Facelet-Bank for Fast Portrait Manipulation

Ying-Cong Chen[1] Huaijia Lin[1] Michelle Shu[3] Ruiyu Li[1] Xin Tao[1]
Xiaoyong Shen[2] Yangang Ye[2] Jiaya Jia[1,2]
[1]The Chinese University of Hong Kong [2]Tencent Youtu Lab [3]Johns Hopkins University
{ycchen, linhj, ryli, xtao}@cse.cuhk.edu.hk goodshenxy@gmail.com
Mshu1@jhu.edu yangangye@tecent.com leojia9@gmail.com

Abstract

Digital face manipulation has become a popular and fascinating way to touch images with the prevalence of smart phones and social networks. With a wide variety of user preferences, facial expressions, and accessories, a general and flexible model is necessary to accommodate different types of facial editing. In this paper, we propose a model to achieve this goal based on an end-to-end convolutional neural network that supports fast inference, edit-effect control, and quick partial-model update. In addition, this model learns from unpaired image sets with different attributes. Experimental results show that our framework can handle a wide range of expressions, accessories, and makeup effects. It produces high-resolution and high-quality results in fast speed.

1. Introduction

Digital face manipulation aims to change semantically expressive and meaningful attributes, such as smiling and mourning, or add virtual makeup/accessories to human faces, including mustache and eyeglasses. With the increasing popularity of smart phones and digital cameras, the demand of a practical and fast system rises drastically. Face manipulation has attracted great interests in computer vision and graphics [14, 3, 6, 4, 33, 31, 28]. Previous methods devoted to face beautification [19, 8], de-beautification [10], expression manipulation [28] and age progression [14], to name a few.

Given these many solutions, it is common knowledge that different face makeup or attribute changes require special manipulation operations. For example, face beautification or de-beautification processes skin color and texture while face expression manipulation focuses more on 2D or 3D geometry. With this fact, most approaches were particularly designed for individual tasks where any specialization requires expert efforts and domain knowledge to establish new solutions for effect generation.

(a) Original Image (b) Get younger (c) Add facial hair

Figure 1. Illustration of face manipulation using our model.

In the following, we elaborate on the underlying problems of face manipulation when seeking a data-driven framework to unify many face effects. It is followed by introducing the intriguing work to construct our system with this pursuit.

1.1. Possible Solutions and Their Problems

Direct Regression The straightforward way to learn face editing operations from external data is to directly regress the input (before edit) and the ground truth image after edit [10, 5]. However, this procedure requires labeled paired data, which in many cases is unavailable or requires intensive human labor to create. For any effects that do not exist before, these operations cannot be established easily.

Generative Adversarial Network Recently, Generative Adversarial Network (GAN) has shown its capacity in set-

to-set unsupervised learning [36]. It uses a cycle consistent loss to preserve image content, and an adversarial loss to transfer the attribute of one set to another. Although the concept is neat and results are quite phenomenal, it is difficult to train, especially for new effects that require system component modification. The training needs to keep a balance of generation and discrimination. We notice non-optimal training would result in unrealistic manipulation, which could be easily noticeable on visually-sensitive human faces.

Deep Feature Interpolation Deep feature interpolation [29] provides another solution to learn image attribute change from two different sets. It is based on the deep feature of two image sets. However, it is not an end-to-end framework and thus cannot be optimized globally. In addition, it is computation-intensive even during testing because of its heavy involvement of hundreds of face warping and convolution operations.

1.2. Our Solution

We pursue a general, flexible and high-quality-output network for face manipulation. Fig. 1 shows the results generated by our method. Our work follows the an encoder-decoder architecture rather than the popular GAN. Inspired by the Style-Bank [9] that learns replaceable style transfer layers, we propose a Facelet-Bank framework that models face effects with respective middle-level convolutional layers. So interestingly, in order to generate different effects, instead of redesigning the framework completely, only the middle-level convolutional layers need to be updated.

