학습 가능한 옵티마이저를 이용한 고속 StyleGAN 인버전

강경국 [O, 1], 김성태 [2], 조성현 [1,2]

포항공과대학교 [1]컴퓨터공학과, [2]인공지능대학원

kkang831@postech.ac.kr.ac.kr, seongtae0205@postech.ac.kr, s.cho@postech.ac.kr

요 약

최근 실제 같은 고품질의 합성 이미지를 생성하는 StyleGAN 이 제안된 이래로, 이를 이미지 편집의 사전 지식 (prior)으로 사용하고자 하는 다양한 연구들이 소개되었다. 이들은 원본 이미지를 StyleGAN 의 잠재 공간 (latent space)에 전사 (projection)하고, 이 잠재 공간 내에서의 편집을 통해 기존의 이미지 편집에서는 불가능했던 다양한 의미론적인 (semantic) 이미지 편집을 가능하게 했다. 하지만 전사 과정에서 주로 사용되는 반복 최적화 기법은 상당한 시간을 필요로 하여 실 생활에서의 적용을 어렵게 했다. 본 연구에서는 깊은 인공 신경망을 이용하여 학습 가능한 옵티마이저를 설계하고 이를 데이터를 통해 최적의 업데이트 방향을 추정하도록 학습하여 기존의 방법들에 비해 보다 빠른 전사를 가능하게 한다.

1. 서론

최근의 이미지 생성 모델은 적대적 생성망 (Generative Adversarial Networks (GANs) [1])의 급격한 발전에 힘입어 매우 높은 품질의 이미지를 생성한다. 적대적 생성망은 랜덤 노이즈 벡터 z로부터 합성 이미지를 생성하는 생성자 (Generator, G)와 합성된 가짜 이미지와 학습 데이터인 실제 이미지를 분별하는 판별자 (Discriminator, D)가 서로 경쟁하며 학습이 진행되는 구조로, 합성 이미지의 분포가 점차 학습 데이터의 분포와 유사해지도록 학습된다.

지난 몇 년간 학습의 안정성을 높이고 높은 품질의 이미지를 생성하기 위해 다양한 GANs 가 제안되었다 [2, 3]. 그 중, StyleGAN [2]은 잠재 벡터 z 를 풀려진 (disentangled) 공간인 W 공간으로 매핑한 후, 매핑 된 $w \in W$ 벡터가 네트워크의 여러 중간 층 (layer)에 투입되어 이미지의 각 스케일의 스타일을 바꾸는 방식으로 이미지를 생성한다. StyleGAN 은 특히 얼굴 이미지의 생성에 있어 매우 높은 성능을 보인다.

한편, 학습된 StyleGAN 의 잠재 벡터에는 다양한 시맨틱 (의미론적인, semantic) 정보가 인코딩 (encoding) 되므로 잠재 벡터의 시맨틱 정보를 바꾸면 생성 이미지의 시맨틱을 변형할 수 있다. 이를 이용해 실제 이미지를 StyleGAN 의 잠재 공간으로 임베딩 (embedding) 하여 이러한 변형을 실제 이미지에 적용하려는 연구들 [4, 5, 6]이 제안되었고 이러한 임베딩을 일반적으로 StyleGAN 인버전 (inversion)이라 지칭한다. [4, 5]에서는 잠재 공간 W 를 이미지의 각 스케일에 대해 구분하여 확장된 공간인 $W+$ 공간을 정의하고 반복 최적화 기법을 이용해 해당 공간에 임베딩을 진행하면 효과적인 임베딩이 가능함을 보였고 임베딩 된 잠재벡터 $w_p \in W+$를 이용해 시맨틱한 이미지 편집 (modification)이 가능함을 보였다. [6]에서는 비선형 최적화 문제의 좋은 초기화를 위해 인코더 (encoder)를 도입하였고 최적화의 결과가 StyleGAN 의 학습된 잠재 공간에 매핑 될 수 있도록 하는 In-domain 로스 (loss)를 제안했다.

하지만, GAN 인버전에 주로 사용되는 반복 최적화 기법은 수행에 많은 시간을 필요로 한다. 예를 들어 256×256 크기의 이미지에 대해 [4,5]는 290 초, [6]은 8 초의 시간이 소요되며 이로 인해 실 사용에 상당한 불편함을 야기한다. 이를 해결하기 위해 본 논문에서는 학습 가능한 옵티마이저 네트워크를 제안하며 이를 이용해 기존의 Adam 옵티마이저에 비해 약 35% 빠른 최적화를 가능하게 한다. 본 네트워크는 현재 반복에서의 잠재 벡터와 Adam 옵티마이저의 업데이트 벡터를 입력으로 받아 다음 반복의 잠재 벡터 업데이트 방향을 산출한다.

