PFA-GAN: Progressive Face Aging with Generative Adversarial Network

Zhizhong Huang, Shouzhen Chen, Junping Zhang, Hongming Shan

IEEE Transactions on Information Forensics and Security, vol. 16, pp. 2031-2045, 2021, doi: 10.1109/TIFS.2020.3047753.

Contents

Contents

Background

- Generative model in Computer Vision

  GAN, Diffusion, etc.

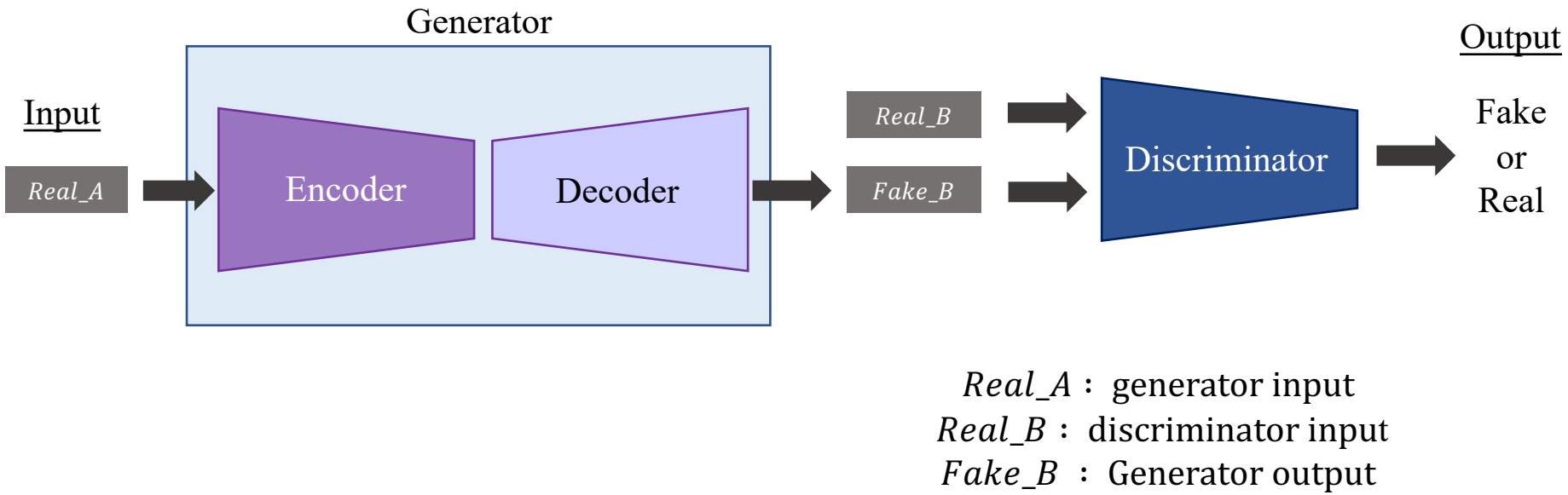


Figure 1. GANs training

Background

- Applications
  - Face aging apps

FaceAPP

- Face recognition
  generate 3D face model
  (GANは使っていないと思う)
  ↓
  face recognition

Face ID
https://support.apple.com/ja-jp/HT208109

Background

- Face aging

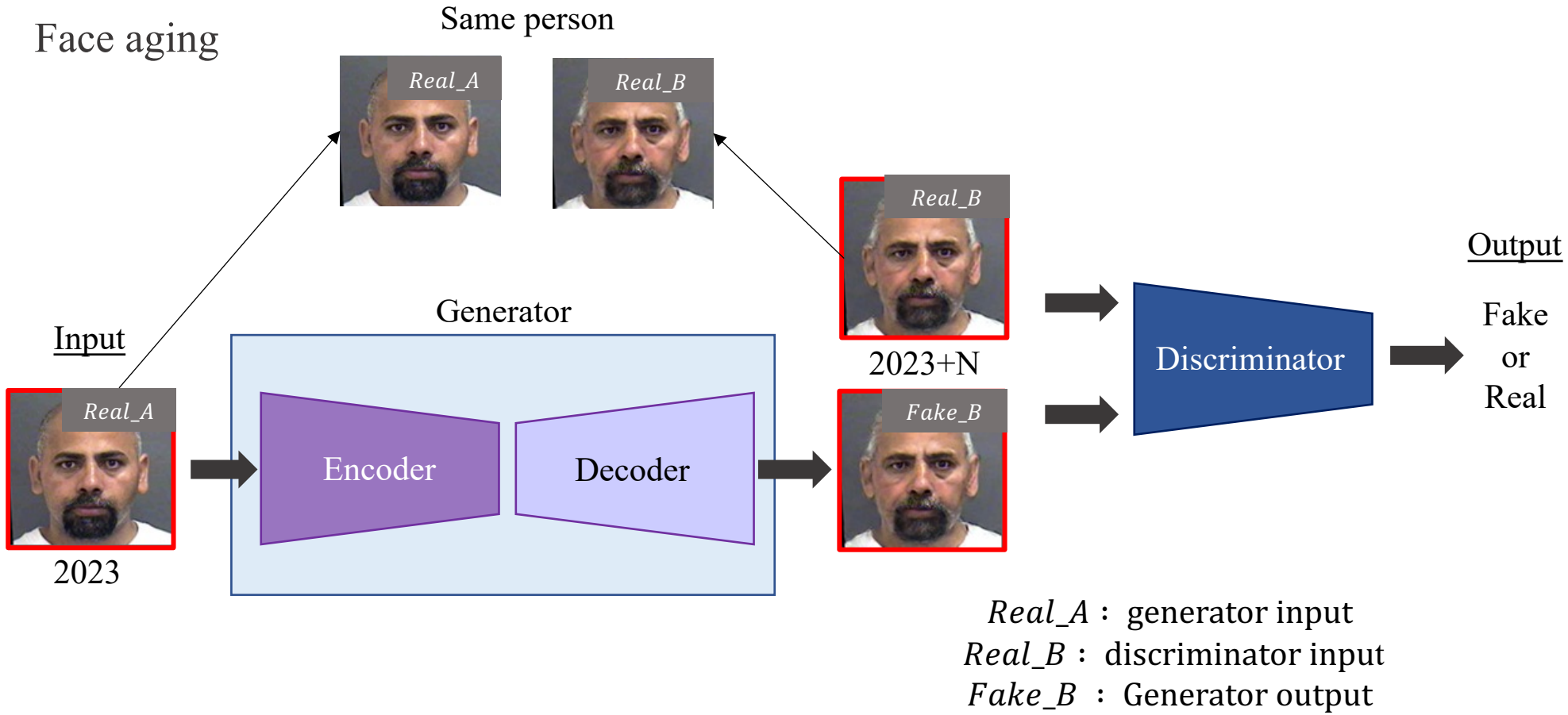


Figure 1. GANs training

Background

- Face aging

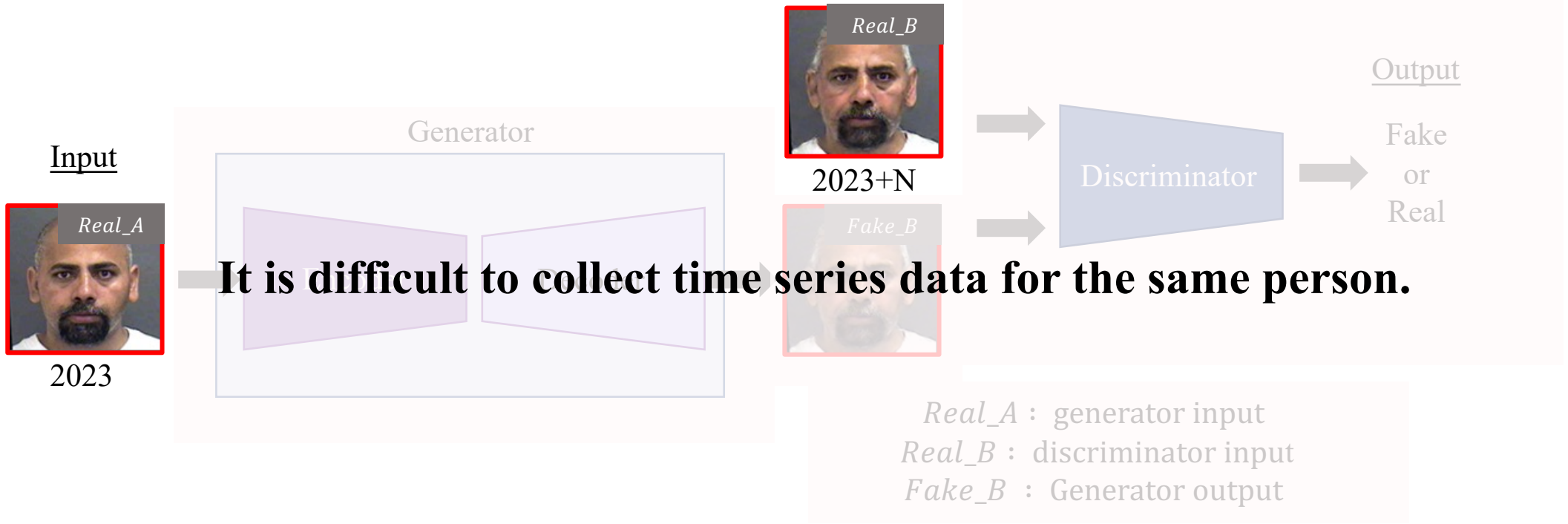


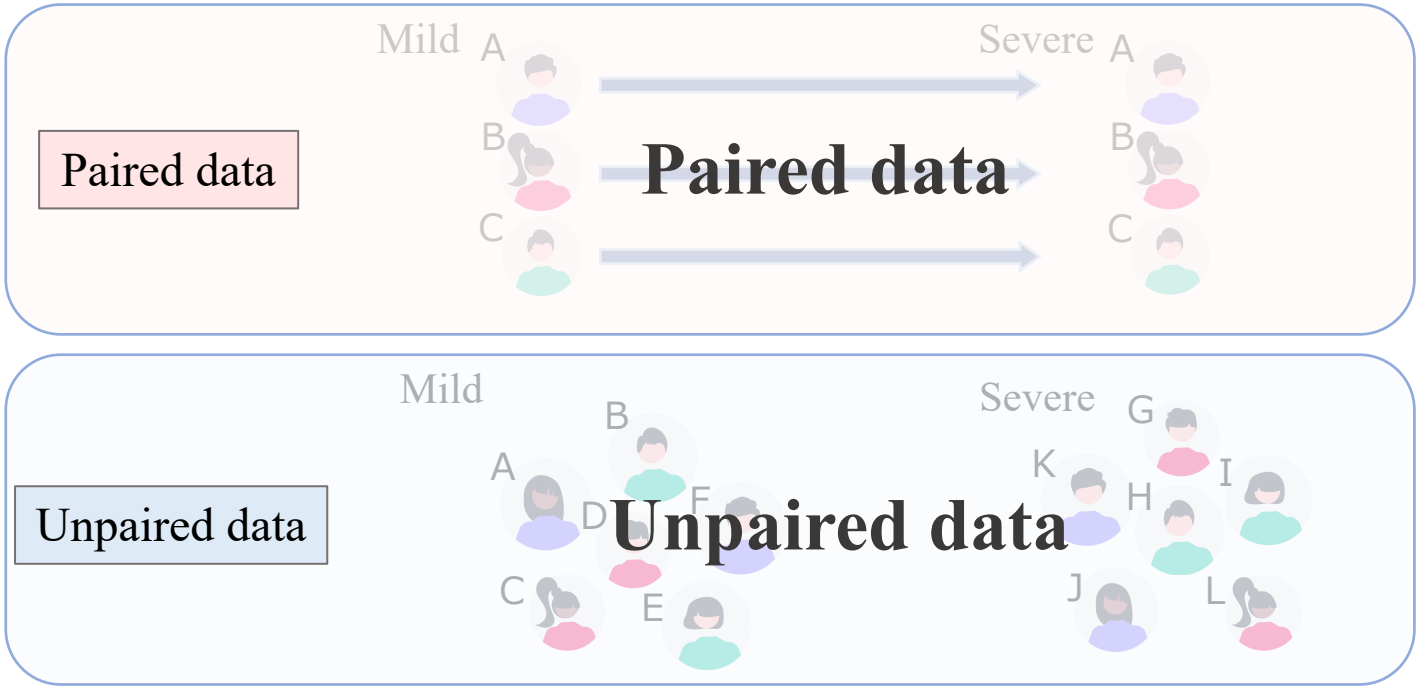
Figure 1. GANs training

Contents

Paired, Unpaired data

Paired (longitudinal)： Time series data collected from the same patient

Unpaired (cross-sectional)： Time series data collected from the different patient



