Supplementary Document for "Face2Face$^{\rho}$: Real-Time High-Resolution One-Shot Face Reenactment"

Kewei Yang[1], Kang Chen[1], Daoliang Guo[1], Song-Hai Zhang[2], Yuan-Chen Guo[2], and Weidong Zhang[1]

[1] NetEase Games AI Lab, Hangzhou, P.R. China
{yangkewei,ckn6763,guodaoliang,zhangweidong02}@corp.netease.com
[2] Tsinghua University, Beijing, P.R. China

This document supplements the paper entitled "Face2Face$^{\rho}$: Real-Time High-Resolution One-Shot Face Reenactment". Specifically, we provide additional comparison results in Sec. 1, give detailed information about the network architectures in Sec. 2, and present more high-resolution results in Sec. 3.

1 More comparison results

We provide some additional comparison results in this section.

Comparisons under similar complexity. To evaluate the performances of the six baseline methods under similar computational complexity (i.e., measured by MACs) to Face2Face$^{\rho}$, we reduce the minimum and the maximum number of channels in their generators to match our framework. The modified channel configurations are as follows:

- FS-VID2VID*: from 32 and 1024 to 8 and 64,
- Bi-layer*: from 32 and 256 to 16 and 128,
- LPD*: from 64 and 512 to 16 and 128,
- FOMM*: from 64 and 512 to 12 and 64,
- MRAA*: from 64 and 512 to 12 and 64,
- HeadGAN*: from 32 and 512 to 6 and 96.

The complexity of the modified methods is shown in Tab. 1. By observing the qualitative and quantitative comparison results in Fig. 1 and Tab. 2, the performances deteriorate greatly after reducing the capacity of their generators, which indicates none of the current state-of-the-art one-shot approaches can be successfully adapted to meet the requirements of real-time applications.

Comparisons with other one-shot methods. We also acknowledge other state-of-the-art one-shot face reenactment methods, i.e., MarioNetTe [1], AAN [7], LSR [3] and FVTH [6], but do not compare to them extensively in the main paper due to the unavailability of the source code. Here, we provide a small-scale qualitative comparison with these methods. Specifically, the results of MarioNetTe [1], AAN [7], LSR [3] are taken from the respective papers. Since an online pose editing demo is available for FVTH [6], we compare against FVTH on the head pose editing task. As shown in Figs. 2 to 5, Face2Face$^{\rho}$ can produce

Fig. 1. Qualitative comparisons with the modified baselines, on the task of reenactment.

results with equal or better visual quality to MarioNetTe, AAN, LSR and FVTH. Note that the online demo of FVTH only supports modifying the Euler angles of the head pose up to 30°. In contrast, Face2Face$^{\rho}$ can support even larger head pose modification without any noticeable artifacts.

It is worth noting that, none of these methods can achieve real-time performance like Face2Face$^{\rho}$. Specifically, the architectures of MarioNetTe, AAN, and FVTH are similar to FOMM while the architecture of LSR is similar to LPD, it can be assumed that their complexities are also at the same levels. We refer readers to Table 1 in the main paper for the complexity of FOMM and LPD.

Comparison with a many-shot method. Besides, we also make a comparison against one of the state-of-the-art many-shot face reenactment methods, i.e., Neural Voice Puppetry (NVP) [5]. The results are illustrated in Fig. 6, from which we can see that the results generated by Face2Face$^{\rho}$ are comparable to those synthesized by NVP when high-quality inputs (i.e., articulate neutral faces under a frontal view) are given.

Ablation study about the loss function. Finally, we conducted another ablation study to assess the significance of each loss term in Eq. 1 of our main paper. As can be seen in Tab. 3, the complete loss function outperforms all variations when one of the loss terms is removed. In terms of scores, we observe that the reconstruction loss $\mathcal{L}_G^r$ and warping loss $\mathcal{L}_G^w$ are two essential components of our loss function which give the fundamental constraints in face reenactment. From the qualitative comparison results in Fig. 8, we can see that the adversarial loss $\mathcal{L}_G^{adv}$ is effective at increasing the photo-realism of the generated results. The use of feature matching loss $\mathcal{L}_G^{fm}$ can stabilize the training to eliminate artifacts in the generated results (e.g., the first row of Fig. 8).

Table 1. Complexity comparisons with the modified state-of-the-art methods. The inference time (Inf.) and GPU memory (Mem.) are measured on an Nvidia GeForce RTX 2080Ti GPU with FP32 and FP16 mode respectively. The "-" indicates unmeasurable, due to not supporting FP16 mode.

