# Homomorphic Interpolation Network for Unpaired Image-to-image Translation

Ying-Cong Chen, Jiaya Jia, *Fellow, IEEE*

**Abstract**—Generative adversarial networks have achieved great success in unpaired image-to-image translation. Cycle consistency, a key component for this task, allows modeling the relationship between two distinct domains without paired data. In this paper, we propose an alternative framework, as an extension of latent space interpolation, to consider the intermediate region between two domains during translation. It is based on the assumption that in a flat and smooth latent space, there exist many paths that connect two sample points. Properly selecting paths makes it possible to change only certain image attributes, which is useful for generating intermediate images between the two domains. With this idea, our framework includes an encoder, an interpolator and a decoder. The encoder maps natural images to a convex and smooth latent space where interpolation is applicable. The interpolator controls the interpolation path so that desired intermediate samples can be obtained. Finally, the decoder inverts interpolated features back to pixel space. We also show that by choosing different reference images and interpolation paths, this framework can be applied to multi-domain and multi-modal translation. Extensive experiments manifest that our framework achieves superior results and is flexible for various tasks.

## 1 Introduction

Unpaired image-to-image translation and latent space interpolation were developed separately and serve different applications. Unpaired image-to-image translation [1], [2], [3], [4], [5], [6] aims to map images from one domain to another, e.g., translating a collection of neutral faces to smiling ones. Since no pair information is available, connection of different domains is usually built upon the cycle-consistency constraint [1], which largely promotes the capacity of generative models and leads to many impressive results.

When the purpose is to generate a sequence of images between the input two domains, intermediate states should be considered, which is however beyond the capability of the cycle-consistency constraint. We show an example in Fig. 1 – directly using StarGAN [4] does not generate a natural sequence (or expression flow) to gradually close mouth, and instead there exists a quick change between (c) and (d).

On the other hand, to generate smooth flow, latent space interpolation [7], [8], [9] focuses on intermediate states based on an assumption that deep neural networks can model natural images as flat and smooth distributions. Specifically, if $x$ and $y$ are sampled from two respective domains $\mathcal{X}$ and $\mathcal{Y}$, moving from $x$ toward $y$ in the latent space continuously produces realistic images from domain $\mathcal{X}$ to $\mathcal{Y}$. Albeit this nice property, this method cannot directly serve image-to-image translation because it does not distinguish among different attribute factors, and thus, for example, makes complicated expression change always tangled with identity or background change.

In this paper, we address latent space interpolation in unpaired image-to-image translation. This solution inherently allows modeling intermediate regions between different domains, with additional important and appealing capacity of multi-domain/multi-modal translation. Since in a flat and smooth latent space, many paths exist to connect two samples, interpolating along different paths leads to diverse intermediate results [10]. Our idea is to *choose the path that only corresponds to a certain attribute component* to make transition natural to human perception. Here the term *attribute* defines image domains, e.g., the *smiling* attribute divides facial images to *smiling* and *non-smiling* domains. Fig. 2 provides an example where translating between *Male* and *Female* (or *Smiling* and *Non-smiling*) can be achieved by interpolating along path 1(i) (or path 2(i)) respectively. Besides multi-domain and continuous translation capacity, as shown in path 1(i) and 3(i), this model can also deal with multi-modal translation.

With this principle, the key to our method is a controllable *interpolator*, whose output is controlled by a vector $\boldsymbol{v}$. Each element of $\boldsymbol{v}$ corresponds to a *mixing* indicator for each

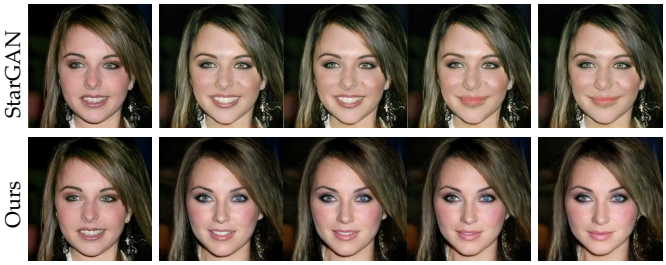



Fig. 1. Rendering intermediate states between (a) "open-mouth" domain and (e) "close-mouth" domain. The first-row results are generated by StarGAN [4]. Rendering intermediate states is achieved by altering the input domain label continuously. (c) and (d) show that abrupt change of expression exists. Our results in the second row model intermediate regions and have smooth translation effect.

• *Ying-Cong Chen is with the Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology. Jiaya-Jia is with the Department of Computer Science, School of Engineering, The Chinese University of Hong Kong. This work is done when Ying-Cong Chen is at the Chinese University of Hong Kong.*
*E-mail: yingcong.ian.chen@gmail.com and leojia@cse.cuhk.edu.hk*

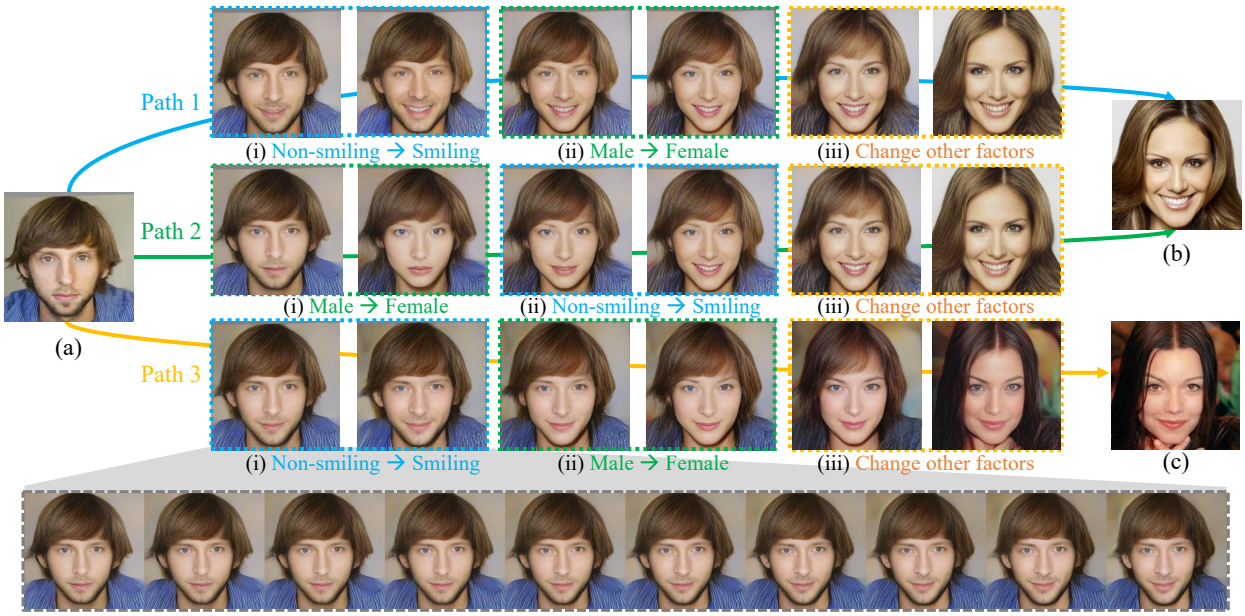



Fig. 2. Illustration of latent space interpolation along different paths. Paths 1 and 2 connect (a) "non-smiling male" and (b) " smiling female". They change facial attributes in different orders – i.e., path 1 changes the smiling expression first while path 2 interpolates gender. They naturally serve the **multi-domain image-to-image translation** task where path 1(i) and 2(i) form translation between smiling-to-nonsmiling domains and male-to-female domains respectively. Path 3(i) synthesizes smile different from path 1(i). Thus, using different target-domain samples, our method can produce results required for each domain, termed as **multi-modal image-to-image translation**. Image sequence of the last row illustrates the continuous change of path 3(i).

attribute. Take path 3(i) of Fig. 2 as an example, a proper $\boldsymbol{v}$ only deals with the smiling attribute between (a) and (c), while keeping other attributes untouched.

Although promising, this strategy requires conquering a few difficulties. First, interpolation is only allowed in a smooth and flat space. VAE [11] imposes Gaussian prior on the latent feature space so that interpolation is allowed. However, it may not lead to satisfactory results, as a Gaussian prior could over-regularize the model, causing the under-fitting problem [12], [13], [14]. Our solution is to directly minimize the Wasserstein distance between the interpolated and real samples of the latent space. This makes interpolated sample distribution as close as possible to the real ones. Besides, it does not impose an explicit prior distribution like VAEs [11], [12], [14], thus less suffer from the over-regularization problem. We also introduce a knowledge guidance loss that leverages a well-trained network to regularize the latent space, which further improves interpolation quality. Finally, a homomorphic loss is introduced to train the controllable interpolator. Our contribution is as follows.

- We propose an interpolation-based framework for unpaired image-to-image translation, which is feasible for multi-domain, multi-modal and continuous translation tasks.
- We propose a few important strategies to train our model, leading to an interpolatable latent space and a controllable interpolator.
- Extensive experiments show that our model can generate high-quality results and is flexible to serve various applications.

This manuscript extends the conference version [15] with the following major differences. First, we build a new network architecture with the encoder and decoder, leading to better quality and higher resolution. Second, during the course of training encoder and decoder, we additionally incorporate the style loss (Eq. (4)) and the image-level adversarial loss (Eq. (5)). Third, we introduce schematic description (Figs. 3, 4 and 5) to help understand our approach and the overall framework. Fourth, we conduct more quantitative analysis in the ablation study, which is included in Section 4.2. Finally, we present more in-depth analysis of the scope of applications in Section 5.2.

The rest of the paper is organized as follows. Section 2 reviews literature in latent space interpolation, attribute conditional image generation, and unpaired image-to-image translation. In Section 3, we introduce our homomorphic interpolation network. We show experimental results in Section 4, discuss the scope of applications in Section 5, and conclude this paper in Section 7.

## 2 Related Work

**Latent Space Interpolation** Latent space interpolation is widely used to visualize the manifold structure in a flat feature space [8], [9], [16], [17], [18], [19], [20]. Intuitively, semantically interpolation in the latent space indicates that the space captures certain high-level information, which is beneficial for both recognition tasks [17] and generation tasks [16]. However, vanilla interpolation between two images may not be sufficiently useful for generation applications, since all attributes would change together along the interpolation path, and users lose control of individuals.

One remedy is to interpolate along *attribute vectors* rather than between samples [11], [16], [21], [22], [23], [24], [25]. Reed *et al.* [25] modeled the analogy relationship with arithmetic operations. However, it is only applicable to

analogy tasks. Larsen *et al.* [11], Kingma *et al.* [16], Radford *et al.* [21] and Karras *et al.* [22], [23] defined the attribute vector by computing the difference between the average of all positive and negative samples in the dataset. Upchurch *et al.* [26] used $k$-nearest positive/negative samples of the query image instead for computing the attribute vector. Chen *et al.* [24] further used a small CNN network to regress the attribute vector extracted by Upchurch *et al.* [26]. These methods aim to cancel out the influence of non-target attributes and allow users to edit only the target one. They ignore the fact that many attributes are intrinsically multi-modal. As illustrated in Fig. 2(b) and (c), smiling can be quite different. Interpolation with a universal *smiling* attribute vector can only generate the average smile. In contrast, our model can produce multi-modal results with different examples.

**Attribute Conditional Image Generation** Attribute Conditional Image Generation [27], [28], [29] synthesize images based on the attributes provided by users. Yan *et al.* [27] proposed a conditional VAE based model that generated an object image from a high-level attribute description. Lample *et al.* [29] used an encoder to map images to attribute-invariant latent features, then generated the results by combining the latent features with user-specified attribute vector. Lu *et al.* [28] proposed a conditional CycleGAN that took a low-resolution face and an attribute vector as an input, then produced a high-resolution image whose appearance was controlled by the attribute vector. Liu *et al.* [30] proposed a flow-based model that can encode a condition to a latent variable, which allowed for controllable image synthesis. Our model is intrinsically different from these methods, as we do not treat the attributes as conditions for the generator. In our model, the attributes determine the interpolation path. This allows for generating different results with different reference images.

**Unpaired Image-to-image Translation** Unpaired image-to-image translation [1], [2], [4], [31], [32] aims to map images of one domain to another. CycleGAN [1], DiscoGAN [2] and DualGAN [33] are three pioneering methods, which introduce the cycle-consistency constraint to build the connection. There are however a few remaining issues.

The domain scalability issue refers to the incapability of handling more than two domains, which is addressed by StarGAN [4], AttGAN [34] and ModularGAN [35]. StarGAN [4] addresses this issue by taking both an image and the target domain label as input of the generator. Then the generator produces a translated image based on the domain label. AttGAN [34] learns a generic latent feature with its encoder, followed by decoder that generates images of the target domain by jointly taking the latent feature and the domain label. ModularGAN [35] handles this issue by training shared encoder/decoder as well as a set of reusable attribute transformation module. Each module handles the translation between two domains.

The multi-modality issue refers to incapability to produce multiple results, which is addressed by BicycleGAN [6], GDWCT [36], MUNIT [37] and DRIT [3]. BicycleGAN [6] introduces a latent code to the CycleGAN framework. Different latent codes lead to styles of the generated results. GDWCT [36] has an efficient whitening-and-coloring transformation for image-to-image translation. Multi-modality can be achieved using different reference/style images. MUNIT [37] and DRIT [3] decompose an image into two components, i.e., a content code that is domain invariant, and a style code that captures domain-specific property. Translation is achieved by recombining the content code with different style codes in another domain.

The discreteness issue refers to the inability to continuously control the transformation strength between two domains, which is addressed by GANimation [38] and DLOW [39]. GANimation [38] focuses on facial expression translation. It achieves continuous translation by introducing action unit annotation, which describes in the anatomical facial movement defining facial expression. Note that the continuous action unit annotation is costly and limited in the field of facial expression. DLOW [39] uses a variable in CycleGAN [1] to translate images from the source domain to intermediate domains. Since there exist many paths between two domains, DLOW [39] follows the shortest geodesic path. It is intrinsically different from our model since ours allows interpolation along different paths, leading to diverse yet reasonable intermediate results.

Compared with these methods, our model seeks another way to tackle the unpaired image-to-image translation problem. It can be deemed as a general alternative that tackles the domain scalability, multi-modality and discreteness issues simultaneously.

## 3 Proposed Method

Without loss of generality, we take the face attribute translation task as an example to explain our method. We define the dataset as $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2) \cdots (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ of $N$ samples, where $\boldsymbol{x}_i \in \mathbb{R}^{H \times W \times 3}$ and $\boldsymbol{y}_i = [\boldsymbol{y}_i^1, \boldsymbol{y}_i^2, \cdots, \boldsymbol{y}_i^d]$ are the $i$-th face image and its corresponding attributes respectively. The subscript and superscript index samples and attributes respectively.

We further introduce the concept of *grouped attribute*. For example, we can group *angry, happy, sad, contemptuous, disgusted, fear,* and *surprise* – these attributes are provided in RaFD [40] dataset as binary attribute labels – to form the group *expression* attribute. Thus, the plain attributes $\boldsymbol{y}_i$ can be rearranged to $\boldsymbol{z}_i = \{\boldsymbol{z}_i^1, \boldsymbol{z}_i^2, \cdots, \boldsymbol{z}_i^c\}$, where $\boldsymbol{z}_i^k \in \mathbb{R}^{c_i \times 1}$ denotes the $k$-th grouped attribute of the $i$-th sample. This makes it more intuitive to use our model. An instance is a path among 1(i), 2(ii) and 3(i) of Fig. 2, with the *expression* attribute. It considers the 8 expressions rather than only *smiling*.

In the model level, we have an encoder $\boldsymbol{E}$, an interpolator $\boldsymbol{I}$ and a decoder $\boldsymbol{D}$. The encoder $\boldsymbol{E}$ maps images $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ to feature $\boldsymbol{F}_i = \boldsymbol{E}(\boldsymbol{x}_i)$ and $\boldsymbol{F}_j = \boldsymbol{E}(\boldsymbol{x}_j)$, so that the interpolated feature $\boldsymbol{I}(\boldsymbol{F}_i, \boldsymbol{F}_j)$ is indistinguishable from real samples. The interpolator $\boldsymbol{I}$ produces interpolated results of two samples. The decoder $\boldsymbol{D}$ maps the latent features back to the image space. The pipeline is illustrated in Fig. 3. In the following, we will elaborate on the design of each part.

### 3.1 Learning Encoder and Decoder

It is well known that natural images usually lie on a non-convex manifold, making interpolation usually difficult. We

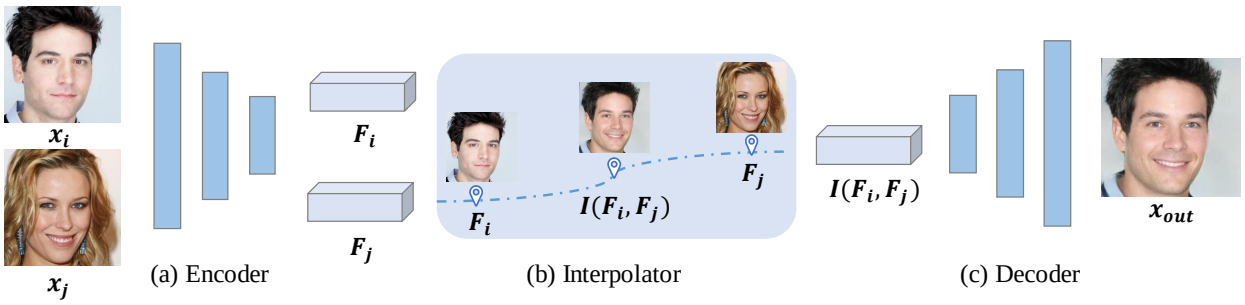



Fig. 3. Pipeline of the proposed homomorphic interpolation framework. (a-c) are the encoder, interpolator and decoder respectively. Given input image $\boldsymbol{x}_i$ and reference image $\boldsymbol{x}_j$, the encoder $\boldsymbol{E}$ maps them to interpolatable latent feature $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$. Then the interpolator $\boldsymbol{I}$ performs interpolation between $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$. Finally, the decoder $\boldsymbol{D}$ inverts the interpolated feature back to the image space.

train an encoder to unfold the image manifold, such that the interpolated samples are easier to be in image space. This is achieved by applying GAN to make interpolated feature similar to that of real samples.

Specifically, we leverage WGAN-GP [41] to train our model. A feature-level discriminator $\boldsymbol{\mathcal{D}_1}$ is trained to maximize the Wasserstein distance between real and interpolated samples. The encoder $\boldsymbol{E}$ and interpolator $\boldsymbol{I}$ are trained to minimize the distance between them. It is formulated as

$$\min_{\boldsymbol{\mathcal{D}_1}} \mathcal{L}_{\boldsymbol{\mathcal{D}_1}} = \mathbb{E}_{\mathbb{P}_{\boldsymbol{I}}}[\boldsymbol{\mathcal{D}_1}(\hat{\boldsymbol{F}})] - \mathbb{E}_{\mathbb{P}_r}[\boldsymbol{\mathcal{D}_1}(\boldsymbol{F})] + \lambda_{gp}\mathcal{L}_{gp}, \quad (1)$$

$$\min_{\boldsymbol{E},\boldsymbol{I}} \mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_1}}} = \mathbb{E}_{\mathbb{P}_r}[\boldsymbol{\mathcal{D}_1}(\boldsymbol{F})] - \mathbb{E}_{\mathbb{P}_{\boldsymbol{I}}}[\boldsymbol{\mathcal{D}_1}(\hat{\boldsymbol{F}})], \quad (2)$$

where $\boldsymbol{F} = \boldsymbol{E}(\boldsymbol{x})$ is the feature extracted by the encoder, $\hat{\boldsymbol{F}}$ is the interpolated feature generated by $\hat{\boldsymbol{F}} = \boldsymbol{I}(\boldsymbol{F}_i, \boldsymbol{F}_j)$, $\mathbb{P}_r$ and $\mathbb{P}_{\boldsymbol{I}}$ are the distributions of real and interpolated samples respectively, and $\mathcal{L}_{gp}$ is the gradient penalty term defined in [41]. Here the interpolator $\boldsymbol{I}$ works with the encoder $\boldsymbol{E}$ cooperatively to generate reasonable images. More details of the interpolator $\boldsymbol{I}$ will be discussed in the later section.

