1. Supplementary Material for StarGAN

1.1. Training with Multiple Datasets

Fig. 1 shows an overview of StarGAN when learning from both the CelebA and RaFD datasets. As can be seen at the top of the figure, the label for CelebA contains binary attributes (Black, Blond, Brown, Male, and Young), while the label for RaFD provides information on categorical attributes (Angry, Fearful, Happy, Sad, and Disgusted). The mask vector is a two-dimensional one-hot vector which indicates whether the CelebA or RaFD label is valid.

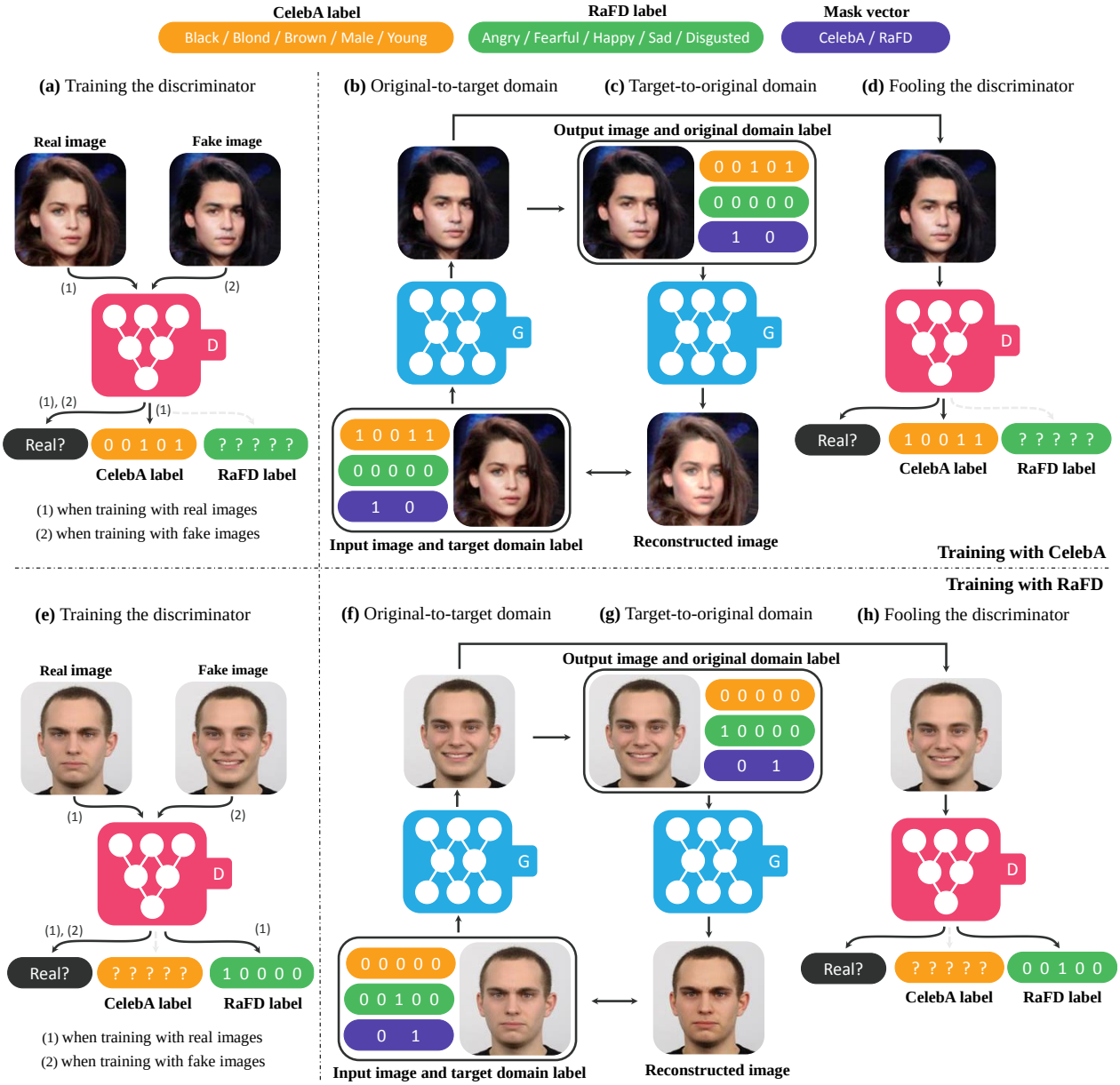

Figure 1. Overview of StarGAN when training with both CelebA and RaFD. **(a)** ∼ **(d)** shows the training process using CelebA, and **(e)** ∼ **(h)** shows the training process using RaFD. **(a), (e)** The discriminator D learns to distinguish between real and fake images and minimize the classification error only for the known label. **(b), (c), (f), (g)** When the mask vector (purple) is [1, 0], the generator G learns to focus on the CelebA label (yellow) and ignore the RaFD label (green) to perform image-to-image translation, and vice versa when the mask vector is [0, 1]. **(d), (h)** G tries to generate images that are both indistinguishable from real images and classifiable by D as belonging to the target domain.

1.2. Network Architecture

The network architectures of StarGAN are shown in Table 1 and 2. For the generator network, we use instance normalization in all layers except the last output layer. For the discriminator network, we use Leaky ReLU with a negative slope of 0.01. There are some notations; n_d: the number of domain, n_c: the dimension of domain labels ($n_d + 2$ when training with both the CelebA and RaFD datasets, otherwise same as n_d), N: the number of output channels, K: kernel size, S: stride size, P: padding size, IN: instance normalization.

Part	Input → Output Shape	Layer Information