Method	256×256			512×512			1024×1024		
	MACs↓ $\times 10^9$	Inf.(ms)↓ FP32/FP16	Mem.(GB)↓ FP32/FP16	MACs↓ $\times 10^9$	Inf.(ms)↓ FP32/FP16	Mem.(GB)↓ FP32/FP16	MACs↓ $\times 10^9$	Inf.(ms)↓ FP32/FP16	Mem.(GB)↓ FP32/FP16
FS-VID2VID*	1.9	10.3/-	0.8/-	7.5	23.2/-	0.9/-	30.1	55.7/-	1.4/-
Bi-layer*	1.5	3.6/3.3	0.9/0.8	6.0	5.5/4.7	1.0/0.9	24.0	16.7/13.9	1.1/0.9
LPD*	2.3	6.8/6.7	0.9/0.8	9.2	10.2/9.6	0.9/0.9	36.8	26.8/19.0	1.4/1.2
FOMM*	2.3	5.9/-	1.0/-	9.2	11.8/-	1.0/-	37.2	28.9/-	1.4/-
MRAA*	2.3	6.5/-	1.0/-	9.4	12.7/-	1.0/-	37.6	34.7/-	1.4/-
HeadGAN*	1.8	4.0/3.6	0.8/0.8	8.0	11.0/9.5	0.9/0.9	32.1	29.3/19.0	1.3/1.2
Face2Face$^{\rho}$	1.9	5.3/4.8	0.9/0.9	9.2	10.9/9.5	1.0/0.9	37.0	27.1/18.9	1.4/1.2
Method	1440×1440			1536×1536			2048×2048		
	MACs↓ $\times 10^9$	Inf.(ms)↓ FP32/FP16	Mem.(GB)↓ FP32/FP16	MACs↓ $\times 10^9$	Inf.(ms)↓ FP32/FP16	Mem.(GB)↓ FP32/FP16	MACs↓ $\times 10^9$	Inf.(ms)↓ FP32/FP16	Mem.(GB)↓ FP32/FP16
FS-VID2VID*	58.6	95.2/-	1.8/-	66.6	106.0/-	1.9/-	118.4	169.1/-	2.6/-
Bi-layer*	48.4	36.8/28.6	1.4/1.2	54.0	39.4/30.6	1.5/1.3	96.0	68.5/53.0	2.0/1.7
LPD*	71.0	50.9/33.6	1.8/1.6	80.9	55.0/36.4	2.0/1.8	143.8	94.7/63.6	2.7/2.4
FOMM*	73.4	58.2/-	1.8/-	83.6	62.3/-	2.0/-	148.6	106.8/-	2.7/-
MRAA*	74.2	64.4/-	1.8/-	84.5	70.8/-	2.0/-	150.2	120.0/-	2.7/-
HeadGAN*	63.5	57.3/35.7	1.8/1.5	72.3	60.6/37.6	1.9/1.8	128.5	101.3/62.1	2.6/2.4
Face2Face$^{\rho}$	71.8	52.1/32.5	1.8/1.6	82.8	56.7/34.8	2.0/1.8	146.2	95.8/60.2	2.7/2.4

Table 2. Quantitative comparisons with the modified baselines.

Method	Reenactment				Reconstruction	
	FID↓	CSIM↑	ARD↓	AU-H↓	LPIPS↓	AKD↓
FS-VID2VID*	100.9	0.505	4.32	0.33	0.191	3.66
Bi-layer*	125.8	0.442	3.95	0.32	0.192	2.76
LPD*	109.4	0.477	5.25	0.29	0.229	2.93
FOMM*	61.9	0.689	8.51	0.36	0.180	2.73
MRAA*	60.2	0.693	8.78	0.33	0.167	2.70
HeadGAN*	59.5	0.643	3.86	0.28	0.165	2.03
Face2Face$^{\rho}$	**44.5**	**0.729**	**2.82**	**0.22**	**0.123**	**1.48**

2 Network architectures

Rendering network. Tab. 4 presents the detailed architecture of our rendering network, which consists of an image encoder E_I, an image decoder D_I, and two pose encoders E_{ps} and E_{pd}. The image encoder E_I and decoder D_I are built with a series of downsampling, upsampling and residual blocks. The inverted residuals bottleneck introduced by MobileNetV2 [4] is used in implementing these blocks. E_{ps} and E_{pd} are identical in structure, i.e., two deconvolution blocks.