Note that simply using Eqs. (1) and (2) may cause the encoder to map all images to a small feature space where interpolation becomes easy. To an extreme, if the encoder maps all images to a single point, the interpolated and real samples yield Wasserstein distance 0. But this trivial solution carries no information about the images. To avoid it, we additionally incorporate a decoder $\boldsymbol{D}$ to invert features back to images. With this decoder $\boldsymbol{D}$, we define a reconstruction loss as

$$\min_{\boldsymbol{E},\boldsymbol{D}} \mathcal{L}_{recon} = \mathbb{E}(||\Phi_3(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}))) - \Phi_3(\boldsymbol{x})||^2), \quad (3)$$

where $\Phi_3(\boldsymbol{x})$ is the RELU3_1 feature of the VGG network.

To encourage the encoder and decoder better describe and reconstruct details, we further add a style loss and an adversarial loss. The style loss is

$$\min_{\boldsymbol{E},\boldsymbol{D}} \mathcal{L}_{style} = \mathbb{E}(||\boldsymbol{G}(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}))) - \boldsymbol{G}(\boldsymbol{x})||^2), \quad (4)$$

where $\boldsymbol{G}(\boldsymbol{x})$ returns the Gram matrix in the VGG feature space of $\boldsymbol{x}$, i.e., $\boldsymbol{G}(\boldsymbol{x})_{i,j} = \Phi_3(\boldsymbol{x})_i^T \Phi_3(\boldsymbol{x})_j$, and $\Phi_3(\boldsymbol{x})_i^T$ denotes the $i$-th location of the feature map.

Defining $\boldsymbol{\mathcal{D}_2}$ in the function of

$$\max_{\boldsymbol{\mathcal{D}_2}} \mathcal{L}_{\boldsymbol{\mathcal{D}_2}} = \mathbb{E}[\log(1 - \boldsymbol{\mathcal{D}_2}(\boldsymbol{x})) + \log(\boldsymbol{\mathcal{D}_2}(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}))))],$$

we further add the adversarial loss as

$$\max_{\boldsymbol{E},\boldsymbol{D}} \mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_2}}} = \mathbb{E}[\log(1 - \boldsymbol{\mathcal{D}_2}(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}))))], \quad (5)$$

where $\boldsymbol{\mathcal{D}_2}$ is an image-level discriminator that tells whether an image is real or fake. Spectral normalization [42] is applied to $\boldsymbol{\mathcal{D}_2}$ to improve the training stability.

The style loss (4) matches local statistics between the output image and the original one in the ReLU3_1 VGG layer. It is shown that this loss is useful in style transfer [43] and super-resolution [44]. It is also useful in boosting textural fidelity of our model. The adversarial loss (5) makes the generated images similar to the real ones. This also leads to improvement of visual quality.

**Semantic Knowledge Guidance** Previous work has observed that a pretrained VGG network [45] can be utilized for latent space interpolation [17], [24], [26]. We leverage this property to guide the training of our encoder. Inspired by [46], [47], we treat a pretrained VGG network as a *teacher*, and use its intermediate layer to guide training of our encoder:

$$\min_{\boldsymbol{E},\boldsymbol{P}} \mathcal{L}_{KG} = \mathbb{E}_{\mathbb{P}_r}||\boldsymbol{P}[\boldsymbol{E}(\boldsymbol{x})] - \Phi_5(\boldsymbol{x})||^2, \quad (6)$$

where $\boldsymbol{P}$ is an $1 \times 1$ convolutional layer that adapts the feature space defined by $\boldsymbol{E}(I)$ to that defined by $\Phi_5(\boldsymbol{x})$. $\Phi_5$ denotes the ReLU5_1 layer of the VGG network [45]. As the VGG network is trained with millions of images, $\Phi_5(\boldsymbol{x})$ contains rich semantic information and provides extra guidance for the encoder. Generally, this term works as regularization and helps the encoder converge to a good point.

Combining Eqs. (2), (3) and (6), the final objective function of the encoder $\boldsymbol{E}$ and decoder $\boldsymbol{D}$ is

$$\mathcal{L}_{\boldsymbol{E},\boldsymbol{D}} = \mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_1}}} + \mathcal{L}_{recon} + \mathcal{L}_{style} + \mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_2}}} + \mathcal{L}_{KG}, \quad (7)$$

where the weights of each term are set to 1.

### 3.2 Learning Interpolator

With a well-learned encoder that maps images to an easier space, interpolation can be done linearly as

$$\boldsymbol{I}(\boldsymbol{F}_i, \boldsymbol{F}_j) = \boldsymbol{F}_i + \alpha(\boldsymbol{F}_j - \boldsymbol{F}_i), \quad (8)$$

where $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$ are two real samples, and $\alpha \in [0, 1]$ is a parameter that controls the level of mixing of two samples. The second term $\alpha(\boldsymbol{F}_j - \boldsymbol{F}_i)$ can also be viewed as a *shifting vector* that points from $\boldsymbol{F}_i$ towards $\boldsymbol{F}_j$.

Note that Eq. (8) only defines one possible path that connects samples $i$ and $j$. Other interpolation methods, like Slerp [48], can also connect them and produce different intermediate results. Nevertheless, all these handcrafted

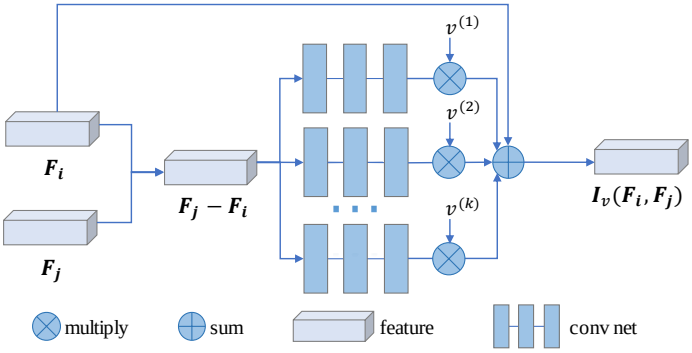



Fig. 4. Illustration of the interpolator. Each conv net branch refers to $\mathcal{T}^{(i)}(\cdot)$, and their weighted sum forms the interpolator.

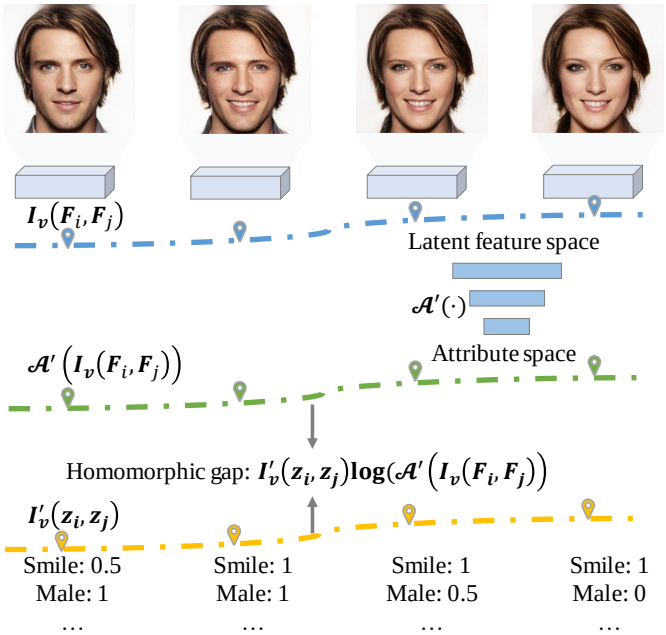



Fig. 5. Illustration of the homomorphic loss. The blue and yellow curves are the interpolation path of the *feature space* and *attribute space* respectively. The 4 cuboids (upper blue curve) and the 4 attribute vectors (lower yellow curve) illustrate sample points in the interpolation paths. Images above the cuboids visualize the decoding results of the features. To compute the homomorphic loss, we first use the attribute classifier $\mathcal{A}'(\cdot)$ to map the features to the attribute space. The predicted attributes are illustrated as the green curve. Finally, the homomorphic loss can be computed by measuring the gap between the $\mathcal{A}'(\boldsymbol{I}_v(\boldsymbol{F}_i, \boldsymbol{F}_j))$ (green curve) and $\boldsymbol{I}'_v(\boldsymbol{z}_i, \boldsymbol{z}_j)$ (yellow curve).

methods do not allow adjusting how attributes are mixed. So they are not usable for our task. To accommodate image-to-image translation, we extend $\boldsymbol{I}(\boldsymbol{F}_i, \boldsymbol{F}_j)$ to a more flexible $\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)$, where $\boldsymbol{v} \in [0, 1]^{c\times 1}$ is a *control vector*. Each dimension of $\boldsymbol{v}$ sets the interpolation strength of each grouped attribute between two samples. More specifically, the linear interpolation defined in Eq. (8) is extended to a piecewise one as

$$\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j) = \boldsymbol{F}_i + \sum_{k=1}^{c} \boldsymbol{v}^k \mathcal{T}^k(\boldsymbol{F}_j - \boldsymbol{F}_i), \tag{9}$$

where $\boldsymbol{v}^k$ is the $k^{th}$ dimension of $\boldsymbol{v}$, and $\mathcal{T}^k(\cdot)$ is a learnable mapping function represented by CNN. The architecture of interpolator is illustrated in Fig. 4.

### 3.2.1 Minimizing Homomorphic Gap

It is expected that $\mathcal{T}^k(\boldsymbol{F}_j - \boldsymbol{F}_i)$ and $\boldsymbol{v}^k$ correspond to the interpolation direction and strength of the $k^{th}$ grouped attribute $\boldsymbol{z}^k$ respectively. As $\boldsymbol{v}^k$ varies from 0 to 1, the $k^{th}$ grouped attribute changes from sample $i$ to $j$ accordingly. It is notable if all possible values of $\boldsymbol{z}$ form an *attribute space*, interpolation in the latent feature space should correspond to interpolation in the attribute space. Let $\mathcal{A}(\cdot)$ be a function that maps latent feature to an attribute vector, i.e., $\mathcal{A}(\boldsymbol{F}_i) = \boldsymbol{z}_i$. We define the relation between the latent space and the attribute space as

$$\mathcal{A}(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)) = \boldsymbol{I}'_{\boldsymbol{v}}(\mathcal{A}(\boldsymbol{F}_i), \mathcal{A}(\boldsymbol{F}_j)), \forall \boldsymbol{v} \in [0, 1]^{c\times 1} \tag{10}$$

where $\boldsymbol{I}'_{\boldsymbol{v}}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ can be viewed as an interpolation function defined in the *attribute space*. Further, $\boldsymbol{I}'_{\boldsymbol{v}}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ is defined as $\boldsymbol{I}'_{\boldsymbol{v}}(\boldsymbol{z}_i, \boldsymbol{z}_j) = [\boldsymbol{I}'_{\boldsymbol{v}}(\boldsymbol{z}_i, \boldsymbol{z}_j)^1, \cdots, \boldsymbol{I}'_{\boldsymbol{v}}(\boldsymbol{z}_i, \boldsymbol{z}_j)^c]$, where $\boldsymbol{I}'_{\boldsymbol{v}}(\boldsymbol{z}_i, \boldsymbol{z}_j)^k = \boldsymbol{z}_i^k + \boldsymbol{v}^k(\boldsymbol{z}_j^k - \boldsymbol{z}_i^k)$. The left-hand side of Eq. (10) denotes the attribute values of interpolated samples $\boldsymbol{I}_{\boldsymbol{v}}(F_i, F_j)$, where the other side contains the corresponding attribute values of the two samples. Since both sides are conditioned on the same control vector $\boldsymbol{v}$, they are expected to be equal. In this regard, Eq. (10) describes an ideal case that interpolation operations $\boldsymbol{I}_{\boldsymbol{v}}$ and $\boldsymbol{I}'_{\boldsymbol{v}}$ share the same structure in the latent feature and attribute space. This property looks similar to *homomorphism* in algebra. In practice, there inevitably exists a gap between two sides of Eq. (10), which we call *homomorphic gap*.

With Eq. (10) introduced, our objective turns to minimizing the homomorphic gap. Recall that $\mathcal{A}(\cdot)$ maps the latent feature to attribute values, which is not defined for interpolated features. We choose to train a network $\mathcal{A}'(\cdot)$ to approximate $\mathcal{A}(\cdot)$ and replace $\mathcal{A}(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j))$ with $\mathcal{A}'(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j))$ in Eq. (10). Then we reduce the homomorphic gap by minimizing the binary cross-entropy loss of $\boldsymbol{I}'_{\boldsymbol{v}}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ and $\mathcal{A}'(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j))$ as Eq. (11). Each attribute is counted separately. We call this loss the *Homomorphic loss*. The idea is also illustrated in Fig. 5.

$$\min_{\boldsymbol{I}_{\boldsymbol{v}}} \mathcal{L}_{\boldsymbol{I}_{hom}} = \mathbb{E}[-\boldsymbol{I}'_{\boldsymbol{v}}(\boldsymbol{z}_i, \boldsymbol{z}_j) \log(\mathcal{A}'(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)))]. \tag{11}$$

Note that $\boldsymbol{v}$ is defined everywhere in the $c$-dimensional unit hypercube. During training, we assign uniformly random values to $\boldsymbol{v}$ to cover the whole feasible set.

### 3.2.2 Rigorous Training

According to Eq. (9), optimizing Eq. (11) needs to optimize $\mathcal{T}^k(\cdot)$, where $k = 1, \cdots, c$. In our experiments, when complicated attributes exist, the corresponding $\mathcal{T}^k(\cdot)$ tends to be lazy – that is, it may update $\boldsymbol{F}_i$ slightly to fool the attribute classification network $\mathcal{A}'(\cdot)$. To alleviate this problem, we turn $\mathcal{A}'(\cdot)$ to a rigorous classifier: instead of mapping $F_i$ to $\boldsymbol{z}_i$, $\mathcal{A}'(\cdot)$ is trained to map the interpolated feature $\boldsymbol{F}_i + \sum_{k=1}^{c} \boldsymbol{v}^k \mathcal{T}^k(\boldsymbol{F}_j - \boldsymbol{F}_i)$ to attribute $\boldsymbol{z}_i$, expressed as

$$\min_{\mathcal{A}'} \mathcal{L}_{\mathcal{A}'} = \mathbb{E}[-\boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)))]. \tag{12}$$

From Eqs. (11) and (12), we note that $\boldsymbol{I}_{\boldsymbol{v}}(\cdot)$ and $\mathcal{A}'(\cdot)$ are mutually dependent. Therefore, they are iteratively updated during training. In this way, $\mathcal{A}'(\cdot)$ keeps checking unchanged parts, making it harder for $\mathcal{T}^k(\cdot)$ to fool it.

Note that this rigorous training strategy is applied to all attributes. For more analysis and experimental validation, please refer to the supplementary material.

#### 3.2.3 Handling Residual Components

When $\boldsymbol{v} = \mathbf{1}$, where $\mathbf{1} = [1, 1, \cdots, 1] \in \mathbb{R}^{c\times 1}$, $\boldsymbol{I_v}(\boldsymbol{F}_i, \boldsymbol{F}_j)$ is expected to reach sample $j$. However, this is not guaranteed with solely the homomorphic loss, because the provided attributes may not explain all information. Therefore, we extend Eq. (9) to

$$\boldsymbol{I_v}(\boldsymbol{F}_i, \boldsymbol{F}_j) = \boldsymbol{F}_i + \sum_{k=1}^{c+1} \boldsymbol{v}^k \boldsymbol{\mathcal{T}}^k(\boldsymbol{F}_j - \boldsymbol{F}_i), \tag{13}$$

where the additional mapping function $\boldsymbol{\mathcal{T}}^{c+1}(\boldsymbol{F}_j - \boldsymbol{F}_i)$ models the residual components that are not explained by the given attributes. Accordingly, we extend the $c$-dim control vector $\boldsymbol{v}$ to $c+1$ dim, where the last dimension is the edit strength of the residual mapping function. Now we can safely impose the terminal of the interpolation curve as $\boldsymbol{F}_j$, which is formulated as

$$\mathcal{L}_{\boldsymbol{I_1}} = ||\boldsymbol{I_v}(\boldsymbol{F}_i, \boldsymbol{F}_j) - \boldsymbol{F}_j||^2, \text{where } \boldsymbol{v} = \mathbf{1}. \tag{14}$$

To summarize this part, the overall loss function of $\boldsymbol{I_v}$ is

$$\mathcal{L}_{\boldsymbol{I_v}} = \mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_1}}} + \mathcal{L}_{\boldsymbol{I}_{hom}} + \lambda_{\boldsymbol{I_1}} \mathcal{L}_{\boldsymbol{I_1}}, \tag{15}$$

where $\mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_1}}}$, $\mathcal{L}_{\boldsymbol{I}_{hom}}$, and $\mathcal{L}_{\boldsymbol{I_1}}$ are defined in Eqs. (2), (11) and (14) respectively. $\lambda_{\boldsymbol{I_1}}$ is set to 10 in our experiments.

### 3.3 Applications

We describe how our model can be applied to multi-domain, multi-modal and continuous translation as follows.

#### 3.3.1 Multi-domain Translation

For each target domain $t$, we preselect a sample $I_t$ as exemplar. Given a query sample $I_q$, domain translation is conducted as

$$\boldsymbol{x}_{out} = \boldsymbol{D}(\boldsymbol{I}_{\boldsymbol{v}_t}(\boldsymbol{E}(\boldsymbol{x}_q), \boldsymbol{E}(\boldsymbol{x}_t))), \tag{16}$$

where $\boldsymbol{v}_t$ is the vector corresponding to the target domain.

#### 3.3.2 Multi-Modal Translation

Using different exemples in Eq. (16), we can generate results like MUNIT [37].

#### 3.3.3 Continuous Translation

By changing $\boldsymbol{v}_t$ in Eq. (16) smoothly, our model allows changing attributes continuously. This controls the edit strength or generates animation along the translation process.

Fig. 8 illustrates how changing $\boldsymbol{v}$ and exemplar affects the results, leading to multi-domain and multi-modal translation. Fig. 1 illustrates the continuous translation.

