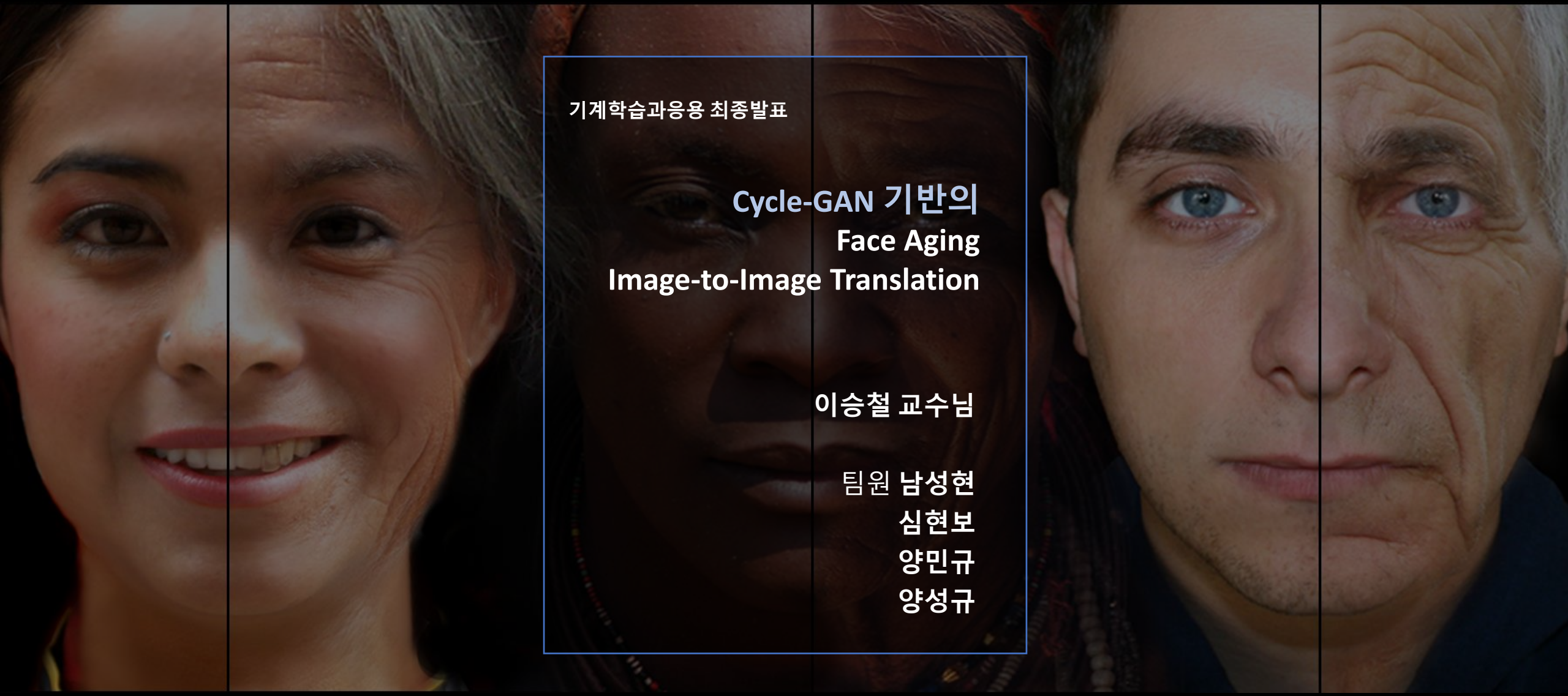


기계학습과응용 최종발표

Cycle-GAN 기반의 Face Aging Image-to-Image Translation

이승철 교수님

팀원 남성현
심현보
양민규
양성규

Cycle GAN을 활용한 Face Aging Image-to-image Translation

1. **Introduction: Face Aging**
2. **GAN / Cycle GAN**
3. **Google Colab**
4. **Data Collection**
5. **Modeling**
    1) Overview
    2) ResNet and Loss
    3) Train and Test
6. **Result Analysis**
7. **Model Evaluation**

주제 설명

Face Aging using Image-to-image Transition

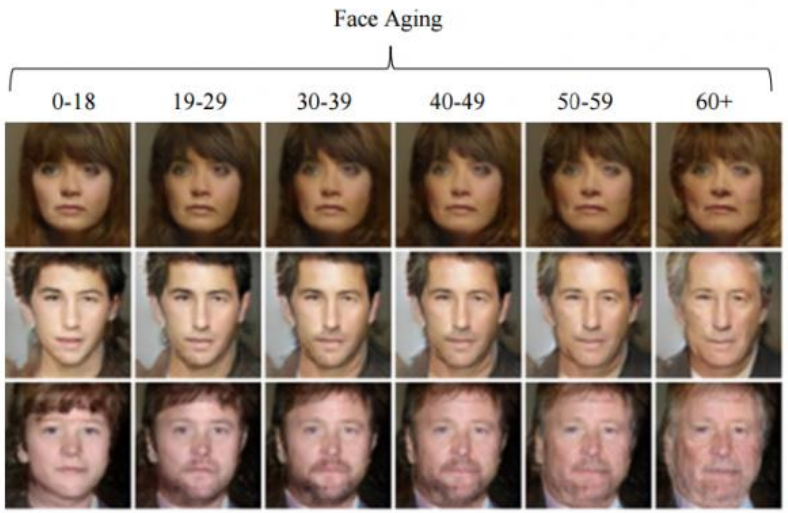
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

주제 설명

Face Aging using Image-to-image Transition

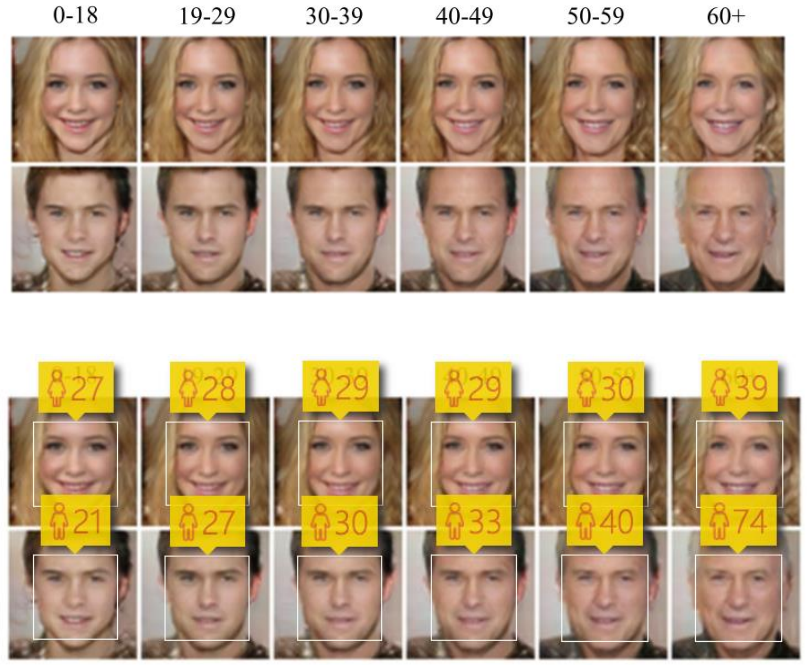
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

Face Aging with Conditional GAN (CVPR 2017)



