Localin Reshuffle Net: Toward Naturally and Efficiently Facial Image Blending

Chengyao Zheng[1], Siyu Xia[1][0000−0002−0953−6501] ⋆, Joseph Robinson[2], Changsheng Lu[3], Wayne Wu[4], Chen Qian[5], and Ming Shao[6]

[1] Southeast University, Nanjing, China
xsy@seu.edu.cn
[2] Northeastern University, Boston, MA, USA
[3] Shanghai Jiao Tong University, Shanghai, China
[4] Tsinghua University, Beijing, China
[5] SenseTime, China
[6] University of Massachusetts Dartmouth, Dartmouth, MA, USA

Abstract. The blending of facial images is an effective way to fuse attributes such that the synthesis is robust to the finer details (*e.g.*, periocular-region, nostrils, hairlines). Specifically, facial blending aims to transfer the style of a source image to a target such that violations in the natural appearance are minimized. Despite the many practical applications, facial image blending remains mostly unexplored with the reasons being two-fold: 1) the lack of quality paired data for supervision and 2) facial synthesizers (*i.e.*, the models) are sensitive to small variations in lighting, texture, resolution and age. We address the reasons for the bottleneck by first building Facial Pairs to Blend (FPB) dataset, which was generated through our facial attribute optimization algorithm. Then, we propose an effective normalization scheme to capture local statistical information during blending: namely, Local Instance Normalization (LAN). Lastly, a novel local-reshuffle-layer is designed to map local patches in the feature space, which can be learned in an end-to-end fashion with dedicated loss. This new layer is essential for the proposed Localin Reshuffle Network (LRNet). Extensive experiments, and both quantitative and qualitative results, demonstrate that our approach outperforms existing methods.

1 Introduction

Facial blending is an effective way to synthesize new faces, which has highly practical value in many applications (*e.g.*, face swapping [1, 2], rendering facial makeup [3–6], and portrait style transfer [2, 7]). Unlike transfer methods tend to transfer global-level statistics from a source to a target domain (*e.g.*, style transfer [8] and color transfer [9]), face blending is a more complex task, as it aims to mix a source face onto a target in specific local areas (Fig. 1). However, a lack of paired face data prohibits the use modern-day data-driven models (*i.e.*,

⋆ Corresponding author.

Fig. 1: Illustration for different tasks.

supervised deep learning), which is one of the challenges posed in the face blending task. Another challenge is the sensitivity to even slight variations in intrinsic image properties (*e.g.*, textures and illuminations). Furthermore, boundary conditions also contribute to the challenge. Specifically, the goal of face blending is to blend a cropped region from source face $\mathbf{x_A}$ onto the target $\mathbf{x_B}$. Inherently, there are typically significant conflicts about the boundaries and in textures differences (*e.g.*, skin color, skin texture, and image resolution).

Formally put, a two-stage solutions are most commonly proposed for facial blending: 1) apply a style or color transfer algorithm to $\mathbf{x_A}$, and 2) use Poisson image blending [10] for the final result. More specifically, (1) is an attempt to disentangle content and style (or color). When employed to arbitrary facial images, there exists no style (or color) transfer algorithm that simultaneously handles skin-tone, skin-texture, and lighting conditions robustly. As for (2), it is known for its efficiency and natural results [10]. However, Poisson image blending performs poorly on faces with variations in textures and lighting conditions since texture consistency between background and foreground is ignored.

Recently, one-stage solutions have been proposed. FSGAN [11] combined Poisson optimization with perceptual loss to blend two faces while preserving target skin color, but they did not take some key factors like local texture into consideration. Li *et al.* proposed a blending method that attempts to modify expressions and gestures, which undesirably alters the identity [12].

To overcome shortcomings of our predecessors, we propose a single stage solution called Localin Reshuffle Network (LRNet). Given a pair of face images $\mathbf{x_A}$ and $\mathbf{x_B}$, LRNet blends $\mathbf{x_A}$ onto $\mathbf{x_B}$, while transferring the local information (*e.g.*, texture and illumination) from $\mathbf{x_B}$ to the final output $\tilde{\mathbf{x}}$. In support of this work, we introduce the first paired facial image dataset for image blending named Facial Pairs to Blend (FPB). Since the blending of facial images requires pairs of faces of different identities to have similar orientation and shape. Hence, collecting such a large number of real-world samples is a time-consuming challenge. Based on Style-GAN2 [13], we design a facial attribute optimization algorithm, to generate images per the requirements.

The contributions of this paper are as follows:

- We introduce the first image dataset for face blending with 12,000 pairs.
- We propose the Local Instance Normalization and Local Reshuffle Layer in our LRNet: the former transfers local statistics and the latter reconstructs texture from a reference to make invariant to intrinsic image properties.

- We clearly demonstrate the improvement over existing methods on high-quality face images using our LRNet to synthesize.

2 Related Works

2.1 Image blending

As a common image composition task, to blend images is to blend cropped regions of a source image onto a target image at specific locations. For this, one can simply replace the pixels of the target with those cropped from the source. However, it would yield artifacts caused by differences in intensity between foreground and background. Although alpha blending alleviates this issue by blurring the boundary [14], it still suffers from inconsistent coloring and lighting.

Gradient-based methods have been proven an ability to produce a smoother transition, which reduces problems from differences in color and illumination [10, 15–17] - the most popular of these is Poisson image blending [10]. The motivation of gradient-base methods is that humans are sensitive to an abrupt change in intensity change and, thus, these methods attempt to ensure the smoothness of blended images in the gradient domain.

Recently, Gaussian-Poisson Generative Adversarial Network (GP-GAN) [18] explores the capability of a GAN in image blending task, while Zhang *et al.* proposed a Poisson blending loss computed from a neural network [19]. However, these methods do not account for local aspects like texture and illumination. Hence, the proposed Localin Reshuffle Network (LRNet) ensures smoothness of blended images via gradient-domain and transfers the local texture and illumination. In the end, our results are most photo-realistic.

