Deep Feature Prior Guided Face Deblurring Supplementary Material

Soo Hyun Jung[1], Tae Bok Lee[2], Yong Seok Heo[1,2]
[1]Department of Electrical and Computer Engineering, Ajou University, South Korea
[2]Department of Artificial Intelligence, Ajou University, South Korea
{tngusdldlt,dolphin0104,ysheo}@ajou.ac.kr

Overview

In this supplementary material, we provide implementation details, extended discussions, and experimental results which could not be included in the main paper due to page limitation. First, we describe the details of our DFPGnet in Section 1. Then, we conduct additional analysis on deep feature priors used for our DFPGnet in Section 2. Lastly, we show additional comparisons with state-of-the-art methods [9, 12, 11, 4] on various test sets in Section 3. Our training code, model weights and result images will be released to facilitate future research of face deblurring.

1. Details of Network Architectures

1.1. Generator

As shown in Fig. 2 in the main paper, our generator consists of three parts: the encoder of the deblurring stream $\mathcal{M}_{\mathcal{E}}$, the prior estimation stream $\mathcal{P}$, and the decoder of the deblurring stream $\mathcal{M}_{\mathcal{D}}$. For our generator, we adopt residual blocks [1] with channel attention (CA) module [2, 14] to enhance the flexibility of the network for restoration process [14]. $\mathcal{P}$ is divided into three subnetworks $\mathcal{P} = \{\mathcal{P}_i | i = 1, 2, 3\}$ with the same architecture. Each $\mathcal{P}_i$ estimates the deep feature priors. The input of $\mathcal{P}_i$ is the i^{th} intermediate feature (E_i) of $\mathcal{M}_{\mathcal{E}}$.

The detailed architectures for $\mathcal{M}_{\mathcal{E}}$, $\mathcal{P}_i$ and $\mathcal{M}_{\mathcal{D}}$ are shown in Table 1, 2 and 3, respectively. Note that "Init conv" denotes the convolutional layer that converts the RGB image to the features. "Down conv" and "Up conv" represent the convolutional layers with stride 2 for the downsampling and upsampling operation, respectively. "$\downarrow_2 (\cdot)$" indicates the downsampling operation by 2. The spatial size of the feature is denoted by "W", "H", and "C", which represent the width, height and channels, respectively. In Table 2, "W_i", "H_i", and "C_i" represent the width, height and channels of the E_i extracted from $\mathcal{M}_{\mathcal{E}}$.

Table 1. The deblurring stream encoder $\mathcal{M}_{\mathcal{E}}$ architecture.

Block	Input	Output	Kernel Size	Output Size
Init conv	I_{blur}	E_0	1	$W \times H \times 64$
Resblock × 5	E_0	E_1	3	$W \times H \times 64$
Down conv	E_1	$\downarrow_2(E_1)$	4	$W/2 \times H/2 \times 128$
Resblock × 5	$\downarrow_2(E_1)$	E_2	3	$W/2 \times H/2 \times 128$
Down conv	E_2	$\downarrow_2(E_2)$	4	$W/4 \times H/4 \times 256$
Resblock × 5	$\downarrow_2(E_2)$	E_3	3	$W/4 \times H/4 \times 256$
Resblock × 5	E_3	E_4	3	$W/4 \times H/4 \times 256$

Table 2. The prior estimation sub-network $\mathcal{P}_i$ architecture.

Block	Input	Output	Kernel Size	Output Size
$SSFT_i$	E_i	E_i^t	3	$W_i \times H_i \times C_i$
			3	$W_i \times H_i \times C_i$
Resblock × 10	E_i^t	$\hat{P}_i$	3	$W_i \times H_i \times C_i$

Table 3. The deblurring stream decoder $\mathcal{M}_{\mathcal{D}}$ architecture.

