Self-Attention Generative Adversarial Networks

SAGAN 논문 구현

Abstract

이 논문에서는 Self-Attention Generative adversarial Network, 줄여서 SAGAN에 대해서 다룹니다.

이 모델은 1. attention-driven, 2. long-range dependency modeling을 이용했습니다.

전통적인 convolutional GAN은 고해상도의 디테일을 저해상도 feature map에서의 부분적인 local points의 함수로만 만들어냅니다. 하지만 SAGAN에서의 디테일한 부분들은 모든 feature locations으로부터의 cue(단서)을 이용해서 만들어내집니다. (즉, 디테일한 부분 생성이 이미지의 모든 특성들을 이용해서 만들어진다는 뜻입니다)

거기다가, discriminator가 이미지의 멀리있는 부분의 디테일들이 서로 일치하다는 것을 확인할 수 있다.(long range dependency와 같은 맥락)

더욱 더, 최근 연구들이 보여준 바로는, generator conditioning은 GAN의 성능에 영향을 미치는데 이러한 관점을 이용하여 이 논문에서는 spectral normalization을 GAN generator에 적용하고 이는 학습에 효과가 있음을 밝혀냈습니다.

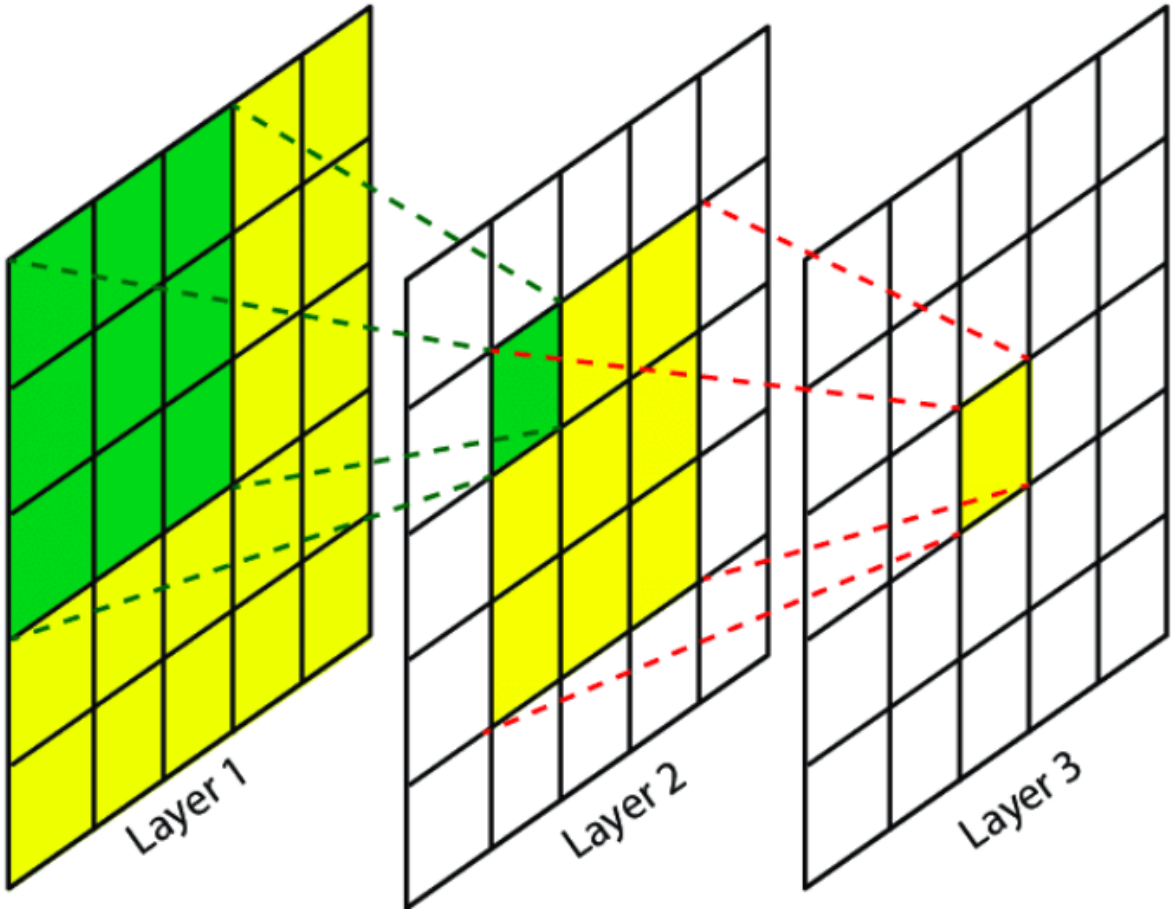
Introduction

이미지 합성은 computer vision에서 중요한 문제인데, 많은 open problems이 여전히 남아있지만 GAN이 등장하면서 많은 진보가 되어왔다. 특히, Deep convolutional network을 베

이스로한 GAN은 특히 성공을 보여줬습니다.

그러나, 이러한 모델들(Deep CNN기반)로부터 만들어진 샘플들을 주의깊게 조사해본 결과, convolutional GAN은 ImageNEt과 같은 multi-class 데이터셋에 훈련된 다른 모델들보다 몇몇의 이미지 classes를 모델링 하는데에 어려움을 겪어왔습니다. 