Down-sampling	$(h, w, 3 + n_c) \rightarrow (h, w, 64)$	CONV-(N64, K7x7, S1, P3), IN, ReLU
	$(h, w, 64) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	CONV-(N128, K4x4, S2, P1), IN, ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	CONV-(N256, K4x4, S2, P1), IN, ReLU
Bottleneck	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
Up-sampling	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	DECONV-(N128, K4x4, S2, P1), IN, ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (h, w, 64)$	DECONV-(N64, K4x4, S2, P1), IN, ReLU
	$(h, w, 64) \rightarrow (h, w, 3)$	CONV-(N3, K7x7, S1, P3), Tanh

Table 1. Generator network architecture

Layer	Input → Output Shape	Layer Information
Input Layer	$(h, w, 3) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	CONV-(N256, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{16}, \frac{w}{16}, 512)$	CONV-(N512, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{16}, \frac{w}{16}, 512) \rightarrow (\frac{h}{32}, \frac{w}{32}, 1024)$	CONV-(N1024, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{32}, \frac{w}{32}, 1024) \rightarrow (\frac{h}{64}, \frac{w}{64}, 2048)$	CONV-(N2048, K4x4, S2, P1), Leaky ReLU
Output Layer (D_{src})	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (\frac{h}{64}, \frac{w}{64}, 1)$	CONV-(N1, K3x3, S1, P1)
Output Layer (D_{cls})	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (1, 1, n_d)$	CONV-(N(n_d), K$\frac{h}{64}$x$\frac{w}{64}$, S1, P0)

Table 2. Discriminator network architecture

1.3. Additional Qualitative Results

Figs. 2, 3, 4, and 5 show additional images with 256×256 resolutions generated by StarGAN. All images were generated by a single generator trained on both the CelebA and RaFD datasets. We trained StarGAN on a single NVIDIA Pascal M40 GPU for seven days.

Figure 2. Single and multiple attribute transfer on CelebA (Input, Black hair, Blond hair, Brown hair, Gender, Aged, Hair color + Gender, Hair color + Aged, Gender + Aged, Hair color + Gender + Aged).

