

# GradCAM final-report

ByeongGeun Shin<sup>1</sup>,

<sup>1</sup>Pusan National University.

## 1 Visual Explanations from Deep Networks via Gradient-Based Localization Paper Summary

### 1.1 Abstract

모델을 해석할 때에는 Simplicity와 Interpretability 사이의 tradeoff 관계가 있다. 즉, 모델이 간단할수록 해석은 용이해지고 모델이 복잡할수록 해석은 어려워지기 때문에 모델의 accuracy를 높지 않으면서 해석하기 위해서는 이 둘 사이의 적정점을 찾는 것이 중요하다.

기존의 CAM[1] 논문의 경우 layer에 GAP(Global Average Pooling)이 있어야 하고, FC layer 1개가 따라와야 한다는 단점이 있었다. 그러나 이런 구조적인 문제에서 벗어나 CAM의 method을 generalization한 모델을 제시한 논문이 Grad-CAM이다. 또한 높은 localization performance를 보이고 구조적인 한계가 없다는 점에서 크게 주목을 받았다.

### 1.2 Introduction

이전의 CAM에서는, CNN 모델의 가장 마지막 layer인 FC layer를 Global average pooling으로 대체하여 overfitting을 줄이고 학습되지 않은 task(weakly-supervised object localization)을 수행하여 시각화할 수 있다는 장점이 있었다. 하지만 GAP로 대체하는 과정이 결국 모델의 해석을 위해 complexity를 줄인 것이다. CAM은 마지막 FC layer를 포함하는 모델에만 한정적으로 적용할 수 있다는 단점이 있다. 이 논문에서 제시하는 Grad-CAM은, 모델의 구조나 complexity에 아무 변형을 주지 않고 모든 모델에 적용할 수 있는 알고리즘이다. 따라서 CAM의 일반화된 방법이라고 볼 수 있고, 다음과 같은 더 넓은 범주의 CNN 모델에 전부 적용할 수 있기 때문에 Image Classification, Localization, Captioning, VQA(Visual Question Answering) 등의 더 넓은 Task에 응용된다.

### 1.3 Method

#### 1.3.1 Gradient-weighted Class Activation Mapping

CAM에서 사용했던 Global average pooling에서는, 마지막 Convolution layer의 Feature map 각각에 대해 전역적인 평균값을 구한 후 각 unit에 대한 중요도를 의미하는 weight들을 사용하여 weighted sum을 하였다. 하지만 위에서 언급했던 문제점처럼, 이 weight 값들이 주어져있지 않으면 CAM을 사용할 수 없기 때문에 Global average pooling을 사용하는 CNN 모델에만 CAM을 적용할 수 있다는 문제점이 생긴다. 따라서 이 논문에서는 이 weight들을 각각의 class  $c$ 에 대한 gradient backpropagation으로 계산하여 대체하고자 한다.

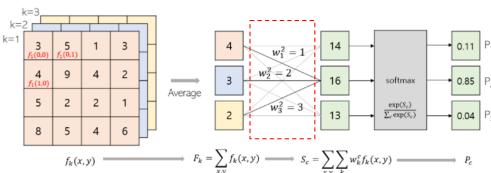


Figure 1: CAM Process.

#### 특정 클래스의 출력과 합성곱 특징 맵의 Gradient 계산

Grad-CAM의 주요 아이디어는 특정 클래스  $c$ 에 대한 모델의 출력  $y^c$ 의 변화가 합성곱 층의 특징 맵  $A^k$ 에 대해 어떻게 달라지는지를 나타내는 Gradient를 계산하고, 이를 이용하여 중요한 특징 맵 영역에 가중치를 부여하는 것이다.

$$y^c = f^c(x),$$

여기서  $f^c(x)$ 는 입력  $x$ 에 대해 모델의 클래스  $c$ 에 대한 출력이다. Grad-CAM은  $y^c$ 에 대해 합성곱 층  $A^k$ 의 Gradient를 계산한다.

$$\frac{\partial y^c}{\partial A^k},$$

여기서  $A^k$ 는 합성곱 층의  $k$ -번째 특징 맵이다.

#### Gradient를 통한 중요도 가중치 계산

특정 맵  $A^k$ 의 중요도  $\alpha_k^c$ 는 공간적 차원에 대해 평균을 취한 Gradient로 정의된다.

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k},$$

여기서  $Z$ 는 특정 맵의 차원 ( $i, j$ 에 걸친 총 요소 수)이다.

#### Class Activation Map (CAM) 생성

Grad-CAM의 클래스 활성화 맵  $L_{\text{Grad-CAM}}^c$ 은 각 특정 맵  $A^k$ 에 중요도  $\alpha_k^c$ 를 곱한 뒤, 이를 합산하여 계산된다.

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left( \sum_k \alpha_k^c A^k \right).$$

ReLU 활성화 함수는 음수 값을 제거하여 중요한 영역만을 강조한다.



Figure 2: Map Generation Process.