Block	Input	Output	Kernel Size	Output Size
SFT_1	D_1, P_3	D_1^t	3	$W/4 \times H/4 \times 256$
Resblock × 5	D_1^t	D_2	3	$W/4 \times H/4 \times 256$
Up conv			4	$W/2 \times H/2 \times 128$
SFT_2	D_2, P_2	D_2^t	3	$W/2 \times H/2 \times 128$
Resblock × 5	D_2^t	D_3	3	$W/2 \times H/2 \times 128$
Up conv			4	$W \times H \times 64$
SFT_3	D_3, P_1	D_3^t	3	$W \times H \times 64$
Resblock	D_3^t	D_4	3	$W \times H \times 64$
Final conv	$D_4 + E_0$	I_{deblur}	1	$W \times H \times 3$

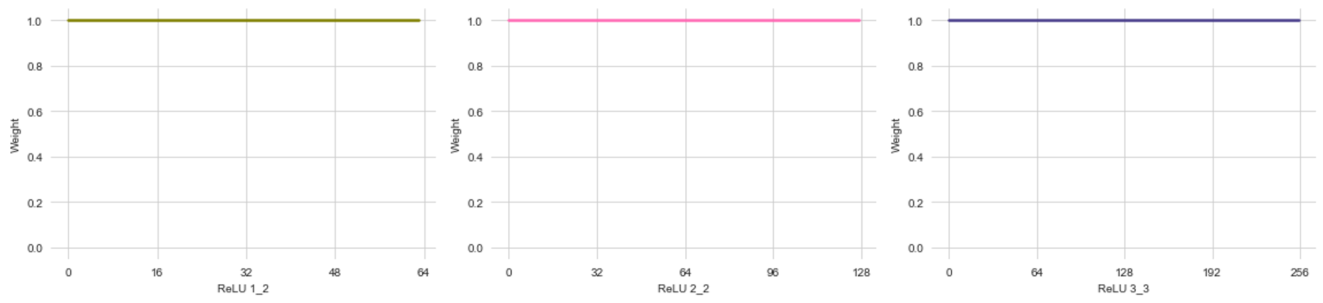
1.2. Channel-attention Feature Discriminator

We modify the feature discriminator [7] and propose to use CA module [2] to emphasize important channels for image restoration task among the channels of the learned features using the VGGFace [8].

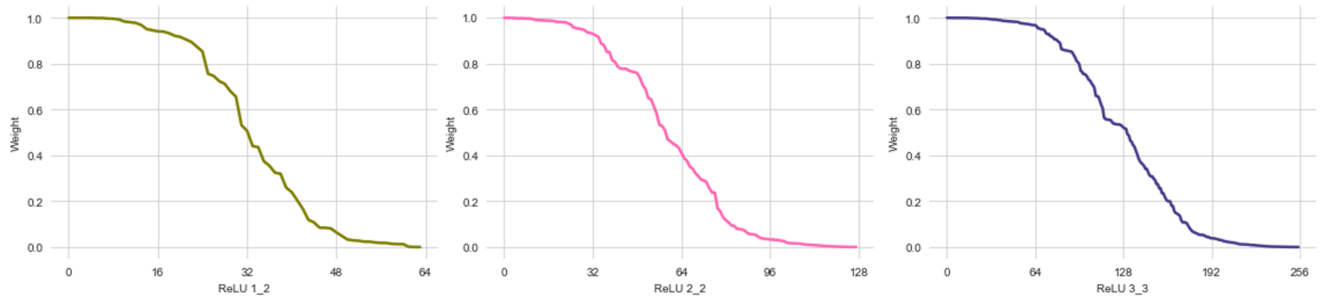
The detailed architecture of $\mathcal{D}^c$ is shown in Table 4. We apply the spectral normalization (SN) [6] and leaky ReLU (lRELU) to all convolutional layers for stable training of our discriminator. "$\oplus$" indicates the operation of channel-wise concatenation.

Table 4. The channel attention feature discriminator $\mathcal{D}^c$ architecture.

