6️⃣

CNN Visualization

강의 소개

Black box 모델인 **CNN의 내부 동작을 가시화**하는 방법들에 대해 설명합니다.CNN의 내부 동작을 확인하기 위해 **모델의 결정 이유를 추론**해내거나 **데이터가 처리되는 과정**을 살펴보는 방법들이 있습니다. 성능이 낮은 모델을 개선하는 데에도 사용할 수 있음은 물론, 높은 성능에 신뢰도를 더하여 건전한 개발을 도울 수 있습니다.딥러닝 모델의 특성을 더 깊이 알아내기 위해 적용했던 직관들과 결과 예시 사진들을 통해, 알고리즘 자체와 처리된 데이터들의 특성을 파악할 수 있는 능력을 기르는 시간이 되면 좋겠습니다. 분명 더 나은 모델을 개발하는 데에 큰 밑거름이 될 거라 믿습니다.

Further Question

(1) 왜 filter visualization에서 주로 첫번째 convolutional layer를 목표로 할까요?

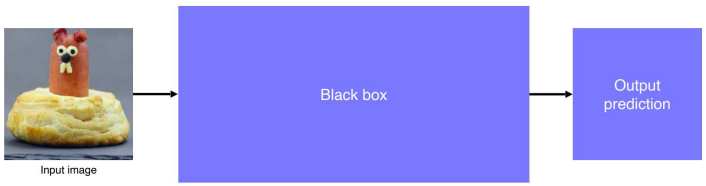
(2) Occlusion map에서 heatmap이 의미하는 바가 무엇인가요?

(3) Grad-CAM에서 linear combination의 결과를 ReLU layer를 거치는 이유가 무엇인가요?

Visualizing CNN

What is CNN visualization?

CNN is a black box



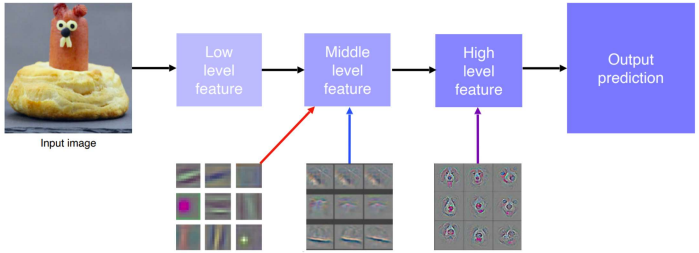
1. What is inside CNNs (black box)?

2. Why do they perform so well?
3. How would they be improved

ZFNet example

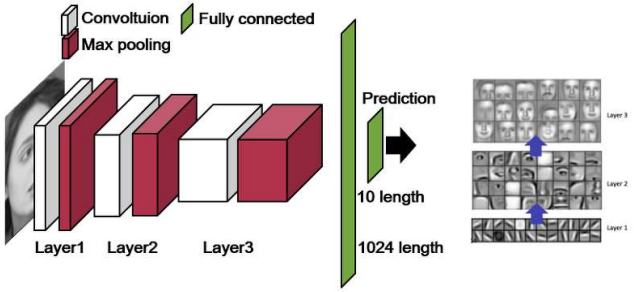
▼ 📎 ZFNet

- 읽어보기 : [Part V. Best CNN Architecture] 4. ZFNet [1] - 라온피플 머신러닝 아카데미 - : 네이버 블로그 (naver.com)



- deconvolution (convolution의 역연산)을 이용해서 각 CNN layer을 시각화

▼ 📎 deconvolution



- 위의 CNN구조를 보자, 예를 들어 레이어3에서 어느 한 필터에 대응하는 feature map의 어떤 값이 상당히 크다(한개의 값). 이때 이 필터의 역할을 어떻게 알 수 있을까? 우리는 위에서 deconvolution이 convolution의 반대과정이라고 이야기했다. 그렇다면 이 deconvolution을 이용하여 feature map의 값이

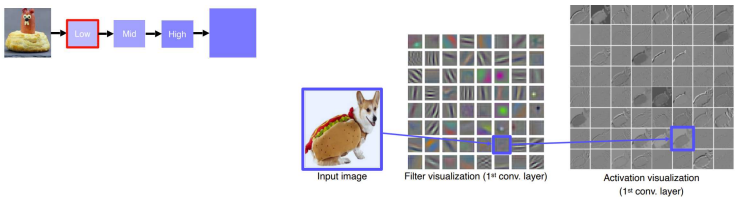
계산되기까지 거쳐왔던 길을 역추적하여 처음의 인풋이미지까지 돌아간다면, 우리는 **인풋이미지의 어떤 부분이 이 feature map을 크게 만들었는지(자극시켰는지) 알 수 있다**.

