Supplementary for Hierarchical Conditional Flow: A Unified Framework for Image Super-Resolution and Image Rescaling

Jingyun Liang[1] Andreas Lugmayr[1] Kai Zhang[1] Martin Danelljan[1] Luc Van Gool[1,2] Radu Timofte[1]
[1]Computer Vision Lab, ETH Zurich, Switzerland [2] KU Leuven, Belgium
{jinliang, andreas.lugmayr, kaizhang, martin.danelljan, vangool, timofter}@vision.ee.ethz.ch
https://github.com/JingyunLiang/HCFlow

In this supplementary, we first give more details on flow layers used in our framework. Then, we analyze the influences of sampling temperatures and random sampling. Last, we show more visual results on general image SR, face image SR and image rescaling.

1. Flow Layer Details

1.1. Squeeze Layer

Squeeze layer [3] is used to increase the number of channels by trading the spatial size. It reshapes each 2×2 neighborhood to the channel dimension. The input and output tensor sizes are $H \times W \times C$ and $\frac{H}{2} \times \frac{W}{2} \times 4C$, respectively. This layer is only used in image SR.

1.2. Haar Transform Layer

Haar transform layer [9] is an alternative to the squeeze layer. We use this layer to replace the squeeze layer for image rescaling. Based on low-pass filtering, it decomposes a $H \times W \times C$ input to a $\frac{H}{2} \times \frac{W}{2} \times C$ low-pass representation and three $\frac{H}{2} \times \frac{W}{2} \times C$ residual components, which contains high-frequencies in vertical, horizontal and diagonal directions, respectively.

1.3. Actnorm Layer

Actnorm layer [5] is used for channel-wise normalization. Similar to batch normalization [4], it is an affine transformation with learnable scale and translation parameters.

1.4. Invertible 1×1 Convolution Layer

Invertible 1×1 convolution layer [5] is a generalization of the channel permutation operation. With learnable parameters, it can better help each dimension affect every other dimension. In image rescaling, to divide the LR image and the rest high-frequencies apart from early flow layers, we remove this layer from the flow-steps of the main flow branch, and simply exchange the first 3 channels and the rest channels as a permutation operation.

1.5. Affine Coupling Layer

Affine coupling layer [3] first splits the input $\mathbf{h}^k$ into two partitions $\mathbf{h}^k_{1:d}$ and $\mathbf{h}^k_{d+1:D}$ along the channel dimension. Then, the output $\mathbf{h}^{k+1}$ are computed as follows,

$$\begin{cases} \mathbf{h}^{k+1}_{1:d} = \mathbf{h}^k_{1:d} \\ \mathbf{h}^{k+1}_{d+1:D} = \mathbf{h}^k_{d+1:D} \odot \exp(s(\mathbf{h}^k_{1:d})) + t(\mathbf{h}^k_{1:d}) \end{cases}, \quad (1)$$

where s and t are the scale and translation functions. $\odot$ represents the Hardmard product. Generally, scale and translation functions are implemented by a single small neural network whose input and output dimensions are d and $2(D-d-1)$, respectively. In experiments, we use three convolutional layers with ReLU activation functions for image SR, and use a RRDB block [7] for image rescaling.

Generally, $\mathbf{h}^k$ is evenly split by setting $d = \frac{D}{2}$. For image rescaling, we set d to 3 or $D-3$ alternatively to divide the low-frequencies and the high-frequencies apart from early flow layers. We also find that using translation operation is enough when we are transforming the low-frequencies (*i.e.*, when $d = D-3$).

1.6. Conditional Affine Coupling Layer

Conditional affine coupling layer [2, 8] helps to construct conditional flows by taking the conditioning variable $\mathbf{u}$ as an input of the affine coupling layer. It is formulated as follows,

$$\begin{cases} \mathbf{h}^{k+1}_{1:d} = \mathbf{h}^k_{1:d} \\ \mathbf{h}^{k+1}_{d+1:D} = \mathbf{h}^k_{d+1:D} \odot \exp(s([\mathbf{h}^k_{1:d}, \mathbf{u}])) + t([\mathbf{h}^k_{1:d}, \mathbf{u}]) \end{cases}, \quad (2)$$

where the concatenation of $\mathbf{u}$ and $\mathbf{h}^k_{1:d}$ is used as the input of scale and translation functions.

