Style Transfer

염지현

CONTENTS

2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

- Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

1. 해결하고자 하는 문제점 및 해결 방법

- 해결하고자 하는 문제점
    1. Paired dataset 수집의 한계
    2. Cycle consistency loss의 한계
    3. Multi-style APDrawings

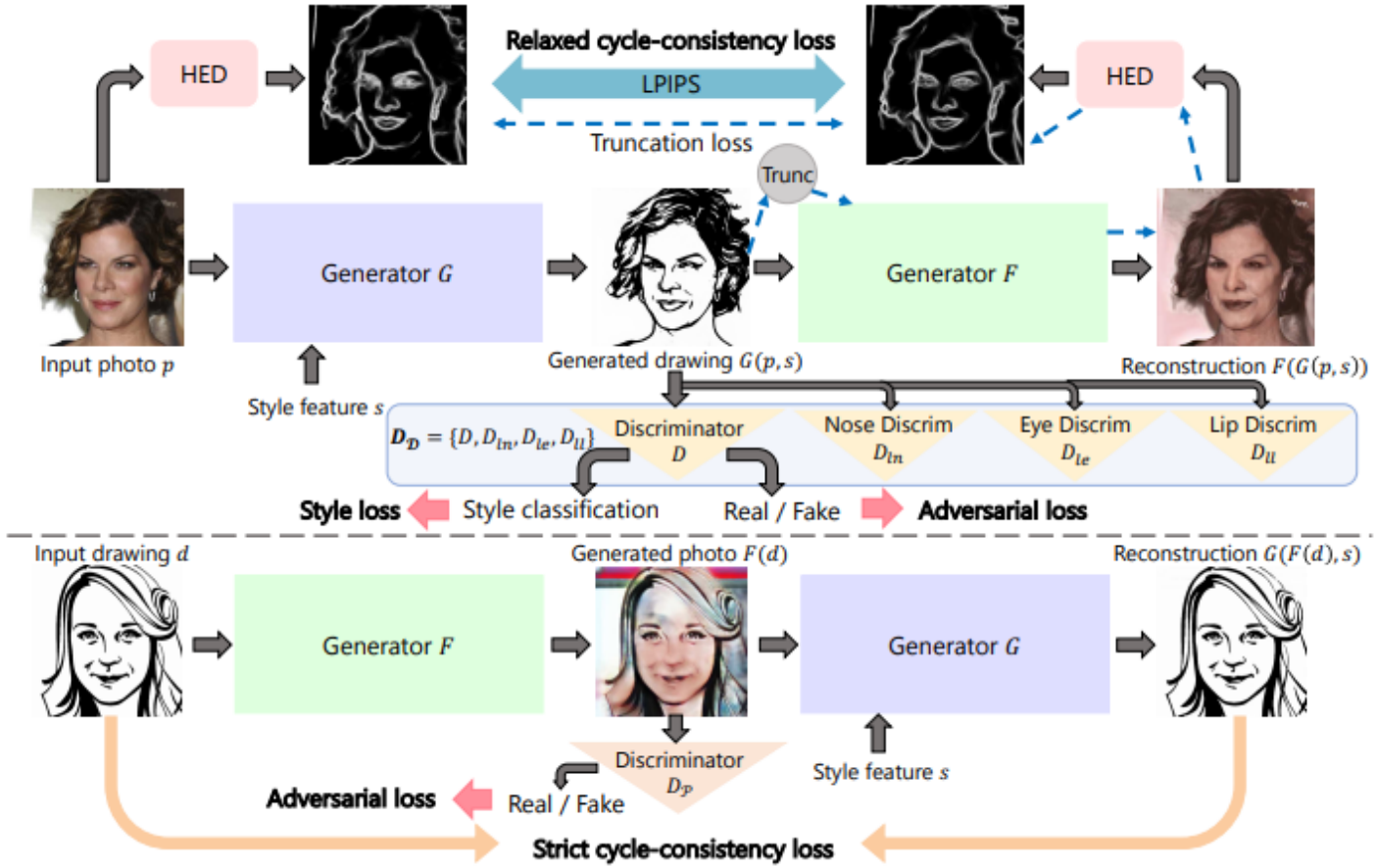
- 해결 방법
    1. Unpaired dataset 수집
    2. Asymmetric Cycle Mapping

2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

– Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

2. 모델 구조



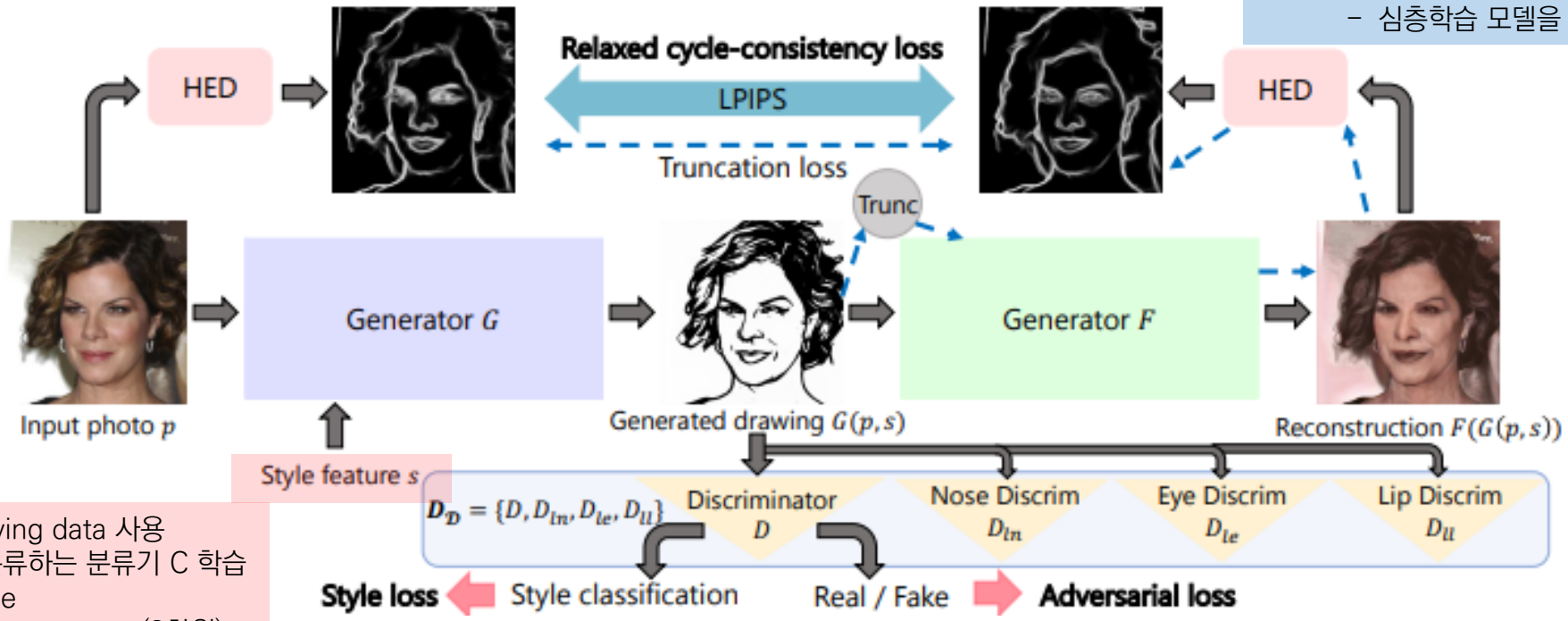
2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

- Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

2. 모델 구조

- HED(Holistically-Neted Edge Detection)
  - 심층학습 모델을 통해 edge detection



- Tagged web drawing data 사용
- 세 가지 스타일로 분류하는 분류기 C 학습
- Input : style image
- Output : style feature vector(3차원)

- D_D
  - Adversarial loss
    - Input : Real drawing image d, Generated drawing G(p,s)
    - Output : Real/Fake
  - Style loss
    - 특정 drawing이 들어왔을 때, 해당 drawing이 style c인 entropy 값 계산

2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

- Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

2. 모델 구조

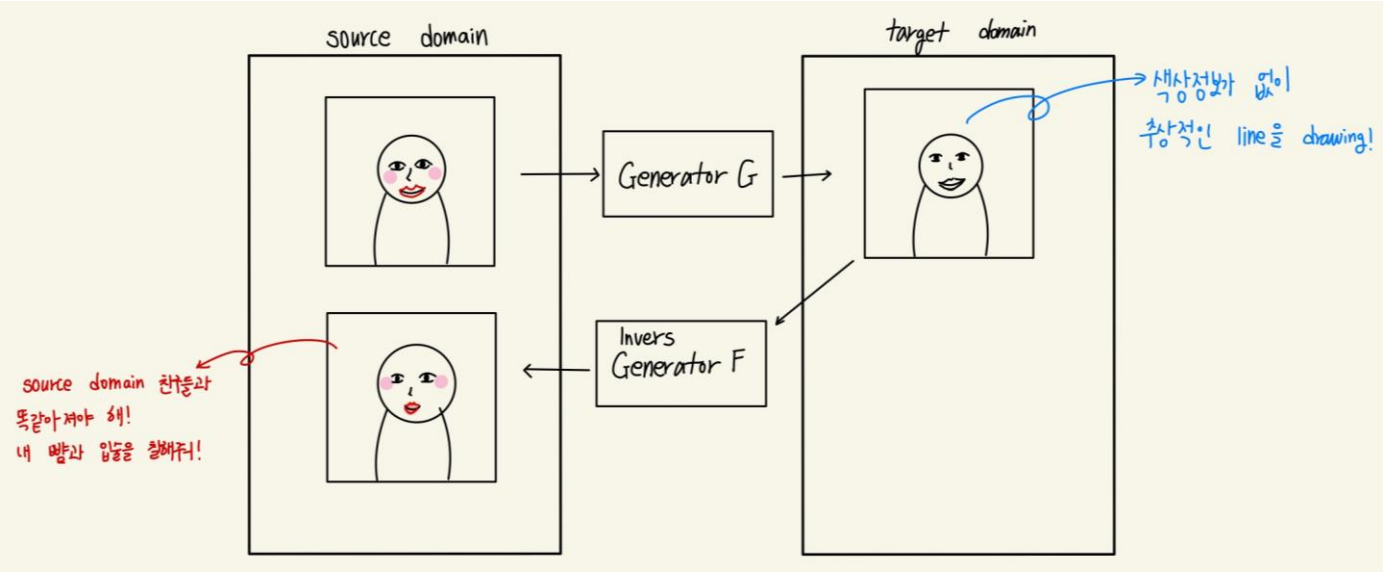


Photo to drawing 시 cycle consistency loss의 한계

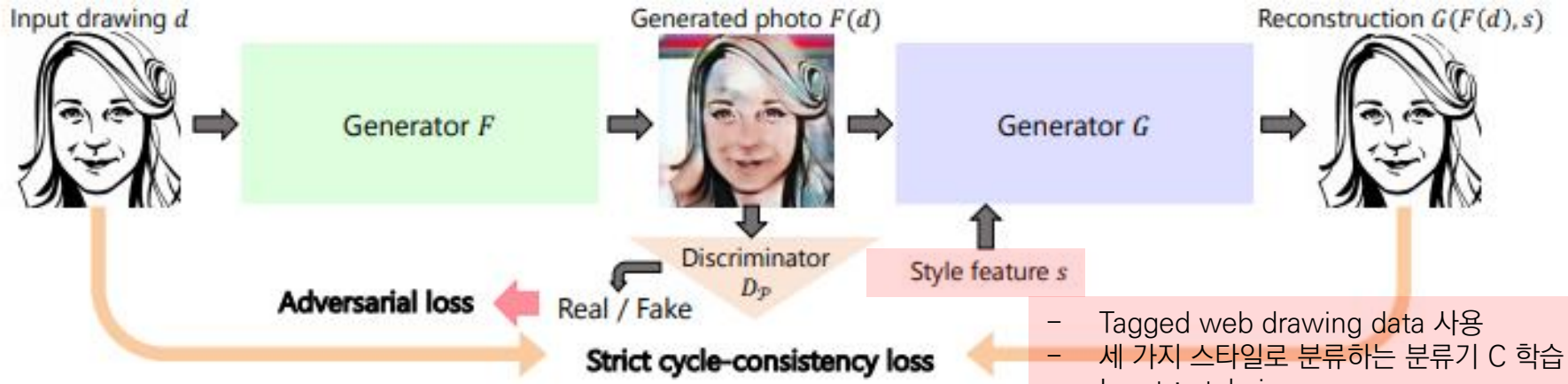
- Generated image는 색상 정보 없이 추상적인 line을 따는 것이 목적
- Reconstructed image는 real photo와 비슷해지는 것이 목적
- 즉, cycle consistency loss가 감소하기 위해서는 reconstructed image에 색상 정보가 포함되어 있어야 하며, 이를 위해서는 Generated image에도 색상 정보가 포함되어야 함
- 따라서 Generated image가 APDrawing이 아닌, real photo에 가까운 output을 도출할 수 있기 때문에 본 논문에서는 relaxed-consistency loss 사용

2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

- Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

2. 모델 구조



- Dp
  - Pixel by pixel을 비교
- Strict cycle-consistency loss
  - Pixel by pixel 비교

- Tagged web drawing data 사용
- 세 가지 스타일로 분류하는 분류기 C 학습
- Input : style image
- Output : style feature vector(3차원)

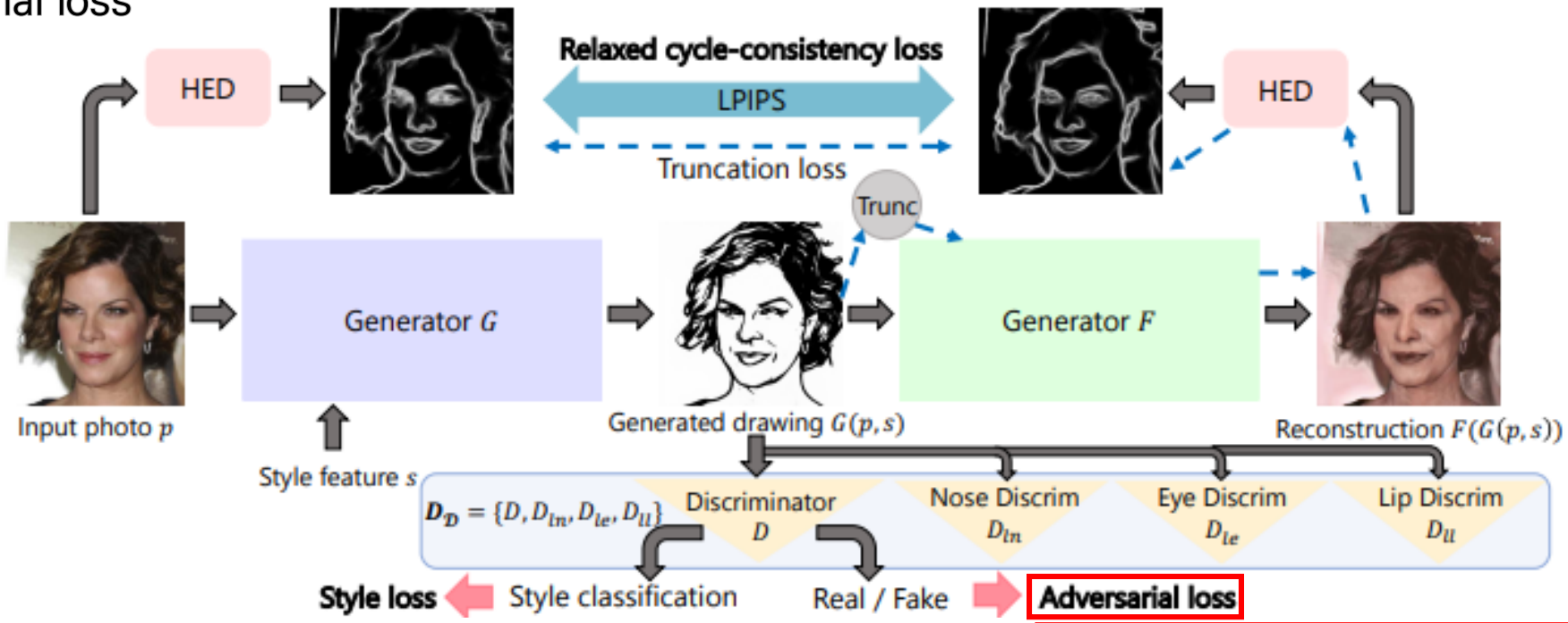
2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

– Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

3. Loss function

3-1. Adversarial loss



$$L_{adv}(G, D_{\mathcal{D}}) = \sum_{D \in D_{\mathcal{D}}} \mathbb{E}_{d \in S(d)}[\log D(d)] + \sum_{D \in D_{\mathcal{D}}} \mathbb{E}_{p \in S(p)}[\log(1 - D(G(p,s))]$$

1. Adversarial loss
   - Real drawing과 fake drawing에 올바른 label을 할당하는가

2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

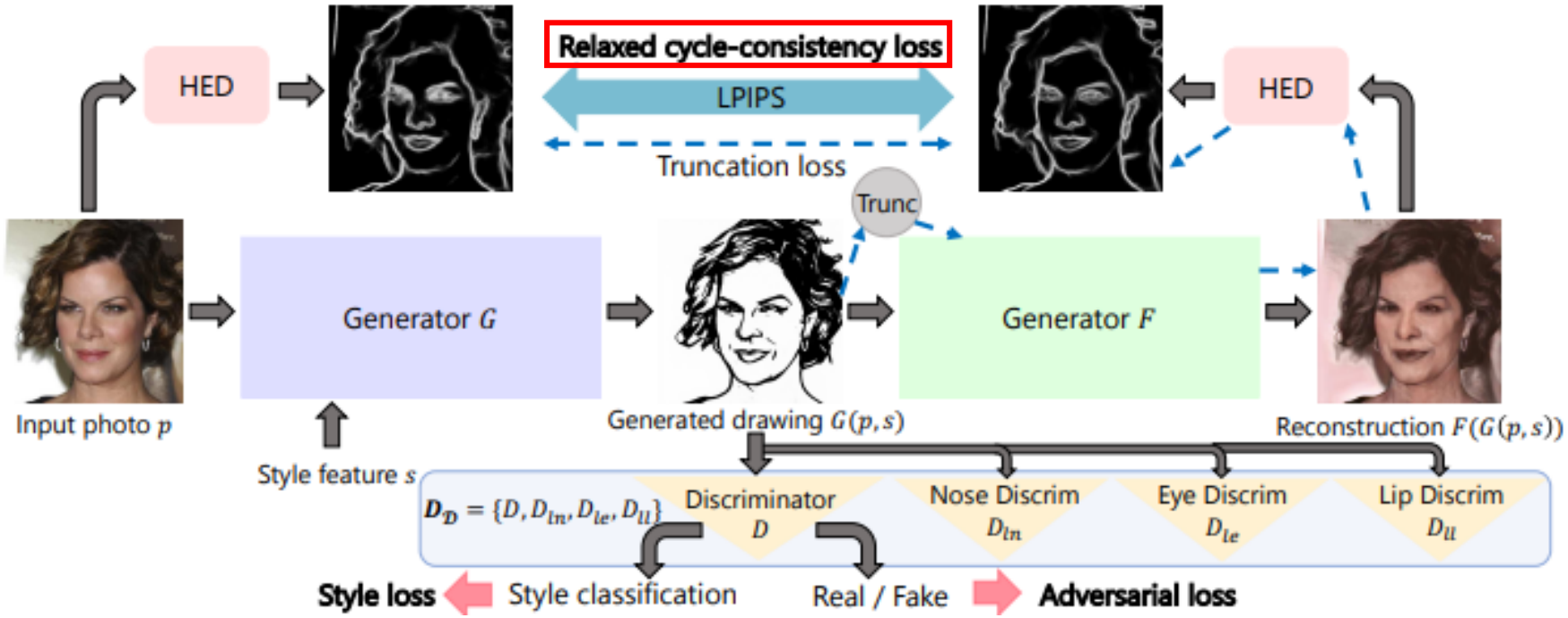
– Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

3. Loss function

3-2. Relaxed forward cycle-consistency loss

$$L_{relaxed-cyc}(G,F) = \mathbb{E}_{p\in S(p)}[L_{lpips}(H(p), H(F(G(p,s))))] \quad (4)$$

2. Relaxed forward cycle-consistency loss
- Real photo와 F(G(p,s))의 edge가 얼마나 유사한가
- HED edge detection을 통해 real photo, F(G(p,s)) edge detection
- LPIPS metric을 통해 edge 유사성 확인



2020
Style Transfer

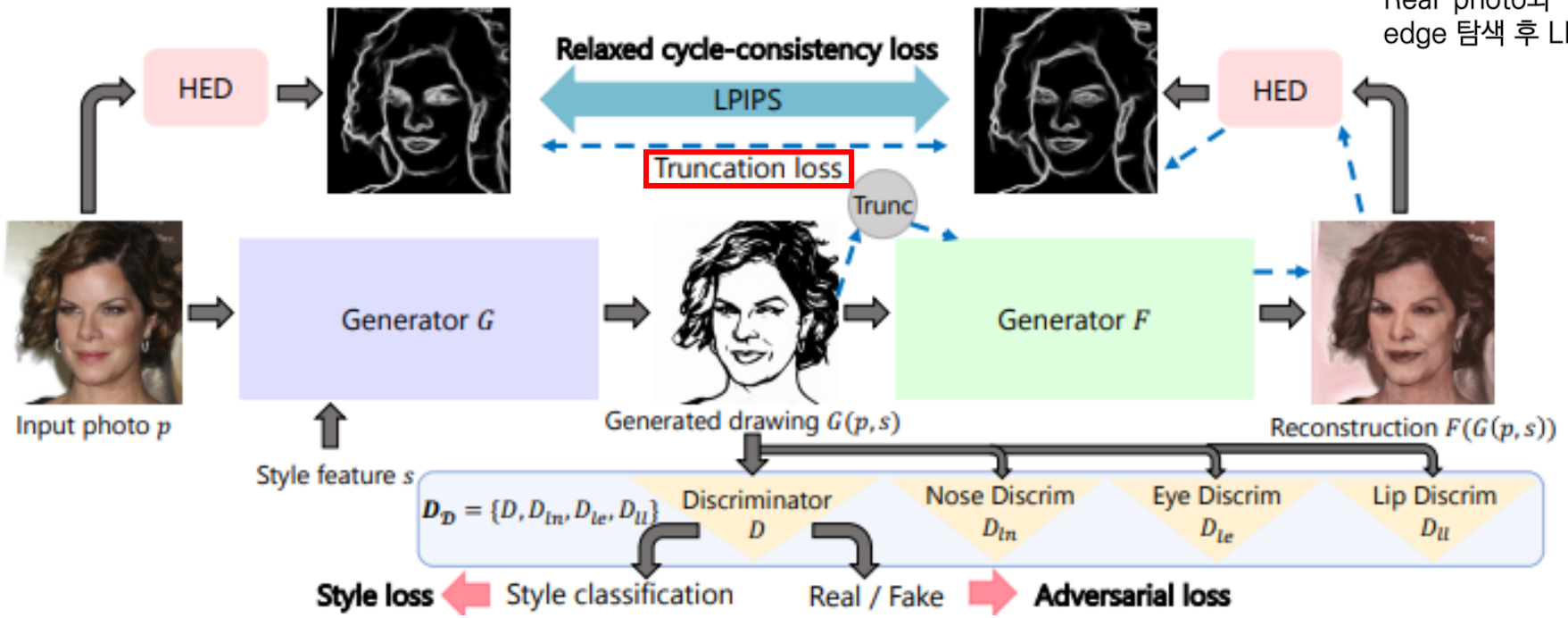
Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

– Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

3. Loss function

3-3. Truncation loss

$$L_{trunc}(G,F) = \mathbb{E}_{p\in S(p)}[L_{lpips}(H(p), H(F(T[G(p,s)])))] \quad (6)$$



3. Truncation loss
- Relaxed forward cycle-consistency loss와 동일한 format
- T[ ] 라는 truncation operation 사용
  → 생성된 G(p,s)의 인코딩 된 정보가 선명하게 보이도록 6비트를 잘라내는 연산을 진행
- 진행 후 F에 입력하여 photo로 재구성
- Real photo와 truncation operation을 거친 재구성 photo의 edge 탐색 후 LIPIS로 유사성 비교

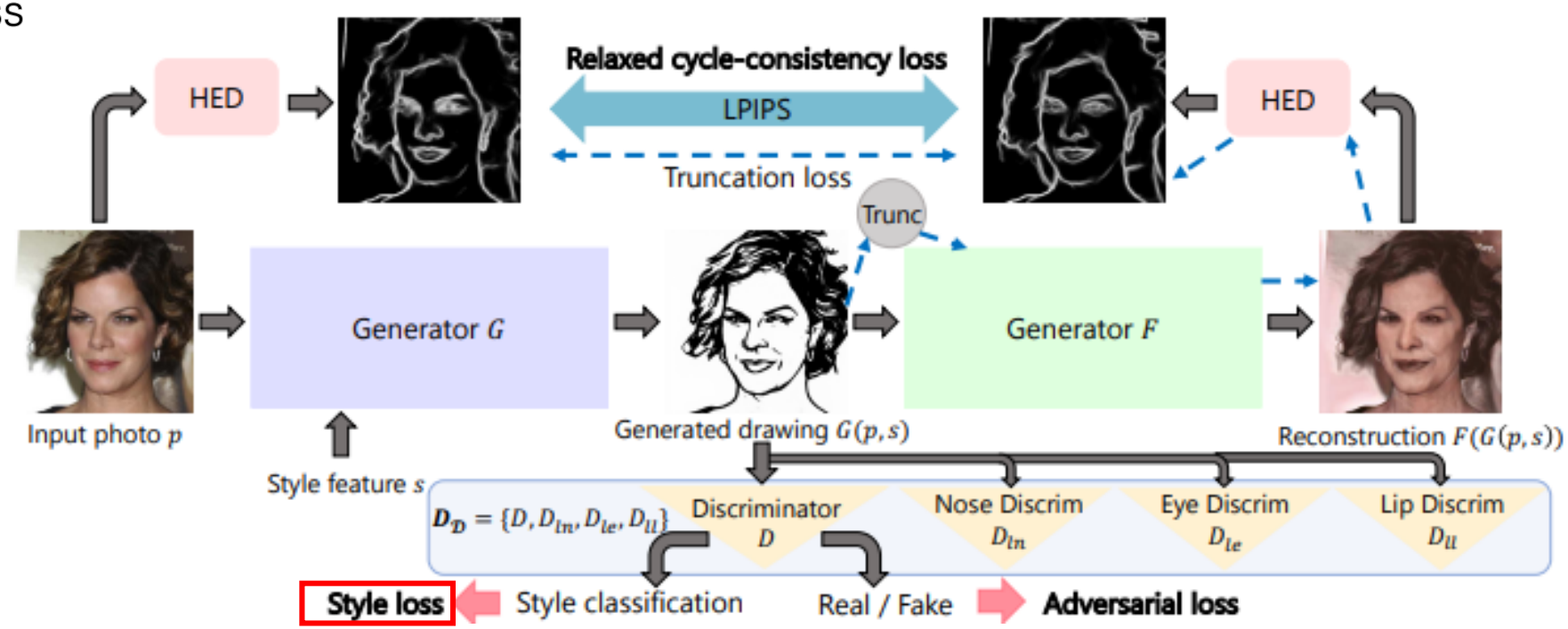
2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

– Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

3. Loss function

3-4. Style loss



$$L_{cls}(G, D_{\mathcal{D}}) = \mathbb{E}_{d \in S(d)}[-\sum_{c} p(c) \log D_{cls}(c|d)]$$
$$+ \mathbb{E}_{p \in S(p)}[-\sum_{c} p'(c) \log D_{cls}(c|G(p,s))]$$

4. Style loss
- G가 주어진 style feature에 가까운 drawing을 generate 하도록 유도
- 주어진 drawing이 style c인 확률 x d가 c로 예측될 softmax 확률
- 생성된 도면 G(p,s)가 style c인 확률 x G(p,s)가 c로 예측될 softmax 확률
- Entropy : 해당 image를 넣었을 때 결과의 무질서도 측정 → 낮을 수록 확률이 높은 것 이기 때문에 – 기호를 붙임

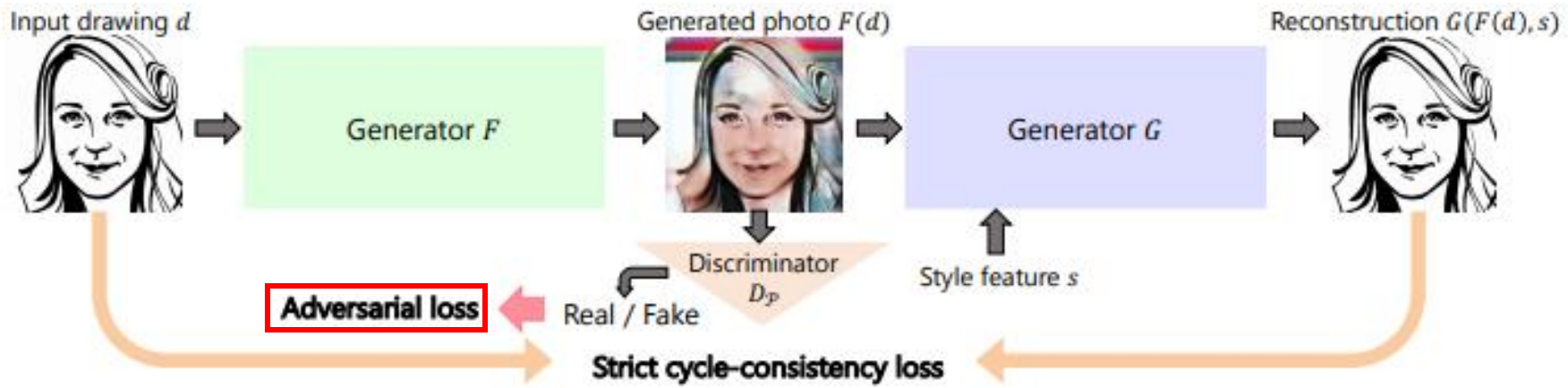
2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

- Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

3. Loss function

3-1. Adversarial loss



$$L_{adv}(F, D_{\mathcal{P}}) = \mathbb{E}_{p \in S(p)}[\log D_{\mathcal{P}}(p)] + \mathbb{E}_{d \in S(d)}[\log(1 - D_{\mathcal{P}}(F(d))]$$

1. Adversarial loss
- Real photo와 inverse photo(fake photo)에 올바른 label을 할당하는가

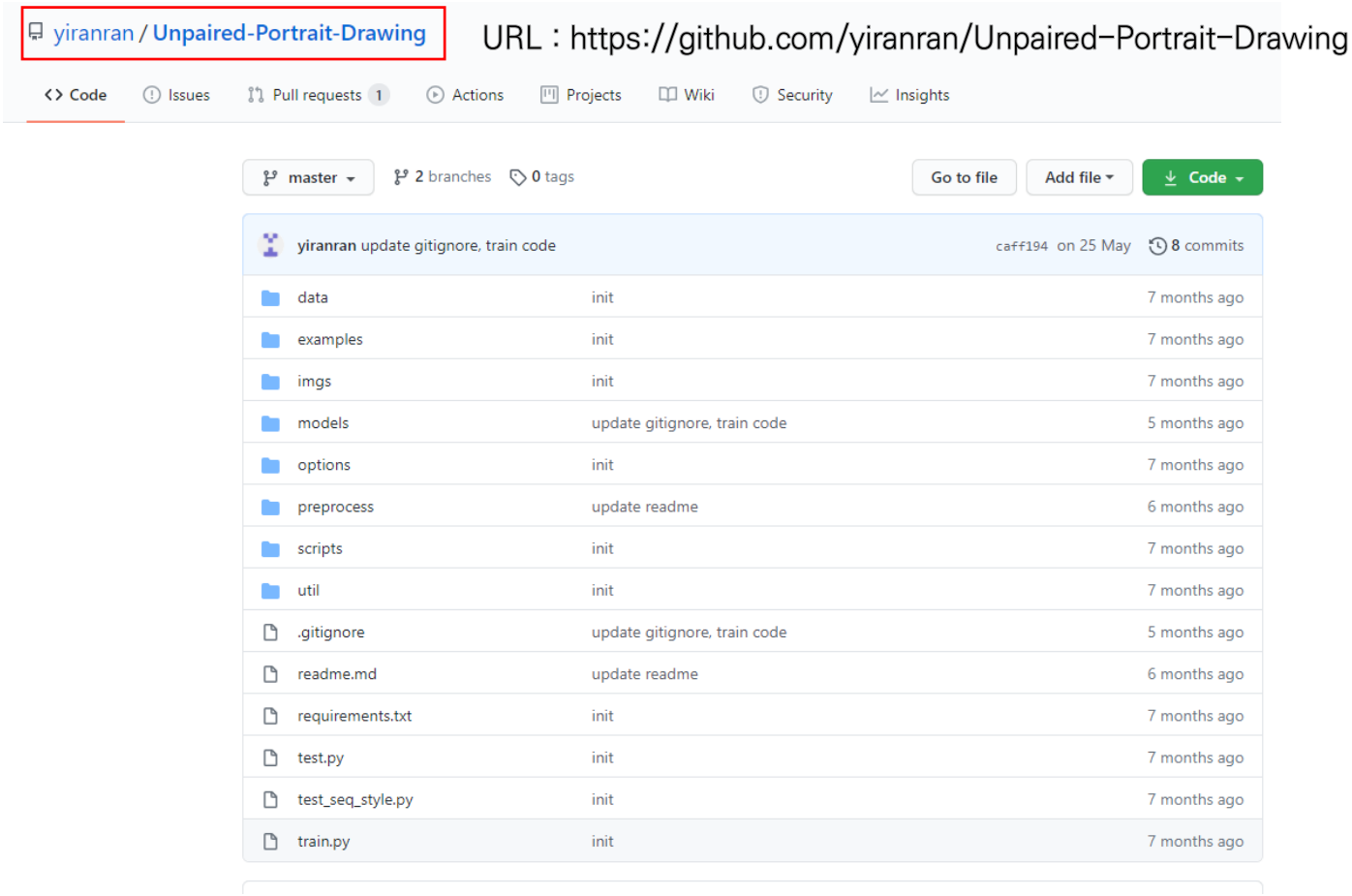
2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

- Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

4. Source code

yiranran / Unpaired-Portrait-Drawing URL : https://github.com/yiranran/Unpaired-Portrait-Drawing



2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

– Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

4. Feedback – 1. Generator 구조

```
if netG == 'resnet_9blocks':
        net = ResnetGenerator(input_nc, output_nc, ngf, norm_layer=norm_layer, use_dropou
elif netG == 'resnet_style2_9blocks':
        net = ResnetStyle2Generator(input_nc, output_nc, ngf, norm_layer=norm_layer, use_
elif netG == 'resnet_6blocks':
        net = ResnetGenerator(input_nc, output_nc, ngf, norm_layer=norm_layer, use_dropou
elif netG == 'unet_128':
        net = UnetGenerator(input_nc, output_nc, 7, ngf, norm_layer=norm_layer, use_dropo
elif netG == 'unet_256':
        net = UnetGenerator(input_nc, output_nc, 8, ngf, norm_layer=norm_layer, use_dropo
else:
        raise NotImplementedError('Generator model name [%s] is not recognized' % netG)
return init_net(net, init_type, init_gain, gpu_ids)
```

```
model = [nn.ReflectionPad2d(3),
                nn.Conv2d(input_nc, ngf, kernel_size=7, padding=0, bias=use_bias),
                norm_layer(ngf),
                nn.ReLU(True)]

n_downsampling = 2
for i in range(n_downsampling):  # add downsampling layers
        mult = 2 ** i
        model += [nn.Conv2d(ngf * mult, ngf * mult * 2, kernel_size=3, stride=2, padding=1, bias=use_bias),
                        norm_layer(ngf * mult * 2),
                        nn.ReLU(True)]

mult = 2 ** n_downsampling
for i in range(n_blocks):       # add ResNet blocks

        model += [ResnetBlock(ngf * mult, padding_type=padding_type, norm_layer=norm_layer, use_dropout=use_dropout, use_bias=use_bias)]

for i in range(n_downsampling):  # add upsampling layers
        mult = 2 ** (n_downsampling - i)
        model += [nn.ConvTranspose2d(ngf * mult, int(ngf * mult / 2),
                                                                kernel_size=3, stride=2,
                                                                padding=1, output_padding=1,
                                                                bias=use_bias),
                        norm_layer(int(ngf * mult / 2)),
                        nn.ReLU(True)]
model += [nn.ReflectionPad2d(3)]
model += [nn.Conv2d(ngf, output_nc, kernel_size=7, padding=0)]
model += [nn.Tanh()]

self.model = nn.Sequential(*model)
```

2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

- Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

4. Feedback – 2. loss function discriminator 값 확인하기

2020
Style Transfer

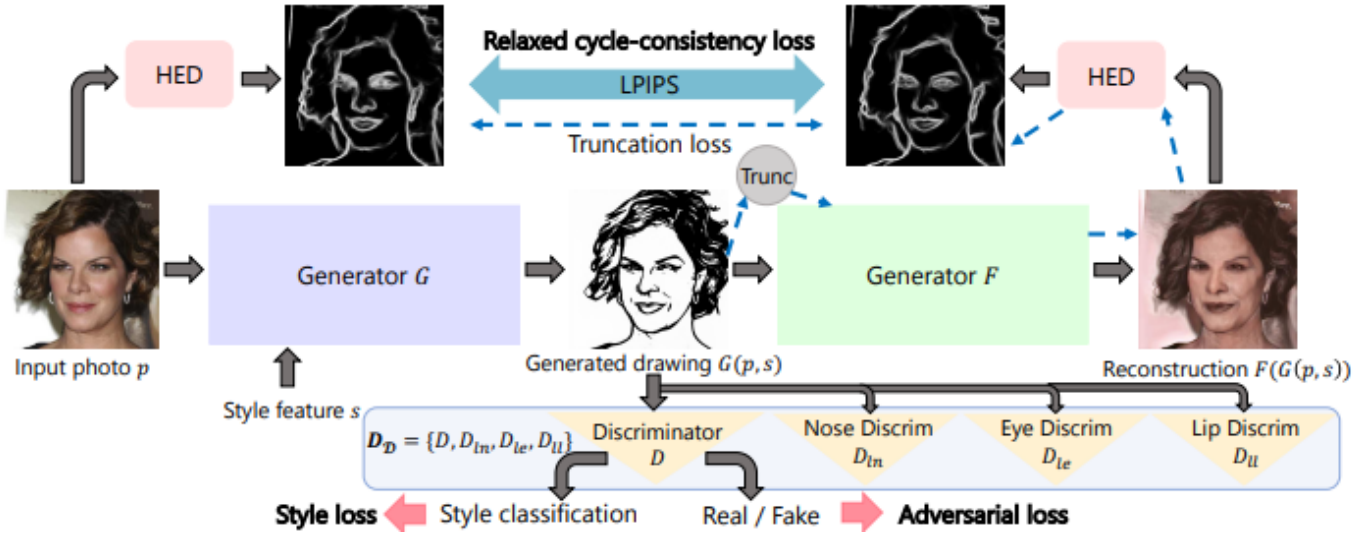
Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

– Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

4. Feedback – 3. relaxed cycle consistency loss와 truncation loss의 차이 및 각 loss의 input image

$$L_{trunc}(G,F) = \mathbb{E}_{p\in S(p)}[L_{lpips}(H(p), H(F(T[G(p,s)])))] \quad (6)$$

$$L_{relaxed-cyc}(G,F) = \mathbb{E}_{p\in S(p)}[L_{lpips}(H(p), H(F(G(p,s))))] \quad (4)$$



2020
Style Transfer

Unpaired Portrait Drawing Generation via Asymmetric Cycle Mapping

- Ran Yi, Yong-Jin Liu, Yu-Kun Lai, Paul L. Rosin

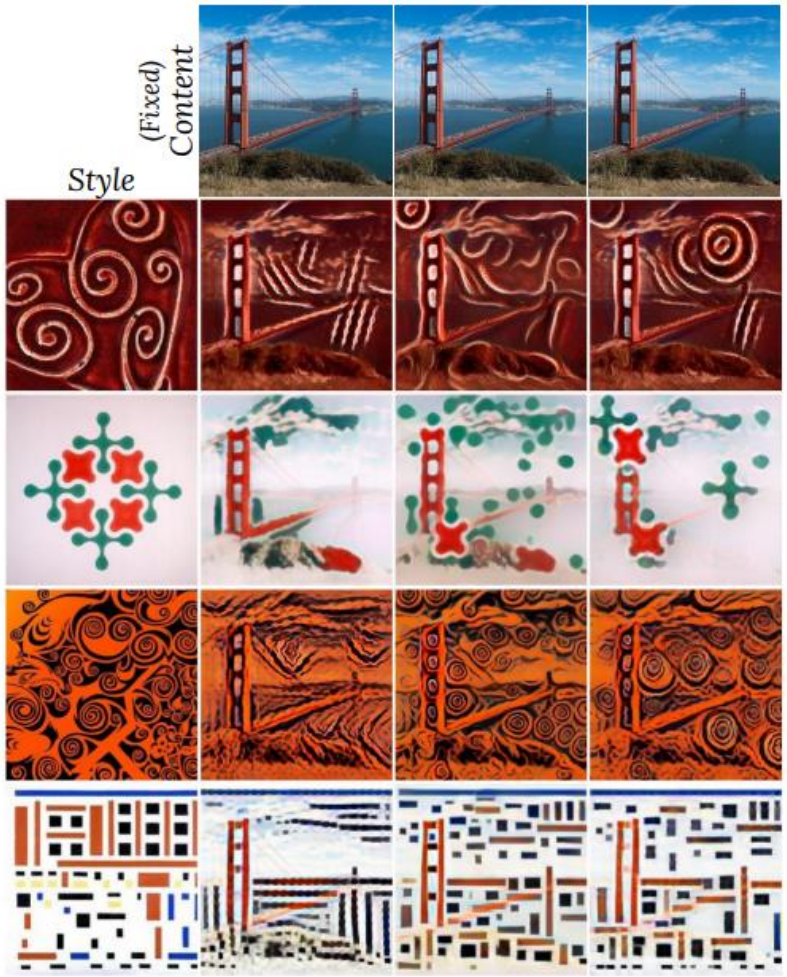
4. Feedback – 3. p'의 의미

→ p'(c) : label이 c일 확률

2020
Style Transfer

Adjustable real-time style transfer

– Mohammad Babaeizadeh, Golnaz Ghiasi



1. 해결하고자 하는 문제점 및 해결 방법

– 해결하고자 하는 문제점

1. Synthesized output을 control 할 수 없음
2. 기존 모델은 한 쌍의 style/content images가 주어지면 single style transfer만 가능
3. 사용자가 선호하는 output을 도출하기 위해 다양한 모델을 사용하거나 여러 하이퍼파라미터 조절 후 재학습

– 해결 방법

1. 수동으로 조정 가능한 parameter set을 통해 training 후에 real-time 으로 중요한 hyper-parameters 조정
   parameter set : additional parameter 사용

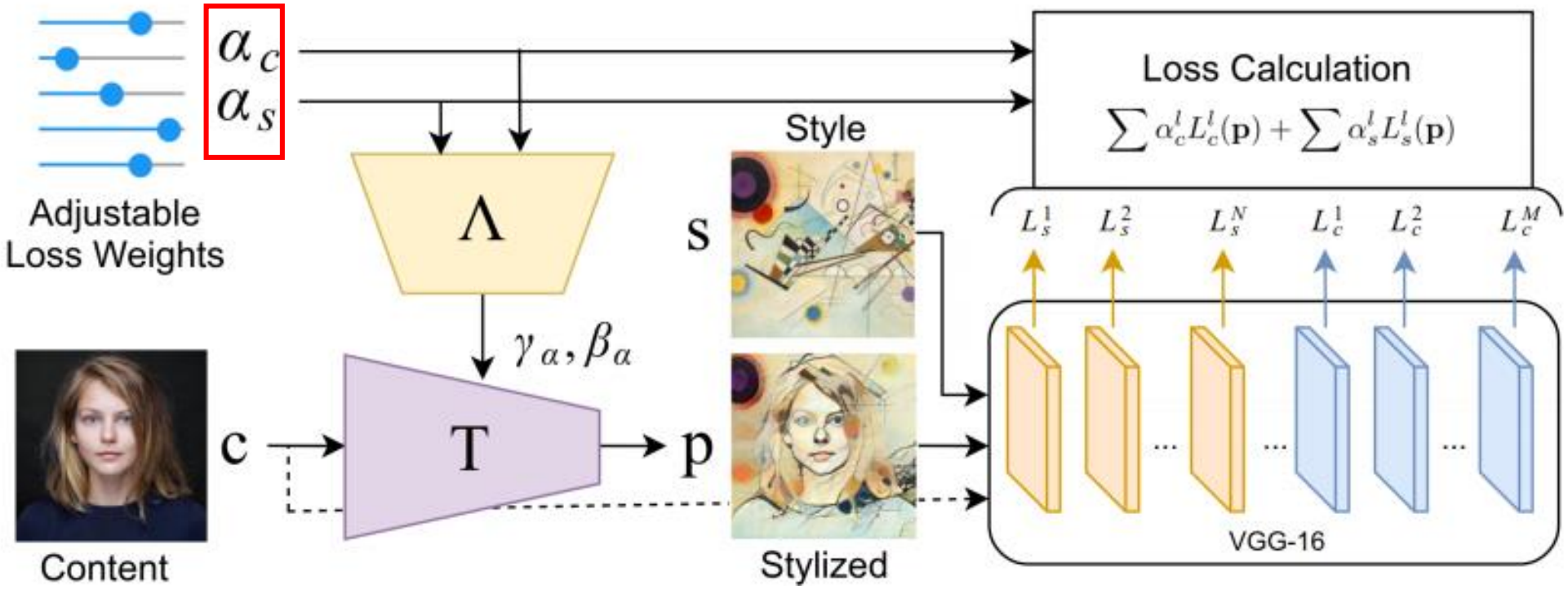
2020
Style Transfer

Adjustable real-time style transfer

– Mohammad Babaeizadeh, Golnaz Ghiasi

2. 모델 구조

- α_c : content layer C에서 다른 layer와 대응하는 각 요소, parameter vector
- α_s : style layer S에서 다른 layer와 대응하는 각 요소, parameter vector