2.2 Neural style transfer

In facial image blending, style transfer algorithms tend to reduce the visual difference between the two images. Gatys *et al.* found that feature statistics calculated from the Gram matrix successfully capture the image style [20]. The authors were then inspired to transfer arbitrary styles by matching the feature statistics [21–23]. However, the early methods are time-consuming, since they are based on an iterative optimization framework.

Feed-forward networks to approximate an optimal result were proposed as a means to reduce time costs [24–26]. However, the algorithms are still unable to transfer unseen styles from an arbitrary image. Later developments alleviated the limitation posed from when unseen styles are mishandled [3, 8, 27]. WCT [3] encodes the style as the feature co-variance matrix to support rich style representation. AdaIN [8] captures arbitrary image style by computing the mean and variance on the feature maps. Adaptive instance normalization is an efficient style transfer method, which is widely used since introduced [8]. MUNIT was proposed with AdaIN adopted as a high-level style expression to do image-to-image translation task [28]. Style-GAN applied AdaIN in generator to control styles at multiple levels [29].

Although better performance were achieved in neural style transfer, developing an algorithm that is all around efficient (*i.e.*, handles textures, semantic-correspondences, and in a realistic manner) is still a challenge to overcome. Nonetheless, the proposed framework is all around efficient.

2.3 Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN) have drawn great attention in research [30]. Originally, the GAN was used to generate low-resolution handwritten digit data. Since its dbut, a lot of literature on improving vanilla GAN, *e.g.*, Laplacian pyramids of adversarial networks [31], deeper neural networks [32], constraints on generator (G) (*e.g.*, cycle consistency), and methods to stabilize training [33].

Other using cases for GAN involve controlling its output. Infogenerative adversarial network (GAN) focuses on learning interpretable latent representations [34]. Conditional GAN (C-GAN) adds a conditional attribute to the latent code and discriminator (D) [35]. Liu *et al.* improves on C-GAN by introducing a flow-based G [36].

Most recently, Style-GAN2 strengthened style controls, along with improved quality in output image [13]. Abda* *et al.* explored a method to embed images into the latent code of the Style-GAN [37]. Specifically, a latent code is randomly initialized for an input, from which an image is generated with a pre-trained Style-GAN. The dissimilarity between the given image (*e.g.*, *real*) and the image generated (*e.g.*, *fake*) is minimized by iterative optimization. In fact, Abda *et al.* inspired our FPB dataset to have minimal dissimilarities in face orientation and shape, and all the while maximizing other face attributes.

3 Facial Pairs to Blend (FPB) Dataset

FPB is proposed as the first image set to support facial blending. For this, our LRNet generated ground-truth (*i.e.*, supervision) for 12,000 pairs (*i.e.*, $\mathbf{x_A}$ blended onto $\mathbf{x_B}$, and vice-versa). This allowed for face pairs of different appearance to have the same orientation (*i.e.*, pitch, yaw, and roll) and facial contour. We next explain our data, and then in the proceeding section introduce LRNet.

3.1 Dataset generation algorithm

As face blending task requires two faces with similar orientation and shape but different appearance, the collection of such data in the real world could be rather expensive. Inspired by the work of Style-GAN encoder [37], we use Style-GAN2 [13] to generate facial images, and define the loss function as constraints on face attributes (*i.e.*, orientation, facial contour, and appearance). Through minimizing the loss function, paired-wise facial images that meet the requirements can be accessed.

Our paired-wise data generation algorithm is presented in Fig. 2. To illustrate the problem clearly, our method is divided into three steps in the figure.

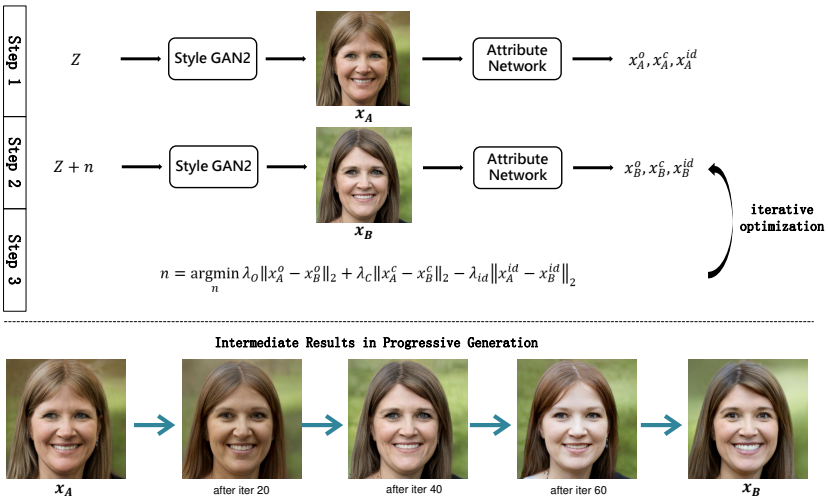


Fig. 2: Our paired-wise facial images generation algorithm. A random latent code Z generates $\mathbf{x_A}$. Then, we add noise to Z to synthesize a completely new image $\tilde{x}_B$. The loss function aims to minimize the difference in the orientation and contour of faces and maximize the identity loss. The bottom row shows the transforming process from $\mathbf{x_A}$ to $\mathbf{x_B}$.

(1) We randomly generate a latent code Z in Style-GAN2 [13] latent space, Z has a shape of 512×1. Image A is simply achieved by feeding Z to Style-GAN2. A pre-trained facial attribute network [38] gives an estimated result of image A, we denote the facial orientation as $\mathbf{x_A^o}$ (with a size of 3×1), facial contour as $\mathbf{x_A^c}$ (16×2), facial identity vector as $\mathbf{x_A^{id}}$ (4,000×1).

