Makeup-Go: Blind Reversion of Portrait Edit*

Ying-Cong Chen[1] Xiaoyong Shen[2] Jiaya Jia[1,2]
[1]The Chinese University of Hong Kong [2]Tencent Youtu Lab
ycchen@cse.cuhk.edu.hk dylanshen@tencent.com leojia9@gmail.com

Abstract

Virtual face beautification (or markup) becomes common operations in camera or image processing Apps, which is actually deceiving. In this paper, we propose the task of restoring a portrait image from this process. As the first attempt along this line, we assume unknown global operations on human faces and aim to tackle the two issues of skin smoothing and skin color change. These two tasks, intriguingly, impose very different difficulties to estimate subtle details and major color variation. We propose a Component Regression Network (CRN) and address the limitation of using Euclidean loss in blind reversion. CRN maps the edited portrait images back to the original ones without knowing beautification operation details. Our experiments demonstrate effectiveness of the system for this novel task.

1. Introduction

Popularity of social networks of Facebook, Snapchat, and Instagram, and the fast development of smart phones make it fun to share portraits and selfies online. This trend also motivates the development of widely used procedures to automatically beautify faces. The consequence is that many portraits found online are good looking, but not real. In this paper, we propose the task of blind reverse of this unknown beautification process and restore faces similar to what are captured by cameras. We call this task *portrait beautification reversion*, or *makeup-go* for short.

Scope of Application Virtual beautification is performed quite differently in software, which makes it impossible to learn all operations that can be performed to smooth skin, suppress wrinkle and freckle, adjust tone, to name a few. To make the problem trackable in a well-constrained space, we learn from data the automatic process for particular software and then restore images output from it without knowing the exact algorithms. Also, we assume subtle cues still exists after virtual beutification, even if they are largely suppressed. Also, at this moment, we do not handle geometric transformation for face-lift.

Difficulty of Existing Solutions Existing restoration work [36, 32, 28, 30] only handles reversion of *known and/or linear* operations. There is no mature study yet what if several unknown nonlinear operations are coupled for restoration. The strategy of [11] is to infer the sequence of operations and then each of them. We however note this scheme does not work in our task for complicated beautification without knowing candidate operations.

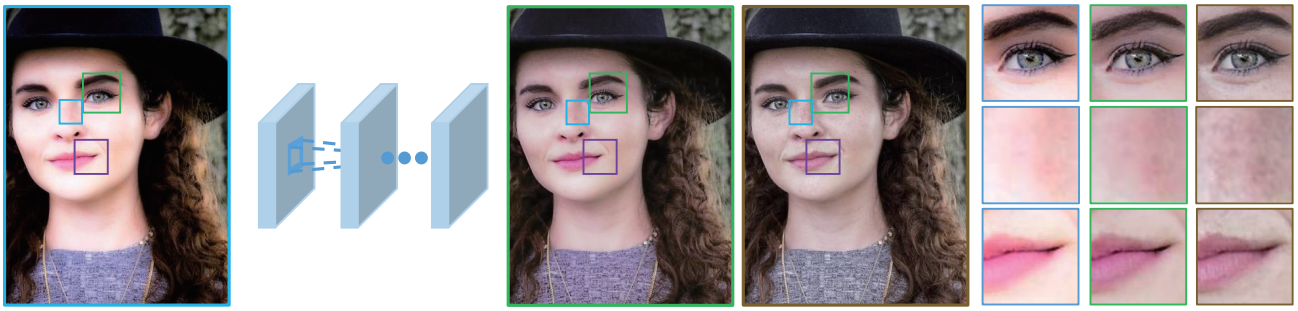
In addition, directly learning reversion by deep neural networks is not trivial. Previous CNN frameworks only address particular tasks of super-resolution [7, 8, 13], noise/artifact removal [6], image filtering [18], *etc*. Learning in these tasks are with some priors, such as the upsampling factors, noise patterns, and filter parameters. Our task may not give such information and needs to be general.

In terms of network structures, most low-level vision frameworks stack layers of convolution and nonlinear rectification for regression of the Euclidean loss. This type of loss is useful for previous tasks such as denoise [6]. But they are surprisingly not applicable to our problem because levels of complex changes make it hard to train a network. We analyze this difficulty below.

Challenge in Our Blind Reversion To show the limitation of existing CNN in our task, we apply state-of-the-art VDSR [14], FSRCNN [8] and PSPNet [8], to directly regress the data after edit. VDSR performs the best among the three. It is an image-to-image regression network with L_2 loss that achieves high PSNRs in super-resolution. It successfully learns large-scale information such as skin color and illumination, as shown in Figure 1.

It is however intriguing to note that many details, especially the region that contains small freckles and wrinkles, are not well recovered in the final output. In fact, VDSR is already a deep model with 20 layers – stacking more layers in our experiments does not help regress these subtle information. This manifests that the network capacity is not the bottleneck. The main problem, actually, is on employment of the Euclidean loss that makes detailed changes be

*This work is in part supported by a grant from the Research Grants Council of the Hong Kong SAR (project No. 413113).