Motion network. Tab. 5 presents the detailed architecture of the mobile network, which consists of three sub-networks, i.e., F_1, F_2 and F_3. All sub-networks share a similar structure, i.e., downsampling, residual, upsampling, and convolution. Each sub-network takes a different scale of the source and landmark images as the input.

Fig. 2. Qualitative comparison with MarioNETte [1] under the one-shot setting.

Fig. 3. Qualitative comparison with AAN [7].

Table 3. Quantitative comparisons with the ablation study cases.

Method	Reenactment				Reconstruction	
	FID↓	CSIM↑	ARD↓	AU-H↓	LPIPS↓	AKD↓
w/o $\mathcal{L}_G^r$	170.0	0.650	3.22	0.26	0.179	1.86
w/o $\mathcal{L}_G^w$	311.1	0.001	2.83	0.38	0.410	2.15
w/o $\mathcal{L}_G^{adv}$	50.6	0.695	2.88	0.23	0.142	1.53
w/o $\mathcal{L}_G^{fm}$	45.1	0.710	2.84	0.22	0.128	1.48
Face2Face$^{\rho}$	**44.5**	**0.729**	**2.82**	**0.22**	**0.123**	**1.48**

3 More high-resolution results

Due to the lack of suitable high-resolution datasets, the 1440×1440 version of Face2Face$^{\rho}$ was trained using the upscaled images from VoxCeleb, i.e., 512×512 to 1440×1440. This inevitably limits the performance of Face2Face$^{\rho}$ on some true high-resolution images. However, as discussed in the paper, such practice is still of benefit. Here, we compare the faces generated by a 512×512 model and a 1440×1440 model in Fig. 7. Images generated by the latter model are clearer and sharper. More 1440×1440 results are shown in Fig. 9. Note that all source images in Fig. 9 are from the internet.

Fig. 4. Qualitative comparison with LSR [3] under the one-shot setting.

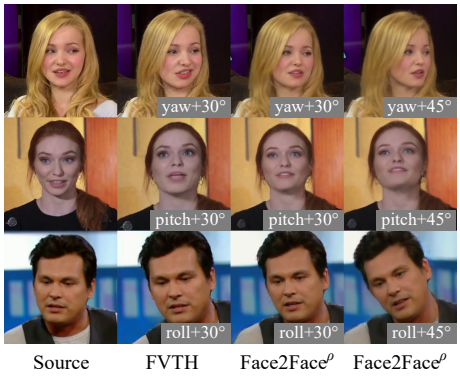

Fig. 5. Qualitative comparison with FVTH [6] on the task of head pose editing. The results are generated by its online demo which only supports modifying the Euler angle (i.e. yaw, pitch, roll) of the head pose up to 30°.

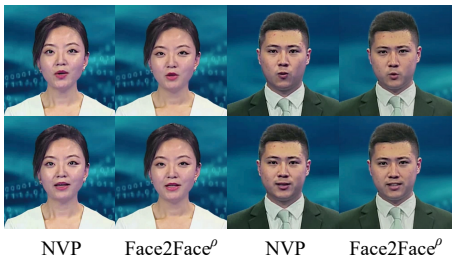

Fig. 6. Qualitative comparison with NVP [5] (using the authors' official implementation). The NVP generator for each actor needs to be trained separately on a 3 minutes video clip, while our method needs only one image and does not require any re-training.

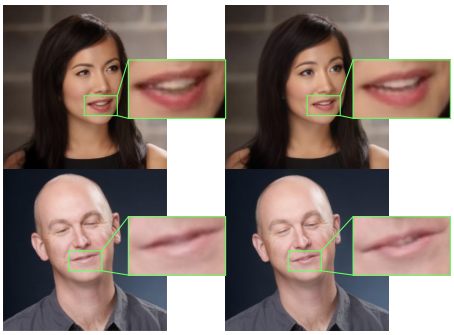

Fig. 7. Left: faked 1440 × 1440 results generated by upscaling the outputs of a 512 × 512 model. Right: results generated by 1440 × 1440 model.

Fig. 8. Qualitative comparisons with the ablation study cases.

Table 4. Detailed architecture of rendering network. In table, Conv., Down., Up., Res., SPADE, DeConv. and Tanh. are the abbreviations of convolution layer, downsampling block, upsampling block, residual block, spatially-adaptive normalization, deconvolution layer and element-wise tanh, respectively. BottleNeck(t,n,s,c) represents the bottleneck block of MobileNetV2 [4] with expansion factor t, layers n, strike s and output channels c.