본 논문의 구성은 다음과 같다. 2 장은 기존의 프레임워크에 대해 간단히 설명하고 본 논문에서 제안하는 방법을 소개한다. 3 장은 실험을 통해 제안하는 방법의 성능을 보인다. 마지막으로 4 장은 결론을 맺는다.

2. 고속 옵티마이저 설계

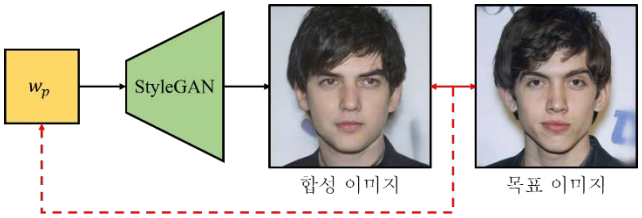


그림 1. 본 논문의 기초 프레임워크. 검은 실선은 네트워크의 순전파 (forward propagation), 붉은 선은 네트워크의 두 이미지의 로스 계산 및 역전파

2.1 일반적인 GAN 인버전 프레임워크

본 논문의 기초 프레임워크를 그림 1 에 나타냈다. 목표 이미지를 임베딩한 $w_p \in W+$ 벡터를 찾기 위해 반복 최적화 기법을 이용한다. 매 반복에서 w_p 벡터를 StyleGAN 의 입력으로 넣어 생성된 이미지와 임베딩하고자 하는 목표 이미지와의 로스를 계산하고 이를 역전파 (backpropagation)하여 얻어진 ∇w_p를 이용해 w_p를 업데이트 한다.

$$w_p = \arg\min_{w_p} \|x - G(w_p)\|_2^2 + \lambda_{vgg} \left\|F(x) - F\left(G(w_p)\right)\right\|_2^2 \quad (1)$$

수식 1 은 로스 함수를 나타내며 이때, x는 목표 이미지, G는 StyleGAN, F는 ImageNet 으로 학습된 vgg 네트워크 [9], λ_{vgg}는 로스 간 상대적 크기를 맞추기 위한 계수로 5×10^{-5}를 사용한다. 로스는 생성된 이미지와 목표 이미지 사이의 MSE 로스와 지각 로스 (perceptual loss)로 이루어져 있다. w_p의 업데이트를 위해 Adam 옵티마이저를 사용하며 스텝 사이즈 (learning rate)는 0.01 을 사용하였다. StyleGAN 인버전은 비선형 최적화이므로, 좋은 초기화를 위해 [6]에서 제공하는 학습된 인코더를 사용한다. 이러한 반복 최적화 기법은 상당한 시간을 요구하며 이를 해결하기 위해 2.2 절에서는 학습 가능한 옵티마이저를 제안한다.

2.2 학습가능한 옵티마이저 설계

일반적인 경사 하강법은 수식 2 와 같이 나타내어진다.

$$w_p^{(t+1)} = w_p^t - \alpha^t f(\nabla w_p^t) \quad (2)$$

옵티마이저는 t번째 반복에서 변수를 로스의 음의 기울기 (gradient) 방향으로 업데이트한다. 이때, $f(x)$ 는 옵티마이저 별로 정의된 기울기에 대한 함수이며, α^t는 미리 정의된 스텝 사이즈이다. 예를 들어, 본 논문의 기초 프레임워크에서 $f(x)$는 Adam 옵티마이저의 업데이트 규칙으로 이전 기울기의 기록을 고려한 기울기 방향을 산출하며, α^t 는 상수인 0.01 이다. Adam 옵티마이저 외에도 NAG, Adagrad, RMSProp 등 다양한 옵티마이저가 존재하며, 이들은 미리 설계된 각각의 업데이트 규칙을

