Rel-GAN

2020.06.07

0 Abstract

- multi-domain image -to-image 가 최근 주목받고 있다.
- 이전의 기법에서는 이미지와 target attribute 을 입력으로 사용하고, 원하는 속성으로 출력하는 이미지를 사용한다.
- 여기에는 두가지 한계가 있다.
  - 이런 방법은 binary - valued attribute로 속성을 가정하므로, 세분화된 제어에 대해 만족스러운 결과를 얻을수 없다. (> 미세한 조정이 어렵다.)
  - 속성이 변경되지 않더라도 전체 대상 속성 집합을 지정해야한다? (속성이 변경되지 않으면 그대로 뭔가 되어야 하는데, 뭔가 지정해야하니까 문제다.)
- 우리는 이런 한계를 해결하기 위해 multi-domain image-to-image RelGAN을 제안한다.
- 핵심 아이디어는 **상대적 속성을 사용**하는것인데, 이것은 선택된 속성에 대해 원하는 변화를 설명한다.
  - The key idea is to use **relative attribute**s, which describes the desired change on selected attributes.
- 우리의 방법은 다를 속성은 보존하면서 관심있는 특정 속성을 연속적으로 변경하여 이미지를 수정할수 있다.
  - Our method is capable of modifying images by changing particular attributes of interest in a continuous manner while preserving the other attributes.
  -

1 Introduction

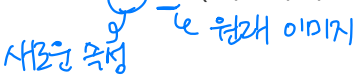
- Multi-domain image-to-image 변환은 한 도메인에서 다른 도메인으로 이미지를 변환하는것을 목표로 한다.
- 도메인은 속성집합으로 특정지어지며, 여기서 각 속성은 이미지에서 의미있는 부분이다.
- 최근 image-to-image 변환 문제는 GAN과 그것의 조건부 변형(conditional variants) 이후 상당한 주목을 받았다.
- 기존의 대부분의 방법은 두 도메인 사이의 image-to-image 변환에 초점을 맞추고 있지만, 최근에는 여러 속성을 동시에 변경할수 있는 여러 가지 **멀티도메인 방법**이 제안되고 있다.
  - 예를 들어 facial attribute editing 에서 머리색과 표현을 동시에 바꿀수 있는것.
- 최근의 multi-domain 방법의 인상적인 결과에도 불구하고, 두가지 한계가 있다.
  - 2진수 속성을 가정하므로, 보간용으로 설계되지 않는다. (not designed for attribute interpolation)
  - 만족스럽지 않는 결과가 나온다.

☆ 한계점 ☆

1. 만족스럽지 않은 결과
2. 보간용으로 설계 X --> 중간 값을 조절할수 없다!!

- **우리 모델은 추가 discriminators 을 통해 실제 가치의 상대적 속성에 대해 train 한다.**
    - Our model remedies this shortcoming by training on real-valued relative attributes with additional discriminators.
- 편집 전과 편집 후의 매끄럽고 사실적인 보간법은 각 속성의 강도를 세밀하게 제어할수 있기때문에 중요하다.
    - 갈색 대 금발 머리색의 비율, 행복의 정도를 세밀하게 제어할수 있다는 것.
- 둘째, 이런 방법은 속성의 부분집합만 조작하더라도 대상 도메인을 지정하기 위해 완전한 속성 표현이 필요하다.
    - 사용자가 변경되지 않은 각 속성의 기본 실제 값을 알지 못하므로 이는 세분화된 제어에 어려움을 제기한다.
    - > 원래는 변하지 않는 부분의 속성표현도 필요하다는 의미!

