HiFaceGAN: Supplementary Material

Anonymous Author(s)*

Figure 1: The "first-page stunner" of face renovation results. Start from left: column 1-2: super resolution; 3-4: hallucination; 5-6: face renovation. Can you guess which row is generated by our HiFaceGAN?

1 IMPLEMENTATION DETAILS

To facilitate the reproduction of the main results in our paper, we provide all necessary implementation details on the proposed face renovation network. Specifically, the degradation simulation script as well as the exact network configuration will be precisely described.

1.1 Degradation Simulation

Here we provide the python script for adding degradations upon a 512×512 image, where most of the intensity parameters can be customized per taste.

```python
import cv2
import os
from tqdm import tqdm
import numpy as np
import imgaug.augmenters as ia

def get_down():
    return ia.Sequential([
        # random downsample between 4x to 8x and get back
        ia.Resize((0.125,0.25)),
        ia.Resize({"height": 512, "width": 512}),
    ])

def get_noise():
    return ia.OneOf([
        ia.AdditiveGaussianNoise(scale=(20,40), per_channel=True),
        ia.AdditiveLaplaceNoise(scale=(20,40), per_channel=True),
        ia.AdditivePoissonNoise(lam=(15,30), per_channel=True),
    ])

def get_blur():
    return ia.OneOf([
        ia.MotionBlur(k=(10,20)),
        ia.GaussianBlur((3.0, 8.0)),
    ])

def get_jpeg():
    return ia.JpegCompression(compression=(50,85))

def get_full():
    return ia.Sequential([
        get_blur(),
        get_noise(),
        get_jpeg(),
        get_down(),
    ], random_order=True)

def get_by_suffix(suffix):
    if suffix == 'down':
        return get_down()
    # elif...
    else:
        raise('%s not supported' % suffix)

def create_mixed_dataset(input_dir, suffix='full'):
    output_dir = input_dir + '_' + suffix
    if not os.path.isdir(output_dir):
        os.mkdir(output_dir)
    trans = get_by_suffix(suffix) # or use other functions
    mix_degrade = lambda x: trans.augment_image(x)

    for item in tqdm(os.listdir(input_dir)):
        # We arrange the data in [LR, HR] format
        # change the following script to fit your needs
```

```
        hr = cv2.imread(os.path.join(input_dir, item))
        lr = mix_degrade(hr)
        img = np.concatenate((lr,hr), axis=1)
        cv2.imwrite(os.path.join(output_dir, item), img)

if __name__ == '__main__':
    suffix = 'degradation_type' # [down/16x/noise/blur/jpeg/full]
    source_dir = '/path/to/your/source/directory'
    create_mixed_dataset(source_dir, suffix)
```

1.2 Network Configuration

Notations To ease the notation, we introduce some abbreviations for frequently used components:

- C_o^s The usual "conv-norm-relu" combination for 2D tensors. The convolution layer contains o filters, with 3×3 kernel size, 1 pixel reflective padding and stride s. Instance Normalization and LeakyReLU(0.2) are used throughout the paper.
- L_m^n A linear layer mapping points in $\mathbb{R}^m$ to $\mathbb{R}^n$. Additionally, $\tilde{L}_m^n$ indicates a linear layer with $p = 0.5$ dropout.
- G_m A two-layer perceptron that is used for adaptive convolution. It is equivalent to L_m^n - IN_n - $ReLU$ - L_n^n where $n = m//2$ in our implementation.
- CAS_{in}^{out} The proposed content-adaptive suppression (CAS) module with an associated adaptive kernel G_{in} and $\mathcal{D} = Tanh(\cdot)$ activation to restrict the range of modulation weights in $[-1, 1]$.
- S_c^s The SPADE module as prescribed in the original paper [2] with c input channels. If $s > 1$, an additional upsampling layer is appended to enlarge the result tensor by s times with bicubic interpolation.
- SR_c^s A SPADE residual block containing two S_c^s modules in the main pathway.