(2) We add noise n to Z, and treat $Z + n$ in the same way as Z, obtaining $\mathbf{x_B^o}$, $\mathbf{x_B^c}$, and $\mathbf{x_B^{id}}$.

(3) We define a loss to measure the similarity between face images $\mathbf{x_A}$ and $\mathbf{x_B}$. The idea is to ensure two images have similar facial orientation and contour, while they look different in appearance as much as possible. The loss is formulated as Eq. (1). λ_o, λ_c and λ_{id} are the weights to balance different components. To minimize $loss_{AB}$, we take the derivative of $loss_{AB}$ with respect to n, and update the noise, in a similar way of Eq. (2).

$$loss_{AB} = \lambda_o \left\|\mathbf{x_A^o} - \mathbf{x_B^o}\right\|_2 + \lambda_c \left\|\mathbf{x_A^c} - \mathbf{x_B^c}\right\|_2 - \lambda_{id} \left\|\mathbf{x_A^{id}} - \mathbf{x_B^{id}}\right\|_2 \tag{1}$$

$$n = n - \frac{\partial loss_{AB}}{\partial n} \tag{2}$$

We noticed that the loss in Eq. (1) usually converged to a minimum after 80 iterations of *Step 2 - 3* - continued training did not improve results. Then, we adopted the final result as $\mathbf{x_B}$. We also set an early stop strategy to accelerate the generation process.

Even though Style-GAN2 is state-of-the-art (SOTA) in generating photo-realistic images, there still exists a domain gap between the distribution of the

Fig. 3: FPB dataset. Columns show face pairs, with the last two being real images. The similarity in orientation and shape but difference in appearances.

real and *fake* data. For this, we collect images from the web and manually select 2,000 pairs that have similar orientations and face-shapes. We then use TPS [39] to warp one image per pair so that edges line-up. In total, Facial Pairs to Blend (FPB) consists of 12,000 face pairs (Fig. 3).

3.2 Dataset implementation details

Although StyleGAN2 performs well most of the time, artifacts still can be observed in some extreme illumination and pose conditions. After 15,467 pairs of images are generated, we manually select 10,000 pairs of good-quality.

The facial orientation ($\mathbf{x_A^o}$, $\mathbf{x_B^o}$) is a 3×1 vector (*i.e.*, pitch, yaw, and roll angles, respectively). The facial contour($\mathbf{x_A^c}$, $\mathbf{x_B^c}$) is a 16×2 vector made-up of the sixteen facial landmarks along the outside of the face. We normalize these landmarks to $[0, 1]$. The size of facial identity vector($\mathbf{x_A^{id}}$, $\mathbf{x_B^{id}}$) is 4,000×1. To generate the vector, we add a fully connected network(3 layers) after the feature extraction layers of [38], and train the fully connected module for face recognition tasks on CelebA [40]. We apply the output of the third layer in fully connected network as facial identity vector. λ_o, λ_c, λ_{id}, are 0.1, 100, and 0.001 in our experiments.

The generation process would stop if reaching the maximum number of iterations, which is 80 in our paper, or meeting the following three stop criteria:

$$\left\|\mathbf{x_A^o} - \mathbf{x_B^o}\right\|_2 < 10 \;\&\; \left\|\mathbf{x_A^c} - \mathbf{x_B^c}\right\|_2 < 0.01 \;\&\; \left\|\mathbf{x_A^{id}} - \mathbf{x_B^{id}}\right\|_2 > 1000$$

Empirically, if above three conditions are satisfied, the orientation and face shape of image B should be similar to A, while looks like two totally different persons.

4 Localin Reshuffle Network (LRNet)

LRNet, consisting of the our local instance normalization (LocalIN) and *local reshuffle* layer. Specifically, LocalIN is a layer that transfers local statistics and the *local reshuffle* layer reconstructs the new feature maps for $\mathbf{x_A}$ with patches from $\mathbf{x_B}$. Both of the novel layers are essential for a precise transfer of local texture and lighting while blending faces. We discuss all of the parts, the network as a whole, and then the loss functions in the remainder of this section.

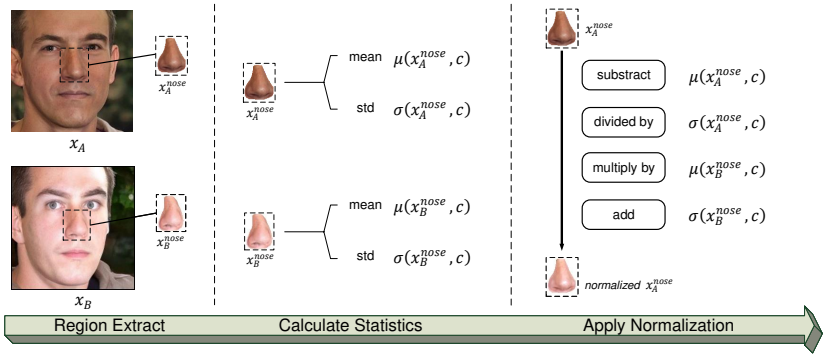


Fig. 4: Example of implementing LocalIN in RGB space about the nose region.

4.1 LocalIN

As mentioned, adaptive instance normalization (AdaIN) has been proved to be an efficient style transfer operation [8]. However, since AdaIN uses global statistics of an image it is insensitive to the local style, which is imperative for objects with finer details (*i.e.*, faces). To achieve style transfer at a semantic level (*e.g.*, nose-to-nose or mouth-to-mouth), we propose LocalIN as a normalization method. It operates in RGB color space (Fig. 4).