・Prediction model with Paired data
Requires years to decades of time to collect

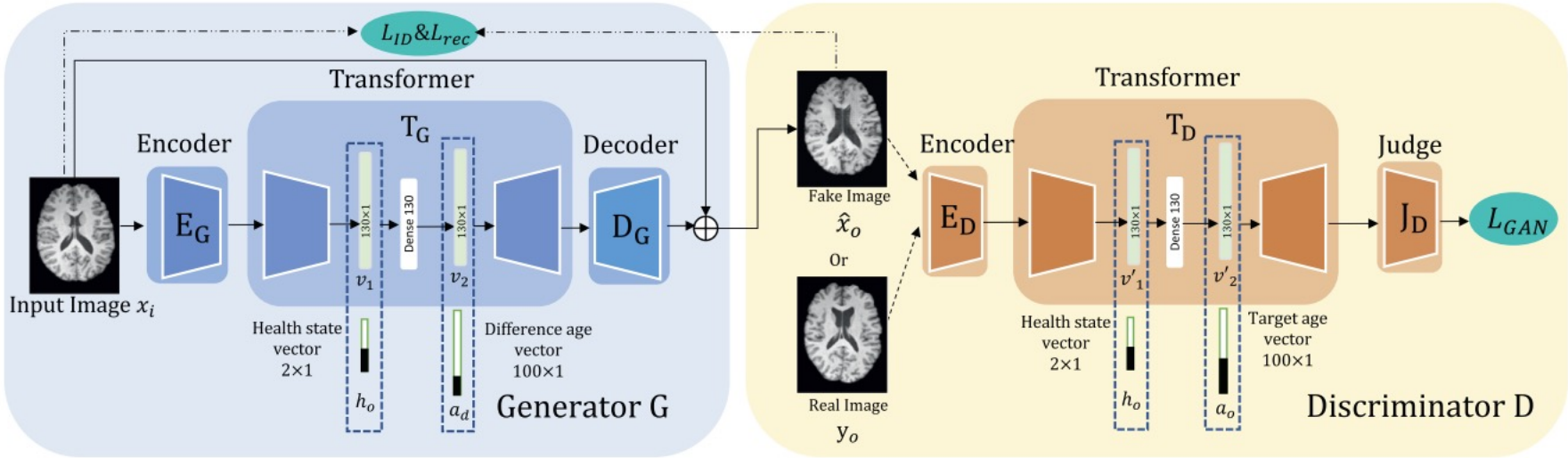
・Prediction model with Unpaired data
Relatively easy to collect than Paired data

Contents

Prediction of aging and disease changes in brain MRI images (cross-sectional data)

- Because it is difficult to collect time series data (longitudinal data) of brain images of the same individual, we attempted to simulate the aging of an individual brain using cross-sectional data without relying on longitudinal data.

- Using GAN, brain images at a certain point in time, the year and health condition to be predicted can be input as conditions, and brain images corresponding to the conditions can be generated.



GAN to predict brain aging [*]

[*] Tian Xia, et al. Medical Image Analysis, 2021.

Prediction of aging and disease changes in brain MRI images (cross-sectional data)

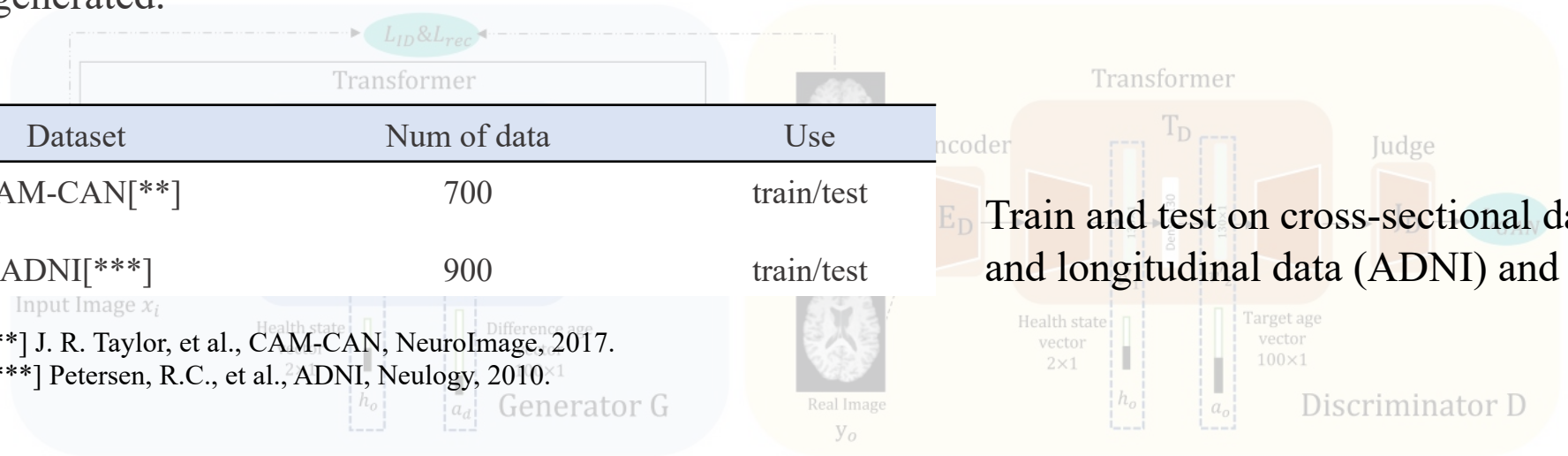
- Because it is difficult to collect time series data (longitudinal data) of brain images of the same individual, we attempted to simulate the aging of an individual brain using cross-sectional data without relying on longitudinal data.

- Using GAN, brain images at a certain point in time, the year and health condition to be predicted can be input as conditions, and brain images corresponding to the conditions can be generated.

Dataset	Num of data	Use
CAM-CAN[**]	700	train/test
ADNI[***]	900	train/test

[**] J. R. Taylor, et al., CAM-CAN, NeuroImage, 2017.
[***] Petersen, R.C., et al., ADNI, Neulogy, 2010.

Train and test on cross-sectional data (CAM-CAN) and longitudinal data (ADNI) and also evaluate



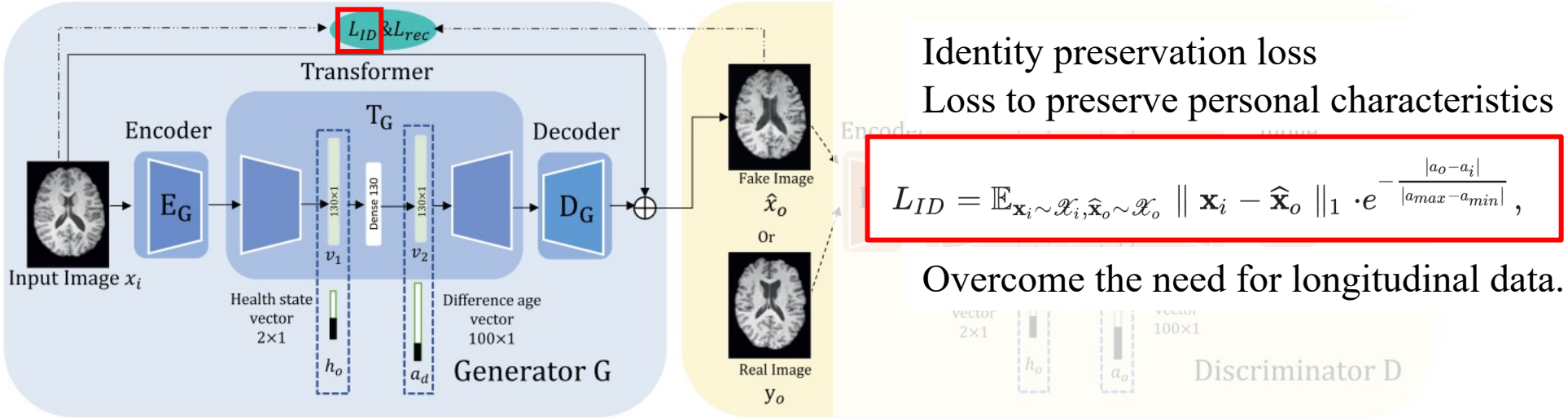
GAN to predict brain aging [*]

[*] Tian Xia, et al. Medical Image Analysis, 2021.

Prediction of aging and disease changes in brain MRI images (cross-sectional data)

- Because it is difficult to collect time series data (longitudinal data) of brain images of the same individual, we attempted to simulate the aging of an individual brain using cross-sectional data without relying on longitudinal data.

- Using GAN, brain images at a certain point in time, the year and health condition to be predicted can be input as conditions, and brain images corresponding to the conditions can be generated.



GAN to predict brain aging [*]

[*] Tian Xia, et al. Medical Image Analysis, 2021.

Prediction of aging and disease changes in brain MRI images (cross-sectional data)

- Slightly better performance was obtained when training on cross-sectional data than when training on longitudinal data
  - ➡ Longitudinal data was trained with 98 participants, while cross-sectional data was trained with 786 participants (taking advantage of the strength of the cross-sectional data)
- However, only a slight difference was observed even when the number of data was increased by a factor of about 8, so it is of course better to have a large amount of longitudinal data

Table 7. Quantitative results of a longitudinal benchmark and our method.

		SSIM	PSNR	MSE
Longitudinal data	Longitudinal	$0.72_{\pm 0.09}$	$24.2_{\pm 3.0}$	$0.076_{\pm 0.013}$
Cross sectional data	Ours	$0.79_{\pm 0.08}$	$26.1_{\pm 2.6}$	$0.042_{\pm 0.006}$

Contents

Purpose

Using unpaired data to achieve highly accurate human face aging prediction with GANs.

Contents

Dataset

Datast	Number of data	Use
MORPH[*]	55,124	Training
CACD[**]	163,446 (2,000people)	Training
FG-NET[***]	1,002 (82people)	Testing

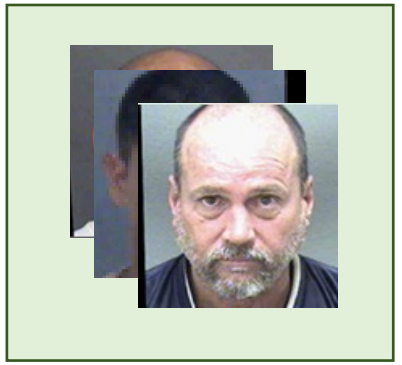
[*] K. Ricanek, et al., MORPH, FGR, 2006.
[**] B. Chen, et al., CACD, IEEE Trans. Multimedia, 2015.
[***] A. Lanitis, et al., FG-NET, IEEE Trans. Pattern Anal, 2002