Below lists the architecture of the suppression part, where the base number of filters are set to 48 to economize the graphic memory usage (roughly 11GB for batch size 2):

$$C_{48}^1 - CAS_{96}^2 - CAS_{192}^2 - CAS_{384}^2 - CAS_{384}^2 - CAS_{384}^2$$

with a $384 \times 16 \times 16$ tensor as the final output. Correspondingly, the architecture of the replenishment part is:

$$SR_{384}^2 - SR_{384}^1 - SR_{384}^2 - SR_{384}^2 - SR_{192}^2 - SR_{96}^2 - S_{48}^1 - C_3^1$$

where the activation is replaced with $Tanh$ in the last 3-channel convolution block. We adhere to the official implementation of SPADE generator [2] by introducing two cascaded SPADE residual blocks in the stage 4 replenishment module, sharing the same piece of semantic guidance encoded with the corresponding suppression module.

For adversarial learning, we adopt the multi-scale discriminator in [3], with two identical subnetworks of the following architecture:

$$C_{64}^1 - C_{128}^2 - C_{256}^2 - C_{512}^2 - C_{512}^2 - C_1^1$$

1.3 Training Settings

We resize all the input images to 512×512 with bicubic interpolation before training, and use the same network architecture and loss functions for face renovation and all related subtasks, where the corresponding weights λ_{FM} and λ_{perc} are both set to 10.0. For parameter updating, we use the Adam optimizer [1] for all networks, with an initial learning rate of $lr = 2e-4$. The learning rate is fixed for 20 epochs, and linearly decays to 0 in another 10 epochs. The network is implemented in PyTorch, which takes roughly 2 days on a single Nvidia P100 GPU with 16 GB memory.

2 MORE HIGH-RESOLUTION RESULTS

We provide more examples of our HiFaceGAN on various face restoration subtasks against state-of-the-art baselines in the following pages. Note that we didn't pose the results for JPEG artifact removal task since the visual difference produced by our method is barely noticeable. The corresponding image ids are listed below for reference.

```
ids_dict_supp = {
    '16x': [65295, 65289, 65287, 65280, 65272, 65267, 65220],
    'noise': [65309, 65312, 65341, 65352, 65360, 65377, 65386],
    'blur': [65415, 65421, 65445, 65460, 65475, 65482],
    'fr': [65578, 65577, 65574, 65561, 65550, 65539, 65532, 65531,
      65517, 65511]
}
```

For image super resolution, baseline names are marked under corresponding patches. For other experiments, the order is the same as main paper, which we list here again for readers convenience (Left first column is the input, right two columns are our renovation result and the ground truth):

- **Hallucination** Super-FAN, ESRGAN, WaveletCNN
- **Denoising** RIDNet, WaveletCNN, VDNet
- **Deblurring** DeblurGAN, DeblurGANv2
- **Face Renovation** Super-FAN, ESRGAN, WaveletCNN, DeblurGANv2, ARCNN, GFRNet

Oh and by the way, the second row in Fig. 1 is generated by us.

REFERENCES

[1] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[2] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic Image Synthesis With Spatially-Adaptive Normalization. In *CVPR*.

[3] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs. In *CVPR*.