More specifically, given two images $\mathbf{x_A}$ and $\mathbf{x_B}$, $\mathbf{x_A}$ will be normalized by $\mathbf{x_B}$. The first step is to divide the image into different semantic regions, normalization will only be implemented in the same region of two images. Take nose as an example, we calculate the mean and standard deviation inside $\mathbf{x_A^{nose}}$ on each channels, respectively. Let $\mu(\mathbf{x_A^{nose}}, c)$ and $\sigma(\mathbf{x_A^{nose}}, c)$ be the mean and standard deviation calculated in $\mathbf{x_A^{nose}}$ on channel c. $\mu(\mathbf{x_B^{nose}}, c)$ and $\sigma(\mathbf{x_B^{nose}}, c)$ are defined in a similar way. Finally, for every pixel inside $\mathbf{x_A^{nose}}$ on each channel, its value $F(\mathbf{x_A^{nose}}, i, c)$ will be calculated as

$$F_n(\mathbf{x_A^{nose}}, i, c) = \frac{F(\mathbf{x_A^{nose}}, i, c) - \mu(\mathbf{x_A^{nose}}, c)}{\sigma(\mathbf{x_A^{nose}}, c)} * \sigma(\mathbf{x_B^{nose}}, c) + \mu(\mathbf{x_B^{nose}}, c). \quad (3)$$

The framework of LocalIN is given in Algorithm 1. The operation is differentiable, and can be executed efficiently. This allows us to easily add the module to a neural network and optimizing with our objective functions. LocalIN transfers high-level style information from $\mathbf{x_B} \rightarrow \mathbf{x_A}$, which includes knowledge of lighting, skin color, and image tone. However, for photo-realistic results we need to handle texture, which is done via the *local reshuffle* layer described next.

4.2 Local reshuffle

We propose a novel *local reshuffle* for inconsistencies in texture. Specifically, we reconstruct the feature maps of $\mathbf{x_A}$ using patches from the feature maps of $\mathbf{x_B}$. The resulting maps then share the same local texture as $\mathbf{x_A}$, while retaining the

structure of $\mathbf{x_B}$. We first demonstrate the proposed *local reshuffle* in RGB space. Then, a more general formula is provided(Algorithm 2).

Given images $\mathbf{x_A}$ and $\mathbf{x_B}$, our goal is to reconstruct a new image $\mathbf{x_A^{reshuffled}}$ made-up of patches from $\mathbf{x_B}$. A patch is defined as a square block with the number of channels equivalent to the feature maps. The patch size is set as 3×3×3 (height*weight*channel) in RGB space for the reported results. As shown in Fig. 5, patch($\mathbf{x_A^{nose}}, i$) represents the i-th patch of $\mathbf{x_A^{nose}}$, while patch($\mathbf{x_B^{nose}}, j$) represents the j-th patch in $\mathbf{x_B^{nose}}$. Then, each patch($\mathbf{x_A^{nose}}, i$) is matched up with the most similar patch($\mathbf{x_B^{nose}}, j$) in $\mathbf{x_B^{nose}}$, which is denoted as $\phi_{\mathbf{x_A}\to\mathbf{x_B}}(i) = j$. We compute $\phi_{\mathbf{x_A}\to\mathbf{x_B}}(i)$ by maximizing Eq. (4). After each patch($\mathbf{x_A^{nose}}, i$) has been matched with a patch($\mathbf{x_B^{nose}}, j$), the new feature maps $F_{re}(\mathbf{x_A^{nose}})$ are reconstructed. By concatenating and feeding all patches in $\mathbf{x_B^{nose}}$ into a convolution kernel, the patch-matches of $\mathbf{x_A^{nose}}$ and $\mathbf{x_B^{nose}}$ can be replaced by a single forward convolution computation. Since an additional *same-region* restriction is included, this was dubbed *local reshuffle*.

$$\phi_{A\to B}(i) = \arg\max_j \left\| patch(\mathbf{x_A^{nose}}, i) * patch(\mathbf{x_B^{nose}}, j) \right\|_2 \tag{4}$$

Algorithm 2 describes the proposed *local reshuffle*. The input feature maps $\mathbf{x_A}$ and $\mathbf{x_B}$ with corresponding channels C are denoted as $F(\mathbf{x_A^k})$ and $F(\mathbf{x_B^k})$. Reshuffling produces the new feature maps $F_n(\mathbf{x_A^k})$. Since patches are 3×3× C, only the center 1×1× C value is used to reconstruct $F_n(\mathbf{x_A^k})$ via patches in $\mathbf{x_B^k}$.

Algorithm 1 Framework of Local Instance Normalization

Input:
The feature maps of $\mathbf{x_A}$ and $\mathbf{x_B}$ in region k (*i.e.*, $F(\mathbf{x_A^k})$ and $F(\mathbf{x_B^k})$, respectively)
Output:
Normalized result: $F_n(\mathbf{x_A^k})$
for each channel c **do**
$\mu(\mathbf{x_A^k}, c) = \sum_i^{i\in\mathbf{x_A^k}} \frac{F(\mathbf{x_A^k}, i, c)}{N}$
$\sigma(\mathbf{x_A^k}, c) = \sqrt{\sum_i^{i\in\mathbf{x_A^k}} \frac{\left[F(\mathbf{x_A^k}, i, c) - \mu(\mathbf{x_A^k}, c)\right]^2}{N}}$
$\mu(\mathbf{x_B^k}, c) = \sum_j^{j\in\mathbf{x_B^k}} \frac{F(\mathbf{x_B^k}, j, c)}{M}$
$\sigma(\mathbf{x_B^k}, c) = \sqrt{\sum_j^{j\in\mathbf{x_B^k}} \frac{\left[F(\mathbf{x_B^k}, j, c) - \mu(\mathbf{x_B^k}, c)\right]^2}{M}}$
for each $i \in \mathbf{x_A^k}$ **do**
$F_n(\mathbf{x_A^k}, i, c) = \frac{F(\mathbf{x_A^k}, i, c) - \mu(\mathbf{x_A^k}, c)}{\sigma(\mathbf{x_A^k}, c)} * \sigma(\mathbf{x_B^k}, c) + \mu(\mathbf{x_B^k}, c)$
end for
end for
return $F_n(\mathbf{x_A^k})$;