2020
Style Transfer

Adjustable real-time style transfer

– Mohammad Babaeizadeh, Golnaz Ghiasi

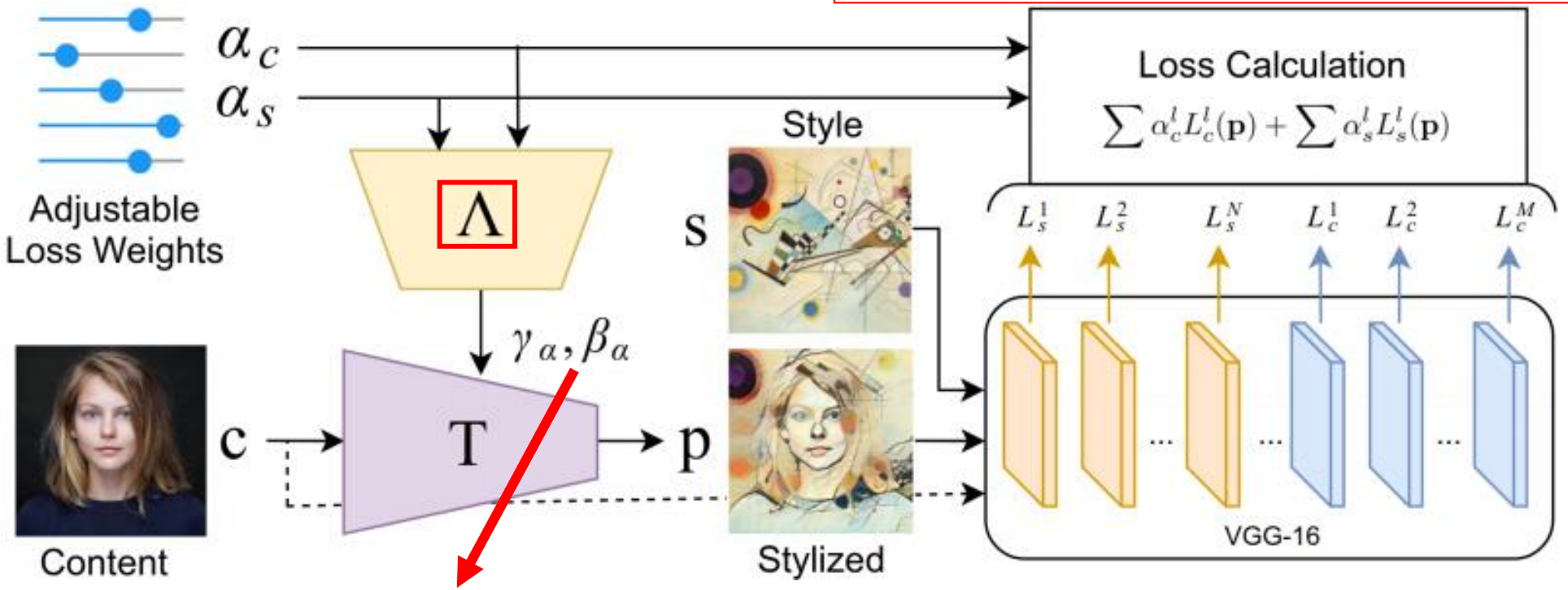
2. 모델 구조

Operation	input dimensions	output dimensions
input parameters $\boldsymbol{\alpha}$	3	1000
10×Dense	1000	1000
Dense	1000	$2(\gamma_{\boldsymbol{\alpha}}, \beta_{\boldsymbol{\alpha}})$
Optimizer	Adam ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$)	
Training iterations	200K	
Batch size	8	
Weight initialization	Isotropic gaussian ($\mu = 0, \sigma = 0.01$)	

Table 1. Network architecture and hyper-parameters of Λ.

$$\gamma_{\boldsymbol{\alpha}}, \beta_{\boldsymbol{\alpha}} = \Lambda(\boldsymbol{\alpha}_c, \boldsymbol{\alpha}_s)$$

- α_c : content layer C에서 다른 layer와 대응하는 각 요소, parameter vector
- α_s : style layer S에서 다른 layer와 대응하는 각 요소, parameter vector



$$z = \gamma_{\boldsymbol{\alpha}}\left(\frac{x - \mu}{\sigma}\right) + \beta_{\boldsymbol{\alpha}}$$

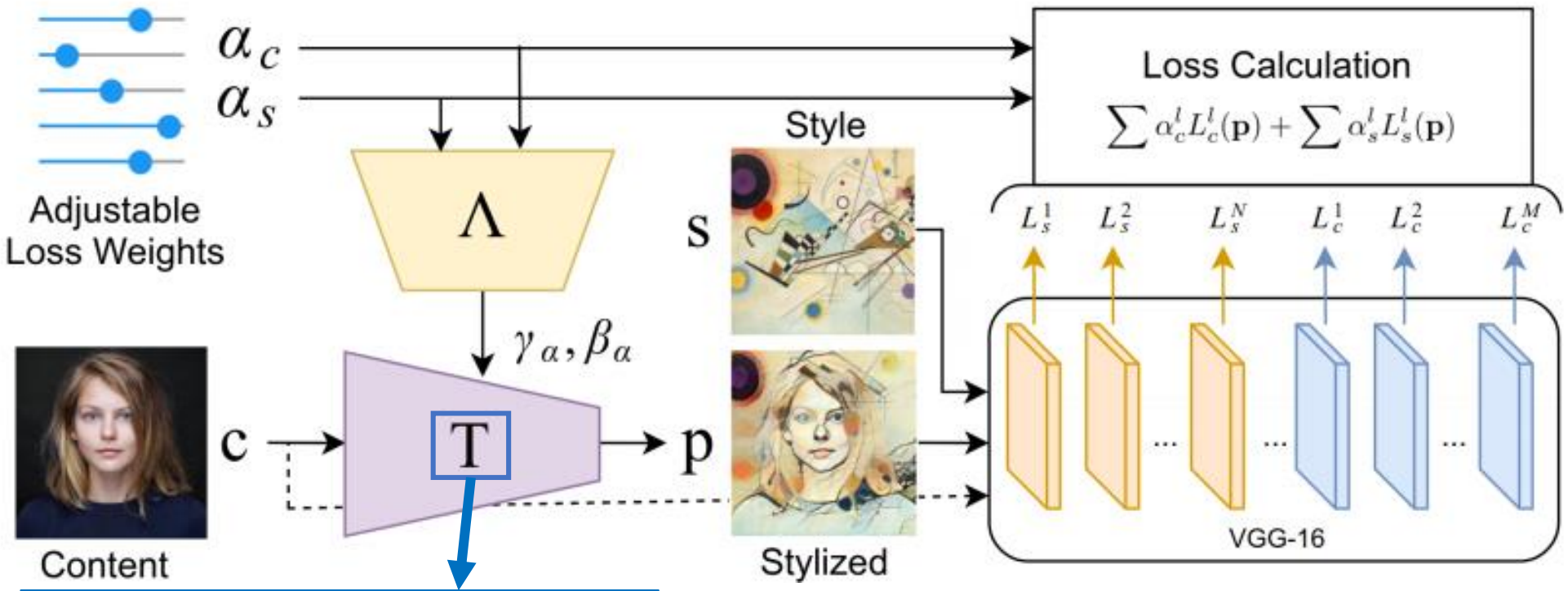
- conditional instance normalization
- Network T의 activation

2020
Style Transfer

Adjustable real-time style transfer

- Mohammad Babaeizadeh, Golnaz Ghiasi

2. 모델 구조



Network T
- feed-forward convolutional network
- Upsampling + Downsampling + residual block

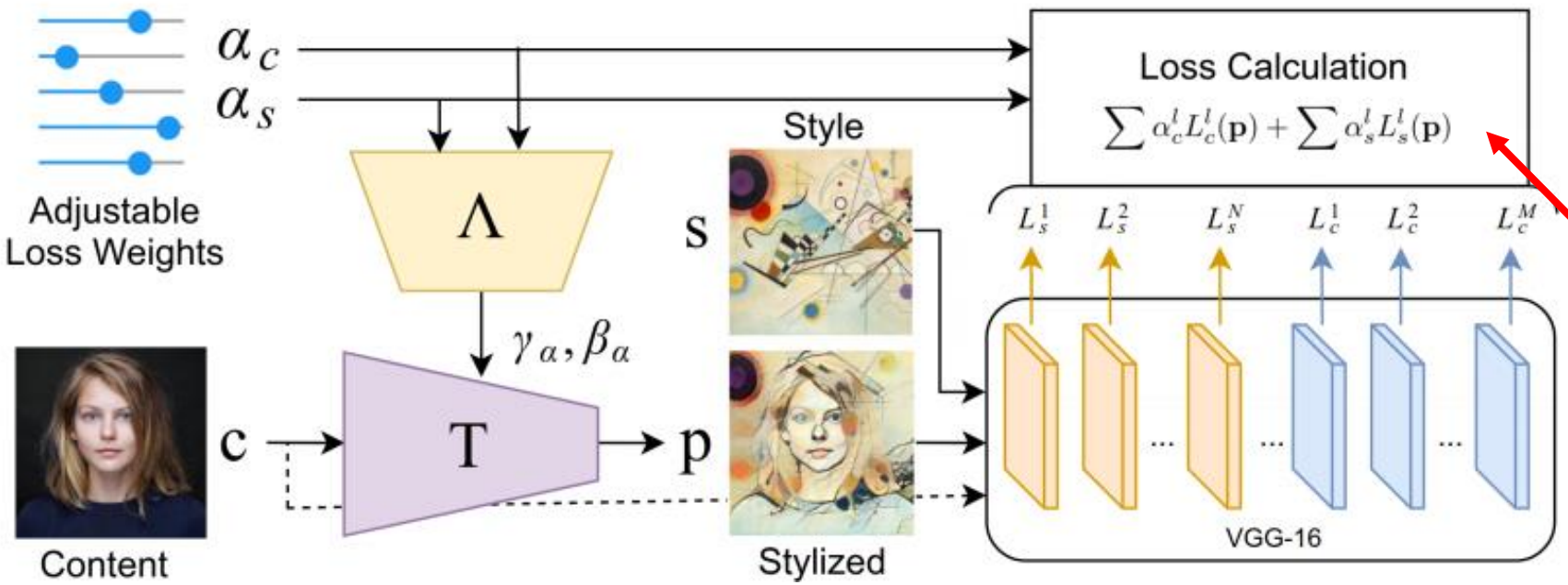
$$\mathbf{p} = \Psi(\mathbf{c}, \mathbf{s}, \boldsymbol{\alpha}_c, \boldsymbol{\alpha}_s)$$

2020
Style Transfer

Adjustable real-time style transfer

– Mohammad Babaeizadeh, Golnaz Ghiasi

3. Loss function



$$\mathcal{L}_c^l(\mathbf{p}) = \left|\left|\phi^l(\mathbf{p}) - \phi^l(\mathbf{s})\right|\right|_2^2$$
$$\mathcal{L}_s^l(\mathbf{p}) = \left|\left|G(\phi^l(\mathbf{p})) - G(\phi^l(\mathbf{s}))\right|\right|_F^2$$

$$\mathcal{L}_c(\mathbf{p}) = \sum_{l \in C} \alpha_c^l \mathcal{L}_c^l(\mathbf{p})$$
$$\mathcal{L}_s(\mathbf{p}) = \sum_{l \in S} \alpha_s^l \mathcal{L}_s^l(\mathbf{p})$$

2020
Style Transfer

Adjustable real-time style transfer

– Mohammad Babaeizadeh, Golnaz Ghiasi

4. Source code

2020
Style Transfer

Deformable Style Transfer

– Sunnie S. Y. Kim, Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

1. 해결하고자 하는 문제점 및 해결 방법

– 해결하고자 하는 문제점

1. Style transfer를 할 경우, texture에 초점을 맞추고 geometry를 무시하는 경향이 있음

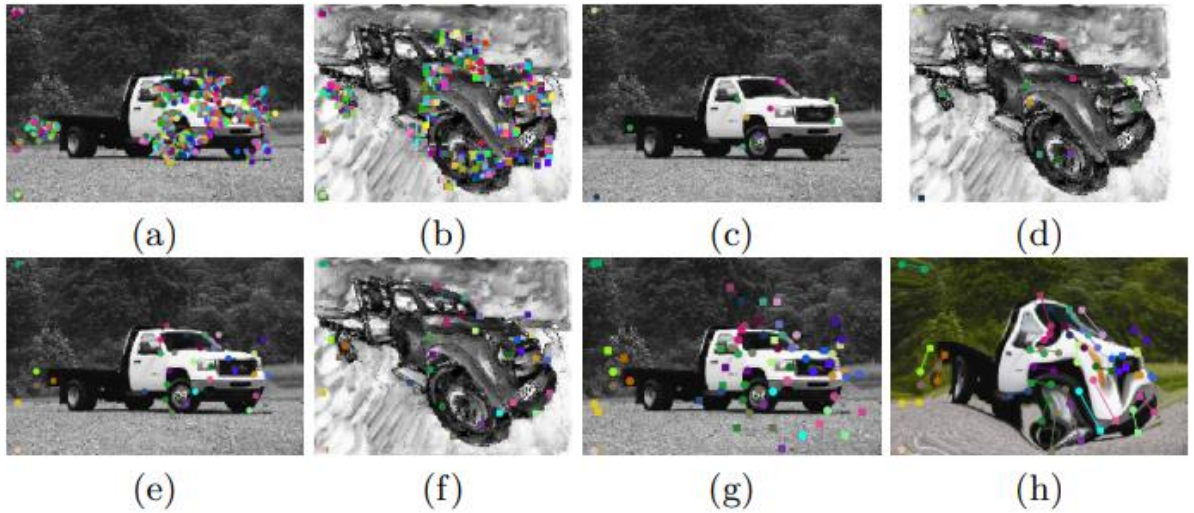
– 해결 방법
  – DST(Deformable Style Transfer)
    – NBB 사용하여 도메인간 겹쳐지는 keypoint detect

2020
Style Transfer

Deformable Style Transfer

– Sunnie S. Y. Kim, Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

2. Keypoint 찾기 – source point를 target point로 이동



(a) : 모든 correspondences를 담은 content image
(b) : 모든 correspondences를 담은 style image
(c) : original NBB keypoint가 있는 content image
(d) : original NBB keypoint가 있는 style image
(e) : 선택한 keypoint가 포함된 content image
(f) : 선택한 keypoint가 포함된 style image
(g) : 중앙에 맞춰 alignment한 keypoint가 포함된 content image
(h) : 유사성 변환에 의해 alignment keypoint를 통해 warp한 content image

2020
Style Transfer

Deformable Style Transfer

– Sunnie S. Y. Kim, Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

3. Loss function

1.1 Content and Style loss term – Gatys ver

- L_{style} : input style image's Gram matrix와 output's Gram matrix 사이의 squared error
- $L_{content}$: input content image's feature representation 과 output's feature representation 사이의 squared error

1.2 Content and Style loss term – STROTSS ver

- L_{style} : Relaxed EMD, moment matching term, color matching term
  - Relaxed EMD : style image의 구조적 형태를 output image로 transfer 하는데 도움
  - Moment matching term : feature vector의 평균과 공분산을 일치시키는 것을 목표로 함
  - Color matching term : output image와 style image가 유사한 팔레트를 갖도록 함

- $L_{content}$: input style image's feature vector와 output's feature vector사이의 normalized pairewise cosine distance

2020
Style Transfer

Deformable Style Transfer

– Sunnie S. Y. Kim, Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

$$L_{\text{style}}(I_s, X) + L_{\text{style}}(I_s, W(X, \theta)).$$

$L_{\text{style}}(I_s, X)$ – Style image I_S와 unwarped output image X간의 loss

$L_{\text{style}}(I_s, W(X, \theta))$. – Style image I_S와 wapred output image W(X, θ) 간의 loss

3. Loss function

2. Style loss of DST
- 공간적 특징 변형 여부와 관계없이 style에만 집중하는 loss function
- Stylization parameter X와 spatial deformation parameter θ 함께 작동
  → 이를 통해 공간적 특징까지 변형된 최종 출력 W(X, θ) 생성
- Image는 stylization을 거친 후에 warp

2020
Style Transfer

Deformable Style Transfer

– Sunnie S. Y. Kim, Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

$$L_{\text{warp}}(P, P', \theta) = \frac{1}{k} \sum_{i=1}^{k} \|p'_i - (p_i + \theta_i)\|_2,$$

$$R_{\text{TV}}(f) = \frac{1}{\text{W} \times \text{H}} \sum_{i=1}^{\text{W}} \sum_{j=1}^{\text{H}} \|f_{i+1,j} - f_{i,j}\|_1 + \|f_{i,j+1} - f_{i,j}\|_1.$$

3. Loss function

2. Deformation Loss Term
- P : source points
- P' : target points
- 스타일화 된 이미지의 geometry shape이 style의 geometry shape에 가까워지도록 loss를 줄이는 것이 목적
- Deformation loss를 최소화 시키면 contents와 style geometry의 비호환성으로 인해 artifact 발생 우려 → R_{TV} 정규화 term 사용

2020
Style Transfer

Deformable Style Transfer

– Sunnie S. Y. Kim, Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

3. Loss function

3. Objective function

$$L(X, \theta, I_c, I_s, P, P') = \alpha L_{\text{content}}(I_c, X)$$ – Content image I_C 와 stylized image(unwarped image) 간의 loss

$$+ L_{\text{style}}(I_s, X) + L_{\text{style}}(I_s, W(X, \theta))$$ – style loss

$$+ \beta L_{\text{warp}}(P, P', \theta)$$ – Deformation loss

$$+ \gamma R_{\text{TV}}(f_\theta),$$ – Normalized term

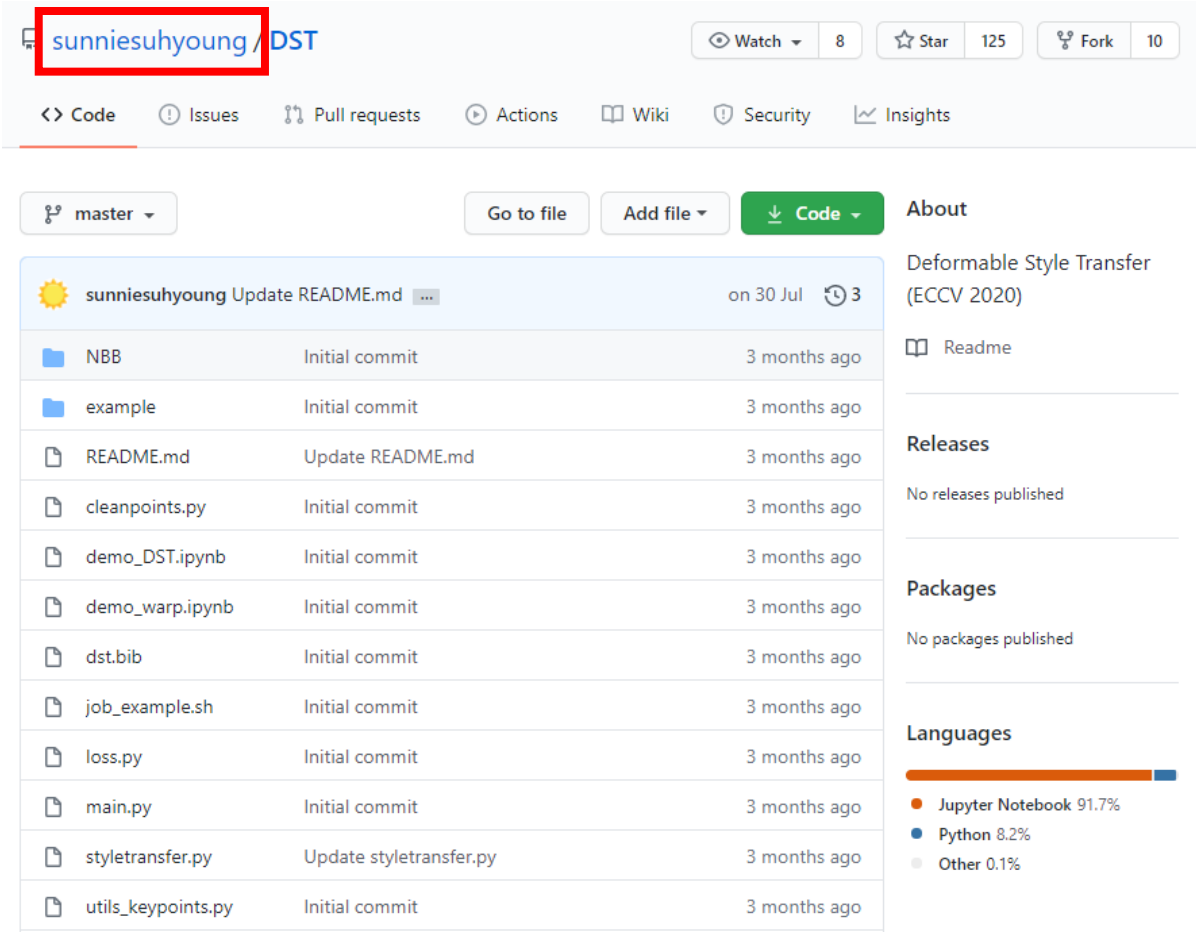
2020
Style Transfer

Deformable Style Transfer

– Sunnie S. Y. Kim, Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

3. Source code

URL : https://github.com/yiranran/Unpaired-Portrait-Drawing



2020
Style Transfer

Learning to Cartoonize Using White-box Cartoon Representations

– Xinrui Wang, Jinze Yu

(a) A frame in animation "Garden of words"

(b) A real photo processed by our method

1. 해결하고자 하는 문제점 및 해결 방법

– 해결하고자 하는 문제점

1. Image cartoonization using White-box

– 해결 방법

1. GAN 기반의 white-box controllable image cartoonization framework
    1. surface representation, structure representation, texture representation으로 분해하여 표현
    2. 각 representation loss function으로 output style 제어