예를 들어, SOTA ImageNet GAN모델이 몇몇의 구조적인 제한(기하학적 특성보다, 텍스쳐로 구분되는 바다, 하늘과 같은 풍경 클래스들)을 가지고 이미지 클래스들을 합성하는데 특출난 반면에, 몇개의 클래스에서 일정하게 생기는 기하학적 구조 또는 구조적인 클래스들을 합성하는데 특출난 반면에, 몇개의 클래스에서 일정하게 생기는 기하학적구조 또는 구조적인 패턴을 잡아내는 데에는 좋은 성능을 보여주지 않습니다. 예를 들어, 개들은 진짜같은 털 질감으로 그려지지만, 명확하게 발을 가지는 “개”를 보여주기에는 CNN기반의 GAN은 어렵다는 의미입니다. (그래서 “개”를 generation했을 때 얼굴이나, 개의 특정 texture는 굉장히 사실적으로 잘 그렸지만, detail한 part를 봤을 때에는 다리나, 꼬리가 굉장히 비정상적으로 생성된 경우가 많았습니다)

이에 대해 저자는 이전의 모델들이 다른 이미지 영역간의 dependency를 모델링할 때 convolution에 크게 의존했다는 것을 문제 삼습니다. Convolution operation은 local receptive field라는 특수성을 가지기 때문에, long range dependency는 초기의 layer가 아니 후반부의 몇개의 convolution layer만 처리될 수 있는 부분입니다. (아래의 그림 참고)



이로 인해 CNN구조를 채택한 GAN의 경우 long-term dependency를 학습하는 것에 많은 어려움이 있습니다. (아마 이로 인해 transformer의 Attention등을 사용해볼까? 하는 생각이 생겼던 것 같습니다. transformer의 Attentnion의 개념 자체가 long-term dependency를 쉽게(?), 병렬적으로(?) 무튼, 학습하기 쉬운 concept이기 때문입니다.)

Self-Attention을 사용함으로써 convolution을 보완하고, 먼 거리에 있는 모델링에 도움을 주며, 이미지 영역 사이에서 long-term dependency로 보완할 수 있습니다.

Self-Attention을 사용함으로써, generator는 모든 위치에서 세심하게 생성할 수 있고 discriminator는 전체적인 이미지에서 복잡한 기하학적인 제약을 가할 수 있습니다.