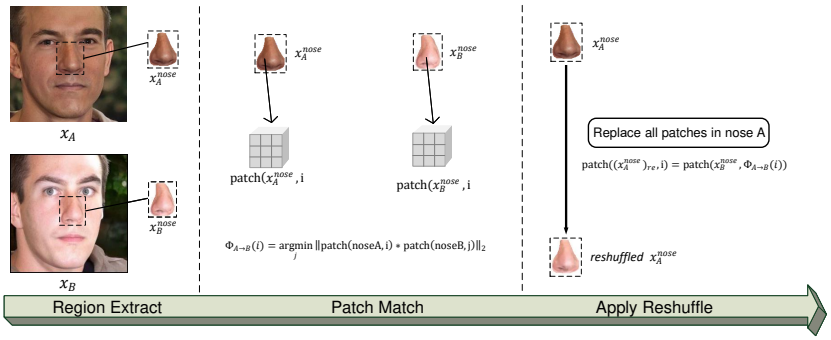


Fig. 5: Example of implementing local reshuffle on RGB space inside nose region.

4.3 Semantic correspondence constraint

When blending $\mathbf{x_A}$ into $\mathbf{x_B}$, we expect blended image $\tilde{\mathbf{x}}$ shows similar style(eg: skin color, texture and illumination) inside same semantic region(eg: left eye) with $\mathbf{x_B}$. In this work, we also proposed a novel way to ensure semantic correspondence in *local reshuffle*.

Firstly, we get the the 3D facial vertices of $\mathbf{x_A}$ and $\mathbf{x_B}$ via PRNet [38], denoted as P_A and P_B. Then both P_A and P_B are mapped to a sphere whose radius is 1. This process can be seen as fitting the 3D facial mesh onto a sphere. After mapping, the product between two vertices can be used to measure the relative distance. The greater the product, the smaller the distance. The proof process is in the supplementary material. After the conversion, we have an elegant and simple way to measure distance.

The product between 3D vertices can be used as in convolution operations. This allows for the merging of semantic correspondence constraint in the *local reshuffle*. In our experiments, we concatenate 3D vertices with feature maps, which extend the channel C of the feature maps to $C + 3$. Patches within the same semantic region tend to have a greater product that leads to more energy (Eq. (4)). By mapping 3D vertices into feature maps, we achieve the semantic correspondence of *local reshuffle*.

4.4 Network architecture

LocalIN and local reshuffle are critical component in the network of this work, with the former for high-level style transfer and the latter for local texture transfer. We named the overall architecture LRNet (Fig. 6). As shown, the goal is to blend face $\mathbf{x_A}$ onto $\mathbf{x_B}$. The result yields similar texture and lighting distribution as $\mathbf{x_B}$, while retaining the identity of $\mathbf{x_A}$.

Firstly, LRNet extracts features of $\mathbf{x_A}$ and $\mathbf{x_B}$ with the same encoder– the same encoder used as the feature maps are assumed to share data distribution, which will be used for the dot-product, comparison, and reshuffle. Feature maps $F(\mathbf{x_A})$ and $F(\mathbf{x_B})$ are divided into K semantic regions, and each pair $F(\mathbf{x_A^k})$ and $F(\mathbf{x_B^k})$ will be sent to two branches: Localin and local reshuffle. For each

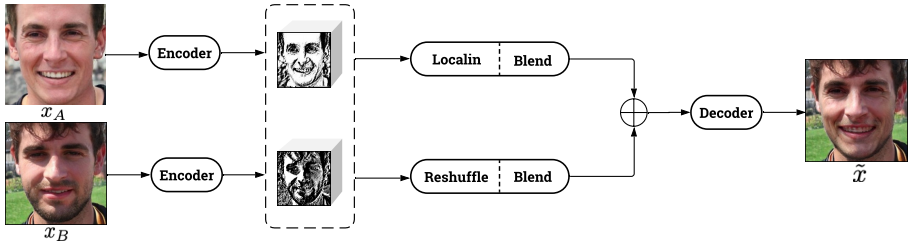


Fig. 6: Overview of Localin-Reshuffle-Net. Concatenation is represented by $\oplus$.

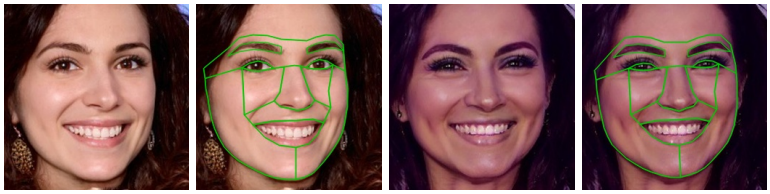
branch, after obtaining $F_{new}(\mathbf{x_A^k})$, we will blend them into $F(\mathbf{x_B})$ through alpha blending [14] to avoid the mutation around the boundary. Next, two blended feature maps are concatenated, and fed to a decoder. Finally, the decoder generates the blended result. Thanks to Localin and local reshuffle, LRNet not only blends $\mathbf{x_A}$ onto $\mathbf{x_B}$, but also transfers the texture and lighting at the same time.

4.5 Loss functions

Reconstruction loss The reconstruction loss of our model is motivated by the following: reconstruction in the *background* should be the same as $\mathbf{x_B}$; the *foreground* should resemble $\mathbf{x_A}$ as much as possible. To that end, we develop a reconstruction loss to penalize induced errors in training. Let $\tilde{\mathbf{x}}$ be the output RGB image. Let $\tilde{\mathbf{x}}(fg)$ and $\tilde{\mathbf{x}}(bg)$ refer to the foreground and background of $\tilde{x}$, respectively. $VGG_{1,2}(\tilde{\mathbf{x}}(fg))$ refers to feeding $\tilde{\mathbf{x}}(fg)$ through a pre-trained VGG19 model [41], and get the feature maps after the $relu1\_1$ and $relu2\_1$ layers. This operation is also known as perceptual loss [26]. Our reconstruction loss can be formulated as Eq. (5). The parameter α is a weighting factor that balances the two, it's 0.2 in this experiment. N and M are the total pixels in the background and foreground, respectively. We chose to use the perceptual loss as penalty from the pixels of the foreground is not as strict as those in the background. This is because difference in detail (e.g., facial texture, illumination), is not only allowed but encouraged.