### 3.4 Implementation Details

| Layer Type | Norm | Activation | Input Size | Output Size |
|---|---|---|---|---|
| Conv(4,2,1) | IN | LReLU | $H \times W \times C$ | $\frac{H}{2} \times \frac{W}{2} \times 2C$ |
| Conv(3,1,1) | IN | LReLU | $\frac{H}{2} \times \frac{W}{2} \times 2C$ | $\frac{H}{2} \times \frac{W}{2} \times 2C$ |
| Conv(3,1,1) | IN | LReLU | $\frac{H}{2} \times \frac{W}{2} \times 2C$ | $\frac{H}{2} \times \frac{W}{2} \times 2C$ |

(a) DownBlock

| Layer Type | Norm | Activation | Input Size | Output Size |
|---|---|---|---|---|
| Deconv(4,2,1) | IN | LReLU | $H \times W \times C$ | $2H \times 2W \times \frac{C}{2}$ |
| Conv(3,1,1) | IN | LReLU | $2H \times 2W \times \frac{C}{2}$ | $2H \times 2W \times \frac{C}{2}$ |
| Conv(3,1,1) | IN | LReLU | $2H \times 2W \times \frac{C}{2}$ | $2H \times 2W \times \frac{C}{2}$ |

(b) UpBlock

| Layer Type | Norm | Activation | Input Size | Output Size |
|---|---|---|---|---|
| Conv(3,1,1) | IN | LReLU | $256 \times 256 \times 3$ | $256 \times 256 \times 16$ |
| DownBlock | IN | LReLU | $256 \times 256 \times 16$ | $128 \times 128 \times 32$ |
| DownBlock | IN | LReLU | $128 \times 128 \times 32$ | $64 \times 64 \times 64$ |
| DownBlock | IN | LReLU | $64 \times 64 \times 64$ | $32 \times 32 \times 128$ |
| DownBlock | IN | LReLU | $32 \times 32 \times 128$ | $16 \times 16 \times 256$ |
| DownBlock | IN | LReLU | $16 \times 16 \times 256$ | $8 \times 8 \times 512$ |
| DownBlock | IN | LReLU | $8 \times 8 \times 512$ | $4 \times 4 \times 1024$ |

(c) Encoder $\boldsymbol{E}$

| Layer Type | Norm | Activation | Input Size | Output Size |
|---|---|---|---|---|
| Conv(3,1,1) | - | LReLU | $4 \times 4 \times 1024$ | $4 \times 4 \times 1024$ |
| Conv(3,1,1) | - | LReLU | $4 \times 4 \times 1024$ | $4 \times 4 \times 1024$ |
| Conv(3,1,1) | - | - | $4 \times 4 \times 1024$ | $4 \times 4 \times 1024$ |

(d) One Branch of Interpolator $\boldsymbol{\mathcal{T}}^i$

| Layer Type | Norm | Activation | Input Size | Output Size |
|---|---|---|---|---|
| UpBlock | IN | LReLU | $4 \times 4 \times 1024$ | $8 \times 8 \times 512$ |
| UpBlock | IN | LReLU | $8 \times 8 \times 512$ | $16 \times 16 \times 256$ |
| UpBlock | IN | LReLU | $16 \times 16 \times 256$ | $32 \times 32 \times 128$ |
| UpBlock | IN | LReLU | $32 \times 32 \times 128$ | $64 \times 64 \times 64$ |
| UpBlock | IN | LReLU | $64 \times 64 \times 64$ | $128 \times 128 \times 32$ |
| UpBlock | IN | LReLU | $128 \times 128 \times 32$ | $256 \times 256 \times 16$ |
| Conv(3,1,1) | - | - | $256 \times 256 \times 16$ | $256 \times 256 \times 3$ |

(e) Decoder $\boldsymbol{D}$

| | | | | |
|---|---|---|---|---|
| Conv(1,1,1) | IN | LReLU | $4 \times 4 \times 1024$ | $4 \times 4 \times 256$ |
| Conv(4,2,1) | IN | LReLU | $4 \times 4 \times 512$ | $2 \times 2 \times 512$ |
| Conv(1,1,1) | IN | LReLU | $2 \times 2 \times 512$ | $1 \times 1 \times 512$ |
| FC($\boldsymbol{\mathcal{D}_1}$) | - | - | $1 \times 1 \times 512$ | $1 \times 1 \times 512$ |
| FC($\boldsymbol{\mathcal{A}'}$) | - | - | $1 \times 1 \times 512$ | $1 \times 1 \times c$ |

(f) Feature-level Discriminator $\boldsymbol{\mathcal{D}_1}$ and Attribute Predictor $\boldsymbol{\mathcal{A}'}$

| | | | | |
|---|---|---|---|---|
| Conv(4,2,1) | IN | LReLU | $256 \times 256 \times 3$ | $128 \times 128 \times 64$ |
| Conv(4,2,1) | IN | LReLU | $128 \times 128 \times 64$ | $64 \times 64 \times 128$ |
| Conv(4,2,1) | IN | LReLU | $64 \times 64 \times 128$ | $32 \times 32 \times 256$ |
| Conv(4,2,1) | IN | LReLU | $32 \times 32 \times 256$ | $16 \times 16 \times 512$ |

(g) Image-level Discriminator $\boldsymbol{\mathcal{D}_2}$

TABLE 1
Architecture of the Homomorphic Interpolation Network. It contains an encoder, an interpolator and a decoder. Conv(3,1,1) denotes the convolutional layer, with the $1^{st} - 3^{rd}$ parameters denoting kernel size, stride and padding respectively. (a) and (b) define two basic blocks of DownBlock and UpBlock. The encoder and decoder are built upon these blocks. (c-g) are the architectures of our model.

#### 3.4.1 Network Architecture

Table 1 shows the architecture of our model. We first define two basic blocks – the down-sampling block (DownBlock) and up-sampling block (UpBlock) in Table 1(a) and Table 1(b). Table 1(c) and Table 1(e) list the architecture of encoder and decoder, which is defined based on the basic blocks. Table 1(d) shows one branch of the interpolator.

In Table 1(f), the feature-level discriminator $\boldsymbol{\mathcal{D}_1}$ and the attribute predictor $\boldsymbol{\mathcal{A}'}$ share convolutional layers except for the fully-connected layers. FC($\boldsymbol{\mathcal{D}_1}$) and FC($\boldsymbol{\mathcal{A}'}$) are fully-connected layers of $\boldsymbol{\mathcal{D}_1}$ and $\boldsymbol{\mathcal{A}'}$ respectively. Note that the output channel of discriminator $\boldsymbol{\mathcal{D}_1}$ is 512. In our experiments, it leads to more stable training than setting the channel number to 1.

Table 1(g) shows the architecture of image-level discriminator $\boldsymbol{\mathcal{D}_2}$. It is in patch-level, and thus mainly focuses on local regions. Spectral normalization [42] is applied to $\boldsymbol{\mathcal{D}_2}$ to stabilize the training.

#### 3.4.2 Training Details

The training procedure is outlined in Algorithm 1. We use the Adam optimizer [49] to train our model, with $\beta_1$ and $\beta_2$ set to $0.5$ and $0.999$ respectively. We set the batch size as $64$. Following TTUR [50], the learning rate for discriminators $\boldsymbol{\mathcal{D}_1}$ and $\boldsymbol{\mathcal{D}_2}$ is set as $1 \times 10^{-5}$. For other components it is $1 \times 10^{-4}$. Before training the whole model as Algorithm 1, we found it benefitial to pretrain the encoder $\boldsymbol{E}$ and decoder $\boldsymbol{D}$ with $\mathcal{L}_{recon}+\mathcal{L}_{style}+\mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_2}}}+\mathcal{L}_{KG}$. After pretraining, the encoder and decoder can well reconstruct training samples, making it easier to achieve high-quality results in the whole-model training.

| Dim | Attribute | Labels |
|---|---|---|
| 1 | Age | Young |
| 2 | Expression | Mouth_Slightly_Open, Smiling |
| 3 | Hair Color | Black_Hair, Blond_Hair Brown_Hair, Gray_Hair |
| 4 | Hair Style | Bald, Receding_Hairline, Bangs |
| 5 | Gender Trait | Male, No_Beard, Mustache, Goatee, Sideburns |

TABLE 2
Grouped Attributes of CelebA-HQ [22]. The 1st-3rd columns: dimension index in the control vector $\boldsymbol{v}$, name of grouped attributes, corresponding attribute labels.

| Dim | Attribute | Labels |
|---|---|---|
| 1 | Expression | happy, angry, contemptuous, sad, disgusted, neutral, fearful, surprised |
| 2 | Gaze | look left, look front, look right |
| 3 | ID Traits | is_Caucasian, is_male, is_kid |

TABLE 3
Grouped attributes of RaFD [40].

## 4 Experimental Results

### 4.1 Datasets

Our experiments are conducted on CelebA-HQ [22] and RaFD [40]. CelebA-HQ contains 30k celebrity images, each with 40 attribute labels. We define grouped attributes based on these labels as shown in Table 2. We randomly use 27k images for training and 3k for testing. RaFD [40] is a smaller dataset that contains 67 identities, each displaying 8 emotional expressions, 3 eye locations, and 3 other attributes about the identities. Similarly, we group these labels into 3 higher-level attributes as shown in Table 3. In our experiments, we use 65 identities for training and the other two for testing. All images are center cropped and resized to $256 \times 256$.

### 4.2 Ablation Analysis

#### 4.2.1 Pivotal Parts in Training

It should be noted that all loss functions, including $\mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_1}}}$ (Eq. (2)), $\mathcal{L}_{style}$ (Eq. (4)), $\mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_2}}}$ (Eq. (5)) $\mathcal{L}_{KG}$ (Eq. (6)), and $\mathcal{L}_{Hom}$ (Eq. (11)), play an important role in our model. Without any of them, training may converge to an inferior point, leading to unsatisfactory results. To illustrate this, we disable each part and compare it with our final result in Fig. 6. As shown in Fig. 6(c), without the homomorphic loss $\mathcal{L}_{Hom}$ (Eq. (11)), the generated images cannot transfer the target attributes from the reference images. The adversarial loss in pixel space $\mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_2}}}$ (Eq. (5)) is related to details of the generated images. As shown in Fig. 6(d), without this term, the generated images lack details and look blurry. The adversarial loss in feature space $\mathcal{L}_{GAN_{\boldsymbol{\mathcal{D}_1}}}$ (Eq. (2)) and the knowledge guidance regularization $\mathcal{L}_{KG}$ (Eq. (6)) are also related to the quality of generated results. Without each of them, the encoder may not learn a smooth enough latent space. As shown in Fig. 6(e-f), it could cause visual artifacts in eyes and hair regions. Disabling the style loss $\mathcal{L}_{style}$ (Eq. (4)) causes relatively smaller quality degradation, which is shown in Fig. 6(g).

**Algorithm 1** Training our model

**Input:** $\boldsymbol{x}_i$ and $\boldsymbol{z}_i$, where $i = 1, 2, \cdots, N$
**Output:** encoder $\boldsymbol{E}$, interpolator $\boldsymbol{I_v}$ and decoder $\boldsymbol{D}$
**while** not converged **do**
  get a new batch from the dataset;
  sample $\boldsymbol{v}$ from $c$ dimensional uniform distribution;
  $t \leftarrow 0$;
  update the feature-level discriminator $\boldsymbol{\mathcal{D}_1}$ based on Eq. (1);
  update the attribute classifier $\boldsymbol{\mathcal{A}'}$ based on Eq. (12);
  **if** $t$ mod $5 == 0$ **then**
    update the image-level discriminator $\boldsymbol{\mathcal{D}_2}$ based on Eq. (5);
    update $\boldsymbol{P}$, encoder $\boldsymbol{E}$ and decoder $\boldsymbol{D}$ based on Eq. (7);
    Update $\boldsymbol{I_v}$ based on Eq. (15).
  **end if**
**end while**

**Quantitative Analysis** To quantitatively evaluate the effectiveness of each loss, we introduce the translation accuracy (ACC) and the Fréchet Inception Distance (FID score) [50] for quantitatively measuring the effectiveness of our approach.

Translation accuracy measures the domain label correctness of the edited images. More specifically, it is defined as the percentage that the edited images have identical target attribute with the reference image, i.e.,

$$ACC = \frac{\sum_{i=1}^{N} \delta[\mathcal{C}(\boldsymbol{x}_i^{out}) == \boldsymbol{y}_i^{ref}]}{N}, \tag{17}$$

where $\boldsymbol{x}_i^{out}$ is the $i$-th edited image, $\boldsymbol{y}_i^{ref}$ is the target attribute (e.g., Gender) of its reference image, $\mathcal{C}(\cdot)$ is an attribute classifier that estimates the target attribute of $\boldsymbol{x}_i^{out}$, and $\delta(\cdot)$ is a function that outputs 1 if the estimated target attribute agrees with that of the reference image $\boldsymbol{y}_i^{ref}$, and 0 otherwise.

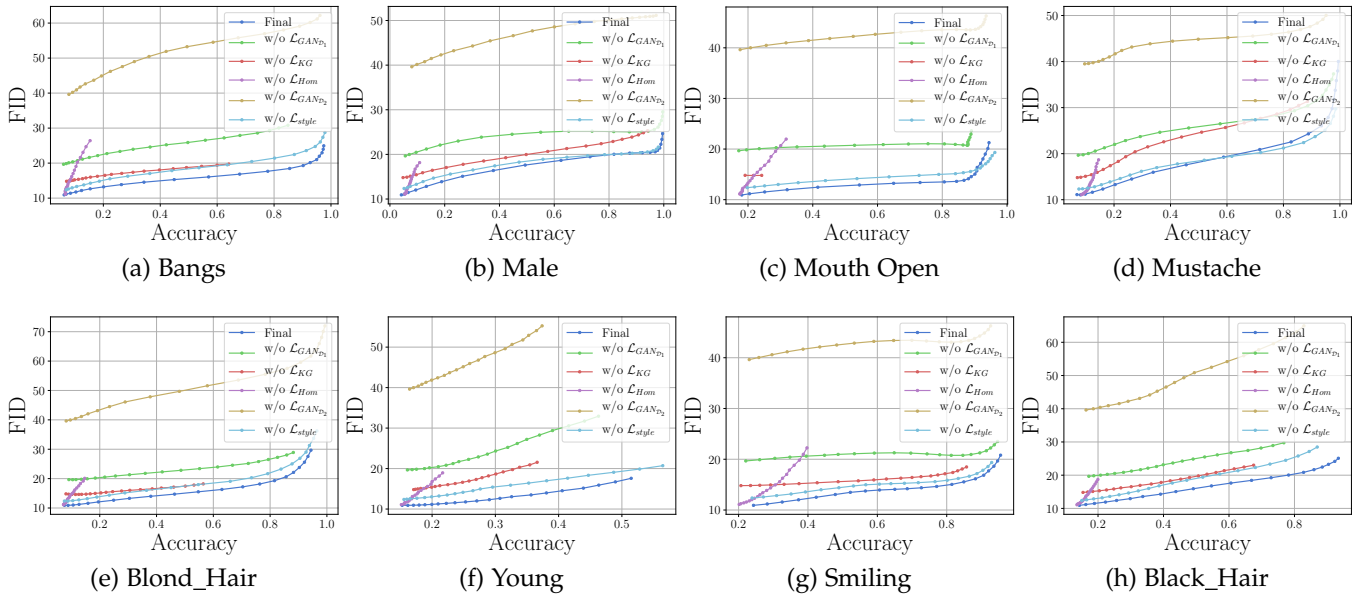
FID score measures mean and covariance difference between generated and real images in a deep feature space (typically the final average pooling layer of Inception_V3). It is defined as

$$FID = ||\boldsymbol{m}_{out} - \boldsymbol{m}||_2^2 + \mathrm{Tr}(\boldsymbol{C}_{out} + \boldsymbol{C} - 2(\boldsymbol{C}_{out}\boldsymbol{C})^{\frac{1}{2}})), \tag{18}$$

where $\boldsymbol{m}_{out}$, $\boldsymbol{C}_{out}$, $\boldsymbol{m}$ and $\boldsymbol{C}$ are the mean and covariance of generated and real images respectively. As shown in [50],


(a) Reference (b) Original (c) w/o $\mathcal{L}_{Hom}$ (d) w/o $\mathcal{L}_{\mathcal{D}_2}$ (e) w/o $\mathcal{L}_{\mathcal{D}_1}$ (f) w/o $\mathcal{L}_{KG}$ (g) w/o $\mathcal{L}_{style}$ (h) Final

Fig. 6. Effectiveness of each term of the loss functions. (a) is the reference image. (b) is the original image. (c-g) are the results without each term. (h) is our final result.




(a) Bangs (b) Male (c) Mouth Open (d) Mustache

(e) Blond_Hair (f) Young (g) Smiling (h) Black_Hair

Fig. 7. Quantitative results of each terms of our model. The x- and y-axis are the translation accuracy (the larger the better) and FID score (the smaller the better). "$w/o\ \mathcal{L}_{GAN_{\mathcal{D}_1}}$", "$w/o\ \mathcal{L}_{style}$", "$w/o\ \mathcal{L}_{GAN_{\mathcal{D}_2}}$", "$w/o\ \mathcal{L}_{KG}$", and "$w/o\ \mathcal{L}_{Hom}$" are results by removing $\mathcal{L}_{GAN_{\mathcal{D}_1}}$ (Eq. (2)), $\mathcal{L}_{style}$ (Eq. (4)), $\mathcal{L}_{GAN_{\mathcal{D}_2}}$ (Eq. (5)) $\mathcal{L}_{KG}$ (Eq. (6)), and $\mathcal{L}_{Hom}$ (Eq. (11)) respectively. Each curve is plotted with different edit strength.

(a) Ref (b) Original (c) Expression (d) Gender (e) Hair Color (f) Ref (g) Original (h) Expression (i) Gaze (j) ID traits

Fig. 8. Illustration of the roles of control vector and exemplar. (a-e) and (f-j) are the results of CelebA-HQ and RaFD datasets respectively. (a) and (f) are the reference images. (b) and (g) are the original images. (c-e) and (h-j) are results of setting different control vectors $\boldsymbol{v}$. Rows 1 and 2 are results of using different reference images.

FID score is sensitive to various variations, such as noise, blurring, and swirling. It can be used for measuring the degree of realism of our generated images.

Note that our model allows changing the edit strength. Larger strength makes the change more significant, and thus leads to higher translation accuracy. On the other hand, artifacts may also be amplified when edit strength is large. In this regard, we compute the translation accuracy and FID under different edit strength, varying from 0 to 2, and plot the "FID vs ACC" curves as shown in Fig. 7.

Without the homomorphic loss $\mathcal{L}_{Hom}$ (Eq. (11)), the translation accuracy remains small even if the edit strength

is large. Without the adversarial loss in pixel space $\mathcal{L}_{GAN_{\mathcal{D}_2}}$ (Eq. (5)), the FID score is much higher than the final model. Disabling other loss functions also raises the FID score when the translation accuracy is the same. This demonstrates the effectiveness of these loss functions.

#### 4.2.2 Pivotal Parts in Testing

During testing, the control vector $\boldsymbol{v}$ and the reference exemplars are important to control the translated results of our model. The control vector determines which attribute to alter, while the exemplars determine how attribute translation is instantiated. Using both of them, we flexibly control the generated results. This is illustrated in Fig. 8, where (a-e) and (f-j) are the results of CelebA-HQ [22] and RaFD [40] respectively.

For results of each dataset, each row shows how results are changed with the same exemplars and different control vectors, while each column gives results with the same control vector and yet different exemplars. As shown in Fig. 8(c-e), by setting $\boldsymbol{v}$ to the one-hot vectors representing the expression, gender, and hair color respectively, we effectively vary corresponding attributes. In Fig. 8(h-j), by choosing specific control vectors, we alter the expression, gaze and identity traits of the man in RaFD [40] dataset respectively. The exemplars also affect the final results. For example, in Fig. 8(c-e), results in the 1st and 2nd rows have quite different gender, expression, hair color change; while in Fig. 8(h-j), the difference in expression, gaze and identity traits is also significant.

### 4.3 Comparison with Other Methods

One of the largest advantages of our model is the ability to handle multi-domain, multi-modal and continuous image-to-image translation simultaneously. In this section, we provide both qualitative and quantitative comparison with other methods.

#### 4.3.1 Qualitative Evaluation

**Multi-Domain Translation** Our model deals with multi-domain image-to-image translation with Eq. (16). Figs. 9 and 10 compare our results with two related methods, i.e., StarGAN [4] and AttGAN [34]. StarGAN [4] takes domain labels as input to generator, and produces target domain results. AttGAN [34] uses an encoder to encode images to domain invariant latent feature, and the decoder generates images based on the latent feature and domain labels. Visually, our model accomplishes more natural – and with significant changes – results than AttGAN [34] and StarGAN [4]. AttGAN [34] extracts latent feature that can work cooperatively with domain labels. When the latent feature contains domain information, the translated results may not be sufficiently strong. StarGAN [4] works well, and yet still occasionally produces unexpected edit, leading to visual artifacts.