Synthesizing the face images of one person at different ages

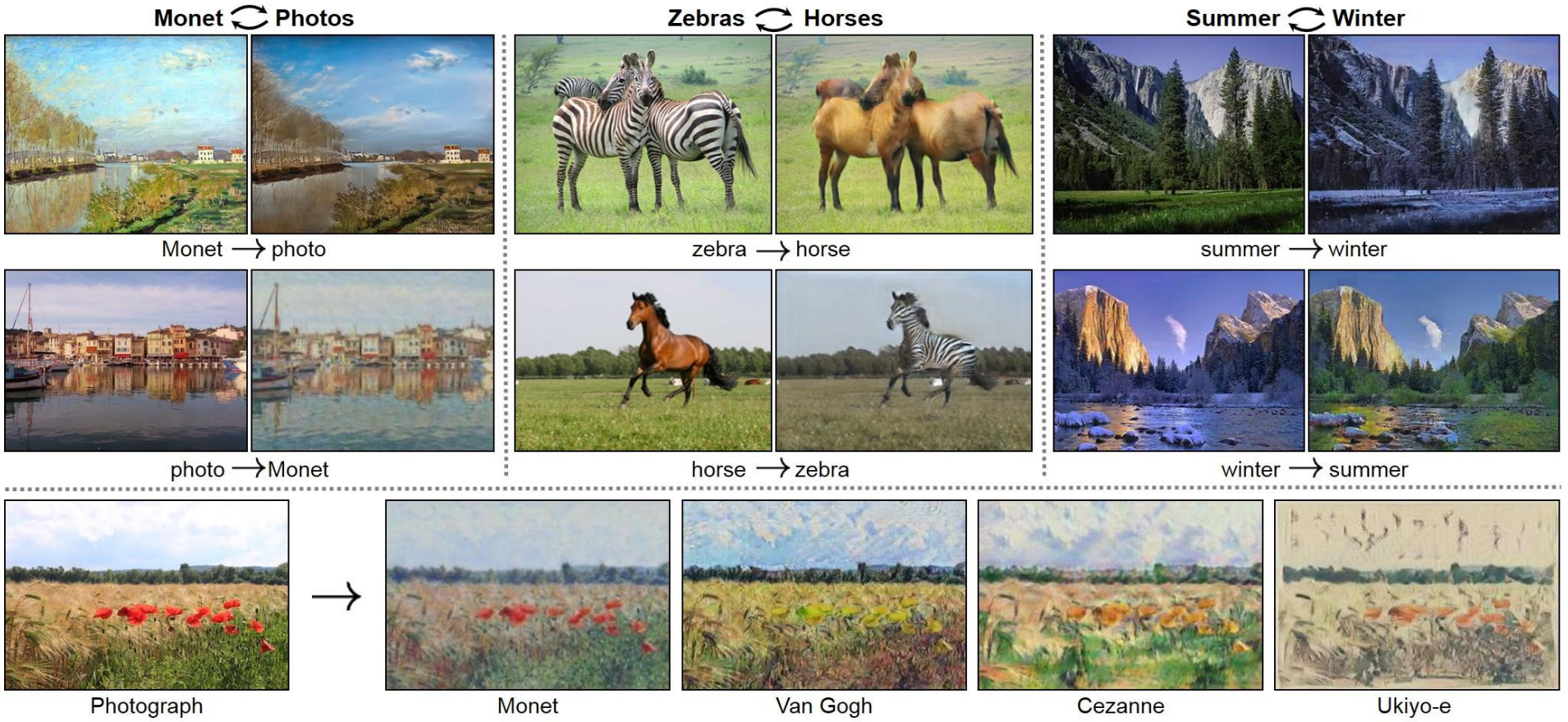
Test result with *how-old.net* by Microsoft



알고리즘 설명

Image-to-Image Translation

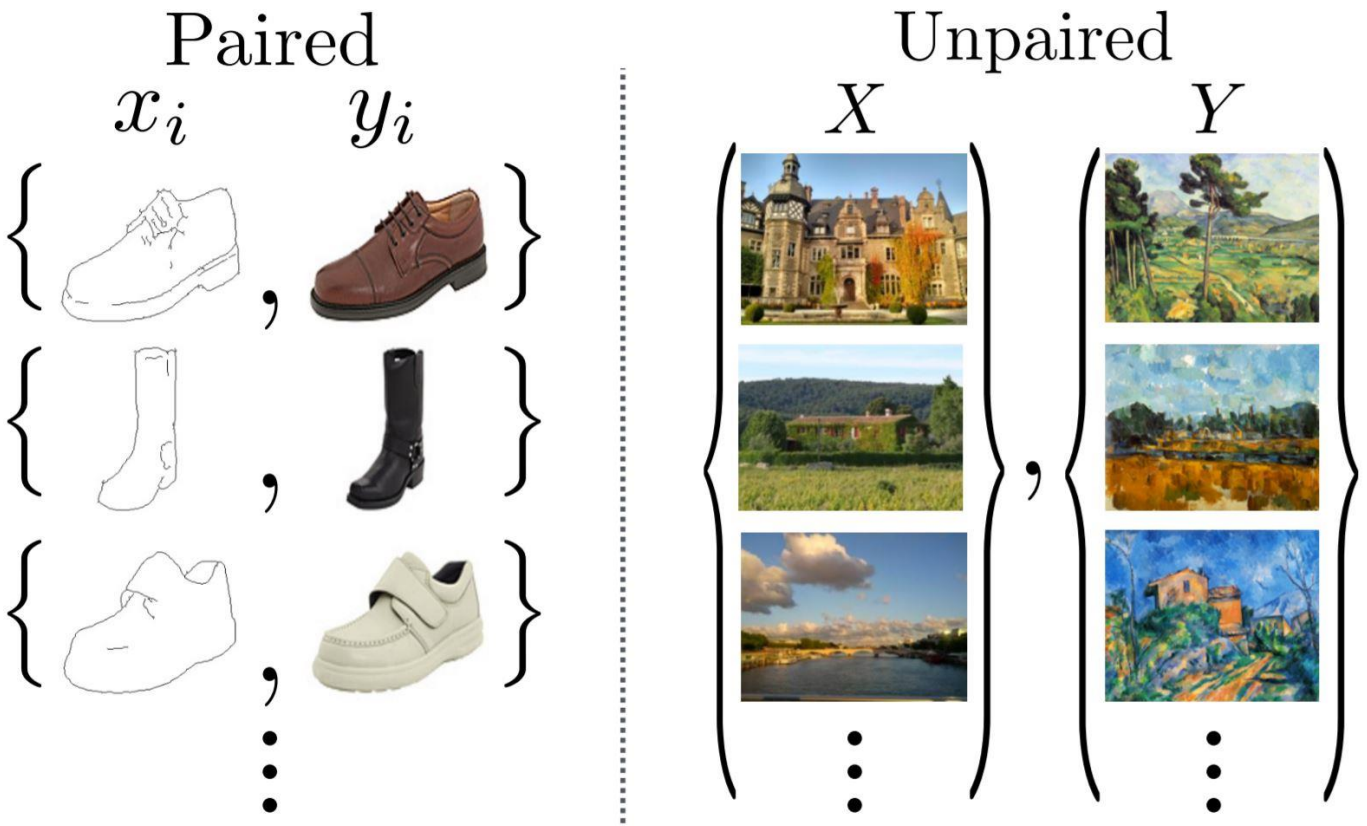
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation



알고리즘 설명

Image-to-Image Translation

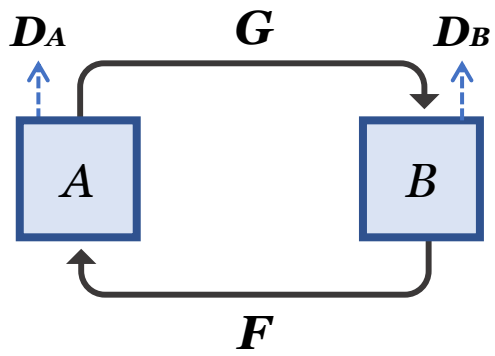
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation



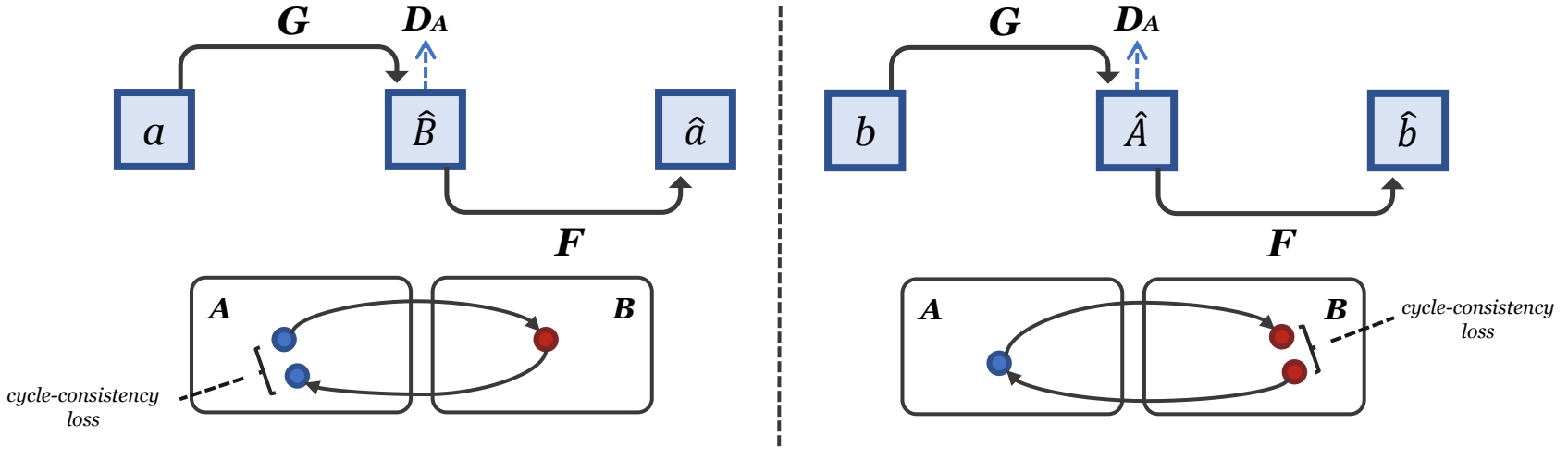
알고리즘 설명

CycleGAN

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation



The Framework of CycleGAN

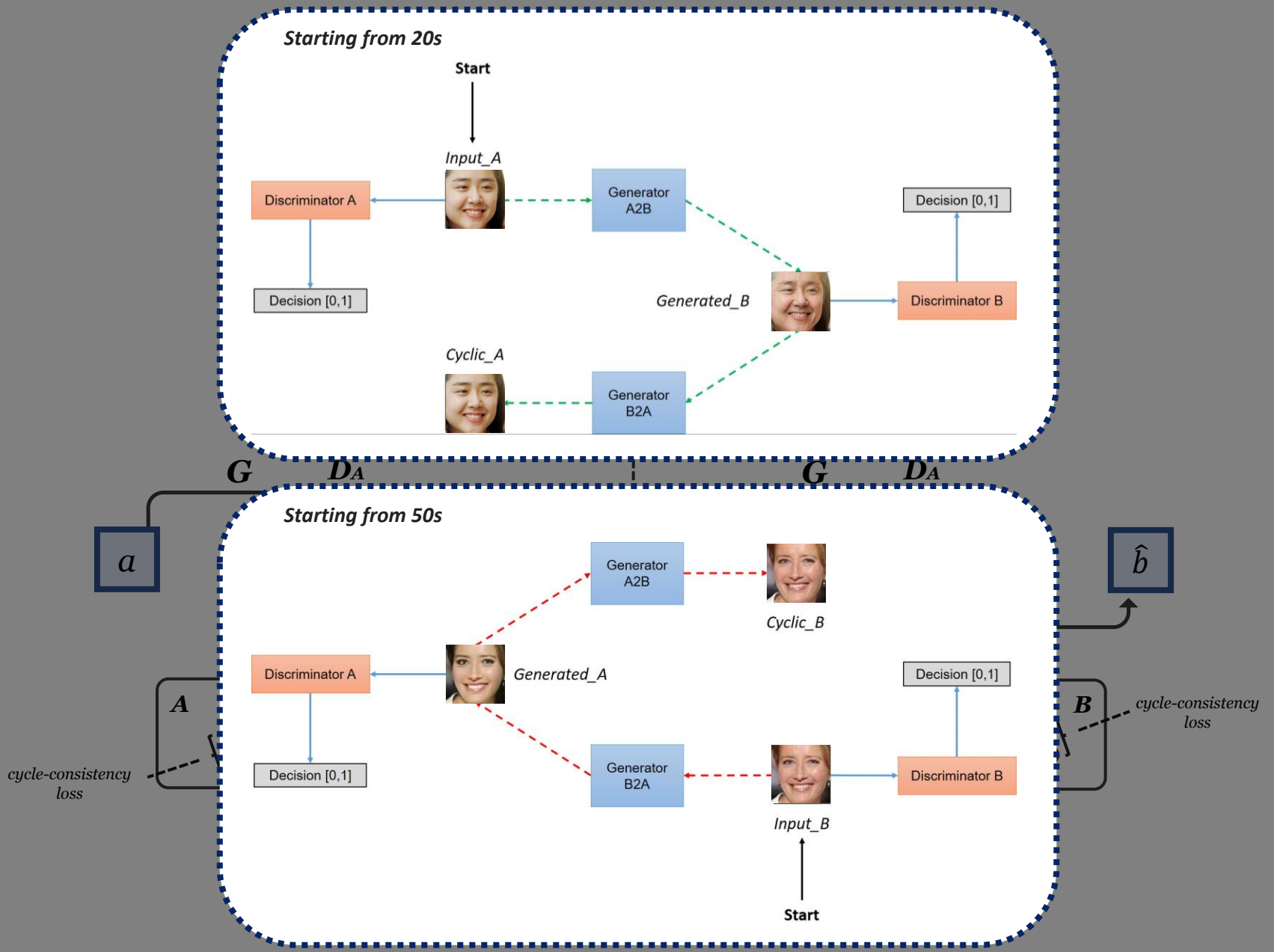


The Cycle-Consistency in CycleGAN

알고리즘 설명

CycleGAN

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation



데이터 수집

Face Dataset

- Face Aging
- Cycle GAN
- **Data Collection**
- Google Colab
- Modeling
- Train & Test
- Evaluation

UTK Face Dataset

- Consists of **20,000+ face images**
  (Only single face in one image)
- Correspondingly **aligned & cropped faces**
- Images are labelled by **age and gender**

Aligned & Cropped Faces

- 18~25 year old **1056 images**
- 50~60 year old **1056 images**

Image format

- 256×256

개발 환경

Google Colab

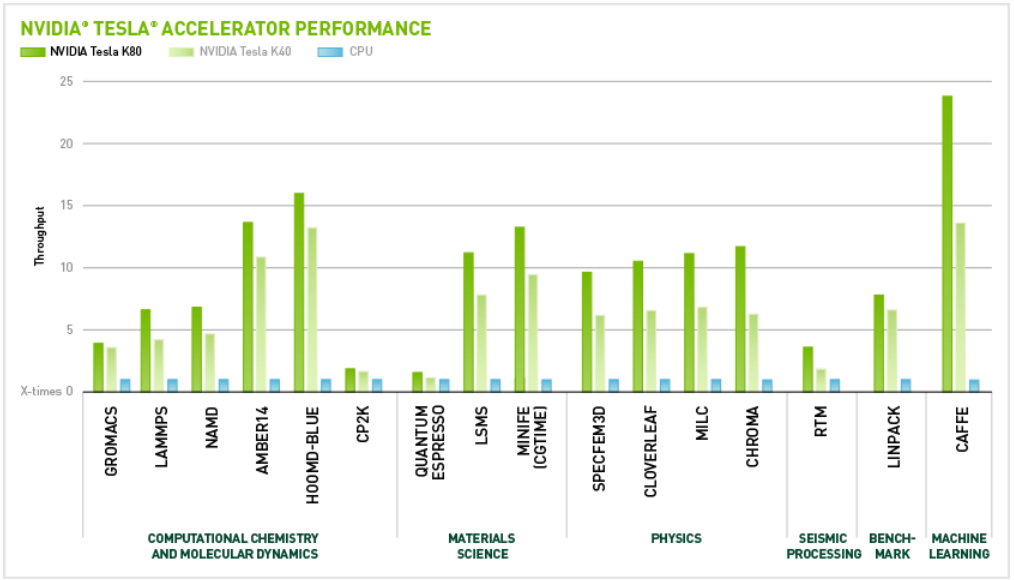
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