- 그림1의 오른쪽 부분의 patch들이 바로 그 결과이다, 레이어2에서 어떤 값을 역추적했더니 눈이 나왔다는건 바로 사람의 얼굴에서 눈이 이 필터를 잘자극시킨다는 것이고, 이 필터는 눈의 특징을 잡아내는 역할을 한다는 것을 의미한다.
- **우리는 deconvolution을 통해 feature map을 역추적하면 특정 필터가 처음의 인풋이미지에서 담당하는 부분을 시각적으로 알 수 있다는 것을 알았다.** 하지만 컨볼루션은 기본적으로 적분이 들어가는 연산이기에 g에 대한 정보가 없는 상태에서는 그 역연산이 쉽지가 않다(해가 하나가 아닌 경우도 많아서 보통은 estimation을 한다). 게다가 image 같이 차원이 크고 갯수까지 많은 데이터에 대해서 매번 deconvolution을 해주는건 연산량이 매우 부담스럽다. 그래서 **딥러닝에서의 deconvolution은 정확한 연산을 한다기보다 그 개념만을 취한다.**
- 참고 : [논문리뷰] CNN에서의 Deconvolution 이해하기 [1] (tistory.com)

- low level → high level이 될 수록 더 유의미한 패턴

Vanilla example: filter visualization

Filter weight visualization



- color / 각도 / 방향 / edge 등을 학습
- Activation visualization → activation map 처리 결과가 한 채널로 나오므로, 흑백

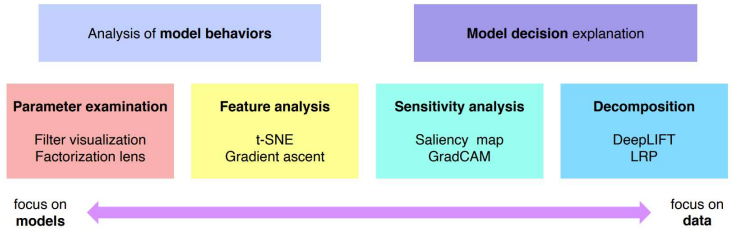
▼ 📎 Activation visualization

- 특정 이미지를 입력했을 때의 **각 feature map이 '활성화된(activation)' 정도**를 보여준다는 맥락에서, 이러한 해석 방법을 보통 'Activation Visualization(활성값 시각화)'이라고 부릅니다
- **Activation Visualization**을 통해, 여러분들은 학습이 완료된 컨볼루션 신경망의 각 layer에서 어떤 feature map이 이미지 상의 어떠한 특징들을 커버하고 있는지 추측할 수 있습니다.
- 이를테면 사람 또는 동물의 얼굴이 포함되어 있는 이미지들을 입력할 때, **공통적으로 최대로 활성화되는 feature map이 무엇인지 조사**하고, 이들을 시각화함으로써 **해당 feature map이 실제로 '얼굴'을 커버하는 경향을 보이는지 확인**할 수 있습니다.
- 참고 : Interpretable Machine Learning 개요: (2) 이미지 인식 문제에서의 딥러닝 모델의 주요 해석 방법 - 블로그 | 코그넥스 (cognex.com)

- Further question (1) 왜 filter visualization에서 주로 첫번째 convolutional layer를 목표로 할까요?
  - 뒤쪽 layer는 filter의 차원 수 ↑ (∵ ex. channel 개수 5, 6 ..) → color로 visualization 불가능 ⇒ 사람의 해석 불가능 + 앞쪽 layer와 합성해서 더 추상적

How to visualize neural network

Types of neural network visualization

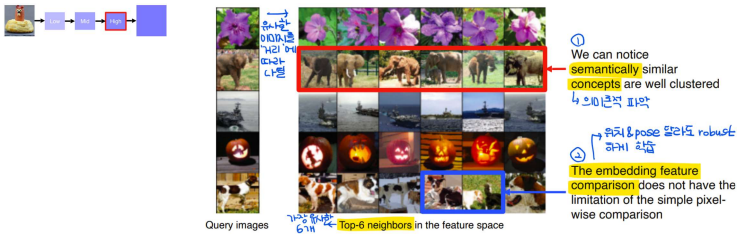


- focus on models → 모델 이해
- focus on data → 데이터 결과 분석

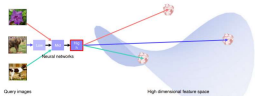
Analysis of model behaviors

Embedding feature analysis

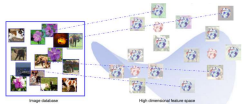
Nearest neighbors (NN) in a feature space



- simple pixel-wise comparison 으로 파악 불가능 ($||image_1\text{-}image_2||_2$ 거리로 측정 불가)한 위치/pose가 다른 이미지도 유사하다고 파악 ⇒ 위치/pose에 robust하게 학습



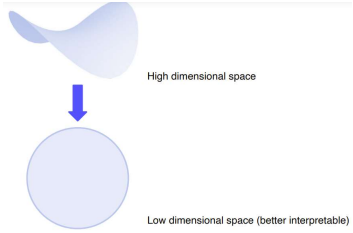
1️⃣ Query image의 feature를 미리 추출



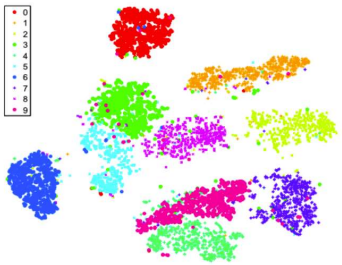
2️⃣ Search the nearest neighbors of features from DB

3️⃣ Analysis by searched neighbor images

Dimensionality reduction



- CNN backbone 거쳐서 나온 feature ⇒ 고차원



Visualization on geometry of feature space
(experiment with MNIST dataset)