**Multi-Modal Translation** Using different exemplars, our model can produce multiple results for image-to-image translation. Fig. 11 compares our approach with two multi-modal translation methods, i.e., MUNIT [37] and GDWCT [36]. GDWCT [36] decomposes images into domain-invariant content space and domain-specific style space. Then it applies group-wise deep whitening-and-coloring transformation to image translation from one domain to another based on exemplars. In addition to the target attributes, it tends to change the color and brightness of the images largely, leading to unnatural results. Compared with our method, MUNIT [37] does not leverage information of multiple domains. When skin, hair color and background are wrongly edited, as shown in Fig. 11, the result quality reduces.

**Continuous Translation** With the well learned latent space, our model allows synthesizing images across different domains. This has already been shown in Figs. 1 and 2. Despite this, we also note that a good latent space should uncover the structure of natural image manifold [18]. To an extreme, it should even gain the capacity of *extrapolation*. This allows exaggerating the difference between two domains. Fig. 12 compares the interpolation/extrapolation capacity of our model with Facelet [24], AttGAN [34] and Fader Networks [29].

Facelet [24] is a feature interpolation approach whose latent feature is defined by a pretrained VGG network. Similar to ours, it requires only discrete attribute labels and has the capability to translate between different domains smoothly. However, when applying very strong edit strength, the result quality could drop. Similarly, when applying strong edit strength to AttGAN [34], much artifact emerges, while the target attribute does not exaggerate. Fader Networks disentangle the attribute-invariant feature of an input image, and generate a new one by combining it with other attributes. It does not explicitly constrain the smoothness of the latent space, and the interpolation/extrapolation results are not as good as ours. In contrast, our model works consistently well in both situations of interpolation and extrapolation. This indicates that the encoder trained by Eq. (7) actually unfolds the natural image manifold, leading to a flat and smooth latent space that allows interpolation and even extrapolation.

#### 4.3.2 User Study

We have also conducted user study on the Amazon Mechanical Turk platform to compare our performance with others. Turing Test and A/B Test are conducted.

**Turing Test** Each time subjects are presented with an arbitrary real image and the other that is edited by one method. They are requested to pick the real one. Table 4 shows the percentage that an edited image is regarded as real. Note that different attributes are counted separately, each including 1,000 comparisons. Higher values mean that human is harder to distinguish between the real image and edited one. The final statistics show that our model has $43\%$ chance to fool human eyes, which outperforms StarGAN [4] (31%), AttGAN [34] (30%), Facelet [24] (28%), GDWCT [36] (32%) and MUNIT [37] (13%).

**A/B Test** A/B Test refers to the pair-wise comparison of our model and another baseline model. Each time subjects are given an original image and two edited ones (our method vs. another), and are asked to pick the one with higher

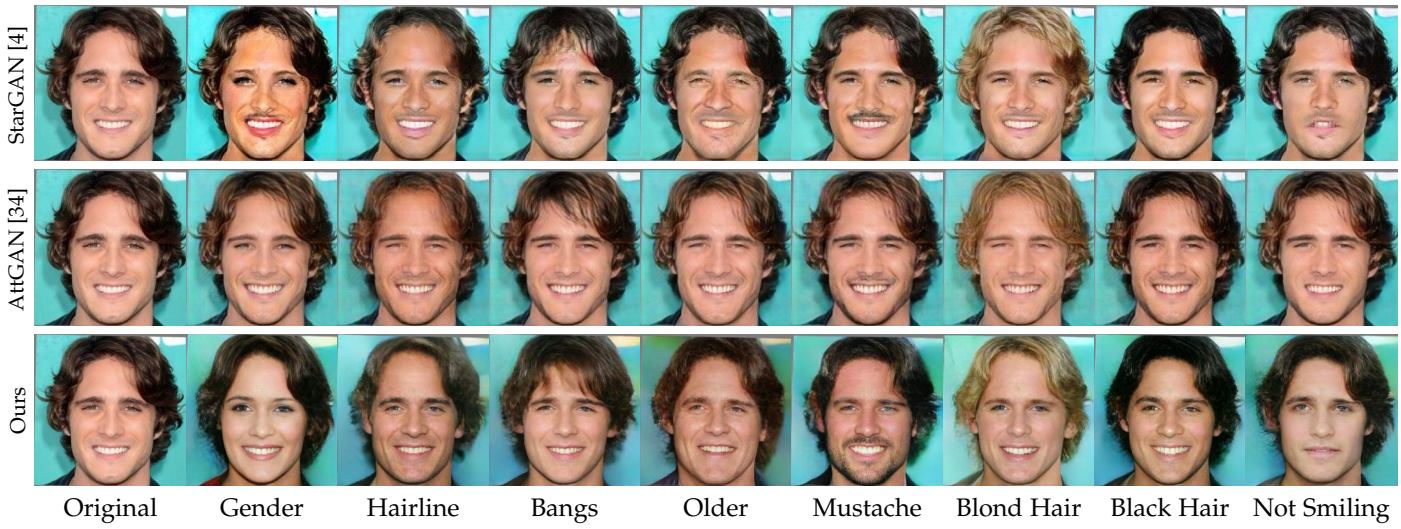



Fig. 9. Multi-domain image-to-image translation on CelebA-HQ [22].

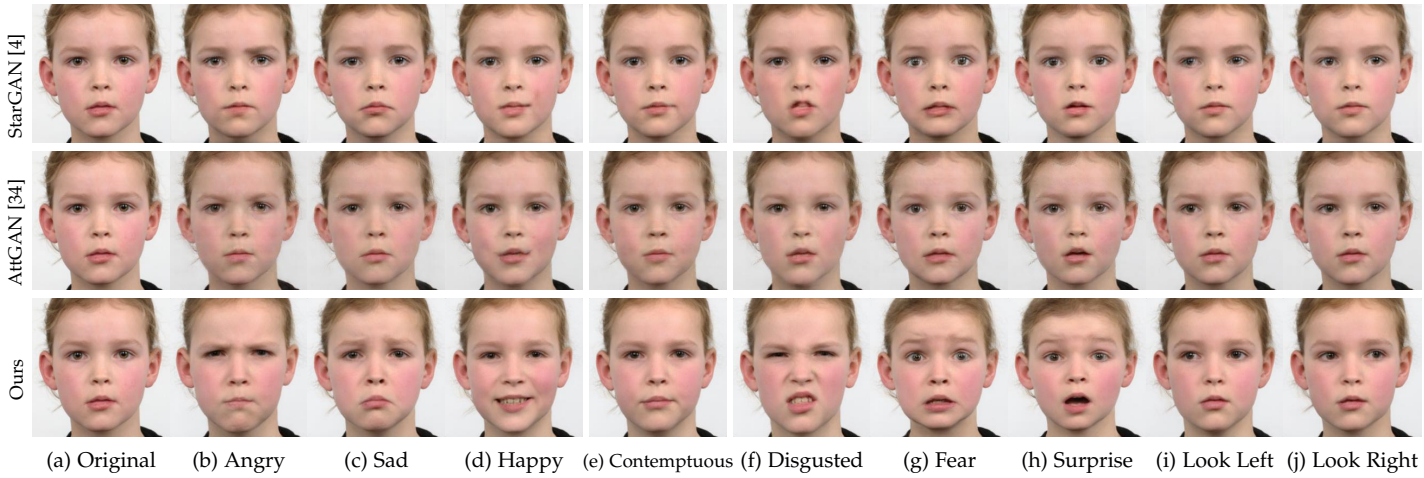



Fig. 10. Multi-domain image-to-image translation on RaFD [40].

| | Young | Male | Smiling | Black_Hair | Bangs | Mustache | Hairline | Mouth Open | Total |
|---|---|---|---|---|---|---|---|---|---|
| MUNIT [37] | 9% | 12% | 22% | 12% | 11% | 8% | 15% | 17% | 13% |
| GDWCT [36] | 20% | 18% | 25% | 22% | 19% | 18% | 31% | 27% | 23% |
| Facelet [24] | 27% | 32% | 33% | 28% | 29% | 44% | 10% | 18% | 28% |
| AttGAN [34] | 29% | 30% | 38% | 30% | 27% | 35% | 19% | 33% | 30% |
| StarGAN [4] | 29% | 24% | 36% | 24% | 31% | 45% | 23% | 33% | 31% |
| Ours | 45% | 36% | 48% | 45% | 38% | 47% | 42% | 44% | 43% |

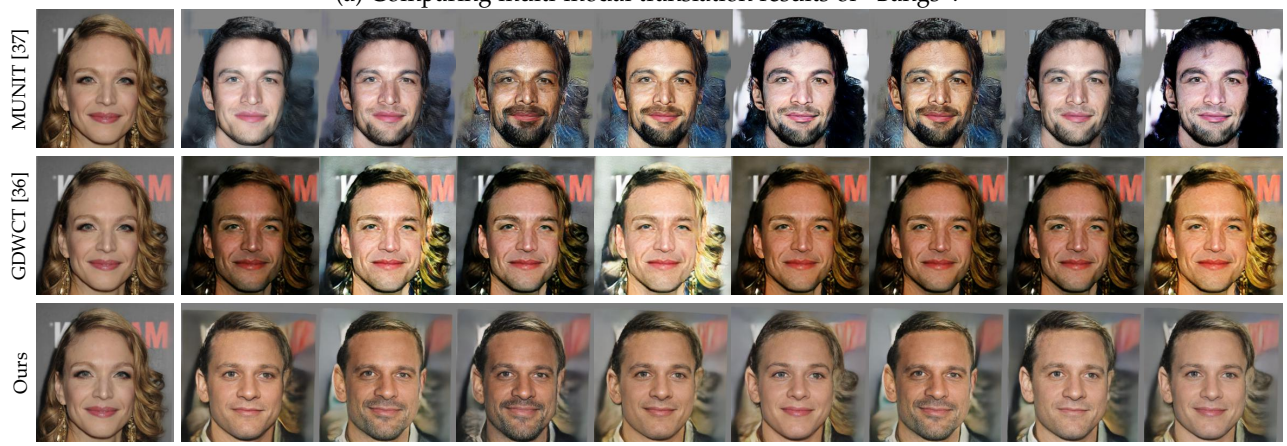
TABLE 4
Turing Test on CelebA-HQ [22] dataset. Each entry reports the percentage of taking the edited image as real (the higher the better).

| | Young | Male | Smiling | Bangs | Black_Hair | Hairline | Mustache | Mouth Open | Total |
|---|---|---|---|---|---|---|---|---|---|
| Ours > MUNIT [37] | 80% | 75% | 73% | 69% | 68% | 67% | 75% | 73% | 73% |
| Ours > GDWCT [36] | 70% | 60% | 63% | 64% | 70% | 52% | 57% | 65% | 63% |
| Ours > Facelet [24] | 67% | 74% | 75% | 62% | 81% | 62% | 55% | 78% | 69% |
| Ours > AttGAN [34] | 65% | 58% | 62% | 60% | 56% | 51% | 53% | 69% | 59% |
| Ours > StarGAN | 73% | 60% | 71% | 58% | 52% | 48% | 57% | 72% | 61% |

TABLE 5
A/B Test on CelebA-HQ [22] dataset. Each entry reports the percentage that our results are preferred. Larger than $50\%$ indicates that our method is statically more preferred by the subjects.

edit quality. Similar to the Turing Test, different attributes are separately counted, each including 1,000 comparisons. Table 5 presents the percentage that images generated by our method are chosen. Overall, our method outperforms StarGAN [4], AttGAN [34], Facelet [24], GDWCT [36] and MUNIT [37] by 61%, 59%, 69%, 64 % and 73 % respectively.


(a) Comparing multi-modal translation results of "Bangs".




(b) Comparing multi-modal translation results of "Gender".

Fig. 11. Multi-modal image-to-image translation. The $1^{st}$ column is the original image, and other columns are different output.

## 5 Applicability Analysis

### 5.1 Failure Cases

Our model relies on the assumption that images of different domains can be embedded in a smooth and flat space. This is hardly achieved when these domains are very different. Fig. 13 illustrates a case that performs translation between facade images and semantic labels. Our model does not perform well in this case, since it is very difficult to find intermediate regions in between.

### 5.2 Discussion of the Scope of Application

To further understand the scope of application of our approach, we also propose a measurement to quantify if our model is applicable.

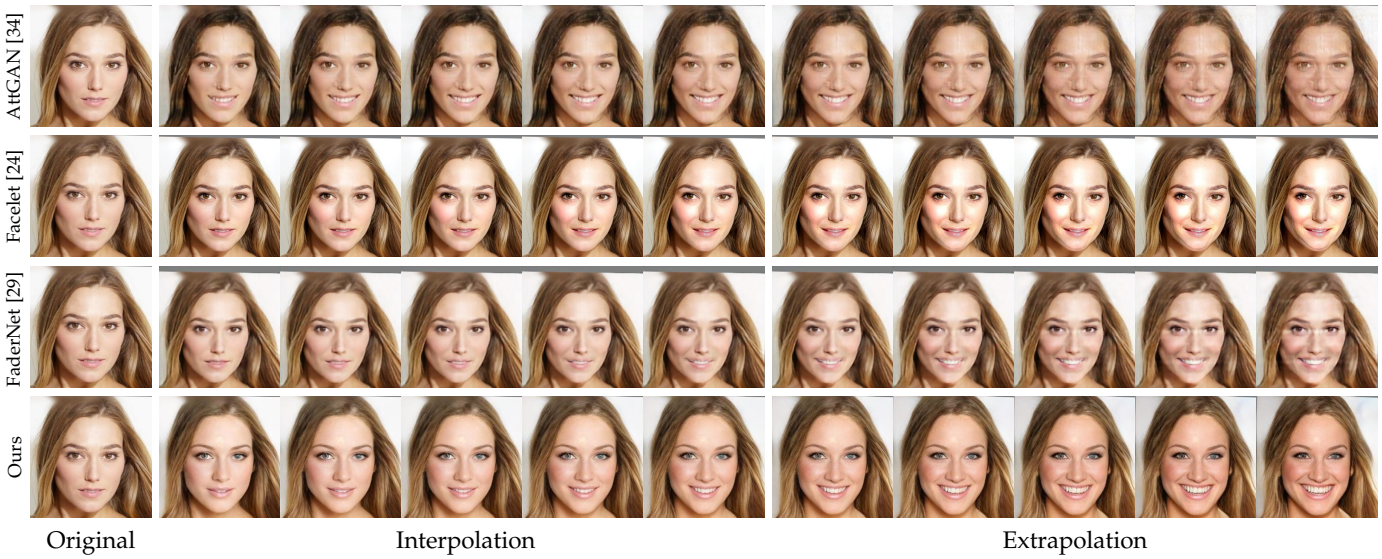
Intuitively, learning an interpolation model would be easier when samples of two domains locate nearby, and there exist samples in-between. Typical tasks include age progression, expression manipulation, hair color change, etc. For these tasks, images of opposite attributes (e.g., young vs. old) are not absolutely different. Instead, there must be ambiguous samples that lie in-between (e.g., neither too young or too old). These ambiguous examples provide guidance to model intermediate regions between different domains, making learning easier.

In contrast, if no or very few samples are between two domains, the intermediate regions are not well defined. Our model is not expected to work well in this case. When training such tasks (e.g., Label vs Facade in Fig. 13), the feature-level discriminator $\mathcal{D}_1$ is optimized quickly, which confidently distinguish between real and interpolated features. As a result, it suffers from the gradient vanishing problem and cannot provide sufficient information to train the interpolator.

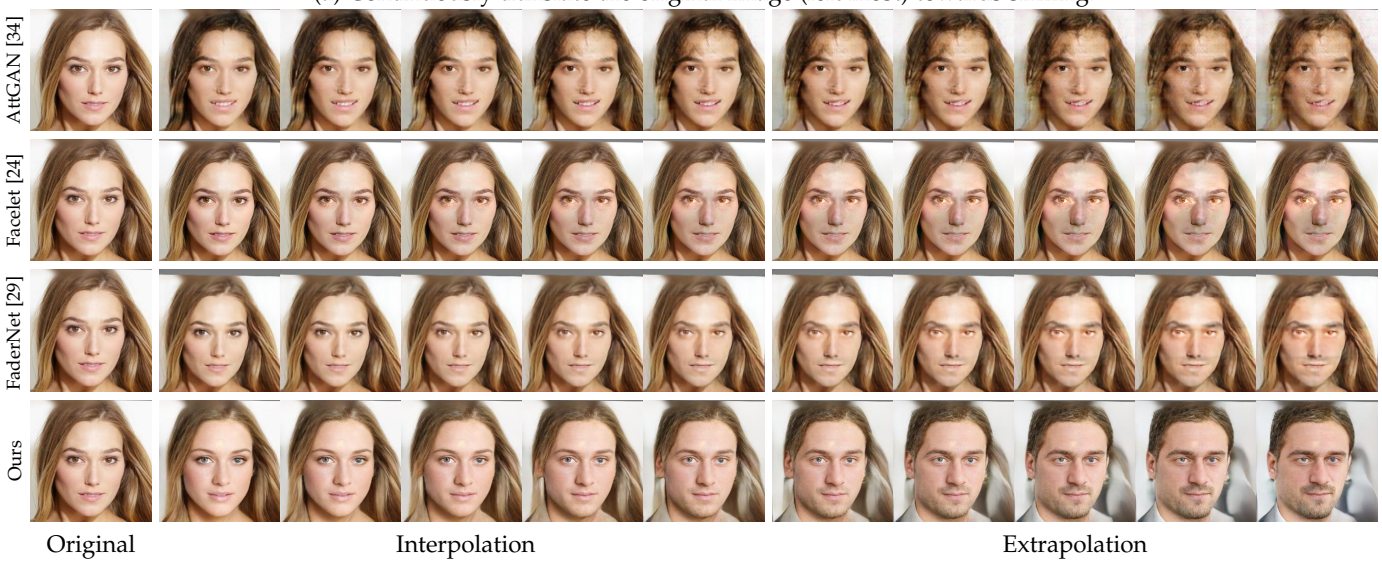
With this understanding, we intriguingly measure to what degree two domains are separated. For our interpolation model, completely separate domains are difficult to learn, while closely located domains are easier to handle.

Intuitively, measuring the separability of domains can be achieved with a domain classifier. To this end, we can train a CNN with a softmax layer to classify two opposite domains (e.g., Young vs. Old). Note that the softmax layer outputs probability of each domain. Thus we measure how confident the CNN is by calculating the entropy of output. A small entropy means the CNN is very confident about its prediction, indicating that the domains are well separated, while a large one suggests that the domains are close.

Note that a standard CNN may exaggerate the confidence. We thus apply temperature scaling to calibrate the estimated probability values [51]. Table 6 shows the estimated entropy on different attributes. Our model is not applicable when the output prediction has a very low entropy.




(a) Continuously translate the original image (left most) towards smiling.




(b) Continuously translate the original image (left most) towards male.

Fig. 12. Illustration of continuous translation. The first column is the original image. The 2nd-6th images are the results of interpolation, and the 7th-11th columns are the extrapolation results achieved by further increasing the interpolation strength.

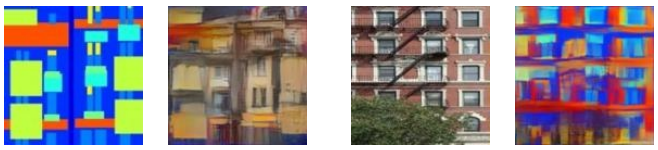



Fig. 13. A failure case. Our method does not perfectly handle the situation when two domains are essentially different.