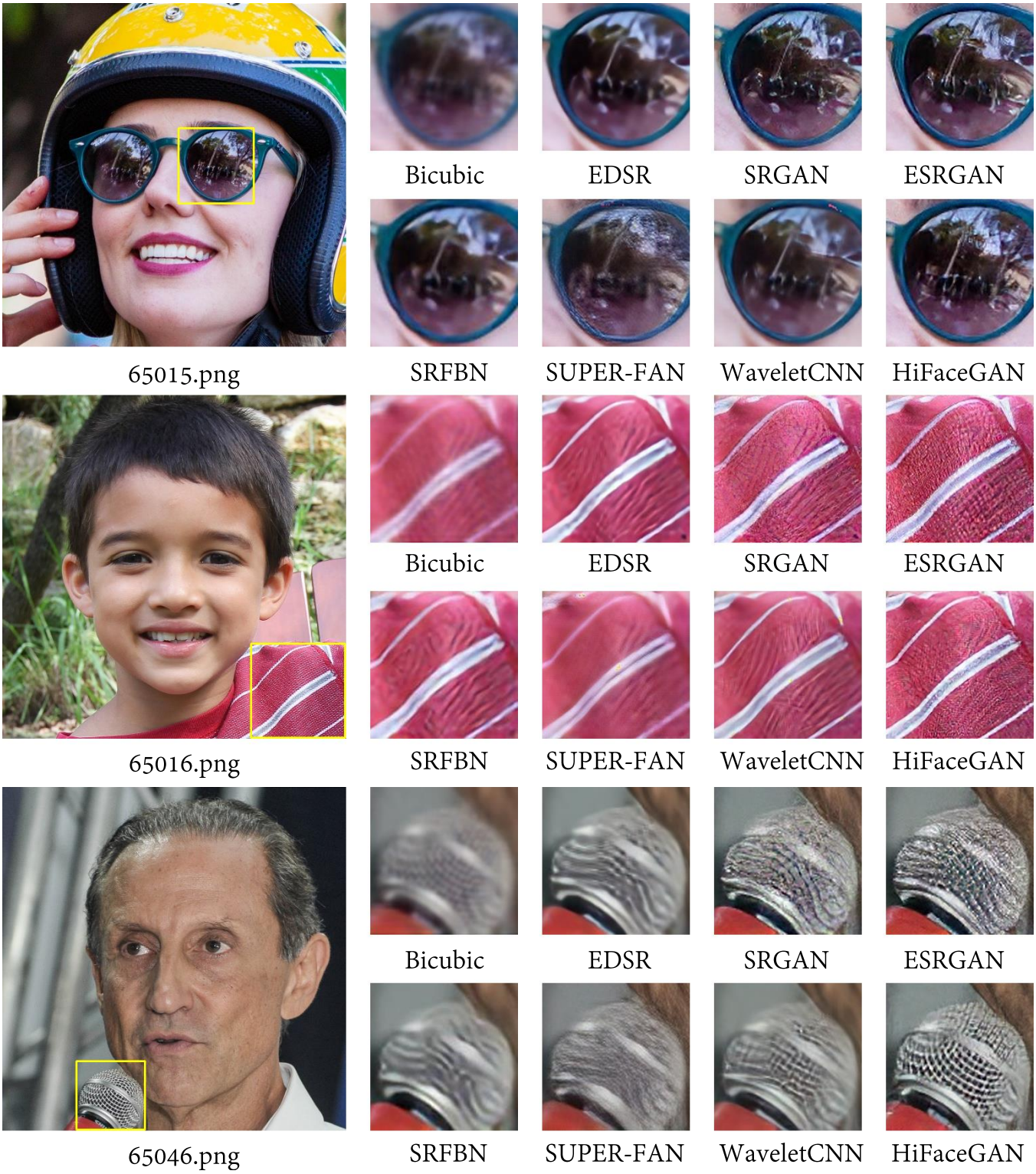

Figure 2: More results on the 4x face super resolution subtask (1/2).