Also, considering lack of ground truth data for many face manipulation tasks, we leverage the result of [29] to produce pseudo targets to learn the facelet-bank layers. The local receptive field of our facelet bank naturally provides regularization, and thus it can capture the correct relation between visual patterns and certain network operations, despite the fact that the pseudo targets are usually noisy.

Finally, we show that these layers can automatically attend to the most important regions so that face manipulation can be performed in an end-to-end fashion. Our method is specially designed to allow users to control the level of effect, and thus it enables interactive face manipulation. Our overall contribution is multifold.

- We propose a set-to-set CNN framework for face manipulation. It does not requires paired data in training.
- The framework is flexible to generate different effects and their levels by simply updating a few convolutional layers, which makes the system developer-friendly.
- Our method naturally benefits from the local prior of convolutional layers, which regularizes noisy labels.
- Experiments show that our approach can handle a wide range of face effects in fast speed.

2. Related Work

Face Editing Our work can be categorized as face editing or manipulation, which has been extensively studied in computer vision and graphics [31, 3, 33, 4, 6, 14]. Traditional face edit includes face relighting [31, 3], expression editing [33], face morphing [4], attribute editing [6], and face aging [14]. However, these models are designed for specific tasks, and rely heavily on domain knowledge. Ours method differs from them as it is a data-driven framework designed to handle general face effects and manipulation.

Image Attribute Manipulation Altering semantic attributes of images is an important topic. Image analogy [11, 7, 30] transfers the attribute or appearance changes from a pair of images to a new one. Recently, generative adversarial networks [12, 36, 25, 27, 22] have become popular in image generation and image-to-image transform. Specifically, the method in [12] learns attribute transformation from two sets of paired data. This idea is further generalized to handle unpaired data [36, 27, 25]. These approaches alter the attributes by training a generator to simulate an online updating domain discriminator. However, when designing and training a system, the balance between generator and discriminator is difficult to deal with. Scaling up image sizes would make training even more difficult. Besides, these approaches do not allow control over the degree of editing, e.g., how bright the face is, which is very important to real-world applications.

Feature Interpolation Our work is also related to feature interpolation [17, 23, 29, 32, 16, 34]. Variational autoencoder (VAE) [17, 34] learns a latent space such that image manipulation can be done with simple arithmetic computation in the deep space. But it implicitly assumes that the target attribute is well isolated. Given unpaired data for face manipulation, this assumption is usually not satisfied. The method of [29] alleviates this problem by proposing a K-NN based approach to isolate the target attribute. Yet only a subset of samples are used for estimation, which lead to undesired visual artifacts in resulted images. In addition, the computational cost is inevitably high, which may be intolerable for fast face manipulation. Our work is different from these approaches. We use convolutional neural networks to learn the shifting direction, so that each transformation is associated with a corresponding visual pattern. We show that the resulting direction is better isolated towards the target attribute. In addition, our method is an end-to-end framework that supports fast inference.