- 차원을 낮춰서, 사람이 해석할 수 있도록

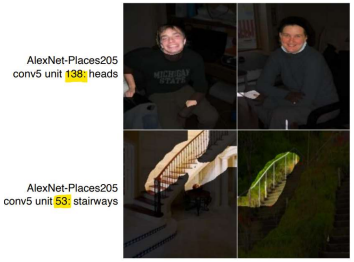
- **t-distributed stochastic neighbor embedding (t-SNE)**
  - ex. 3, 5, 8 → 세 클래스의 분포가 맞닿아 있음
  - ex. 클래스끼리 잘 구분되어 있음
  - ▼ t-SNE
    - `t-SNE`는 매니폴드 학습의 하나로 복잡한 데이터의 시각화가 목적입니다. 높은 차원의 데이터를 2차원 또는 3차원으로 축소시켜 시각화 합니다.
    - `t-SNE`를 사용하면 높은 차원 공간에서 **비슷한 데이터 구조는 낮은 차원 공간에서 가깝게 대응**하며, **비슷하지 않은 데이터 구조는 멀리 떨어져 대응**됩니다.
    - 참고 : t-SNE 개념과 사용법 - gaussian37

Activation investigation

Layer activation

- Behaviors of mid- to high-level hidden units

- layer의 activation을 보고 model 특징 파악
  - 138번째 channel은 얼굴을 찾는 node구나 → 모델 특징 파악



High activation mask visualization
(The masks are generated by thresholding)

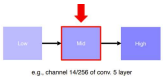
- 53번째 channel은 계단을 찾는 node구나

Maximally activating patches

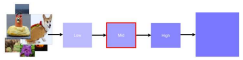
- hidden layer의 한 채널에서, 가장 큰 activation 값을 갖는 patch를 뜯음 (각 데이터 마다) + 나열

Image patches obtained from
the locations of maximum activations of each hidden node

- **Patch acquisition**



- 1) **Pick** a channel in a **certain layer**

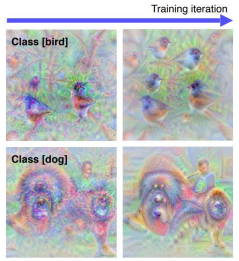


- 2) Feed a chunk of images and record **each activation map** (of the chosen channel)

- 3) **Crop image patches** around maximum activation values

Class visualization

- network가 기억하는 이미지를 시각화
  - ex. 새 뿐만 아니라 나뭇가지도 시각화 → 학습 데이터가 bias 되어 있음을 확인 가능
  - 최종적인 결과를 보고, CNN에 대한 해석



Visualized classes

- Gradient ascent
  - Generate **a synthetic image** that triggers maximal class activation
    - I → input image
    - f → CNN model ⇒ f(I) → 해당 클래스의 score
    - argmax f(I) → 클래스 score를 최대화하는 I 찾기
    - Regularization term → 0~255의 범위를 벗어나는 I를 찾을 수도 있음 → 사람이 해석할 수 있는 범위내로 제한 ⇒ L_2 norm 으로 구현
  - loss를 최대화하므로, gradient ascent
    - ▼ 📎 gradient ascent
      - Gradient Ascent는 말 그대로 해당 모델에서 loss가 최대가 되는 parameter를 찾는 방법입니다. (backprop 과정에서 gradient ascent를 사용한다는 의미는 아닙니다.)
      - Gradient Ascent가 Deep learning visualization에서 중요한 역할을 차지하는 이유는, 이 방법을 통해 **중요한 feature들을 확대하여 (부각하여) 보여줄 수 있기 때문**입니다. Gradient Ascent를 통해 **뉴런을 최대로 활성화** 시키는 합성 이미지를 만들어낼 수 있고, 이것이 해당 레이어에서 일어나는 일을 시각화하는데에 단서를 제공하게 됩니다.
      - Input image에 Gradient Ascent를 적용함으로써, **특정한 필터의 response를 극대화할 수 있고, 이를 통해 특정 필터에서 무슨 일이 일어나고 있는지를 시각화**할 수 있습니다.
      - 참고 : Gradient Ascent (tistory.com)

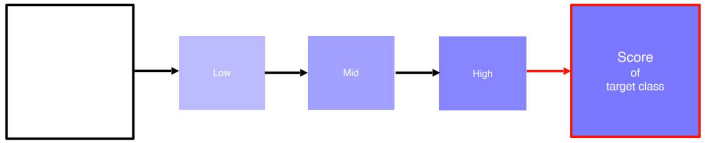
$$I^* = \arg\max_I f(I) - \boxed{Reg(I)}$$