Divide the ages in the Dataset into **four groups.**

Age 30-

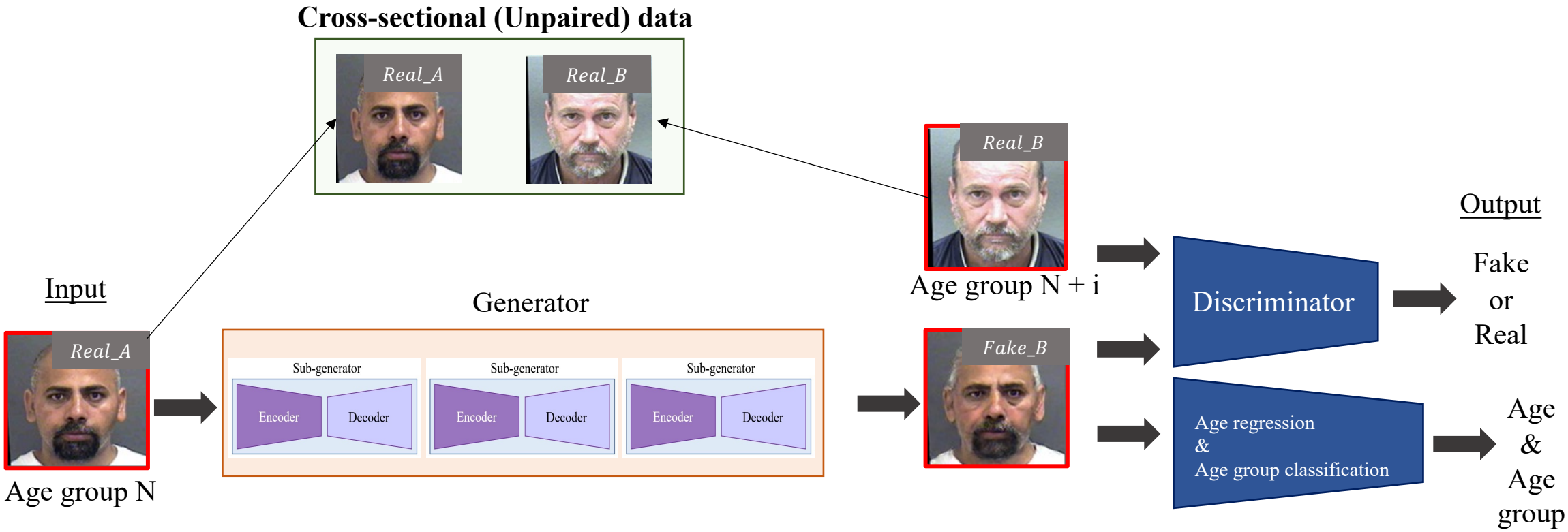
Age 31-40

Age 41-50



Age 51+

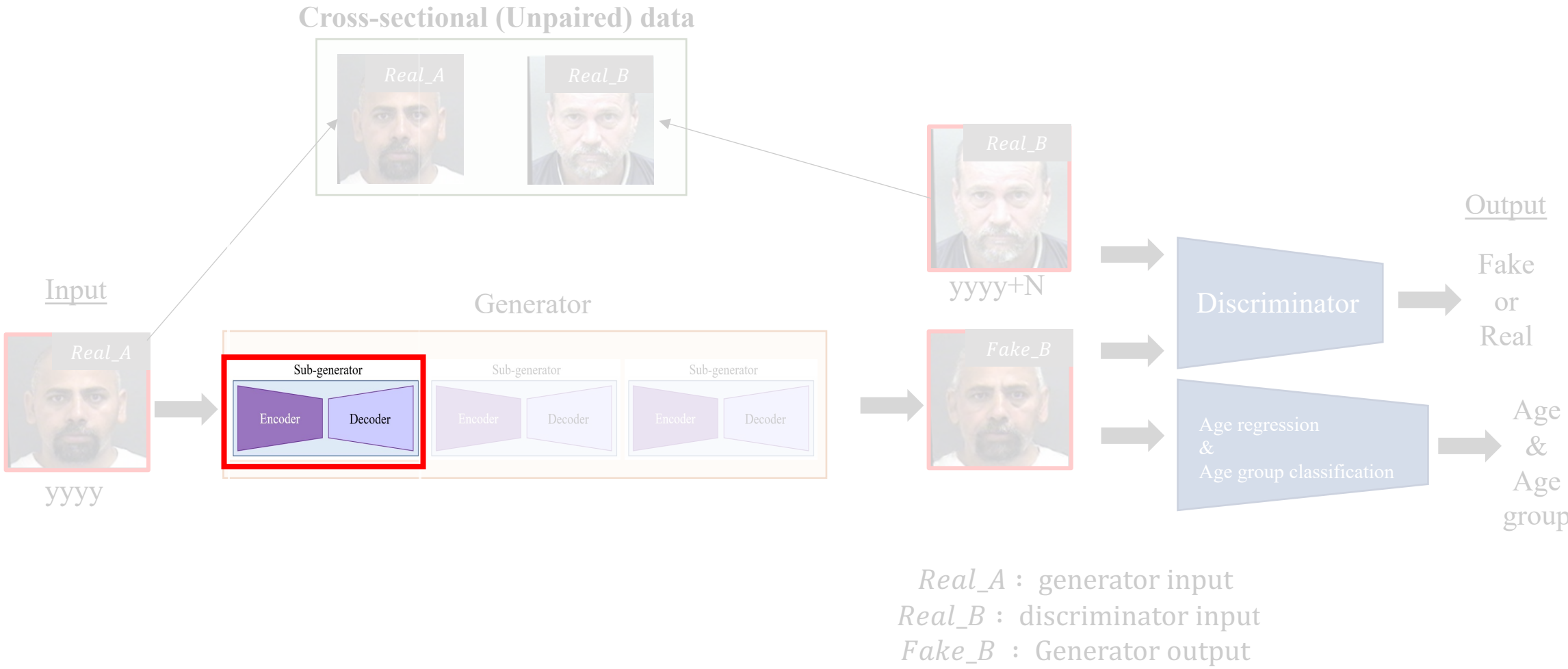
PFA-GAN



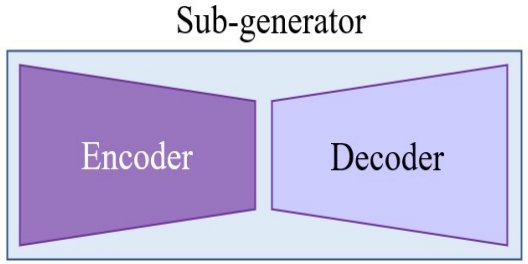
While conventional cGAN-based methods generate face images by one-network, and learning each age group sequentially, but the proposed method generates face images by several sub-network, and learning each age group in an end-to-end manner.

$Real\_A$: generator input
$Real\_B$: discriminator input
$Fake\_B$: Generator output

Sub-generator



Sub-generator




```
(SubG): ModuleList(
  (0): SubGenerator(
    (Encoder): Encoder(
      (conv1): Conv2d(1, 32, kernel_size=(9, 9), stride=(1, 1), padding=(4, 4))
      (norm1): InstanceNorm2d(32, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
      (act1): LeakyReLU(negative_slope=0.2, inplace=True)
      (conv2): Conv2d(32, 64, kernel_siz                    =(2, 2), padding=(1, 1))
      (norm2): InstanceNorm2d(64, eps=1e                    1, affine=False, track_running_stats=False)
      (act2): LeakyReLU(negative_slope=0                    )
      (conv3): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (norm3): InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
      (act3): LeakyReLU(negative_slope=0.2, inplace=True)
    )
    (ResidualBlock): ResidualBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (norm1): InstanceNorm2d(128                                   ine=False, track_running_stats=False)
      (act1): LeakyReLU(negative_
      (conv2): Conv2d(128, 128, I                                   ), padding=(1, 1))
      (norm2): InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
    )
    (Decoder): Decoder(
      (deconv1): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (norm1): InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
      (act1): LeakyReLU(negative_slope=0                    )
      (deconv2): ConvTranspose2d(64, 32,                    4), stride=(2, 2), padding=(1, 1))
      (norm2): InstanceNorm2d(32, eps=1e                    1, affine=False, track_running_stats=False)
      (act2): LeakyReLU(negative_slope=0.2, inplace=True)
      (conv3): Conv2d(32, 1, kernel_size=(9, 9), stride=(1, 1), padding=(4, 4))
    )
  )
```

Encoder

Residual Block x 4

Decoder

* Slightly simplified

Sub-generator

Number of sub-generator

```
_aging_effects = []
for i in range(self.year_group - 1):
    aging_effects = self.SubG[i](x)
    aging_effects.append(aging_effects)
    x = x + aging_effects * binary_gate_vector[:, i]
return x, _aging_effects
```

X_{in}

+

=

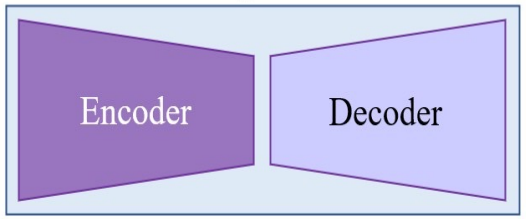
X_{out}

age *group i* + 1

X_{in}

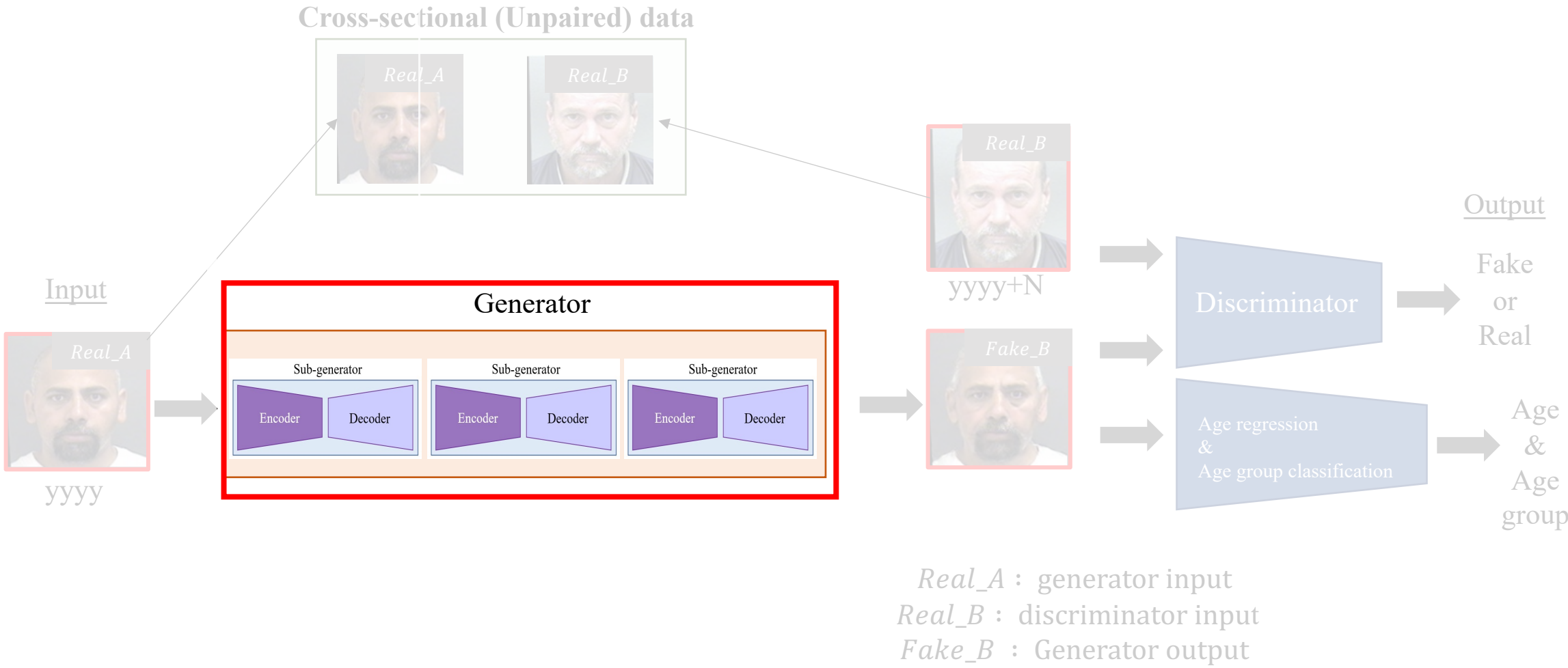
age *group i*

Sub-generator



Aging effects

Generator


Cross-sectional (Unpaired) data
Real_A
Real_B
Input
Real_A
yyyy
Generator
Sub-generator
Encoder
Decoder
Sub-generator
Encoder
Decoder
Sub-generator
Encoder
Decoder
Real_B
yyyy+N
Fake_B
Discriminator
Output
Fake
or
Real
Age regression
&
Age group classification
Age
&
Age
group
Real_A : generator input
Real_B : discriminator input
Fake_B : Generator output

Generator

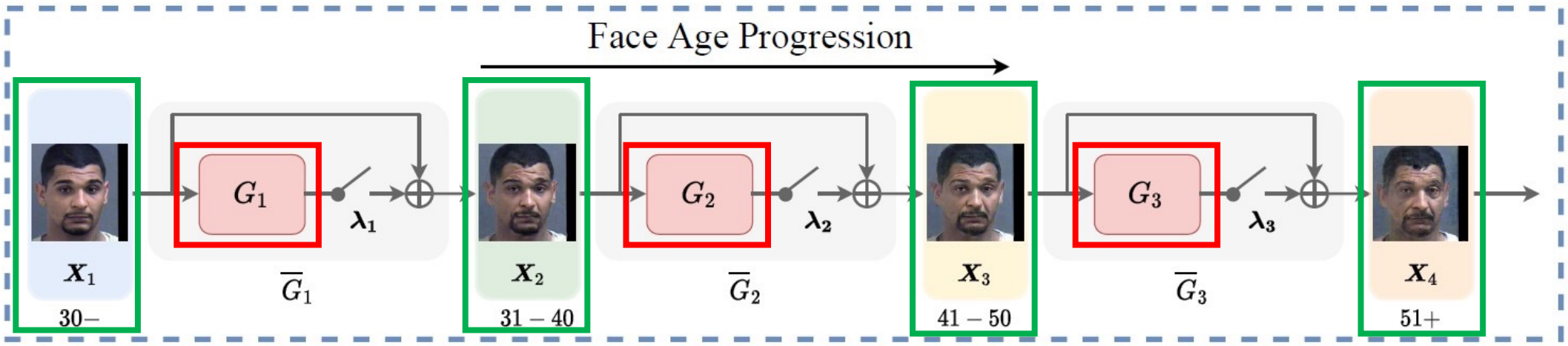


Fig. 2: The proposed progressive face aging framework for a face aging task with 4 age groups. Each sub-network $\overline{G}_i$ aims at aging faces from age group i to $i+1$, which consists of a residual skip connection, a binary gate λ_i, and a light sub-network G_i outputting aging effects. The residual skip connection from input to output can prevent the sub-network from memorizing the exact copy of the input face. The binary gate λ_i can control the aging flow and determine if the aging mapping needs the sub-network G_i to be involved.