2020
Style Transfer

Learning to Cartoonize Using White-box Cartoon Representations

– Xinrui Wang, Jinze Yu

2. 모델 구조

White box →

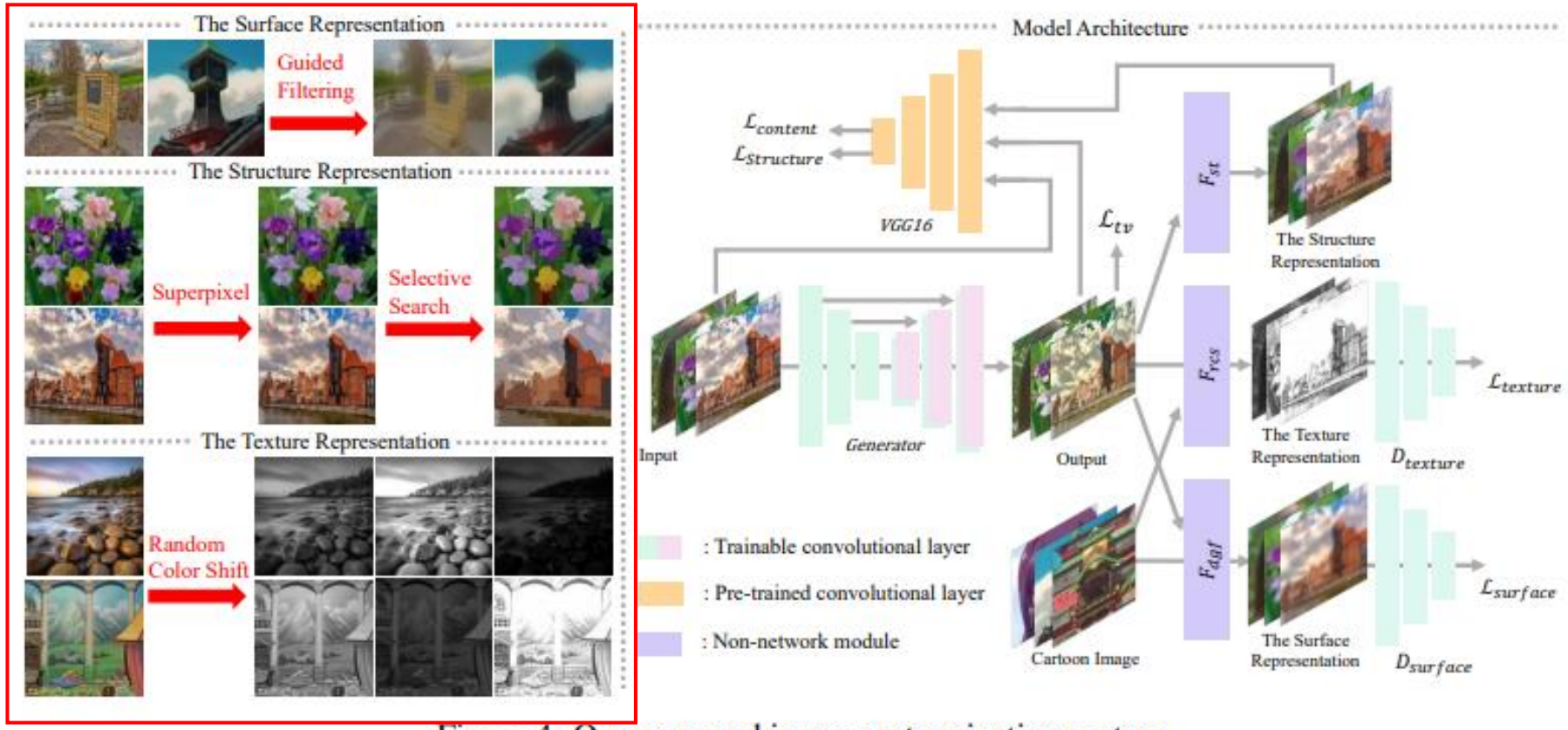


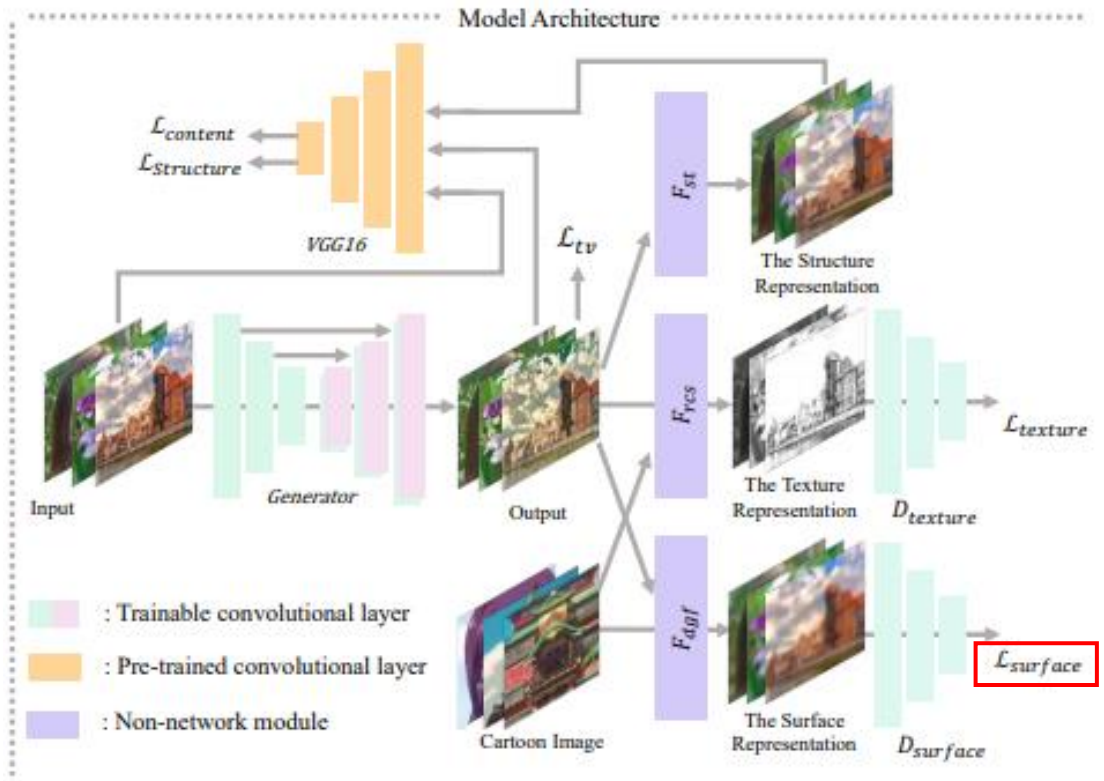
Figure 4: Our proposed image cartoonization system

2020
Style Transfer

Learning to Cartoonize Using White-box Cartoon Representations

– Xinrui Wang, Jinze Yu

3. Loss function



1. Surface Representation loss

– F_{dgf}() : Differentiable Guided filter

→ image smoothing & global semantic structure 유지

→ texture와 detail이 제거된 surface representation 반환

– D_s : output과 cartoon image의 surface의 유사한지 여부 판단

$$\mathcal{L}_{surface}(G,\ D_s) = logD_s(\mathcal{F}_{dgf}(\boldsymbol{I}_c, \boldsymbol{I}_c)) + log(1 - D_s(\mathcal{F}_{dgf}(G(\boldsymbol{I}_p), G(\boldsymbol{I}_p))))$$

2020
Style Transfer

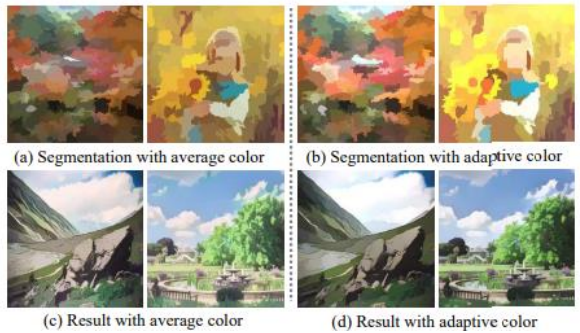
Learning to Cartoonize Using White-box Cartoon Representations

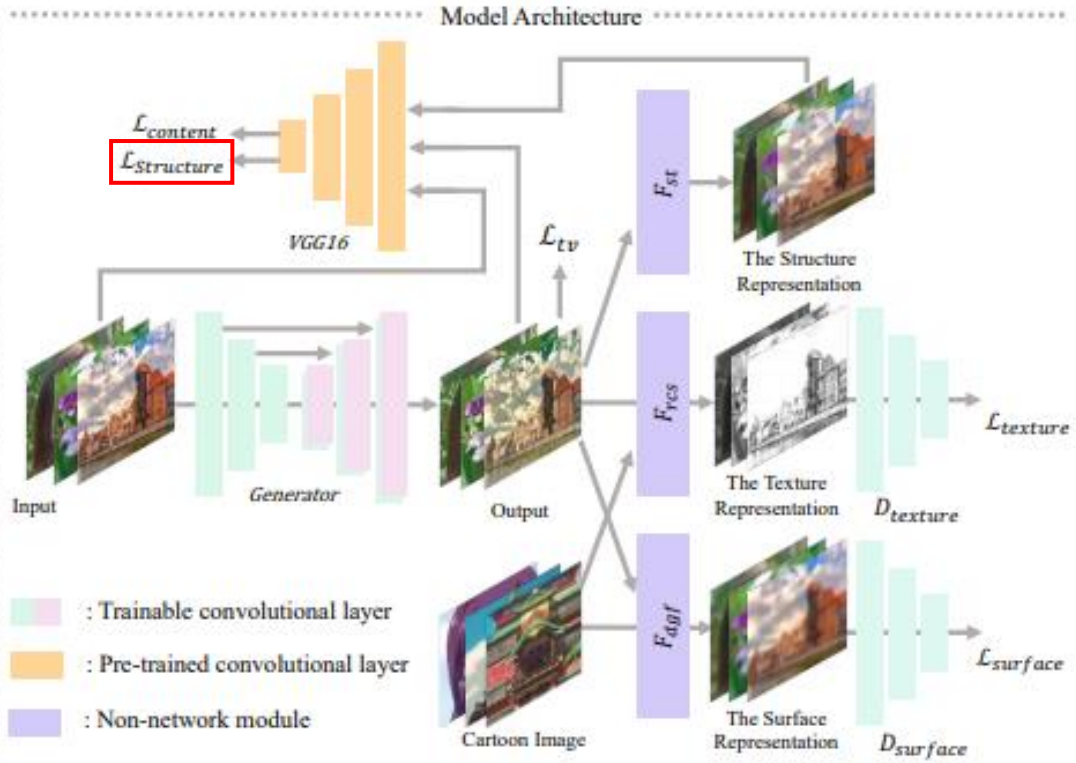
- Xinrui Wang, Jinze Yu

3. Loss function

$$\mathcal{L}_{structure} = \|VGG_n(G(\boldsymbol{I}_p)) - VGG_n(\mathcal{F}_{st}(G(\boldsymbol{I}_p)))\|$$

2. Structure Representation loss
- Structure representation
  - Felzenszwalb 알고리즘을 사용하여 이미지를 영역별로 분할
  - Super pixel algorithm : pixel의 유사성만 고려하고 semantic 정보 무시
    → 분할된 영역 병합 + sparse segmentation map을 추출하는 selective search 추가 도입
  - Adaptive coloring algorithm을 사용하여 기존의 최종 결과에 헤이징 효과를 유발하는 알고리즘을 보완
- F_{st} : structure representation 추출
- VGG16 network에서 추출한 high-level feature를 사용하여 output과 추출된 structure representation 사이에 공간적 공간적 제약 적용



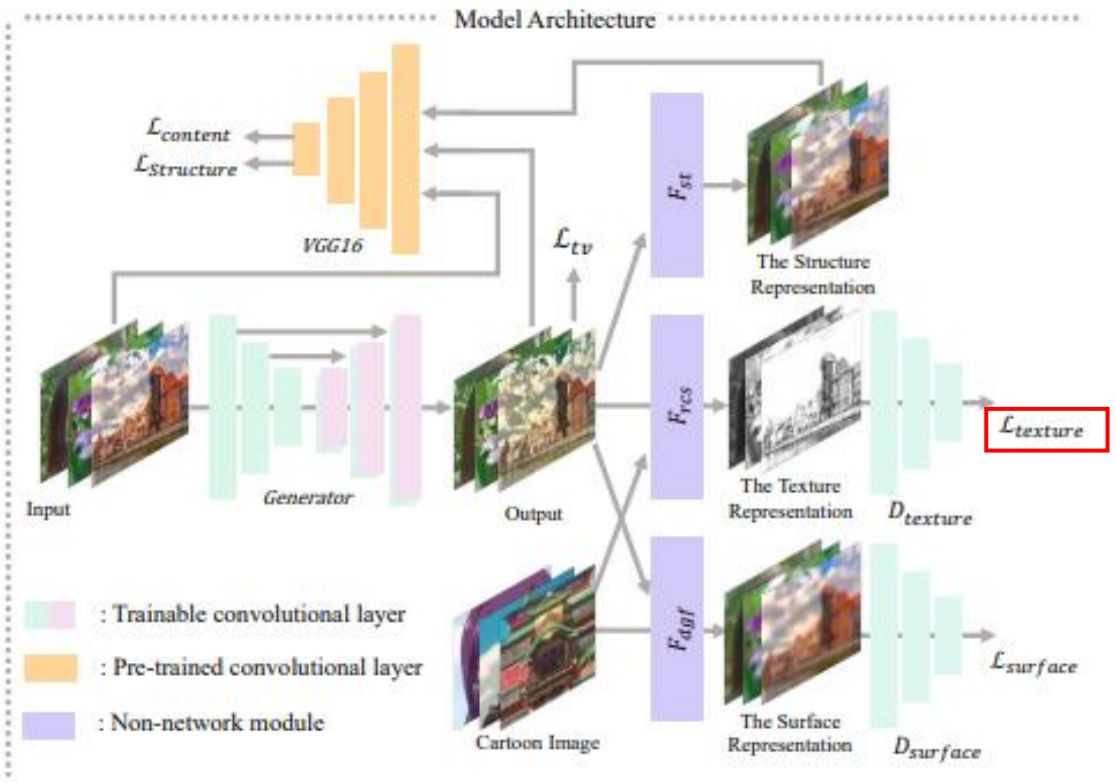


2020
Style Transfer

Learning to Cartoonize Using White-box Cartoon Representations

– Xinrui Wang, Jinze Yu

3. Loss function



$$\mathcal{F}_{rcs}(\boldsymbol{I}_{rgb}) = (1-\alpha)(\beta_1 * \boldsymbol{I}_r + \beta 2 * \boldsymbol{I}_g + \beta_3 * \boldsymbol{I}_b) + \alpha * \boldsymbol{Y}$$

$$\mathcal{L}_{texture}(G, D_t) = logD_t(\mathcal{F}_{rcs}(\boldsymbol{I}_c)) + log(1 - D_t(\mathcal{F}_{rcs}(G(\boldsymbol{I}_p))))$$

3. Texture Representation loss
- Texture representation
  - high-level frequency feature 유지 & 색상, 휘도 제거
  - F_{rcs} : color shift algorithm, 임의의 색상 이동은 휘도와 색상 정보가 제거된 random intensity maps을 생성할 수 있음

2020
Style Transfer

Learning to Cartoonize Using White-box Cartoon Representations

– Xinrui Wang, Jinze Yu

$$\mathcal{L}_{total} = \lambda_1 * \mathcal{L}_{surface} + \lambda_2 * \mathcal{L}_{texture} + \lambda_3 * \mathcal{L}_{structure} + \lambda_4 * \mathcal{L}_{content} + \lambda_5 * \mathcal{L}_{tv}$$

3. Loss function

4. Total loss
– 각 람다의 값을 조정하고 균형을 맞추면 다양한 예술적 스타일을 다양한 응용 프로그램에 적용 가능

2020
Style Transfer

Learning to Cartoonize Using White-box Cartoon Representations

– Xinrui Wang, Jinze Yu

$$\mathcal{L}_{tv} = \frac{1}{H * W * C} \| \nabla_x \left(G(\boldsymbol{I}_p) \right) + \nabla_y (G(\boldsymbol{I}_p)) \|$$

3. Loss function

5. Total variation loss
- 생성된 image에 spatial smoothness(공간적 평활도)를 부여하는데 사용
- H, W, C : 이미지의 공간적 차원(height, width, channel)

2020
Style Transfer

Learning to Cartoonize Using White-box Cartoon Representations

– Xinrui Wang, Jinze Yu

$$\mathcal{L}_{content} = \|VGG_n(G(\boldsymbol{I}_p)) - VGG_n(\boldsymbol{I}_p)\|$$

3. Loss function

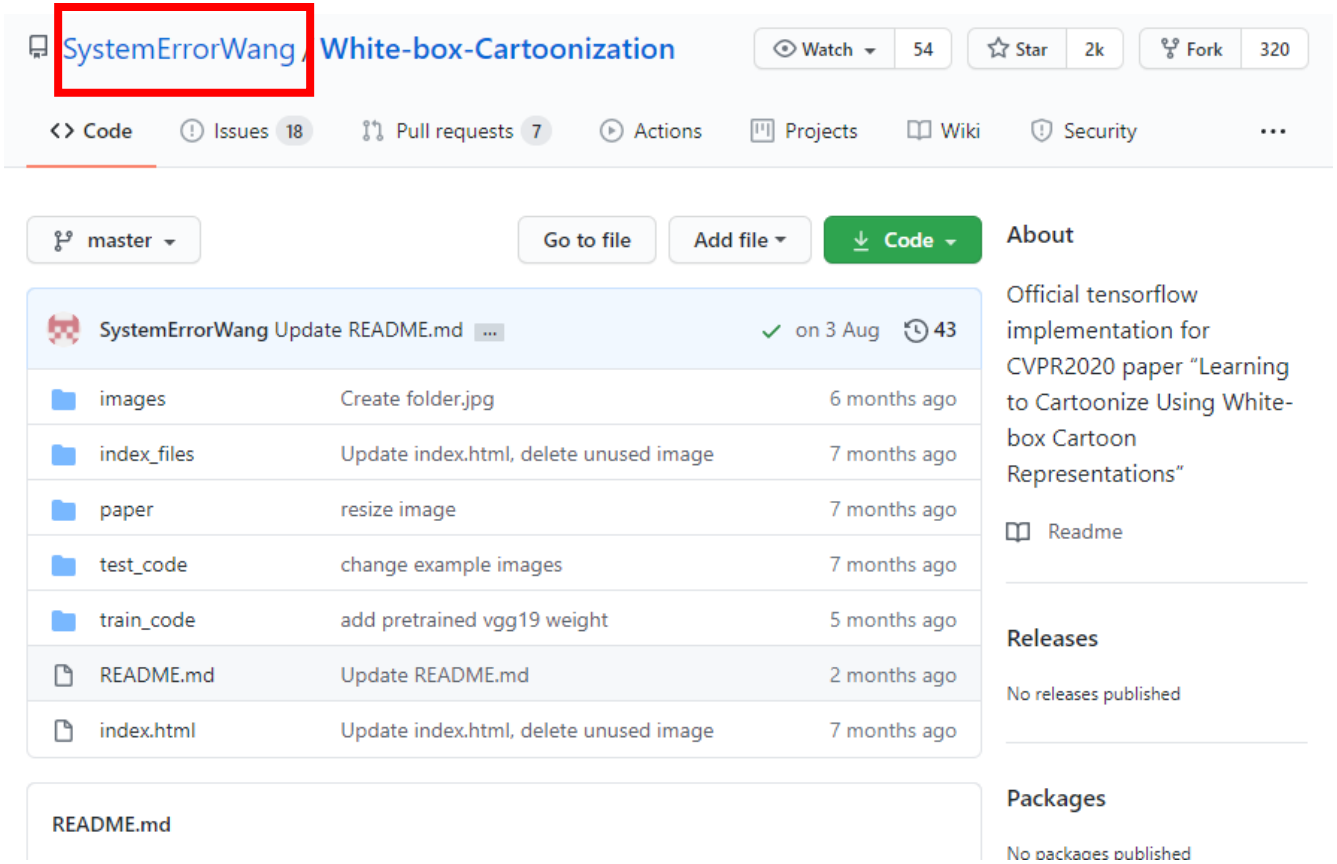
6. Content loss
- 생성된 cartoonized image와 input photo의 semantically invariant 보장
- Content 정보 비교

2020
Style Transfer

Learning to Cartoonize Using White-box Cartoon Representations

– Xinrui Wang, Jinze Yu

4. Source code

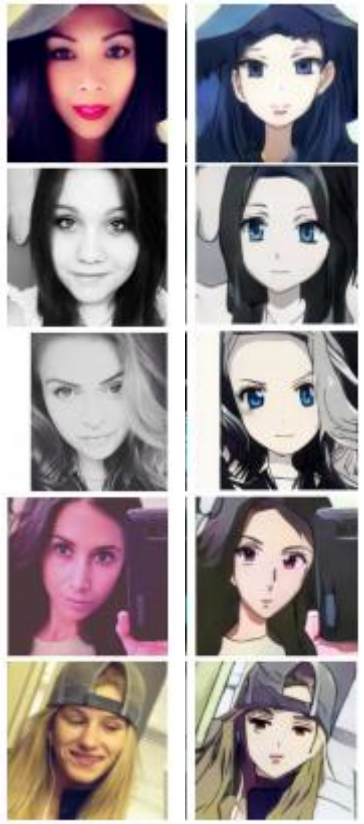


URL https://github.com/SystemErrorWang/White-box-Cartoonization

2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee



1. 해결하고자 하는 문제점 및 해결 방법

- 해결하고자 하는 문제점

1. 기존 모델은 도메인 간의 Geometry transfer 처리하지 못함

- 해결 방법

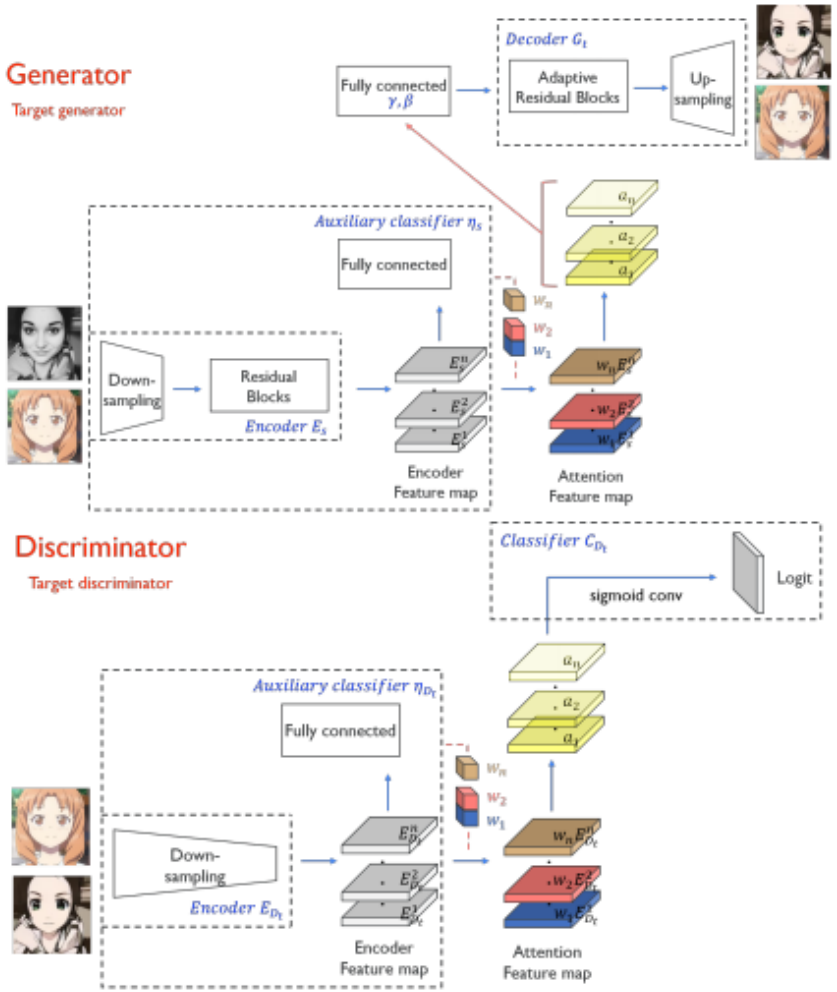
1. U-GAT-IT
    1. Attention module ; 보조 분류기에서 얻은 attention map을 기반으로 source와 target domain을 구별할 때 중요한 영역에 초점을 맞추도록 도와줌
    2. AdaLIN(Adaptive Layer-Instance Normalization) : 학습된 파라미터에 의해 shape과 texture의 변화량을 유연하게 제어할 수 있도록 도와줌