Algorithm 2 Framework of Local Reshuffle

Input:
The feature maps of $\mathbf{x_A}$ and $\mathbf{x_B}$ in region k (*i.e.*, $F(\mathbf{x_A^k})$ and $F(\mathbf{x_B^k})$, respectively)
Output:
Reshuffled result: $F_{re}(\mathbf{x_A^k})$
for each $i \in \mathbf{x_A^k}$ **do**
$\phi_{\mathbf{x_A}\to\mathbf{x_B}}(i) = \arg\min_j \|patch(\mathbf{x_A^{nose}}, i) * patch(\mathbf{x_B^{nose}}, j)\|_2$
end for
for each $i \in \mathbf{x_A^k}$ **do**
$\text{patch}((\mathbf{x_A^{nose}})_{re}, i) = patch(\mathbf{x_B^{nose}}, \phi_{\mathbf{x_A}\to\mathbf{x_B}}(i))$
end for
return $F_{re}(\mathbf{x_A^k})$;



(a) (b) (c) (d)

Fig. 7: A pair of segmentation results in our experiments. There are 10 regions in total. Shown here is $\mathbf{x_A}$ (a), $\mathbf{x_A^{segmented}}$ (b), $\mathbf{x_B}$ (c), and $\mathbf{x_B^{segmented}}$ (d).

$$L_{rec} = \frac{1}{N} \left\| \mathbf{x_B}(bg) - \tilde{\mathbf{x}}(bg) \right\|_2 + \frac{\alpha}{M} \left\| VGG_{1,2}(\mathbf{x_A}(fg)) - VGG_{1,2}(\tilde{\mathbf{x}}(fg)) \right\|_2 \quad (5)$$

Cycle Consistency Loss If we blend $\tilde{\mathbf{x}}$ onto $\mathbf{x_A}$, the most accurate output should look like $\mathbf{x_A}$ itself. $LRNet(\mathbf{x_A} \rightarrow \mathbf{x_B})$ means blending $\mathbf{x_A}$ onto $\mathbf{x_B}$. Our cycle consistency loss is defined as

$$L_{cycle} = \frac{1}{N+M} \left\| \mathbf{x_A} - LRNet(LRNet(\mathbf{x_A} \rightarrow \mathbf{x_B}) \rightarrow \mathbf{x_A}) \right\|_1 . \quad (6)$$

Gradient Consistency Loss We have discussed in Section 2.1 that gradient-based methods plays a key role in producing a seamless blending boundary. We extend this idea to our gradient consistency loss. Let $\nabla A(fg)$ be the gradient of $\mathbf{x_A}$ in background area. The gradient consistency loss can be defined as

$$L_{grad} = \frac{1}{N+M} \left\| \nabla A(fg) + \nabla B(bg) - \nabla out \right\|_1 . \quad (7)$$

Local style loss In order to ensure the output share a similar texture and lighting with $\mathbf{x_B}$ in local areas, we define a new texture loss based on previous style transfer works [8]. Same as the perceptual loss in Section Reconstruction Loss, we leverage a pre-trained VGG19 model to extract the feature maps for $\mathbf{x_B}$ and $\tilde{\mathbf{x}}$. Denote $\mu(\mathbf{x_B^k}, c)$ as the mean value of $\mathbf{x_B}$'s feature maps on the c channel, our local style loss is formalized in Eq. (8). Note that feature maps are calculated at the $relu1\_1$ layer, making the loss more sensitive to the low-level style information. The weighting factor β is 2.0 in this work.

$$L_{style} = \frac{1}{KC} \sum_{k}^{K} \sum_{c}^{C} \left\| \mu(\mathbf{x_B^k}, c) - \mu(\tilde{\mathbf{x}}^{\mathbf{k}}, c) \right\|_2 + \beta \left\| \sigma(\mathbf{x_B^k}, c) - \sigma(\tilde{\mathbf{x}}^{\mathbf{k}}, c) \right\|_2 \quad (8)$$

Total loss Integrating aforementioned losses, the total loss function is

$$L_{total} = \lambda_1 L_{rec} + \lambda_2 L_{cycle} + \lambda_3 L_{grad} + \lambda_3 L_{style}, \quad (9)$$

where weighting factors λ_1, λ_2, λ_3, and λ_4 are 3.0, 20, 100, and 0.35, respectively, which are obtained via cross validation experiments.

5 Experiments

5.1 Implementation details

Network architecture The encoder in LRNet accepts a $H \times W \times 3$ ($H > 32$, $W > 32$) RGB image as input, and outputs feature maps with a size of $\frac{H}{2} \times \frac{W}{2} \times 128$. We adopt layers from $conv1-1$ to $relu2-1$ of VGG19 [41] as the backbone of the encoder, because a too deep network can leads to low efficiency and image details' loss. The decoder in LRNet accepts a $(H/2) \times (W/2) \times 256$ feature maps, and generates a $H \times W \times 3$ RGB image as output. In our experiments, the H and W are set as 256. The decoder uses deconvolution layers to enlarge feature maps and generate RGB images.

Training details Facial images are segmented into 10 regions (Fig. 7). Each connected domain is an independent region in the figure. LRNet is trained end-to-end with Adam [42] optimizing with a learning rate of 0.0001 and batch size of 1. 70% pairs in the dataset are used for training, and the rests are for validation. We use stratified sampling to select from both generated and real images.