1. 작은(layer가 깊지 않은) 모델은 long-term dependency를 제대로 얻을 수 없습니다.
    - Convolution 커널의 사이즈를 증가시키는 것은 네트워크의 representational capacity(이미지를 잘 represent하는 능력)을 증가시킬 수 있지만, 또한 computational과 local 컨볼루션의 구조를 사용함으로써 얻는 통계적인 효율성을 떨어뜨린다.(커널 사이즈가 작으면 세세하게 이미지의 부분을 이용하므로 제너럴하게 이미지를 represent하는 능력이 떨어지므로 커널 사이즈가 커지면 representational capacity를 얻는다. 하지만, 당연히 계산 효율성이 떨어지게 된다. 요약하자면 deep CNN기반의 GAN은 이러한 trade-off가 있다는 것.)
    - 반대로 Self-attention은 long-range dependency를 모델링하는 능력과 compuational and statistical 효율성간의 더 나은 발란스를 보여준다. Self-attention 모듈은 한 위치에서의 반응을 **모든 위치에서의 features의 weighted sum**으로 계산한다. (참고로, 여기서 weights나 attention vectors을 계산하는데에는 매우 작은 cost만 필요하다.)
2. optimizing algorithm이 최적의 파라미터 값들을 찾아내는데에 어려움을 겪도록 하는데, 이 말은 loss function을 최적화하는 알고리즘이 제대로 parameter를 조정하지 못하도록 한다는 의미라고 해석됩니다.(?)
3. 이러한 매개변수화가 통계적으로 취약할 수 있으며 새로운 input에 적용될 때 제대로 작동하지 않을 수 있습니다.

Self-Attention Generative Adversarial Network

The image features from the previous hidden layer $\boldsymbol{x} \in \mathbb{R}^{C \times N}$ are first transformed into two feature spaces $\boldsymbol{f}, \boldsymbol{g}$ to calculate the attention, where $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W_f x}$, $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{W_g x}$

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})}, \text{ where } s_{ij} = \boldsymbol{f}(\boldsymbol{x_i})^T \boldsymbol{g}(\boldsymbol{x_j}), \quad (1)$$

and $\beta_{j,i}$ indicates the extent to which the model attends to the i^{th} location when synthesizing the j^{th} region. Here, C is the number of channels and N is the number of feature locations of features from the previous hidden layer. The output of the attention layer is $\boldsymbol{o} = (\boldsymbol{o_1}, \boldsymbol{o_2}, ..., \boldsymbol{o_j}, ..., \boldsymbol{o_N}) \in \mathbb{R}^{C \times N}$, where,

$$\boldsymbol{o_j} = \boldsymbol{v}\left(\sum_{i=1}^{N} \beta_{j,i} \boldsymbol{h}(\boldsymbol{x_i})\right), \ \boldsymbol{h}(\boldsymbol{x_i}) = \boldsymbol{W_h x_i}, \ \boldsymbol{v}(\boldsymbol{x_i}) = \boldsymbol{W_v x_i}. \quad (2)$$

In the above formulation, $\boldsymbol{W_g} \in \mathbb{R}^{\bar{C} \times C}$, $\boldsymbol{W_f} \in \mathbb{R}^{\bar{C} \times C}$, $\boldsymbol{W_h} \in \mathbb{R}^{\bar{C} \times C}$, and $\boldsymbol{W_v} \in \mathbb{R}^{C \times \bar{C}}$ are the learned weight matrices, which are implemented as 1×1 convolutions. Since We did not notice any significant performance decrease when reducing the channel number of $\bar{C}$ to be C/k, where $k = 1, 2, 4, 8$ after few training epochs on ImageNet. For memory efficiency, we choose $k = 8$ (*i.e.*, $\bar{C} = C/8$) in all our experiments.

$$\beta_{j,i} = \frac{\exp\left(s_{ij}\right)}{\sum_{i=1}^{N} \exp\left(s_{ij}\right)}, \text{ where } s_{ij} = f\left(\boldsymbol{x}_i\right)^T \boldsymbol{g}\left(\boldsymbol{x_j}\right)$$

$$\boldsymbol{o_j} = \boldsymbol{v}\left(\sum_{i=1}^{N} \beta_{j,\,i}\, \boldsymbol{h}\left(\boldsymbol{x_i}\right)\right), \ \boldsymbol{h}\left(\boldsymbol{x_i}\right) = \boldsymbol{W_h}\, \boldsymbol{x_i}, \ \boldsymbol{v}\left(\boldsymbol{x_i}\right) = \boldsymbol{W_v}\, \boldsymbol{x_i}$$

$$\boldsymbol{W_g} \in \mathbb{R}^{\bar{C} \times C}, \boldsymbol{W_f} \in \mathbb{R}^{\bar{C} \times C}, \boldsymbol{W_h} \in \mathbb{R}^{\bar{C} \times C}, \text{ and } \boldsymbol{W_v} \in \mathbb{R}^{C \times \bar{C}}$$

왼쪽부터 가봅시다.