Block	Operation	Input	Output	Kernel Size	Output Size
$\mathcal{CA}_1$	Channel attention	$\hat{P}_1$	$\hat{P}_1{}'$	3	$W \times H \times 64$
$\mathcal{D}_{f,1}$	Conv2d, SN, lReLU	$\hat{P}_1{}'$	$\hat{C}_1$	3	$W \times H \times 64$
	Down Conv, SN	$\hat{C}_1$	$\downarrow_2 (\hat{C}_1)$	4	$W/2 \times H/2 \times 64$
$\mathcal{CA}_2$	Channel attention	$\hat{P}_2$	$\hat{P}_2{}'$	3	$W/2 \times H/2 \times 128$
$\mathcal{D}_{f,2}$	Conv2d, SN, lReLU	$\downarrow_2 (\hat{C}_1) \oplus \hat{P}_2{}'$	$\hat{C}_2$	3	W/2×H/2×128
	Down Conv, SN	$\hat{C}_2$	$\downarrow_2 (\hat{C}_2)$	4	$W/2 \times H/2 \times 128$
$\mathcal{CA}_3$	Channel attention	$\hat{P}_3$	$\hat{P}_3{}'$	3	$W/4 \times H/4 \times 256$
$\mathcal{D}_{f,3}$	Conv2d, SN, lReLU	$\downarrow_2 (\hat{C}_2) \oplus \hat{P}_3{}'$	$\hat{C}_3$	3	$W/4 \times H/4 \times 256$
	Down Conv, SN	$\hat{C}_3$	$\downarrow_2 (\hat{C}_3)$	4	$W/4 \times H/4 \times 256$
$\mathcal{D}_h$	Conv2d, SN, lReLU	$\downarrow_2 (\hat{C}_3)$	Final output	4	$W/8 \times H/8 \times 512$
	Conv2d, SN, Sigmoid			4	$W/16 \times H/16 \times 1$



(a) Unlearned weights



(b) Learned weights

Figure 1. Visualization of the channel weights of the deep features. Each curve shows the channel weights of the deep features which are extracted from each layer of VGGFace [8] model. (a) Unlearned weights which weight each channel of the deep features equally. (b) Learned weights which weight each channel of the deep features [8] differently according to importance of deblurring. They are sorted in descending order. The sigmoid activation function in the CA module [2] constrains the weights in the range [0,1].

2. Analysis on Deep Feature Prior

2.1. Analysis on Channel Weights of Deep Feature Prior

To verify that there are more important channels of deep features for deblurring, we analyze the distribution of learned weight value corresponding each channel. We extract the deep features from pretrained VGGFace [8] model using blurry images in the MSPL-Center Testset [4]. By passing those deep features into the CA module [2] in $\mathcal{D}^{\mathcal{C}}$, we can obtain the channel weights of the deep features for restoration. The results are shown in Fig. 1b. The deep features from `relu1_2`, `relu2_2`, and `relu3_3` layers include 64, 128, 256 channels, respectively. Among them, 39.1%, 32.3%, 37.5% of the weights for channels in each layer are over 0.8, respectively. On the other hand, 34.4%, 39.84%, 36.7% of them are less than 0.2, respectively. This means that only one-third of the channels in the deep features are considered important for restoration. Also compared to the unlearned model (see Table 4 B5 model in main paper) which gives equal weight to all channels in the deep features (Fig. 1a), the performance increases in the learned model (see Table 4 B6 model in main paper) which gives different weight to each channels in the deep features (Fig. 1b). From these experiments, we can demonstrate that some of the channels in the deep features are more important for

Table 5. PSNR and SSIM results on MSPL-Center [4] and Shen testset [9].

	MSPL-Center		Shen	
Method	PSNR	SSIM	PSNR	SSIM
VGGFace Encoder	19.92	0.691	20.18	0.719
DFPG-B(ours)	**29.23**	**0.931**	**27.13**	**0.913**

deblurring, and it is helpful to give them a higher weight rather than to give them the same weight as other channels.

2.2. Using VGGFace [8] as an Encoder Instead of Synthesizing Deep Feature Priors

One key insight of our method is that well-trained face recognition networks (*e.g*. VGGFace [8]) are powerful feature descriptors of "sharp" facial images not "blurry" facial images. It is inferred from the results of face detection and verification experiment (see Table 3 in main paper). The performances of face detection and verification are significantly decreased from 96.00% to 77.40% in face detection, and from 93.47% to 77.05% in face verification, when input image is blurry. This indicates that even the well-trained face recognition networks fail to capture accurate information about the face when the input image is blurry. If such inaccurate information is used as prior information for restoration process, we hypothesize that restoration performance will be degraded. Based on this hypothesis, we aim to synthesize the deep features using blurry input image, so that they can be similar to "sharp deep features" obtained from VGGFace using "GT sharp face image".