### 1.4 Guided Grad-CAM (Guided Backpropagation + Grad-CAM)

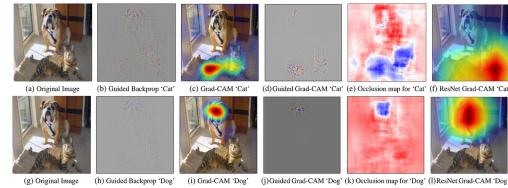


Figure 3: Tiger cat and Dog Class Analysis.

위의 그림은 하나의 이미지에 대해 "Tiger cat"(상단)과 "Dog"(하단) 분석을 시각화한 것이다. Guided Backpropagation이나 Deconvolution과 같은 Pixel-space Gradient Visualizations의 경우 이미지의 세부적인 디테일을 하이라이트하기하기 때문에 각 class에 대한 차별적인 결과를 만들어주지는 못한다. 반면, CAM이나 Grad-CAM과 같은 localization approaches의 경우 각 class에 대한 차별적인 결과를 만들어주지만 세부적인 디테일은 잡아내지 못한다. Guided Grad-CAM은 이 두 방법의 장점을 융합하여 만들어졌다. 그림에서 (d)와 (j)를 보면, pixel 단위까지 세부적으로 분석하면서도 각각의 class에 맞는 영역들만 하이라이트한 것을 볼 수 있다. 이렇게 class-discriminative한 알고리즘은, 모델의 prediction이 틀렸을 경우에도 왜 틀린 예측을 만들었는지에 대한 논리적 근거를 찾을 수 있다.

#### 1.4.1 Application Tasks

앞서 backpropagation에 사용했던  $y$ 는, Image Classification의 class score 값일 수도 있지만, 다른 여러 가지 형태를 가질 수 있다. 구조화된 output caption의 activation 값이라면 Image Captioning에 적용될 수도 있고, 질문에 대한 answer의 activation 값이라면 VQA에 적용될 수도 있다. 또한, CAM은 마지막 특성 맵만 한정해서 시각화할 수 있었지만, Grad-CAM은 역전파를 통해 가중치를 계산하기 때문에 중간의 다른 conv layer에 대해서 시각화할 수 있다.

### 1.5 Result

"What Color is the firehydrant?"라는 질문에 대한 VQA에 알고리즘들을 적용해본 결과, "red", "yellow", "yellow and red"라는 세 개의 class에 대해 시각화했을 때, Guided Backpropagation은 모두 비슷한 이미지를 보였고, Grad-CAM과 Guided Grad-CAM은 각 class에 영향을 크게 미치는 영역들을 정확히 하이라이트하였다. Image Captioning model에 적용한 결과에서도 caption output을 예측하는데 크게 영향을 미친 영역들을 하이라이트한 것을 볼 수 있었다. VGG-16이 분류를 실패한 이미지들에 대해 Guided Grad-CAM을 시각화해보면, 왜 잘못된 예측을 만들어냈는지 확인하기 어렵지만, 이 알고리즘으로 이유를 확인할 수 있다.

## 2 Visual Explanations from Deep Networks via Gradient-Based Localization Practice

### 2.1 Introduction

Grad-CAM (Gradient-weighted Class Activation Mapping)은 CNN 기반 모델의 의사결정 과정을 시각적으로 설명하기 위한 기술이다. Grad-CAM은 특정 클래스와 관련된 가장 중요한 이미지 영역을 강조하여 모델의 투명성을 높이고 직관적인 해석을 가능하게 한다. 기존 CAM(Class Activation Mapping)과는 달리, Grad-CAM은 네트워크 구조 변경이나 재훈련 없이도 적용 가능하다. 본 실험에서는 AlexNet과 VGG-16 모델을 활용하여 Grad-CAM의 작동 원리를 구현하고, 이를 통해 CNN 기반 분류 모델의 결정 영역을 시각화한다.

### 2.2 Practice Method

Grad-CAM은 다음과 같은 방식으로 작동한다. 먼저, 모델의 마지막 convolutional layer의 feature map을 추출한다. 그 다음, 특정 클래스 점수의 gradient를 feature map에 대해 계산한다. 계산된 gradient를 글로벌 평균 풀링하여 클래스별 중요도를 나타내는 가중치( $\alpha$ )를 얻는다. 이후 중요도 가중치와 feature map을 가중합하여 Grad-CAM을 생성한다. 이렇게 생성된 Grad-CAM을 입력 이미지 크기로 리사이즈하고 ReLU를 적용한 후 정규화한다. 이 실험에서는 PyTorch를 사용해 AlexNet과 VGG-16 모델에 Grad-CAM을 적용하여 결과를 확인한다.

### 2.3 Dataset

실험 데이터셋으로는 ImageNet에서 추출한 샘플 이미지를 사용하였다. 이미지들은 CNN 모델의 입력 크기에 맞춰 224x224 크기로 리사이즈된 후, 평균 및 표준편차를 기반으로 정규화되었다. 이러한 사전 처리 과정은 모델 학습 시 적용된 규칙을 그대로 따르기 위함이다.