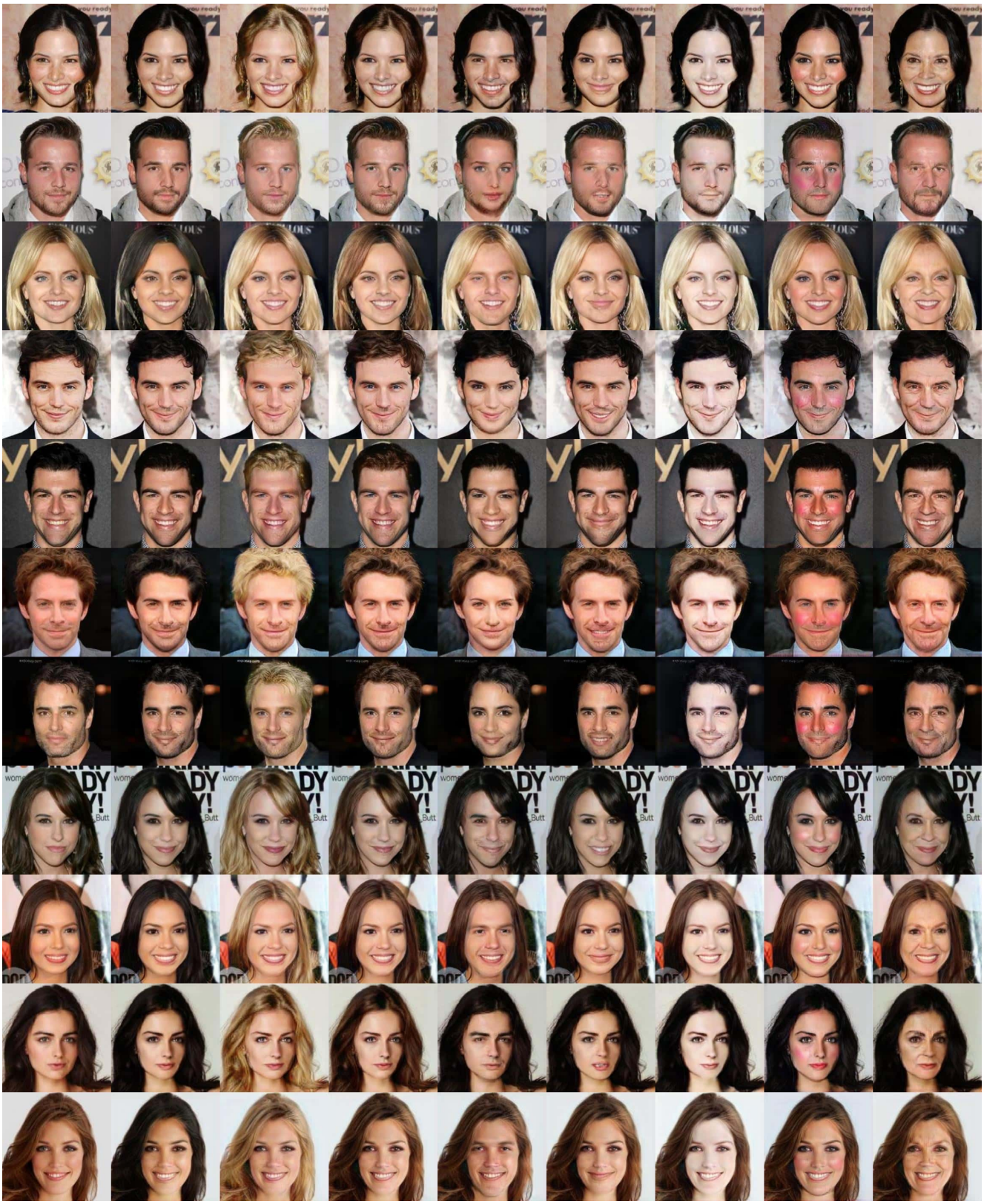

Figure 3. Single attribute transfer on CelebA (Input, Black hair, Blond hair, Brown hair, Gender, Mouth, Pale skin, Rose cheek, Aged).

