Abstract

Figure 1: synthetic images

Figure1은 [1]에서 만들어낸 synthetic 이미지들로, Generative Adversarial Nets(GAN, 생성적 적대 신경망 [2])을 기반으로 한 AI모델들이 육안으로 구분하기 어려울 정도의 이미지를 만들어낼 수 있는 역할을 수행하고 있다. 또한 GAN의 발전으로 'Fake Face'를 만드는 기법이 고도화 되며, 이 기술을 이용하여 유명 SNS의 가상 인플루언서로 활동 중인 로지가 있다. 심지어는, GAN의 기술의 악용을 방지하기 위해 이 가상 인물에 대한 진위 여부를 확인할 수 있는 방안도 제시되었다. 이 외에도 GAN은 이를 기반으로한 소프트웨어를 통해 디자인, 예술적으로 사용되는 등 다양한 방면에서 사용되고 있다.

1 GAN - Generative Adversarial Nets

딥러닝을 활용한 classification model이 점점 더 발전하고 있다. 이는 piecewise linear unit을 사용함에 따라 layer가 깊어져도 gradient가 vanishing 되지 않아 backpropagation이 잘 이루어지는 것이 근간이 된다. 하지만, 딥러닝 기반의 생성모델(Deep generative models)은 존재하지 않지만 실제와 같은 이미지를 생성하는 모델로, intractable probabilistic computations를 근사하는 것이 어려웠으며 generative context에 Relu와 같은 piecewise linear units의 장점을 활용하는 것이 쉽지 않았다. 따라서 [2])에서는 이러한 문제를 회피할 수 있는 새로운 생성적 모델 추정 절차를 제안한다.

1.1 Two models of new Framework - Adversarial nets

GAN model은 두 개의 Adversarial 네트워크를 활용한다. G :생성자(Generator)와 D : 판별자(discriminator), 두 개의 네트워크가 활용되며 두 네트워크는 objective function $V(D,G)$에 따라 학습된다. 생성자(D)는 노이즈 sample을 입력으로 받아 새로운 이미지 instance를 생성하고, 판별자(G)는 데이터를 받아서 이 데이터가 실제 데이터인지 가짜 데이터인지 확률값(0 1)을 내보내어 판단한다. 최종적으로 사용하고자 하는 모델은 생성자이며, 판별자는 이 생성자가 잘 학습할 수 있도록 도와주기 위한 목적으로 사용된다. 목적함수(objective function)는 아래와 같고, 이는 학습에 대한 의미가 잘 담겨져 있다.

$$\min_G \max_D V(D,G) = \mathbf{E}_{x\sim p_{data}}[logD(x)] + \mathbf{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \quad (1)$$

목적함수 $V(D,G)$는 D와 G의 함수로 구성이 되어 있는데, 이때 G는 V를 낮추려고하고 D는 V의 값을 높이고자하는 것이 목적이다. 우변의 두 항에 대해 자세히 살펴본다.

$$\mathbf{E}_{x\sim p_{data}}[logD(x)] \quad (2)$$

2의 p_{data}는 원본 데이터의 distribution으로, 샘플링된 학습 데이터 x에 함수 D와 log를 취하여 기댓값을 구하는 것이다.

$$\mathbf{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \quad (3)$$

3의 생성자(G)는 노이즈 variables $p(z)$의 sample z를 입력으로 받아 새로운 이미지를 만들어내고, 이를 판별자 D에 넣은 값을 1에서 빼고 log를 취하는 수식이다. 2, 3의 극단적인 두 경우를 살펴봄으로써 목적함수에 대해 더 잘 이해할 수 있다. 만약 모든 fakereal image를 판단할 수 있는 판별자 네트워크의 경우 $logD(x)=1, D(G(z))=0$이 됨으로써 V의 값이 0이 되어 최대값을 갖는다. 반면, G가 생산해낸 이미지가 실제이미지와 같아 $D(G(z))=1$에 가까워 지는 경우 V의 값이 $-\infty$로 최소값을 갖는다. 즉, G는 V를 낮추려고하고 D는 V를 높이고자하는 것은 학습을 통해 판별자(D)가 원본 데이터에 대해서는 1, 가짜 데이터에 대해서는 0을 출력하도록 하고, 생성자(G)가 만든 이미지가 판별자에 의해서 원본 이미지라고 인식될 수 있도록 하는 것이다.

