HyperReenact: One-Shot Reenactment via Jointly Learning to Refine and Retarget Faces
– Supplementary Material –

Stella Bounareli[1], Christos Tzelepis[2], Vasileios Argyriou[1], Ioannis Patras[2], Georgios Tzimiropoulos[2]
[1] School of Computer Science and Mathematics, Kingston University London
[2] School of Electronic Engineering and Computer Science, Queen Mary University of London

In this supplementary material, we first provide a detailed analysis of the network architecture of the proposed framework (HyperReenact) in Section 1. In Section 2, we show results of two state-of-the-art inversion methods, namely, HyperStyle [1] and HyperInverter [6], on real image editing and compare our method against HyperStyle [1]. In Section 3, we discuss the limitations of our method and in Section 4 we present comparisons against two methods, namely, StyleHEAT [22] and StyleMask [3], while in Section 6 we provide additional results of our method on the VoxCeleb1 [13] and the VoxCeleb2 [5] datasets. Finally, along with this report, please find attached an external video file that includes 10 randomly selected identities for self reenactment and 10 randomly selected pairs for cross-subject reenactment (for both VoxCeleb1 [13] and VoxCeleb2 [5] datasets).

1. Network architecture

In this section we provide details of the various components of the proposed framework (HyperReenact). An overview of HyperReenact is shown in Fig. 2 in the main paper. More specifically, we propose to blend the appearance feature map f_{app} of size $512 \times 7 \times 7$ and the pose feature map f_{p} of size $2048 \times 7 \times 7$ using the Reenactment Module (RM), which is illustrated in Fig. 1. As shown, we first project the f_{p} using a convolutional layer (kernel=(1,1), stride=1, pad=0) into the same channel size of f_{app}. Then for each feature map we learn the two modulation parameters γ and β using convolutions with a kernel size of 1, stride set to 1 and padding set to 0. The output feature map f_{r} of size $512 \times 7 \times 7$, computed using Eq. 1 from the main paper, is then fed into the different reenactment blocks of the hypernetwork.

Our hypernetwork $\mathcal{H}$ consists of $\text{M} < \text{N}$ Reenactment Blocks (RB), where M is the number of generator layers that we control and N is the total number of layers in the generator. Each of those blocks takes as input the feature map f_{r} and outputs the corresponding offset $\Delta\theta_\ell$ for the weights of each layer of the generator. In Table 1 we report the structure of StyleGAN2 generator trained on 256×256 image resolution. From the blocks shown in Table 1 we only modify the convolutional layers (Conv), as the ToRGB layers mainly affect the colors of the generated images [21]. Hence, we propose to modify $\text{M} = 13$ layers.

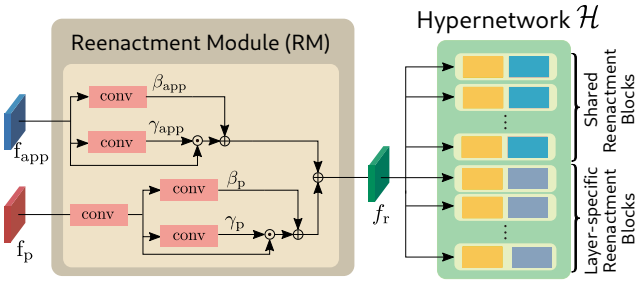


Figure 1: Illustration of the two learnable components of our architecture, namely, the Reenactment Module (RM) and the hypernetwork $\mathcal{H}$. The RM module fuses appearance features from the source face and pose features from the target face and outputs the fused feature map f_{r}, that drives the hypernetwork $\mathcal{H}$. The hypernetwork $\mathcal{H}$ consists of multiple Reenactment Blocks with each one of them corresponding to a particular layer of the generator.

In order to reduce the number of trainable parameters, similarly to HyperStyle [1], we propose to use two types of Reenactment Blocks, namely, the Shared Reenactment Blocks and the Layer-specific Reenactment Blocks, as shown in Fig. 1. We present an overview of the architecture of both blocks in Fig. 2. In both types of blocks, the input feature map f_{r} is first fed into a series of convolutional layers that process and down-sample the input into the shape of $512 \times 1 \times 1$. Regarding the Shared Reenactment Blocks, as shown on the top row of Fig. 2, the down-sampled feature map is then flattened and fed into

a fully-connected layer. Afterwards, the two shared fully-connected layers are used to calculate the output feature map of shape $C_\ell^{out} \times C_\ell^{in} \times 1 \times 1$, which is repeated spatially so as to match the shape of the convolutional kernels ($C_\ell^{out} \times C_\ell^{in} \times k_\ell \times k_\ell$). Regarding the Layer-specific Reenactment Blocks, as shown on the bottom row of Fig. 2, after the series of down-sampling convolutional layers, the computed feature map has a shape of $512 \times 1 \times 1$. This feature map, upon being flattened, is fed into a final fully-connected layer which outputs a feature map of shape $C_\ell^{out} \times C_\ell^{in} \times 1 \times 1$, which is then spatially repeated to have a shape of $C_\ell^{out} \times C_\ell^{in} \times k_\ell \times k_\ell$.