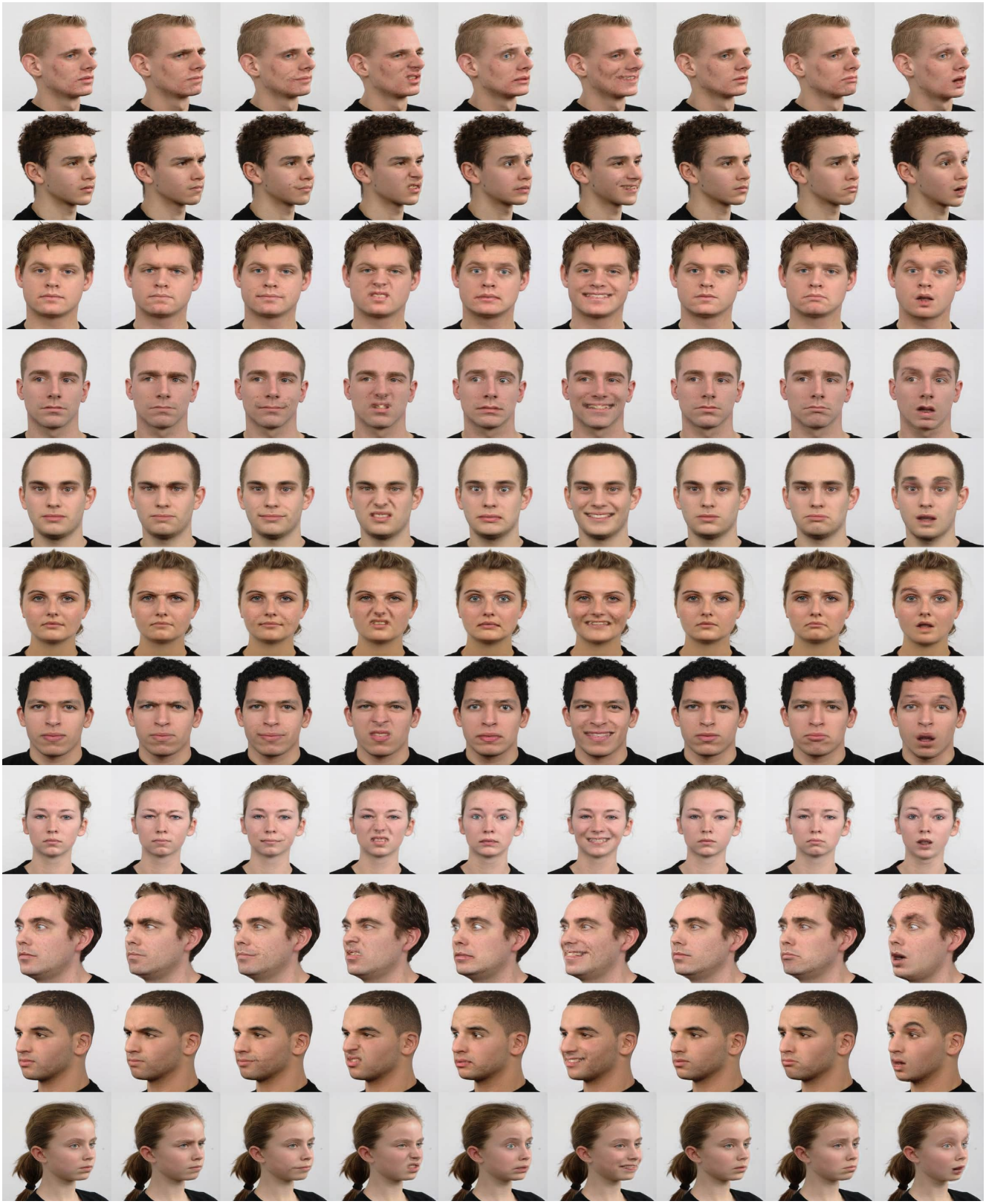

Figure 4. Emotional expression synthesis on RaFD (Input, Angry, Contemptuous, Disgusted, Fearful, Happy, Neutral, Sad, Surprised).

Figure 5. Emotional expression synthesis on CelebA (Input, Angry, Contemptuous, Disgusted, Fearful, Happy, Neutral, Sad, Surprised).

Figure 6. Collection style transfer. We trained StarGAN on a landscape dataset with the following seven domains: real photographs and six different styles of artworks. The images show the results of transferring the real photographs into the six artistic styles of Cezanne, Corot, Earle, Monet, Ukiyoe, and van Gogh.