To verify our hypothesis, we conduct an experiment that directly utilizes the "blurry deep features" of VGGFace for face deblurring. Here, the blurry deep features represent the deep features obtained from VGGFace using "blurred facial image". To be specific, we replace the encoder of our generator with the pre-trained VGGFace and fix the weights of VGGFace. We remove our prior estimation stream and directly connect the VGGFace and our decoder with skip connection. For fair comparison, the `relu1_2`, `relu2_2`, and `relu3_3` layers of VGGFace [8] is connected to the decoder, as same as our proposed DFPG network. Then, we train the decoder using Eq. (7) in our main paper.

The results are shown in Table 5. We denote this additional experiment as "VGGFace Encoder" in Table 5. The results of the VGGFace Encoder are worse than those of the proposed method. In addition, the VGGFace Encoder performs worse than the B1 model of Table 4 in the main paper. Considering that the B1 model is a model that does not use deep feature priors for face deblurring, it can be seen that it is very inefficient to directly use the deep features of VGGFace obtained from the blurry image.

Table 6. Comparison with state-of-the-art methods for average run time and model parameters.

Method	Implementation	Run time (sec)	Parameters (M)
Shen *et al*. [9]	MATLAB(GPU)	0.05	14.8
Lu *et al*. [5]	Pytorch(GPU)	0.02	53.0
Xia *et al*. [10]	Tensorflow(GPU)	0.19	41.8
Yasarla *et al*. [11]	Pytorch(GPU)	0.16	14.4
Lee *et al*. [4]	Pytorch(GPU)	0.08	18.5
Ours	Pytorch(GPU)	0.05	44.7

3. Additional Experimental Results

3.1. Inference Time and Model Parameters

In Table 6, we compare the inference time and the number of model parameters with existing methods. Following [9, 4], we measure the average of inference time for 10 images with input size of $128 \times 128 \times 3$ on a single NVIDIA Tital XP GPU. The results show that our model is clearly faster than other prior-based models [11, 4]. The speeds of [9] and our model are comparable. However, when considering both the deblurring accuracy (see Table 2 in the main paper) and the inference speed, we can say that our model performs best in face deblurring.

3.2. Qualitative Comparisons on Real Blur

The more results on real blur are shown in Fig. 2. The results of our method (see the last column in Fig. 2) demonstrate that our method performs best for the real blurred facial images. For example, our method restores more fine textures (*i.e*., hair and wrinkles) compared to methods of [9, 11, 4], as shown in the 1^{st} and 2^{nd} rows in Fig. 2. Our method produces images with more details of small facial components (*i.e*., eyes, lips, and teeth) than others (see the 3^{rd} and 4^{th} rows). In addition, the results in the last row show that our method can restore images with fewer artifacts than others.

3.3. Qualitative Comparisons on Benchmark

We provide more qualitative comparisons with state-of-the-art methods [9, 12, 11, 4] on face deblurring test sets [9, 4]. Note that the notations for the models compared in this supplementary material are the same as described in the main paper. The results for Shen test set [9] and MSPL test set [4] are shown in Fig. 3 and 4, respectively. In our comparison, our method restores more visually pleasing results with sharper edges and richer textures compared to others.