TABLE 6
Applicability analysis of our model. We train a resnet-6 model until it achieves optimal accuracy. The entropy of output prediction is computed in the testing set.

| Task | Young | Smile | Male | Cat/Dog | Facades | Cat/Bird |
|---|---|---|---|---|---|---|
| Entropy | 0.297 | 0.206 | 0.143 | 0.071 | 9.64e-6 | 8.58e-6 |
| Accuracy | 87.6% | 89.3% | 92.0% | 100% | 100% | 100% |
| Applicable | True | True | True | True | False | False |

## 6 Complexity Analysis

Our network contains an encoder $\boldsymbol{E}$, an interpolator $\boldsymbol{I}$, a decoder $\boldsymbol{D}$, and two discriminators $\boldsymbol{\mathcal{D}_1}$ and $\boldsymbol{\mathcal{D}_2}$. We analyze the complexity of each part in a TITAN XP GPU, with input size $256 \times 256$. Table 7 summarizes running time, GFLOPS and the number of parameters of each part. As shown, the total number of parameters is 934.25 MB and the GFLOPs is 12.12. Although it is not light-weighted, the actual running time is 11.48 ms per image during inference, which allows real-time applications. The reason for the high-speed is that our model is most built upon convolutions, which largely benefit from the highly optimized CUDA implementation.

## 7 Concluding Remarks

We have proposed a framework for unpaired image-to-image translation focusing on generating natural and gradually changing intermediate results. Our method is based on latent space interpolation, which intrinsically allows continuous translation. In addition, by learning a controllable interpolator, we flexibly select the interpolation path, which alters the target attribute while keeping others almost intact.

TABLE 7
The running time, GFLOPS and model size of our model. "Total(test)" accumulates the metrics of the encoder, interpolator, decoder, indicating the computational cost during testing. "Total" accumulates that of all the five modules.

| | Encoder ($\boldsymbol{E}$) | Interpolator ($\boldsymbol{I}$) | Decoder ($\boldsymbol{D}$) | Discriminator ($\boldsymbol{\mathcal{D}_1}$) | Discriminator ($\boldsymbol{\mathcal{D}_2}$) | Total (test) | Total |
|---|---|---|---|---|---|---|---|
| running Time (ms) | 4.31 | 2.82 | 4.35 | 1.22 | 2.16 | 11.48 | 14.86 |
| GFLOPS | 2.52 | 2.72 | 5.19 | 0.03 | 1.66 | 10.43 | 12.12 |
| Parameters (MB) | 149.56 | 679.55 | 69.99 | 24.12 | 11.03 | 899.10 | 934.25 |

We have also shown that our method can serve multi-domain and multi-modal image-to-image translation.

## REFERENCES


[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision*, 2017.

[2] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *International Conference on Machine Learning*, 2017.

[3] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *European Conference on Computer Vision*, 2018.

[4] Y. Choi, M. Choi, and M. Kim, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[5] A. Almahairi, S. Rajeshwar, A. Sordoni, P. Bachman, and A. Courville, "Augmented cyclegan: Learning many-to-many mappings from unpaired data," in *International Conference on Machine Learning*, 2018.

[6] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Advances in Neural Information Processing Systems*, 2017.

[7] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014.

[8] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," *arXiv e-prints*, 2017.

[9] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "It takes (only) two: Adversarial generator-encoder networks," in *AAAI Conference on Artificial Intelligence*, 2018.

[10] T. White, "Sampling generative networks," *arXiv e-prints*, 2016.

[11] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv e-prints*, 2015.

[12] J. Tomczak and M. Welling, "Vae with a vampprior," in *International Conference on Artificial Intelligence and Statistics*, 2018.

[13] M. Bauer and A. Mnih, "Resampled priors for variational autoencoders," in *International Conference on Artificial Intelligence and Statistics*, 2019.

[14] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *arXiv e-prints*, 2016.

[15] Y.-C. Chen, X. Xu, Z. Tian, and J. Jia, "Homomorphic latent space interpolation for unpaired image-to-image translation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[16] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *arXiv e-prints*, 2018.

[17] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai, "Better mixing via deep representations," in *International Conference on Machine Learning*, 2013.

[18] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, "Understanding and improving interpolation in autoencoders via an adversarial regularizer," *arXiv e-prints*, 2018.

[19] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv e-prints*, 2015.

[20] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2018.

[21] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations*, 2015.

[22] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations*, 2018.

[23] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[24] Y.-C. Chen, H. Lin, M. Shu, R. Li, X. Tao, Y. Ye, X. Shen, and J. Jia, "Facelet-bank for fast portrait manipulation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[25] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Advances in Neural Information Processing Systems*, 2015.

[26] P. Upchurch, J. R. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Q. Weinberger, "Deep feature interpolation for image content changes." in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[27] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*, 2016.

[28] Y. Lu, Y.-W. Xu, Tai, and C.-K. Tang, "Attribute-guided face generation using conditional cyclegan," in *European Conference on Computer Vision*, 2018.

[29] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer *et al.*, "Fader networks: Manipulating images by sliding attributes," in *Advances in Neural Information Processing Systems*, 2017.

[30] R. Liu, Y. Liu, X. Gong, X. Wang, and H. Li, "Conditional adversarial generative flow for controllable image synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7992–8001.

[31] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017.

[32] W. Shen and R. Liu, "Learning residual images for face attribute manipulation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[33] Z. Yi, H. R. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation." in *IEEE International Conference on Computer Vision*, 2017.

[34] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Attgan: Facial attribute editing by only changing what you want," *IEEE Transactions on Image Processing*, 2019.

[35] B. Zhao, B. Chang, Z. Jie, and L. Sigal, "Modular generative adversarial networks," in *European Conference on Computer Vision*, 2018.

[36] D. K. P. I. S. J. C. Wonwoong Cho, Sungha Choi, "Image-to-image translation via group-wise deep whitening-and-coloring transformation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[37] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *European Conference on Computer Vision*, 2018.

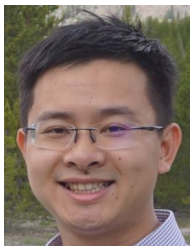
[38] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer, "Ganimation: Anatomically-aware facial animation from a single image," in *European Conference on Computer Vision*, 2018.

[39] R. Gong, W. Li, Y. Chen, and L. V. Gool, "Dlow: Domain flow for adaptation and generalization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[40] O. Langner, R. Dotsch, G. Bijlstra, D. H. Wigboldus, S. T. Hawk, and A. Van Knippenberg, "Presentation and validation of the radboud faces database," *Cognition and emotion*, 2010.

[41] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017.

[42] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations*, 2018.
[43] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, 2016.
[44] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *IEEE International Conference on Computer Vision*, 2017.
[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv e-prints*, 2014.
[46] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *International Conference on Learning Representations*, 2014.
[47] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv e-prints*, 2015.
[48] K. Shoemake, "Animating rotation with quaternion curves," 1985.
[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv e-prints*, 2014.
[50] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017.
[51] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, 2017.



**Ying-Cong Chen** received his PhD degree from the Chinese University of Hong Kong. He is currently a postdoctoral associate at the Computer Science and Artificial Intelligence Lab (CSAIL), Massachusetts Institute of Technology. He obtained the Hong Kong PhD Fellowship in 2016. He serves as a reviewer for IJCV, TIP, CVPR, ICCV, ECCV, BMVC, IJCAI, AAAI, etc. His research interest includes deep learning, image generation and editing, generative adversarial networks, etc.

**Jiaya Jia** received the PhD degree in Computer Science from Hong Kong University of Science and Technology in 2004 and is currently a full professor in Department of Computer Science and Engineering at the Chinese University of Hong Kong (CUHK). He was a visiting scholar at Microsoft Research Asia from March 2004 to August 2005 and conducted collaborative research at Adobe Systems in 2007. He is in the editorial boards of IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) and International Journal of Computer Vision (IJCV). He continuously served as area chairs for ICCV, CVPR, AAAI, ECCV, and several other conferences for organization. He was on program committees of major conferences in graphics and computational imaging, including ICCP, SIGGRAPH, and SIGGRAPH Asia. He received the Young Researcher Award 2008 and Research Excellence Award 2009 from CUHK. He is a Fellow of the IEEE.

# Supplementary Material of Homomorphic Interpolation Network for Unpaired Image-to-image translation

Ying-Cong Chen, Jiaya Jia, *Fellow, IEEE*

## 1 Additional Analysis

### 1.1 Discussion of the Attribute Classifier $\mathcal{A}'$

In our setting, the loss function of $\mathcal{A}'$ is

$$\min_{\mathcal{A}'} \mathcal{L}_{\mathcal{A}'} = \mathbb{E}[-\boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)))], \tag{1}$$

which means that $\mathcal{A}'$ is *always* trained to map the interpolated features to the original attributes. This could be counter-intuitive to some extent, i.e., $\mathcal{A}'$ has never been trained with real samples, and it could be hard to understand how it works. In this section, we will provide a more in-depth analysis to better understand this training scheme.

To facilitate the analysis, we divide the value of control vector $\boldsymbol{v}$ to 3 groups, i.e., small (close to 0), medium (around 0.5), and large (close to 1). We note that such a division is not strict, i.e., there is no clear boundary between the 3 groups. However, it can largely simplify the following discussion.

According to Eq. (9) of the manuscript, our interpolator is formulated as

$$\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j) = \boldsymbol{F}_i + \sum_{k=1}^{c} \boldsymbol{v}^k \boldsymbol{\mathcal{T}}^k (\boldsymbol{F}_j - \boldsymbol{F}_i). \tag{2}$$

When $\boldsymbol{v}$ is small, the second term of Eq. (2) is ignorable, and $\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)$ is close to $\boldsymbol{F}_i$. As a result, Eq. (1) is reduced to

$$\begin{aligned} \min_{\mathcal{A}'} \mathcal{L}_{\mathcal{A}'} &= \mathbb{E}[-\boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)))] \\ &\approx \mathbb{E}[-\boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{F}_i))], \end{aligned} \tag{3}$$

which is approximately equivalent to training with real features $\boldsymbol{F}_i$. In this sense, although $\mathcal{A}'$ have never seen real samples, during the training process, it can still obtain meaningful supervision when $\boldsymbol{v}$ is small.

When $\boldsymbol{v}$ is in the medium region, both terms in Eq. (2) cannot be ignored, and $\mathcal{A}'$ keeps checking the unchanged parts as the interpolator $\boldsymbol{I}_{\boldsymbol{v}}$ improves. As discussed in the manuscript, this could force $\boldsymbol{I}_{\boldsymbol{v}}$ to make more significant changes.

When $\boldsymbol{v}$ is large, Eq. (1) is reduced to

$$\begin{aligned} \min_{\mathcal{A}'} \mathcal{L}_{\mathcal{A}'} &= \mathbb{E}[-\boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)))] \\ &\approx \mathbb{E}[-\boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{F}_j))]. \end{aligned} \tag{4}$$

In this case, $\boldsymbol{z}_i$ and $\boldsymbol{F}_j$ are not matched, which means that Eq. (4) trains $\mathcal{A}'$ with *noisy labels*. Although this could harm the accuracy of $\mathcal{A}'$, it is still useful to the training of the interpolator $\boldsymbol{I}_{\boldsymbol{v}}$, because it prevents $\mathcal{A}'$ from getting saturated. Note that once $\mathcal{A}'$ gets saturated (producing extreme values), it could no longer provide any gradient to supervise the training of $\boldsymbol{I}_{\boldsymbol{v}}$. In this regard, it is similar with the "label flipping" [1] trick in training GAN.

In our implementation, $\boldsymbol{v}$ is uniformly sampled. When it is small, Eq. (3) associates the latent feature to its corresponding attributes. When it is large, Eq. (4) could prevent the $\mathcal{A}'$ from getting saturated. Besides, it is worthy to point out that Eq. (4) is unlikely to dominate the optimization, as fitting random labels is slower than fitting the correct ones [2]. In this regard, $\mathcal{A}'$ can still lead to satifactory prediction accuracy, yet the prediction value is not saturated.

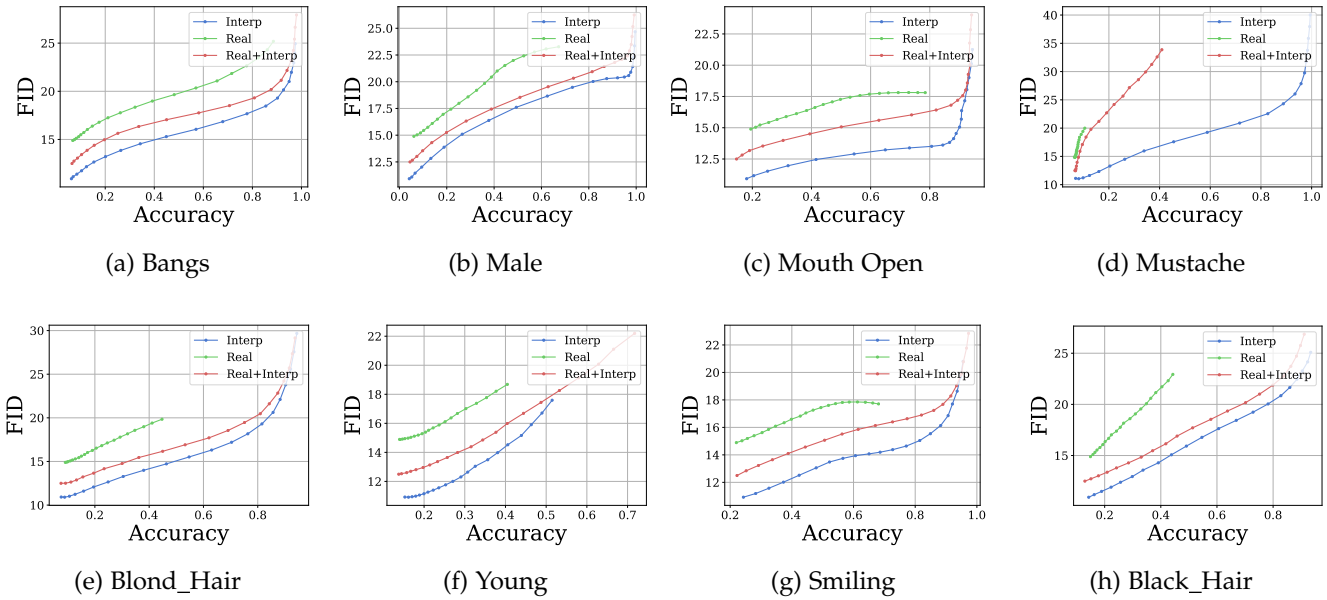
To validate this, we compare different performance metrics between Eq. (1) and two other variants, Eq. (5) and Eq. (6). Eq. (5) trains $\mathcal{A}'$ with real features, and Eq. (6) trains it with both real and interpolated features.

$$\min_{\mathcal{A}'} \mathcal{L}_{\mathcal{A}'} = \mathbb{E}[-\boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{F}_i))], \tag{5}$$

$$\min_{\mathcal{A}'} \mathcal{L}_{\mathcal{A}'} = \mathbb{E}[-\boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{F}_i)) - \boldsymbol{z}_i \log(\mathcal{A}'(\boldsymbol{I}_{\boldsymbol{v}}(\boldsymbol{F}_i, \boldsymbol{F}_j)))]. \tag{6}$$

As both Eq. (5) and Eq. (6) train $\mathcal{A}'$ with real samples, we can expect that they can achieve slightly higher average bprediction accuracy than our original model Eq. (1). This is shown in Table 1. However, a higher **prediction accuracy** of $\mathcal{A}'$ does not mean a better **generation quality** of the whole interpolation model. To demonstrate this, we plot the "FID vs ACC" curves in Fig. 1. As shown, the original model ("Interp") performs better than the other two variants, even if the prediction accuracy in Table 1 is slightly lower. The reason is that, when trained with real features, $\mathcal{A}'$ tends to produce extreme values (see the prediction score of different models in Fig. 2). In this case, $\mathcal{A}'$ could get saturated, and

1. https://github.com/soumith/ganhacks [1]




(a) Bangs (b) Male (c) Mouth Open (d) Mustache

(e) Blond_Hair (f) Young (g) Smiling (h) Black_Hair

Fig. 1. Quantitative comparison of different alternatives. The x-axis and y-axis are the translation accuracy (the larger the better) and FID score (the smaller the better). Each curve is plotted with different edit strength. "Interp" denotes training $\mathcal{A}'$ with Eq. (1), which is used in our manuscript. "Real" denotes training with Eq. (5). "Real+Interp" denotes training with Eq (6).

| Attribute | Bangs | Male | Mouth Open | Mustache | Blond Hair | Young | Smiling | Black Hair | Average |
|---|---|---|---|---|---|---|---|---|---|
| Interp | 95.3% | 95.2% | 93.1% | 94.3% | 90.9% | 87.2% | 92.1% | 88.6% | 92.1% |
| Real | 94.9% | 97.8% | 91.3% | 94.8% | 94.7% | 87.3% | 93.2% | 88.4% | 92.8% |
| Real+Interp | 94.1% | 94.0% | 93.3% | 95.9% | 94.8% | 88.2% | 93.5% | 88.1% | 92.7% |

TABLE 1
Comparison of the prediction accuracy. "Interp" denotes training $\mathcal{A}'$ with Eq. (1), which is used in our manuscript. "Real" denotes training with Eq. (5). "Real+Interp" denotes training with Eq (6).

does not back-propagate enough information to train the interpolator $\boldsymbol{I_v}$. Fig. 2 also shows that the model of Eq. (5) (denoted as "Real") produces milder edits than the other two models, and the model of Eq. (6) does not lead to better results than the model of Eq. (1). From the prediction scores below each image, we can see that Eq. (5) could produce extreme values even if the changes are mild, while our model does not produce extreme values. Therefore, we choose Eq. (1) to train $\mathcal{A}'$ in our model.

### 1.2 Discussion of the Interpolator

Comparing with existing approaches [3], [4] that modify attributes by simply shifting the latent feature in certain directions, our model uses a more complicated CNN-based interpolator to model the transformation. Our main concern is that a simple feature shifting cannot handle the multi-modal nature of attribute transformation. To illustrate this, we added a prejection-based interpolator baseline by replacing the CNN-based interpolator defined in Eq. (2) with

$$\boldsymbol{I_v}(\boldsymbol{F}_i, \boldsymbol{F}_j) = \boldsymbol{F}_i + \sum_{k=1}^{c} \boldsymbol{v}^k [\boldsymbol{P}^k \odot (\boldsymbol{F}_j - \boldsymbol{F}_i)] \boldsymbol{P}^k, \qquad (7)$$

where $\boldsymbol{P}^k, k = 1, 2, \cdots, c$ is the learned directions, $\odot$ denotes the inner product operation. Then, we compare the multi-modal translation results between the CNN-based interpolator (Eq. (2)) sand the projection-based one (Eq. (7)) in Fig. 3. As shown, the projection-based interpolator cannot generate diverse results. With different $\boldsymbol{F}_j$, the generated results differ in strength instead of style. This is because projecting $\boldsymbol{F}_j - \boldsymbol{F}_i$ to a *single* direction could discard the variations the target attribute. As a result, it can only produce one style for each attribute.