$\boldsymbol{E_I}$	$\boldsymbol{D_I}$	$\boldsymbol{E_{ps}}$&$\boldsymbol{E_{pd}}$
3×3-Conv.-1-16, BN,ReLU	SPADE	4×4-DeConv.-1-64, BN,ReLU
Down. 4	Res. (2nd)	4×4-DeConv.-4-32
SPADE	SPADE	
Down. 2 (1st)	Up. 2 (1st)	
SPADE	SPADE	
Down. 2 (2nd)	Up. 2 (2nd)	
SPADE	SPADE	
Res. (1st)	Up. 4	
SPADE	3×3-Conv.-1-3,BN,Tanh.	
Building Blocks		
$k \times k$-Conv.-s-c	Convolution layer with kernel size k, stride s and output channels c	
Down. 4	BottleNeck(t=1,n=1,s=1,c=8), BottleNeck(t=6,n=2,s=2,c=12), BottleNeck(t=6,n=2,s=2,c=28)	
Down. 2 (1st)	BottleNeck(t=6,n=2,s=2,c=64), BottleNeck(t=6,n=2,s=1,c=72)	
Down. 2 (2nd)	BottleNeck(t=6,n=2,s=2,c=140)	
Res. (1st)	BottleNeck(t=6,n=1,s=1,c=280)	
Res. (2nd)	BottleNeck(t=6,n=1,s=1,c=140)	
Up. 2 (1st)	BottleNeck(t=6,n=2,s=1,c=96), $\times$2 Bilinear upsampling	
Up. 2 (2nd)	BottleNeck(t=6,n=2,s=1,c=64), BottleNeck(t=6,n=2,s=1,c=24), $\times$2 Bilinear upsampling	
Up. 4	BottleNeck(t=6,n=2,s=1,c=14), $\times$2 Bilinear upsampling, BottleNeck(t=6,n=2,s=1,c=8), $\times$2 Bilinear upsampling, BottleNeck(t=1,n=1,s=1,c=16)	
$k \times k$-DeConv.-s-c	Deconvolution layer with kernel size k, stride s and output channels c	

Table 5. Detailed architecture of motion network. In the table, Conv., Down., Up. and Res. are the abbreviations of convolution layer, downsampling block, upsampling block, and residual block, respectively.

<table>
<tr><th colspan="2">$\boldsymbol{F_1}$</th><th colspan="2">$\boldsymbol{F_2}$</th><th colspan="2">$\boldsymbol{F_3}$</th></tr>
<tr><td colspan="2">3×3-Conv.-1-64,BN,ReLU</td><td>Down. 2</td><td>Down.-64</td><td>Down. 2</td><td>Down.-32</td></tr>
<tr><td>Down. 4</td><td>Down.-128,
Down.-256</td><td>Down. 4</td><td>Down.-128,
Down.-256</td><td>Down. 4</td><td>Down.-64,
Down.-128</td></tr>
<tr><td colspan="2">Res.-256</td><td colspan="2">Res.-256</td><td colspan="2">Res.-128</td></tr>
<tr><td rowspan="2">Up. 4</td><td rowspan="2">Up.-128,
Up.-64</td><td colspan="2">Res.-256</td><td colspan="2">Res.-128</td></tr>
<tr><td colspan="2">Res.-256</td><td colspan="2">Res.-128</td></tr>
<tr><td colspan="2">3×3-Conv.-1-2</td><td rowspan="2">Up. 8</td><td rowspan="2">Up.-128,
Up.-64,
Up.-32,</td><td rowspan="2">Up. 8</td><td rowspan="2">Up.-64,
Up.-32,
Up.-16,</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2"></td><td colspan="2">3×3-Conv.-1-2</td><td colspan="2">3×3-Conv.-1-2</td></tr>
<tr><td colspan="6">Building Blocks</td></tr>
<tr><td>$k \times k$-Conv.-s-c</td><td colspan="5">Convolution layer with kernel size k,
stride s and output channels c</td></tr>
<tr><td>Down.-c</td><td colspan="5">4×4-Conv.-2-c,BN,ReLU</td></tr>
<tr><td>Up.-c</td><td colspan="5">$\times 2$ Bilinear upsampling,
3×3-Conv.-1-c,BN,ReLU</td></tr>
<tr><td>Res.-c</td><td colspan="5">Residual block [2] with kernel size 3×3
and output channel c</td></tr>
</table>