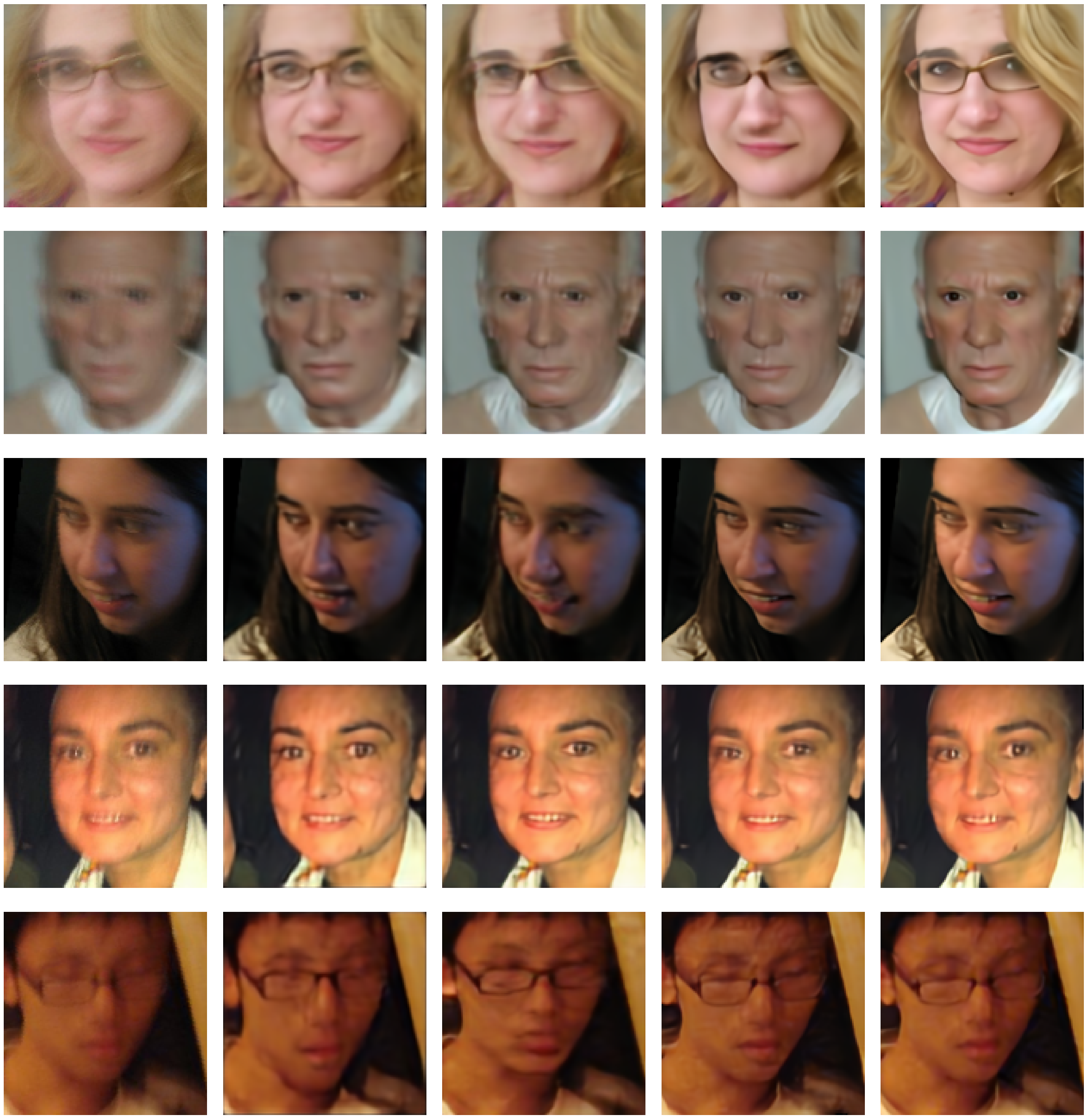


Figure 2. Qualitative comparisons on Real-Blur test set [3].