(a) Edited Image (b) CNN (c) Output (d) Groundtruth (e) Close-up

Figure 1. Illustration of using existing CNNs for portrait beautification reversion. (a) is the image edited by Photoshop Express; (b) represents a CNN network for image regression; (c) is the output of the network; (d) is the ground truth image; (e) is the close-up patches cropped from the edited image, output and ground truth image. It cannot achieve good performance on face detail recovery.

ignored. This is not desirable to our task since the main objective is *exactly* to restore these small details on faces. Ignoring them would completely fail operation reversal. We provide more analysis in Section 3, which gives the idea of constructing our Component Regression Network (CRN).

Our Contributions We design a new network structure to address above elaborated detail diminishing issues and handle both coarse and subtle components reliably. Instead of directly regressing the images towards the unedited version, our network regresses levels of principal components of the edited parts separately. As a result, subtle detail information would not be ignored. It can now be similarly important as strong patterns in final image reconstruction, and thus is guaranteed to get lifted in regression. Our contribution in this paper is threefold.

- We propose the task of general blind reversion for portrait edit or other inverse problems.
- We discover and analyze the component domination effect in blind reversion and ameliorate it with a new component-based network architecture.
- Extensive analysis and experiments prove the effectiveness of our model.

2. Related Work

We in this section review related image editing techniques and the convolutional neural networks (CNNs) for image processing.

Image Editing Image editing is a big area. Useful tools include those for image filtering [19, 3, 9, 29, 35, 10], retargeting [23, 1], composition [21], completion [2, 5], noise removal [22, 27], and image enhancement [33]. Most of these operations are not studied for their reversibility. Only the work of [11] recovers image editing history assuming that candidate editing operations are known in advance, which is not applicable to our case where unknown edit comes out from commercial software.

CNNs for Image Regression Convolutional neural networks are effective now to solve the regression problem in image processing [8, 14, 7, 30, 17, 31]. In these methods, the Euclidean loss is usually employed. In [31], Euclidean loss in gradient domain is used to capture strong edge information for filter learning. This strategy does not fit our problem since many details do not contain strong edges.

Our method is also related to the perception loss based CNNs [13, 16], which are recently proposed for style transfer and super-resolution. In [13], perception loss is defined as the Euclidean distance between features extracted by different layers of an ImageNet [15] pretrained VGG network [25]. The insight is that layers of the pretrained network capture levels of information in edges, texture or even object parts [34]. In [16], an additional adversarial part is incorporated to encourage the output to reside on the manifold of target dataset. However, as explained in [13, 16], perceptional loss produces lower PSNR results than the Euclidean one. Since the perception loss is complex, it is difficult to know what component is not regressed well. Our approach does not use this loss accordingly.

3. Elaboration of Challenges

As shown in Figure 1, our blind reversion system needs to restore several levels of details. When directly applying the methods of [14], many details are still missing even if PSNRs are already reasonable, as shown in Figure 1(c,e). We intriguingly found that the main reason is on the choice of Euclidean loss function [8, 14, 7, 30, 17] for this regression task, which is used to measure the difference between the network output and target. Perception loss [13, 16], alternatively, is used for remedying details at the cost of sacrificing PSNRs, i.e., overall fidelity to the target images.

In this section, we explain our finding that Euclidean loss

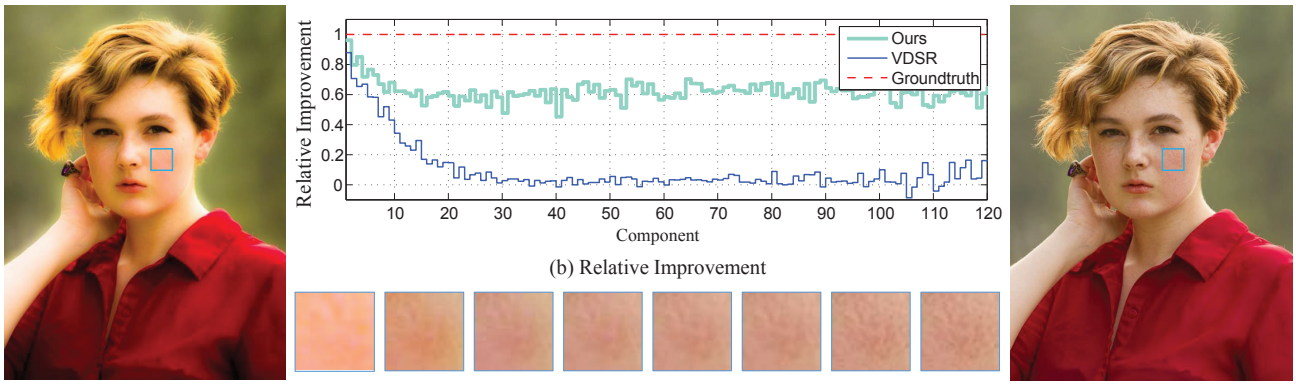


Figure 2. Relative improvement of different principal components. (a) and (d) are the edited and ground truth images respectively. (b) shows the relative similarity improvement of different principal components regarding network output. (c) shows close-up patches with appearance change using only the top 1, 2, 3, 5, 10, 20, 50 and 100 components. Best viewed in color.