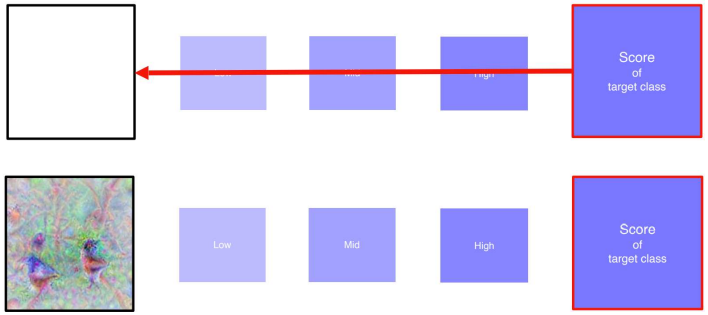
Regularization term

$$I^* = \arg\max_I f(I) - \boxed{\lambda ||I||_2^2}$$

Regularization term

- Image synthesis
  - 1) Get **a prediction score (of the target class)** of a dummy image **(blank or random initial)**
    - ▼ 📎 blank or random inital ?
      - 특정 이미지를 분류하는 문제가 아니라, **레이어의 feature들을 시각화하는 것이 목적이기 때문에 처음에 이미지를 input으로 넣는 것보다 zero값을 넣어주는 것이 효과적**입니다. 이 zero 초기화 된 이미지는 나중에 Gradient Ascent를 통해 점차 특정 필터가 나타내는 이미지 값으로 바뀌게 됩니다.
      - 참고 : Gradient Ascent (tistory.com)
  - 2) **Backpropagate** the gradient **maximizing the target class score** w.r.t. the input image
    - ▼ 📎 gradient descent VS. gradient ascent
      - backpropagate에서는 gradient descent를 사용 ↔ input image를 update 할 때, gradient ascent 사용
      - 참고 : Gradient Ascent (tistory.com)
  - 3) **Update the current image**
  - 위의 과정을 여러 번 반복 + 눈으로 보면서 패턴 파악





Model decision explanation

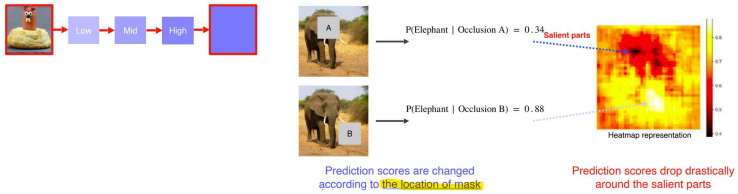
Saliency test

▼ 📎 saliency

- **visual saliency 모델**은 이미지 내에서 시각적으로 중요한 부분들이 어딘지 또한 얼마나 중요한지를 예측해주는 알고리즘이다.
- 참고 : [Visual saliency] 중요 물체 검출(salient object detection, SOD)이란? by bskyvision

Occlusion map

- patch로 이미지를 가림
- 물체의 중요 영역을 가리면, score가 떨어짐
- Further question (2) Occlusion map에서 heatmap이 의미하는 바가 무엇인가요?
  - 각 occlusion을 모두 실험해보고 score를 map으로 표현



via Backpropagation

- random or black image 대신 특정 이미지로 classification
- 최종 class를 결정할 때 영향을 미친 영역이 어디인지 heatmap으로 표현
  - 밝은 부분 → 관심 영역
- gradient ascent 사용

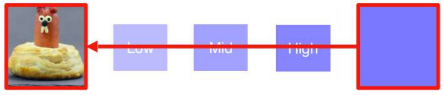
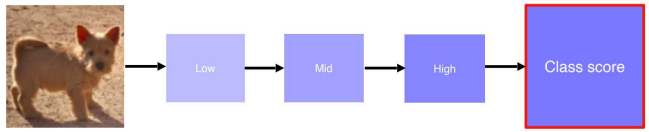
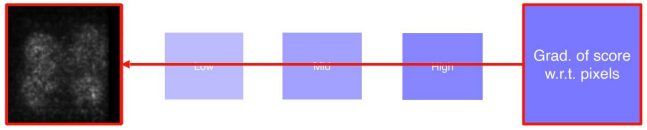


Image-specific class saliency maps

- Derivatives of a **class score** w.r.t. input domain
  - 1) Get a class score of the **target source image**
  - 2) **Backpropagate** the gradient of the class score w.r.t. input domain
  - 3) Visualize the **obtained gradient magnitude map** (optionally, can be accumulated)

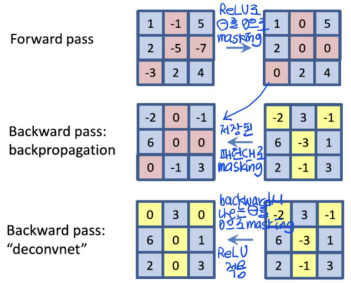




- 특정 input image를 넣고 class score 구함 → 특정 '데이터'에 대한 모델의 '결과' 확인
  - cf. random or black image를 넣고 class score 구함 → 입력에 대한 '모델의 반응' 확인
- backpropagation으로 input layer까지 gradient 계산 + 절댓값 or 제곱 ⇒ 절대적인 **'크기'** 구하기 → 이미지 시각화 ⇒ **gradient 변화 정도** 확인

Backpropagate features

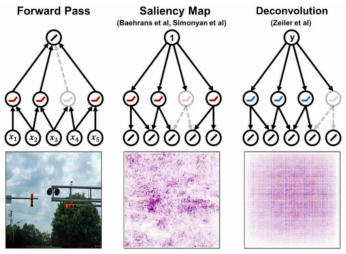
Rectified unit (backward pass)



$$h^{l+1} = max(0, h^l)$$

$$\frac{\partial L}{\partial h^l} = [(h^l > 0)] \frac{\partial L}{\partial h^{l+1}}$$

$$\frac{\partial L}{\partial h^l} = [(h^{l+1} > 0)] \frac{\partial L}{\partial h^{l+1}}$$