Sub-generator × 3

Age group × 4

Generator

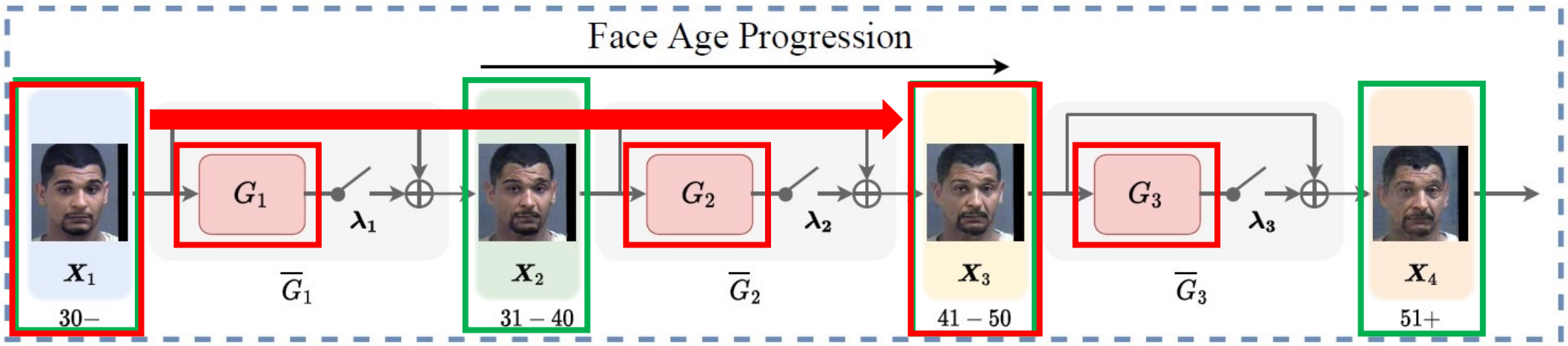


Fig. 2: The proposed progressive face aging framework for a face aging task with 4 age groups. Each sub-network $\overline{G}_i$ aims at aging faces from age group i to $i+1$, which consists of a residual skip connection, a binary gate λ_i, and a light sub-network G_i outputting aging effects. The residual skip connection from input to output can prevent the sub-network from memorizing the exact copy of the input face. The binary gate λ_i can control the aging flow and determine if the aging mapping needs the sub-network G_i to be involved.

Sub-generator × 3

Age group × 4

Generator

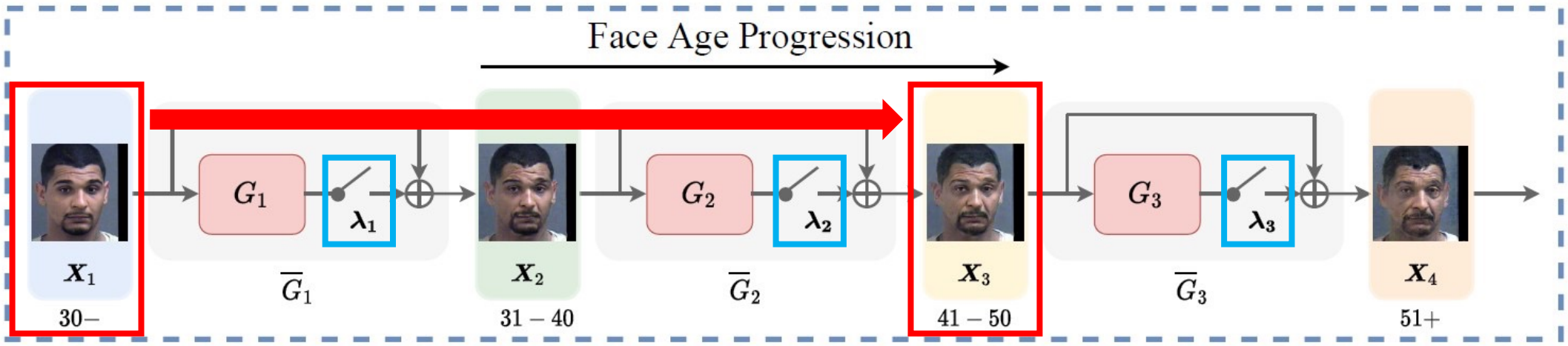


Fig. 2: The proposed progressive face aging framework for a face aging task with 4 age groups. Each sub-network $\overline{G}_i$ aims at aging faces from age group i to $i+1$, which consists of a residual skip connection, a binary gate λ_i, and a light sub-network G_i outputting aging effects. The residual skip connection from input to output can prevent the sub-network from memorizing the exact copy of the input face. The binary gate λ_i can control the aging flow and determine if the aging mapping needs the sub-network G_i to be involved.

Binary gate × 3

```
binary_gate_vector = torch.zeros((input.size(0), year_group - 1, 1, 1, 1)).to(input)
for i in range(input.size(0)):
    print(i, input[i], target[i])
    binary_gate_vector[i, input[i]: target[i], ...] = 1
```

Age group X_1 to X_3

Imaging-based Computational Biomedicine

Generator

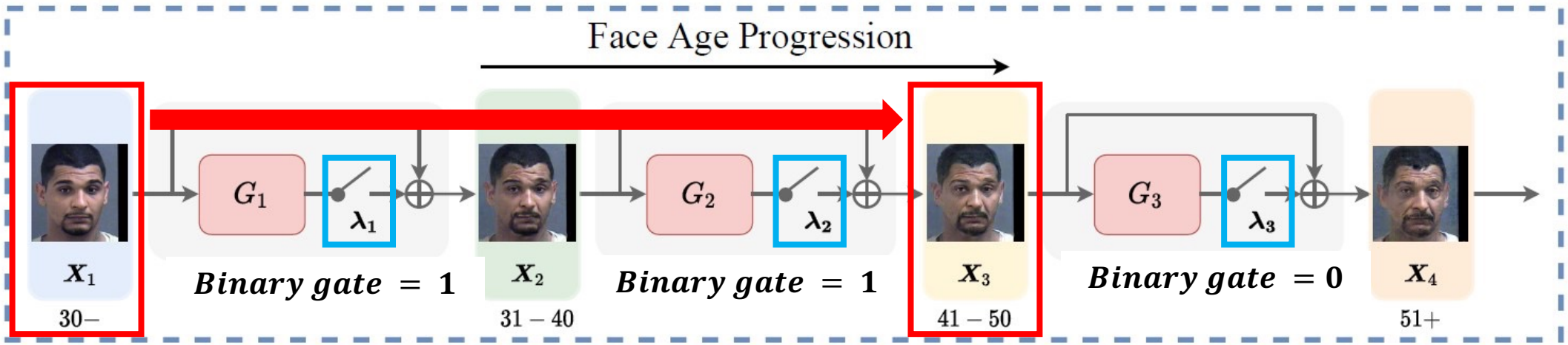


Fig. 2: The proposed progressive face aging framework for a face aging task with 4 age groups. Each sub-network $\overline{G}_i$ aims at aging faces from age group i to $i+1$, which consists of a residual skip connection, a binary gate λ_i, and a light sub-network G_i outputting aging effects. The residual skip connection from input to output can prevent the sub-network from memorizing the exact copy of the input face. The binary gate λ_i can control the aging flow and determine if the aging mapping needs the sub-network G_i to be involved.

$$\textbf{Binary gate vector} = [1, 1, 0]$$

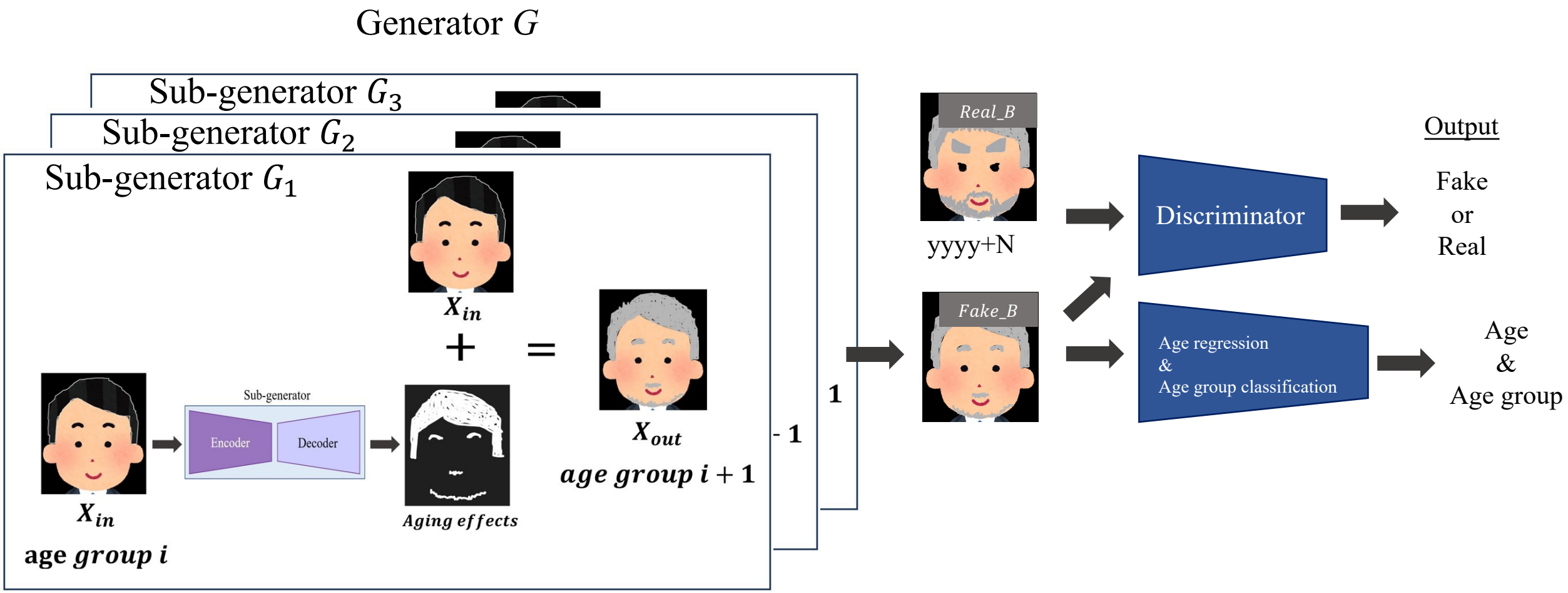
$$X_3 = \lambda_1 G_1(X_1) + \lambda_2 G_2(X_2)$$

$$\underline{X_3 = G_1(X_1) + G_2(X_2)}$$

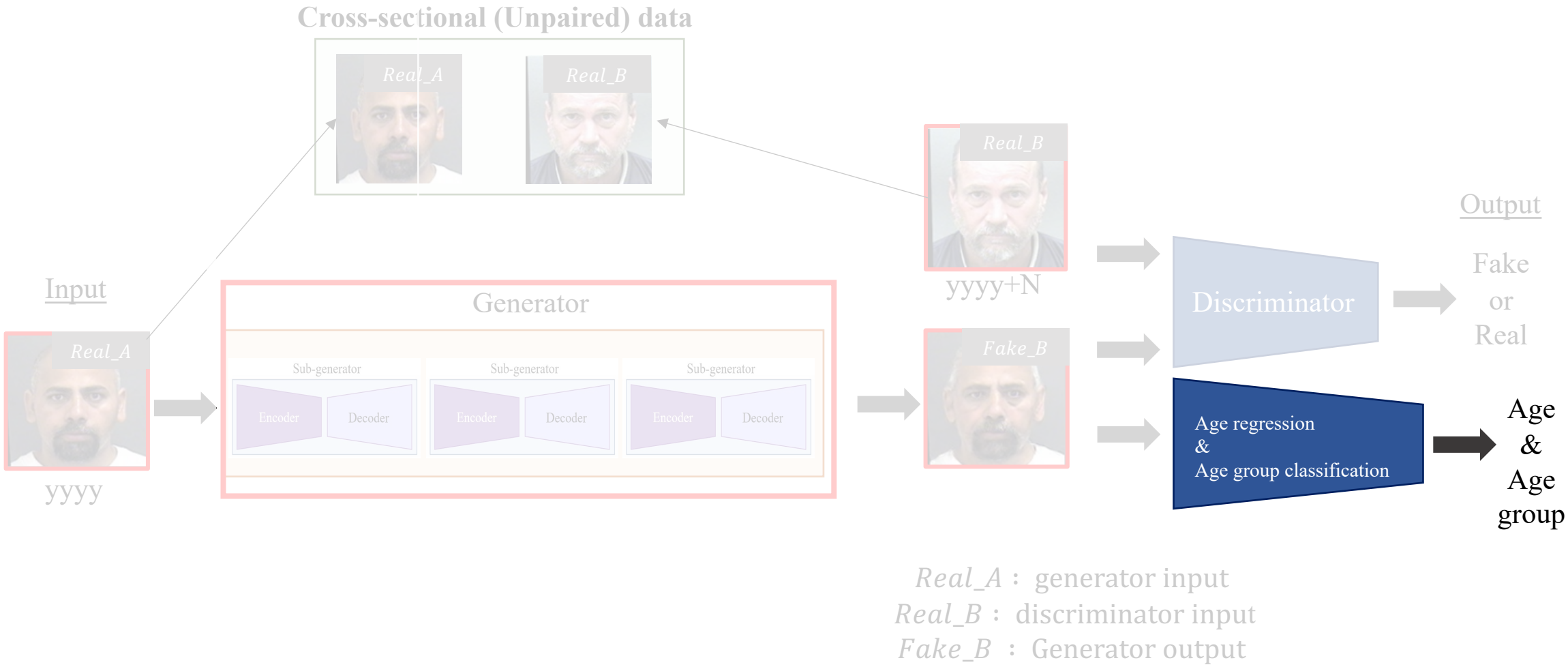
1. Larger gaps between age groups are likely to generate ghost-like artifacts due to the inability to deal with rapid aging.