GPU Enabled Environment

- Tesla K80 GPU : Over $1,700
- Colab : Free (12hr per day)

Performance

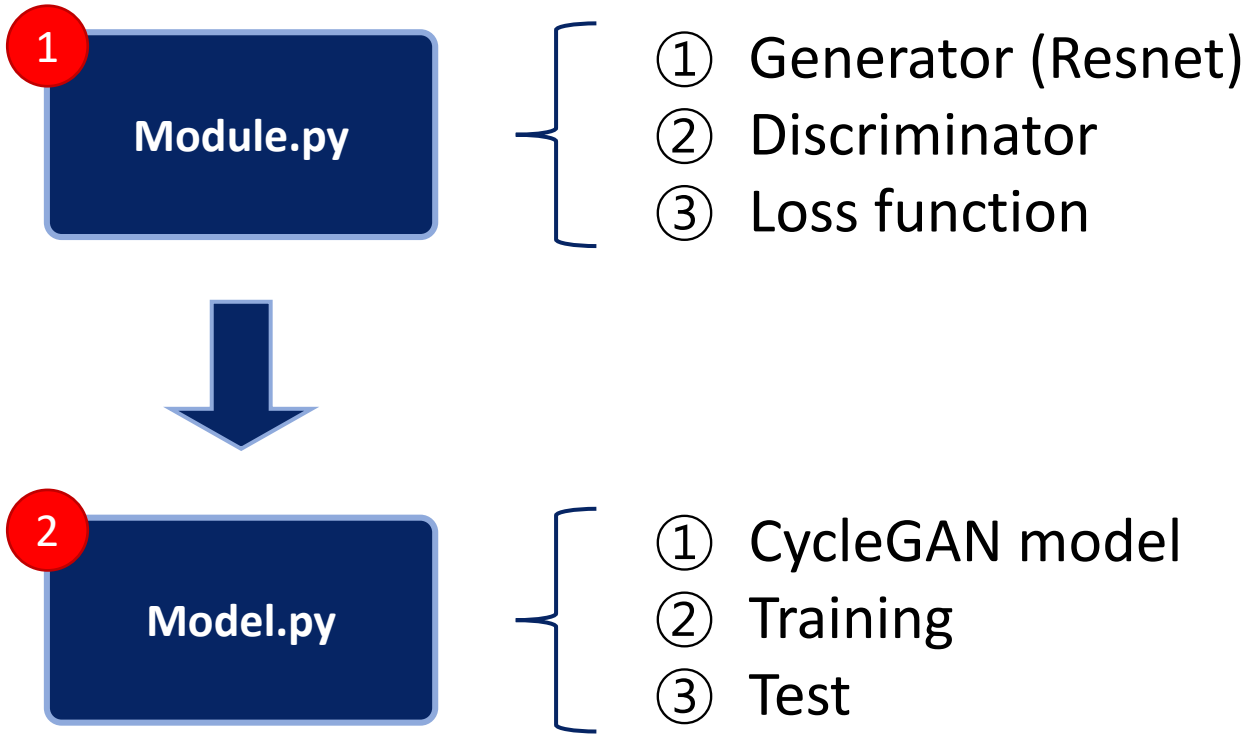
The most powerful GPU available without costs



모델 설명

Overview

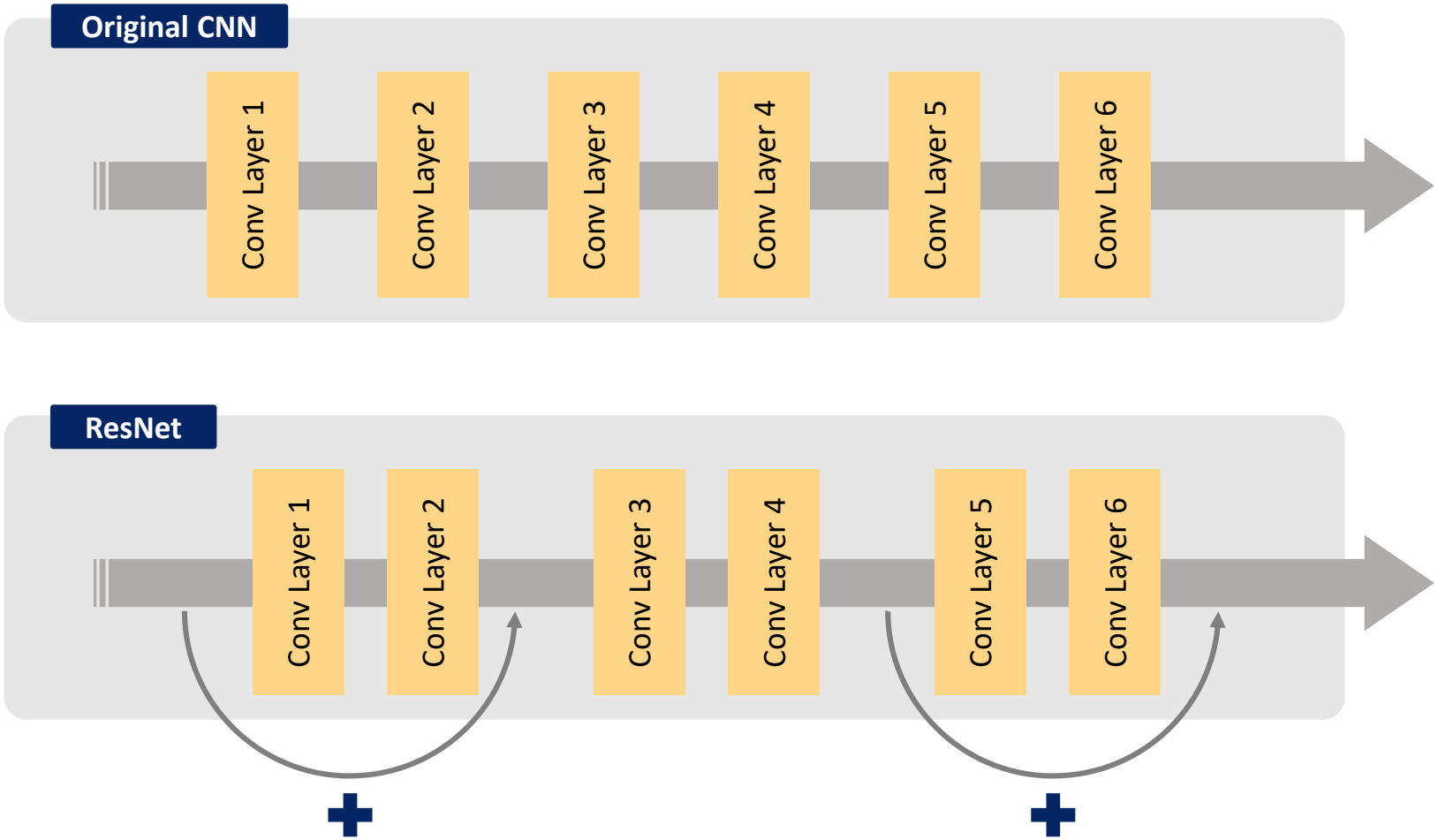
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- **Modeling**
- Train & Test
- Evaluation



모델 설명

ResNet Background

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- **Modeling**
- Train & Test
- Evaluation