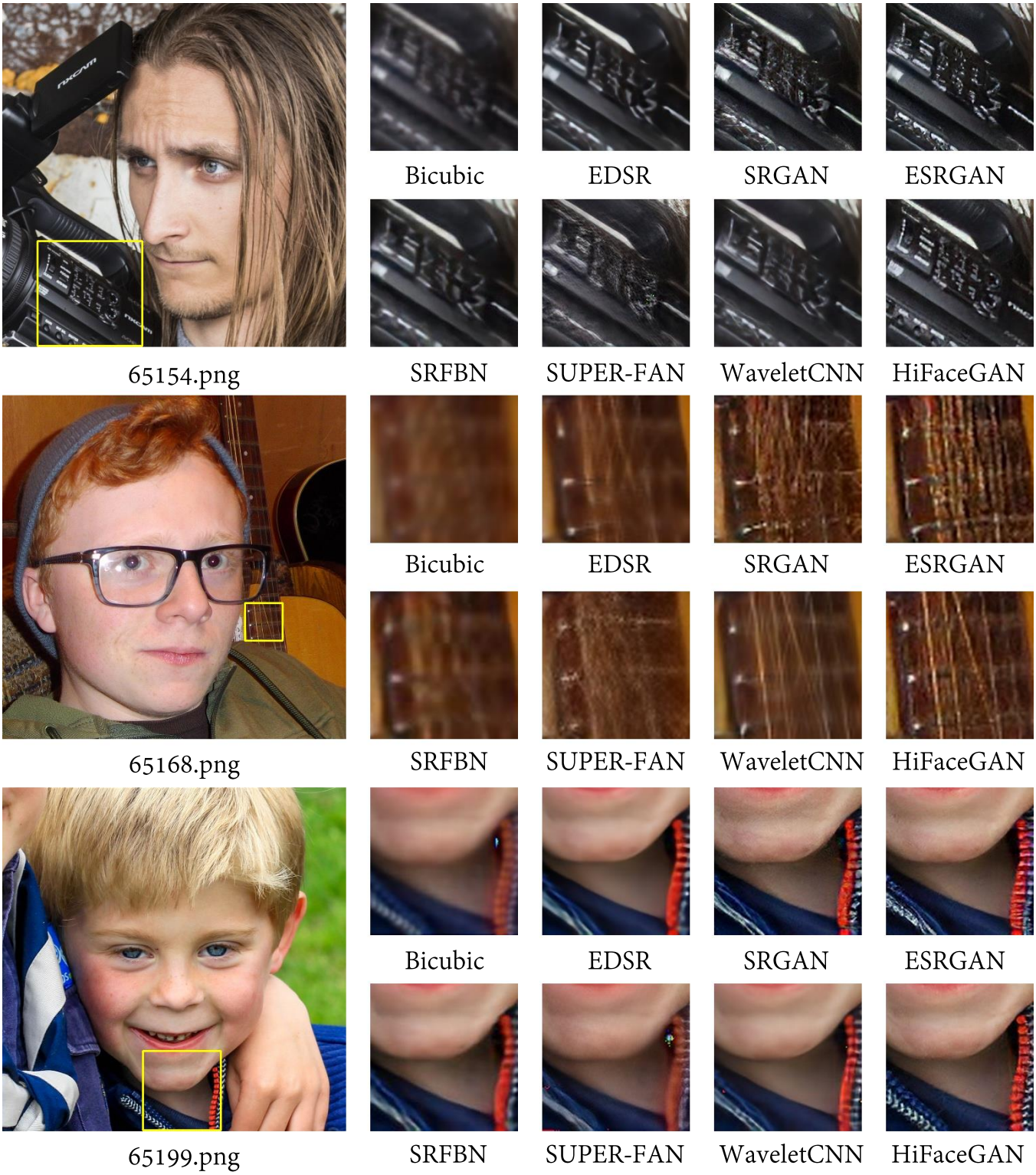

Figure 3: More results on the 4x face super resolution subtask (2/2).

Figure 4: More results on the 16x face hallucination subtask.

Figure 5: More results on the image denoising subtask.

Figure 6: More results on the image deblurring subtask.

Figure 7: More results on the main task: face renovation.

Figure 8: More results on real-world face renovation. From Top to bottom: Bruce Lee and Nora Miao; Charles Chaplin and Paulette Goddard, *Modern Times*; Jim Parsons as *Sheldon Lee Cooper*, *The Big Bang Theory*.