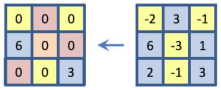


▼ backward pass

- `Gradient-based Backpropagation`은 **relu outputs이 양수인 부분에만 gradients를 구하는** 방식
- `Gradients-based Deconvnet`은 **gradients가 0보다 큰 gradients를 구하는 방식**
- 반면에, `gradients-based guided Backpropagation`은 특정 class(확률이 가장 높은)에 대해서
  - **1) gradients(w.r.t last conv)가 0보다 크며**
  - **2) relu outputs이 양수인 부분인 gradients**만 feature map에 곱하는 방식 (weighted-average)
- 참고 : Guided Grad-CAM (donghwa-kim.github.io)

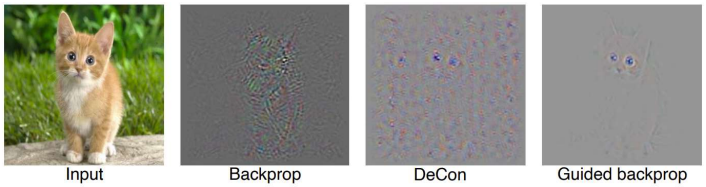
Guided backpropagation

Backward pass: *guided backpropagation*



$$\frac{\partial L}{\partial h^l} = [(h^l > 0) \& (h^{l+1} > 0)] \frac{\partial L}{\partial h^{l+1}}$$

- Further question (3) Grad-CAM에서 linear combination의 결과를 ReLU layer를 거치는 이유가 무엇인가요?
  - 두 방법을 혼합 → forward에서도 ReLU + backward에서도 ReLU ⇒ class 결정에 도움이 되는 부분만 양수(+)로 추출

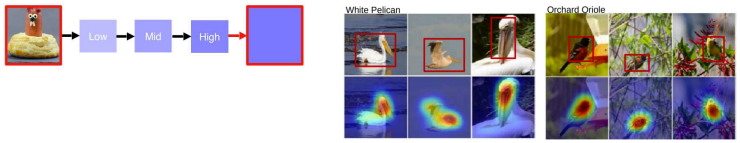


- 어떤 영역을 보고 input image를 고양이로 인식? → Guided backprop이 가장 clear (귀 + 얼굴 영역 확인)

Class activation mapping (CAM)

Class activation mapping (CAM)

- **Visualize** which part of image **contributes** to the final decision

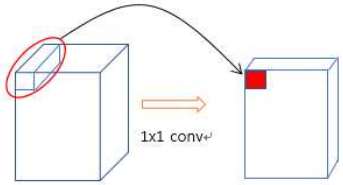


- threshold를 잘 설정할 경우, bounding box도 구할 수 있음 (?)
- **Global average pooling (GAP) layer** instead of the FC layer

▼ 📎 FCN + GAP

- **FCN**

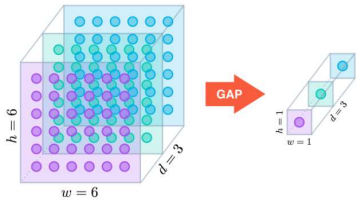
  - **FC layer에서 위치 정보를 담고있던 conv feature map이 dense하게 쫙 펴지기 때문에 위치정보를 모두 손실**하고 만다. 따라서 이 문제점을 해결하고자, 위치정보를 보존하기 위해 Fully convolutional network가 나오게 되었다. FCN은 **1x1 conv filter를 사용함으로써 위치정보가 손실되는 것을 막아준다.**
  - 예를들어, **HxWxN**(H:height, W:width, N:filter #)의 feature map이 있다고 가정하자. 여기에 1x1 conv **filter K개**를 사용하면 **HxWxK**의 output이 나오게 된다.



  - 두번째 장점으로는 위 그림과 같이 HxW. 즉, 이미지의 크기 자체는 그대로 보존하기 때문에 filter의 개수에 따라서 output의 dimension이 달라진다. **"따라서 feature map의 크기를 유지하면서 dimension을 줄여주고 싶을 때 1x1 conv를 자주 사용한다."**

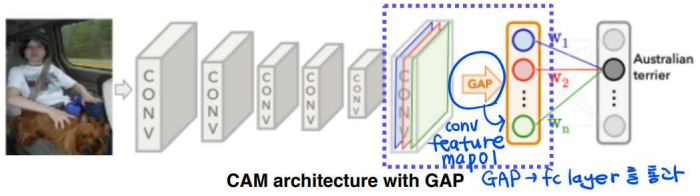
- **GAP**
  - 위에서 설명한대로 FC layer 대신 FCN을 사용한다고 해도 softmax를 통해 **classification작업을 최종적으로 수행해주는 FC layer의 기능**을 대신해주지는 못한다. 예를 들어 설명해보겠다.
  - 10개의 label이 있는 classification을 진행한다고 가정하자. 이때 CNN모듈을 통과한 후 마지막에는 FC layer를 통해 conv feature map을 10개의 node로 수렴시키기 위해 feature map을 dense하게 펼치고 10개의 node와 연결해준다.
  - 예를들어, 이때 FCN을 이용하면 FC layer에 의해 무작정 HxWxN의 feature map을 10개의 node로 수렴시키기 대신 **위치정보를 유지하여 HxWx10의 output**을 산출한다.
  - **그러나 이는 classification을 하기위해 다시 1x1x10, 즉 10개의 node로 resize해줘야 한다.**

- FCN은 단지 filter의 dimension만 줄여줄뿐 height, width는 그대로 유지하기에 위치정보를 보존하는 것이기 때문이다.
- FCN으로 위치정보를 유지한채 dimension을 줄이고 난 후 **FC layer의 기능을 수행**하기 위해 **GAP를 사용**한다.
- GAP는 conv feature map에서 하나의 filter내에 있는 모든 pixel값을 average pooling하여 1개의 value로 수렴시킨다.
- 따라서 **GAP에서는 FCN과 반대로 위치정보를 유지하지 못하지만** filter 사이즈가 어떻든 1개의 value로 수렴시키므로 **"GAP를 사용하면 FC layer와 달리 input size에 유동적이다"**
- GAP는 HxW의 픽셀을 1개의 value로 수렴시키기 때문에 filter의 개수만 고정시키고 **input size는 유동적이게 할 수 있다**.



