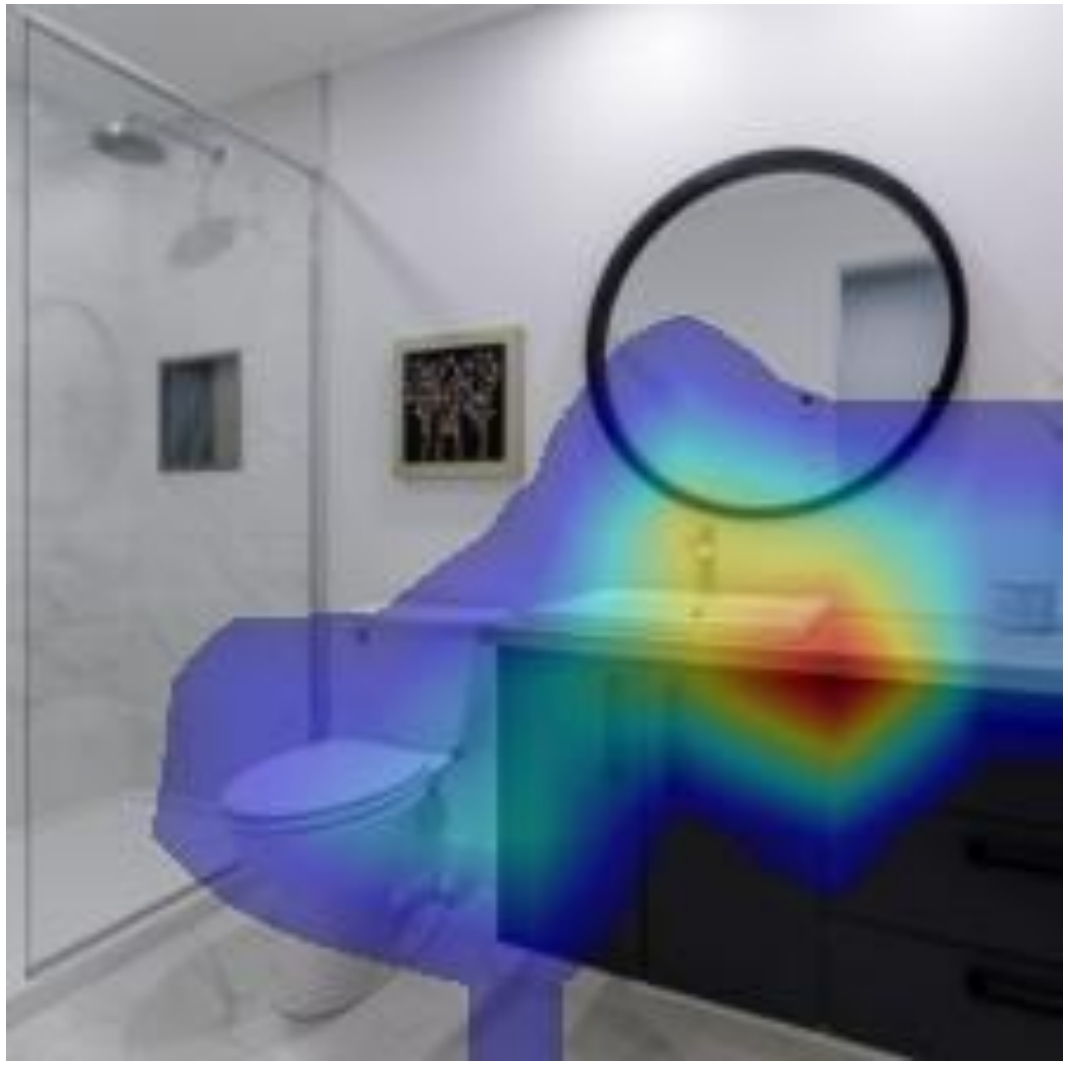


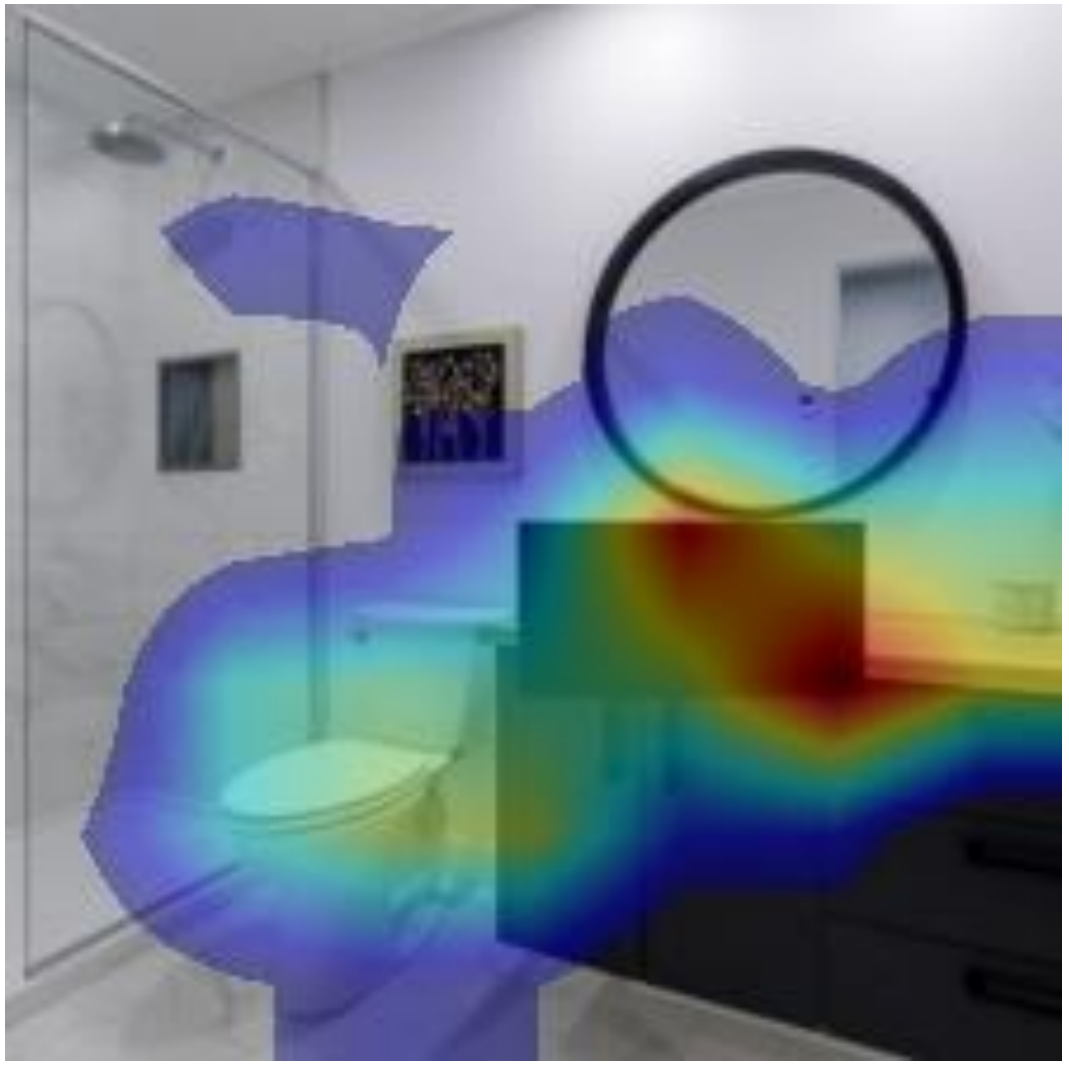
5. Code Results

BrAIn Lab.
Brain and Artificial Intelligence Lab.

5. Code Results

BrAIn Lab.
Brain and Artificial Intelligence Lab.





6. Code - points to consider

BrAIn Lab.
Brain and Artificial Intelligence Lab.

```python
# 모델이 예측을 수행하고 Grad-CAM 히트맵을 생성하는 데 사용한 데이터는 원본 이미지가 아니라 전처리(transforms)가 모두 완료된 input_tensor이기 때문에,
# input_tensor를 역으로 변환하여 사용하면, 모델이 본 정확히 그 224x224 영역의 이미지 위에 히트맵을 완벽하게 겹칠거라 판단
def tensor_to_image(tensor):
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    """
    모델이 보는 이미지 (정규화된 텐서):
    transforms.Normalize를 거친 input_tensor의 픽셀 값은 우리가 흔히 아는 0~255 범위가 아니고,
    평균이 0, 표준편차가 1에 가깝게 분포하며, 음수 값도 포함된 특이한 숫자들의 집합이다.
    컴퓨터(모델)는 이 상태의 데이터를 학습하고 예측하는 데 익숙하다.

    사람이 보는 이미지:
    우리의 눈과 모니터는 픽셀 값이 0(검은색)~255(흰색/컬러) 범위에 있는 이미지를 정상적으로 보여준다.