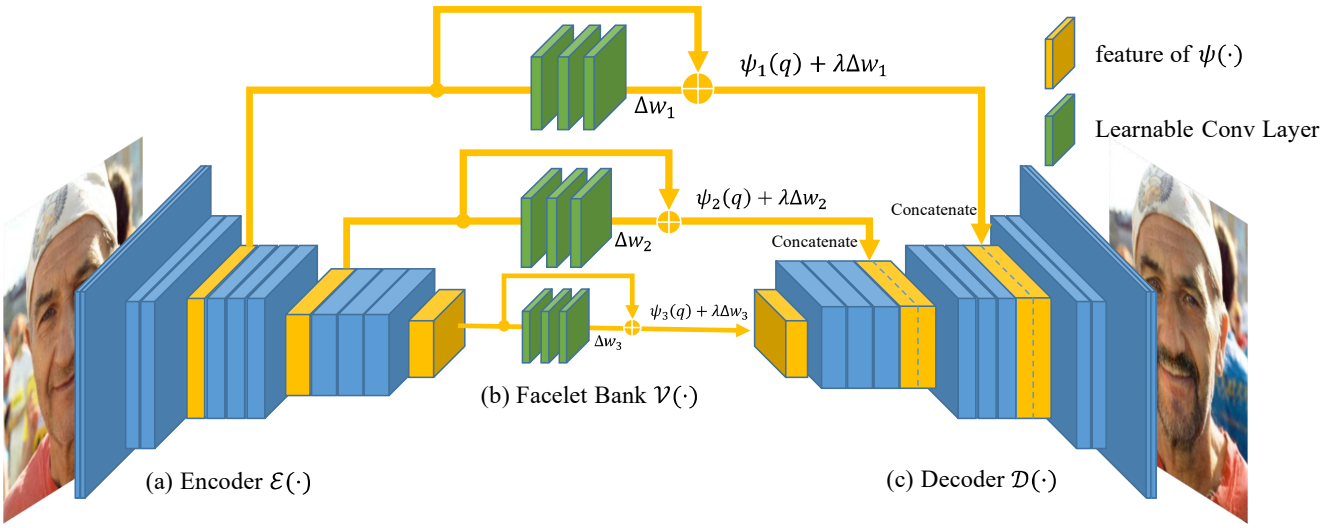


Figure 2. Illustration of our framework. (a) is the encoder $\mathcal{E}(\cdot)$; (b) are convolutional layers of our facelet bank $\mathcal{V}(\cdot)$; (c) is the decoder $\mathcal{D}(\cdot)$. The structure of our facelet bank is Conv-ReLU-Conv-ReLU-Conv, where all Convs are with 3×3 kernels. Also, all Convs of the facelet bank do not change the height, width and number of channels given the previous input.

3. Proposed Method

Suppose there are two face domains $\mathcal{X}$ and $\mathcal{Y}$ with different properties. Our goal is to shift images of domain $\mathcal{X}$ towards domain $\mathcal{Y}$ without guidance from any paired data. As for face manipulations, we want the algorithm to produce intermediate results so that users can control the strength of operations, such as smoothness of the skin.

This process can be represented as

$$z = \mathcal{O}(x, \lambda), \tag{1}$$

where x and z are the input and output images, λ controls the strength, and $\mathcal{O}$ denotes the operation that changes x from domain $\mathcal{X}$ toward domain $\mathcal{Y}$.

Note that domains $\mathcal{X}$ and $\mathcal{Y}$ usually differ *semantically*, so $\mathcal{O}$ can be highly complex. In order to simplify $\mathcal{O}$, we instead define the operation $\mathcal{O}'$ in a deep space, and change Eq. (1) to

$$\psi(z) = \mathcal{O}'(\psi(x), \lambda), \tag{2}$$

where $\psi(\cdot)$ denotes the deep space. To one extreme, if $\psi(\cdot)$ captures enough rich semantic information, $\mathcal{O}'$ can be further simplified as linear shift in the deep space [20], which leads to

$$\psi(z) = \psi(x) + \lambda\Delta v, \tag{3}$$

where Δv denotes the direction from domain $\mathcal{X}$ to domain $\mathcal{Y}$ in the deep space.

As indicated in [35], a network trained on large-scale data captures semantic attributes in the convolutional layers. This idea is validated in the work of [20, 29], which uses a VGG network [26] to encode the semantic information for deep feature interpolation or style transfer. We follow this idea to use the 5 convolutional layers to construct $\psi(\cdot)$. To represent face attributes at different levels, ReLU3_1, ReLU4_1 and ReLU5_1 are jointly used to represent $\psi(\cdot)$.

Reversing $\psi(\cdot)$ Following [20, 9], we train a fixed decoder network $\mathcal{D}$ to decode $\psi(z)$ to obtain the final output. The decoder $\mathcal{D}$ has reverse architecture of the encoder $\mathcal{E}$, except for concatenating the manipulated features to the corresponding layers. We use the following loss function to train $\mathcal{D}$:

$$\mathcal{L} = \sum_{i=1}^{n} ||z_i - x_i||_2^2 + \omega \sum_{i=1}^{n} ||\mathcal{E}(x_i) - \mathcal{E}(\mathcal{D}(\mathcal{E}(x_i)))||_2^2, \tag{4}$$

where the first and second terms impose consistencies in pixel and feature space respectively, and ω is the weight to balance the two losses. We also observe that pre-training of the decoder with $\mathcal{L} = \sum_{i=1}^{n} ||(\mathcal{E}(x_i) + \lambda\Delta v_i) - \mathcal{E}(\mathcal{D}(\mathcal{E}(x_i) + \lambda\Delta v_i))||_2^2$ is helpful, where Δv_i are pseudo labels of different attributes computed with Eq (6).