1.7. Split layer

Split layer [3] is used to split the tensor into two partitions. After the squeeze operation and other transformations, the $\frac{H}{2} \times \frac{W}{2} \times 4C$ tensor is split to two $\frac{H}{2} \times \frac{W}{2} \times 2C$

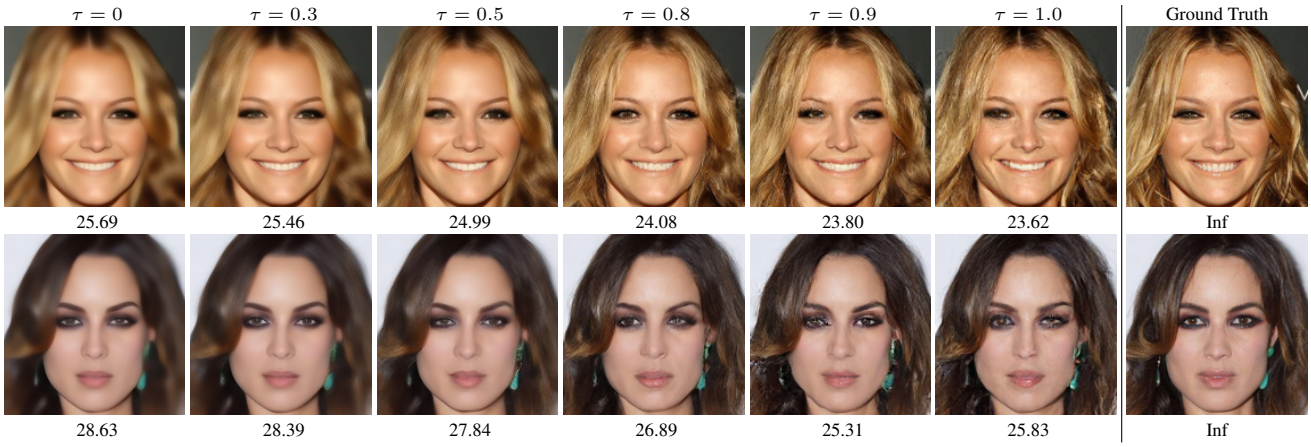

Figure 1: Visual results of face image SR ($\times 8$) when sampled with different temperatures. The corresponding PSNR is shown under each image.

Figure 2: Visual results of face image SR ($\times 8$) when randomly sampled for several times.

tensors. For the last flow level, we split it to a $\frac{H}{2} \times \frac{W}{2} \times 3$ tensor (*i.e.*, the LR image) and a $\frac{H}{2} \times \frac{W}{2} \times (4C-3)$ tensor.

2. Influence of the Sampling Temperature

As analyzed in the paper, sampling temperature has great impact on both PSNR and visual metrics. We show the SR images sampled with different temperatures in Fig. 1. As we can see, HCFLow achieves best PSNR when $\tau = 0$, though the images tend to be blurry. When τ is increased, the visual quality is improved, at the cost of dropping PSNR. When $\tau = 0.8$, HCFlow generates the most visual-pleasing images with clear edges and photo-realistic details. When τ is further increased to be 0.9 or 1.0, the generated images may be over-sharpened and suffer from artifacts.

3. Influence of Random Sampling

In Fig. 2, we show diverse photo-realistic SR results of HCFlow by randomly sampling from the latent space. The sampling temperature τ is set to 0.8. As can be seen, different samples of the latent variable lead to SR images with different details such as eyes and eyebrows. Note that most of these SR images are visual-pleasing and consistent with the LR image.

4. More Visual Comparisons.

We provide more visual comparisons on general image SR, face image SR and image rescaling in Fig. 3, Fig. 4 and Fig. 5 respectively, to clearly show the effectiveness of the proposed HCFlow.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017. 3, 4

