Enhancing Face Data Diversity with Style-based Face Aging

Supplementary Material

1 Additional qualitative results

We present in Fig. 1 additional samples from the proposed as well as the baseline methods. These results support our conclusion that our method outperforms the baselines, especially for the $0 - 30$ and 50+ age classes, which is crucial for diversity-enhancing augmentation methods.

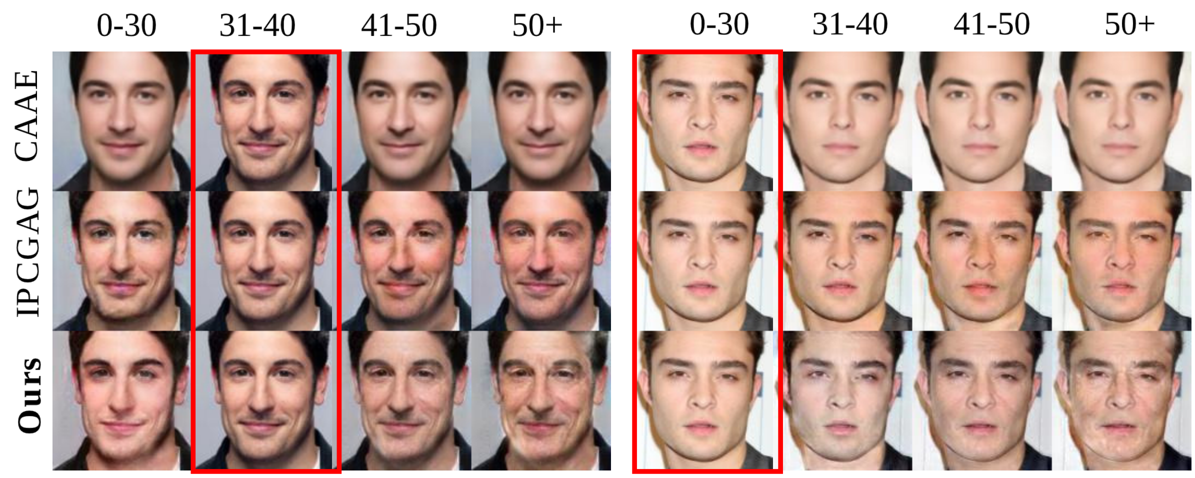
2 Intra-class Aging

We present in Fig. 2 further results on intra-class aging. In particular, given an older-looking target, our method is even able to age faces that belong to the oldest age class, by transferring more crude aging patterns.

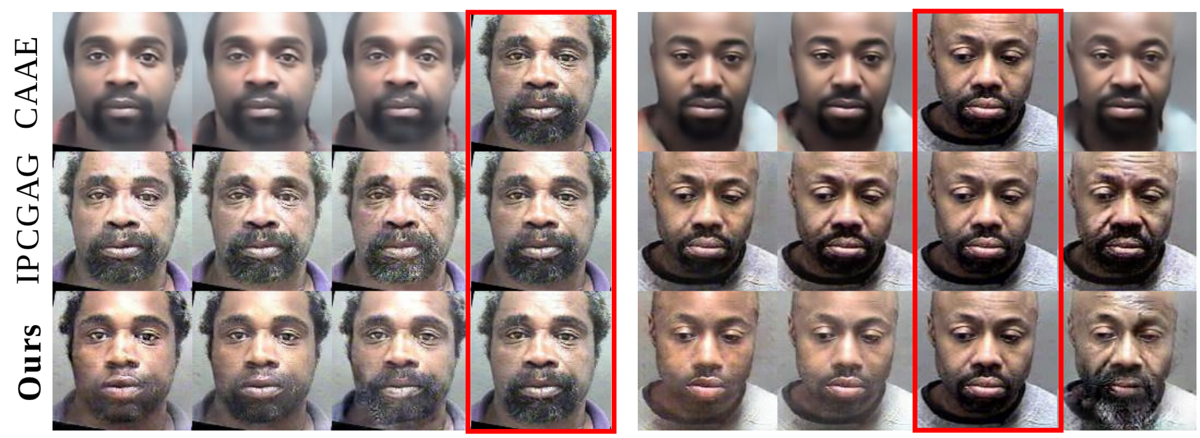
Figure 2: Intra-class diversity: Given input images (top row) of the age class 50+, we can synthesize even older looking faces by transfering the aging patterns of an older image from the same class.

3 Detailed Network Architecture

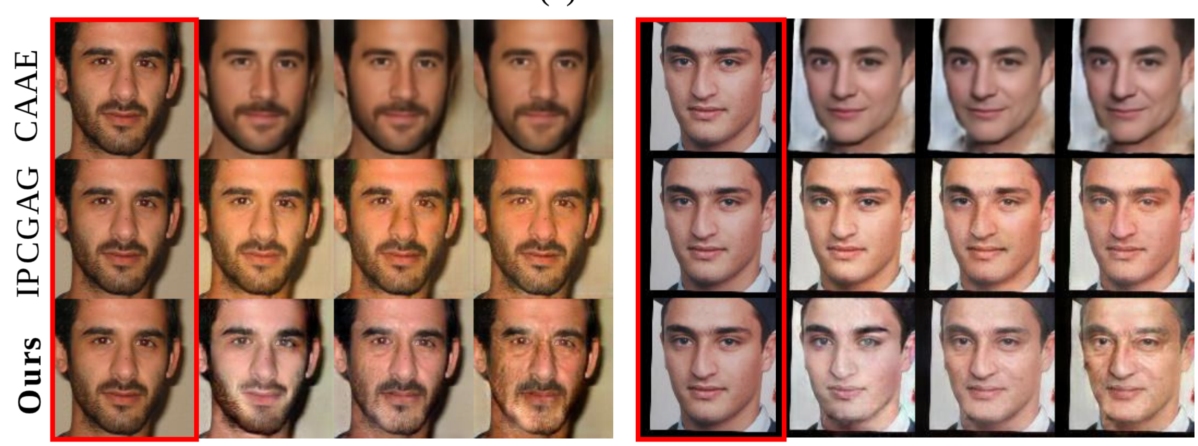
The specifics of our networks, including our choice of layers and activation functions, can be found in Table 3 and Table 4.