Overall Structure The overall network architecture is illustrated in Fig. 2, which is composed of an encoder $\mathcal{E}$ that transforms images to a deep space, a ConvNet $\mathcal{V}$ that estimates the domain direction shift Δv, and a decoder that transforms the operated deep feature back to an image. With this effective pipeline, the encoder and decoder are responsible for general operations, while $\mathcal{V}$ is the key component that determines the specific effect required for a face manipulation.

Now, realizing differences across all types of face effects, from a smile on the face to adding mustache, no

longer needs frequent redesign of the framework. By using different $\mathcal{V}$, these apparently dissimilar face operations can be accomplished accordingly. Thus, we name the collection of $\mathcal{V}$ as **Facelet Bank**, representing the collection of different face effects we can achieve.

3.1. Learning Facelet Bank

Estimating Δv requires cancellation of all factors except for the target attributes. Without paired data, a straightforward approach is to compute the average difference between $\mathcal{X}$ and $\mathcal{Y}$ [17, 34], namely

$$\Delta v \leftarrow \frac{1}{m}\sum_{i=1}^{m}\psi(y_i) - \frac{1}{n}\sum_{i=1}^{n}\psi(x_i), \tag{5}$$

where m and n are the training samples of domains $\mathcal{X}$ and $\mathcal{Y}$ respectively. However, this implies a strong assumption that all training samples have similar attributes to the queried one. When this assumption is not satisfied, the model does not produce realistic results.

Generating Pseudo Labels We relax this assumption by adopting a query-adaptive attribute transform model $\mathcal{V}(x)$, where x is a query sample.

Note that learning $\mathcal{V}(\cdot)$ is difficult as we do not have paired-data to infer Δv for each training sample. Inspired by the method in [29], we average neighbors of each training sample of *both* domains to construct the attribute vector. Specifically, for sample x, the corresponding Δv is computed as

$$\Delta v^*(x) = \frac{1}{K}\sum_{i\in N_{\mathcal{Y}}^K(x)}\psi(y_i) - \frac{1}{K}\sum_{i\in N_{\mathcal{X}}^K(x)}\psi(x_i), \tag{6}$$

where $N_y^K(x)$ and $N_x^K(x)$ refer to K nearest neighbors of x among sets $\mathcal{Y}$ and $\mathcal{X}$ respectively. To reduce the influence of pose, viewpoint and rotation, all training face images are aligned to a frontal face template in advance[1]. Here the "average" operation suppresses undesired noise and thus Δv^* tends to cancel out other factors except those that divide domains $\mathcal{X}$ and $\mathcal{Y}$.

Network Design Since Δv^* relates strongly to the deep feature, we reuse the extracted deep feature by setting the input of $\mathcal{V}(\cdot)$ to $\mathcal{V}(\mathcal{E}(q))$. We stack 3 fully convolutional layers with ReLU activation to capture the non-linearity relation. In this way, $\mathcal{V}$ can be learned as

$$\mathcal{L}_{\mathcal{V}} = \sum_{i=1}^{n}||\mathcal{V}(\mathcal{E}(x_i)) - \Delta v^*(x_i)||_2^2, \tag{7}$$

where x_i refers to training samples of set $\mathcal{X}$. $\Delta v^*(x_i)$ is defined in Eq. (6).

[1] A face alignment tool [13] in DLIB [2] is used in our experiment.

3.2. More Analysis of the Facelet Bank

Although our facelet bank learns with the help of pseudo paired data Δv^* defined in Eq. (7), it has additional advantages compared with directly using Δv^*.