2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

2. 모델 구조




```python
def discriminate_real(self, x_A, x_B):
    real_A_logit, real_A_cam_logit, _, _ = self.discriminator(x_A, scope="discriminator_A")
    real_B_logit, real_B_cam_logit, _, _ = self.discriminator(x_B, scope="discriminator_B")

    return real_A_logit, real_A_cam_logit, real_B_logit, real_B_cam_logit

def discriminate_fake(self, x_ba, x_ab):
    fake_A_logit, fake_A_cam_logit, _, _ = self.discriminator(x_ba, reuse=True, scope="discriminator_A")
    fake_B_logit, fake_B_cam_logit, _, _ = self.discriminator(x_ab, reuse=True, scope="discriminator_B")

    return fake_A_logit, fake_A_cam_logit, fake_B_logit, fake_B_cam_logit
```

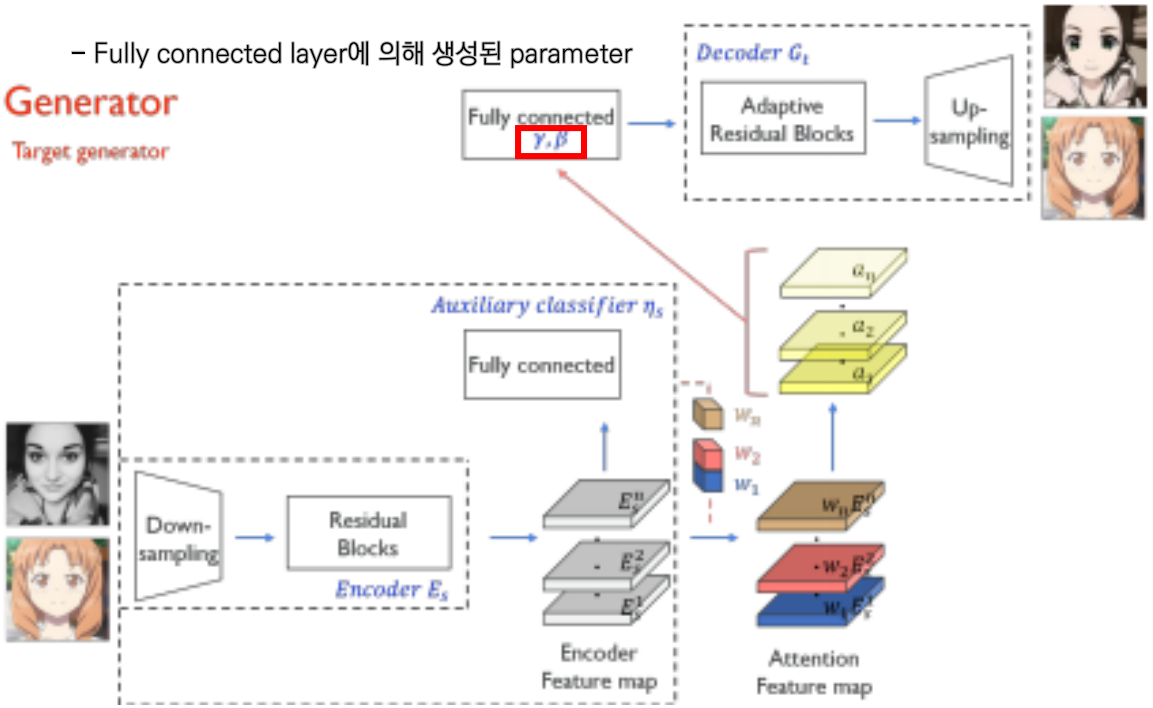
2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

2. 모델 구조

$$AdaLIN(a,\gamma,\beta) = \gamma \cdot (\rho \cdot \hat{a_I} + (1-\rho) \cdot \hat{a_L}) + \beta,$$
$$\hat{a_I} = \frac{a-\mu_I}{\sqrt{\sigma_I^2+\epsilon}}, \hat{a_L} = \frac{a-\mu_L}{\sqrt{\sigma_L^2+\epsilon}},$$
$$\rho \leftarrow clip_{[0,1]}(\rho - \tau\Delta\rho)$$

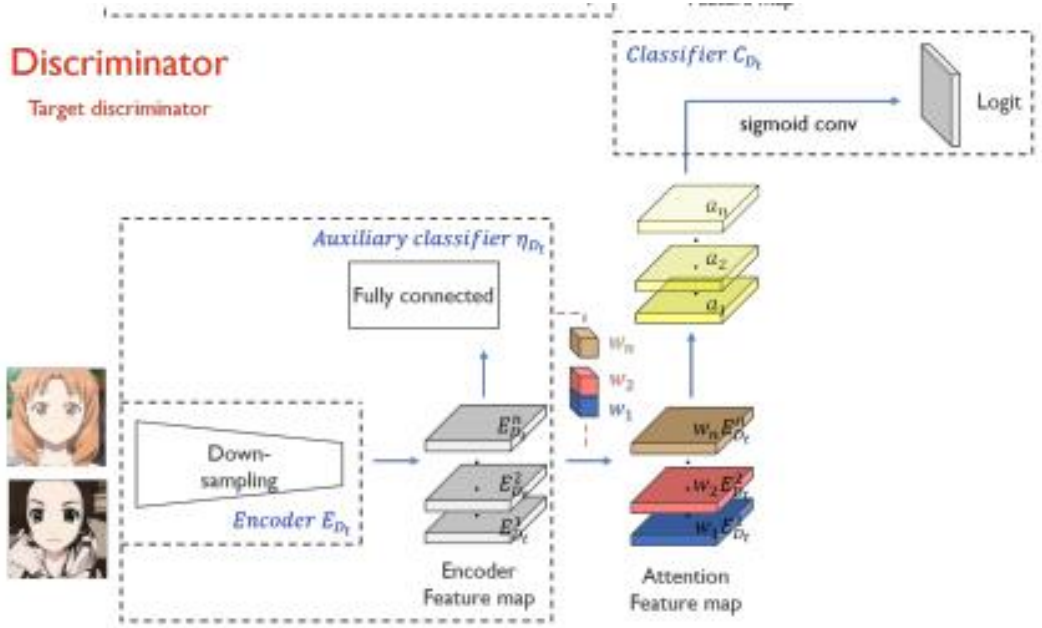


2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

2. 모델 구조



2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

$$L_{lsgan}^{s \to t} = (\mathbb{E}_{x \sim X_t}[(D_t(x))^2] + \mathbb{E}_{x \sim X_s}[(1 - D_t(G_{s \to t}(x)))^2]).$$

3. Loss function

1. Adversarial loss
- Real target image와 Fake targe image를 구별
- D 입장은 값이 클수록, G 입장은 값이 작을수록 좋음

2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

$$L_{cycle}^{s \to t} = \mathbb{E}_{x \sim X_s}[|x - G_{t \to s}(G_{s \to t}(x))|_1].$$

3. Loss function

2. Cycle loss
- Real source image와 reconstruction source image 사이의 loss

2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

$$L_{identity}^{s \to t} = \mathbb{E}_{x \sim X_t}[|x - G_{s \to t}(x)|_1].$$

3. Loss function

3. Identity loss
- Real source image와 reconstruction source image 사이의 loss

2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

$$L_{cam}^{s \to t} = -(\mathbb{E}_{x \sim X_s}[log(\eta_s(x))] + \mathbb{E}_{x \sim X_t}[log(1 - \eta_s(x))]),$$

$$L_{cam}^{D_t} = \mathbb{E}_{x \sim X_t}[(\eta_{D_t}(x))^2] + \mathbb{E}_{x \sim X_s}[(1 - \eta_{D_t}(G_{s \to t}(x))^2].$$

3. Loss function

4. CAM loss
– 보조 분류기 η_s와 η_{Dt}의 정보를 이용
– 개선이 필요한 부분이나 현재 상태에서 두 도메인 간의 가장 큰 차이점을 알게 됨

2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation

– Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

$$\min_{G_{s\to t}, G_{t\to s}, \eta_s, \eta_t} \max_{D_s, D_t, \eta_{D_s}, \eta_{D_t}} \lambda_1 L_{lsgan} + \lambda_2 L_{cycle} + \lambda_3 L_{identity} + \lambda_4 L_{cam},$$

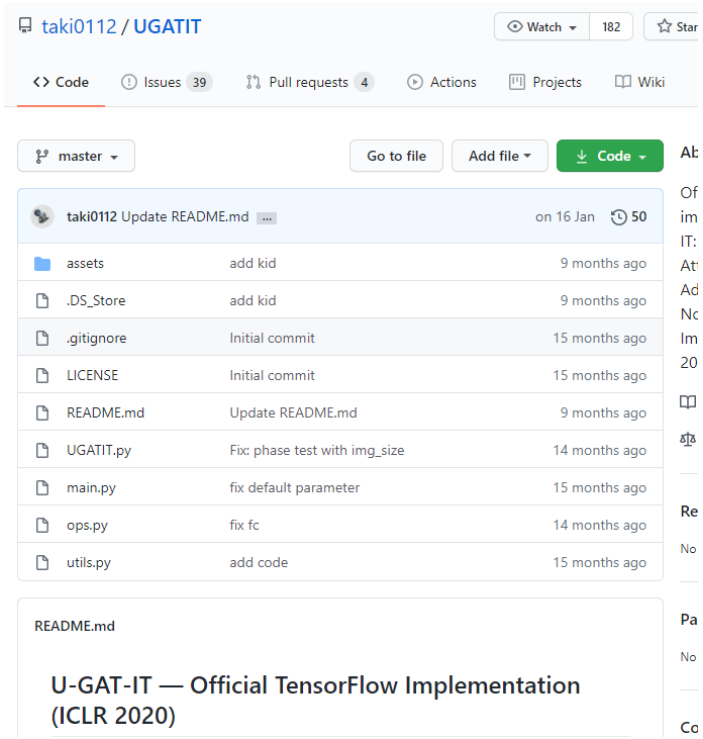
3. Loss function

5. Full objective

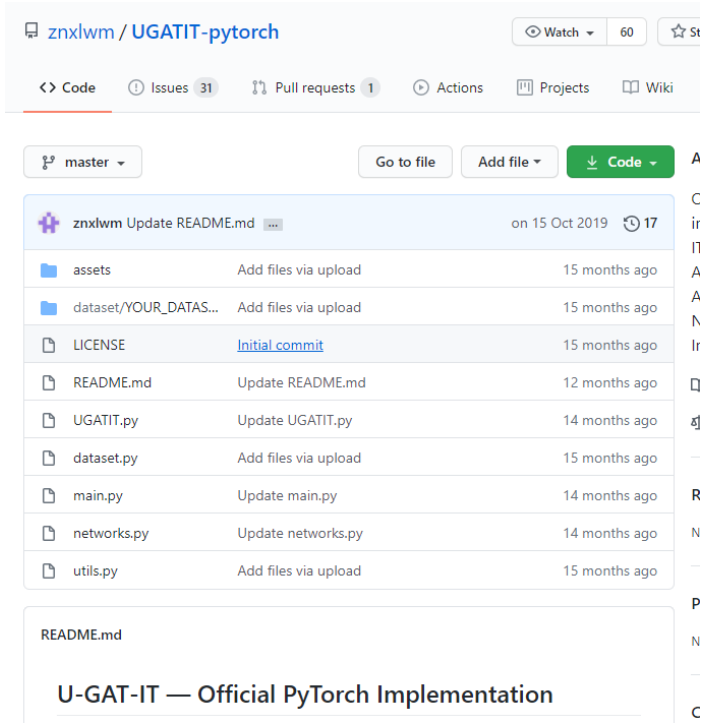
2020
Style Transfer

U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation – Junho Kim, Minjae Kim, Hyeonwoo Kang, Kwanghee Lee

4. Source code



URL https://github.com/taki0112/UGATIT



URL https://github.com/znxlwm/UGATIT-pytorch

2018
Style Transfer

Neural Style Transfer via Meta Networks

– Falong Shen, Shuicheng Yan, Gang Zeng

(a) Input (b) Style (e) Ours

1. 해결하고자 하는 문제점 및 해결 방법

– 해결하고자 하는 문제점

1. 기존 style transfer는 새로운 스타일에 대한 일반화 기능 부족

– 해결 방법

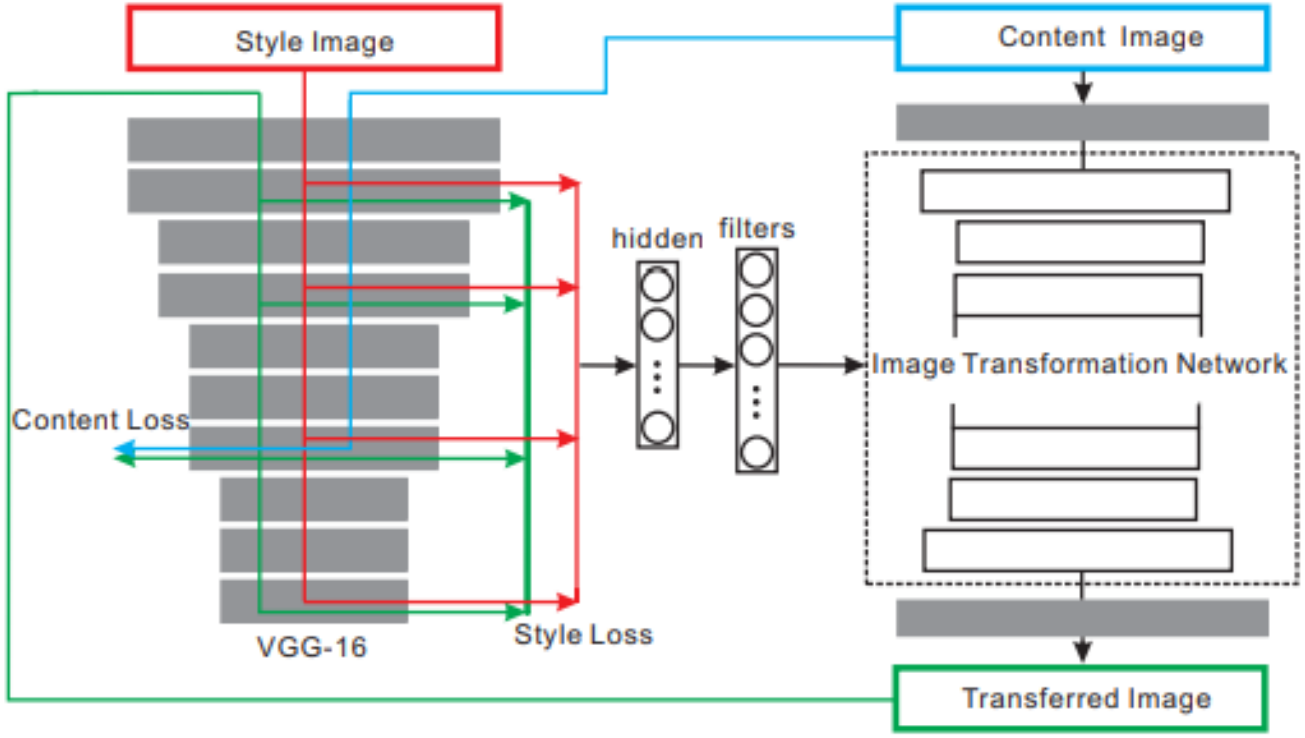
1. Meta network 구축

2018
Style Transfer

Neural Style Transfer via Meta Networks

– Falong Shen, Shuicheng Yan, Gang Zeng

2. 모델 구조



2018
Style Transfer

Neural Style Transfer via Meta Networks

– Falong Shen, Shuicheng Yan, Gang Zeng

$$\min_{I_x} \mathbf{PLoss}(I_x | I_c, I_s),$$

$$\lambda_c ||\mathbf{CP}(I) - \mathbf{CP}(I_c)||_2^2 + \lambda_s ||\mathbf{SP}(I) - \mathbf{SP}(I_s)||_2^2.$$

3. Loss function

1. Neural Style Transfer
    1. CP: Content perceptron & VGG-16 relu3_3에서 뽑은 features
    2. SP: Style perceptron & VGG-16 relu2, relu2_2, relu3_3, relu4_3에서 뽑은 features & Gram matrix
    3. 많은 반복 필요 → 계산량 증가

2018
Style Transfer

Neural Style Transfer via Meta Networks

– Falong Shen, Shuicheng Yan, Gang Zeng

$$\min_{I_x} \mathbf{PLoss}(I_x|I_c, I_s),$$

$$\mathcal{N} : I_c| \rightarrow I_x^*.$$

3. Loss function

1. Fast Neural Style Transfer via Image Transformation Networks
    1. 스타일 이미지 고정되면 주어진 콘텐츠 이미지에 대해 항상 방정식을 만족시키는 I_X*이 존재
    2. 이 문제를 해결하기 위해 매핑함수 N이 필요
    3. ERM(empirical risk minimization) 사용

2018
Style Transfer

Neural Style Transfer via Meta Networks

– Falong Shen, Shuicheng Yan, Gang Zeng

$$\min_{w} \sum_{I_c \in \mathcal{D}_c} \mathbf{PLoss}(I_x | I_c, I_s),$$

3. Loss function

1. Fast Neural Style Transfer via Image Transformation Networks
    1. I_x = N(Ic ; w)
    2. I_C : 콘텐츠 이미지, N의 입력이자 Ploss 안에서 supervised target으로 사용
    3. I_s는 고정된 스타일 이미지
    4. 훈련이 완료되면 매개변수 w에서 I_S 스타일을 인코딩하는 이미지 translation network N을 얻게 됨

2018
Style Transfer

Neural Style Transfer via Meta Networks

– Falong Shen, Shuicheng Yan, Gang Zeng

$$I_s \xrightarrow{SGD} \mathcal{N}(\cdot; w_*).$$

3. Loss function

1. Meta Networks for Neural Style Transfer
    1. 주어진 스타일 이미지에 대해 w로 매개변수화된 이미지 translation network를 찾으려면 수많은 SGD를 반복해야 한다.
    2. 즉, 모든 스타일 이미지에 대해 항상 이 네트워크가 존재
    3. 그러나 앞서 말한 것처럼 수많은 SGD를 진행하면 시간이 너무 많이 걸림
    4. 따라서 메타네트워크에 의해 변환 네트워크 얻는 법을 제안

2018
Style Transfer

Neural Style Transfer via Meta Networks

– Falong Shen, Shuicheng Yan, Gang Zeng

$$\min_{\theta} \sum_{I_c \in \mathcal{D}_c} \sum_{I_s \in \mathcal{D}_s} \mathbf{PLoss}(I_x | I_c, I_s), \quad \begin{aligned} I_x &= \mathcal{N}(I_c; w_\theta) \\ w_\theta &= Meta\mathcal{N}(I_s; \theta). \end{aligned}$$

3. Loss function

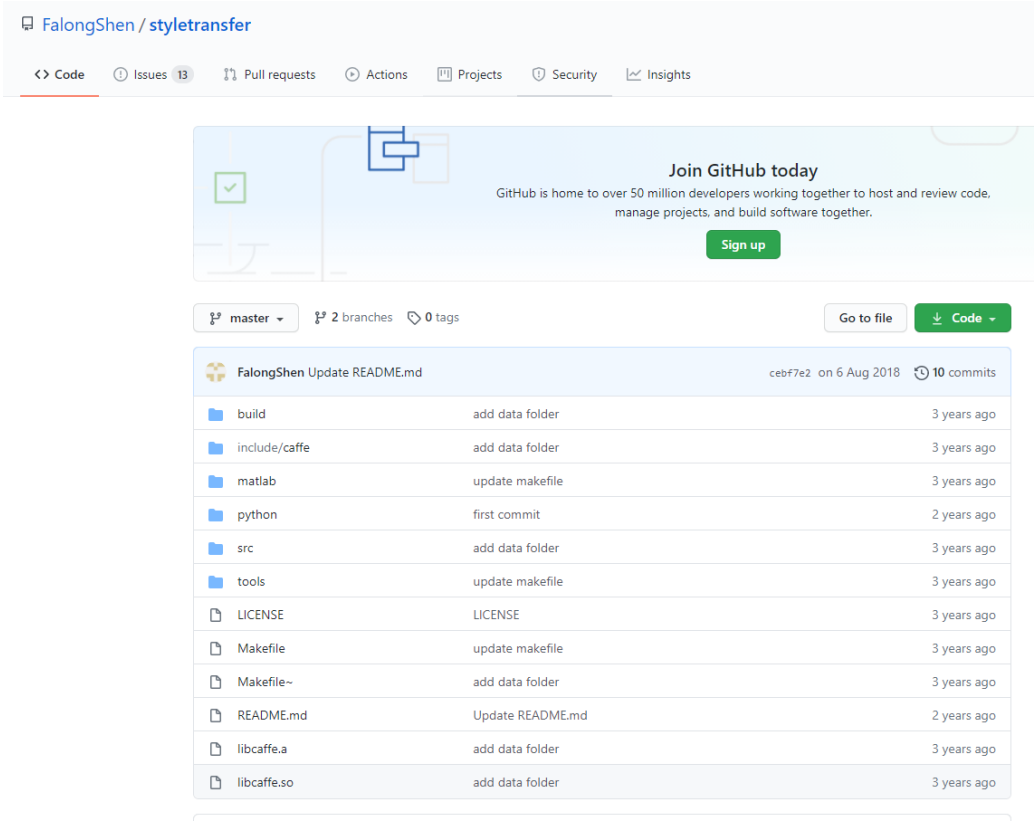
1. Meta Networks for Neural Style Transfer
    1. ERM으로 최종 출력 이미지 I_X 와 비교
    2. N : 실제 이미지 변환 네트워크
    3. MetaN : 스타일을 뽑아내는 네트워크