5.2 Ablation Study

We conduct an ablation study to seeing how each branch in LRNet affects the performance. Let $F_{localin}$ be the feature maps generated by Localin branch, while $F_{reshuffle}$ is the feature maps of the Reshuffle branch. We designed two network structures: (a) remove the reshuffle branch, make a copy of $F_{localin}$, and concatenate $F_{localin}$ with its copy as the input to the following decoder and (b) remove the Localin branch, and process $F_{reshuffle}$ similar to (a).

The convergence processes of 4 losses are shown in Fig.8. For reconstruction loss, structure(a) converge fastest. But in a face blending task, reducing reconstruction loss to 0 is not our real goal. It's just a constraint to the general structure of the generated image. For style loss, the result shows that both (a) and (b) have the ability to transfer style. By putting them together, LRNet improves even more. Based on our observations, (a) tends to transfer image statistics, like color and illumination, while (b) is more sensitive to local texture. We also found that the consistency of local texture is critical for face blending, making the reshuffle branch indispensable. For grad loss and cycle loss, (b) performs worse than the others. The reason is that the Localin branch retains many of the details, dropping it makes it harder to recover the original face. Except for convergence processes, we also provide visual results in our supplementary materials.

5.3 Comparison with baselines

Here, we adopted several methods that are widely adopted in industry and research: *alpha blending*, apply color transfer [9] on $\mathbf{x_A}$ according to $\mathbf{x_B}$, and blend $\mathbf{x_A}$ onto $\mathbf{x_B}$ using alpha blending [14]; *Possion blending*, blend $\mathbf{x_A}$ onto

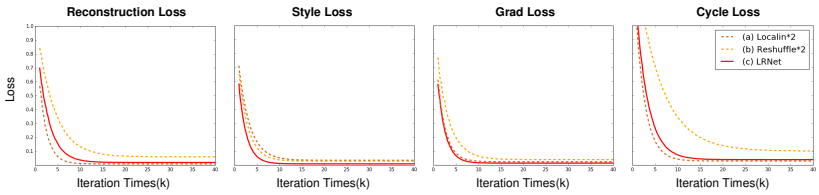


Fig. 8: The ablation study. We graph the convergence processes of loss functions as a function of iteration step k. Note, each loss was scaled by its weight factor.

$\mathbf{x_B}$ using Poisson blending [10]; *style blending*, apply style transfer [8] on $\mathbf{x_A}$ according to $\mathbf{x_B}$, and blend $\mathbf{x_A}$ onto $\mathbf{x_B}$ using alpha blending [14]; *deep blending*, a gradient-based blending method using numerical optimization [19] (< 50 seconds to completion).

Qualitative Comparison We show qualitative comparison results alongside the baselines in Fig.9. Before blending, we warp [39] the target image to ensure the target face aligns with the face edges of the source. Results show that *Alpha blending* as the simplest way to blend faces, which has the worst performance. It shows significant mutations around the blending boundary. What's worse, it has nothing to do with the inconsistency of texture and illumination at all. *Possion blending*, although successfully produce a smooth transition and reduce the color illumination differences, the problem of texture inconsistency still exists. *Style blending* tends to generate highly stylized images, which doesn't look like a real face. *Deep blending* takes much longer computation time, and sometimes produce artifacts leading to unrealistic faces.

The aforementioned methods inaccurately transfer texture and lighting from the source to the blended results. In face blending, this is critical, and leads to obvious, unwanted artifacts. From this, the improvement of our LRNet is clear. For instance, examine row 5 in Fig. 9: our method is the only to generate results that preserve the highlights and shadows of the source image. At the local-level, LRNet was the only to preserve the facial texture and skin tone of the source. Note that the first two rows are results for real-world images, our method performs as well on real data as it does on generated data.

Quantitative Comparison First, we performed a user study on 75 volunteers. For this, 200 image pairs were randomly sampled from the validation set. Then, each volunteer was shown the source and target faces, along with the results for the respective pair. Note that the order of results was set randomly each time. Volunteers were asked to score the resulting faces, with a score scale of 0.5-to-4

Table 1: Quantitative results of different face blending methods.

	Alpha	Possion	Style	Deep	LRNet
User rating	2.43±0.68	2.94±0.47	1.28±0.67	2.19±0.65	**3.54±0.39**
Avg. time(s)	**0.12**	0.23	0.19	54.31	0.37
FID score	39.73	31.59	73.41	26.18	**15.47**

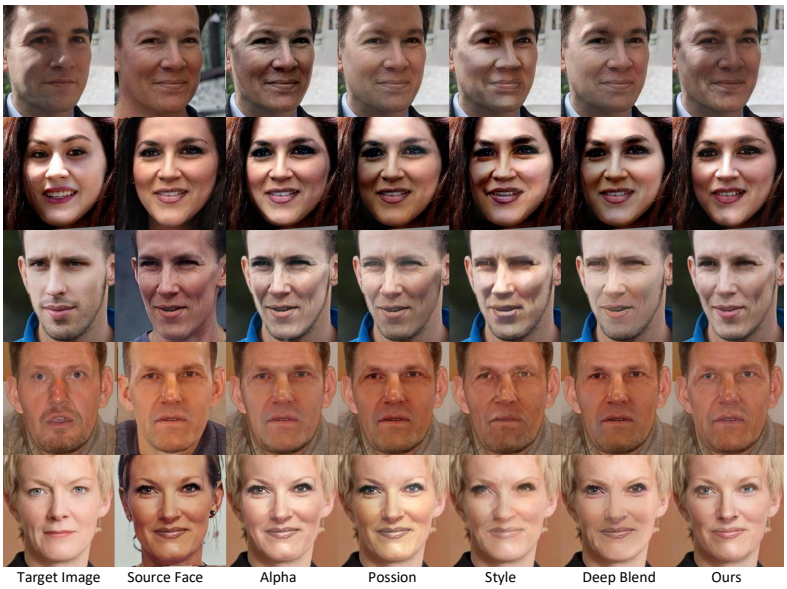


Fig. 9: Qualitative comparisons between LRNet and existing methods. The task is to blend the source faces onto the target.