Noise Resistance In practice, our facelet bank can even outperform the pseudo label Δv^*. To demonstrate it, we compute the heat map of Δv as

$$H_{i,j} = \sum_{k=1}^{c} v_{i,j,k}^2, \tag{8}$$

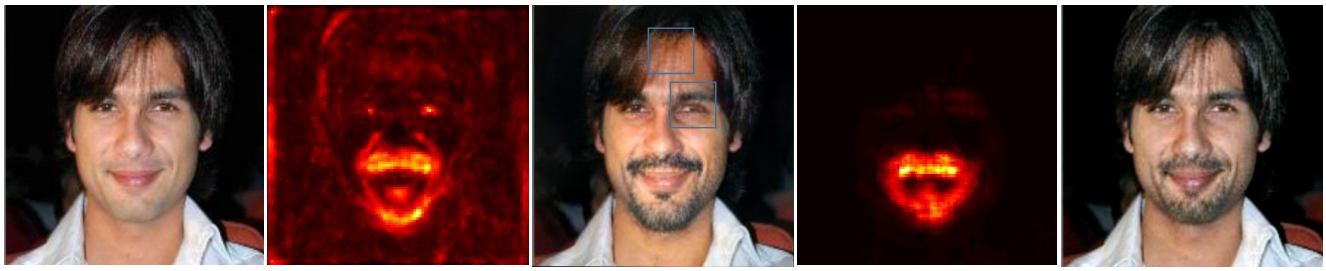
where $i = \{1, 2, \cdots, h\}$, $j = \{1, 2, \cdots, w\}$, and $k = \{1, 2, \cdots, c\}$ denotes the row, column and channel indexes respectively. Fig. 3 shows the heat map of "add beard" operation. Intuitively, adding beard is only relevant to the mouth region, while other place should not be updated. However, the pseudo target Δv^* activates in many regions as shown in the heat map. This may cause undesirable changes in wrong places as shown in Fig 3(c). Note that the pseudo label is computed by only a subset of samples[2], not sufficient to cancel out all noise. On the contrary, our facelet bank has a much cleaner heat map, which only activates in the correct region.

Although this may look counterintuitive at the first glance, it actually makes sense if we consider the built-in regularization of the convolutional neural network. Note that CNN has local receptive fields, which force the system to capture the relation between certain visual patterns (the mouth in this case) and the corresponding manipulation operations ("add beard" in this case). By training on a large set of samples, unrelated activations can be well suppressed, and only the relevant regions are activated.

Implicit Attention Mechanism Another advantage of our Facelet Bank is the implicit attention mechanism. Note that computing the pseudo target requires faces to be aligned in advance. This is not needed for our facelet bank during testing. As shown in Fig. 4, the trained operations are adaptive to the position and pose of the face, and keep staying in the correct region. This is another major advantage stemming from the fully convolution architecture. Essentially, convolution layers of our facelet bank capture the relation of certain visual patterns and corresponding Δv, which are not dependent of the position. As a result, the facelet bank is capable of detecting certain patterns (e.g., mouth for beard), and adding the effects accordingly.

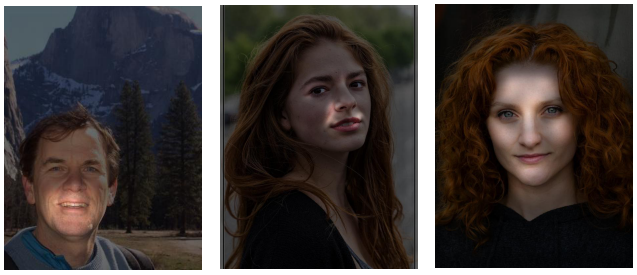
Speed Calculating Δv^* with Eq. (3) is computationally expensive. The cost is mostly on face warping, deep feature

[2] $2K = 200$ samples are used in our method, which is less than 1% compared with the training set.