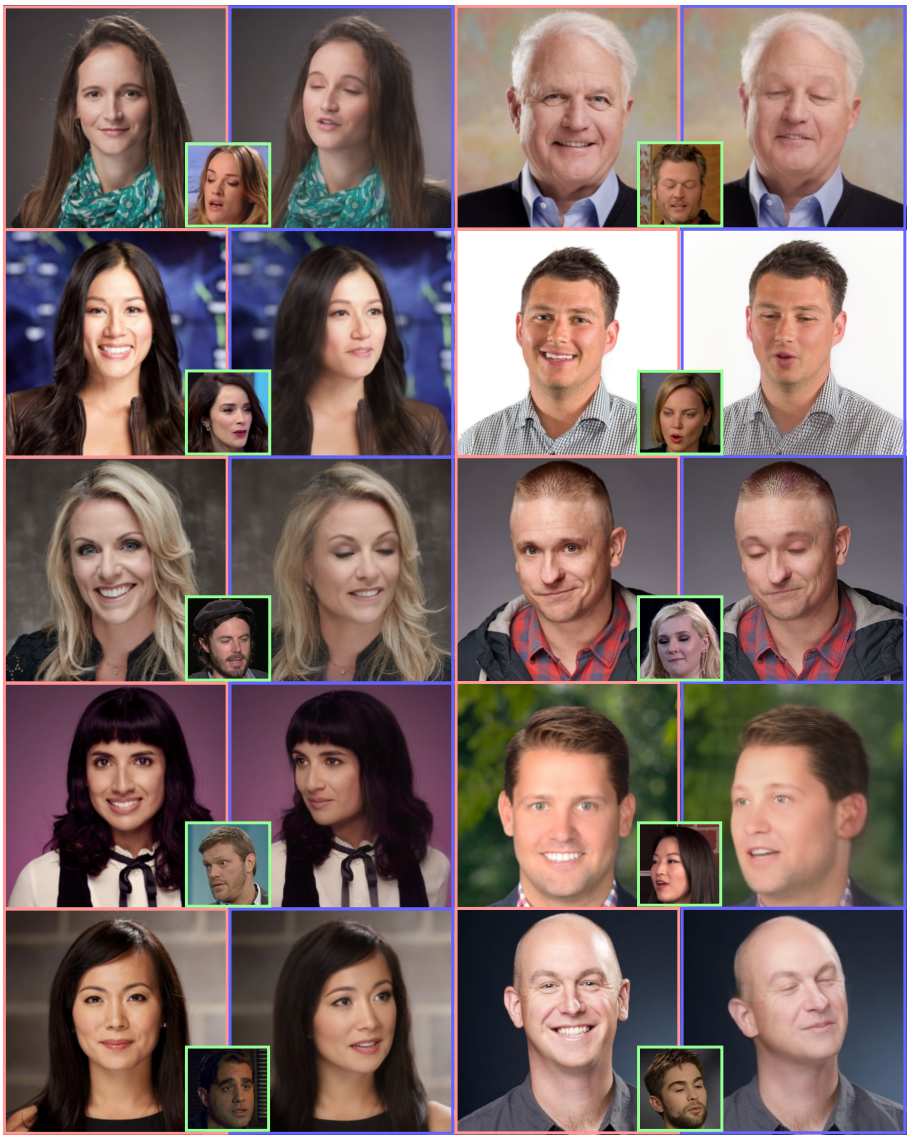

Fig. 9. Results of high-resolution images (1440×1440). Images with red boxes demonstrate the source images, images with green boxes demonstrate the driving images, and images with blue boxes demonstrate the reenacted face images.

References

1. Ha, S., Kersner, M., Kim, B., Seo, S., Kim, D.: MarioNETte: Few-shot face reenactment preserving identity of unseen targets. In: AAAI. pp. 10893–10900 (2020)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
3. Meshry, M., Suri, S., Davis, L.S., Shrivastava, A.: Learned spatial representations for few-shot talking-head synthesis. In: ICCV. pp. 13829–13838 (2021)
4. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: CVPR. pp. 4510–4520 (2018)
5. Thies, J., Elgharib, M., Tewari, A., Theobalt, C., Nießner, M.: Neural Voice Puppetry: Audio-driven facial reenactment. In: ECCV. pp. 716–731 (2020)
6. Wang, T.C., Mallya, A., Liu, M.Y.: One-shot free-view neural talking-head synthesis for video conferencing. In: CVPR. pp. 10039–10049 (2021)
7. Yao, G., Yuan, Y., Shao, T., Li, S., Liu, S., Liu, Y., Wang, M., Zhou, K.: One-shot face reenactment using appearance adaptive normalization. In: AAAI. pp. 3172–3180 (2021)