2. The intermediate generated faces cannot be seen, the same person's features may be lost to a large extent.

Generator

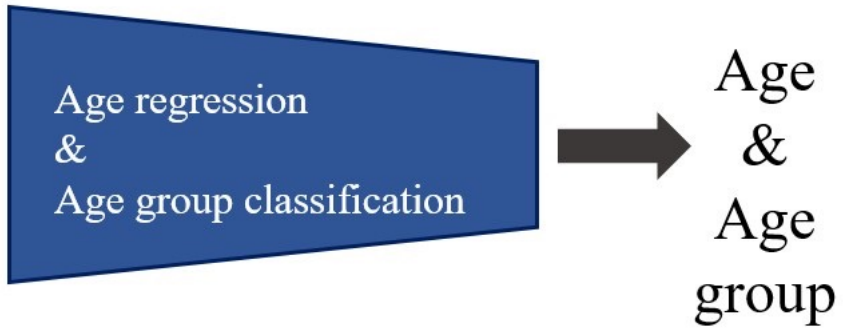

Generator G
Sub-generator G_3
Sub-generator G_2
Sub-generator G_1
X_{in}
+
=
X_{out}
Sub-generator
Encoder
Decoder
X_{in}
age group i
Aging effects
age group $i+1$
- 1
1
Real_B
yyyy+N
Fake_B
Discriminator
Age regression
&
Age group classification
Output
Fake
or
Real
Age
&
Age group

Regression, Classification head



Regression, Classification head

To better characterize the face age distribution for an improved aging accuracy.



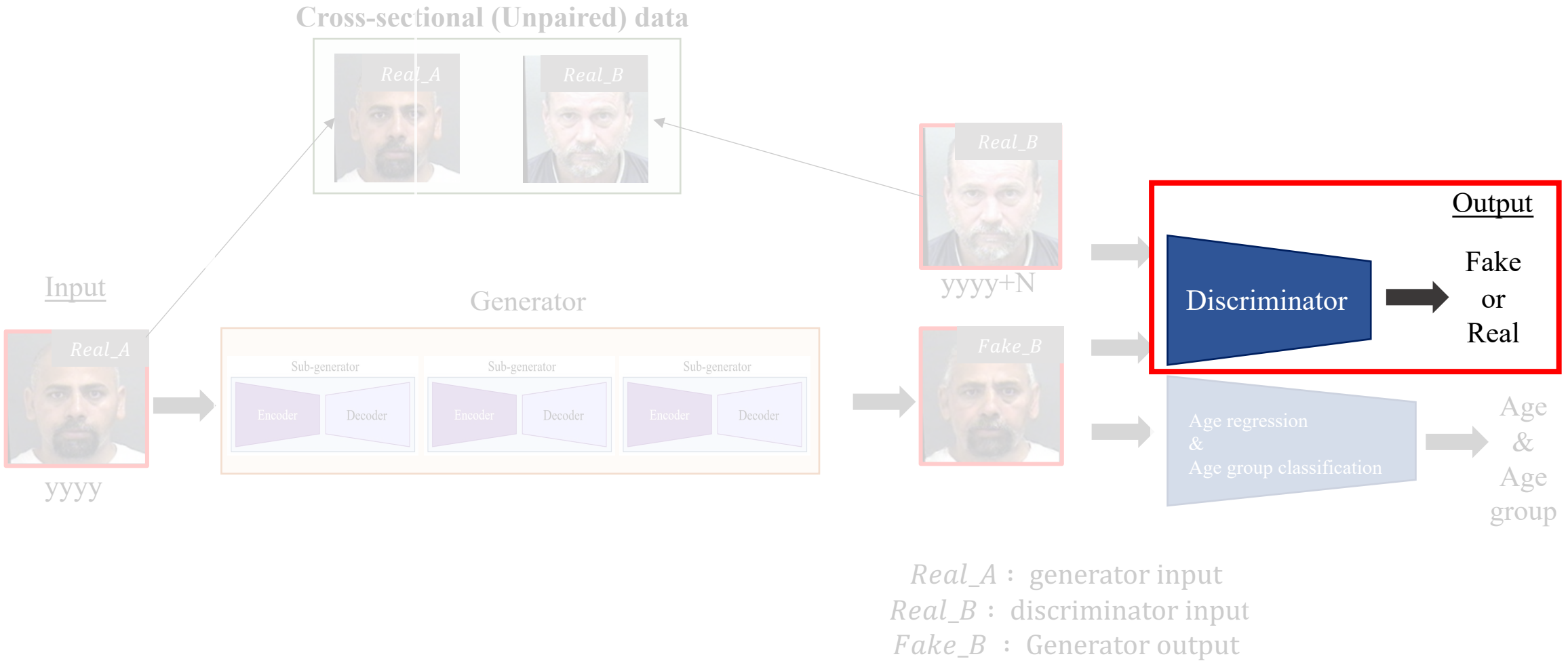

```python
class AuxiliaryAgeClassifier(nn.Module):

    def __init__(self, age_group, conv_dim=64, repeat_num=3):
        super(AuxiliaryAgeClassifier, self).__init__()
        age_classifier = [
            nn.Conv2d(3, conv_dim, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(conv_dim),
            nn.ReLU(True),
        ]
        nf_mult = 1
        for n in range(1, repeat_num + 2):
            nf_mult_prev = nf_mult
            nf_mult = min(2 ** n, 8)
            age_classifier += [
                nn.Conv2d(conv_dim * nf_mult_prev,
                          conv_dim * nf_mult, kernel_s
                nn.BatchNorm2d(conv_dim * nf_mult),
                nn.ReLU(True),
            ]
        age_classifier += [
            nn.Flatten(),
            nn.Linear(conv_dim * nf_mult * 16, 101),
        ]
        self.age_classifier = nn.Sequential(*age_classifier)
        self.group_classifier = nn.Linear(101, age_group)

    def forward(self, inputs):
        age_logit = self.age_classifier(F.hardtanh(inputs))
        group_logit = self.group_classifier(age_logit)
        return age_logit, group_logit
```

Basic CNN

Although the authors describe it as a regression, it is actually a 101-class classification.

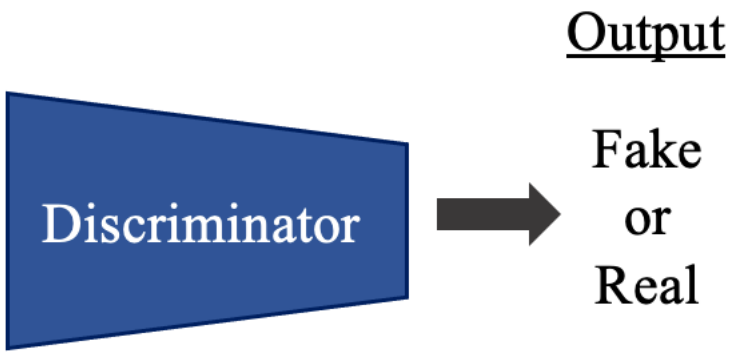
Output two logits

Discriminator



$Real\_A$: generator input
$Real\_B$: discriminator input
$Fake\_B$: Generator output

Discriminator




```
Discriminator(
  (conv1): Conv2d(1, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (act1): LeakyReLU(negative_slope=0.2, inplace=True)
  (convblock1): Conv2d(68, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (act2): LeakyReLU(negative_slope=0.2, inplace=True)
  (convblock2): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (act3): LeakyReLU(negative_slope=0.2, inplace=True)
  (convblock3): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (act4): LeakyReLU(negative_slope=0.2, inplace=True)
  (convblock4): Conv2d(512, 512, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (act5): LeakyReLU(negative_slope=0.2, inplace=True)
  (conv2): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
)
```

1. Encode age groups

```
Discriminator(
  (conv1): Conv2d(1, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (act1): LeakyReLU(negative_slope=0.2, inplace=True)
  (convblock1): Conv2d(68, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
```

```
def forward(self, x, condition):
    x = self.conv1(x) # 1x256x256 -> 64x128x128
    x = self.act1(x)

    condition = group2feature(
        condition,
        feature_size=x.size(2),
        year_group=self.year_group
    ).to(x)

    x = torch.cat([x, condition], dim=1)

    x = self.convblock1(x) # (64+year_group)x128
```

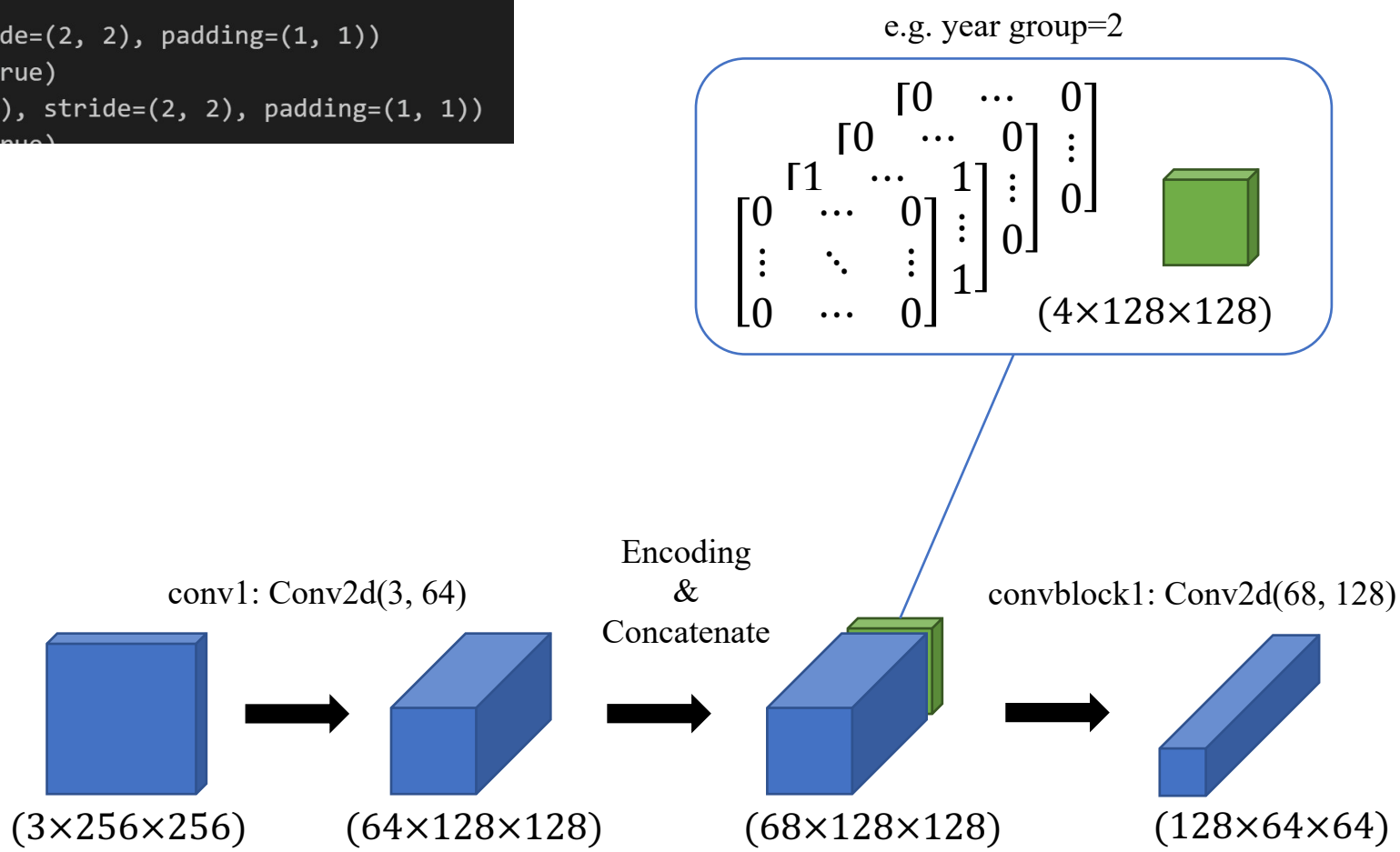
Encoding function

```
def group2onehot(
        condition : torch.Tensor,
        age_group : int
    ) -> torch.Tensor:
    """
    return : onehot_vector
                torch.Size([batch_size, year_group])
    """
    onehot_vector = torch.eye(age_group)[condition.long().squeeze()]
    if len(onehot_vector.size()) > 1:
        return onehot_vector
    return onehot_vector.unsqueeze(0)

def group2feature(
        group,
        year_group,
        feature_size
    ) -> torch.Tensor:
    """
    return : onehot padded vector
                torch.Size([batch_size, year_group, feature_size, feature_size])
    """
    onehot = group2onehot(group, year_group)
    onehot_padded_vector = onehot.unsqueeze(-1).unsqueeze(-1).repeat(1, 1, feature_size, feature_size)
    return onehot_padded_vector
```