(a) Original Image (b) Heat Map of $\Delta v^*(x)$ (c) Result of $\Delta v^*(x)$ (e) Heat Map of $\mathcal{V}(x)$ (e) Result of $\mathcal{V}(x)$

Figure 3. Illustration of the noise resistance effect. (a) The original image. (b) The heat map of pseudo shifting direction computed by Eq. (3). The blue rectangles mark the undesired change regions. (c) Corresponding result of Eq. (3). (d) Heat map of our estimated direction shift. (e) Our result.



(a) beard (b) smile (c) age

Figure 4. Visualization of the attention region. The attention mask is computed by Eq. (8). (a), (b) and (c) correspond to the operations of adding beards, making smiles and changing age. Note that for the beard effect, our facelet bank focuses on the mouth area. For the smile effect, it attends to smile-related facial muscles. As for the age changing effect, the attention region covers the whole face. These results match our intuition.

extraction and nearest-neighbor search. The total testing time for a 448×448 image is 1.09 minutes with the system implemented with Pytorch, and running on a server with a Titan XP GPU and an E5-2623 v4 CPU. But for our Facelet Bank, computing $\mathcal{V}(x)$ only requires network forwarding, which costs only 0.0194 second.

Discussion Compared with cycleGAN [36], our framework has the following advantages. First, it is very easy to train. Compared with cycleGAN that jointly optimizes two generators and two discriminators, our system only needs to learn the convolutional layers of facelet bank, which is both easier and faster.

Second, our method can handle higher-resolution images. Note that the original cycleGAN has demonstrated its effectiveness on 256×256 images, whereas our method handles much higher resolutions, e.g., 640×480. Third, our work allows change of operation strength of effect conveniently by modifying the value of λ in Eq. (3). In contrast, changing the effect strength of cycleGAN requires the use of different training data and re-training of the model. Another related approach is DFI [29], which needs no training and handles the highest resolution input data. Compared with DFI [29], our approach is much faster during testing because it does not need to perform face alignment and KNN search in the deep space.

4. Experiments

Implementation Details We implement our model using PyTorch [1]. Before training the facelet bank, we train our decoder with Adam optimizer [15] with an initial learning rate of 0.0001 and step weight decay. After training, the decoder is fixed. The face effects are trained separately. Each corresponds to a set of convolution layers of the facelet bank. Adam optimizer with default hyper-parameters are used to train the facelet-bank layers.

Dataset Celeba [21] is a large face dataset that contains 202,599 images belonging to 10,177 identities. We randomly sample 90% for training and the rest are used for testing. The landmark and attribute information is used for computing the pseudo label Δv^* computed by Eq. (3). But it is not used during testing. In addition to the Celeba data, we also use the Portrait [24] and Helen [18] datasets for testing. The images of [24] and [18] are collected on Flickr with varying ages, skin colors, clothing, hair styles, etc. We use them to validate the cross-dataset generalization ability.

4.1. Evaluation of Our Approach

The vital part of our approach is the facelet bank. In the following, we split evaluation of this design into three parts, i.e., effectiveness of the facelet bank, reason to use multi-layer aggregation, and flexibility in edit strength control.

4.1.1 Effectiveness of Facelet Bank

Estimation of attribute direction shift Δv is critical to the result of manipulation. We compare our facelet-bank solution with several baseline approaches. The simplest way to estimate Δv is to average both positive and negative samples, as the methods of [17, 23, 32, 34]. As shown in Fig.

(a) Original Image (b) Global Mean (c) K-NN Mean (d) Ours

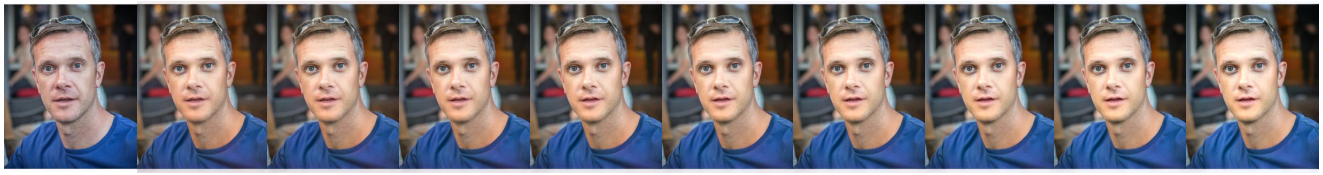
Figure 5. Comparing our facelet-bank method with baseline approaches. Please zoom in to see details.