2018
Style Transfer

Neural Style Transfer via Meta Networks

– Falong Shen, Shuicheng Yan, Gang Zeng

4. Source code

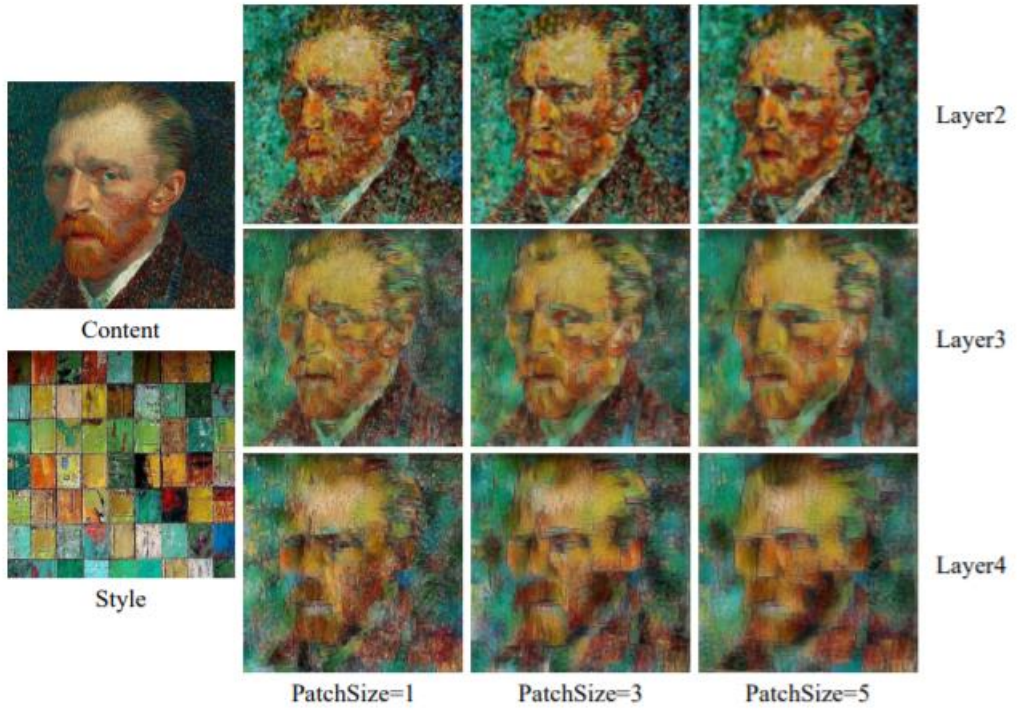


URL : https://github.com/FalongShen/styletransfer

2018
Style Transfer

Arbitrary Style Transfer With Deep Feature Reshuffle

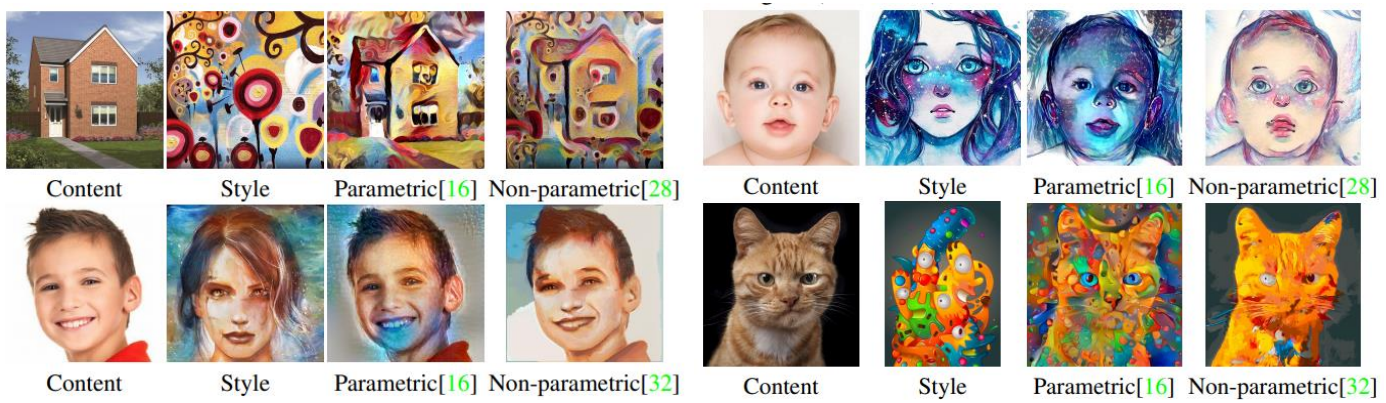
– Shuyang Gu, Congliang Chen, Jing Liao, Lu Yuan



1. 해결하고자 하는 문제점 및 해결 방법

– 해결하고자 하는 문제점
1. 비사실적인 렌더링(texture 합성 & 전송)
2. Semantic structure 캡처 못함

– 해결 방법
1. Style image의 deep feature를 재편성



2018
Style Transfer

Arbitrary Style Transfer With Deep Feature Reshuffle

– Shuyang Gu, Congliang Chen, Jing Liao, Lu Yuan

2. 모델 구조

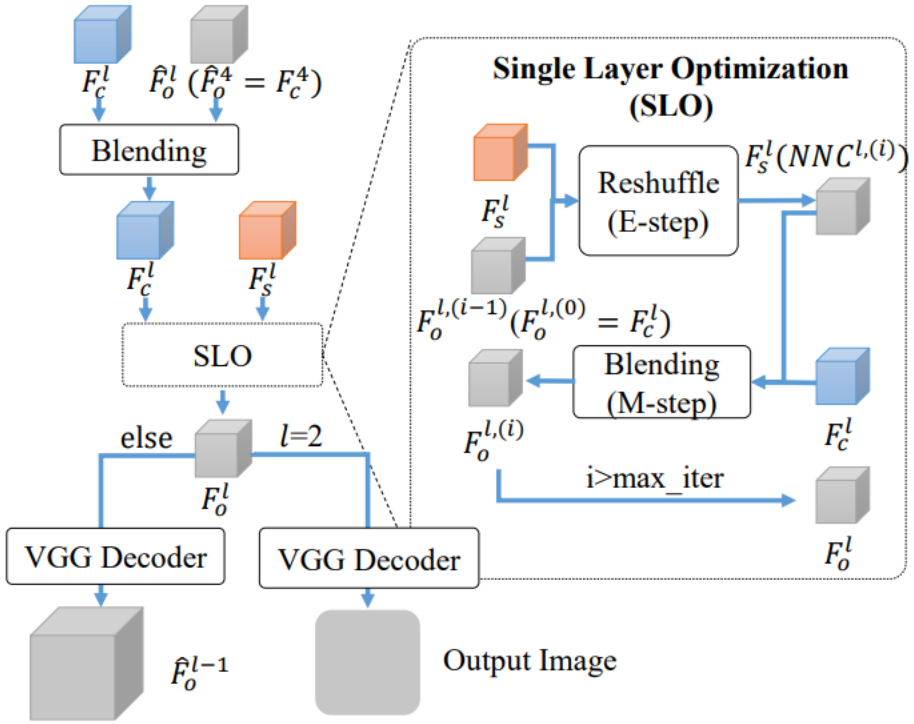


Figure 4. System Pipeline

Algorithm 1: The deep feature reshuffle algorithm.

Input : One content image I_c and one style image I_s.

Initialization:

$\{F_c^l\}_{l=2}^4, \{F_s^l\}_{l=2}^4 \leftarrow$ feed I_c, I_s to VGG-19.

$\hat{F}_o^4 = F_c^4$.

for $l = 4$ *to* 2 **do**

  $F_c^l \leftarrow \beta \hat{F}_o^l + (1-\beta) F_c^l$.

  $F_o^{l,(0)} \leftarrow F_c^l$.

  for $i = 1$ *to* max_iter **do**

    $\text{NNC}^{l,(i)} \leftarrow$ match $F_o^{l,(i-1)}$ and F_s^l (Eq. (10)).

    $F_s^l(\text{NNC}^{l,(i)}) \leftarrow$ warp F_s^l with $\text{NNC}^{l,(i)}$.

    $F_o^{l,(i)} \leftarrow \alpha F_c^l + (1-\alpha) F_s^l(NNC^{l,(i)})$.

  end

  if $l > 2$ **then**

    $\hat{F}_o^{l-1} \leftarrow$ decode $F_o^{l,(max\_iter)}$ from l to $l-1$.

  end

end

$I_o \leftarrow$ decode $F_o^{2,(max\_iter)}$ to image.

Output: Style transfer result image I_o.

$$\text{NNC}(p) = \underset{p'=1,2,\ldots,\Theta}{\arg\max} \left(\frac{\Psi_p(F_o) \cdot \Psi_{p'}(F_s)}{||\Psi_p(F_o)||_{\text{F}}^2 \cdot ||\Psi_{p'}(F_s)||_{\text{F}}^2} - \lambda \frac{\Gamma(\Psi_{p'}(F_s))}{R \times R} \right), \quad (10)$$

2018
Style Transfer

Arbitrary Style Transfer With Deep Feature Reshuffle

– Shuyang Gu, Congliang Chen, Jing Liao, Lu Yuan

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{cont} + (1-\alpha)\mathcal{L}_{shuf}.$$

$$\mathcal{L}_{cont} = ||F_o - F_c||_{\mathrm{F}}^2,$$

중심위치가 p인 Feature map F로부터 추출한 모든 local patches의 list

$$\mathcal{L}_{shuf} = \sum_p ||\Psi_p(F_o) - \Psi_{\mathrm{NNC}(p)}(F_s)||_{\mathrm{F}}^2,$$

F_o에서 추출한 local patch list와 가장 유사한 F_s patch list의 index

$$\mathrm{NNC}(p) = \underset{p'=1,2,\ldots,\Theta}{\arg\max} \left(\frac{\Psi_p(F_o) \cdot \Psi_{p'}(F_s)}{||\Psi_p(F_o)||_{\mathrm{F}}^2 \cdot ||\Psi_{p'}(F_s)||_{\mathrm{F}}^2} - \lambda \frac{\Gamma(\Psi_{p'}(F_s))}{R \times R} \right), \quad (10)$$

- 분자 : patch 총 사용량
- R : patch size
- Patch 사용 횟수 제한하는 term

$$\Gamma(\Psi_p) = \sum_{p \in \Psi_p} \frac{\Gamma(p)}{R \times R}.$$

3. Loss function

1. Total loss : content loss와 reshuffle loss(style loss)
2. Content loss : feature간의 유사도 측정(Frobenius norm)
3. Reshuffle loss : F_o와 가장 유사한 F_s 패치 찾기

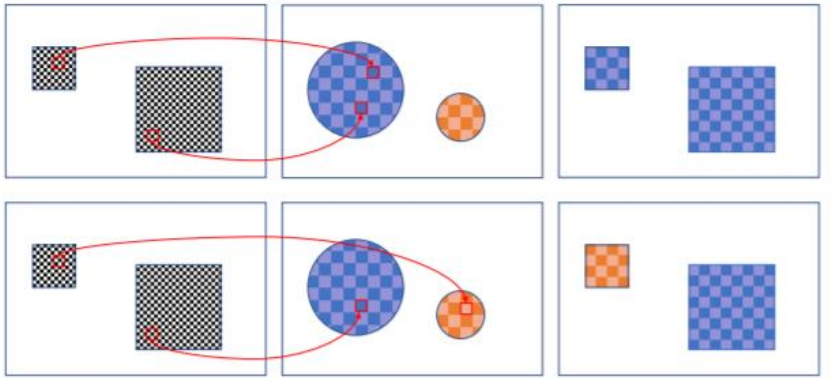


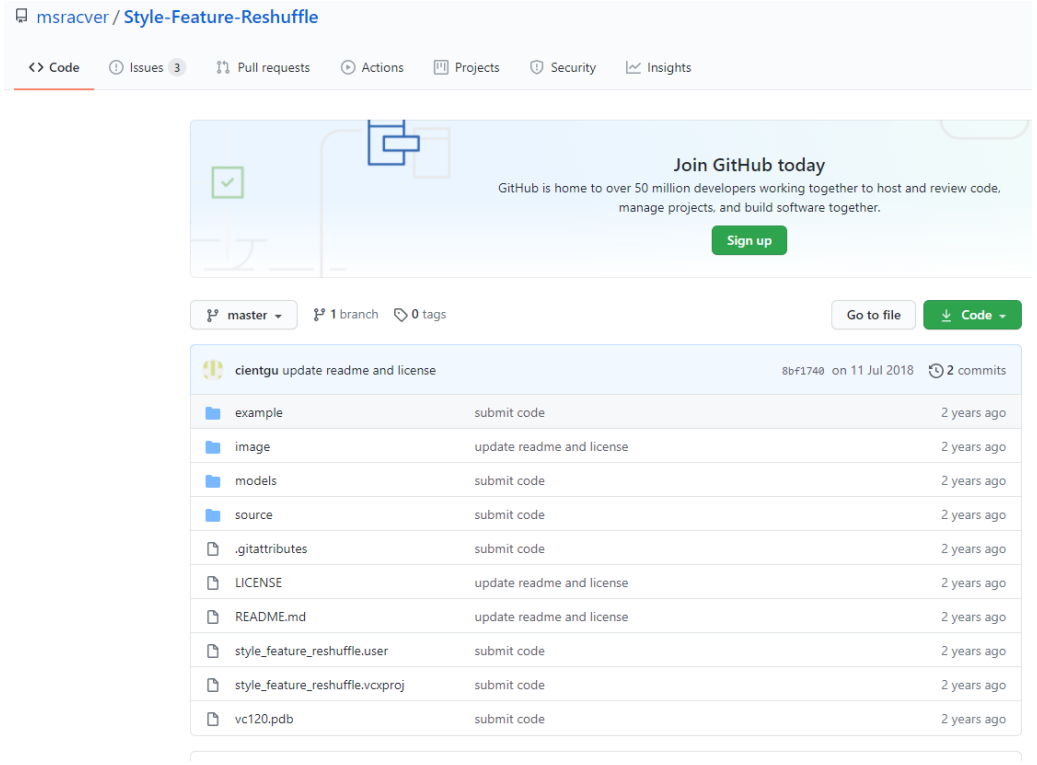
Figure 3. The comparison between normal NN search (upper row) and our NN search constrained by reshuffle (lower row). Ours strictly requires each patch in the source to be only mapped once. Notice that the bottom only shows a possible solution of reshuffle.

2018
Style Transfer

Arbitrary Style Transfer With Deep Feature Reshuffle

– Shuyang Gu, Congliang Chen, Jing Liao, Lu Yuan

4. Source code

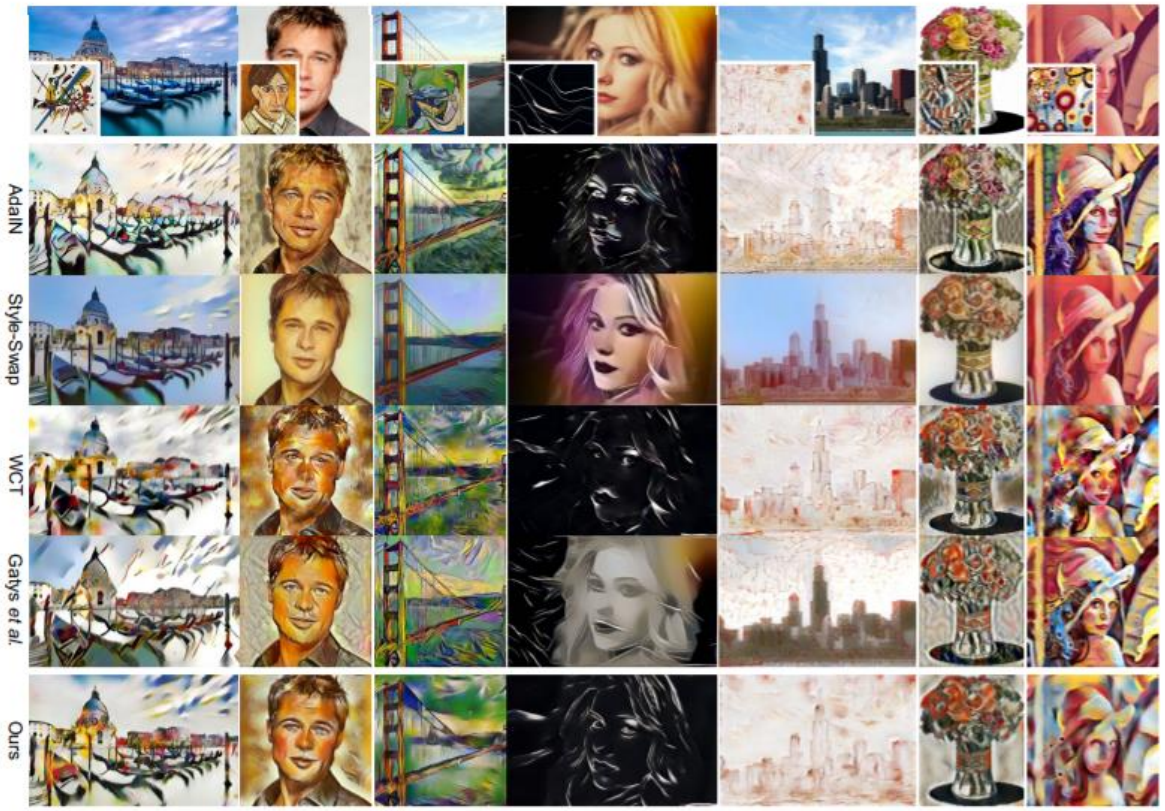


URL : https://github.com/msracver/Style-Feature-Reshuffle

2018
Style Transfer

Avatar-Net: Multi-Scale Zero-Shot Style Transfer by Feature Decoration

- Lu Sheng, Ziyi Lin, Jing Shao, Xiaogang Wang



1. 해결하고자 하는 문제점 및 해결 방법

- 해결하고자 하는 문제점

1. 기존의 style transfer는 일반화와 효율성 사이의 균형으로 인해 고품질 zero-shot style transfer가 방해됨

- 해결 방법

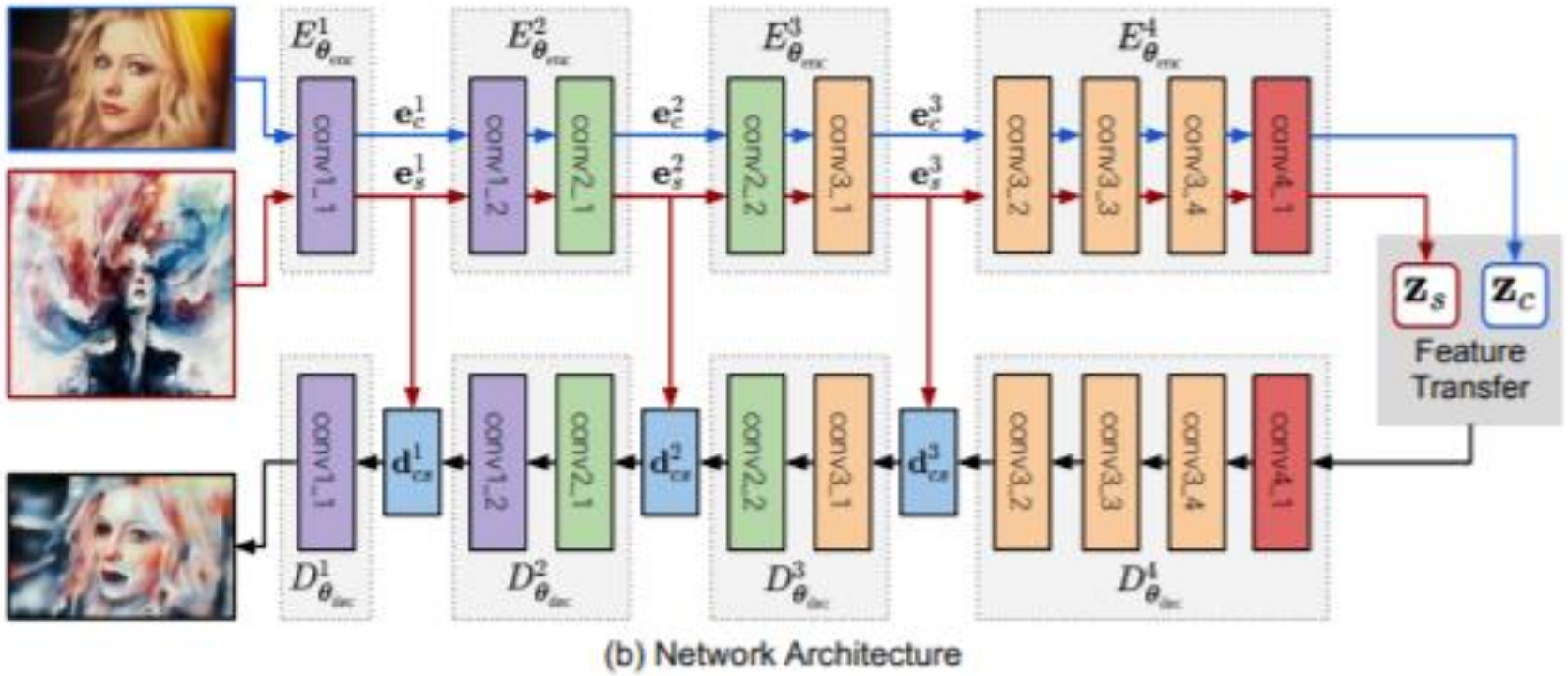
1. Avatar-Net
    1. 핵심 요소 : style decorator(임의의 스타일 이미지에서 semantically aligned style features에 의해 content features를 구성)
        1. 패치 기반
        2. Contents의 semantic을 인식할 수 있도록 유지하면서 style patter의 features로 contents feature를 decorate하는

2018
Style Transfer

Avatar-Net: Multi-Scale Zero-Shot Style Transfer by Feature Decoration

- Lu Sheng, Ziyi Lin, Jing Shao, Xiaogang Wang

2. 모델 구조



(b) Network Architecture

2018
Style Transfer

Avatar-Net: Multi-Scale Zero-Shot Style Transfer by Feature Decoration

– Lu Sheng, Ziyi Lin, Jing Shao, Xiaogang Wang

$$\ell_{\text{total}} = \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 + \lambda_1 \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \|\mathbf{\Psi}^i_{\text{VGG}}(\mathbf{x}) - \mathbf{\Psi}^i_{\text{VGG}}(\tilde{\mathbf{x}})\|_2^2 + \lambda_2 \ell_{\text{TV}}(\mathbf{x}),$$