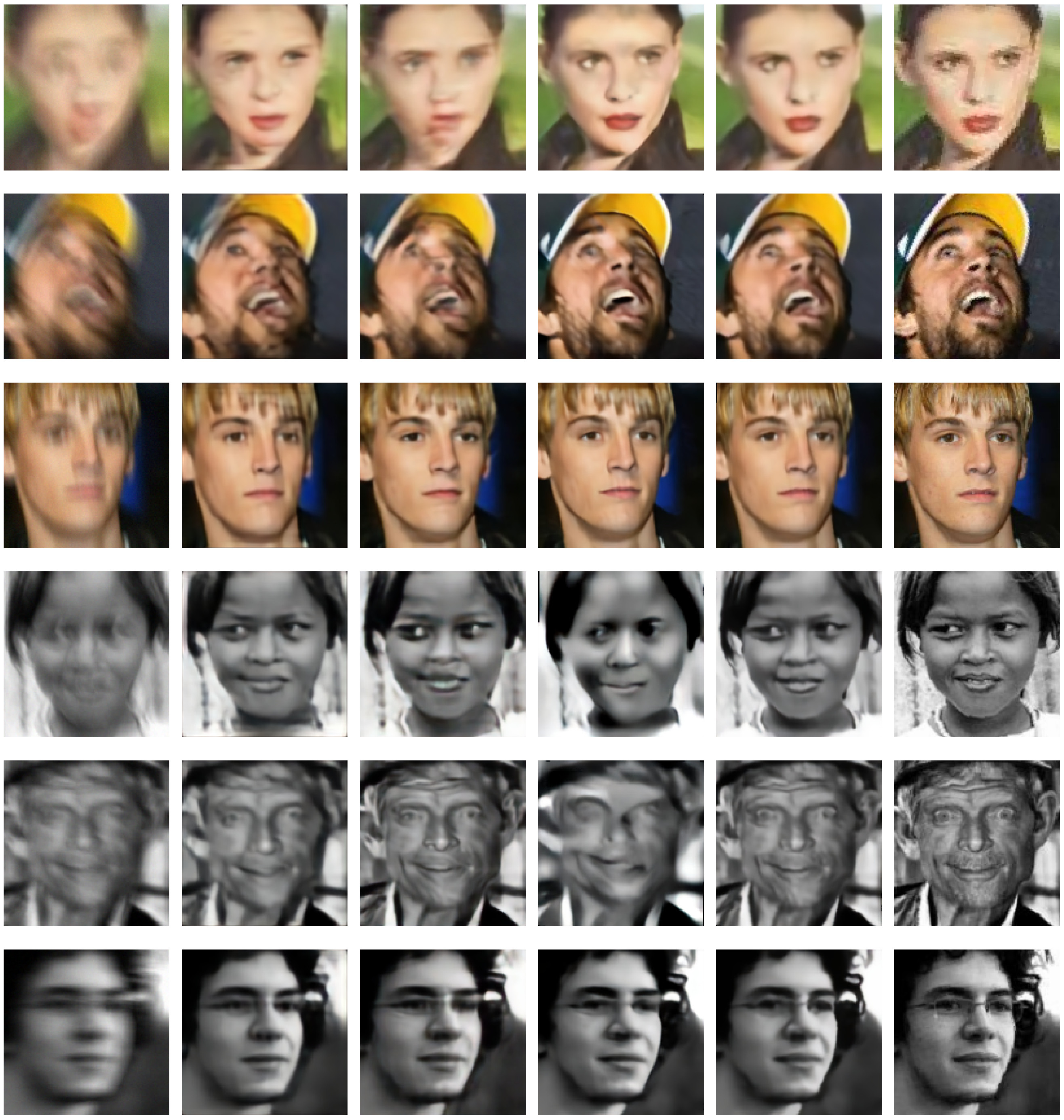


Figure 3. Qualitative comparisons on Shen test set [9].