모델 설명

Generator

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

Building the generator

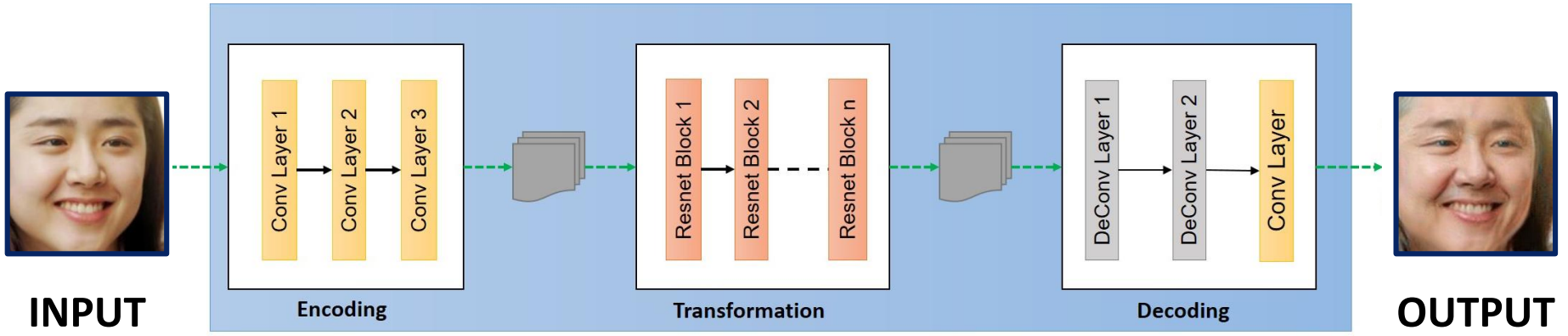


Image Size

모델 설명

Generator

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

```
# Justin Johnson's model from https://github.com/jcjohnson/fast-neural-style/
# The network with 9 blocks consists of: c7s1-32, d64, d128, R128, R128, R128,
# R128, R128, R128, R128, R128, R128, u64, u32, c7s1-3
c0 = tf.pad(image, [[0, 0], [3, 3], [3, 3], [0, 0]], "REFLECT")     # 위 아래 가로 세로로 패딩 3씩.
c1 = tf.nn.relu(instance_norm(conv2d(c0, options.gf_dim, 7, 1, padding='VALID', name='g_e1_c'), 'g_e1_bn'))    # 패딩을 이미해서 valid인듯.
c2 = tf.nn.relu(instance_norm(conv2d(c1, options.gf_dim*2, 3, 2, name='g_e2_c'), 'g_e2_bn'))
c3 = tf.nn.relu(instance_norm(conv2d(c2, options.gf_dim*4, 3, 2, name='g_e3_c'), 'g_e3_bn'))        # 64 *64

# define G network with 9 resnet blocks
# 총 cnn 18층.
r1 = residule_block(c3, options.gf_dim*4, name='g_r1')
r2 = residule_block(r1, options.gf_dim*4, name='g_r2')
r3 = residule_block(r2, options.gf_dim*4, name='g_r3')
r4 = residule_block(r3, options.gf_dim*4, name='g_r4')
r5 = residule_block(r4, options.gf_dim*4, name='g_r5')
r6 = residule_block(r5, options.gf_dim*4, name='g_r6')
r7 = residule_block(r6, options.gf_dim*4, name='g_r7')
r8 = residule_block(r7, options.gf_dim*4, name='g_r8')
r9 = residule_block(r8, options.gf_dim*4, name='g_r9')

d1 = deconv2d(r9, options.gf_dim*2, 3, 2, name='g_d1_dc')
d1 = tf.nn.relu(instance_norm(d1, 'g_d1_bn'))
d2 = deconv2d(d1, options.gf_dim, 3, 2, name='g_d2_dc')
d2 = tf.nn.relu(instance_norm(d2, 'g_d2_bn'))
d2 = tf.pad(d2, [[0, 0], [3, 3], [3, 3], [0, 0]], "REFLECT")

# 위에 패딩을 해놨고 7 by 7 필터사용.
pred = tf.nn.tanh(conv2d(d2, options.output_c_dim, 7, 1, padding='VALID', name='g_pred_c'))
```

UTPUT

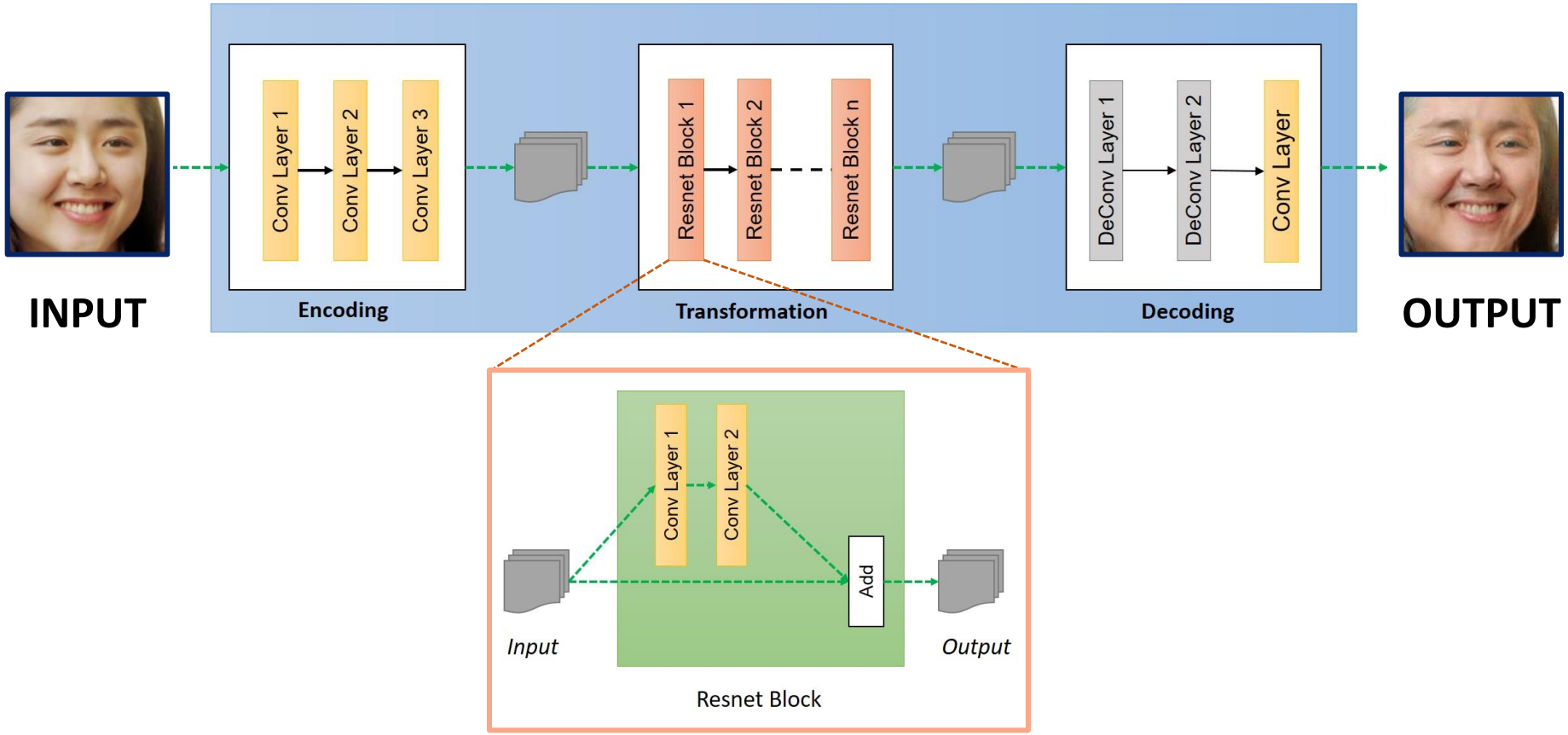
56

모델 설명

Generator

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

Building the generator



모델 설명

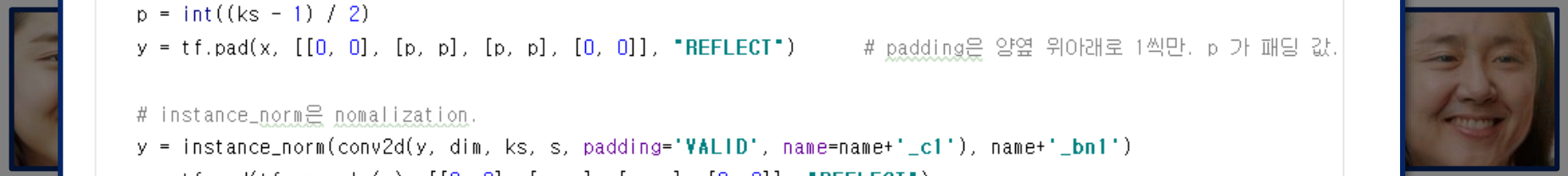
Generator

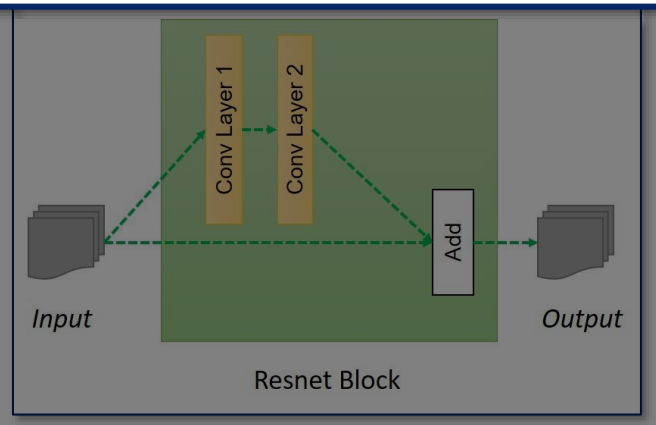
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