    만약 정규화된 텐서를 그대로 이미지로 표시하려고 하면, 색이 완전히 틀어지고 이상하게 보일 것이다.

    역정규화 후 유지되는 것:
    이미지 크기: 224x224 사이즈가 그대로 유지된다.
    잘린 영역 (Content): CenterCrop으로 잘려나간 정확히 그 이미지 영역이 그대로 유지되며, 이미지 속 고양이의 귀 모양, 건물의 창문 위치 등 모든 공간적 정보가 동일하다.
    """
    img_np = tensor.squeeze(0).cpu().numpy() # 배치 차원 제거
    img_np = np.transpose(img_np, (1, 2, 0)) # (C, H, W) -> (H, W, C)
    img_np = std * img_np + mean
    # img_np는 이론적으로는 0.0 ~ 1.0 사이의 값을 가져야 하지만 부동 소수점 연산 시 발생하는 미세한 오차 때문에 간혹 -0.0001처럼 0보다 약간 작거나 1.0002처럼 1보다 약간 큰 값이 생길 수 있다.
    # np.clip(array, min, max) 함수는 배열의 모든 값들을 확인해서 min 값보다 작으면 min 값으로, max 값보다 크면 max 값으로 만든다.
    img_np = np.clip(img_np, 0, 1)
    # JPG, PNG 같은 이미지 파일 형식은 각 색상 채널을 0부터 255까지의 정수로 표현하기 때문에,
    # 0 ~ 255 범위의 값으로 스케일링 후 스케일링된 소수점 값들을 uint8 자료형으로 변환 ex. 127.5 -> 127
    img_np = (img_np * 255).astype(np.uint8)
    return img_np
```

7. Recent Advances

BrAIn Lab.
Brain and Artificial Intelligence Lab.