(a) Original Image (b) Layer 5 Only (c) Layer 4 + 5 (d) All 3 Layers

Figure 6. Results of removing facial hair. (a) the original image; (b), (c) and (d) are the results of using layer 5, layer 5+layer 4 and all three layers respectively. Please zoom in to see details.

(a) Facial hair

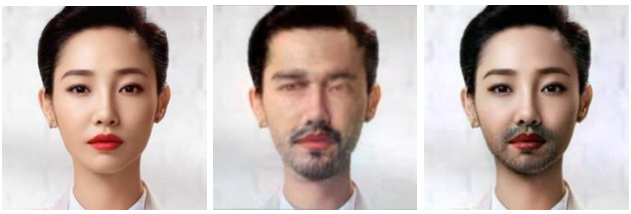
(b) Smile



(c) Younger

Figure 7. Illustration of different edit strength. (a), (b) and (c) show the results of different edit strength respectively.

Figure 8. Comparing with CycleGAN [36] and DFI [29].



(a) Original Image (b) DFI (c) Ours

Figure 9. Adding uncommon facial hair on a woman face; (a) is the original image; (b) is the result of DFI [29]; (c) is our result.

5, the beard position can be wrong since the globally computed Δv is not adaptive to the query face. If the face pose is far from the average of the dataset, the method fails inevitably. Directly applying Δv^* computed by Eq. (6) alleviates this problem. However, in this case, the skin color is changed since color factor is not fully canceled out by the query's nearest neighbors. Our method learns the relation between semantic visual pattern (mouth in this case) and the corresponding beard add-up operation, which cause small influence on the other regions.

4.1.2 Effectiveness of Multi-layer Aggregation

It was shown that different levels of layers are complementary to each other [29]. Therefore, using all of them jointly yield better results. Fig. 6 shows the result using different layers for the "remove facial hair" task. Using only layer 5 removes a large proportion of facial hair, but not complete. By incorporating layer 4 and layer 3, the amount of facial hair gradually decreases. It is eventually removed completely with all 3 layers.

4.1.3 Different Operation Strength

Our model provides a fast and convenient way to control the strength of operations. Both deep feature $\mathcal{E}(x)$ and the estimated attribute direction shift Δv are computed only once. After that, changing the strength only requires to forward $\mathcal{E} + \lambda \Delta v$ to the decoder network. This takes only 9ms on our server. We believe that it can similarly achieve a high speed on recent high-end mobile devices.

We show results of controlling different edit strength in Fig. 7. For each case, we use strength of 0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, and 1.8 to edit the original image. Fig. 7 shows the resulting images accordingly. When $\lambda > 1$, the edit can be seen as extrapolation rather than interpolation. This adds stronger edit to the original image, but sometimes yields unnatural results.

4.2. Comparison with State-of-the-Arts

Fig. 8 shows the overall comparison with deep feature interpolation (DFI) [29] and cycleGAN [36], which also perform set-to-set image attribute transform. Three effects are tested, which includes smiling, adding facial hair, and getting younger. Generally, our method can achieve better results than those of CycleGAN where the results contain stronger effect without introducing too many visual artifacts.

Compared with the DFI, our approach achieves better performance on local effects (e.g., facial hair), since our approach introduces less irrelevant changes. For other effects, the two methods perform similarly. In terms of the amount of computation, our method is much lighter than DFI. It does not require face warping, nearest-neighbor computation and backward optimization. As a result, it is 3,371 times faster than DFI with running time of 0.0194 second and 65.4 seconds respectively. cycleGAN takes 0.0185 second for one direction, which is slightly faster than our approach.