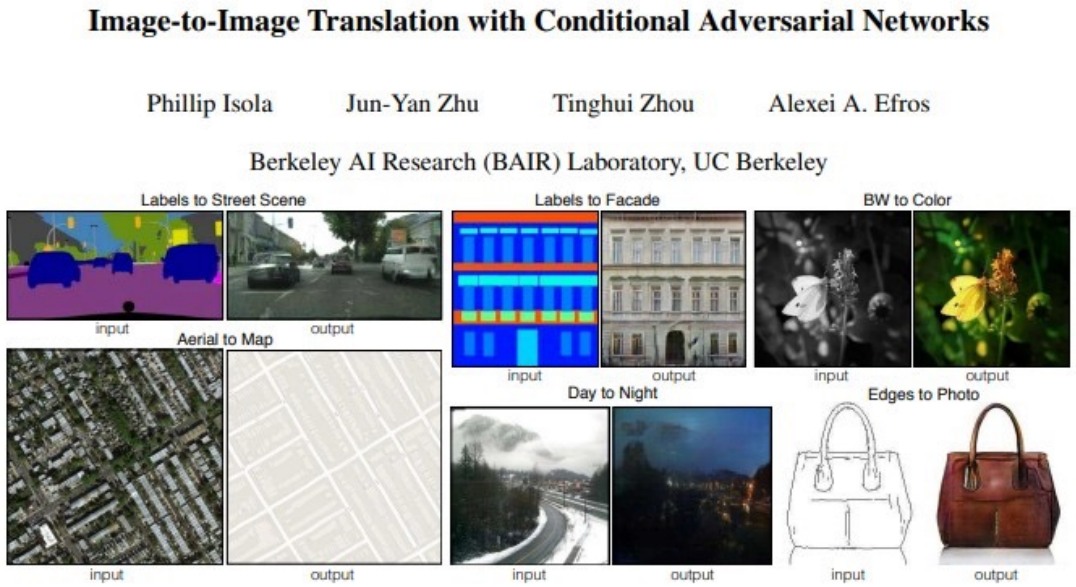
1. Encode age groups

```
Discriminator(
  (conv1): Conv2d(1, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (act1): LeakyReLU(negative_slope=0.2, inplace=True)
  (convblock1): Conv2d(68, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
```

```
def forward(self, x, condition):
    x = self.conv1(x) # 1x256x256 -> 64x128x128
    x = self.act1(x)

    condition = group2feature(
        condition,
        feature_size=x.size(2),
        year_group=self.year_group
    ).to(x)

    x = torch.cat([x, condition], dim=1)

    x = self.convblock1(x) # (64+year_group)x128
```



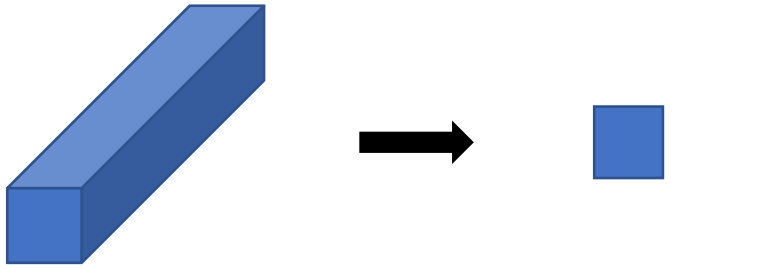
2. Patch GAN



[*] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A.Efros., Image-toImage Translation with Conditional Adversarial Networks, CVPR, 2017.

```
(conv2): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
```

conv2: Conv2d(512, 1)



(512×15×15) (1×14×14)

it is equivalent to **"make the final output a feature map with a certain size and make a true/false judgment at each pixel"** and to **"make the input image into patches and make a true/false judgment at the output of each patch"**.

*** These network architectures can be found in the appendix, page 18.**

Loss function

1. **Adversarial Loss**
   Least-Squares
2. **Age Estimation Loss**
   MSE Loss, Cross-entropy Loss
3. **Identity Consistency Loss**
   L1 Loss, SSIM Loss, Feature Loss
4. **Final Loss**

1. Adversarial Loss

$$\mathcal{L}_{\text{adv}} = \frac{1}{2}\mathbb{E}_{\boldsymbol{X}_s}\left[D\left([G(\boldsymbol{X}_s, \boldsymbol{\lambda}_{s:t}); \boldsymbol{C}_t]\right) - 1\right]^2$$

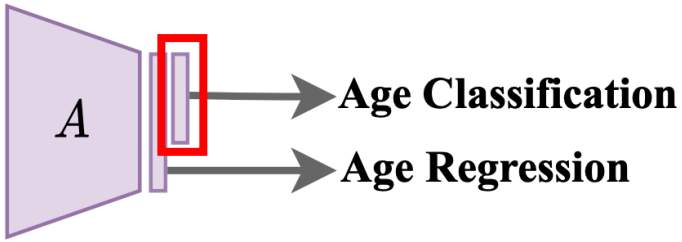
X_s: Young face from age group s

$\lambda_{s:t}$: Binary gate vector

C_t: Age group vector

```
class G_Loss(object):
    def adv_loss(g_logit):
        return 0.5*torch.mean((g_logit - 1.) ** 2)
```

2. Age Estimation Loss



$$\mathcal{L}_{\text{age}} = \mathbb{E}_{\boldsymbol{X}_s}\left[\left\|\boldsymbol{y} - \hat{\boldsymbol{y}}\right\|_2 + \ell\left(A(\boldsymbol{X})\boldsymbol{W}, \boldsymbol{c}_t\right)\right]$$

X: Image
y: Estimated age from **Real** image
$\hat{y}$: Estimated age from **Fake** image
A: Age Estimation Network
W: Age group Estimation head
c_t: Age group
l: Cross entropy loss
$\|y - \hat{y}\|^2$: Squared L2 norm

```
def age_criterion(self, input, gt_age):
    opt = self.opt
    age_logit, group_logit = self.age_classifier(input)
    return F.mse_loss(get_dex_age(age_logit), gt_age) + \
           F.cross_entropy(group_logit, age2group(gt_age, opt.age_group).long())
```

3. Identity Consistency Loss

$$\mathcal{L}_{\text{pix}} = \mathbb{E}_{\boldsymbol{X}_s}\left|G(\boldsymbol{X}_s, \boldsymbol{\lambda}_{s:t}) - \boldsymbol{X}_s\right|,$$ 単純にpixel単位の誤差

$$\mathcal{L}_{\text{ssim}} = \mathbb{E}_{\boldsymbol{X}_s}\left[1 - \text{SSIM}\big(G(\boldsymbol{X}_s, \boldsymbol{\lambda}_{s:t}), \boldsymbol{X}_s\big)\right],$$ どれだけ画像の構造が似ているか

$$\mathcal{L}_{\text{fea}} = \mathbb{E}_{\boldsymbol{X}_s}\left\|\phi(G(\boldsymbol{X}_s, \boldsymbol{\lambda}_{s:t})) - \phi(\boldsymbol{X}_s)\right\|_F^2.$$ VGG16の10層分だけ使って、特徴マップ同士のSquared L2 norm

SSIM: Structual similarity

ϕ: VGG based CNN model (10-layers conv)

$$\mathcal{L}_{\text{ide}} = (1 - \alpha_{\text{ssim}})\mathcal{L}_{\text{pix}} + \alpha_{\text{ssim}}\mathcal{L}_{\text{ssim}} + \alpha_{\text{fea}}\mathcal{L}_{\text{fea}}$$

α: Hyper parameter

4. Final Loss

$$\mathcal{L}_G = \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{age}}\mathcal{L}_{\text{age}} + \lambda_{\text{ide}}\mathcal{L}_{\text{ide}}$$

$$\begin{aligned}\mathcal{L}_D = &\frac{1}{2}\mathbb{E}_{\boldsymbol{X}}\Big[D\big([\boldsymbol{X};\boldsymbol{C}]\big) - 1\Big]^2 + \\ &\frac{1}{2}\mathbb{E}_{\boldsymbol{X}_s}\Big[D\big([G(\boldsymbol{X}_s, \boldsymbol{\lambda}_{s:t});\boldsymbol{C}_t]\big)\Big]^2\end{aligned}$$

Contents

Experiments

Dataset

Datast	Number of data	Use
MORPH[*]	55,124	Training (80% for train, 20% for test)
CACD[**]	163,446 (2,000people)	Training (80% for train, 20% for test)
FG-NET[***]	1,002 (82people)	Testing

[*] K. Ricanek, et al., MORPH, FGR, 2006.
[**] B. Chen, et al., CACD, IEEE Trans. Multimedia, 2015.
[***] A. Lanitis, et al., FG-NET, IEEE Trans. Pattern Anal, 2002

Data Preprocessing

Registration : Landmark detected by Face++ API [*4] (No details provided in the paper.)

Resize : 256 × 256
Normalize : mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]
Input Pixel range : [0, 255] ➡ [-1, 1]

[*4] https://www.faceplusplus.com/face-detection/

Experiments

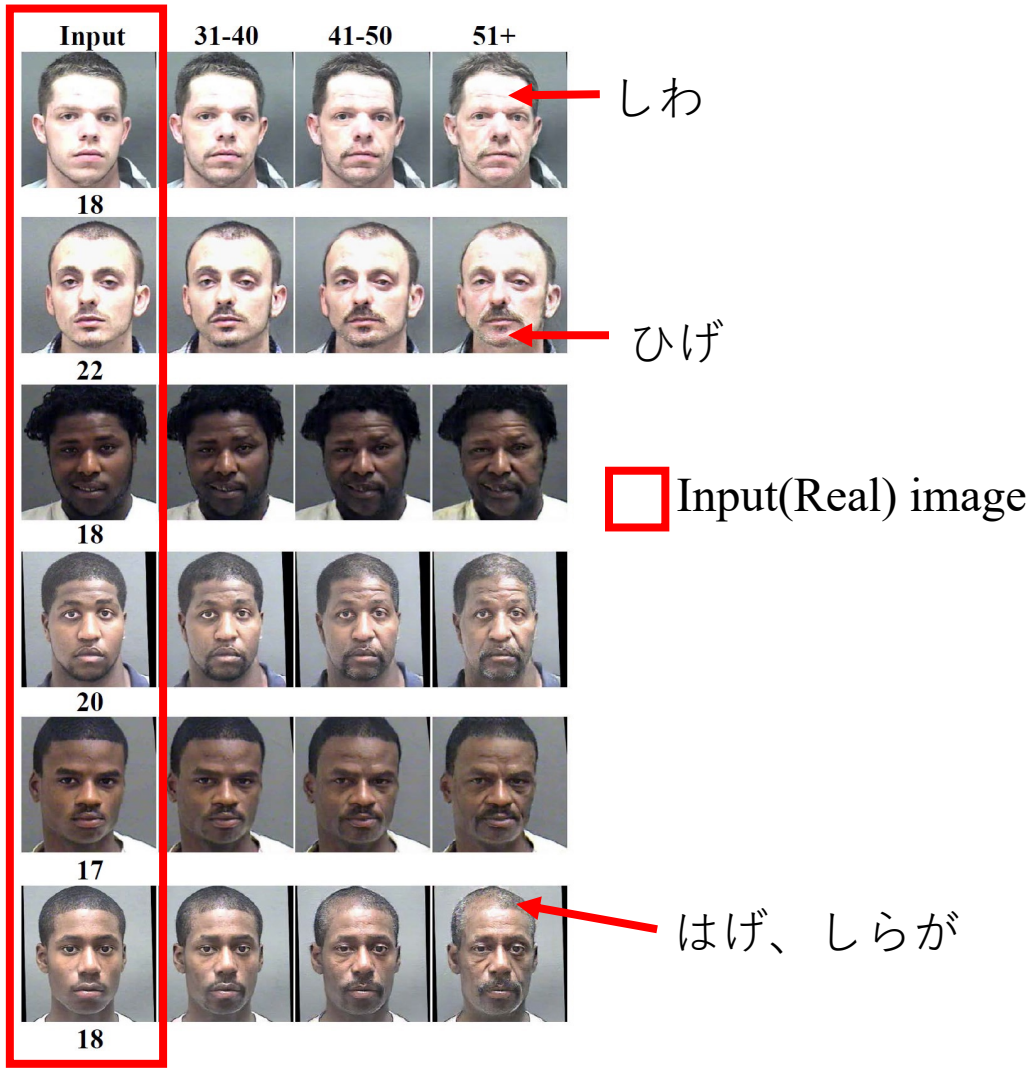
Dataset

	parameter
GPU	NVIDIA GTX 2080 Ti ×1
Iteration	200,000
Optimizer	Adam (lr=1e-4, β1=0.5, β2=0.99)
Batch size	12
Loss function param	$\alpha_{ssim} = 0.15, \alpha_{fea} = 0.025$ $\lambda_{adv} = 100, \lambda_{ide} = 0.02, \lambda_{age} = 0.4$
Framework	PyTorch v1.3.1