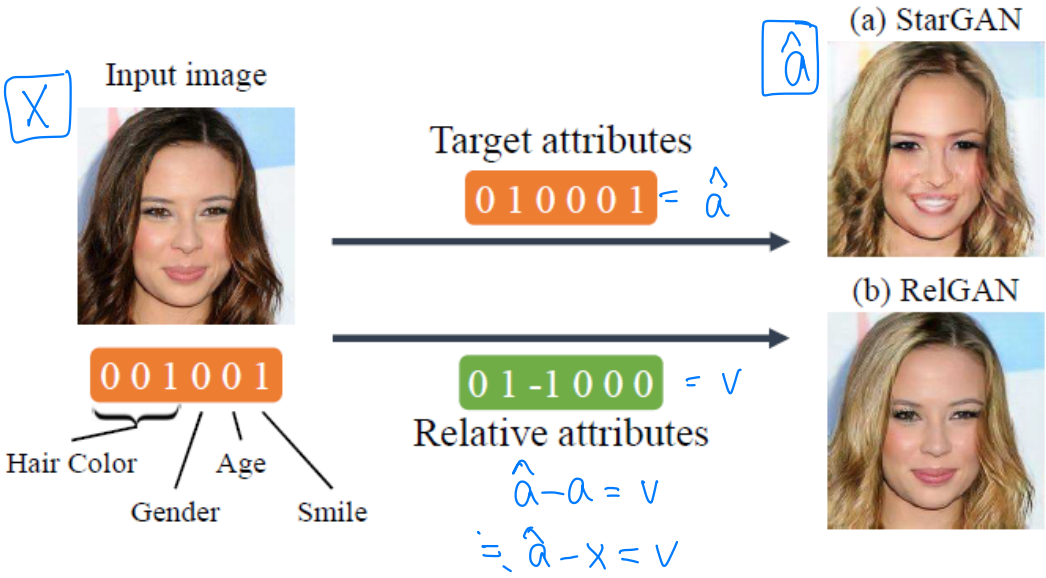
- 이런 문제를 극복하기 위해 우리의 핵심 아이디어는
    - 원래 이미지 x
    - 대상의 속성 a^ 쌍을 입력으로 하는 이전의 방식과 달리
    - 차이에 정의된 상대속성 v 를 입력으로 한다. (원래의 속성과 달라진 정도)
- 입력의 변화 (x,a^) ----> (x,v)
    - v = a^ -a (대상의 속성과 원래 이미지와의 차이)