1.2 Goal of Objective Function

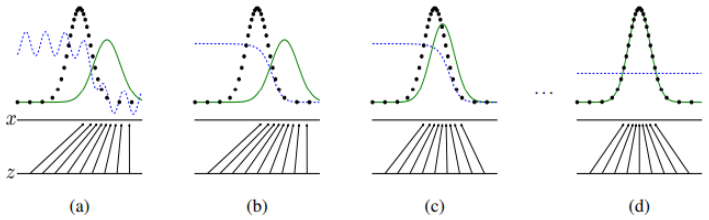

Figure 2:

이미지와 같은 context는 모두 벡터나 행렬과 같은 데이터 형태를 가질 수 있기 때문에 확률 분포로 모델링을 할 수 있을 것이고, 많은 픽셀과 rgb값이 존재하기 때문에 확률 분포는 고차원 공간에 존재하는 분포로 나타난다. Figure2는 목적함수에 따라 train을 진행할 때 최종적인 결과가 어떻게 되는지 단순화하여 잘 보여준다. 검정색 이산적인 분포는 원본 데이터의 분포(P_{data})를 나타내고, 초록색 연속분포는 생성 모델(P_g)의 분포, 파란색 분포는 판별 모델의 분포를 나타낸다. 최종적인 목적함수의 목표는 $P_g = P_{data}$로, 판별자 D가 확률분포 P_g를 따르는 입력(z)을 받아 생성자를 통해 만들어진 이미지를 50%의 확률로 판단하는 것이다. 즉 학습을 함에 따라 생성모델의 분포가 원본 데이터의 분포를 따르게 되며, 판별자가 원본 이미지인지 생성 이미지인지 구분할 수 없는 상황으로 Figure2의 (d)와 같이 분포가 나타나는 것이다. 그렇다면, P_g가 P_{data}로 어떻게 수렴할 수 있는지 알아본다.

1.2.1 Global Optimality

P_g가 P_{data}로 수렴을 확인하기 위해 먼저 Global Optimality에 대해 알아본다. Global Optimality는 generator G가 고정되어 있을 때, optimal 판별자 D는 아래의 식과 같다는 것이다.

$$D^*_G(x) = \frac{p_{data}}{p_{data}+p_g(x)} \quad (4)$$

이 optimal D는 목적함수 V를 maximize하는 것으로 부터 구할 수 있다.

$$V(G,D) = \mathbf{E}_{x\sim p_{data}}[logD(x)] + \mathbf{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

$$(E[X] = \int_{-\infty}^{\infty} xf(x)dx)$$

$$\rightarrow \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log(1-D(g(z)))dz$$

$$\rightarrow \int_x p_{data}(x)log(D(x)) + p_g(x)log(1-D(x))dx \quad (5)$$

최종적으로 5에서 optimal D는 $alog(y)+blong(1-y)$가 $\frac{a}{a+b}$에서 최대 값을 갖는 것에서 $D^*_G(x) = \frac{p_{data}}{p_{data}+p_g(x)}$임을 구할 수 있다.
이제, 위의 global optimal G를 이용하여 생성자의 global optimum point를 구할 수 있다. 이를 구하기 위해 별도의 함수 $C(G) = \max_D V(D,G) = \mathbf{E}_{x\sim p_{data}}[logD(x)] + \mathbf{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$가 정의된다. 이때 4에 의해서 아래와 같이 식을 전개할 수 있다.

$$C(G) = \mathbf{E}_{x\sim p_{data}}[log\frac{p_{data}}{p_{data}+p_g(x)}] + \mathbf{E}_{x\sim p_g(x)}[log\frac{p_g}{p_{data}+p_g(x)}]$$

$$\rightarrow \mathbf{E}_{x\sim p_{data}}[log\frac{2*p_{data}}{p_{data}+p_g(x)}] + \mathbf{E}_{x\sim p_g(x)}[log\frac{2*p_g}{p_{data}+p_g(x)}] - log(4)$$

(by Kullback–Leibler divergence (KL divergence))

$$\rightarrow KL(p_{data}||\frac{p_{data}+p_g(x)}{2}) + KL(p_g||\frac{p_{data}+p_g(x)}{2}) - log(4)$$

Jensen–Shannon divergence

$$\rightarrow 2*JSD(p_{data}||p_g) - log(4) \quad (6)$$

6에서 $JSD(p_{data}||p_g)$의 값은 non-negative하기 때문에 $p_g = p_{data}$일 때 최솟값으로 $-log(4)$를 갖는다. 따라서 $C(G)$의 global minimum, 즉 생성자 G의 global optimum point는 $p_g = p_{data}$이다.