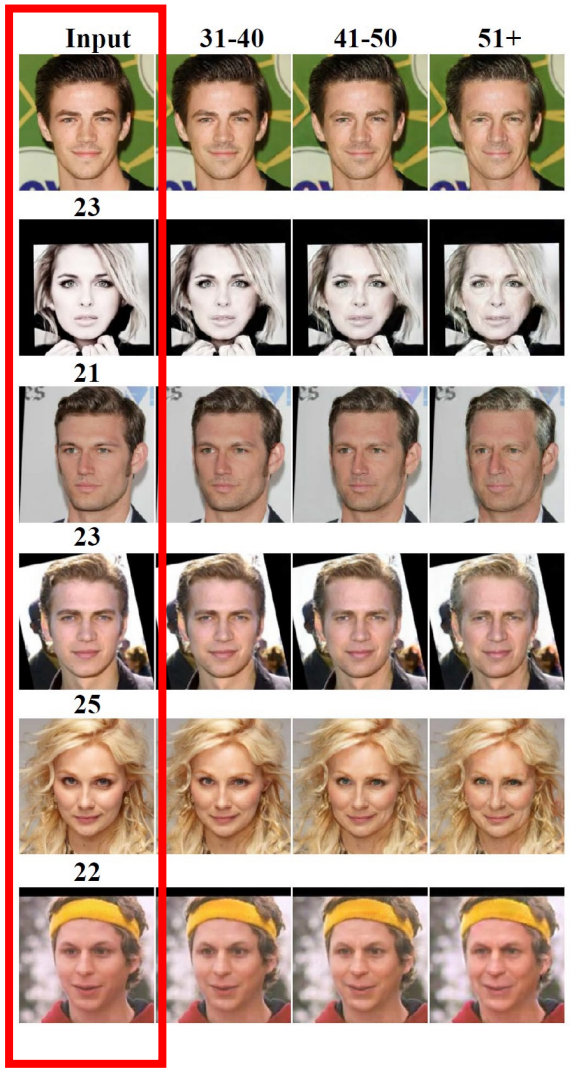
Contents

Qualitative Results (Unpaired data)

Generated aged faces on the MORPH dataset.



Generated aged faces on the CACD dataset.



Qualitative Results (Unpaired data)

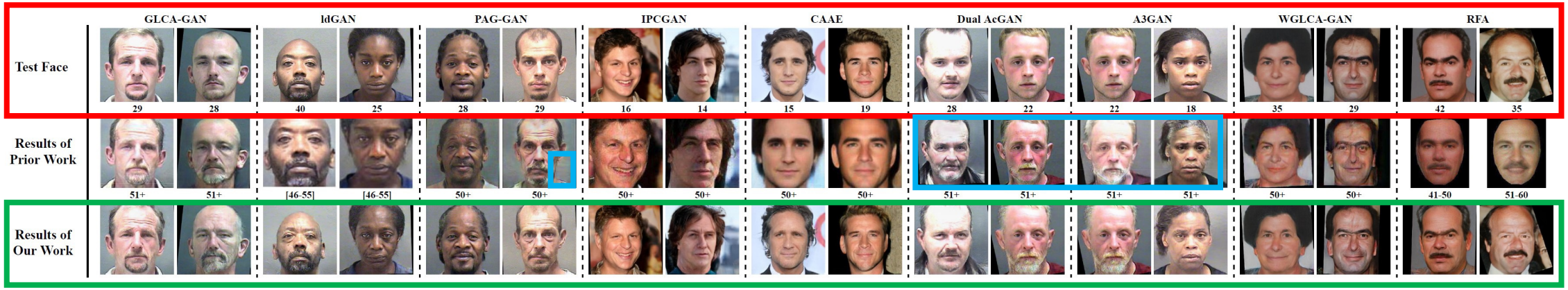


Fig. 6: Performance comparison with prior work on the MORPH and CACD datasets. We showcase the input young faces in the first row with their real age labels below the image. The second row presents two sample results of prior work of seven recently published face aging methods. The third row shows our results of the same input faces in the same age groups as the prior work. Zoom in for a better view of image details.

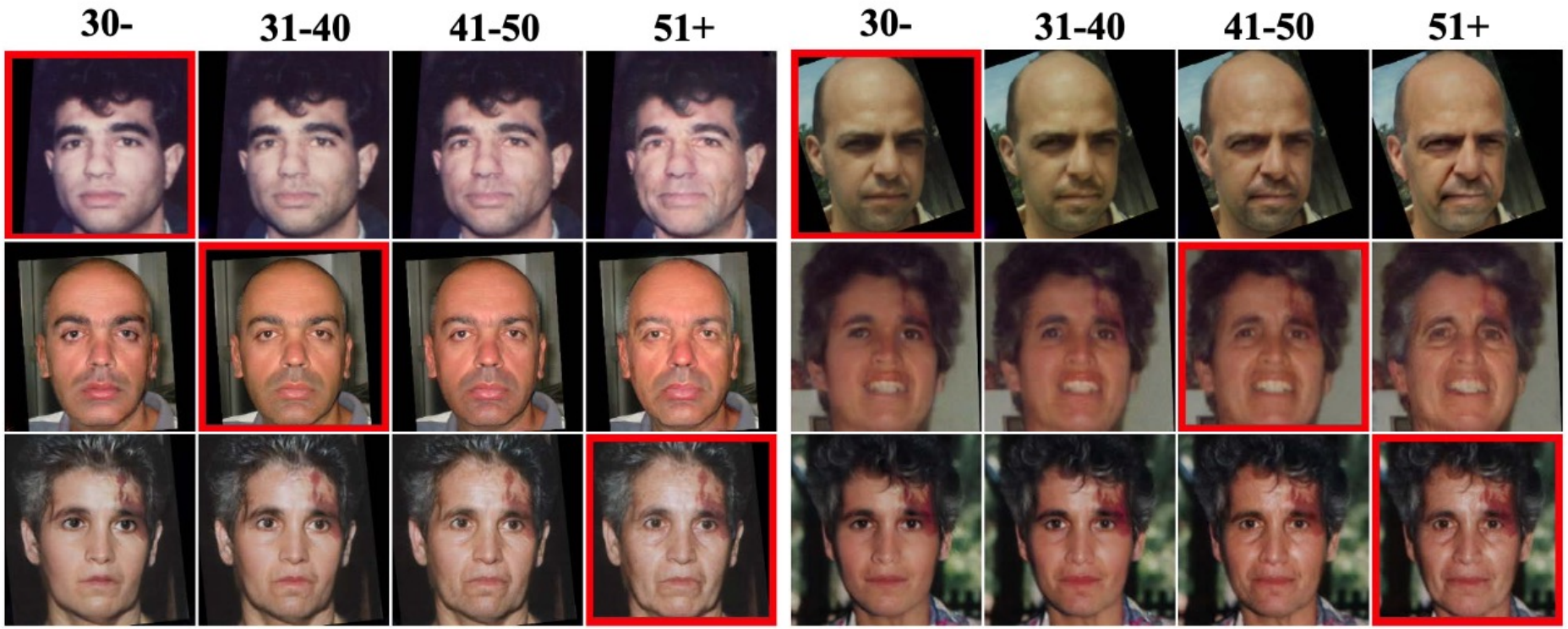
Input(Real) image

Output image using PFA-GAN

It seems to preserve the shape and characteristics of the individuals better than conventional methods.

Qualitative Results (model trained by CAAD dataset)

FG-NET[*]

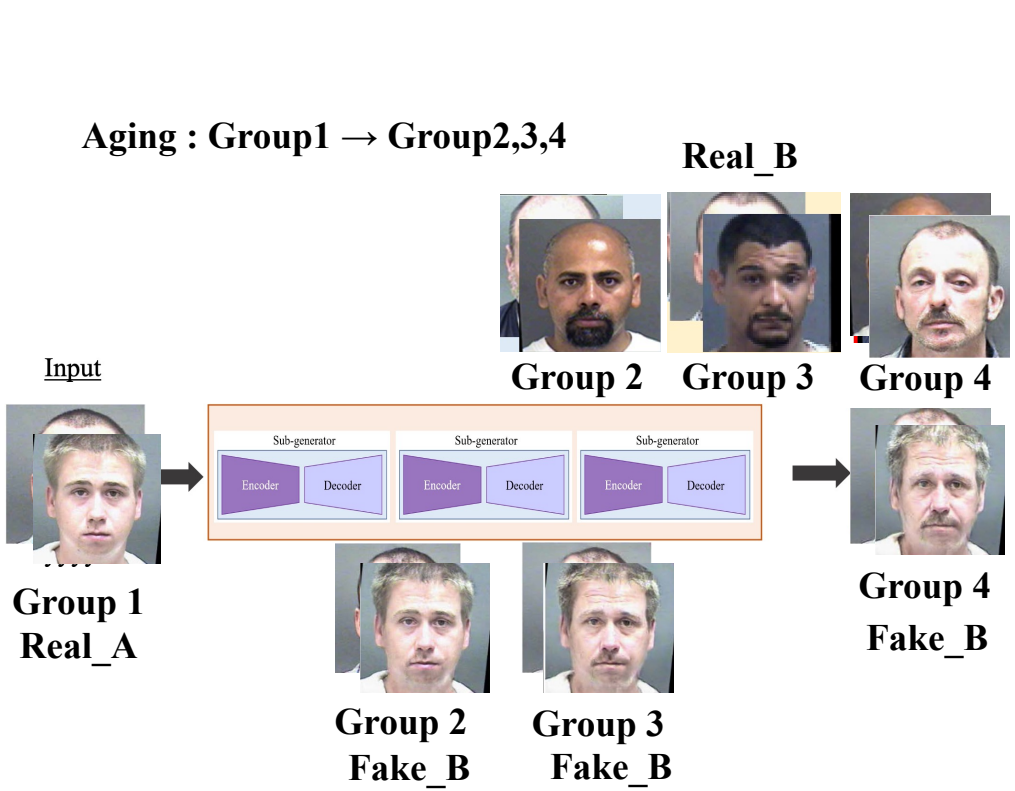


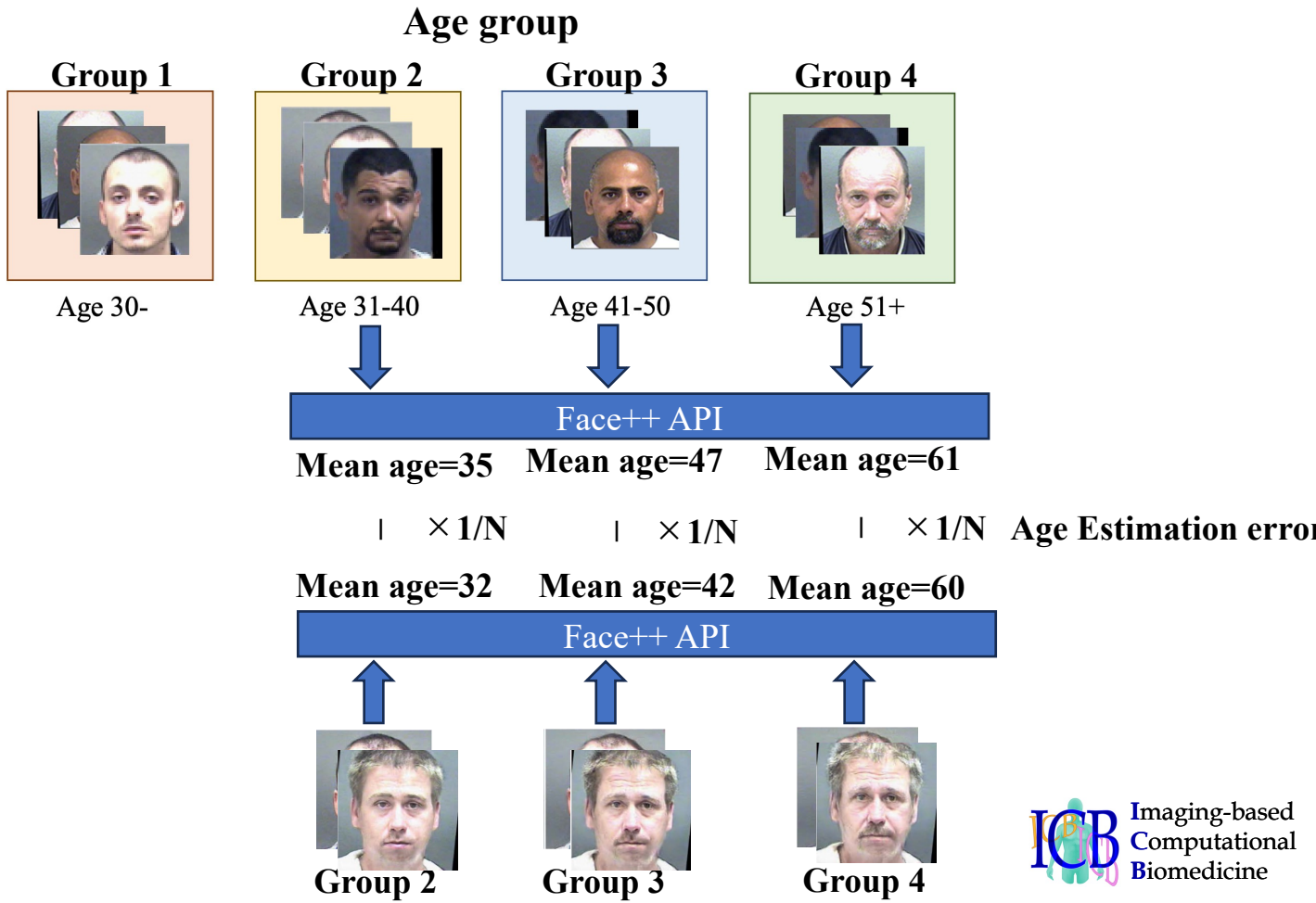
Input(Real) image

[*] A. Lanitis, et al., FG-NET, IEEE Trans. Pattern Anal, 2002

Quantitative Results

Age Estimation Error : The absolute value between the mean estimate ages of real faces and fake faces in each age group. Lower values are considered better.