## 2 Additional Experiments

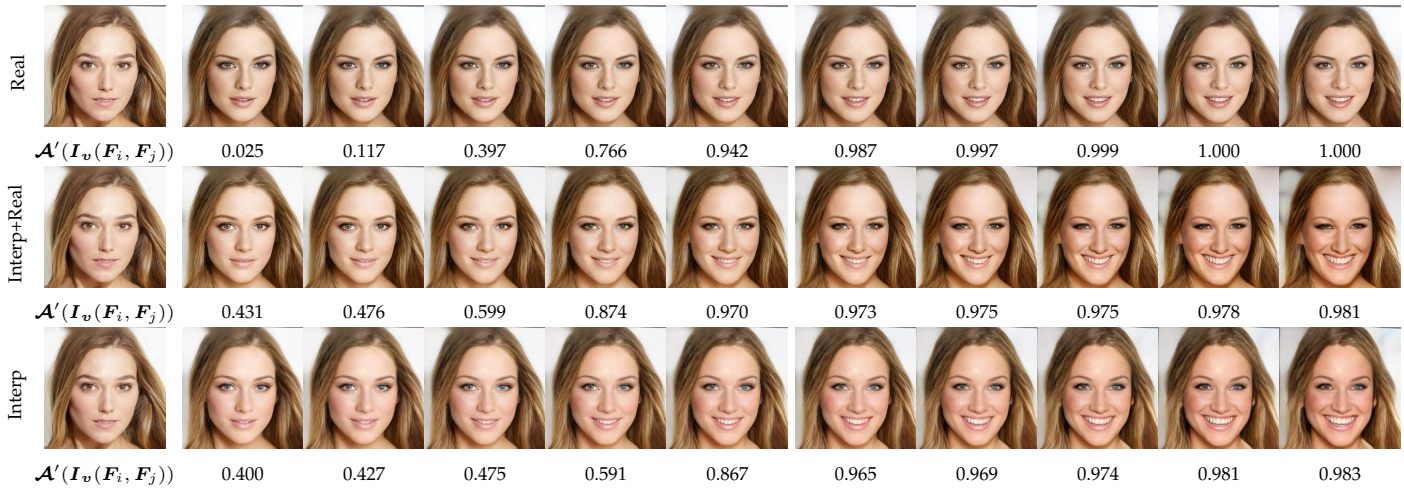
### 2.1 Results on Misaligned Faces

In this paper, we aligning keypoints of the query/reference images during training and testing. This makes the model converges faster. In this section, we demonstrate that our model can also deal with unaligned faces. To achieve this, during training, we randomly shift the training images along both x-axis and y-axis for 80 pixels. This makes the model robust to misalignment in some extent. Then we show the interpolation curves between shifted samples and non-shifted samples in Fig. 4. We can see that although the two images are misaligned, the intermediate results are still realistic, and the transition is still smooth.

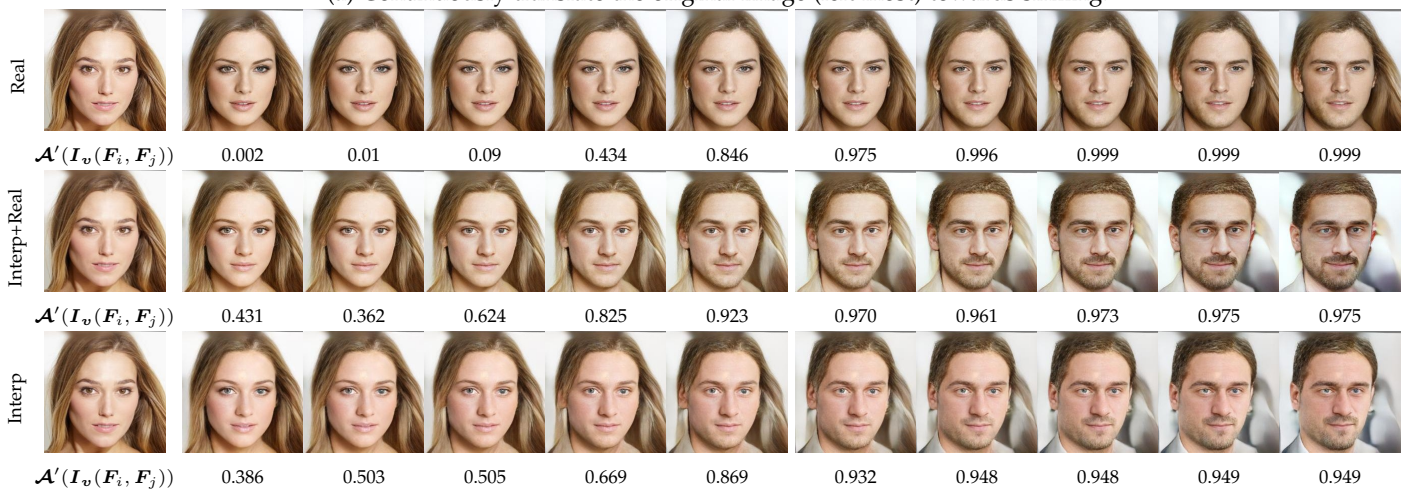
### 2.2 Results on Non-face datasets

Except for faces, Our model can also be applied to other image-to-image translation tasks. We take the "cat vs dog" dataset [5] and the Market-1501 [6] dataset for illustration.

The "cat vs dog" dataset contains 771 cat images and 1,264 dog images for training, and 100 both categories for




(a) Continuously translate the original image (left most) towards smiling.




(b) Continuously translate the original image (left most) towards male.

Fig. 2. Generated results and attribute prediction scores of "Interp", "Real" and "Interp+Real". "Interp" denotes training $\mathcal{A}'$ with Eq. (1), which is used in our manuscript. "Real" denotes training with Eq. (5). "Real+Interp" denotes training with Eq (6).For each output, we show attribute prediction score $\mathcal{A}'(\boldsymbol{I_v}(\boldsymbol{F}_i, \boldsymbol{F}_j))$ below.


Fig. 3. Comparing the multi-modal image-to-image translation results between projection-based interpolator and CNN-based interpolator. The $1^{st}$ column is the original image, and other columns are different output.

testing. To deal with this task, our interpolator contains two branches, i.e., the first branch modifies the domain label, and the second branch handles the residual component. Fig. 5 shows the results by continuously interpolating with the first branch. As shown, our model can conduct continuous translation between cats and dogs. For example, when changing a cat towards a dog domain, its nose gradually becomes longer, and the eyes become smaller and darker.

The Market-1501 dataset contains 32688 pedestrian images captured by surveillance cameras. Each image is annotated with 27 attributes, such as clothes colors, gender, age, etc. The clothes color attributes are used to test our model. Specifically, Our model contains two branches, i.e., one deals with the grouped attributes of clothes colors, and the other handles other components. To translate a query image to a certain color, we randomly pick one image of that color

as the reference image, and conduct interpolation with the color branch. As shown in Fig. 6, our model can correctly modify the colors of clothes of the pedestrians without introducing other irrelevant changes. Fig. 7 shows an example of continuous translation. As shown, with different interpolation strengths, our model can smoothly change the color of clothes from yellow to the target ones.

### 2.3 Interpolation along Different Paths

Fig. 8 and Fig. 14 shows examples of interpolation along different paths. The model trained with CelebA is used for illustration. By gradually changing the control vector $\boldsymbol{v}$ from $\mathbf{0}$ to $\mathbf{1}$, we can synthesize the intermediate results from the input to the reference. Also, modifying the grouped attributes in different orders leads to different interpolation paths.

### 2.4 More Comparisons with State-of-the-arts

Figs. 15 and 16 compare more to state-of-the-art results. MUNIT [7] tends to impose strong edit, and it occasionally modifies irrelevant information. GDWCT [8] sometimes generates blurry and unnatural results. StarGAN [9] and AttGAN [10] work well. But the quality can still be improved.

### 2.5 More Results on CelebA

Figs. 18-20 present more results on CelebA-HD [4]. Given an input image, we randomly choose a reference from an opposite target domain, and process it with our model. Each row shows one example, and each column shows the translated version of a certain attribute.

### 2.6 More Results on RaFD

Because most identities of RaFD [11] are used for training, we instead use images gathered from the web to test our model. Specifically, given an additional image, we detect and crop the face region, process it with our model, and finally blend it back with the original image. Also, we leverage images of the RaFD dataset as exemplars. Figs. 9-13 show the results on different styles of images, including movie snapshot, oil painting, magazine cover, and sketch images. Our model correctly modifies the expression or gaze of these faces, which demonstrates its effectiveness in practical use.

### 2.7 Results of Larger Resolution

Note that the resolution of all images is $256 \times 256$. Our model can also deal with images of higher resolutions, i.e., $512 \times 512$. On the model presented in Table 1 of the manuscript, we add a down-sampling block and an up-sampling block to the encoder and decoder respectively. Besides, we add one more layer in the pixel-level discriminator $\boldsymbol{\mathcal{D}_2}$. Due to the memory issue, we also replace convolution layers of the interpolator with separable convolutional layers. The details are summarized in Table 2. The architecture of the interpolator and feature-level discriminator remains the same. Figs. 21 and 22 show results in this resolution.

| Layer Type | Norm | Activation | Input Size | Output Size |
|---|---|---|---|---|
| Conv(3,1,1) | IN | LReLU | $512 \times 512 \times 3$ | $512 \times 512 \times 16$ |
| DownBlock | IN | LReLU | $512 \times 512 \times 16$ | $256 \times 256 \times 32$ |
| DownBlock | IN | LReLU | $256 \times 256 \times 32$ | $128 \times 128 \times 64$ |
| DownBlock | IN | LReLU | $128 \times 128 \times 64$ | $64 \times 64 \times 128$ |
| DownBlock | IN | LReLU | $64 \times 64 \times 128$ | $32 \times 32 \times 256$ |
| DownBlock | IN | LReLU | $32 \times 32 \times 256$ | $16 \times 16 \times 512$ |
| DownBlock | IN | LReLU | $16 \times 16 \times 512$ | $8 \times 8 \times 1024$ |
| DownBlock | IN | LReLU | $8 \times 8 \times 1024$ | $4 \times 4 \times 2048$ |

(a) Encoder $\boldsymbol{E}$

| Layer Type | Norm | Activation | Input Size | Output Size |
|---|---|---|---|---|
| UpBlock | IN | LReLU | $4 \times 4 \times 2048$ | $8 \times 8 \times 1024$ |
| UpBlock | IN | LReLU | $8 \times 8 \times 1024$ | $16 \times 16 \times 512$ |
| UpBlock | IN | LReLU | $16 \times 16 \times 512$ | $32 \times 32 \times 256$ |
| UpBlock | IN | LReLU | $32 \times 32 \times 256$ | $64 \times 64 \times 128$ |
| UpBlock | IN | LReLU | $64 \times 64 \times 128$ | $128 \times 128 \times 64$ |
| UpBlock | IN | LReLU | $128 \times 128 \times 64$ | $256 \times 256 \times 32$ |
| UpBlock | IN | LReLU | $256 \times 256 \times 32$ | $512 \times 512 \times 16$ |
| Conv(3,1,1) | - | - | $512 \times 512 \times 16$ | $512 \times 512 \times 3$ |

(b) Decoder $\boldsymbol{D}$

| Layer Type | Norm | Activation | Input Size | Output Size |
|---|---|---|---|---|
| SepConv(3,1,1) | - | LReLU | $4 \times 4 \times 2048$ | $4 \times 4 \times 2048$ |
| SepConv(3,1,1) | - | LReLU | $4 \times 4 \times 2048$ | $4 \times 4 \times 2048$ |
| SepConv(3,1,1) | - | - | $4 \times 4 \times 2048$ | $4 \times 4 \times 2048$ |

(c) Interpolator $\boldsymbol{I}$

| | | | | |
|---|---|---|---|---|
| Conv(4,2,1) | IN | LReLU | $512 \times 512 \times 3$ | $256 \times 256 \times 64$ |
| Conv(4,2,1) | IN | LReLU | $256 \times 256 \times 64$ | $128 \times 128 \times 128$ |
| Conv(4,2,1) | IN | LReLU | $128 \times 128 \times 128$ | $64 \times 64 \times 256$ |
| Conv(4,2,1) | IN | LReLU | $64 \times 64 \times 256$ | $32 \times 32 \times 512$ |
| Conv(4,2,1) | IN | LReLU | $32 \times 32 \times 512$ | $16 \times 16 \times 1024$ |

(d) Image-level Discriminator $\boldsymbol{\mathcal{D}_2}$

TABLE 2
Architecture of $512 \times 512$ model. (a-d) are the encoder, decoder, interpolator and image-level discriminator respectively. SepConv(3,1,1) denotes the separable convolutional layer [12], which is composed of a depth-wise convolutional layer and a point-wise convolutional layer.

## 3 The Video

The video shows some examples of generating animation from still images with continuous translation. Specifically, this is accomplished by *continuously* translating the input image to the target domains. Images of the RaFD dataset is used as exemplars. Our model can effectively accomplish this task.

## References


[1] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv e-prints*, 2016.

[2] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv e-prints*, 2016.

[3] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2018.

[4] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[5] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *European Conference on Computer Vision*, 2018.

[6] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[7] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *European Conference on Computer Vision*, 2018.

[8] D. K. P. I. S. J. C. Wonwoong Cho, Sungha Choi, "Image-to-image translation via group-wise deep whitening-and-coloring transformation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[9] Y. Choi, M. Choi, and M. Kim, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[10] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Attgan: Facial attribute editing by only changing what you want," *IEEE Transactions on Image Processing*, 2019.

[11] O. Langner, R. Dotsch, G. Bijlstra, D. H. Wigboldus, S. T. Hawk, and A. Van Knippenberg, "Presentation and validation of the radboud faces database," *Cognition and emotion*, 2010.
[12] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

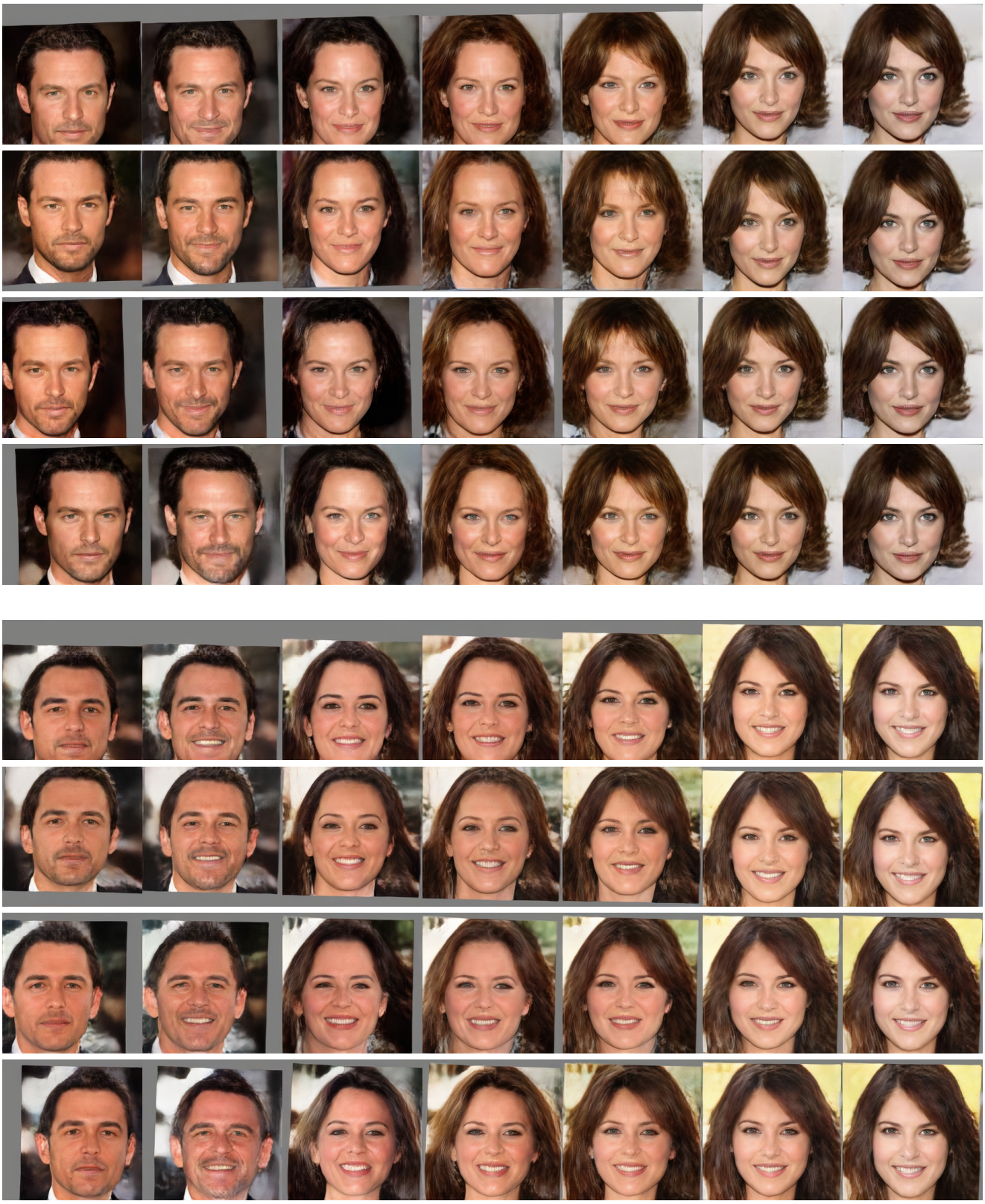
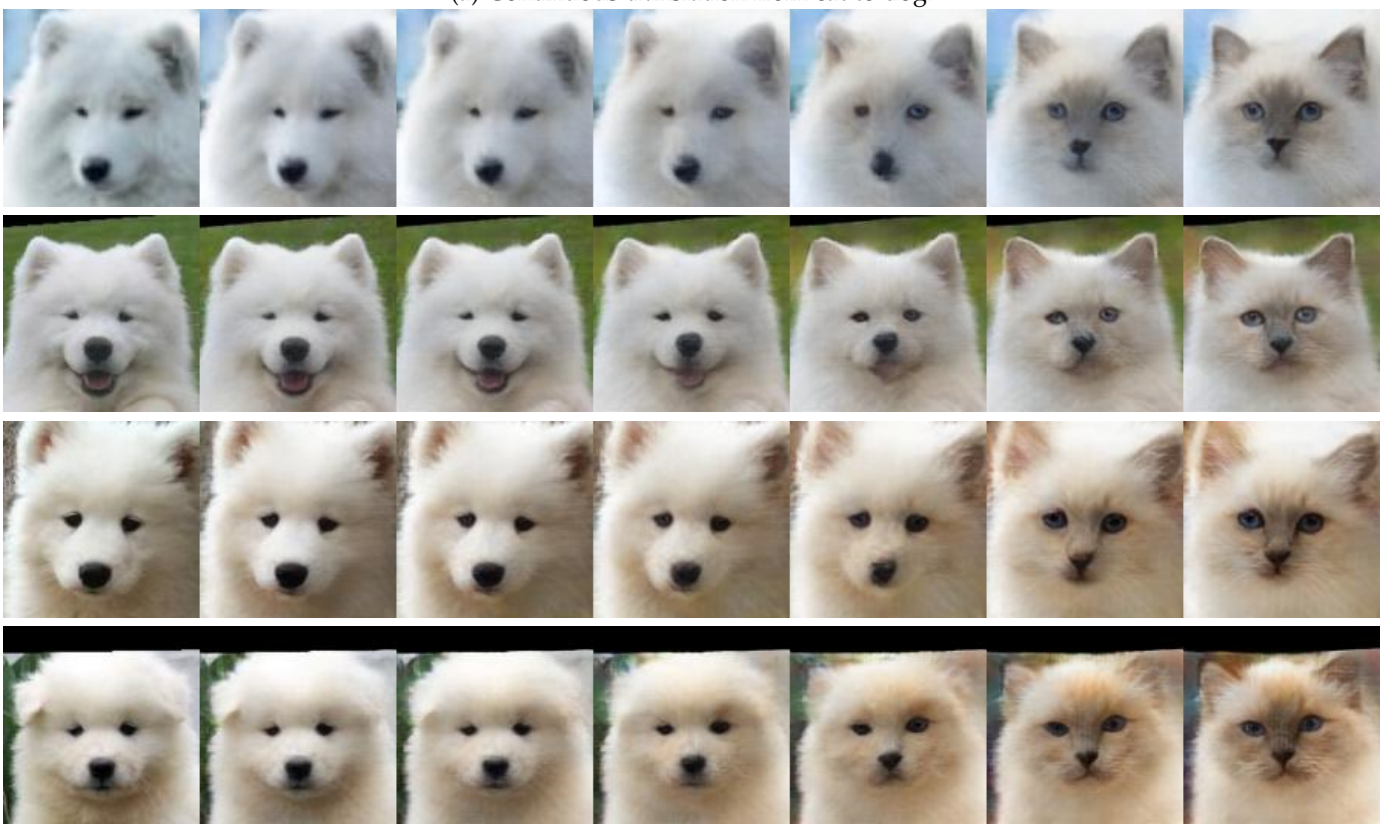


Fig. 4. Results of misaligned faces. Each row illustrates an interpolation curve. The first and the last images are the input and reference. We sequentially modify the expression, gender, hair color, hair style and age attributes. For row 1-4, 5-8, the inputs are shifted down/up/left/right for 80 pixels respectively.