(increments of 0.5). We ask users to rate according to whether the results look natural, realistic. As illustrated in Table 1, $x \pm y$ represents the mean $\pm$ the variance of all user ratings. In the end, our proposed method was perceived as superior to the others.

We apply "Fréchet Inception Distance" (FID) [43] as another quantitative evaluation criterion. The FID score are also listed in Table 1. Our method outperforms all the baselines under the objective evaluation criterion. Note that the FID score is measured across the validation set, and image size is 256×256.

Table 1 also shows the average computational time of five methods with same configuration - a PC with an Intel i7 4.20GHz CPU and an NVIDIA GTX 1080Ti GPU. It can be seen that our method is comparable with the fastest face blending method (*i.e.*, Style Blending) in terms of speed.

6 Conclusions

In this paper, we proposed an effective method for facial image blending. Firstly, we introduced a labeled facial image dataset for this task, which contains 12,000 face pairs with the same orientation and facial contour, but different appearance. Secondly, we proposed a normalization method, *i.e.*, LocalIN, to transfer local statistical information. Thirdly, we also introduced a new network, *i.e.*, LRNet, with instances of a new layer-type designed to reshuffle local patches on the forward pass. The extensive experiments demonstrate our approach are very effective compared to existing methods.

References

1. Bitouk, D., Kumar, N., Dhillon, S., Belhumeur, P., Nayar, S.K.: Face swapping: automatically replacing faces in photographs. In: ACM SIGGRAPH 2008 papers. (2008) 1–8
2. Nirkin, Y., Masi, I., Tuan, A.T., Hassner, T., Medioni, G.: On face segmentation, face swapping, and face perception. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), IEEE (2018) 98–105
3. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. In: Advances in neural information processing systems. (2017) 386–396
4. Liu, S., Ou, X., Qian, R., Wang, W., Cao, X.: Makeup like a superstar: Deep localized makeup transfer network. arXiv preprint arXiv:1604.07102 (2016)
5. Chang, H., Lu, J., Yu, F., Finkelstein, A.: Pairedcyclegan: Asymmetric style transfer for applying and removing makeup. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 40–48
6. Li, T., Qian, R., Dong, C., Liu, S., Yan, Q., Zhu, W., Lin, L.: Beautygan: Instance-level facial makeup transfer with deep generative adversarial network. In: Proceedings of the 26th ACM international conference on Multimedia. (2018) 645–653
7. Shih, Y., Paris, S., Barnes, C., Freeman, W.T., Durand, F.: Style transfer for headshot portraits. ACM Transactions on Graphics (TOG) **33** (2014) 148
8. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 1501–1510
9. Reinhard, E., Adhikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Computer graphics and applications **21** (2001) 34–41
10. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM SIGGRAPH 2003 Papers. (2003) 313–318
11. Nirkin, Y., Keller, Y., Hassner, T.: Fsgan: Subject agnostic face swapping and reenactment. (2019)
12. Li, L., Bao, J., Yang, H., Chen, D., Wen, F.: Faceshifter: Towards high fidelity and occlusion aware face swapping. (2019)
13. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. arXiv preprint arXiv:1912.04958 (2019)
14. Porter, T., Duff, T.: Compositing digital images. In: Proceedings of the 11th annual conference on Computer graphics and interactive techniques. (1984) 253–259
15. Fattal, R., Lischinski, D., Werman, M.: Gradient domain high dynamic range compression. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques. (2002) 249–256
16. Levin, A., Zomet, A., Peleg, S., Weiss, Y.: Seamless image stitching in the gradient domain. In: European Conference on Computer Vision, Springer (2004) 377–389
17. Szeliski, R., Uyttendaele, M., Steedly, D.: Fast poisson blending using multi-splines. In: 2011 IEEE International Conference on Computational Photography (ICCP), IEEE (2011) 1–8
18. Wu, H., Zheng, S., Zhang, J., Huang, K.: Gp-gan: Towards realistic high-resolution image blending. In: Proceedings of the 27th ACM International Conference on Multimedia. (2019) 2487–2495
19. Zhang, L., Wen, T., Shi, J.: Deep image blending. arXiv preprint arXiv:1910.11495 (2019)

20. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2414–2423
21. Li, Y., Wang, N., Liu, J., Hou, X.: Demystifying neural style transfer. arXiv preprint arXiv:1701.01036 (2017)
22. Li, S., Xu, X., Nie, L., Chua, T.S.: Laplacian-steered neural style transfer. In: Proceedings of the 25th ACM international conference on Multimedia. (2017) 1716–1724
23. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2479–2486
24. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.S.: Texture networks: Feed-forward synthesis of textures and stylized images. In: ICML. Volume 1. (2016) 4
25. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stylebank: An explicit representation for neural image style transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 1897–1906
26. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision, Springer (2016) 694–711
27. Li, Y., Liu, M.Y., Li, X., Yang, M.H., Kautz, J.: A closed-form solution to photorealistic image stylization. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 453–468
28. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 172–189
29. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 4401–4410
30. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. (2014) 2672–2680
31. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in neural information processing systems. (2015) 1486–1494
32. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
33. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)
34. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in neural information processing systems. (2016) 2172–2180
35. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 1125–1134
36. Liu, R., Liu, Y., Gong, X., Wang, X., Li, H.: Conditional adversarial generative flow for controllable image synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 7992–8001

37. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 4432–4441
38. Feng, Y., Wu, F., Shao, X., Wang, Y., Zhou, X.: Joint 3d face reconstruction and dense alignment with position map regression network. In: ECCV. (2018)
39. Erikson, A.P., strm, K.: On the Bijectivity of Thin-Plate Splines. (2012)
40. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV). (2015)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
42. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
43. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in neural information processing systems. (2017) 6626–6637