The model was trained separately for each target age group.

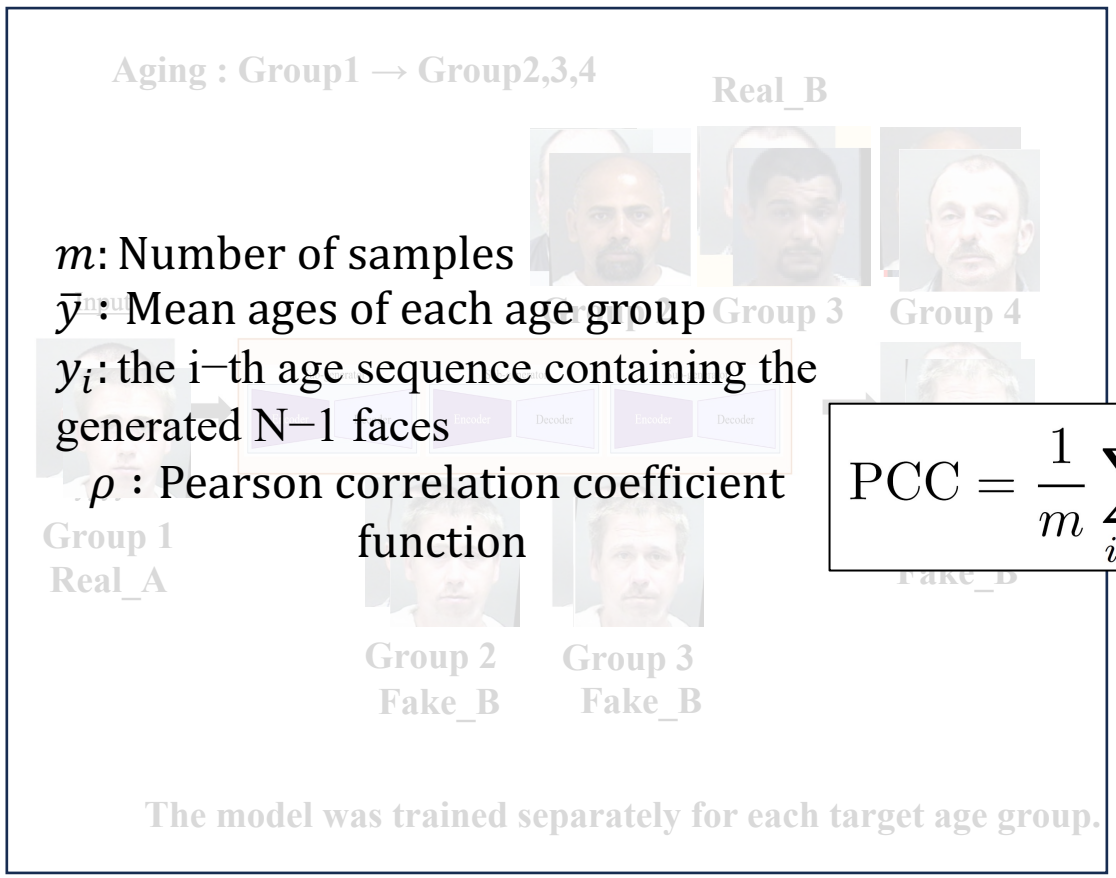
Imaging-based Computational Biomedicine

Quantitative Results

Age Estimation Error : The absolute value between the mean estimate ages of real faces and fake faces in each age group. Lower values are considered better.

PCC : Correlation between the average age of each age group and the generated age belonging to each age group. Higher values are considered better.

Aging : Group1 → Group2,3,4



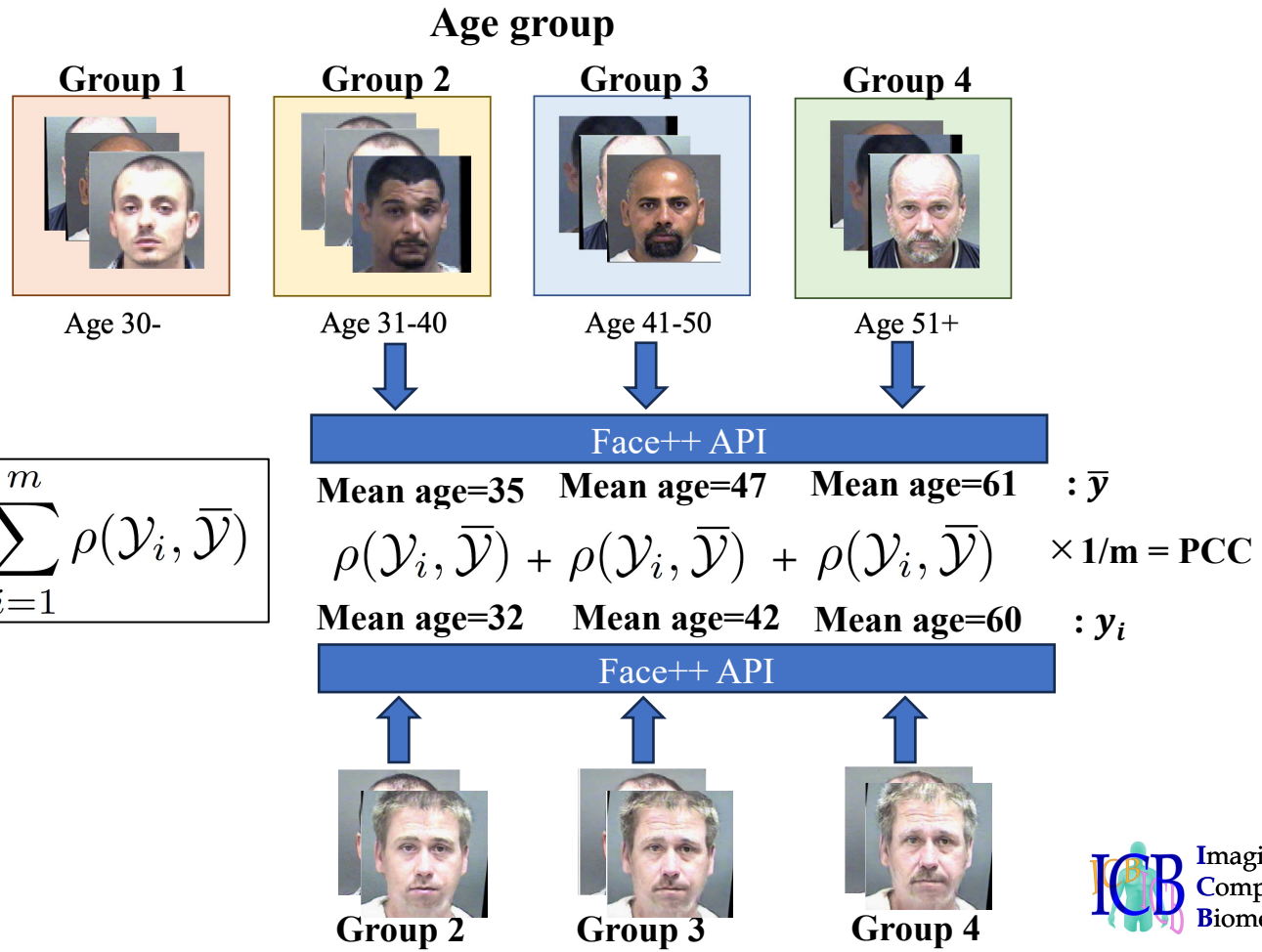
m: Number of samples

$\bar{y}$: Mean ages of each age group

y_i: the i−th age sequence containing the generated N−1 faces

ρ : Pearson correlation coefficient function

$$\mathrm{PCC} = \frac{1}{m}\sum_{i=1}^{m}\rho(\mathcal{Y}_i, \bar{\mathcal{Y}})$$



Imaging-based Computational Biomedicine

Quantitative Results

Age Estimation Error : The absolute value between the mean estimate ages of real faces and fake faces in each age group. Lower values are considered better.

PCC : Correlation between the average age of each age group and the generated age belonging to each age group. Higher values are considered better.

Inception Score : Evaluate the image quality and diversity, Higher values are considered better. (I will not explain. https://arxiv.org/abs/1606.03498)

TABLE I: Quantitative results of age estimation error of three age groups, the Pearson correlation coefficient (PCC), and the inception score (IS) on the MORPH and CACD datasets. For age estimation error, we report the absolute value between the mean estimate ages of real faces and fake faces in each age group. IS are calculated over all aged test faces. We also report the results of applying IPCGAN and PAG-GAN sequentially to age faces, which are denoted as IPCGAN$^\sharp$ and PAG-GAN$^\sharp$, respectively. The best results are highlighted in bold.

MORPH						CACD					
Method	Age Estimation Error			PCC	IS	Method	Age Estimation Error			PCC	IS
	31 - 40	41 - 50	51+				31 - 40	41 - 50	51+		
CAAE	4.41	5.84	6.07	0.937	2.38	CAAE	3.55	5.07	5.32	0.946	6.26
IPCGAN	2.04	2.68	2.01	0.978	3.09	IPCGAN	1.78	0.50	2.64	0.972	29.07
IPCGAN$^\sharp$	2.04	1.05	1.57	0.982	2.71	IPCGAN$^\sharp$	1.78	3.00	1.66	0.978	22.58
WGLCA-GAN	3.52	1.41	2.98	0.974	3.15	WGLCA-GAN	0.63	1.75	2.33	0.981	29.60
PAG-GAN	0.77	0.43	1.56	0.955	3.67	PAG-GAN	0.94	1.09	1.27	0.951	26.43
PAG-GAN$^\sharp$	0.77	1.01	1.34	0.968	2.87	PAG-GAN$^\sharp$	0.94	1.12	0.72	0.971	19.20
PFA-GAN	**0.38**	**0.14**	**1.11**	**0.989**	**3.90**	PFA-GAN	**0.41**	**0.11**	**0.37**	**0.986**	**33.39**
w/o DEX	1.74	1.55	1.40	0.983	3.49	w/o DEX	1.13	0.38	1.38	0.983	32.28
w/o PFA	0.69	1.27	1.49	0.979	3.01	w/o PFA	0.72	0.47	1.04	0.976	30.37

Discussion

Unlike previous cGAN-based methods, such as GLCA-GAN, ldGAN, IPCGAN, and CAAE, PFA-GAN exhibits superior performance in aging faces at higher resolutions (x2), showing enhanced aging details and realistic aging effects.

The success of PFA-GAN in generating improved results can be attributed to its simultaneous learning of multiple sub-generators, exclusive focus on aging by age group, and the propagation of each error backward.

As the number of age groups increases, the difficulty of generation rises, and patterns between adjacent networks become blurred, which can be a problem.
In other words, it can learn changes when the differences are somewhat large, but it cannot learn minute aging changes when the differences are small.

A possible approach to address this problem seems to be to learn each subnetwork independently and in advance, and then to perform end-to-end fine-tuning as the next step.

Conclusion

Unpaired data could be used to generate facial aging with high accuracy. Compared to conventional cGAN-based methods, the end-to-end generation enabled smooth generation while preserving identity.