- **요약**
  - FCN : 위치정보를 그대로 유지하면서 dimension, 즉 filter의 개수에 해당하는 차원을 줄일 수가 있다. HxWxN의 feature map에 1x1 conv filter K개를 사용하면 HxWxK의 feature map으로 만들어 줄 수 있다. layer가 깊어져서 parameter의 개수가 급증할때 gradient vanishing 현상을 예방하기 위해 FCN을 사용한다.
  - GAP : FC layer를 사용하려면 아래 그림과 같이 dense하게 펼쳐진 node들과 모두 연결이 되어야한다. 이렇게 feature map을 dense하게 펼친 후 FC layer와 연결하게 되므로 FC layer의 input size는 고정이여야 한다. 그러나 GAP를 사용하면 HxW의 size인 1개의 feature map을 1개의 value로 수렴시키므로 filter의 개수만 고정일뿐 input size는 유동적이다. 요즘에는 FC layer의 위와 같은 한계를 극복하기 위해 GAP로 대체시킨다.
- 참고 : GAP(Global Average Pooling) vs FCN(Fully Convolutional Network) :: 프라이데이 (tistory.com)

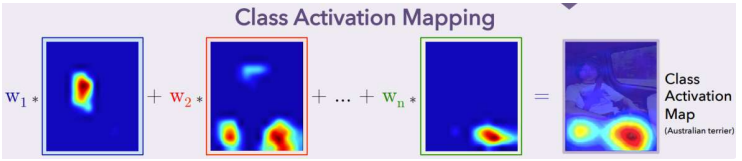
- CAM은 Conv Layer을 Fc-Layer로 납작하게 하지 않고, **GAP을 통해 새로운 Weight값을 만들어 내고 있는 것입니다.** 마지막 Conv Layer가 총 n개의 channel로 이루어져 있다면 각각의 채널들은 GAP에 의해 하나의 Weight 값으로 나타나게 되며, 총 n개의 Weight들이 생기게 됩니다.
- 마지막 Softmax 함수로 인해 연결되어 Weight들도 백프롭을 통해 학습시키는 것입니다. **N 개의 Weight**가 생겼다면 **CAM은 이 Weight들과 마지막 n개의 Conv Layer들과 Weighted Sum을 하여 하나의 특정 분류의 이미지의 히트맵**이 나오게 합니다.
- 참고 : Grad CAM 에 대한 이해 (hygradcam.blogspot.com)



$$S_c = \sum_k w_k^c F_k \quad (F_k = \text{GAP feature})$$

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

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



- 위치 정보를 따로 input으로 제공하지 않았는데도 찾음
- threshold 주고 + bounding box를 만들면 ⇒ 자동으로 object detection

- object detection처럼 정교한 task를 rough한 영상인식 task로 간접 해결 ⇒ weakly supervised learning

▼ 📎 weakly supervised learning

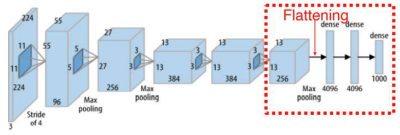
- `Weakly Supervised Learning`은 학습에있어 필수적인 Label을 최대한 사용하지 않는 방법이 없을까? 하는 의문에서 시작된 연구입니다. 따라서, 이미지 분할이나 객체 탐지를 할 때, 상대적으로 약한(Weakly) 정보로 분할이나 탐지를 해보자는 아이디어 입니다.
- 여기서 **약한 정보**라는 것은 이미지 분할이나 객체 탐지를 할 때, 기존에 주어지던 분할 **Mask(Label) 또는 Bounding box는 상대적으로 강한 정보**이고, **Class에 대한 Label 정보는 약한정보**라고 할 수 있습니다.
- 실제로 픽셀별로 분할을 하거나 바운딩박스를 치는 것은 단순히 해당 이미지에 어떤 물체의 Class가 있다라는 라벨링보다 비용이 훨씬 많이 듭니다. 따라서, 이미지 분할 또는 **객체 탐지 시에 Class에 대한 정보(or +위치 정보)만을 이용하자는 것**입니다.
- 참고 : [논문 리뷰 및 코드구현] Simple Does It: Weakly Supervised Instance and Semantic Segmentation :: Barcelona (tistory.com)