3. Loss function

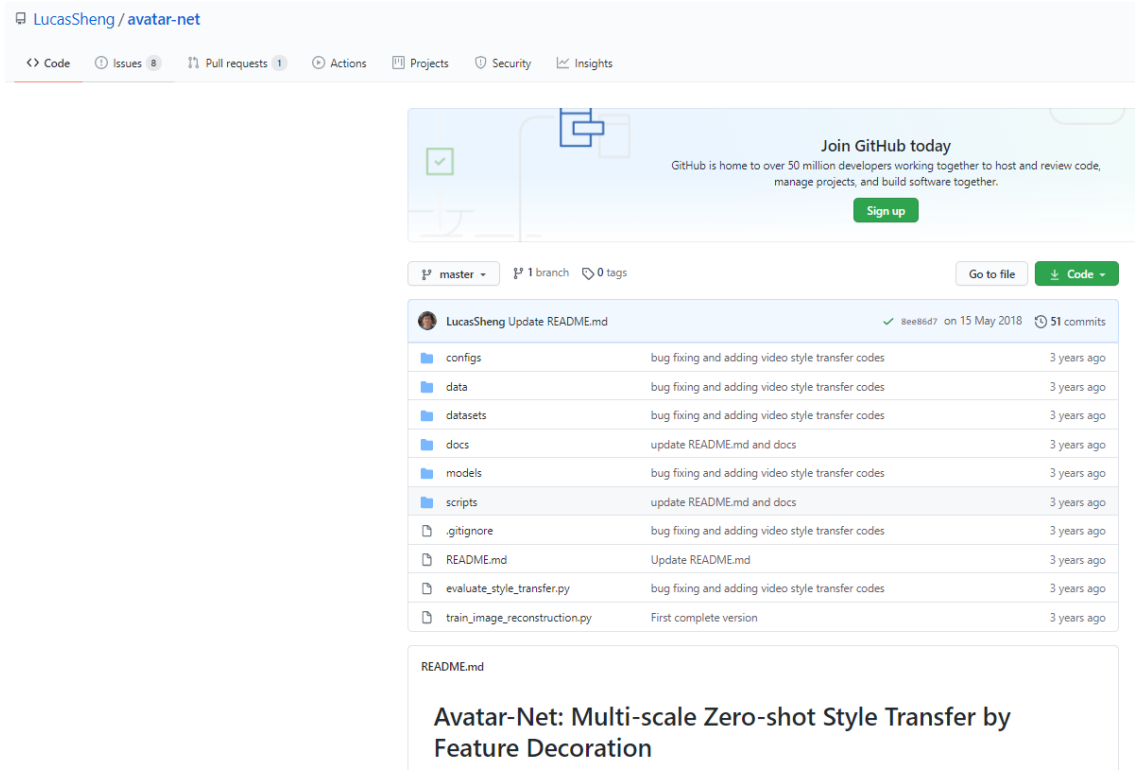
1. Total loss

2018
Style Transfer

Avatar-Net: Multi-Scale Zero-Shot Style Transfer by Feature Decoration

– Lu Sheng, Ziyi Lin, Jing Shao, Xiaogang Wang

4. Source code

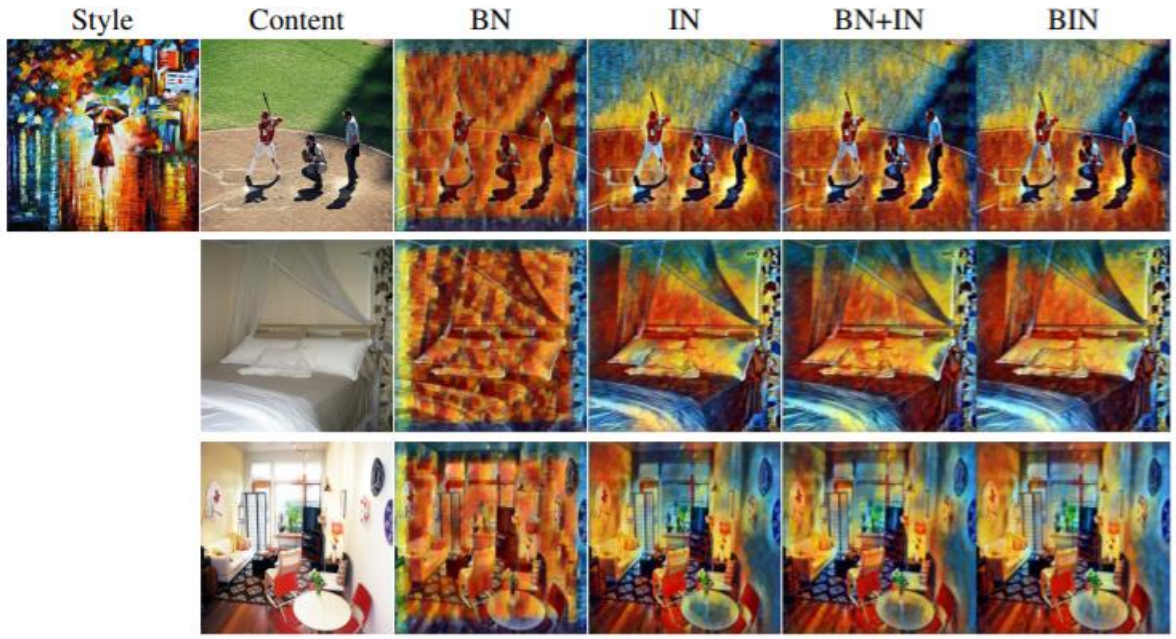


URL https://github.com/LucasSheng/avatar-net

2018
Style Transfer

Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks

- Hyeonseob Nam, Hyo-Eun Kim



1. 해결하고자 하는 문제점 및 해결 방법

- 해결하고자 하는 문제점

1. Real-world image recognition은 객체 텍스처, 조명 조건, 필터 효과 등을 포함한 시각적 스타일의 가변성

- 해결 방법

1. BIN(Batch-Instance Normalization)
    1. 유용한 스타일 유지 및 방해가 되는 스타일만 선택적으로 정규화
    2. 스타일 transfer에서 보존과 제거 간의 균형을 제어

2018
Style Transfer

Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks

- Hyeonseob Nam, Hyo-Eun Kim

2. 모델 구조

- 각 채널에 대한 스타일 transfer를 유지 or 폐기 결정하는 gate vector

$$\rho \leftarrow \mathrm{clip}_{[0,1]}\left(\rho - \eta \Delta \rho\right),$$

$$\mathbf{y} = \left(\rho \cdot \hat{\mathbf{x}}^{(B)} + (1-\rho)\cdot \hat{\mathbf{x}}^{(I)}\right)\cdot \gamma + \beta,$$

$$\hat{x}^{(B)}_{nchw} = \frac{x_{nchw} - \mu^{(B)}_{c}}{\sqrt{\sigma^{2(B)}_{c} + \epsilon}},$$

$$\mu^{(B)}_{c} = \frac{1}{NHW}\sum_{N}\sum_{H}\sum_{W} x_{nchw},$$

$$\sigma^{2(B)}_{c} = \frac{1}{NHW}\sum_{N}\sum_{H}\sum_{W}\left(x_{nchw} - \mu^{(B)}_{c}\right)^{2},$$

$$\hat{x}^{(I)}_{nchw} = \frac{x_{nchw} - \mu^{(I)}_{nc}}{\sqrt{\sigma^{2(I)}_{nc} + \epsilon}},$$

$$\mu^{(I)}_{nc} = \frac{1}{HW}\sum_{H}\sum_{W} x_{nchw},$$

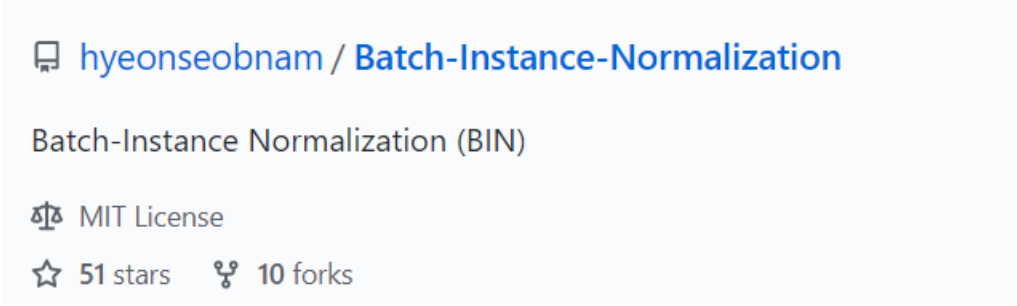
$$\sigma^{2(I)}_{nc} = \frac{1}{HW}\sum_{H}\sum_{W}\left(x_{nchw} - \mu^{(I)}_{nc}\right)^{2},$$

2018
Style Transfer

Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks

– Hyeonseob Nam, Hyo-Eun Kim

4. Source code



URL https://github.com/hyeonseobnam/Batch-Instance-Normalization

2018
Style Transfer

Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields

– Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, Mingli Song

(a) Content&Style (b) Stroke Size #1 (c) Stroke Size #2 (d) Stroke Size #3 (e) Mixed Strokes

1. 해결하고자 하는 문제점 및 해결 방법

– 해결하고자 하는 문제점

1. Stroke size를 제어
2. 반복적인 모델의 최적화 프로세스의 느린 속도

– 해결 방법

1. Stroke controllable style transfer network
    1. Stroke size에 영향을 미치는 요인을 분석하여 receptive field와 style image scale을 명시적으로 설명

2018
Style Transfer

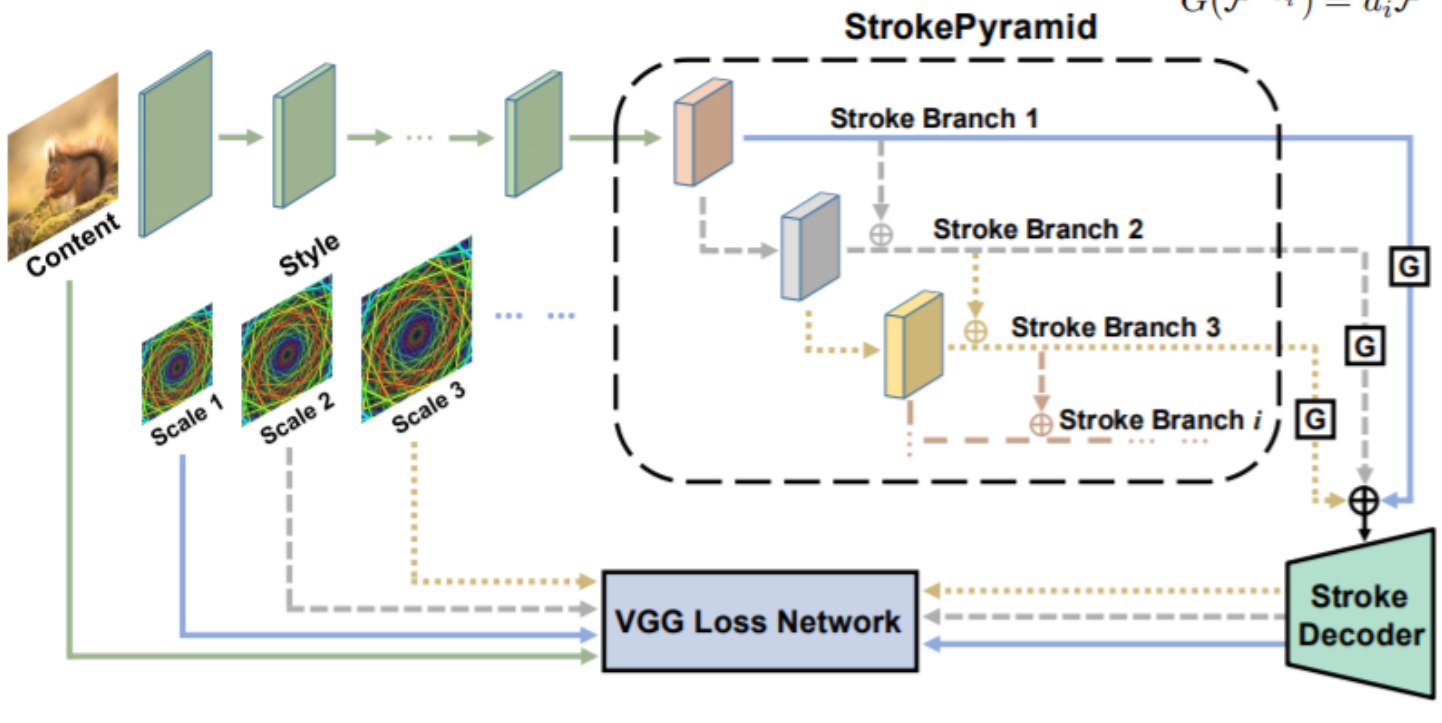
Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields

– Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, Mingli Song

2. 모델 구조

1. G : 각 분기 별 gating function
2. F : branch의 feature map

$$G(\mathcal{F}^{\mathcal{B}_{s_i}}) = a_i \mathcal{F}^{\mathcal{B}_{s_i}}, \quad \sum_i a_i = 1 \ (0 \le a_i \le 1),$$



$$Dec(\sum_i G(\mathcal{F}^{\mathcal{B}_{s_i}})) = I_o^{\mathcal{T}_k}.$$

2018
Style Transfer

Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields

– Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, Mingli Song

$$\mathcal{L}_c = \sum_{l \in \{l_c\}} \|\mathcal{F}^l(I_c) - \mathcal{F}^l(I_o)\|^2,$$

$$\mathcal{G}(\mathcal{F}^l(I_s)') = [\mathcal{F}^l(I_s)'][\mathcal{F}^l(I_s)']^T.$$

$$\mathcal{L}_{\mathcal{T}_k} = \sum_{l \in \{l_s\}} \|\mathcal{G}(\mathcal{F}^l(\mathcal{R}(I_s, \mathcal{T}_k))') - \mathcal{G}(\mathcal{F}^l(I_o^{\mathcal{B}_{s_k}})')\|^2,$$

$$\mathcal{L}_{\mathcal{B}_{s_k}} = \alpha\mathcal{L}_c + \beta_k\mathcal{L}_{\mathcal{T}_k} + \gamma\mathcal{L}_{tv},$$

3. Loss function

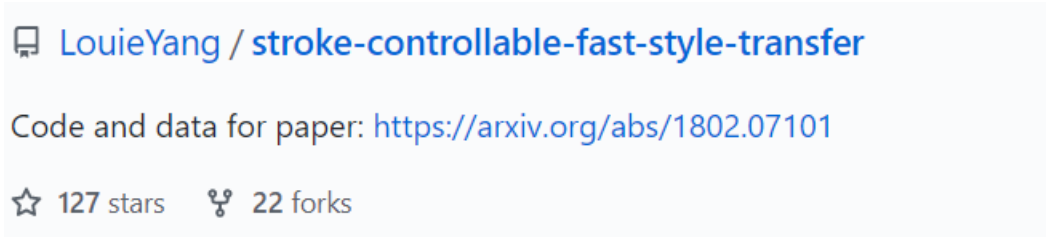
1. Content loss : Euclidian distance
2. Gram matrix
3. Stroke loss
4. Total loss

2018
Style Transfer

Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields

– Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, Mingli Song

4. Source code



URL https://github.com/LouieYang/stroke-controllable-fast-style-transfer

2018
Style Transfer

A Style-Aware Content Loss for Real-time HD Style Transfer

- Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, Bjorn Ommer

1. 해결하고자 하는 문제점 및 해결 방법

- 해결하고자 하는 문제점

1. 고흐의 이미지로 스타일을 변환시켰을 때, 고흐의 스타일을 학습한 것이 아닌 입력된 고흐의 단 한 장의 그림 스타일을 학습(단일 인스턴스로 제한)
2. 기존 모델들은 RGB 이미지 영역 또는 ImageNet에서 pre-trained된 CNN에서 비교
   → 직접 비교하기 위해 수백만 개의 label이 지정되어 있어야 함
   → 예술적인 고려없이 만들어졌기 때문에 편향 발생 가능성이 있음

- 해결 방법

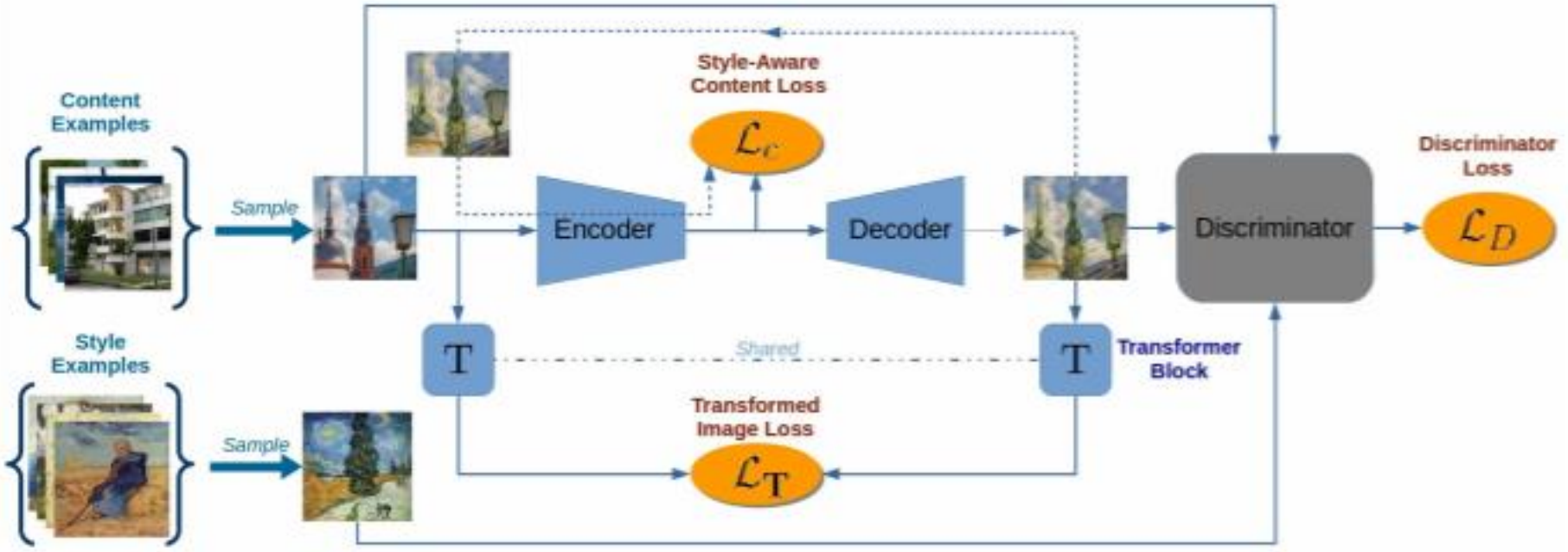
1. 심층 인코더-디코더 네트워크
2. 공동으로 훈련된 style-aware content loss

2018
Style Transfer

A Style-Aware Content Loss for Real-time HD Style Transfer

- Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, Bjorn Ommer

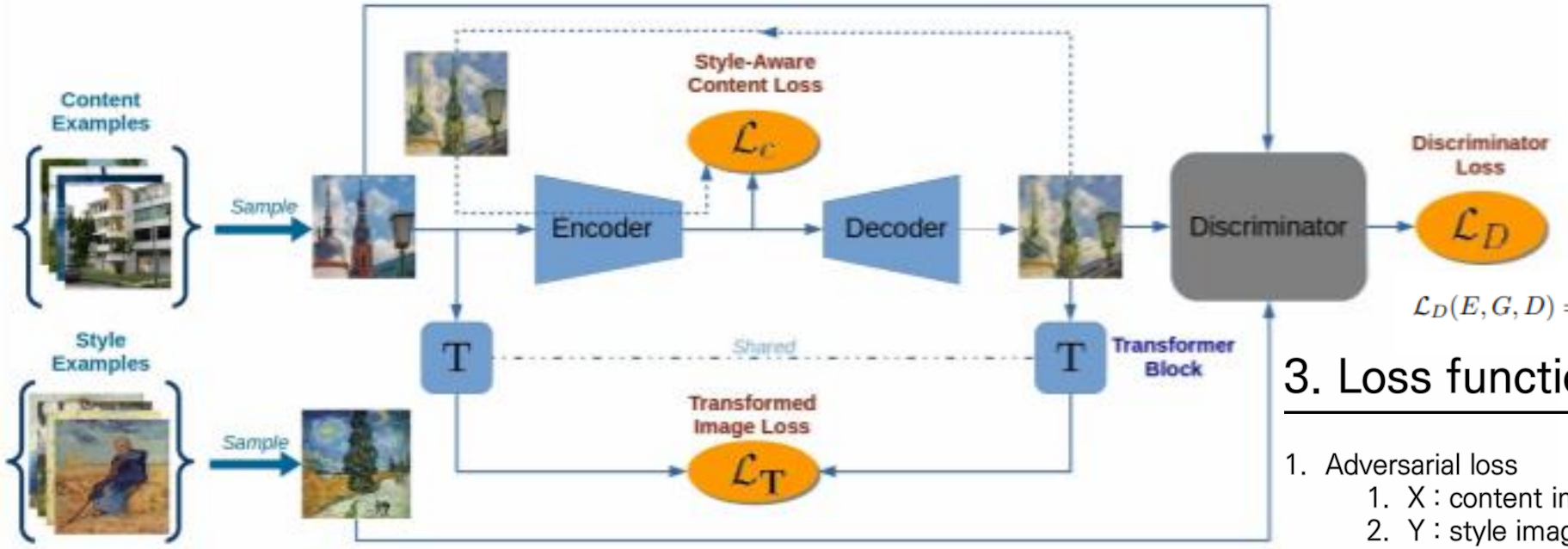
2. 모델 구조



2018
Style Transfer

A Style-Aware Content Loss for Real-time HD Style Transfer

- Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, Bjorn Ommer



$$\mathcal{L}_D(E,G,D) = \mathop{\mathbb{E}}_{y\sim p_Y(y)}[\log D(y)] + \mathop{\mathbb{E}}_{x\sim p_X(x)}[\log(1 - D(G(E(x))))]$$

3. Loss function

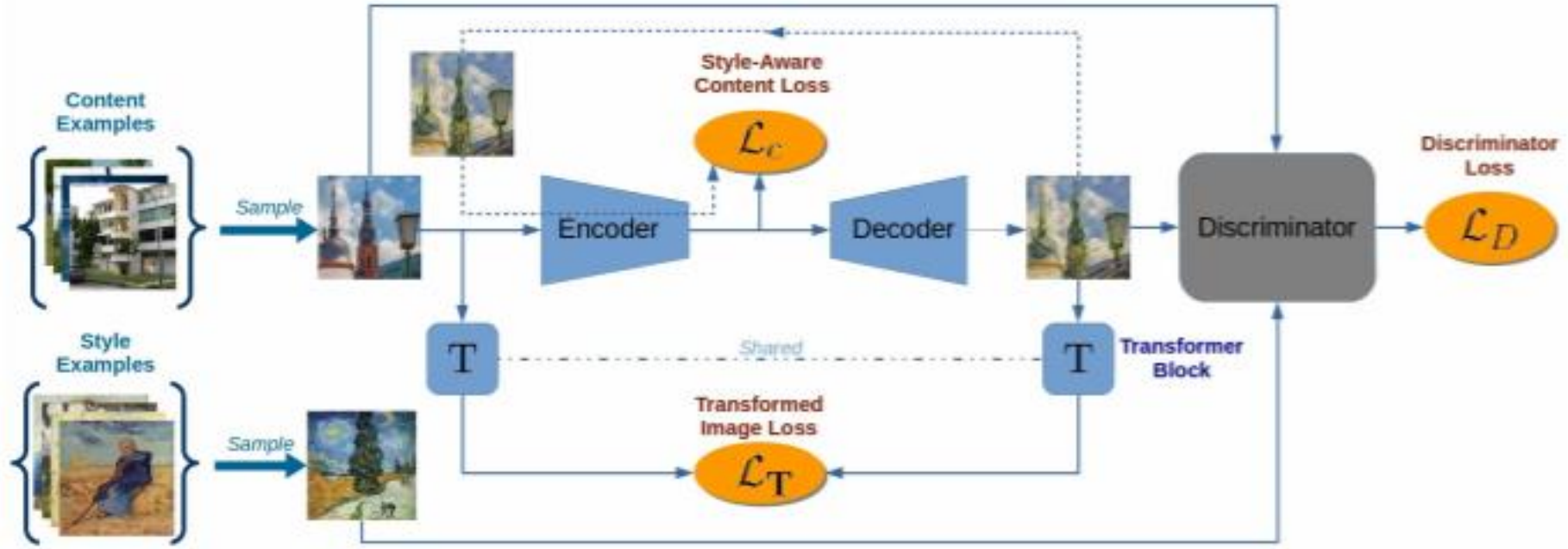
1. Adversarial loss
    1. X : content image
    2. Y : style image
    3. E(X) : latent representation
    4. G(E(X)) : stylized image