Building the generator

```
def residule_block(x, dim, ks=3, s=1, name='res'):
    p = int((ks - 1) / 2)
    y = tf.pad(x, [[0, 0], [p, p], [p, p], [0, 0]], "REFLECT")      # padding은 양옆 위아래로 1씩만. p 가 패딩 값.

    # instance_norm은 nomalization.
    y = instance_norm(conv2d(y, dim, ks, s, padding='VALID', name=name+'_c1'), name+'_bn1')
    y = tf.pad(tf.nn.relu(y), [[0, 0], [p, p], [p, p], [0, 0]], "REFLECT")
    y = instance_norm(conv2d(y, dim, ks, s, padding='VALID', name=name+'_c2'), name+'_bn2')
    return y + x           # 1/2을 안해도 되나??? 뒤에 노말라이즈해서 상관없는건가....
```



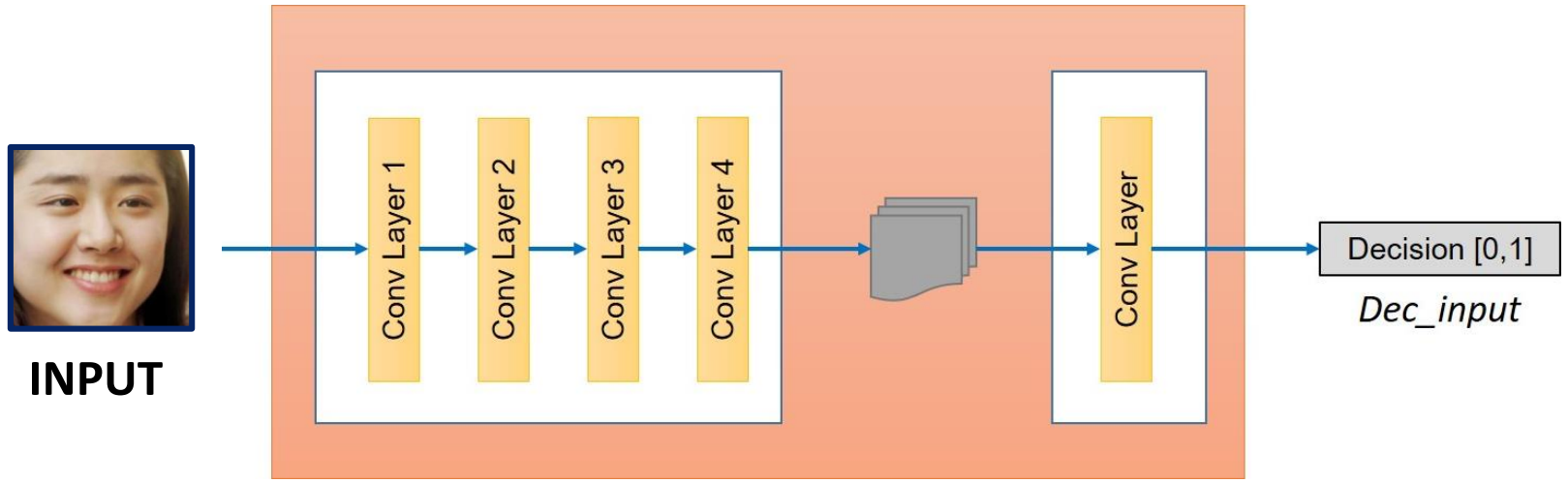


모델 설명

Discriminator

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

Building the discriminator



모델 설명

Discriminator

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

Building the discriminator

```
h0 = lrelu(conv2d(image, options.df_dim, name='d_h0_conv'))
# h0 is (128 x 128 x self.df_dim)
h1 = lrelu(instance_norm(conv2d(h0, options.df_dim*2, name='d_h1_conv'), 'd_bn1'))
# h1 is (64 x 64 x self.df_dim*2)
h2 = lrelu(instance_norm(conv2d(h1, options.df_dim*4, name='d_h2_conv'), 'd_bn2'))
# h2 is (32x 32 x self.df_dim*4)
h3 = lrelu(instance_norm(conv2d(h2, options.df_dim*8, s=1, name='d_h3_conv'), 'd_bn3'))
# h3 is (32 x 32 x self.df_dim*8)
h4 = conv2d(h3, 1, s=1, name='d_h3_pred')
# h4 is (32 x 32 x 1)
return h4
```

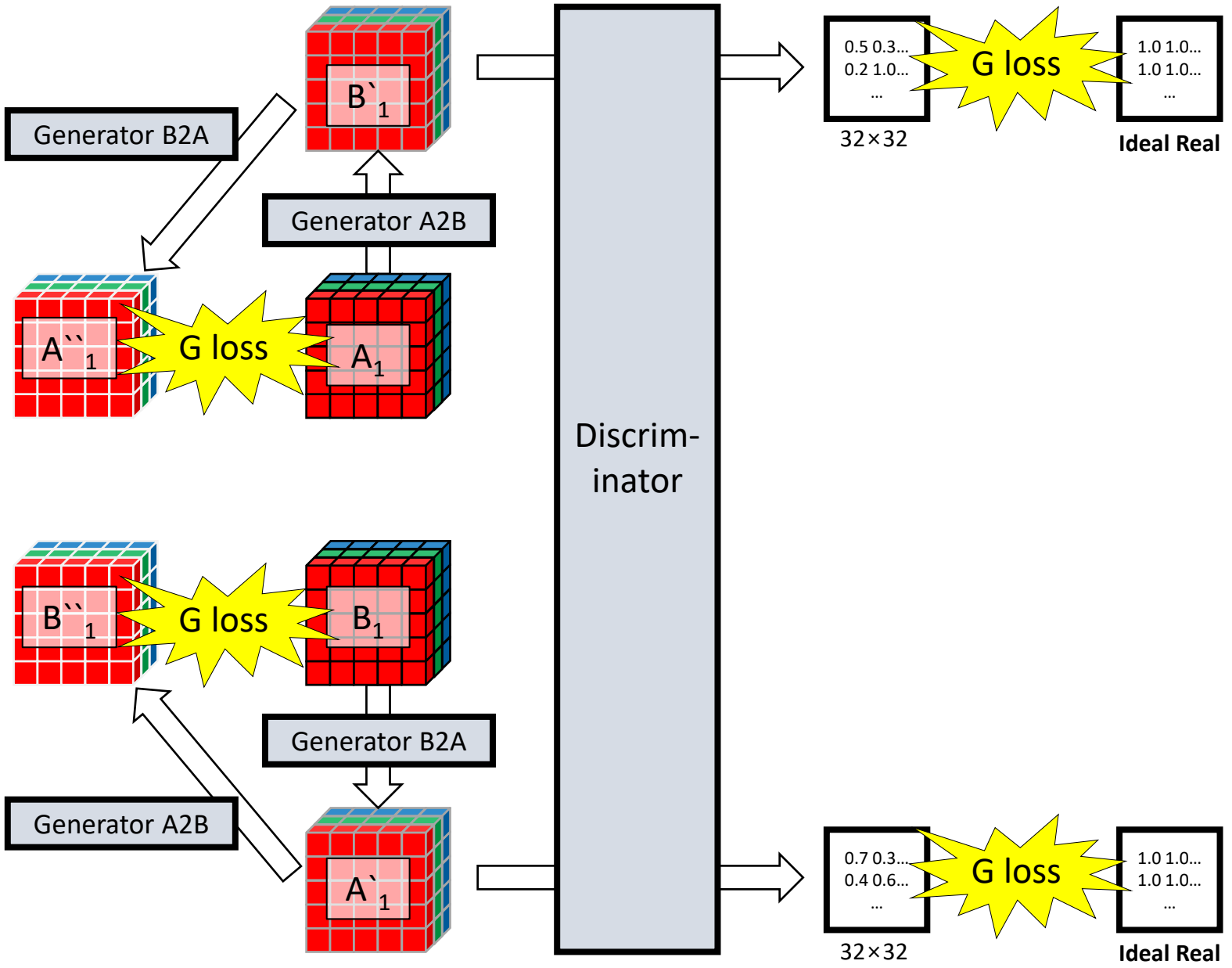
Decision [0,1]