cannot let network learn details well due to the *component domination effect*. We take the VDSR network [14] as an example, which uses Euclidean loss and achieves state-of-the-art results in super-resolution. So it already has the good ability for detail regression. When applied to our task, it does not produces many details, as illustrated in Figure 1.

Improvement Measure Regarding Levels of Details To understand this process, we analyze the difference between the network output and the ground truth image regarding different principal components, which is shown in Figure 2. Our computation is explained below.

We denote the edited input and unedited ground truth image as I_X and I_Y, and the network output as I_Z. We compute the discrepancy maps as $I_e = I_Y - I_X$ and $I_{\hat{e}} = I_Y - I_Z$ on collected 10^7 patches where each patch is with size 11×11. These patches are vectorized where PCA coefficients $U = \{u_1, u_2, \cdots, u_{m^2}\}$ are extracted from them in a *descending* order. Then $e_{ij} = u_i^T v_j$ and $\hat{e}_{ij} = u_i^T \hat{v}_j$ denote the i^{th} components of the j^{th} discrepancy patches, where v_j and $\hat{v}_j$ are the vectorized patches. The final improvement measure of the i^{th} PCA component between the network output and the ground truth image is expressed as

$$d_i = \frac{\sum_j (e_{ij}^2 - \hat{e}_{ij}^2)}{\sum_j e_{ij}^2}. \tag{1}$$

When the network output is exactly the ground truth, $d_i = 1$ for all i, which is the red dash line in Figure 2(b). A large d_i indicates that the network regresses the i^{th} component well.

Relative Improvement Analysis As shown in Figure 2 (the cyan curve), VDSR achieves high d_i for the top-ranking components. Intriguingly the performance drops dramatically for other smaller ones. As shown in Figure 2(c), the lower-ranking components mostly correspond to details of faces, such as freckles. Their unsuccessful reconstruction greatly affects our blind reversion task where details are the vital visual factors.

The main reason that VDSR fails to regress low-ranking components is the deployment of the Euclidean loss, which is mainly controlled by the largest principal components. This loss is expressed as

$$J = \frac{1}{2m^2} ||F(v_X) - v_Y||^2, \tag{2}$$

where $v_X \in \mathbb{R}^{m^2 \times 1}$ is the input edited image patch, $F(\cdot) \in \mathbb{R}^{m^2 \times 1}$ is mapping function defined by the network, and $v_Y \in \mathbb{R}^{m^2 \times 1}$ is the target image patch.

Note that the PCA base matrix U satisfies $UU^T = I$, where I is an identity matrix. Thus projecting $F(v_X)$ and v_Y to U does not change the loss J, i.e.,

$$\begin{aligned} J &= \frac{1}{2m^2} ||U^T F(v_X) - U^T v_Y||^2 \\ &= \frac{1}{2m^2} \sum_{i=1}^{m^2} ||f_i(v_X) - u_i v_Y||^2, \end{aligned} \tag{3}$$

where $f_i(v_X) = u_i^T F(v_X)$. In this regard, J can be viewed as a sum of different objectives that regress components of the target image patches. Since these components are extracted by PCA, there is limited information shared among different principal components.

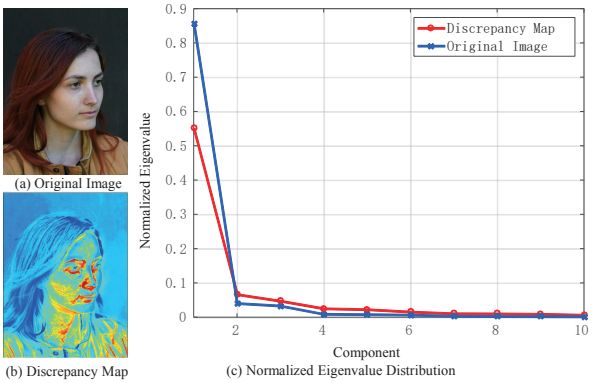
Note that the variance of those components differs greatly. As shown in Figure 3, the eigenvalues for the first a few components are much larger than others and the top 5 components contribute more than 99% variance. This makes the Euclidean loss function dominated by the major ones, while other components can be safely ignored during regression, causing the *component domination effect*.

In the following, we describe our model to remedy this problem. As shown in Figure 8, it can yield much better performance in reconstructing face details, such as wrinkles.