2018
Style Transfer

A Style-Aware Content Loss for Real-time HD Style Transfer

- Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, Bjorn Ommer

$$\mathcal{L}_c(E,G) = \mathop{\mathbb{E}}_{x\sim p_X(x)}\left[\frac{1}{d}\left\|E(x) - E(G(E(x)))\right\|_2^2\right]$$



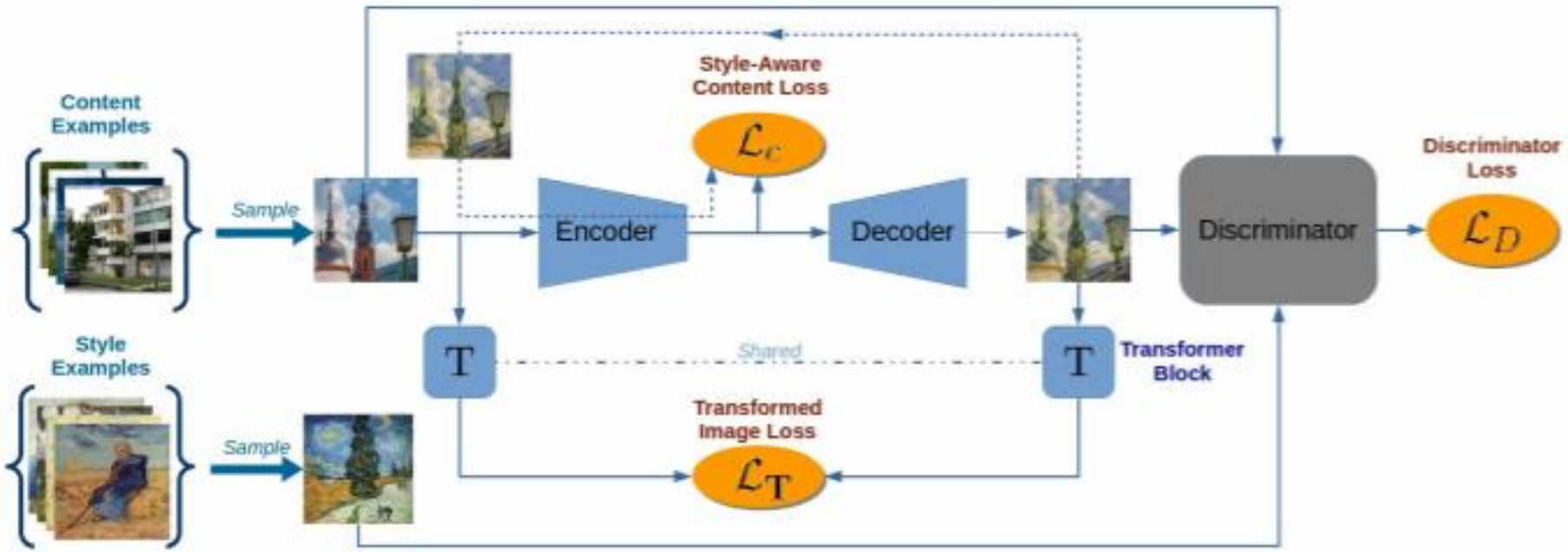
3. Loss function

2. Style-aware content loss
    1. E(X): latent representation
    2. E(G(E(X))): 생성된 이미지의 latent representation
    3. d: latent vector's dimension
    4. 실제 이미지를 비교하는 것이 아닌 latent vector를 비교하는 것이 핵심
    5. Euclidean distance 사용

2018
Style Transfer

A Style-Aware Content Loss for Real-time HD Style Transfer

- Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, Bjorn Ommer



$$\mathcal{L}_{\mathbf{T}}(E, G) = \mathop{\mathbb{E}}_{x \sim p_X(x)} \left[\frac{1}{CHW} ||\mathbf{T}(x) - \mathbf{T}(G(E(x))||_2^2 \right],$$

3. Loss function

3. Transformed image loss
    1. T : 하나의 Fully convolution neural network
    2. Unsupervised learning 을 위해 모델을 훈련
    3. Input image x와 output image G(E(x))에 약한 변환을 적용하여 차이를 측정하는 loss

2018
Style Transfer

A Style-Aware Content Loss for Real-time HD Style Transfer

- Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, Bjorn Ommer

$$\mathcal{L}(E, G, D) = \mathcal{L}_c(E, G) + \mathcal{L}_t(E, G) + \lambda \mathcal{L}_D(E, G, D),$$

$$E, G = \arg \min_{E,G} \max_{D} \mathcal{L}(E, G, D).$$

3. Loss function

1. Total loss

2018
Style Transfer

A Style-Aware Content Loss for Real-time HD Style Transfer

- Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, Bjorn Ommer

4. Source code

URL https://github.com/CompVis/adaptive-style-transfer

2018
Style Transfer

CartoonGAN: Generative Adversarial Networks for Photo Cartoonization

– Yang Chen, Yu-Kun Lai, Yong-Jin Liu

(a) Original scene (b) Our result

1. 해결하고자 하는 문제점 및 해결 방법

– 해결하고자 하는 문제점

1. 기존의 style transfer
    1. 만화 스타일이 높은 수준의 단순화 및 추상화와 함께 고유한 특성을 가지고 있음
    2. 만화 이미지가 선명한 가장 자리, 부드러운 색상 음영 및 비교적 단순한 경향

    ➔ 만화화에 대한 만족스러운 결과를 생성하기 어려움

– 해결 방법

1. CartoonGAN
    1. VGG network의 high-level feature map에서 사진과 만화 간의 상당한 스타일 transfer에 대처하는 content loss(semantic loss)
    2. 명확한 edge를 유지하기 위한 adversarial loss
    3. Unpaired data 사용

2018
Style Transfer

CartoonGAN: Generative Adversarial Networks for Photo Cartoonization

– Yang Chen, Yu-Kun Lai, Yong-Jin Liu

2. 모델 구조

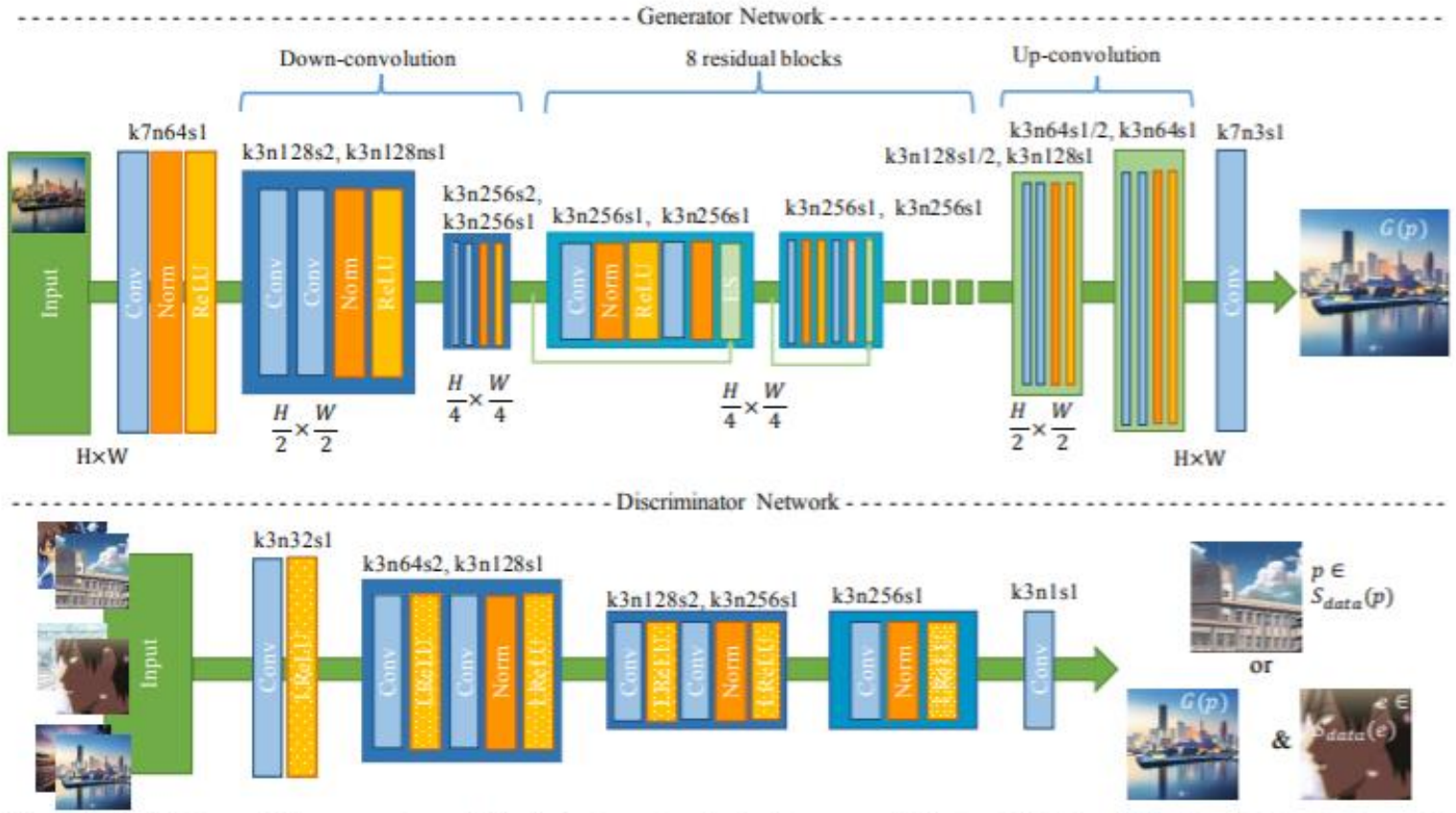


Figure 2. Architecture of the generator and discriminator networks in the proposed CartoonGAN, in which k is the kernel size, n is the number of feature maps and s is the stride in each convolutional layer, 'norm' indicates a normalization layer and 'ES' indicates elementwise sum.

2018
Style Transfer

CartoonGAN: Generative Adversarial Networks for Photo Cartoonization

– Yang Chen, Yu-Kun Lai, Yong-Jin Liu

$$\mathcal{L}(G, D) = \mathcal{L}_{adv}(G, D) + \omega \mathcal{L}_{con}(G, D),$$

$$\begin{aligned}\mathcal{L}_{adv}(G, D) = & \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)] \\ & + \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))] \\ & + \mathbb{E}_{p_k \sim S_{data}(p)}[\log(1 - D(G(p_k)))].\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{con}(G, D) = & \\ & \mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1]\end{aligned}$$

3. Loss function

1. Total loss
    1. W는 균형을 위해 추가
2. Adversarial loss
    1. G, D에 모두 적용
    2. C_i: cartoon image, e_i: removing clear edges, p_k: photo
    3. 선명한 edge의 표현이 만화 이미지에서 중요한 특징이지만 이 비율을 현저히 적기 때문에 명확하게 재현된 edge가 없지만 정확한 음영이 있는 출력 이미지는 standard loss와 혼동할 가능성이 있음
3. Content loss
    1. 생성된 이미지와 기존 이미지 사이의 l1 sparse regulaization

2018
Style Transfer

CartoonGAN: Generative Adversarial Networks for Photo Cartoonization

– Yang Chen, Yu-Kun Lai, Yong-Jin Liu

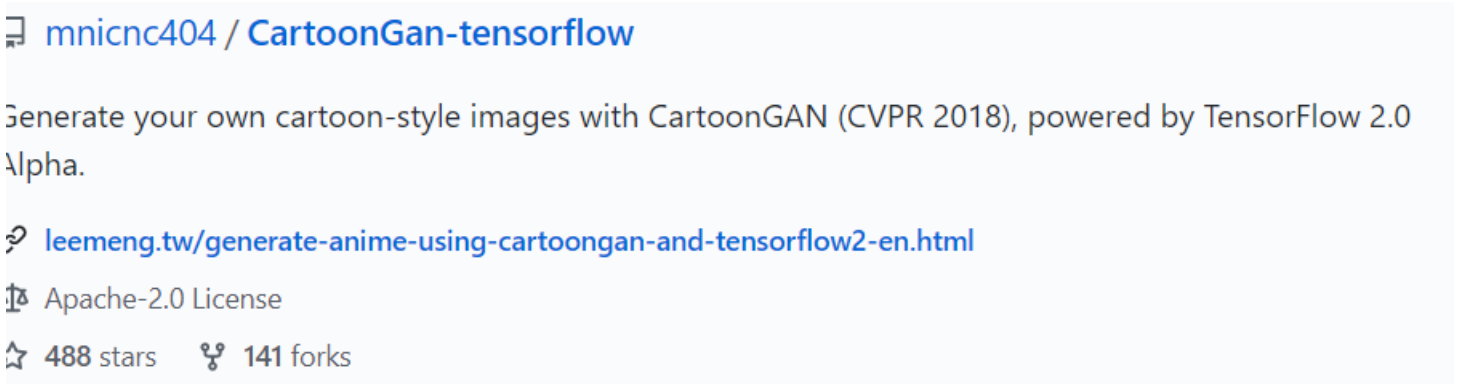
3.1. edge loss 유뮤 비교

2018
Style Transfer

CartoonGAN: Generative Adversarial Networks for Photo Cartoonization

– Yang Chen, Yu-Kun Lai, Yong-Jin Liu

4. Source code



URL https://github.com/mnicnc404/CartoonGan-tensorflow

2019
Style Transfer

Style Transfer by Relaxed Optimal Transport and Self-Similarity

– Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

1. 해결하고자 하는 문제점 및 해결 방법

– 해결하고자 하는 문제점

1. 특정 visual effect를 얻기
2. Unconstrained style transfer에 의한 correct errors 수정

– 해결 방법

1. STROTSS(Style Transfer by Relaxed Optimal Transport and Self-Similarity)

2019
Style Transfer

Style Transfer by Relaxed Optimal Transport and Self-Similarity

- Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

2. 모델 구조

2019
Style Transfer

Style Transfer by Relaxed Optimal Transport and Self-Similarity

– Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

Stylization에 대한 content preserving의 상대적 중요성

$$L(X, I_C, I_S) = \frac{\alpha \ell_C + \ell_m + \ell_r + \frac{1}{\alpha}\ell_p}{2 + \alpha + \frac{1}{\alpha}}$$

3. Loss function

1. Objective functions
    1. l_c : content loss
    2. l_m: Moment matching loss
    3. l_r : REMD(Relaxed EMD)
    4. l_p : Coloring matching loss

2019
Style Transfer

Style Transfer by Relaxed Optimal Transport and Self-Similarity

– Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

$$\mathbf{EMD}(A,B) = \min_{T \geq 0} \sum_{ij} T_{ij} C_{ij}$$

$$s.t. \sum_j T_{ij} = 1/m$$

$$\sum_i T_{ij} = 1/n$$

1. EMD : Image $X^{(t)}$(time step t에서의 중간 output image)에서 추출한 feature vector A와 I_s에서 추출한 feature vector 분포 간의 차이 측정
2. T : transport matrix
3. C : cost matrix (element A와 element B 사이의 distance 측정)
4. 그러나 최적의 T를 찾기 위해서 $O((m,n)^3)$이기 때문에 Relaxed EMD를 사용

$$\ell_r = REMD(A,B) = \max(R_A(A,B), R_B(A,B))$$

$$R_A(A,B) = \min_{T \geq 0} \sum_{ij} T_{ij} C_{ij} \quad s.t. \quad \sum_j T_{ij} = 1/m \quad (5)$$

$$R_B(A,B) = \min_{T \geq 0} \sum_{ij} T_{ij} C_{ij} \quad s.t. \quad \sum_i T_{ij} = 1/n \quad (6)$$

$$\ell_r = \max\left(\frac{1}{n}\sum_i \min_j C_{ij}, \frac{1}{m}\sum_j \min_i C_{ij}\right)$$

$$C_{ij} = D_{\cos}(A_i, B_j) = 1 - \frac{A_i \cdot B_j}{\|A_i\| \|B_j\|}$$

3. Loss function

1. REMD
    1. REMD : C(cost matrix)를 계산하자!
    2. A_i에서 B_i인 두 feature vector 간의 코사인 거리로 계산
    3. 단, cosine distance는 feature vector의 크기를 무시 → output에 시각적 artifact, 특히 과/저 채도를 초래 → moment matching loss 추가

2019
Style Transfer

Style Transfer by Relaxed Optimal Transport and Self-Similarity

– Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

Feature vectors set A의 평균

Feature vectors set A의 공분산

$$\ell_m = \frac{1}{d}\|\mu_A - \mu_B\|_1 + \frac{1}{d^2}\|\Sigma_A - \Sigma_B\|_1$$

Feature vectors set B의 평균

Feature vectors set B의 공분산

3. Loss function

1. Moment matching loss
    1. Mean and covariance of the feature vectors set A
    2. Mean and covariance of the feature vectors set B
2. Coloring matching loss l_p
    1. 유사한 팔레트를 갖도록 하는 loss
    2. Relaxed EMD 사용
    3. RGB의 색상을 하나의 채널로 사용하는 평균 색상 공간으로 변환하는 것이 유익
    4. 이는 콘텐츠 preserving과 상충되기 때문에 $\frac{1}{\alpha}$을 가중

2019
Style Transfer

Style Transfer by Relaxed Optimal Transport and Self-Similarity

– Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

$$\mathcal{L}_{content}(X, C) = \frac{1}{n^2} \sum_{i,j} \left| \frac{D^X_{ij}}{\sum_i D^X_{ij}} - \frac{D^{I_C}_{ij}}{\sum_i D^{I_C}_{ij}} \right|$$

$$C_{ij} = \begin{cases} \beta * D_{cos}(A_i, B_j), \text{ if } i \in X_{tk}, j \in S_{sk} \\ \infty, \text{ if } \exists k \text{ s.t. } i \in X_{tk}, j \notin S_{sk} \\ D_{cos}(A_i, B_j) \text{ otherwise,} \end{cases}$$

3. Loss function

1. Content loss
    1. D_{ij}^X : $X^{(t)}$에서 추출한 모든 feature vector의 쌍별(pair-wise)cosine distance matrix
    2. D_{ij}^{IC} : I_c에서 추출한 모든 feature vector의 쌍별(paire-wise) cosine distance matrix

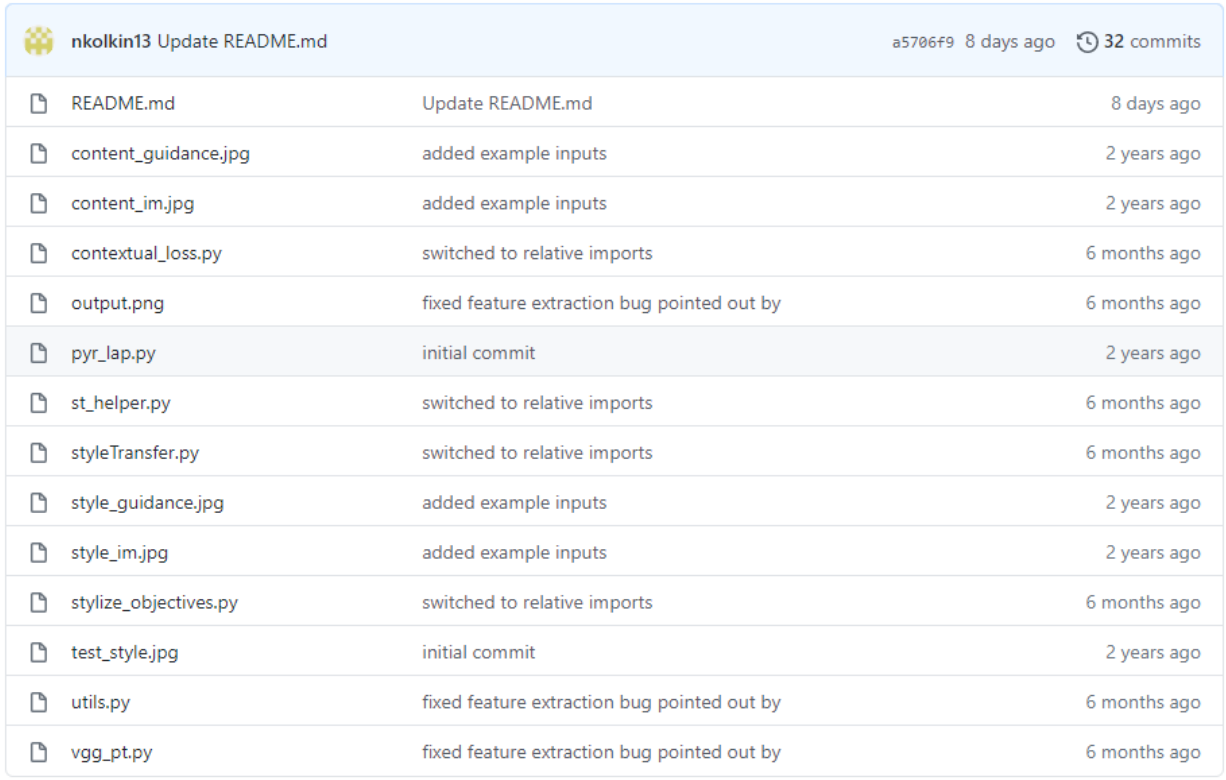
- 파란색, 빨간색 및 녹색 heatmap은 사진에 표시된 해당 지점과 관련된 feature 공간의 코사인 유사성을 시각화 함
- Content loss는 content image에서 무작위로 선택한 1024개의 위치 간 상대적인 쌍별 유사성을 유지하려 함

2019
Style Transfer

Style Transfer by Relaxed Optimal Transport and Self-Similarity

– Nicholas Kolkin, Jason Salavon, Gregory Shakhnarovich

4. Source code



nkolkin13 Update README.md — a5706f9 8 days ago — 32 commits

README.md	Update README.md	8 days ago
content_guidance.jpg	added example inputs	2 years ago
content_im.jpg	added example inputs	2 years ago
contextual_loss.py	switched to relative imports	6 months ago
output.png	fixed feature extraction bug pointed out by	6 months ago
pyr_lap.py	initial commit	2 years ago
st_helper.py	switched to relative imports	6 months ago
styleTransfer.py	switched to relative imports	6 months ago
style_guidance.jpg	added example inputs	2 years ago
style_im.jpg	added example inputs	2 years ago
stylize_objectives.py	switched to relative imports	6 months ago
test_style.jpg	initial commit	2 years ago
utils.py	fixed feature extraction bug pointed out by	6 months ago
vgg_pt.py	fixed feature extraction bug pointed out by	6 months ago

URL https://github.com/nkolkin13/STROTSS

Thank you