1.3 Training algorithm of GAN

epoch 수 만큼 반복한다 매 epoch당 k번 판별자(D)를 학습한 뒤 생성자(G)를 학습한다. 판별자(D)를 학습할 때, m개의 noise data와 원본 데이터를 sampling하고 이 samples를 이용하여 아래와 같은 gradient를 구한 후, ascending하여 학습된다.

$$\Delta_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}[logD(x^i) + log(1-D(G(z^i)))]$$

생성자(G)를 학습할 때, m개의 noise data를 sampling한 뒤에 m개의 fake image를 만들어 아래의 gradient를 통해 descending하여 학습된다.

$$\Delta_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}log(1-D(G(z^i)))$$

2 DCGAN - DEEP CONVOLUTIONAL GAN

논문이 쓰일 당시 CNN의 비지도 학습은 지도 학습보다 덜 주목받고 있었다. 하지만 지도학습은 lable이 있는 데이터셋이 반드시 요구되기 때문에 시간, 비용이 들어가는 한계점이 존재하는 반면 비지도 학습은 라벨 없이 학습을 할 수 있는 장점이 있다. GAN은 비지도 학습에 속하며, 데이터를 직접 생성한다는 특징이 있어 지도 학습의 데이터로도 사용될 수 있다. 이 논문([3])에서는 적대적 생성 신경망(GAN, Generative Adversarial Networks)의 한계점을 개선할 수 있는 방법을 제시하고, 발전된 결과와 활용성을 보여준다.

2.1 Architecture of DCGAN

GAN은 기존의 학습방법과 달리 MinMax problem을 푸는 방식으로, 학습을 하기에 불안정한 구조를 갖는다. 이에 따라 Generator가 생성하는 이미지는 무의미한 경우가 종종 발생하였다. 또한 Fully-Connected 구조를 단순히 CNN으로 바꾸어 GAN의 성능 향상을 꾀했던 연구는 성공적이지 못하였다. 따라서 본 논문에서는 GAN의 학습의 안정성과 성능 향상을 도모하기 위해 GAN architecture에 5가지 방법을 적용한다.

- 판별자의 풀링 레이어를 strided convolution으로 대체하고, 생성자의 경우 fractional-strided convolution으로 대체한다

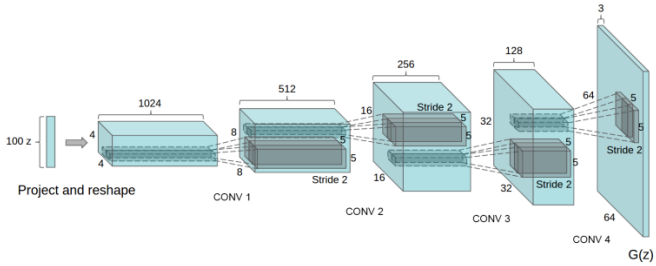

Figure 3:

- 생성자와 판별자 두 네트워크에 모두 batchnorm을 사용한다.
- 더 깊은 architecture를 만들기 위해 fully-connected layer를 삭제한다.
- 생성자 네트워크가 color space를 cover하고 더 빠르게 학습되도록 출력 레이어를 제외한 모든 layer에 ReLU를 적용하고, 출력 layer는 Tanh를 사용한다.
- 판별자가 높은 해상도에서도 잘 동작하도록 모든 layer에 Leaky ReLU를 사용한다.

결과적으로, CNN architecture를 유지하고 최근 증명된 3가지 연구(위에서 언급된 방법)를 적용함으로써 더 큰 데이터셋에 대해 안정적이고, 고해상도의 이미지와 깊은 생성모델을 학습할 수 있다. 각 연구가 학습에 있어 어떤 역할을 하는지 자세히 알아본다.

2.1.1 Strided convolution

먼저, maxpooling과 같은 deterministic spatial pooling functions를 strided convolution으로 대체하면서 네트워크가 고유의 spatial downsampling을 학습할 수 있도록 하였다. 이를 생성자에 적용하여 생성자의 spatial up-sampling을 학습할 수 있도록 하였다. 판별자도 역시 이 방식으로 학습을 진행하였다.

2.1.2 batch normalization

GAN은 생성자에서 모든 샘플이 하나로 붕괴되는 문제를 막기 위해 deep generator가 필요하다. 이에 따라batch bormalization을 적용하는데, batch-norm은 입력을 zero mean과 unit variance를 갖도록 정규화하여 학습을 안정화 시킨다. 이에 따라 좋지 못한 초기화로 인해 발생한 학습 문제를 다루는 것과 더 깊은 모델의 gradient flow를 돕는다. 하지만 GAN에서 모든 레이어에 batchnorm을 적용한 결과 sample oscillation과 모델의 불안정성의 문제가 발생하였고 이에 따라 생성자의 출력 layer와 판별자의 입력 layer에는 적용하지 않는다.

3 Conclusion

DCGAN은 Generative Adversarial Nets의 한계점을 극복할 수 있는, 즉 많은 task에서 안정적으로 학습할 수 있는 Convolution GAN의 구조를 제안하였다. DCGAN은 GAN을 이용한 많은 연구에 초석이 되었고 이를 활용한 기술이 다양하게 사용되고 있어, 꼭 이해하고 있어야할 논문이라 생각된다.

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[3] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.