4. Our Framework

The pipeline of our network is shown in Figure 4. To recover details on faces, first, we learn the discrepancy map $I_e = I_Y - I_X$. It makes final image reversion achieved as $I_Y = I_e + I_X$. It is similar to the residual learning defined in [14], which originally aimed to address the gradient vanishing/explosion problem. We note that our network does not suffer greatly from such gradient problems. Instead it is vital to mitigate the component domination effect.

The reason to learn the discrepancy map I_e instead of the image revision I_Y directly is illustrated in Figure 3. The first principal component of I_Y is more dominating than that of I_e, while other components are much less significant. This indicates that subtle details in I_Y are more difficult to learn than those in I_e. Our choice to work on I_e makes the system run more reliably.



(a) Original Image
(b) Discrepancy Map
(c) Normalized Eigenvalue Distribution

Figure 3. Illustration of normalized eigenvalues regarding principal components. (a) and (b) illustrate an example of original image and discrepancy map respectively. (c) compares the distributions between the ground truth image and the discrepancy map. For fair comparison, we normalize eigenvalues by their L_1-norm, and show only the top 10 eigenvalues.

4.1. Component Decomposition

Our component decomposition is conducted in the vector space where the i^{th} component is represented as

$$v_y{}^{(i)} = u_i u_i^T v_y. \tag{4}$$

This operation can be further transformed into a simple convolution operation [4] in the image space, allowing to input a whole image without any cropping and vectorization operations. This profits computation since both the input and target are in the image space instead of the vector one.

More specifically, each basis $u_i \in \mathbb{R}^{m^2}$ can be rearranged into $\kappa_i \in \mathbb{R}^{m \times m}$, so that $u_i^T v_y$ ($v_y \in \mathbb{R}^{m^2}$) is equivalent to $\kappa_i * y$ ($y \in \mathbb{R}^{m \times m}$) where $*$ is a convolutional operation, and y is the image patch. Similarly, $u_i u_i^T v_y$ is equivalent to $\kappa_i' * \kappa_i * y$, where κ_i' is 180° rotation of κ_i. This convolution implementation makes decomposition extensible to the whole image. Given an image $I \in \mathbb{R}^{h \times w}$ where h and w are the height and width respectively, decomposition is implemented as

$$I^{(i)} = \kappa_i' * \kappa_i * I, \tag{5}$$

where $I^{(i)}$ is the i^{th} component. Note that this decomposition is invertible – that is, we can recover I as

$$I = \sum_{i=1}^{n} I^{(i)}. \tag{6}$$

4.2. Component-Specific Learning

Once the target image is decomposed, subnetworks can be used for regressing components respectively. This procedure makes the major components not dominate optimization during learning. Therefore, our framework is to give each subnetwork a specific loss function that drives regression only in the corresponding component rather than the whole image mixing all of them. The final image can be reconstructed by summing all outputs of subnetworks, according to Eq. (6). This framework is called Component Regression Network (CRN).

The implementation detail is as follows. Given an input image I_X, we first process it with 3 convolutional layers (3×3 kernel size and 56 channels) with PReLU rectification, which aims for extracting common features among all components. We pad zeros for all convolution layers to preserve the image size. Then the common features are fed into each component-specific network. The architecture of the subnetworks, like [8], contains the following parts.

Shrinking We use 1 convolution kernel for reducing the number of feature maps from 56 to 12 for accelerating the mapping process. This reduces network parameters and accelerates both training and testing.

Nonlinear mapping We stack 3 convolution layers with PReLU rectification for nonlinear mapping. Each convolutional layer contains 12 3×3 kernels.

Expending Reconstruction is the reverse of dimension reduction. We use the 1 convolution kernel to expand the shrunk feature maps back to 64 channels for final process.

4.3. Other Design Details

In addition to addressing the component domination problem, we incorporate a few other techniques to further improve the performance and efficiency.