▼ 📎 수식

- `GAP(Global Average Pooling)` 방식을 적용하면 위의 그림에서와 같이 **각 채널에서 색 별로 average 하나의 값만을 가져오게 됩니다.** 그 이후에 `FC`를 적용하는데 이때의 weight값이 **CAM을 만드는 w1, w2, w3, ... 값**이 됩니다. FC의 weight는 [n_Ch, n_classes]의 크기를 갖는 matrix인데, FC matrix의 클래스 index 열 벡터가 각각 w1, w2, ...,wn가 됩니다.(*n_classes는 분류하려는 객체의 수) 이 **weight들을 마지막 feature map에 각각 곱한 후에 더해주면 CAM이 됩니다.**
- 이를 수식으로 더 자세히 표현해봅시다. 마지막 Conv layer를 거쳐서 얻은 feature map을 `fk` 라고 하고, **k번째 채널의 값**들 중 (x,y)에 위치한 값을 `fk(x,y)` 라고 표현합니다. 그렇다면 **GAP를 거쳐 그림에서 각 원 하나에 해당**하는 것, `Fk` 는 ∑fk(x,y)가 됩니다. 이를 **softmax에 집어넣기 위해 FC를 하나 추가**해주고, 이때의 **weight**들을 w_k^c 라고 합니다. 이것은 본질적으로 Fk의 중요성을 나타내며 그 크기가 클수록 c에서 Fk가 미치는 영향이 커지게 됩니다. class c에 대해서 **softmax에 입력으로 주어지는 값**, `Sc`는 $\sum w_k^c F_k$가 됩니다. 식을 조금 더 변형하면

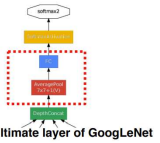
$$\begin{aligned} S_c &= \sum_k w_k^c F_k \\ &= \sum_k w_c^k \sum_{x,y} f_k(x,y) \\ &= \sum_{x,y} \sum_k w_k^c f_k(x,y) \end{aligned}$$

- 참고 : Grad CAM 에 대한 이해 (hygradcam.blogspot.com)



AlexNet architecture

To be modified to CAM, the flattening layer should be changed to GAP



The penultimate layer of GoogLeNet

- Requires **a modification of the network architecture** and **re-training** (단점)
  - parameter를 모두 다시 학습해야 ⇒ (CAM으로 모델 구조를 바꾸기 이전과) 성능이 바뀔수도
  - ResNet and GoogLeNet already have **the GAP layer**

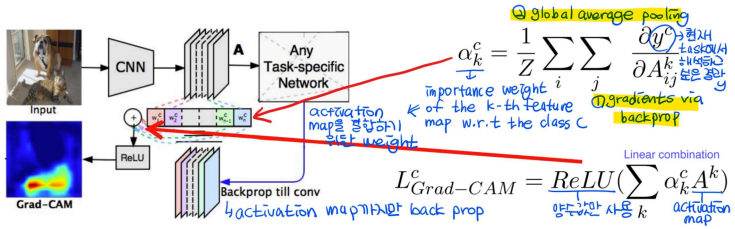
Grad-CAM

- Get the CAM result **without modifying and re-training** the original network

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

Key idea:

How to obtain the importance weights



- backprop에서 saliency 유도한 방법 응용 → ReLU 사용

▼ 수식

- 첫 번째 식(α_k^c)에서 `A`는 Softmax의 전단계 k 번째 클래스에 대한 특징 맵을 뜻하고, `y`는 각 클래스에 대한 Score를 뜻합니다. 결국 첫 번째 식이 의미하는 것은 **특징 맵 A에 대한 y의 gradient를 전부 합산한 값이 a**이고 이는 **CAM에서의 global average pooling을 한 값과 동일**하다는 것입니다.
- 두 번째 식($L^c_{Grad-CAM}$)을 보면 CAM과 동일하게 모든 클래스에 대한 a와 특징 맵을 곱한 뒤 더하면 Grad-CAM을 구할 수 있습니다. 여기서 **활성화 함수인 ReLU가 들어가는데 이는 클래스의 interest에서 양의 값에만 관심이 있기 때문입니다.**
- 참고 : Grad CAM 에 대한 이해 (hygradcam.blogspot.com)

▼ CAM VS. Grad-CAM

CAM:

$$L^c_{CAM} = \sum_k w_k^c A^k$$

Grad-CAM:

$$L^c_{Grad-CAM} = ReLU(\sum_k a_k^c A^k)$$

- ReLu를 제외하고 CAM과 Grad-CAM의 수식을 비교해 보면 w와 a의 차이에서 기인합니다.
- 우선 `CAM`의 w는 **GAP에서 얻을 수 있는 Softmax layer의 weight값**이고, `Grad-CAM`의 a는 **Softmax 함수의 input값의 피쳐맵에 대한 편미분값**을 구해 **GAP방식으로 출력**한 결과입니다.
- 하지만 아래의 수식을 보면 Grad-CAM의 **a는 CAM의 w와 동일한 값을 얻을 수 있습니다.** 결국에는 Grad-CAM의 수식은 CAM에서 얻은 수식을 일반화 시킨 값으로