Disentangling Correlated Attributes We compare the results of "adding beard to woman" in Fig. 9. Intriguingly, our model does a much better job than DFI. It is noticeable that DFI changes the overall appearance of the resulting face, while our approach maintains most of the subject's feminine characteristics. This is because DFI intrinsically is an instance-based method, which relies heavily on training samples. Since "beard" always comes with man's face features, it cannot disentangle the different attributes.

In contrast, our method captures the relation between "beard" and "mouth". Since the former is a masculine trait and the latter can be found in both genders, our model naturally disregards other similar relations, including the comparison of "beard" with "gender". Thus in the above example, albeit uncommon, our model yields better performance.

5. Concluding Remarks

In this paper, we have proposed a general framework for face manipulation. Our framework can learn face attribute shift from two image sets without any paired-example information. Thus, it does not need intensive human efforts for labeling. In addition, our framework is highly flexible – each operation is related to only a few computed convolutional layers. We have proved in our experiments that this approach yields superior results compared with other set-to-set image translation models.

References

[1] http://pytorch.org. 5

[2] https://pypi.python.org/pypi/dlib. 4

[3] V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. In *CGF*, 2003. 1, 2

[4] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Ann. Conf. Comput. Graph. Interactive Tech.*, 1999. 1, 2

[5] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv*, 2016. 1

[6] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3d facial expression database for visual computing. *TVCG*, 2014. 1, 2

[7] T. Cao, C. Zach, S. Modla, D. Powell, K. Czymmek, and M. Niethammer. Registration for correlative microscopy using image analogies. In *WBIR*, 2012. 2

[8] C.-W. Chen, D.-Y. Huang, and C.-S. Fuh. Automatic skin color beautification. In *Int. Conf. Arts Technol.*, 2009. 1

[9] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. *arXiv*, 2017. 2, 3

[10] Y.-C. Chen, X. Shen, and J. Jia. Makeup-go: Blind reversion of portrait edit. In *ICCV*, 2017. 1

[11] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Ann. Conf. Comput. Graph. Interactive Tech.*, 2001. 2

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv*, 2016. 2

[13] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 2014. 4

[14] I. Kemelmacher-Shlizerman, S. Suwajanakorn, and S. M. Seitz. Illumination-aware age progression. In *CVPR*, 2014. 1, 2

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. 5

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv*, 2013. 2

[17] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv*, 2015. 2, 4, 5

[18] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang. Interactive facial feature localization. In *ECCV*, 2012. 5

[19] T. Leyvand, D. Cohen-Or, G. Dror, and D. Lischinski. Digital face beautification. In *Siggraph*, 2006. 1

[20] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. *arXiv*, 2017. 3

[21] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 5

[22] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible conditional gans for image editing. *arXiv*, 2016. 2

[23] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 2015. 2, 5

[24] X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia. Deep automatic portrait matting. In *ECCV*, 2016. 5

[25] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras. Neural face editing with intrinsic image disentangling. In *CVPR*, 2017. 2

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. 3

[27] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *arXiv*, 2016. 2

[28] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt. Real-time expression transfer for facial reenactment. *TOG*, 2015. 1

[29] P. Upchurch, J. Gardner, K. Bala, R. Pless, N. Snavely, and K. Weinberger. Deep feature interpolation for image content changes. *arXiv*, 2016. 2, 3, 4, 5, 7, 8

[30] G. Wang, T.-T. Wong, and P.-A. Heng. Deringing cartoons by image analogies. *TOG*, 2006. 2

[31] Y. Wang, L. Zhang, Z. Liu, G. Hua, Z. Wen, Z. Zhang, and D. Samaras. Face relighting from a single image under arbitrary unknown lighting conditions. *TPAMI*, 2009. 1, 2

[32] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016. 2, 5

[33] F. Yang, J. Wang, E. Shechtman, L. Bourdev, and D. Metaxas. Expression flow for 3d-aware face component transfer. *TOG*, 2011. 1, 2

[34] R. Yeh, Z. Liu, D. B. Goldman, and A. Agarwala. Semantic facial expression editing using autoencoded flow. *arXiv*, 2016. 2, 4, 5

[35] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 3

[36] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv*, 2017. 2, 5, 7, 8