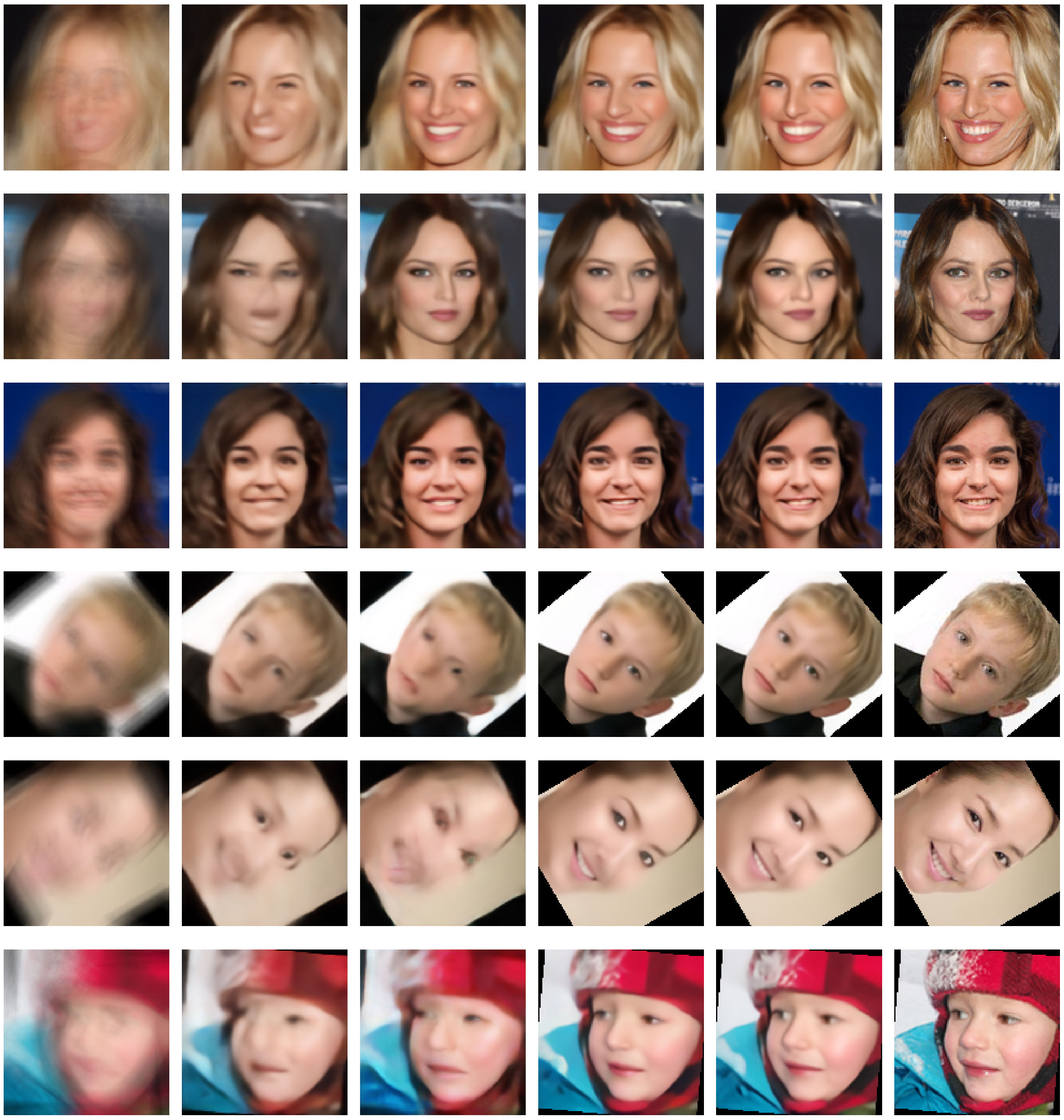


Figure 4. Qualitative comparisons on MSPL test set [4].

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Jie Hu, Li Shen, and Gang Sun. *Squeeze-and-excitation networks*. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 7132–7141, June 2018.

[3] Wei-Sheng Lai, Jia-Bin Huang, Zhe Hu, Narendra Ahuja, and Ming-Hsuan Yang. A comparative study for single image blind deblurring. In *IEEE Conferene on Computer Vision and Pattern Recognition*, 2016.

[4] Tae Bok Lee, Soo Hyun Jung, and Yong Seok Heo. Progressive semantic face deblurring. *IEEE Access*, 8:223548–223561, 2020.

[5] Boyu Lu, Jun-Cheng Chen, and Rama Chellappa. *Unsupervised domain-specific deblurring via disentangled representations*. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 10225–10234, June. 2019.

[6] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[7] Seong-Jin Park, Hyeongseok Son, Sunghyun Cho, Ki-Sang Hong, and Seungyong Lee. Srfeat: Single image super-resolution with feature discrimination. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 439–455, 2018.

[8] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. *Deep face recognition*. 2015.

[9] Ziyi Shen, Wei-Sheng Lai, Tingfa Xu, Jan Kautz, and Ming-Hsuan Yang. *Deep semantic face deblurring*. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 8260–8269, June 2018.

[10] Zhihao Xia and Ayan Chakrabarti. *Training Image Estimators without Image Ground Truth*. In *Adv. Neural Inf. Process. Syst. (NIPS)*, pages 2436–2446, June. 2019.

[11] Rajeev Yasarla, Federico Perazzi, and Vishal M Patel. *Deblurring face images using uncertainty guided multi-stream semantic networks*. *IEEE Trans. Image Process.*, Apr. 2020.

[12] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5978–5986, 2019.

[14] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301, 2018.