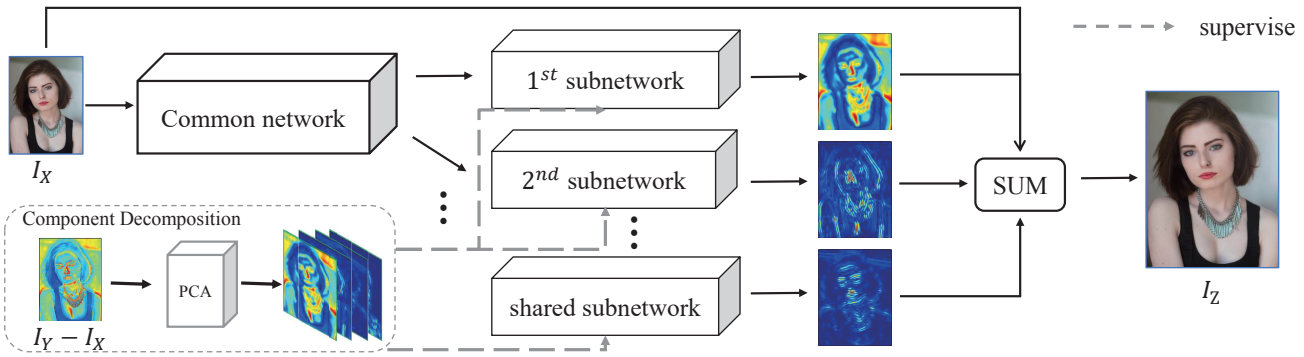


Figure 4. Our CRN Pipeline. CRN learns the discrepancy map $I_e = I_Y - I_X$. It is further decomposed into different components according to Eq. (5) to supervise each subnetwork. During testing, we feed the network with an edited image where each subnetwork outputs the corresponding component. We sum network output to obtain the final result.

(a) Portrait samples (b) Effect of edit

Figure 5. Examples in our dataset. (a) shows a few samples in our dataset. (b) shows the effect of edit. The 4 images are the original image, and images edited by MT, PS and INS respectively.

Subnetwork Sharing One critical factor that influences performance is the number of subnetworks. Ideally, it should equal to the number of pixels of an image patch. However, allocating one subnetwork to each component is not affordable considering the computation cost. As illustrated in Figure 3, very low-ranking components are with similar variance. This finding inspires us to empirically use only 7 subnetworks to regress the top 7 components. There is one last *shared subnetwork* to regress all remaining components. This strategy works quite well in our task. It saves a lot of computation while not much affecting result quality.

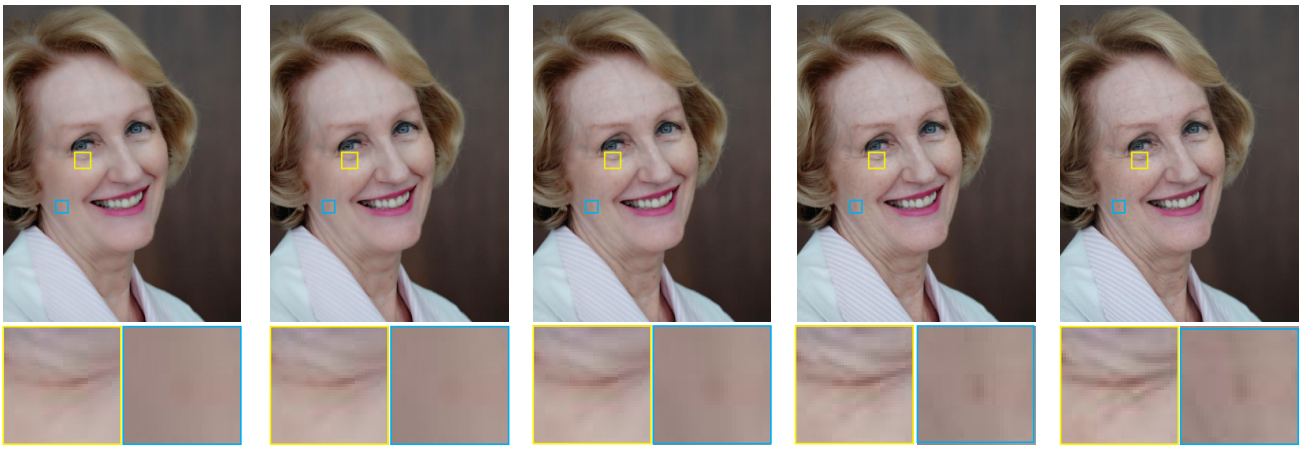
Scale Normalization Practically, the lower-ranked components have small variance, making gradients easily dominated by higher-ranked ones in the shared network. To solve this problem, each component is divided by its corresponding standard deviation, i.e., $I^{(i)}_{norm} = \frac{1}{\sqrt{\lambda_i}} I^{(i)}$ where $I^{(i)}$ is defined in Eq. (5) and λ_i is the i^{th} eigenvalue. With this operation, all components have similar scales. During testing, we scale the subnetwork output back to the required quantity by multiplying corresponding standard deviation before summing them up.

5. Experimental Settings

Implementation setting All CRN models have the same network architecture described in Section 4. Unless otherwise stated, our models use 7 subnetworks to regress the top 7 components, and take a shared subnetwork to regress others. Our models are implemented in Caffe [12]. During training, the initial learning rate is 0.01. It decreases following a polynomial policy with 40,000 iterations. Gradient clipping [20, 14] is also used to avoid gradient explosion, so that large learning rates are allowed.

Dataset We utilized the portrait dataset [24] for evaluation. This dataset contains 2,000 images with various age, color, clothing, hair style, etc. We manually select images that contain many details and are not likely to have been edited before. They form our final dataset. A few examples are shown in Figure 5(a). These images are referred to as the ground truth data (not edited) in our experiments. Then we use three most popular image editing systems, i.e., Photoshop Express (PS), Meitu (MT) and Instagram (INS), to process the images. PS and MT are stand-alone software, while INS is photo sharing application that contains built-in editing functions. These systems have many users. Figure 5(b) shows editing results of different systems. We trained our models to reverse edit for each of the three systems.