GAP을 사용하지 않는 네트워크 모델에도 CAM과 동일한 결과가 나오도록 적용 할 수 있는 장점이 있습니다.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{global\,average\,pooling} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{gradients\,via\,backprop}$$

$$L^c_{Grad-CAM} = ReLU\ \underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{linear\,combination}$$

$$y^c = \sum_k \underbrace{w_k^c}_{class\,feature\,weights} \overbrace{\frac{1}{Z}\sum_i\sum_j}^{global\,average\,pooling} \underbrace{A_{ij}^k}_{feature\,map}$$

$$y^c = \frac{1}{Z}\sum_i\sum_j\sum_k \underbrace{w_k^c A_{ij}^k}_{L^c_{CAM}}$$

$$F_k = \frac{1}{Z}\sum_i\sum_j A_{ij}^k$$

$$y^c = \sum_k w_k^c F_k$$

$$\frac{\partial y^c}{\partial F_k} = \frac{\frac{\partial y^c}{\partial A_{ij}^k}}{\frac{\partial F_k}{\partial A_{ij}^k}}$$

$$\frac{\partial F_k}{\partial A_{ij}^k} = \frac{1}{Z}$$

$$\frac{\partial y^c}{\partial F_k} = w_k^c$$

$$\frac{\partial y^c}{\partial F_k} = w_k^c = Z \cdot \frac{\partial y^c}{\partial A_{ij}^k}$$

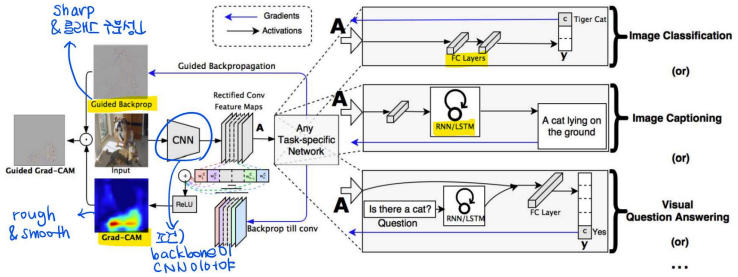
$$\sum_i\sum_j w_k^c = Z \cdot \sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

$$Z \cdot w_k^c = Z \cdot \sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

$$w_k^c = \sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

$$\therefore w_k^c = Z \cdot \alpha_k^c$$

- 참고 : Grad CAM 에 대한 이해 (hygradcam.blogspot.com)

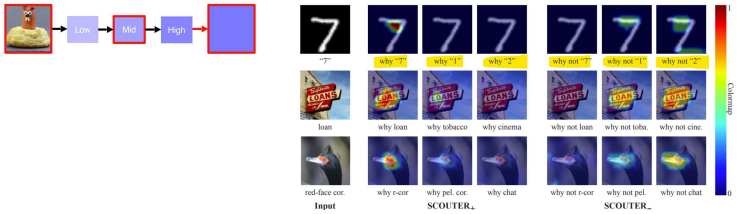


▼ 📎 guided backprop → sharp + 클래스 구분성

- class 결정에 도움이 되는 부분만 양수(+)로 추출 → 가장 clear
- Backpropagate features - Guided backpropagation 항목 참고

SCOUTER

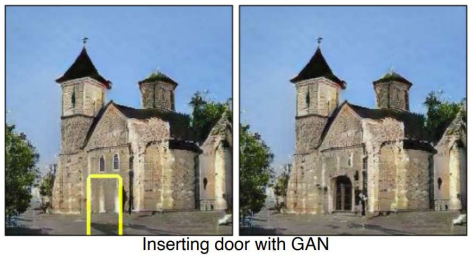
- Scouter tells "**why the image is of a certain category**" or "**why the image is not of a certain category.**"
  - Grad-CAM + 왜 다른 클래스는 정답이 아닌지도 밝힘



Conclusion

GAN dissection

- Spontaneously emerging **interpretable representation during training**
- Not only for **analysis** but also for **manipulation applications**



Inserting door with GAN

- 생성 모델의 어떤 hidden node가 어떤 부분에 기여하는지 해석 → 사용자가 수정 가능하도록
  - ex. user가 door 담당 채널을 masking → GAN이 문 생성

▼ 📎 gan dissection

- 읽어보기 : GAN Dissection 논문 리뷰 - GAN Dissection Visualizing and Understanding Generative Adversarial Networks | mocha's machine learning (ysbsb.github.io)
- 읽어보기 : GAN Dissection (mit.edu)

Reference

1. Visualizing CNN
   - Zeiler and Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
2. Analysis on model behavior
   - Krizhevsky et al., ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012
   - Maaten and Hinton, Visualizing Data using t-SNE, JMLR 2008
     - Bau et al., Network Dissection: Quantifying Interpretability of Deep Visual Representations, CVPR 2017
   - Springenberg et al., Striving for Simplicity: The All Convolutional Net, ICLR 2015
   - Goodfellow et al., Explaining and Harnessing Adversarial Examples, ICLR 2015
   - Szegedy et al., Intriguing properties of neural networks, CVPR 2014
3. Model decision explanation
   - Simonyan et al., Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, CoRR 2013
   - Li at el., Instance-Level Salient Object Segmentation, CVPR 2017
   - Kim et al., Why are Saliency Maps Noisy? Cause of and Solution to Noisy Saliency Maps, ICCV 2019
   - Zhou et al., Learning Deep Features for Discriminative Localization, CVPR 2016
   - Selvaraju et al., Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, ICCV 2017
   - Bau et al., GAN DISSECTION: VISUALIZING AND UNDERSTANDING GENERATIVE ADVERSARIAL NETWORKS, ICLR 2019
   - Li et al., SCOUTER: Slot Attention-based Classifier for Explainable Image Recognition, arXiv 2020