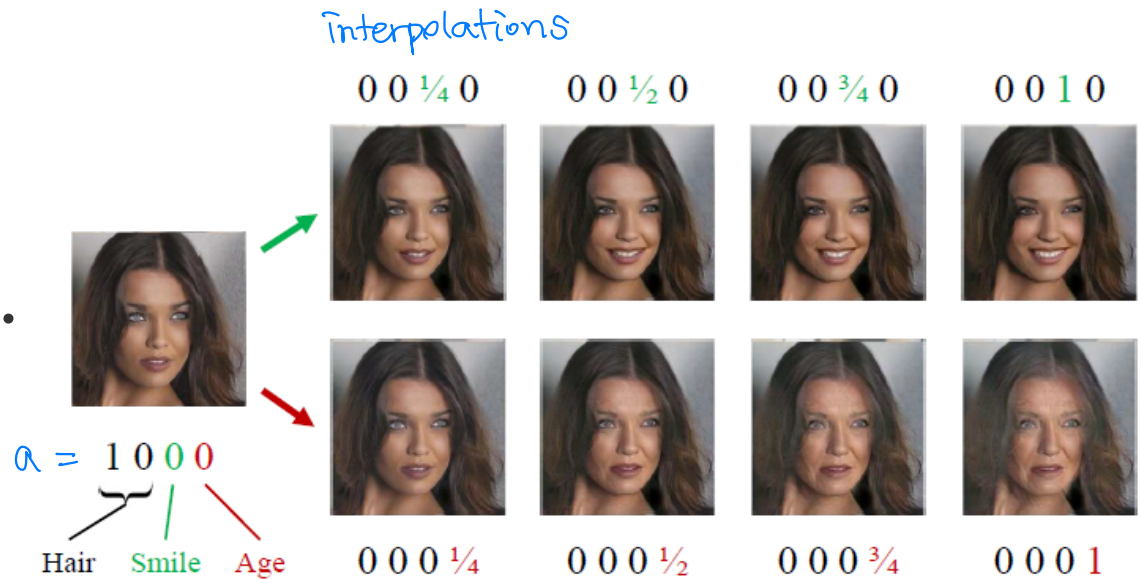


- RelGAN consists of a single generator G and three discriminators D = {DReal, DMatch, DInterp}
    - Generator 1개 Discriminator 3개
- G to learn to generate **(1) realistic images, (2) accurate translations in terms of the relative attributes, and (3) realistic interpolations.**

☆RelGan 의 가장 큰 특징

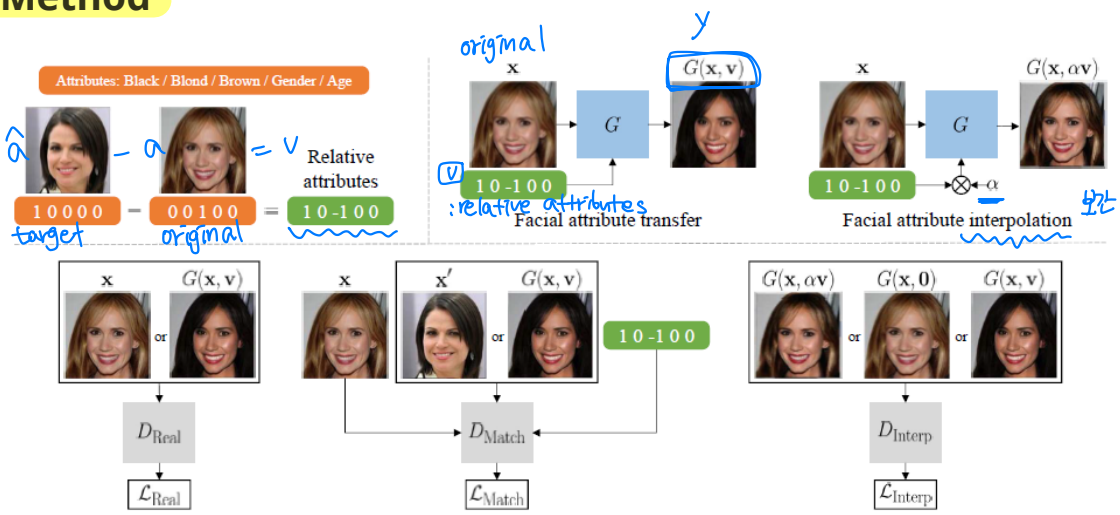
- 각 속성의 변화를 기반으로 하고, 입력 이미지의 전체 속성을 알 필요가 없다.
- a matching-aware discriminator : 입력 출력 쌍이 상대 속성과 일치하는지 판단
- an interpolation discriminator : 보간 품질 향상을 위한 보간판별기





-

2 Method



- 본 눈문에서는 도메인이 n-dimensional attribute vector a = [a (1), a(2), . . . , a(n)] T 로 특정지어지며, 여기서 a(i) 는각 속성 나이, 성별, 머리색을 의미한다.
- 우리의 목표는 입력 이미지 x를 출력이미지 y로 변환하여 y가 realistic 하게 보이게 하고, target 속성을 가지게 하는것이다.
  - 그럴듯하게 우리가 원하는 속성을 가지는 이미지 출력을 원한다!
- **사용자가 지정한 속성만 변하고, 다른 속성은 그래로 인것을 원한다!**
- v는 원하는 속성의 변화를 나타내는 상대 속성 벡터.