(a) CACD



(b) MORPH



(c) FG-NET

Figure 1: Additional baseline comparisons: translating an image from the test set to all ages classes on CACD, MORPH and FG-NET. The red column highlights the input image, positioned in its corresponding age class.

Section	Name	Dimensions (In)	Dimensions (Out)	Layers
Encoder	e1	$(b, 128, 128, 3)$	$(b, 128, 128, 64)$	Conv2d(f=64, k=7, s=1) $\rightarrow$ IN
	e2	$(b, 128, 128, 64)$	$(b, 64, 64, 128)$	ReLu $\rightarrow$ Conv2d(f=128, k=4 s=2) $\rightarrow$ IN
	e3	$(b, 64, 64, 128)$	$(b, 32, 32, 256)$	ReLu $\rightarrow$ Conv2d(f=256, k=4 s=2) $\rightarrow$ IN
	e4	$(b, 32, 32, 256)$	$(b, 16, 16, 512)$	ReLu $\rightarrow$ Conv2d(f=512, k=4 s=2) $\rightarrow$ IN
	e5	$(b, 16, 16, 256)$	$(b, 8, 8, 512)$	ReLu $\rightarrow$ Conv2d(f=512, k=4 s=2) $\rightarrow$ IN
	e6	$(b, 8, 8, 512)$	$(b, 4, 4, 512)$	ReLu $\rightarrow$ Conv2d(f=512, k=4 s=2) $\rightarrow$ IN
	e7	$(b, 4, 4, 512)$	$(b, 2, 2, 512)$	ReLu $\rightarrow$ Conv2d(f=512, k=4 s=2) $\rightarrow$ IN
Decoder	z	$(b, 2, 2, 512)$	$(b, 2, 2, 512)$	ReLu $\rightarrow$ Conv2d(f=512, k=3 s=1) $\rightarrow$ AdaIN $\rightarrow$ + e7 $\rightarrow$ ReLu
	d1	$(b, 2, 2, 512)$	$(b, 4, 4, 512)$	ConvT2d(f=512, k=4 s=2) $\rightarrow$ AdaIN $\rightarrow$ + e_6 $\rightarrow$ ReLu
	d2	$(b, 4, 4, 512)$	$(b, 8, 8, 512)$	ConvT2d(f=512, k=4 s=2) $\rightarrow$ AdaIN $\rightarrow$ + e_5 $\rightarrow$ ReLu
	d3	$(b, 8, 8, 512)$	$(b, 16, 16, 512)$	ConvT2d(f=512, k=4 s=2) $\rightarrow$AdaIN $\rightarrow$ + e_4 $\rightarrow$ ReLu
	d4	$(b, 16, 16, 512)$	$(b, 32, 32, 256)$	ConvT2d(f=256, k=4 s=2) $\rightarrow$ AdaIN $\rightarrow$ + e_3 $\rightarrow$ ReLu
	d5	$(b, 32, 32, 256)$	$(b, 64, 64, 128)$	ConvT2d(f=128, k=4 s=2) $\rightarrow$ AdaIN $\rightarrow$ + e_2 $\rightarrow$ ReLu
	d6	$(b, 64, 64, 128)$	$(b, 128, 128, 64)$	ConvT2d(f=64, k=4 s=2) $\rightarrow$ AdaIN $\rightarrow$ + e_1 $\rightarrow$ ReLu
	d7	$(b, 128, 128, 64)$	$(b, 128, 128, 3)$	ConvT2d(f=3, k=7 s=1)

Figure 3: Encoder and Decoder architecture: b denotes the mini-batch size, and 'IN' refers to the Instance Normalization operation. We use '+e_i' to donate a symmetric skip-connection element-wise addition operation from the encoder to the decoder.

Section	Name	Dimensions (In)	Dimensions (Out)	Layers
Discriminator	d1	$(b, 128, 128, 3)$	$(b, 128, 128, 64)$	Conv2d(f=64, k=7, s=1) $\rightarrow$ LeakyReLu
	d2	$(b, 128, 128, 64)$	$(b, 64, 64, 128)$	Conv2d(f=128, k=4 s=2) $\rightarrow$ LeakyReLu
	d3	$(b, 64, 64, 128)$	$(b, 32, 32, 256)$	Conv2d(f=256, k=4 s=2) $\rightarrow$ LeakyReLu
	d4	$(b, 32, 32, 256)$	$(b, 16, 16, 512)$	Conv2d(f=512, k=4 s=2) $\rightarrow$ LeakyReLu
	d5	$(b, 16, 16, 256)$	$(b, 8, 8, 512)$	Conv2d(f=512, k=4 s=2) $\rightarrow$ LeakyReLu
	d6	$(b, 8, 8, 512)$	$(b, 4, 4, 512)$	Conv2d(f=512, k=4 s=2) $\rightarrow$ LeakyReLu
	d7	$(b, 4, 4, 512)$	$(b, 2, 2, 512)$	Conv2d(f=512, k=4 s=2) $\rightarrow$ LeakyReLu
	logits	$(b, 2, 2, 512)$	$(b, 4)$	Flatten $\rightarrow$ FC(4)

Figure 4: Discriminator architecture: b again denotes the mini-batch size.