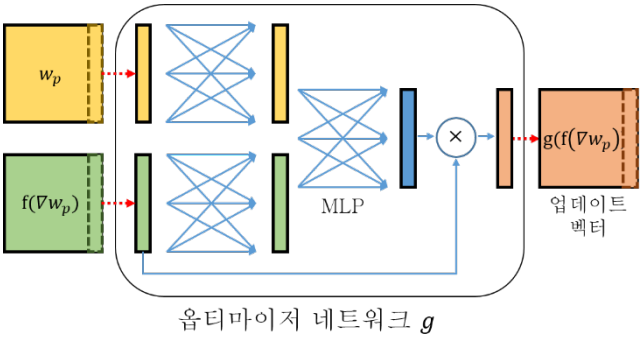


그림 2. 본 논문에서 제시하는 옵티마이저 네트워크

이용하여 업데이트 방향을 산출한다. 하지만 이러한 옵티마이저들은 풀고자 하는 문제에 따라, 사용하는 데이터 셋에 따라 다른 성능을 보이게 되며 각각의 옵티마이저에 대한 하이퍼 파라미터도 사용자가 실험을 통해 찾아야하는 번거로움이 있다. 이러한 문제를 해결하고 학습을 통해 최적의 옵티마이저를 찾기 위해 학습 가능한 옵티마이저를 설계하는 방법들이 제시되었다 [7,8]. [8]에서는 경사 하강법을 묘사하는 LSTM 네트워크를 설계하여 이미지 분류에 적용하였으며, [7]에서는 이미지 복원을 위한 이미지 디컨볼루션 (deconvolution) 최적화 네트워크를 제안했다.

본 논문에서는 Adam 옵티마이저를 기반으로 보다 좋은 경사 하강법을 묘사하는 네트워크를 설계하여 보다 빠른 StyleGAN 인버전을 구현한다. 본 논문에서 제안하는 네트워크 g는 그림 2 에 나타나 있다. 네트워크는 Adam 옵티마이저의 업데이트 방향과 현재 잠재 벡터를 입력으로 받아 새로운 업데이트 방향을 생성한다. 구체적으로 $f(\nabla w_p^t)$ 와 w_p^t 는 각각 4 개의 MLP (multi-layer perceptron)를 통과한 후 이어 붙여져 (concatenate) 다시 4 개의 MLP 를 통과한다. 이후 $f(\nabla w_p^t)$ 과 곱해져 $g(f(\nabla w_p^t, w_p^t))$를 산출한다. 한편 $w_p \in W+$는 잠재 벡터 $w \in W$를 이미지의 스케일 개수에 맞게 확장시킨 형태이므로 잠재 벡터 크기의 배수 형태의 크기를 갖는다. 예를 들어 256×256 크기의 이미지를 생성하는 StyleGAN 은 14 개의 layer 를 갖으며 따라서 w_p는 14×512 (512는 잠재 벡터의 크기) 크기를 갖는다. 본 논문에서는 w_p와 $f(\nabla w_p^t)$를 네트워크에 입력할 때 각 스케일별로 나눈 각각 512 차원의 벡터들을 입력으로 하며 네트워크는 이미지 스케일에 따라 다른 네트워크 가중치를 갖도록 설계했다.

네트워크를 학습하는 과정은 다음과 같다. 먼저 랜덤 샘플링 (sampling)을 통해 얻은 잠재 벡터 w_p^{gt}와 이를 StyleGAN 에 통과시켜 합성 이미지 $I = \text{StyleGAN}(w_p^{gt})$ 를 생성할 수 있다. 이 과정을 반복하여 학습 데이터 셋을 생성한다. 학습 데이터 셋의 각 이미지에 대해 알고리즘 1 을 통해 네트워크를 학습한다. 구체적으로 목표 이미지를 [6]에서 제공하는 인코더에 통과하여 이를 반복

Algorithm 1

Input: Target image I, network parameter θ

Get initial vector : $w_p^0 = \text{encoder}(I)$

For $t = 0$ **to** max iteration **do**

calculate loss :

$$\text{loss} = \left\|\text{StyleGAN}\left(w_p^t\right) - I\right\|_2^2 + \lambda_{vgg}\left(\left\|F\left(\text{StyleGAN}\left(w_p^t\right)\right) - F(I)\right\|_2^2\right)$$

loss backpropagation : $\text{loss} \rightarrow \nabla w_p^t, \nabla\theta_I^t$

If $t = 0$ then, pass

Else

Calculate Gt loss : $\text{Gt loss} = \left\|w_p^t - w_p^{gt}\right\|_2^2$

Gt loss backpropagation : $\text{Gt loss} \rightarrow \nabla\theta_{w_p}^t$

Calculate final network gradient :

$\nabla\theta^t = \nabla\theta_I^t + \nabla\theta_{w_p}^t$

Update optimizer network

End If

Calculate new update direction :