3.1 Relative atrributes

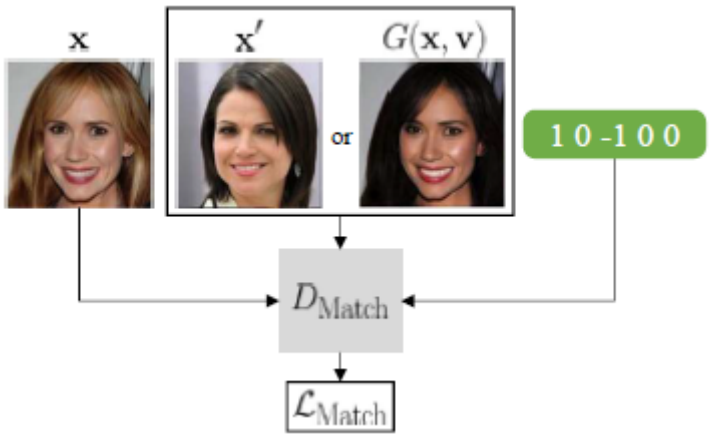
- x : image
- a : original domain
- a^ : target attribute vector
  - a와 a^ 는 n-dimensional vector
- v = a^ - a : relative vector
- y : output image
- 사용자의 editing requirement 를 상대적 속성으로 표현하는 것은 간단하고 직관적이라고 우리는 주장한다.

- 예를들어, image 속성이 binary-valued 인경우, 해당 속성표현값은 3가지(-1,0,1) 이며,
  - turn on 1
  - turn off -1
  - unchanged 0
- 우리는 상대적 속성이 사용자 요구사항을 encode하고 직관적인 의미를 갖는다는것을 알수 있다.
- 다음으로 facial attribute interpolation은 간단하다.
  - G(x, v), we simply apply G(x, αv), where α ∈ [0, 1] is an interpolation coefficient.

3-2. Adversarial Loss

- $$\min_G \max_{D_{\text{Real}}} \mathcal{L}_{\text{Real}} = \mathbb{E}_{\mathbf{x}} \left[\log D_{\text{Real}}(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x},\mathbf{v}} \left[\log(1 - D_{\text{Real}}(G(\mathbf{x}, \mathbf{v})))\right],$$
- 생성된 이미지와 실제 이미지를 구별할수 없도록 하기 위해.
- Generator 는 사실적으로 보이는 이미지를 생성하려고 노력한다.
- Discriminator는 실제 image 와 생성된 image 를 구별하는것을 목표로 한다.
- 실제 image 를 1로, 가짜 image 를 0으로 판단하는 능력임.

3-3. Conditional Adversarial Loss (그림으로 이해하자!)

- 

x - x' = v
x - v = G(x, v)

- **출력이미지 G(x,v) 가 현실적으로 보이는것 뿐 아니라, x와 G(x,v) 의 차이가 상대 속성 v 와 일치해야한다.**
  - 출력이미지를 만들때, 원래이미지 x와 v 만큼 차이가 나게 만들었으니까 x와 G(x,v) 의 차이를 v로 만들고 싶어한다!!
- 이 requirement를 달성하기 위해 conditional discriminator 개념을 이용한다.
- D match 는 input으로 이미지와 조건 변수 쌍(x,v) 를 입력을 도입한다. Dmatch는
- $$\min_G \max_{D_{\text{Match}}} \mathcal{L}_{\text{Match}} = \mathbb{E}_{\mathbf{x},\mathbf{v},\mathbf{x}'} \left[\log D_{\text{Match}}(\mathbf{x}, \mathbf{v}, \mathbf{x}')\right] + \mathbb{E}_{\mathbf{x},\mathbf{v}} \left[\log(1 - D_{\text{Match}}(\mathbf{x}, \mathbf{v}, G(\mathbf{x}, \mathbf{v})))\right].$$ → 이걸 1로 → 이걸 0으로 판단하는 loss
- DMatch는 잘못된 트리플릿을 추가함으로써 진짜 트리플릿을 +1(실제 및 매치)으로 분류하고, 가짜 트리플릿과 잘못된 트리플릿을 모두 -1(가짜 또는 불일치)으로 분류하려고 한다.
- 특히 다음과 같은 간단한 절차를 사용하여 잘못된 트리플릿을 만든다: (x, a0 - a, x0)로 표현되는 실제 트리플릿을 주어진다면, 이 네 변수 중 하나를 새로운 변수로 교체하여 잘못된 트리플릿을 만든다.