[2] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019. 1

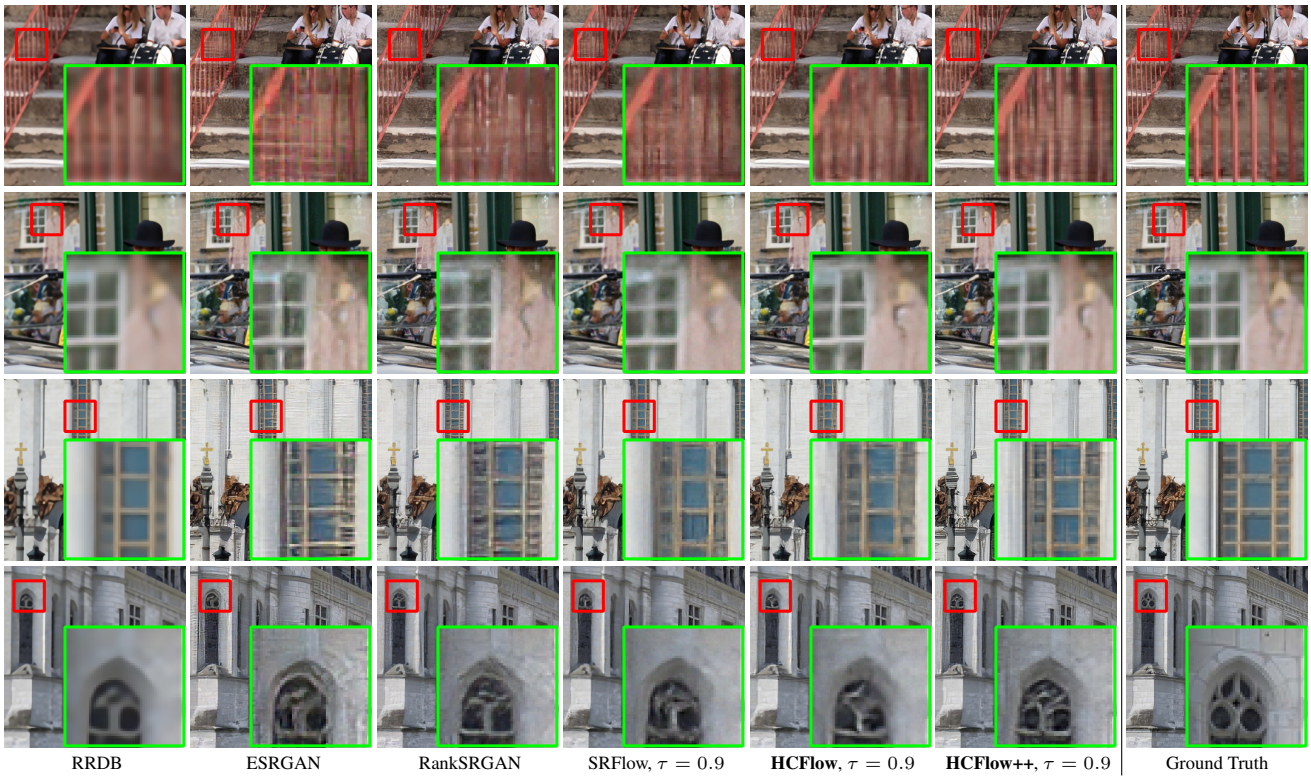

Figure 3: More visual results of general image SR ($\times 4$) on the DIV2K [1] validation set.

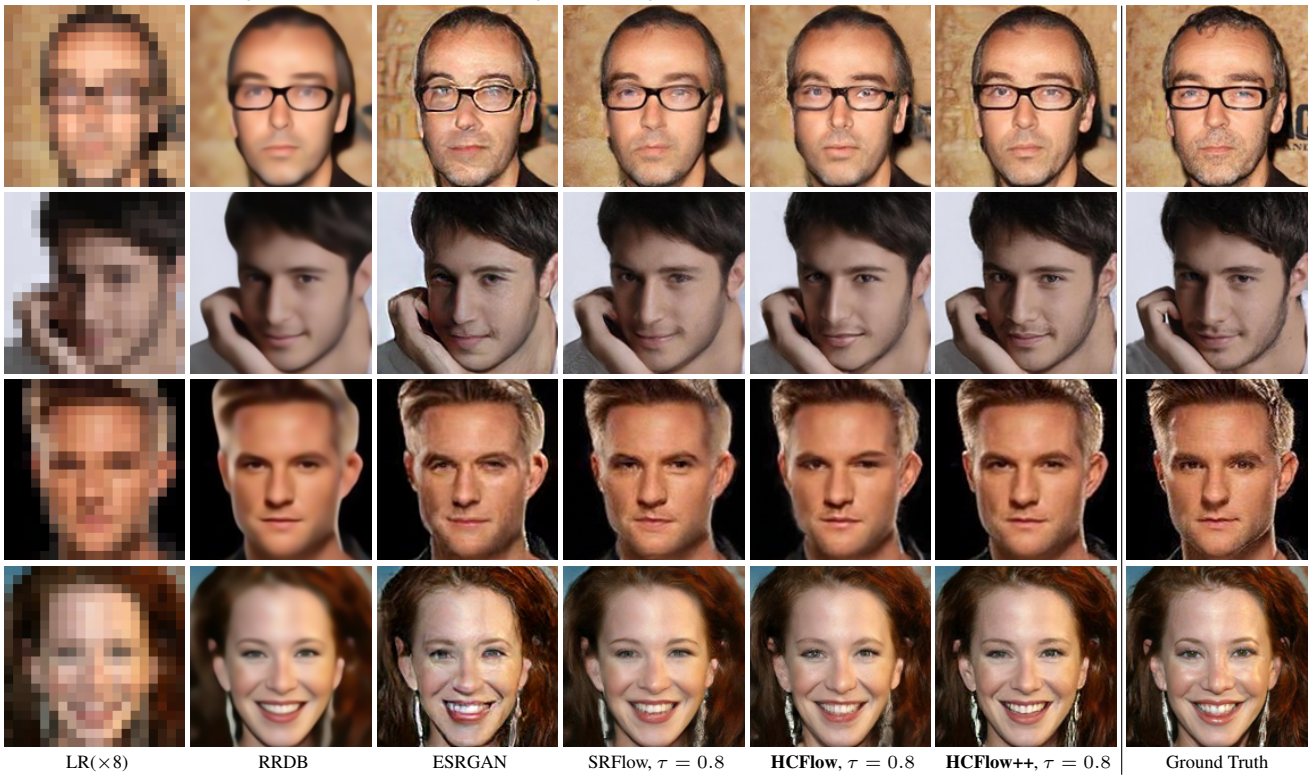

Figure 4: More visual results of face image SR ($\times 8$) on the CelebA [6] testing set.