Evaluation metrics The Peak Signal-to-Noise ratio (PSNR) and structural similarity (SSIM) [26] index are used for quantifying the performance of our approach. However, these measures are not sensitive to the details in hu-



(a) Edited Image (b) 2 Component (c) 4 Component (d) 7 Component (e) Ground truth

Figure 6. Visualization of contribution of each branch. (a) is the edited image; (b)-(d) show results using different numbers of branches; (e) is the original image (best viewed in color).

man faces. We thus utilize the relative improvement defined in Eq. (1) to form the Accumulated Relative Improvement (ARI) measure. ARI is set as

$$ARI = \sum_i d_i, \tag{7}$$

where d_i is introduced in Eq. (1). ARI measures the relative improvement over all components. They contribute similarly to the final result disregarding their initial weights in eigenvalues.

6. Evaluation of Our Approach

The critical issue for the blind reversion task is the component domination effect. In the following, we explain the effectiveness of different steps in our approach.

6.1. Contribution of Component-specific Learning

Statistical evaluation As the main contribution, the component-specific learning mitigates the component domination effect effectively. Table 1 shows PSNR, SSIM and ARI for different numbers of branches. When the number of branches is 1, all components are mixed, which becomes the baseline approach of our model. As the number of branches increases, the performance betters consistently, which means the domination effect is gradually suppressed. Note that the improvement is large in the beginning (e.g., 76.8% improvement for 2 branches in terms of ARI), and slows down with more branches (e.g., 4.32% with 7 branches in terms of ARI). This indicates that component domination effect is more severe in high-ranking principal components than low-ranking ones. Thus it is reasonable to use respective branches for major components, and a shared branch to regress the rest.

Branch	1	2	3	4	5	6	7
PSNR	36.7	39.9	39.0	41.4	41.5	41.6	**41.9**
SSIM	0.96	0.97	0.97	0.98	0.98	0.98	**0.98**
ARI	12.5	22.1	32.4	50.1	53.7	56.1	**60.5**

Table 1. Evaluation of effectiveness of component-specific learning.

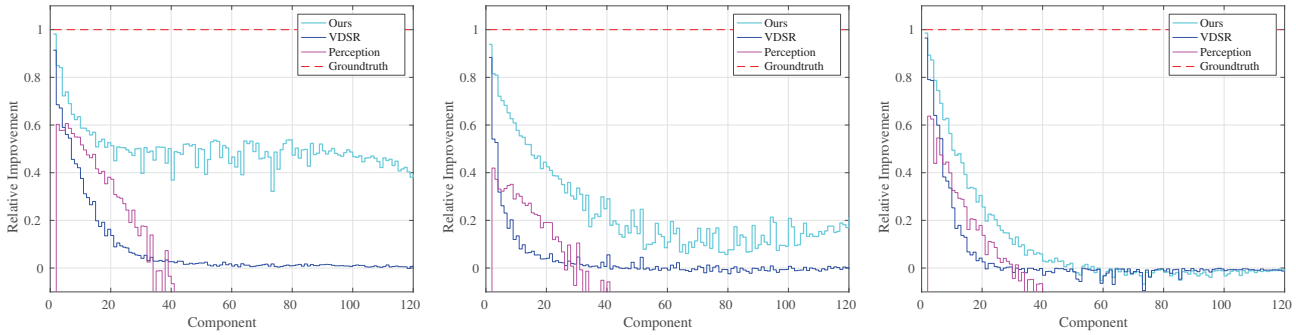
		PSNR	SSIM	ARI
MT	Full Model	**40.9**	**0.98**	**60.5**
	w/o DisMap	39.8	0.98	49.7
	w/o ScaleNorm	37.4	0.97	-6.0
PS	Full Model	**33.3**	**0.95**	**28.9**
	w/o DisMap	32.9	0.95	20.8
	w/o ScaleNorm	31.2	0.95	-15.2
INS	Full Model	**35.9**	**0.94**	**12.9**
	w/o DisMap	34.7	0.94	3.5
	w/o ScaleNorm	32.1	0.93	-25.7

Table 2. Evaluation of effectiveness of discrepancy map learning and scale normalization.

Visualization We further visualize the information that different branches contribute in Figure 6. As shown in Figure 6(a), face details are largely smoothed by the editing system. Then we use our 7-branch CRN model to process it. We block some branch output to understand what each branch learns. The result is shown in Figure 6(b-d). The smoothed wrinkle becomes more noticeable as the number of component increases. When all 7 branches are used, even a very subtle freckle can be recovered.