(a) Continuous translation from cat to dog.



(b) Continuous translation from dog to cat.

Fig. 5. Domain translation between dog and cat. Each row represents a case. The first column is the input, the 2nd-6th columns are the intermediate results between the two domains, and the last column is the translated version to the opposite domain.

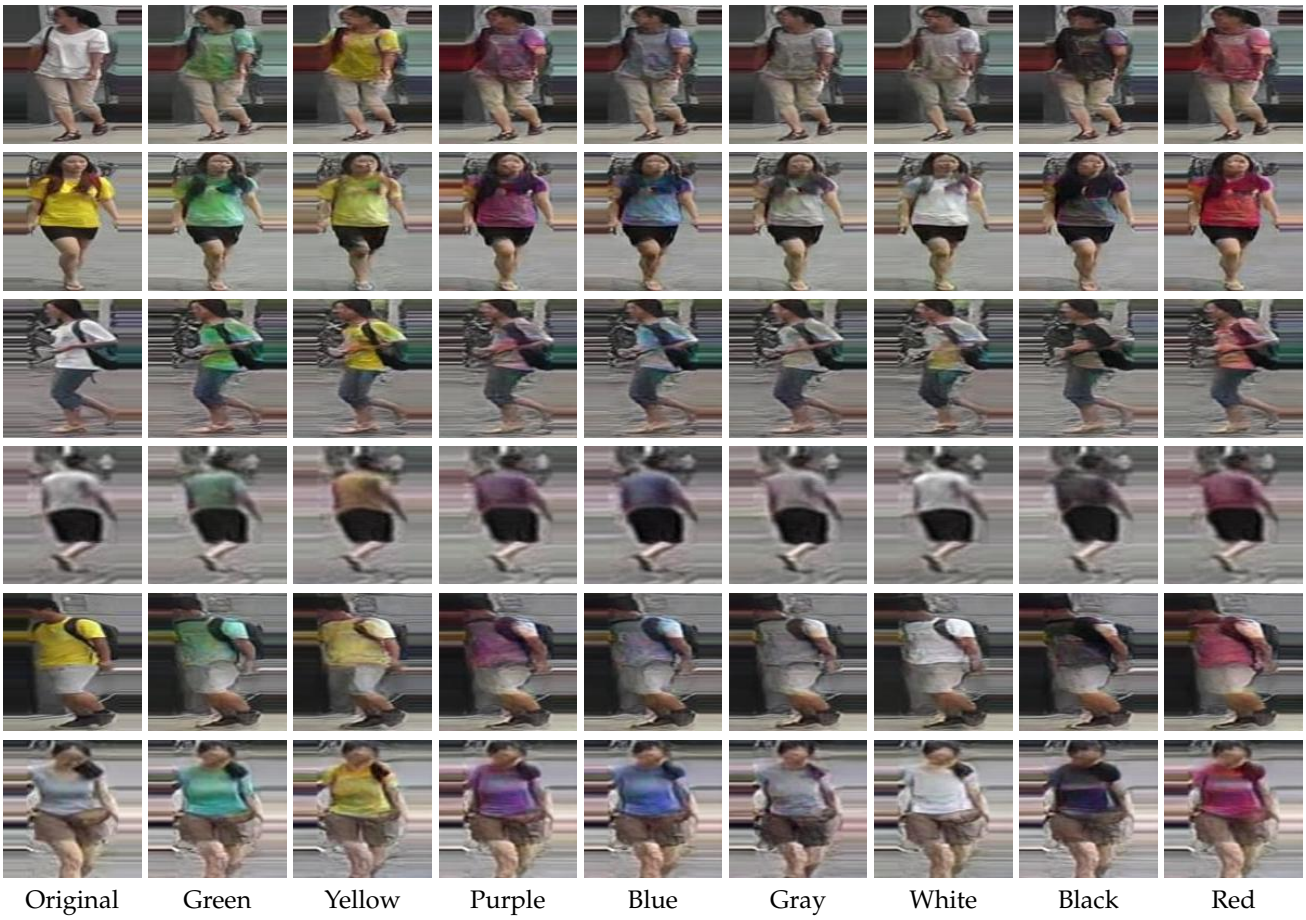



Fig. 6. Multi-domain translation on Market-1501 [6].

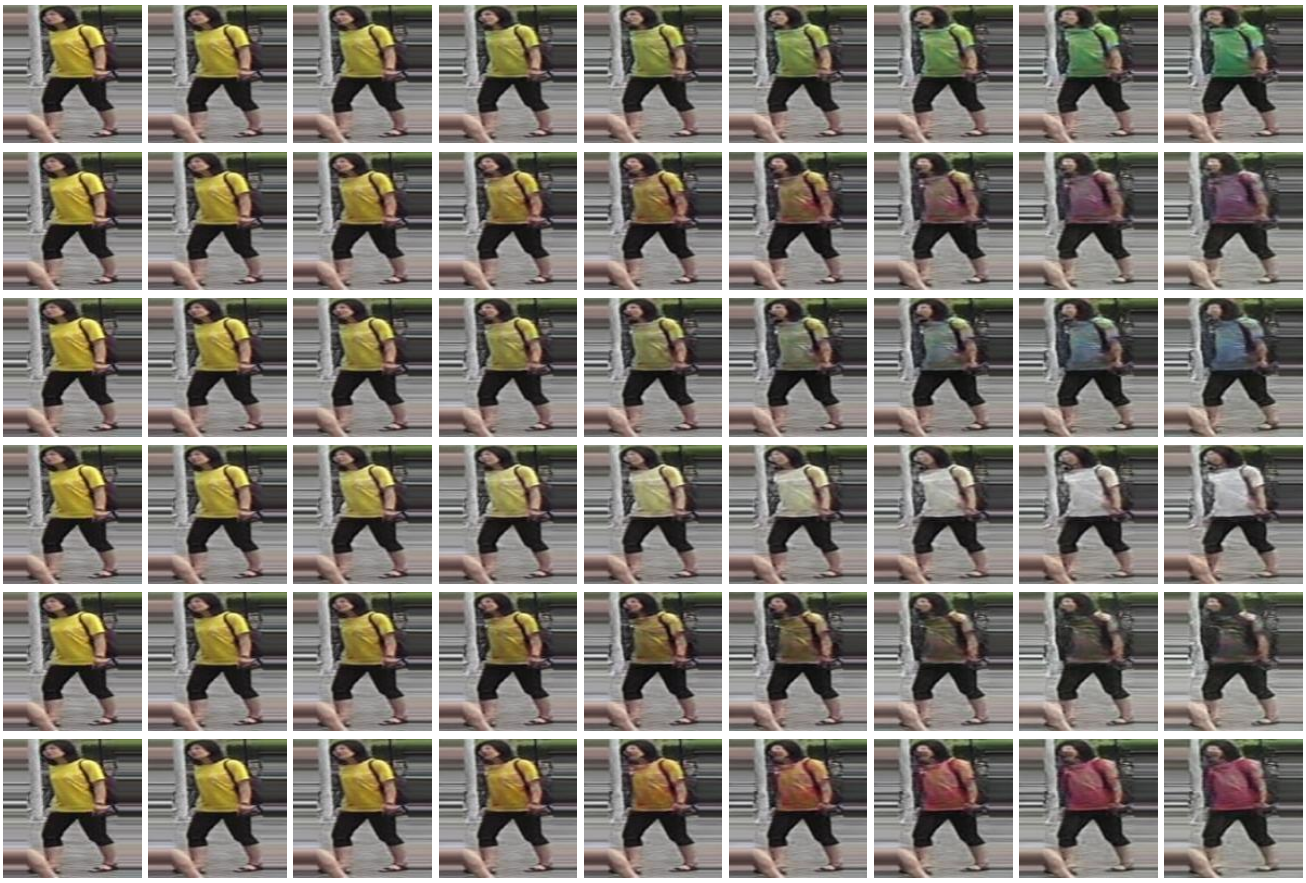


Fig. 7. Continuous translation on Market-1501 [6].

Path #1: Expression → Gender → Hair Color → Hair Style → Age

Path #2: Age → Hair Style → Hair Color → Gender → Expression

Path #3: Age → Gender → Expression → Hair Color → Hair Style

Path #4: Gender → Hair Color → Hair Style → Age → Expression

Path #5: Hair Color → Hair Style → Age → Expression → Gender

Path #6: Expression → Hair Color → Age → Gender → Hair Style

Path #7: Gender → Age → Expression → Hair Color → Hair Style

Fig. 8. Interpolation on different paths. The first and last images are input and reference. The 2nd-6th images are intermediate results, each of which modifies one attribute. The order of modified attributes is presented below each row of images.

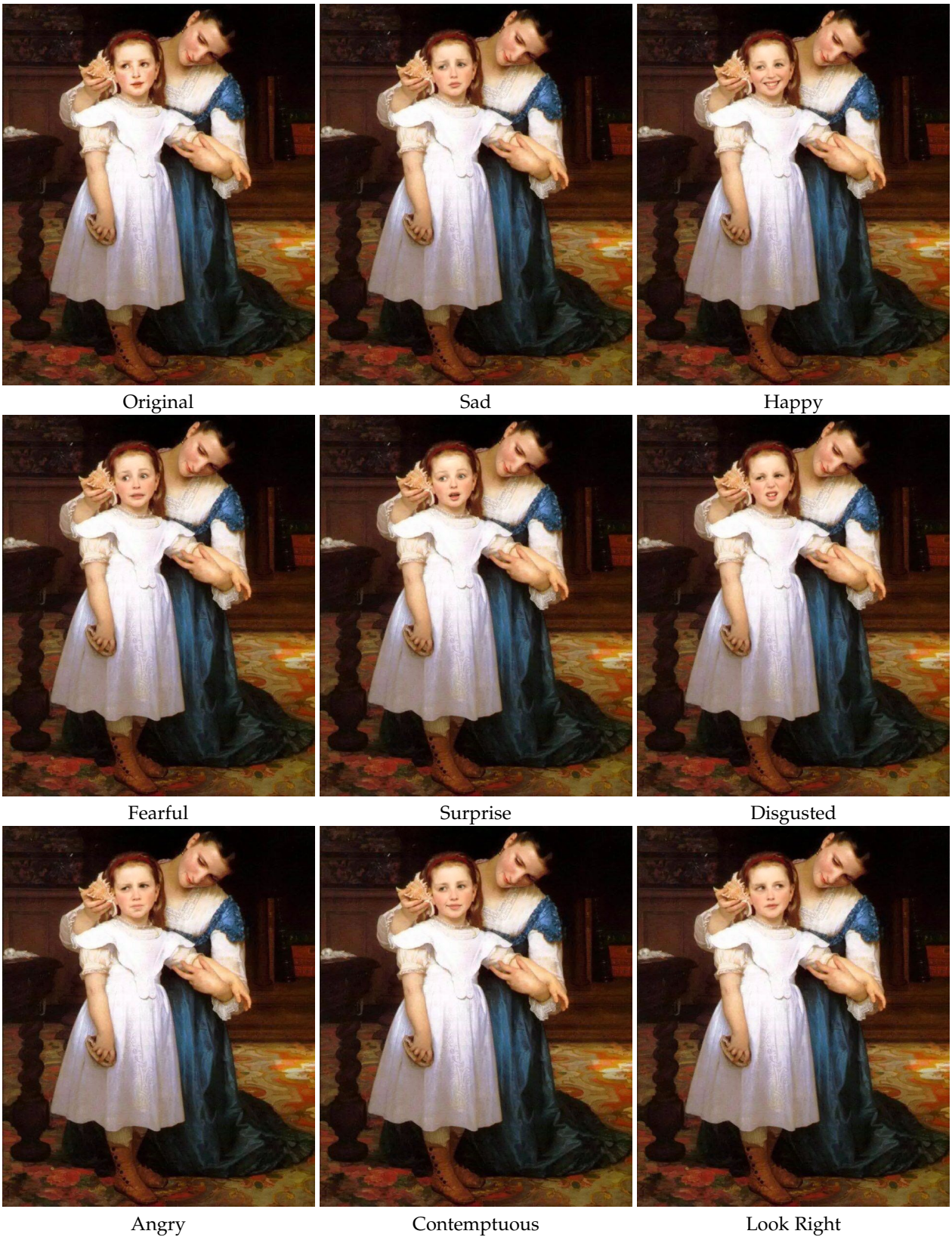



Fig. 9. Testing our RaFD model on oil painting.

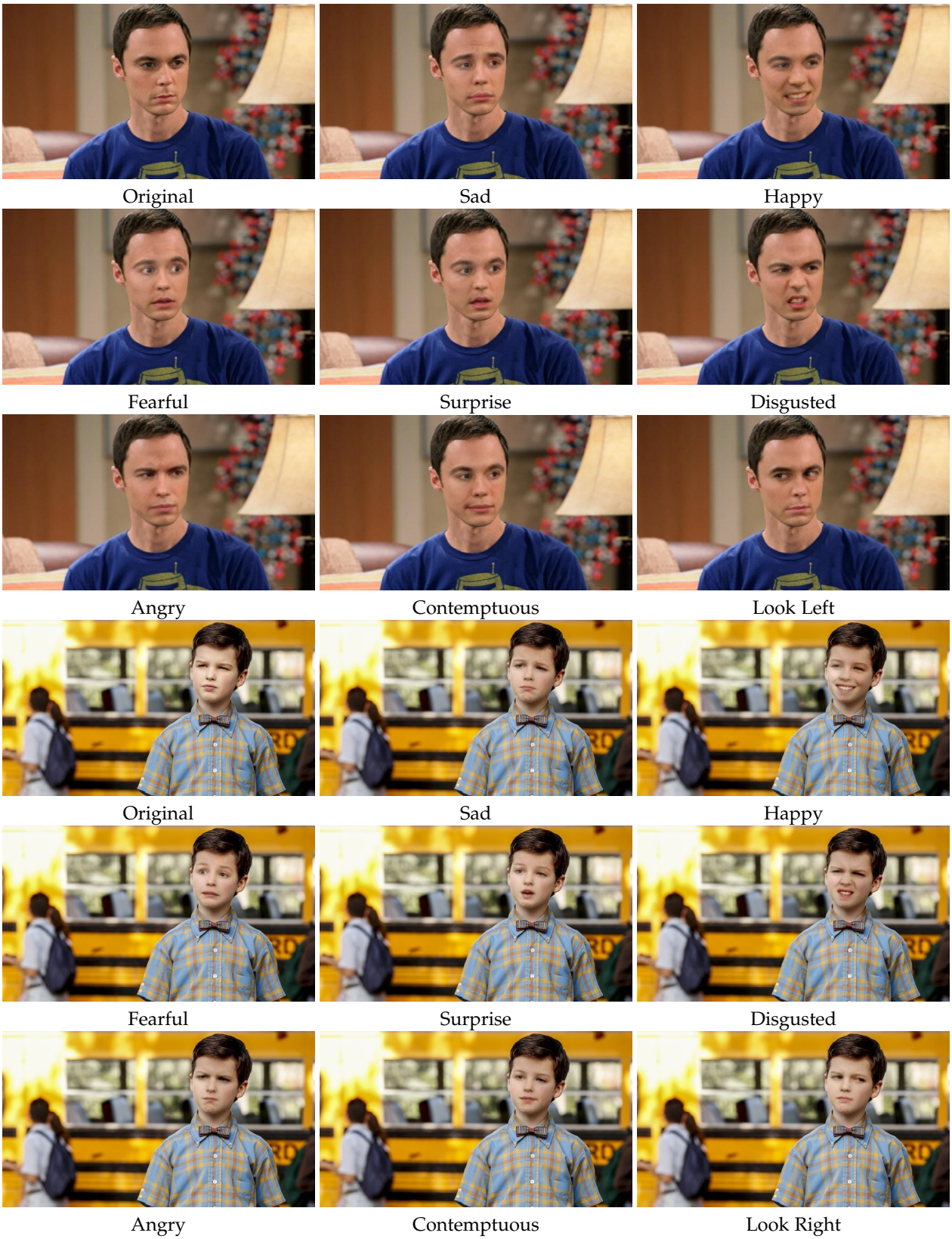



Fig. 10. Testing our RaFD model on movie snapshots.


Original | Neutral | Sad

Fearful | Surprise | Disgusted

Angry | Contemptuous | Look Right

Fig. 11. Testing our RaFD model on magazine cover.

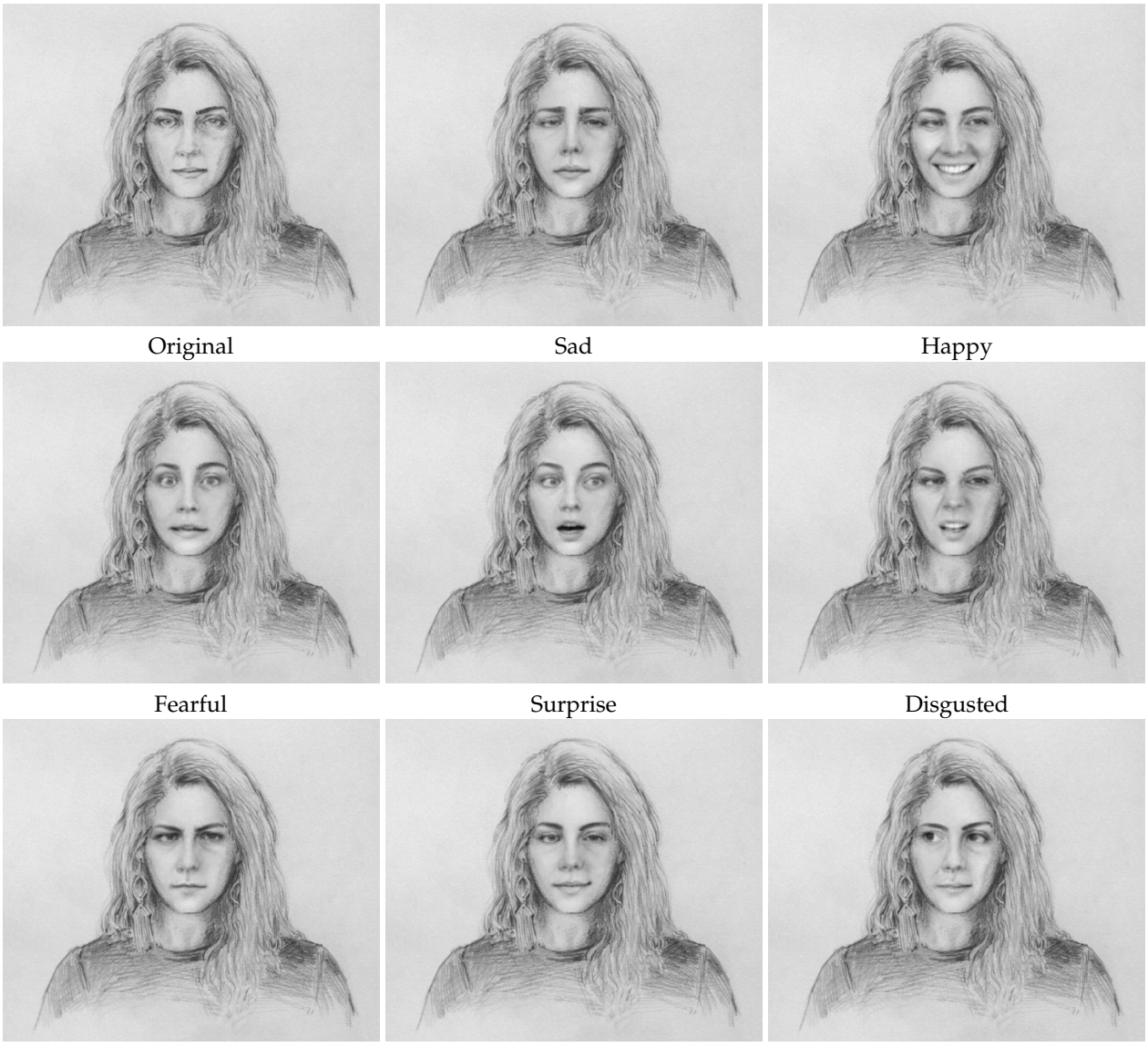



Fig. 12. Testing our RaFD model on sketch images.