Algorithm 1 Conditional adversarial loss

function MATCH_LOSS($x_1, x_2, x_3, a_1, a_2, a_3$)

$v_{12}, v_{32}, v_{13} \leftarrow a_2 - a_1, a_2 - a_3, a_3 - a_1$

$s_r \leftarrow D_{\text{Match}}(x_1, v_{12}, x_2)$ {real triplet}

$s_f \leftarrow D_{\text{Match}}(x_1, v_{12}, G(x_1, v_{12}))$ {fake triplet}

$s_{w_1} \leftarrow D_{\text{Match}}(x_3, v_{12}, x_2)$ {wrong triplet}

$s_{w_2} \leftarrow D_{\text{Match}}(x_1, v_{32}, x_2)$ {wrong triplet}

$s_{w_3} \leftarrow D_{\text{Match}}(x_1, v_{13}, x_2)$ {wrong triplet}

$s_{w_4} \leftarrow D_{\text{Match}}(x_1, v_{12}, x_3)$ {wrong triplet}

$\mathcal{L}^D_{\text{Match}} \leftarrow (s_r - 1)^2 + s_f^2 + \sum_{i=1}^{4} s_{w_i}^2$

$\mathcal{L}^G_{\text{Match}} \leftarrow (s_f - 1)^2$

return $\mathcal{L}^D_{\text{Match}}, \mathcal{L}^G_{\text{Match}}$

+1

-1

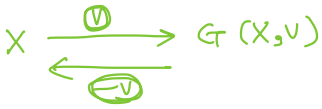
↳ 잘못된 triplet을 만드는 방법

- 여기서 왜 잘못된 트리플릿을 만드는건지,,,, 모르겠음,,,ㅠ

3.4. Reconstruction Loss

- unconditional , conditional adversarial loss를 최소화 함으로써, G(x,v) 가 realistic 하게 보이고, x와 G(x,v)의 차이가 상대 속성 v 와 일치하도록 출력 이미지 G(x,v) 를 생성하도록 학습한다.
- 하지만, G가 낮은 수준(배경)에서 높은 수준(얼굴)에 이르는 다른 모든 측면을 보존하면서 그러한 속성관련 내용만 수정한다는 보장은 없다.
  - G가 다른 모든 특징들은 보존하면서, attribute related contents만 수정한다는 보장이 없다.

- 이 문제를 완화하기 위해, cycle-reconstruction loss, self-reconstruction loss를 제안한다.
- **Cycle-reconstruction loss** : 한바퀴 돌고오면 다시 나



  - $\min_G \mathcal{L}_{\text{Cycle}} = \mathbb{E}_{\mathbf{x},\mathbf{v}} \left[\| G(G(\mathbf{x}, \mathbf{v}), -\mathbf{v}) - \mathbf{x} \|_1 \right].$ 다시 나오면 X
- **Self - reconstruction loss** : v=0 인 경우(수정사항이 없는경우) 자기 자신이 나와야 한다.
  - $\min_G \mathcal{L}_{\text{Self}} = \mathbb{E}_{\mathbf{x}} \left[\| G(\mathbf{x}, 0) - x \|_1 \right],$

3.5 Interpolation Loss

- G(x, v) --> G(x, av)
  - a 는 interpolation coefficient
- high-quality 의 interpolation을 하기위해 , 일단 생성된 사진이 진짜 사진 같아야한다.
- G(x, v)과 G(x,av), G(x,0) 을 구별할수 없게 하는것을 목표로한다.
- 이를위해 G와 경쟁할 세번째 D가 등장한다!
  - D 의 목적은 생성된 영상을 입력으로 가져와서 보간정도(a) 를 예측하는것이다.
  - a = 0 (보간없음), a=0.5 (보간 최대치)