6.2. Contribution of Other Components

In addition to the component-specific learning, scale normalization and discrepancy map learning are also very im-



(a) Meitu (b) Photoshop (c) Instagram

Figure 7. Results of different methods in the three different datasets.

		PSNR	SSIM	ARI
MT	Ours	**40.9**	**0.98**	**60.5**
	VDSR [14]	39.5	0.96	12.8
	Perception [16]	28.1	0.96	-83.7
PS	Ours	**33.3**	**0.95**	**28.9**
	VDSR [14]	32.2	0.94	4.2
	Perception [16]	22.4	0.91	-89.2
INS	Ours	**35.9**	**0.94**	**12.9**
	VDSR [14]	33.4	0.93	4.8
	Perception [16]	18.7	0.89	-44.2

Table 3. Comparison with VDSR and perception loss on MT, PS and INS.

portant to our model. To evaluate them, in Table 2, we compare our full model with models that 1) directly regress the ground truth image (denoted as w/o DisMap); 2) do not perform scale normalization (denoted as w/o ScaleNorm). The quantities in tables indicate that the performance drops without discrepancy map learning. This is because directly learning the original image suffers more severely from the component domination effect. Note that if scale normalization is not used, our model fails because the small-scale components are not learned successfully.

6.3. Comparison with Other Methods

There is no exact work addressing blind reversion of unknown image edit. But in a broader context, our model is related to image-to-image FCN regression. VDSR [14] is an FCN model that achieves state-of-the-art results in super-resolution where high-frequency information needs to be restored. Our problem is different on the fact that image edit can change all levels of information regarding all coarser and subtle structures, while super-resolution generally keeps the coarse level of edges.

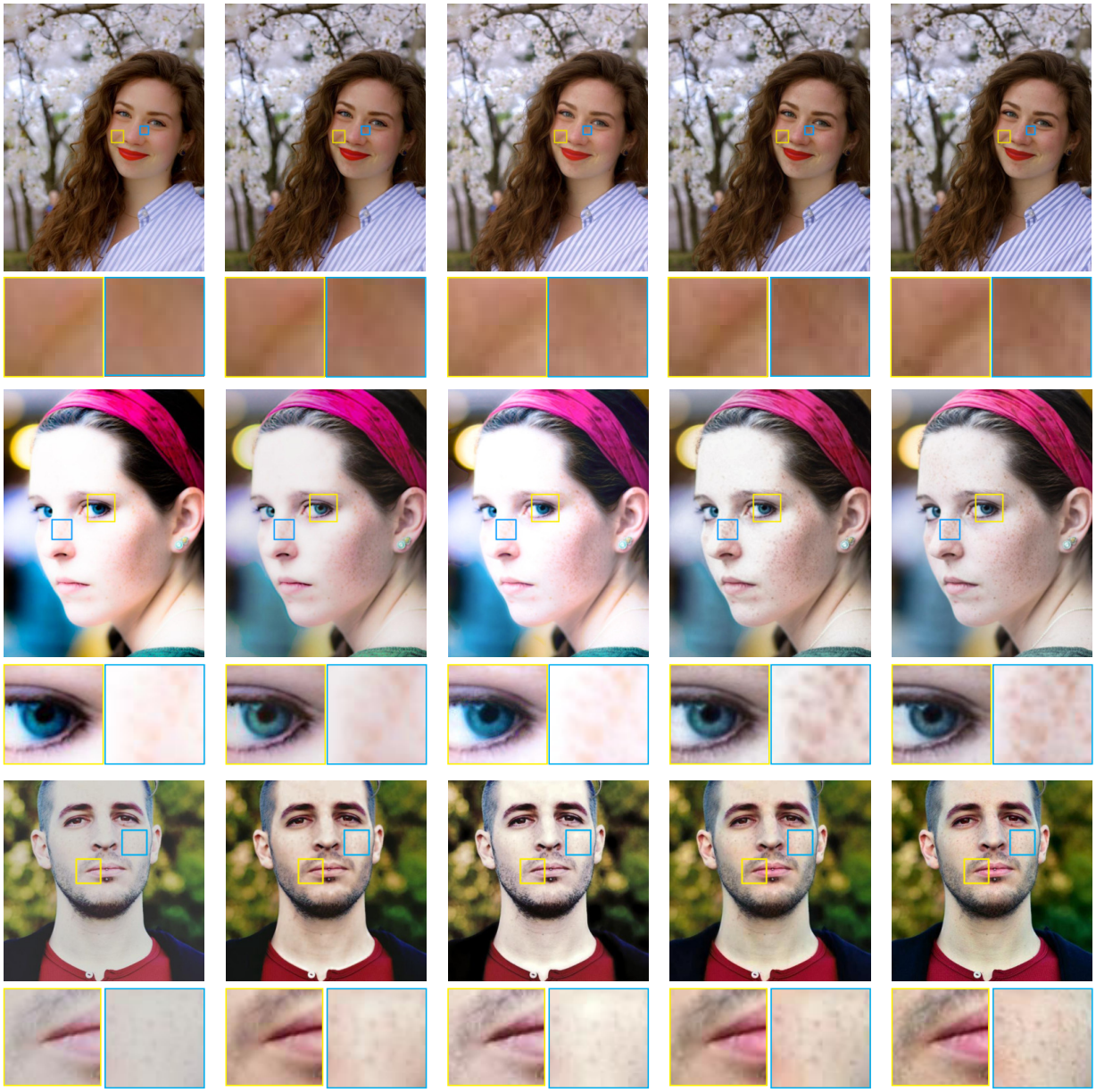
Perception loss [16] is another regression loss function. It is combination of VGG-based content loss and adversarial loss. Compared to Euclidean loss, perception loss is performed in the feature level. As indicated in [13, 16], it yields better performance on details than Euclidean loss, but achieves lower PSNR.