$w_p^{t+1} = w_p^t + g(f(\nabla w_p^t), w_p^t)$

End For

최적화의 초기값으로 사용한다. 첫번째 반복에서 그림 1 의 순전파를 진행하여 얻어진 합성 이미지와 목표 이미지 사이의 로스를 계산하고 이를 역전파하여 ∇w_p^t 를 계산한다. 이때, 옵티마이저 네트워크를 통과하지 않았기 때문에 $\nabla\theta_I^t$는 0이다. 다음으로 $f(\nabla w_p^t)$ 와 w_p^t 를 옵티마이저 네트워크를 통과시켜 $g(f(\nabla w_p^t), w_p^t)$ 를 얻고 이를 이용해 다음 잠재 벡터 w_p^{t+1} 를 계산한다. 이후 반복부터는 역전파시 $\nabla\theta_I^t$ 가 생성되어 네트워크의 학습에 이용될 수 있다. 또한, 최종 목표인 w_p^{gt}를 이용하여 이미지 레벨에서의 로스가 아닌 잠재 벡터 레벨의 지도학습 로스를 도입하고 이를 Gt loss 라 명명한다. Gt loss 는 현재의 잠재 벡터와 정답 (ground truth)과의 차이로 정의되며 학습시에만 이용된다. 이미지 레벨의 로스와 마찬가지로 이를 역전파하여 Gt loss 에 대한 네트워크 파라미터의 기울기인 $\nabla\theta_{w_p}^t$ 를 얻을 수 있다. 네트워크에 대한 두 기울기를 더해 최종 기울기를 계산하고 네트워크를 업데이트 한다. 이때, 네트워크 업데이트를 위한 옵티마이저로는 Adam 옵티마이저를 사용하였다.

3. 실험 결과 및 분석

3.1 실험 설정

본 논문에서는 [6]에서 제공하는 256×256 크기의 이미지를 생성하는 StyleGAN 에 대해 실험을 진행하였으며 이는 70,000 장의 얼굴 영상으로 구성된 FFHQ 데이터 셋 [2]을 이용해

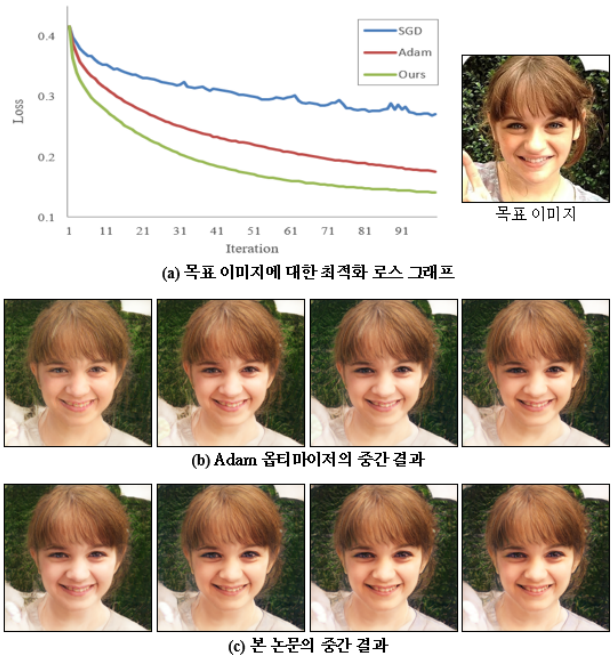


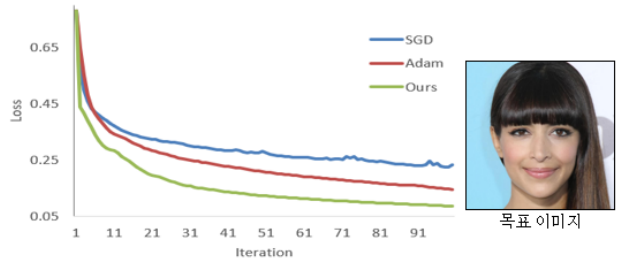
그림 3. 한 목표 이미지에 대한 기존 옵티마이저들과 본 논문의 결과 비교.

학습되었다. 본 논문에서 제시하는 네트워크를 학습하기 위한 학습 데이터 셋으로 5000 장의 합성 이미지를 생성하였고 테스트 데이터 셋은 CelebA-HQ 데이터 셋을 사용했다. 학습 데이터 셋에 대한 과적합 (overfitting)을 방지하기 위해서 학습 조기 종료 (early stopping)을 수행했다. 구체적으로 100 개의 학습 이미지마다 테스트 셋에 대한 로스를 계산하고 1000 번동안 로스의 감소가 없다면 조기 종료한다. 이 때 학습할 때와 달리 테스트 셋에 대해서는 정답을 알 수 없기 때문에 지도 학습 로스는 제외한다.