$B_{j,i}$는 j번째 영역을 합성할 때 i번째 위치를 참여하는 범위를 나타냅니다.

여기서 C는 channel의 수 N은 이전 hidden layer로부터 피처들의 피처 위치의 수라고 하고, $k = 8$이라는 값도 사용했다고 합니다. (k는 조금 있다가 설명하겠습니다)

위 수식보다는 아래의 그림이 조금 더 이해하기 좋습니다.

결국 이전 convolution연산을 통해 나온 feature map을 1x1 conv를 통해 channel reduction + Attention에 Fit한 연산이 가능하도록 진행되는데 이는 기존의 FC layer를 통해 bottle neck구조를 가지던 구조와 동일하다고 생각하면 좋을 듯 합니다.

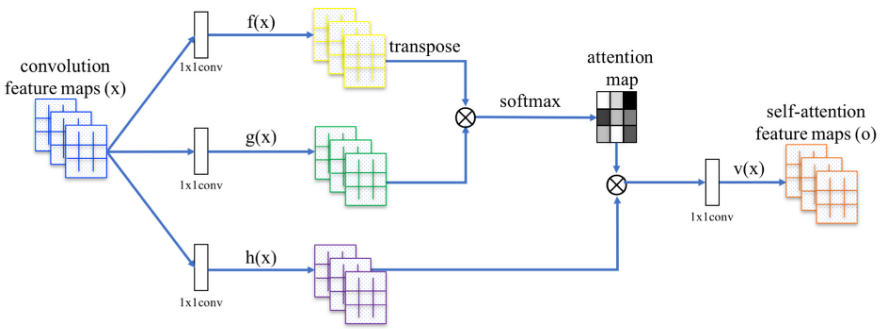


Figure 2. The proposed self-attention module for the SAGAN. The ⊗ denotes matrix multiplication. The softmax operation is performed on each row.

여기에서 O는 **output(self-attention feature maps)**입니다.

attention output에다가 **scale parameter**를 곱하고 input feature map을 더합니다.

$$\boldsymbol{y}_i = \gamma\, \boldsymbol{o_i} + \boldsymbol{x_i}$$

즉 결국 위와 같이 표현 가능하고, 직관적으로 이해가 쉬울 듯 합니다. 여기서 γ는 학습 가능한 상수값이고, 처음에는 0으로 초기화됩니다. γ를 도입하는 것은 네트워크가 처음에는 local neighborhood에서 단서를 얻다가 점진적으로 non-local(long distance를 포함한 attention해야하는)단서들에 중점을 맞추도록 합니다. → 이는 다르게 표현하면 먼저 주변부 이해하고, 직접적으로 network상에서 영향을 주는 것들을 배운뒤에, complexity를 높이며 학습을 한다(?) 라는 느낌…?!ㅋㅋㅋ

Techniques to Stabilize the Training of GANs

아래의 2가지로 정리 가능할 듯 합니다.

1. Spectral Normalization의 사용
2. TTUR(two timesacle update rule)이 효과적이라는 것을 확인함.

4.1 Spectral normalization for both generator and discriminator

generator에서 spectral normalization은 비정상적인 gradient를 막아주고 parameter의 크기가 너무 커지는 것을 막아줍니다. 다른 블로거 분들의 경험적인 부분도 보면, SN을 generator와 discriminator에 사용하는 것이 generator당 discriminator update를 더 적게 하는 것을 확인했고, 이로 인해 계산량도 줄어들고 안정적이라고 합니다.

이런 의견도 있네요

수학적으로 정확한 정의는 아니지만, Lipschitz constant를 억제시킨다는 것은 gradient, 즉 파라미터간의 차이를 억제시킨다고 간단하게 이해할 수 있다.

→ 이거 하나로도 논문 1개라 다음번에 읽어보겠습니다…

4.2 Imbalanced learning rate for generator and discriminator

discriminator의 noramlization 기법들은 종종 학습 process를 느리게 만든다고 합니다. 실제로 정규화된 discriminator을 사용한 방법들은 보통 generator 한번 업데이트 시 5번의 업데이트를 한는 등의 방법을 주로 사용했다고 합니다.