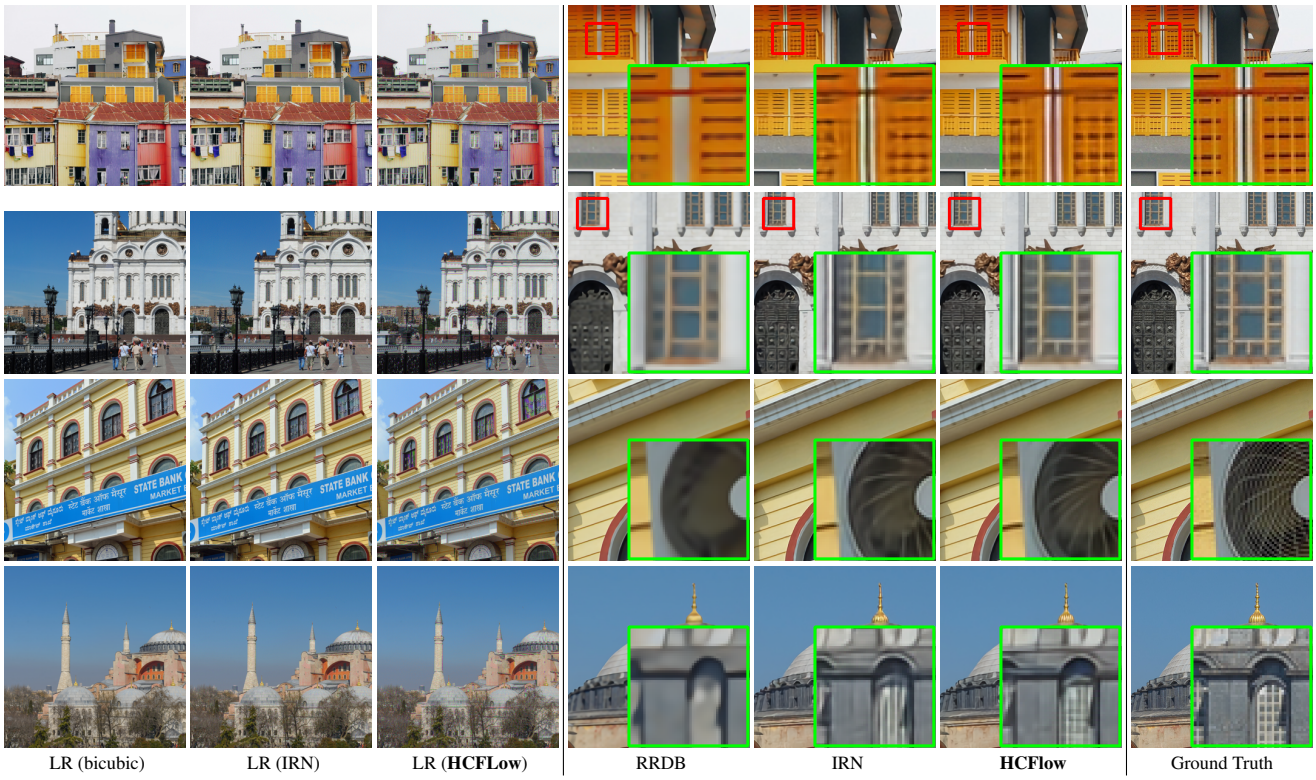

Figure 5: More visual results of image rescaling ($\times 4$) on the DIV2K [1] validation set. The first, second and third columns show the LR images generated by bicubic interpolation, IRN and HCFlow, respectively.

[3] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 1

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 1

[5] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018. 1

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *IEEE Conference on International Conference on Computer Vision*, pages 3730–3738, 2015. 3

[7] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *European Conference on Computer Vision Workshops*, pages 701–710, 2018. 1

[8] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019. 1

[9] Mingqing Xiao, Shuxin Zheng, Chang Liu, Yaolong Wang, Di He, Guolin Ke, Jiang Bian, Zhouchen Lin, and Tie-Yan Liu. Invertible image rescaling. In *European Conference on Computer Vision*, pages 126–144, 2020. 1