Dec_input

모델 설명

Loss Function

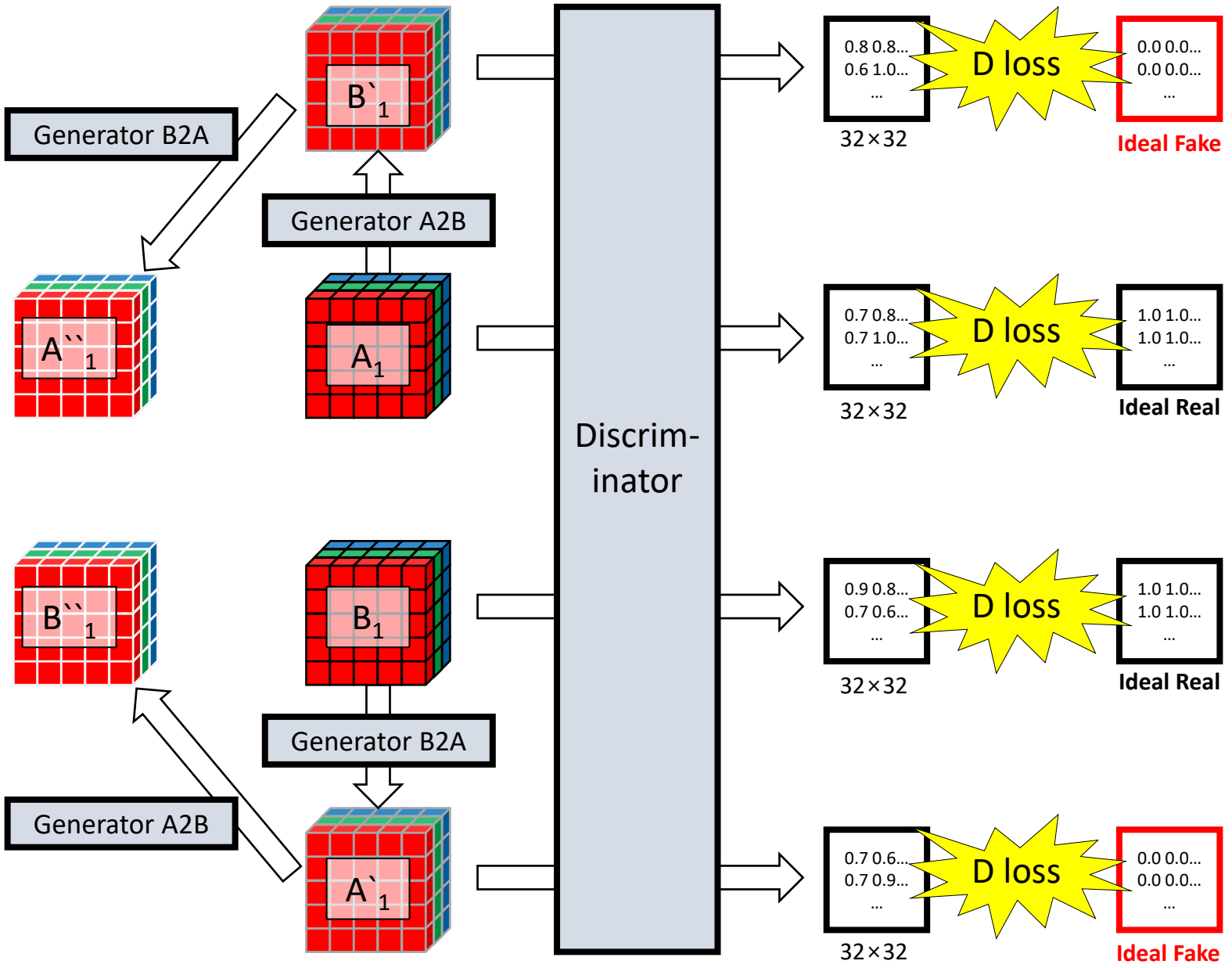
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- **Modeling**
- Train & Test
- Evaluation



모델 설명

Loss Function

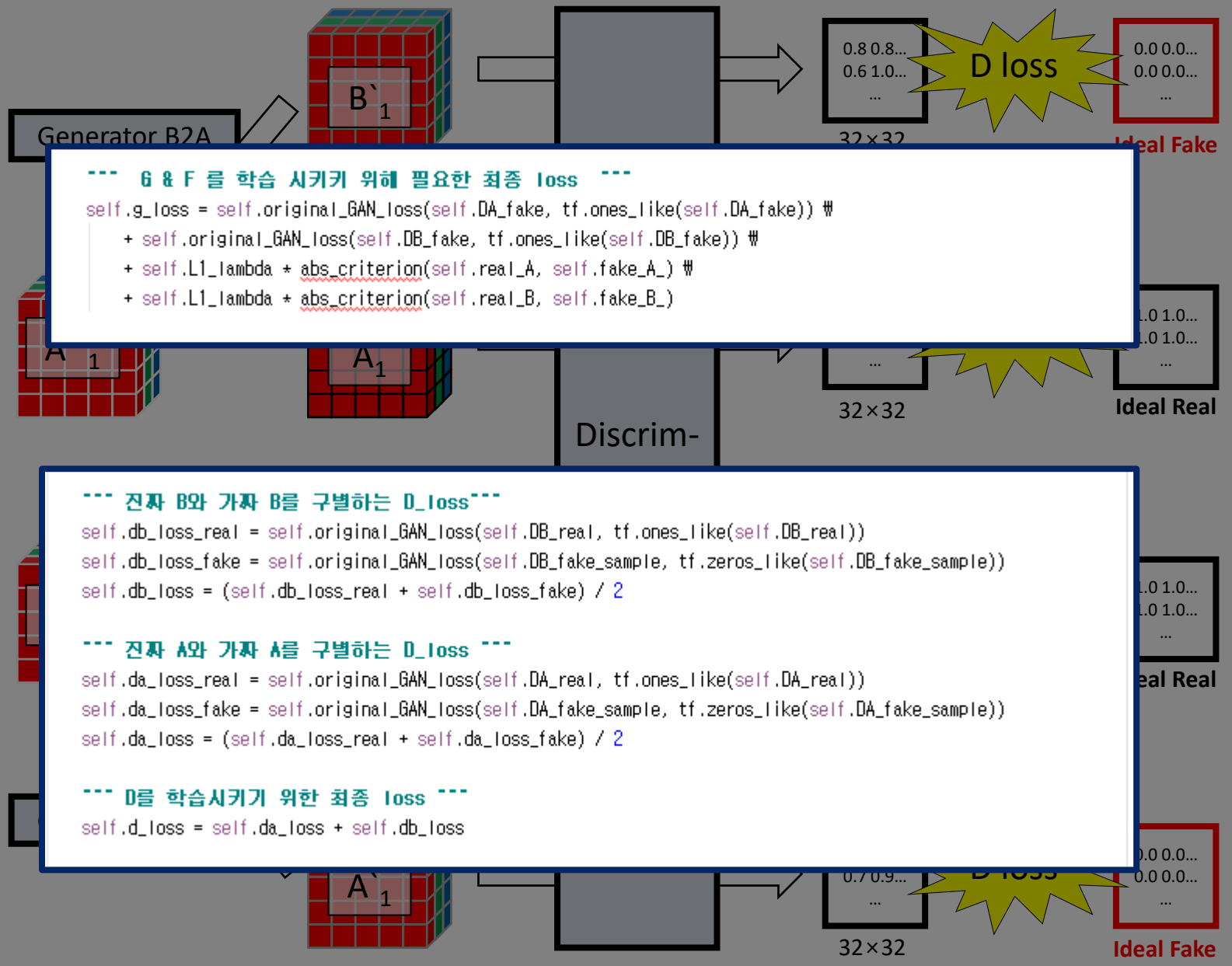
- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation



모델 설명

Loss Function

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation




```
--- G & F 를 학습 시키키 위해 필요한 최종 loss ---
self.g_loss = self.original_GAN_loss(self.DA_fake, tf.ones_like(self.DA_fake)) \
    + self.original_GAN_loss(self.DB_fake, tf.ones_like(self.DB_fake)) \
    + self.L1_lambda * abs_criterion(self.real_A, self.fake_A_) \
    + self.L1_lambda * abs_criterion(self.real_B, self.fake_B_)
```

```
--- 진짜 B와 가짜 B를 구별하는 D_loss---
self.db_loss_real = self.original_GAN_loss(self.DB_real, tf.ones_like(self.DB_real))
self.db_loss_fake = self.original_GAN_loss(self.DB_fake_sample, tf.zeros_like(self.DB_fake_sample))
self.db_loss = (self.db_loss_real + self.db_loss_fake) / 2

--- 진짜 A와 가짜 A를 구별하는 D_loss ---
self.da_loss_real = self.original_GAN_loss(self.DA_real, tf.ones_like(self.DA_real))
self.da_loss_fake = self.original_GAN_loss(self.DA_fake_sample, tf.zeros_like(self.DA_fake_sample))
self.da_loss = (self.da_loss_real + self.da_loss_fake) / 2

--- D를 학습시키기 위한 최종 loss ---
self.d_loss = self.da_loss + self.db_loss
```

모델 설명

Training and Testing

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- **Modeling**
- Train & Test
- Evaluation

Hyper Parameters

- **Batch size** = 1
- **Initial learning rate** = 0.0002
- **Epoch cycle** = 100
- **Number of convolution layer** = 58

Configuration

- **Number of parameters** = about 5200
- **The largest number of fitter** = 12
- **Learning time** = 12 hours (+ 48 hours)

모델 설명

Training and Testing

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

```
Allocator (GPU_0_bfc) ran out of memory trying to allocate 297.00MiB. The caller indicates that this is not a failu
Allocator (GPU_0_bfc) ran out of memory trying to allocate 297.00MiB. The caller indicates that this is not a failu
Allocator (GPU_0_bfc) ran out of memory trying to allocate 25.52MiB.  Current allocation summary follows.
Bin (256):   Total Chunks: 317, Chunks in use: 317. 79.2KiB allocated for chunks. 79.2KiB in use in bin. 9.1KiB cl
Bin (512):   Total Chunks: 48, Chunks in use: 48. 24.0KiB allocated for chunks. 24.0KiB in use in bin. 24.0KiB clie
Bin (1024):  Total Chunks: 332, Chunks in use: 332. 337.8KiB allocated for chunks. 337.8KiB in use in bin. 337.5KiB
Bin (2048):  Total Chunks: 33, Chunks in use: 33. 82.2KiB allocated for chunks. 82.2KiB in use in bin. 81.5KiB clie
Bin (4096):  Total Chunks: 115, Chunks in use: 113. 689.0KiB allocated for chunks. 678.0KiB in use in bin. 678.0KiB
Bin (8192):  Total Chunks: 12, Chunks in use: 12. 140.0KiB allocated for chunks. 140.0KiB in use in bin. 138.0KiB
Bin (16384):         Total Chunks: 3, Chunks in use: 3. 72.0KiB allocated for chunks. 72.0KiB in use in bin. 72.0K
Bin (32768):         Total Chunks: 22, Chunks in use: 22. 770.5KiB allocated for chunks. 770.5KiB in use in bin. 7
Bin (65536):         Total Chunks: 0, Chunks in use: 0. 0B allocated for chunks. 0B in use in bin. 0B client-reque
Bin (131072):        Total Chunks: 0, Chunks in use: 0. 0B allocated for chunks. 0B in use in bin. 0B client-reque
Bin (262144):        Total Chunks: 13, Chunks in use: 13. 3.66MiB allocated for chunks. 3.66MiB in use in bin. 3.6
Bin (524288):        Total Chunks: 7, Chunks in use: 7. 4.24MiB allocated for chunks. 4.24MiB in use in bin. 3.50M
Bin (1048576):       Total Chunks: 15, Chunks in use: 14. 18.85MiB allocated for chunks. 16.87MiB in use in bin. 1
Bin (2097152):       Total Chunks: 121, Chunks in use: 120. 274.66MiB allocated for chunks. 272.41MiB in use in bi
Bin (4194304):       Total Chunks: 11, Chunks in use: 11. 54.38MiB allocated for chunks. 54.38MiB in use in bin. 5
Bin (8388608):       Total Chunks: 17, Chunks in use: 17. 172.46MiB allocated for chunks. 172.46MiB in use in bin.
Bin (16777216):      Total Chunks: 252, Chunks in use: 251. 6.07GiB allocated for chunks. 6.04GiB in use in bin. 5
Bin (33554432):      Total Chunks: 27, Chunks in use: 27. 1.25GiB allocated for chunks. 1.25GiB in use in bin. 1.2
Bin (67108864):      Total Chunks: 23, Chunks in use: 23. 2.19GiB allocated for chunks. 2.19GiB in use in bin. 2.1
Bin (134217728):     Total Chunks: 3, Chunks in use: 3. 487.32MiB allocated for chunks. 487.32MiB in use in bin. 2
Bin (268435456):     Total Chunks: 0, Chunks in use: 0. 0B allocated for chunks. 0B in use in bin. 0B client-reque
Bin for 25.52MiB was 16.00MiB, Chunk State:
  Size: 24.00MiB | Requested Size: 24.00MiB | in_use: 0, prev:   Size: 24.00MiB | Requested Size: 24.00MiB | in_us
Chunk at 0x703f20000 of size 256
Chunk at 0x703f20100 of size 256
```

모델 테스트

Generation 테스트 결과

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

20s to 50s Generation

Input	Output

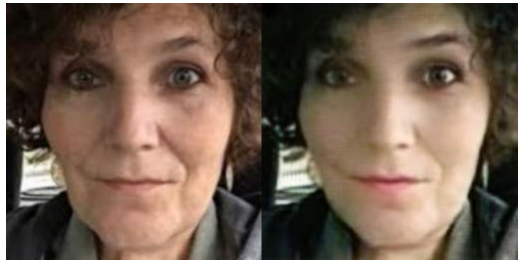
모델 테스트

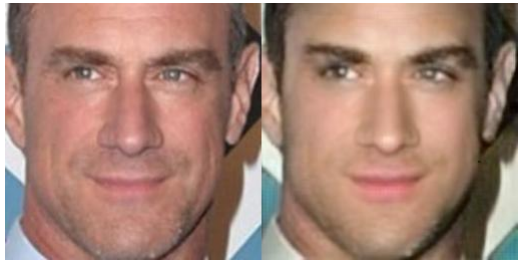
Generation 테스트 결과

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

50s to 20s Generation

Input	Output





모델 개선

Cycle Consistency 테스트 결과

- Face Aging
- Cycle GAN
- Data Collection
- Google Colab
- Modeling
- Train & Test
- Evaluation

20s

Original

Cyclic

50s

Original

Cyclic

End of Document