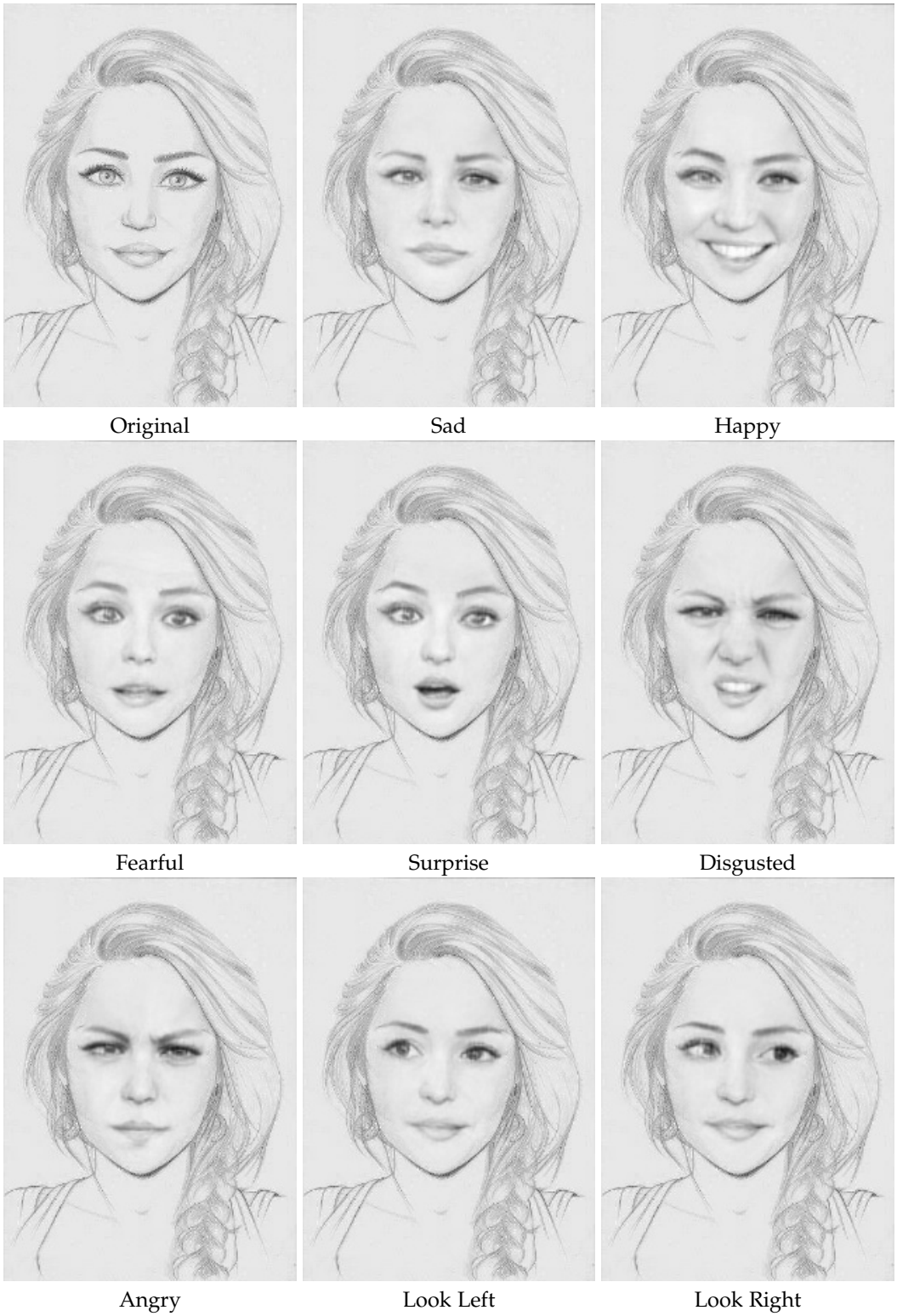
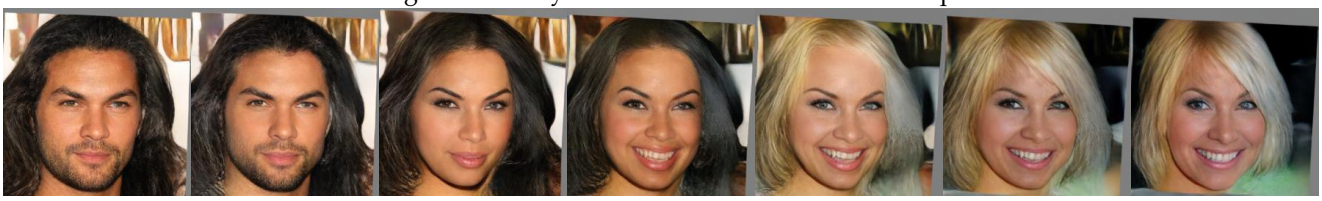



Fig. 13. Testing our RaFD model on sketch images.

Path #1: Expression → Gender → Hair Color → Hair Style → Age

Path #2: Age → Hair Style → Hair Color → Gender → Expression

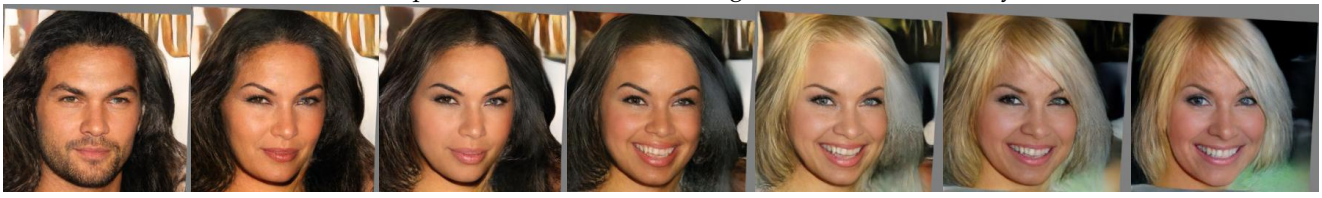


Path #3: Age → Gender → Expression → Hair Color → Hair Style

Path #4: Gender → Hair Color → Hair Style → Age → Expression

Path #5: Hair Color → Hair Style → Age → Expression → Gender

Path #6: Expression → Hair Color → Age → Gender → Hair Style



Path #7: Gender → Age → Expression → Hair Color → Hair Style

Fig. 14. Translation on different paths. The first and last images are input and reference. The 2nd-6th images are intermediate results, each of which modifies one attribute. The order of modified attributes is presented below each row of images.

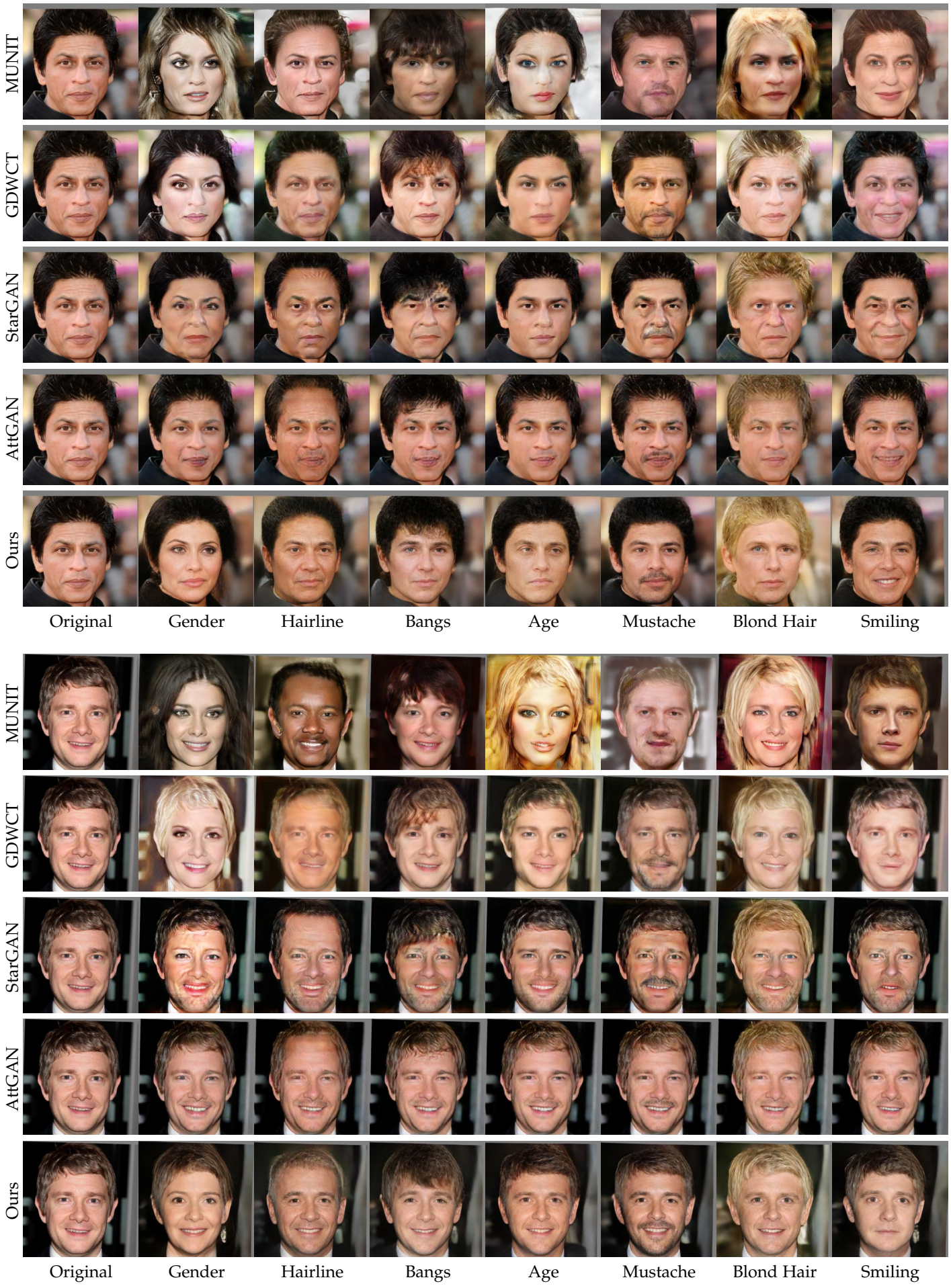



Fig. 15. More comparison with the state-of-the-art.


Fig. 16. More comparison with the state-of-the-art.


Fig. 17. More results on CelebA-HQ [4]. Each row presents one sample. The 1st column is the original image. The 2nd-7th columns are translated results to different opposite domains respectively.

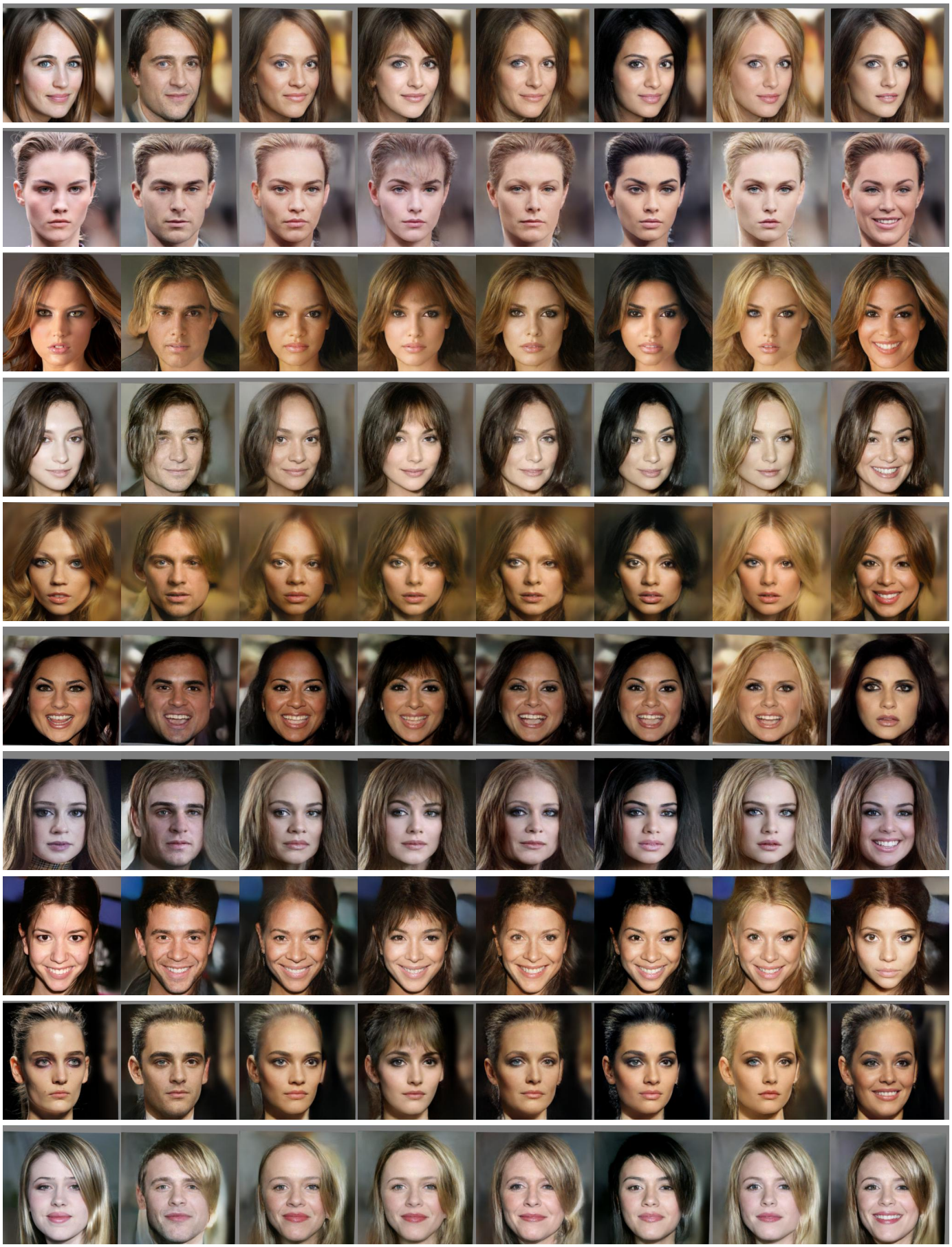



Fig. 18. More results on CelebA-HQ [4]. Each row presents one sample. The 1st column is the original image. The 2nd-7th columns are translated results to different opposite domains respectively.

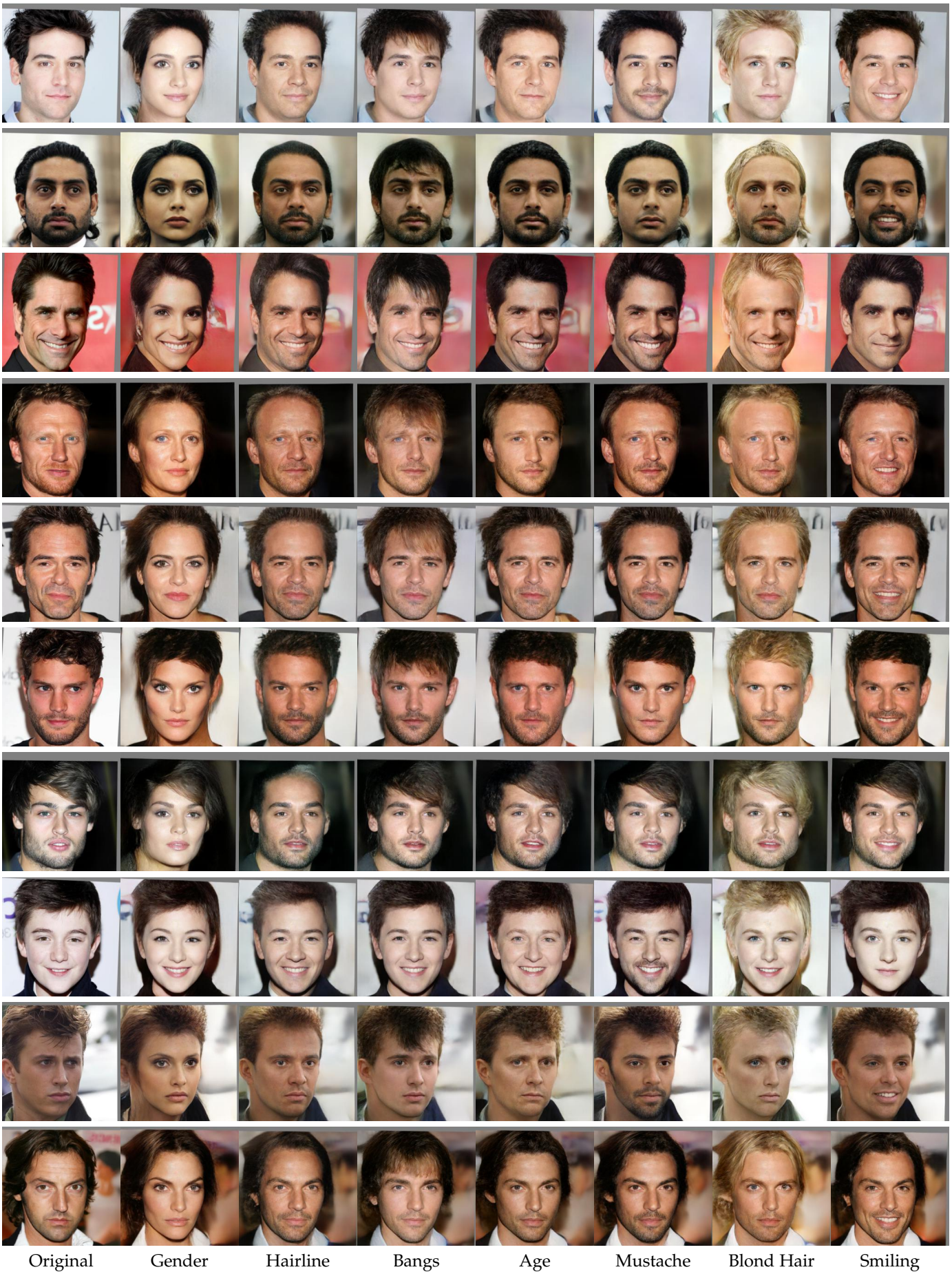



Fig. 19. More results on CelebA-HQ [4]. Each row presents one sample. The 1st column is the original image. The 2nd-7th columns are translated results to different opposite domains respectively.


Fig. 20. More results on CelebA-HQ [4]. Each row presents one sample. The 1st column is the original image. The 2nd-7th columns are translated results to different opposite domains respectively.

(a) Original (b) Expression (c) Gender (d) Hair Color

Fig. 21. Results of $512 \times 512$ images.

(a) Original (b) Expression (c) Gender (d) Hair Color

Fig. 22. Results of $512 \times 512$ images.