- $$\min_{D_{\text{Interp}}} \mathcal{L}^D_{\text{Interp}} = \mathbb{E}_{\mathbf{x},\mathbf{v},\alpha}[\,\|D_{\text{Interp}}(G(\mathbf{x},\alpha\mathbf{v})) - \hat{\alpha}\|^2 + \|D_{\text{Interp}}(G(\mathbf{x},0))\|^2 + \|D_{\text{Interp}}(G(\mathbf{x},\mathbf{v}))\|^2],$$
  -
- $$\min_{G} \mathcal{L}^G_{\text{Interp}} = \mathbb{E}_{\mathbf{x},\mathbf{v},\alpha}\left[\|D_{\text{Interp}}(G(\mathbf{x},\alpha\mathbf{v}))\|^2\right],$$
  - G는 D가 G(x,av) 가 non-interpolate 하다고 예측하게 만들고 싶어함.
-
- $$\min_{D_{\text{Interp}}} \mathcal{L}^D_{\text{Interp}} = \mathbb{E}_{\mathbf{x},\mathbf{v},\alpha}[\,\|D_{\text{Interp}}(G(\mathbf{x},\alpha\mathbf{v})) - \hat{\alpha}\|^2 + \|D_{\text{Interp}}(G(\mathbf{x},\mathbb{I}[\alpha > 0.5]\mathbf{v}))\|^2],$$

Algorithm 2 Interpolation loss

function INTERP_LOSS($\mathbf{x}$, $\mathbf{v}$)
  $\alpha \sim \mathrm{U}(0,1)$
  $y_0 \leftarrow D_{\text{Interp}}(G(\mathbf{x},0))$ {non-interpolated image}
  $y_1 \leftarrow D_{\text{Interp}}(G(\mathbf{x},\mathbf{v}))$ {non-interpolated image}
  $y_\alpha \leftarrow D_{\text{Interp}}(G(\mathbf{x},\alpha\mathbf{v}))$ {interpolated image}
- **if** $\alpha \leq 0.5$ **then**
    $\mathcal{L}^D_{\text{Interp}} \leftarrow y_0^2 + (y_\alpha - \alpha)^2$
  else
    $\mathcal{L}^D_{\text{Interp}} \leftarrow y_1^2 + (y_\alpha - (1-\alpha))^2$
  $\mathcal{L}^G_{\text{Interp}} \leftarrow y_\alpha^2$
  return $\mathcal{L}^D_{\text{Interp}}, \mathcal{L}^G_{\text{Interp}}$

3.6 Full - Loss

3.6. Full Loss

To stabilize the training process, we add orthogonal regularization [17] into our loss function. Finally, the full loss function for $D = \{D_{\text{Real}}, D_{\text{Match}}, D_{\text{Interp}}\}$ and for G are expressed, respectively, as

$$\min_D \mathcal{L}^D = -\mathcal{L}_{\text{Real}} + \lambda_1 \mathcal{L}^D_{\text{Match}} + \lambda_2 \mathcal{L}^D_{\text{Interp}} \tag{9}$$

and

$$\begin{aligned} \min_G \mathcal{L}^G = \mathcal{L}_{\text{Real}} + \lambda_1 \mathcal{L}^G_{\text{Match}} + \lambda_2 \mathcal{L}^G_{\text{Interp}} \\ + \lambda_3 \mathcal{L}_{\text{Cycle}} + \lambda_4 \mathcal{L}_{\text{Self}} + \lambda_5 \mathcal{L}_{\text{Ortho}}, \end{aligned} \tag{10}$$

where λ_1, λ_2, λ_3, λ_4, and λ_5 are hyper-parameters that control the relative importance of each loss.