3.2 기존 옵티마이저들과의 비교

먼저, 테스트 셋에 대해 본 논문에서 제시하는 옵티마이저와 일반적으로 사용되는 옵티마이저들인 SGD, Adam 옵티마이저와의 비교 실험을 진행한다. 그림 3 (a)는 한 이미지를 StyleGAN 인버전 할 때 반복횟수에 따른 로스의 그래프를 나타낸다. 그림에서 볼 수 있듯이 학습 가능한 옵티마이저의 경우 기존 옵티마이저들에 비해 빠르게 로스가 감소한다. 그림 3 (b)는 반복 횟수 10 번마다 인버전의 중간 결과를 나타낸 것으로 시각적으로도 빠르게 인버전 됨을 보여준다.

테스트 셋 전체에 대한 정량적 비교를 위해 기존 옵티마이저를 100 번 반복한 결과에 본 네트워크가 얼마나 빠르게 도달할 수 있는지를 계산했다. 그 결과로 SGD 옵티마이저 대비 86%의 속도 향상이 있었으며, Adam 옵티마이저 대비 35%의 속도 향상을 보였다.



(a) 목표 이미지에 대한 최적화 로스 그래프

(b) Adam 옵티마이저의 중간 결과

(c) 본 논문의 중간 결과

그림 4. 학습된 인코더가 없는 환경에서의 기존 옵티마이저들과 본 논문의 결과 비교.

3.3 일반적인 StyleGAN 인버전

Algorithm1 에서 초기 벡터를 계산하기 위해 학습된 인코더를 사용했다. 하지만 많은 StyleGAN 인버전 문제에서 데이터 셋마다 인코더를 설계하고 학습하는 것은 상당한 시간을 요구하기 때문에 본 논문에서는 상대적으로 학습이 쉬운 옵티마이저 네트워크를 인코더가 없는 환경에서 학습하여 기존 옵티마이저와 비교하였다. 초기 벡터는 랜덤 샘플링을 통해 계산했다. 그림 4 (a)에서는 목표 이미지와 최적화 로스 그래프를 나타내었으며, 그림 4 (b)는 반복 수 20 번마다의 인버전 중간 결과를 나타낸다. 3.2 절과 마찬가지로 기존 옵티마이저들에 비해 높은 성능을 보이며 테스트 셋 전체에 대해 SGD 옵티마이저 대비 77%의 속도 향상이 있었으며, Adam 옵티마이저 대비 49% 의 속도 향상이 있었다. (100 번 반복 기준)

4. 결론

본 논문에서는 학습 가능한 옵티마이저 네트워크를 설계하여 보다 빠른 StyleGAN 인버전이 가능함을 보였다. 본 논문에서는 현재 잠재 벡터의 위치와 로스에 대한 기울기를 이용하여 최적의 업데이트 방향을 추정하였으며, 기존에 사용되는 옵티마이저들 대비 35% 이상의 속도 향상을 보였다.

감사의 글

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No. 2019-0-01906, 인공지능대학원지원(포항공과대학교))과 한국연구재단의 지원(No. 2020R1C1C1014863)을 받아 수행된 연구임.

참고문헌

[1] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[2] Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2019.

[3] Karras, Tero, et al. "Progressive growing of gans for improved quality, stability, and variation." arXiv preprint arXiv:1710.10196 (2017).

[4] Abdal, Rameen, Yipeng Qin, and Peter Wonka. "Image2stylegan: How to embed images into the stylegan latent space?." Proceedings of the IEEE international conference on computer vision. 2019.

[5] Abdal, Rameen, Yipeng Qin, and Peter Wonka. "Image2StyleGAN++: How to Edit the Embedded Images?." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[6] Zhu, Jiapeng, et al. "In-domain gan inversion for real image editing." arXiv preprint arXiv:2004.00049 (2020).

[7] Gong, Dong, et al. "Learning Deep Gradient Descent Optimization for Image Deconvolution." IEEE Transactions on Neural Networks and Learning Systems (2020).

[8] Andrychowicz, Marcin, et al. "Learning to learn by gradient descent by gradient descent." Advances in neural information processing systems. 2016.

[9] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).