TTUR은 쉽게 말하면 seperate learning rate를 의미해, generator와 discriminator에 각기 다른 learning rate를 사용하는 것을 의미합니다.

time-scale stochastic approximations algorithms. For a two time-scale update rule (TTUR), we use the learning rates $b(n)$ and $a(n)$ for the discriminator and the generator update, respectively:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + b(n)\left(\boldsymbol{g}(\boldsymbol{\theta}_n, \boldsymbol{w}_n) + \boldsymbol{M}_n^{(w)}\right), \ \boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + a(n)\left(\boldsymbol{h}(\boldsymbol{\theta}_n, \boldsymbol{w}_n) + \boldsymbol{M}_n^{(\theta)}\right). \quad (1)$$

음~ 모르겠습니다~!

5. Experiment

Attention map을 시각화한 이미지들로써 다음과 같습니다.

구현 코드에서 해당 부분도 얻어내, 사용해보고 싶습니다.

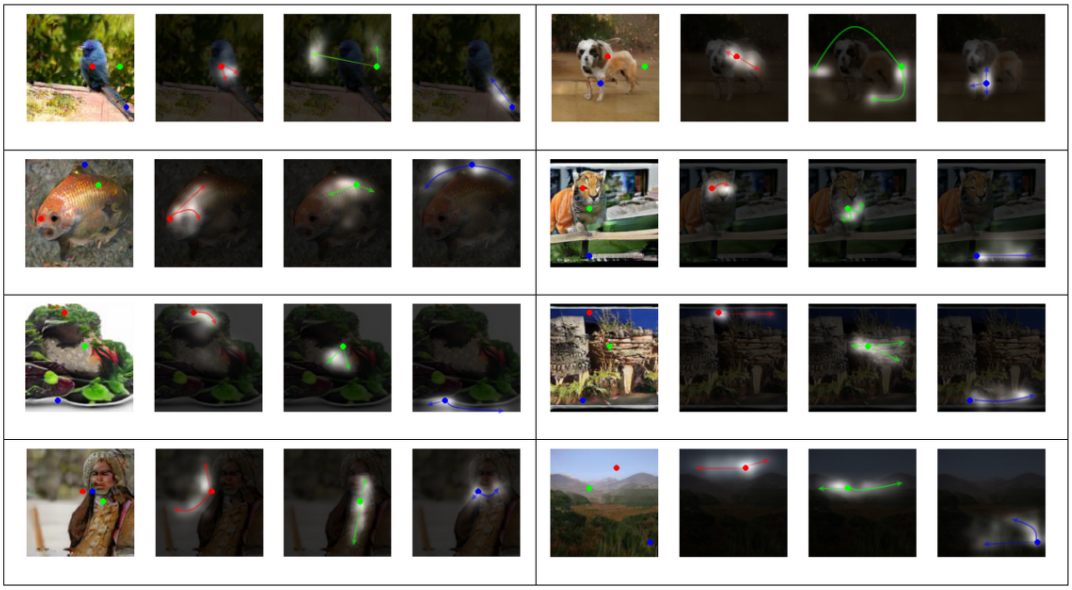


Figure 5. Visualization of attention maps. These images were generated by SAGAN. We visualize the attention maps of the last generator layer that used attention, since this layer is the closest to the output pixels and is the most straightforward to project into pixel space and interpret. In each cell, the first image shows three representative query locations with color coded dots. The other three images are attention maps for those query locations, with corresponding color coded arrows summarizing the most-attended regions. We observe that the network learns to allocate attention according to similarity of color and texture, rather than just spatial adjacency (see the top-left cell). We also find that although some query points are quite close in spatial location, their attention maps can be very different, as shown in the bottom-left cell. As shown in the top-right cell, SAGAN is able to draw dogs with clearly separated legs. The blue query point shows that attention helps to get the structure of the joint area correct. See the text for more discussion about the properties of learned attention maps.

참고할 만한 article

- Spectral Normalization : https://hichoe95.tistory.com/61
- TTUR

```
parser.add_argument(
    '--lr_g', default=1e-4, type=float, help='learning rate of generator'
)
parser.add_argument(
    '--lr_d', default=4e-4, type=float, help='learning rate of discriminator'
)
```

  실험적으로 검증했다고 함.