Real-world Examples Figure 8 compares VDSR [14], perception loss [16], and our model in real-world cases. It shows that VDSR can regress color and illumination well, and yet fail to learn details. On the contrary, by combining VGG loss and adversarial loss, perception loss learns details better. But it does not handle color similarly well. This is because perception loss utilizes layers of VGG network, which is trained for classification. It requires robust features invariant to color or small texture change. Our model handles both problems for satisfying detail regression.

Statistical Comparison To prove the generality, we further show PSNR, SSIM and ARI (Eq (7)) over all testing images edited by MT, PS and INS in Table 3. Statistics indicate that our model yields better performance than VDSR and perception loss. It means our model can regress portraits more accurately.

In addition, benefitted from component-specific learning, our model outperforms other methods in terms of ARI by a large margin. It is a bit surprising that perception loss does not perform that well regarding these statistical metrics, especially the ARI. It actually is because perception loss relies on the VGG network [13, 25], which is trained for classification and may discard information that is not discriminative. Thus these components would be completely ignored by the network. The adversarial part does not help find them since it does not impose similarity between the network output and the target. As will be shown later, perception loss works well on certain components, but not all of them.

Component Level Comparison In addition to the statistic metrics, it is also interesting to see relative improvement in the component level. Figure 7 shows the relative improvement curve for each method. As discussed in Section 3, because of the component domination problem, VDSR achieves large improvement only in the top-ranking compo-



(a) Edited Image (b) VDSR (c) Perception (d) Ours (e) Ground truth

Figure 8. Reverting different editing systems. The 1^{st}, 2^{nd} and 3^{rd} rows correspond to MT, PS and INS respectively.

nents. Our model tackles this problem by using component-specific learning. Also, it is clear that the perception loss helps accomplish better result than VDSR only on median-ranking components. This finding complies with the observation in [13, 16] that perception loss yields better visual quality. But PSNRs may be lower.

7. Concluding Remarks

In this paper, we have proposed a blind image reversion task for virtual portrait beautification. We have addressed the component domination effect, and proposed a multi-branch network to tackle this problem. The relative improvement is used to quantify the overall performance of each component. Extensive experiments verified the effectiveness of our method.

There are inevitably limitations. First, we do not handle geometric transformation yet. Second, if details are processed quite differently in image regions, the system may need many data to learn this pattern. Third, our method is regression-based. Thus it cannot handle edit that totally removes details in stylization or strong makeup. We will address these challenges in our future work.

.

References

[1] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Trans. Graph.*, 26(3):10, 2007.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[3] T. Carlo and M. Roberto. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[4] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 2015.

[5] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman. The patch transform and its applications to image editing. In *CVPR*, 2008.

[6] C. Dong, Y. Deng, C. Change Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, 2015.

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2016.

[8] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016.

[9] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, 30(4):69, 2011.

[10] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV*, 2010.

[11] S.-M. Hu, K. Xu, L.-Q. Ma, B. Liu, B.-Y. Jiang, and J. Wang. Inverse image editing: Recovering a semantic editing history from a before-and-after image pair. *ACM Trans. Graph.*, 2013.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[13] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[14] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[16] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[17] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep joint image filtering. In *ECCV*, 2016.

[18] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016.

[19] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV*, 2006.

[20] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML*, 2013.

[21] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.

[22] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In *NIPS*, 2015.

[23] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir. A comparative study of image retargeting. In *ACM transactions on graphics*, 2010.

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

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

[27] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, 2012.

[28] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*, 2010.

[29] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via l 0 gradient minimization. *ACM Trans. Graph.*, 30(6):174, 2011.

[30] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *NIPS*, 2014.

[31] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, 2015.

[32] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *CVPR*, 2013.

[33] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu. Automatic photo adjustment using deep neural networks. *ACM Trans. Graph.*, 35(2):11:1–11:15, 2016.

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

[35] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter. In *CVPR*, 2014.

[36] S. Zheng, L. Xu, and J. Jia. Forward motion deblurring. In *ECCV*, 2013.