### 2.4 Implementation

#### 2.4.1 AlexNet & VGG-16

AlexNet은 5개의 convolutional layer와 3개의 fully connected layer로 구성된다. 각 convolutional layer는 ReLU 활성화 함수와 필요한 경우 max-pooling을 포함한다. Fully connected layer는 드롭아웃(dropout)을 적용하여 과적합을 방지한다.

```
self.features = nn.Sequential(
    AlexNet_Block(3, 64, kernel_size=11, stride=4,
                 padding=2, maxpool=True),
    AlexNet_Block(64, 192, kernel_size=5, stride=1,
                 padding=2, maxpool=True),
    AlexNet_Block(192, 384, kernel_size=3, stride=1,
                 padding=1),
    AlexNet_Block(384, 256, kernel_size=3, stride=1,
                 padding=1),
    AlexNet_Block(256, 256, kernel_size=3, stride=1,
                 padding=1, maxpool=True)
)
```

```
self.classifier = nn.Sequential(
    nn.Dropout(),
    nn.Linear(256 * 6 * 6, 4096),
    nn.ReLU(inplace=True),
    nn.Dropout(),
    nn.Linear(4096, 4096),
    nn.ReLU(inplace=True),
    nn.Linear(4096, num_classes)
)
```

VGG-16은 네트워크가 깊어지면서 더 작은 커널(3x3)을 사용하는 특징이 있다. Configurations를 사용해 네트워크의 layer構성을 동적으로 정의하며, make\_layers 함수를 활용해 모델 구조를 구현한다. 이번 구현에선 VGG-16 모델인 논문의 D모델을 사용한다.

```
cfg = {
    'D': [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 'M',
          512, 512, 512, 'M', 512, 512, 512, 'M'],
}
def make_layers(cfg, batch_norm=False):
    layers = []
    in_channels = 3
    for v in cfg:
        if v == 'M':
            layers += [nn.MaxPool2d(kernel_size=2, stride
                                   = 2)]
        else:
            conv2d = nn.Conv2d(in_channels, v, kernel_size
                             = 3, padding=1)
            if batch_norm:
                layers += [conv2d, nn.BatchNorm2d(v), nn.
                           ReLU(inplace=True)]
            else:
                layers += [conv2d, nn.ReLU(inplace=True)]
            in_channels = v
    return nn.Sequential(*layers)
```

```
self.classifier = nn.Sequential(
    nn.Linear(512 * 7 * 7, 4096),
    nn.ReLU(True),
    nn.Dropout(),
    nn.Linear(4096, 4096),
    nn.ReLU(True),
    nn.Dropout(),
    nn.Linear(4096, num_classes),
)
```

#### 2.4.2 Grad-CAM Generator Class

Grad-CAM Generator는 다음 기능을 포함한다.

- save\_gradient: 특정 레이어의 gradient를 저장하는 후크 함수이다.
- forward\_model: Grad-CAM을 생성하기 위한 feature map과 모델 출력값을 계산한다.
- gen\_CAM: feature map과 저장된 gradient를 기반으로 Grad-CAM 마스크를 생성한다.

### 2.5 Training

모델은 사전 학습된 가중치(pretrained weights)를 활용하였으며, 추가적인 학습은 수행하지 않았다. AlexNet과 VGG-16 모두 ImageNet 데이터셋에서 사전 학습된 모델을 기반으로 한다.

### 2.6 Inference

이미지는 Grad-CAM generator 클래스에 입력되어 convolutional feature map과 gradient를 기반으로 Grad-CAM 마스크를 생성하였다. 생성된 마스크는 입력 이미지와 시각적으로 합성되어 모델의 주목 영역을 강조한다.

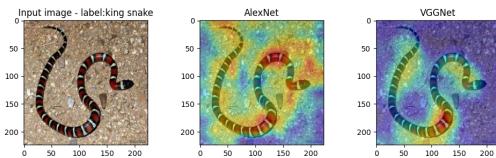


Figure 4: King Snake Grad-CAM.

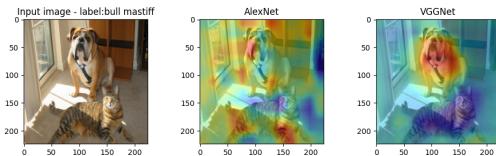


Figure 5: Bull Mastiff Grad-CAM.

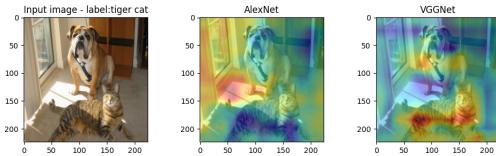


Figure 6: Tiger Cat Grad-CAM.

### 2.7 Conclusion

본 실험을 통해 Grad-CAM을 구현하고 AlexNet과 VGG-16 모델에 적용하여 특정 클래스에 대한 모델의 주목 영역을 시각화하였다. Grad-CAM은 모델의 의사결정 과정을 직관적으로 이해할 수 있게 해주는 유용한 도구임을 확인하였다. 특히, 재훈련 없이 사전 학습된 모델에 적용 가능하다는 점에서 실용성이 높다. 향후에는 다양한 CNN 모델과 데이터셋에서 Grad-CAM의 일반성을 평가할 예정이다.

- [1] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.