We provide in more detail the structure of the two Reenactment Blocks in Tables 2 and 3. Finally, in Table 4, we report the StyleGAN2 layers that we propose to modify as well as the type of the Reenactment Block that we use for each layer. As shown in Table 4, we use the Shared Reenactment Blocks for the first seven layers of the generator. As a result, the parameters of two fully-connected layers across all the Shared Reenactment Blocks, are common. For the last six layers of the generator, we use the Layer-specific Reenactment Blocks. We note that the use of the shared layers reduces our trainable parameters from 1.2B to 300M.

2. Comparisons with HyperStyle

As discussed in the main paper, the proposed framework draws inspiration from two state-of-the-art methods (for the task of real image inversion), namely, HyperStyle [1] and HyperInverter [6]. Specifically, similarly to [1, 6], we also incorporate a hypernetwork [7] in order to learn how to effectively modify the weights of a pretrained StyleGAN2 [10] generator. However, we note that [1, 6] aim at the problem of real image inversion, not neural face reenactment.

Both HyperStyle [1] and HyperInverter [6] produce high-quality results on real image inversion, however their quality degenerates significantly when manipulating the inverted images, especially on head pose editing. In Fig. 3, we show results of HyperStyle [1] (Fig. 3a) and HyperInverter [6] (Fig. 3b) on head pose editing using the the CelebA dataset [9]. It is worth noting that, while both methods excel on real image inversion (the inverted images are inside the red boxes), they produce several visual artifacts on the edited images, which renders them unsuitable for the task of real image reenactment. We note that we obtain the edited images using the InterFaceGAN method [17] to shift the latent codes.

To further compare our method with HyperStyle [1], instead of simply editing the head pose as shown above, we perform face reenactment using the learned directions from the work of FD [2]. We note that FD learns the directions in the W^+ latent space of a StyleGAN2 model trained on the VoxCeleb1 [13] dataset that are responsible for controlling the facial pose. In order to test HyperStyle along with FD, we train HyperStyle on the VoxCeleb1 dataset. We refer to this pipeline as HyperStyle-FD. In Fig. 4 we show comparisons of our method against HyperStyle-FD. We note that the reenacted images using HyperStyle-FD not only present visual artifacts, but also look unnatural, especially when the source and target images have large head pose differences.

3. Limitations

As shown in the main paper and in the additional experimental results provided in this supplementary material, the proposed HyperReenact, in contrast to several state-of-the-art works, achieves to effectively reenact a source face given a target facial pose, preserving the source identity characteristics and producing artifact-free images, especially in the cases where the target and the source faces differ largely in head pose. Nevertheless, we observe that in cases where the source facial images depict accessories such hats or eyeglasses, the proposed method fails to reconstruct them in detail. For instance, as shown in Fig. 5, our method cannot fully reconstruct the style of the glasses in the example of the first row. Similarly, regarding the examples of the second and third row of Fig. 5, our method is not able to reconstruct every detail on the hats. We attribute this to the fact that such items are underrepresented on the VoxCeleb1 dataset and, as a result, our method is not able to learn how to reconstruct them. Additionally, we do not refine any details on the background of the generated images.

4. Comparisons with StyleHEAT [22] and StyleMask [3]

As discussed in Section 4, StyleHEAT [22] is trained on the HDTF dataset [24], that consists of facial images exhibiting only small roll angle variations and showing mostly frontal views. Moreover, StyleMask [3] is a face reenactment method based on a pretrained StyleGAN2 model trained on the FFHQ dataset, which learns to disentangle the identity characteristics from the facial pose using the disentangled properties of the style space $\mathcal{S}$ of StyleGAN2. Both StyleHEAT and StyleMask require the input images to be aligned, similarly to the FFHQ dataset [9]. In Fig. 6 and in Table 5, we present qualitative and quantitative comparisons with StyleHEAT and StyleMask on the VoxCeleb1 dataset [13]. Clearly, StyleHEAT performs poorly by generating many visual artifacts when the source and target images have large pose variations, while StyleMask is not able to faithfully reconstruct the source identity characteristics. For a fair comparison, we additionally compare on the HDTF dataset [24] (where StyleHEAT has been trained

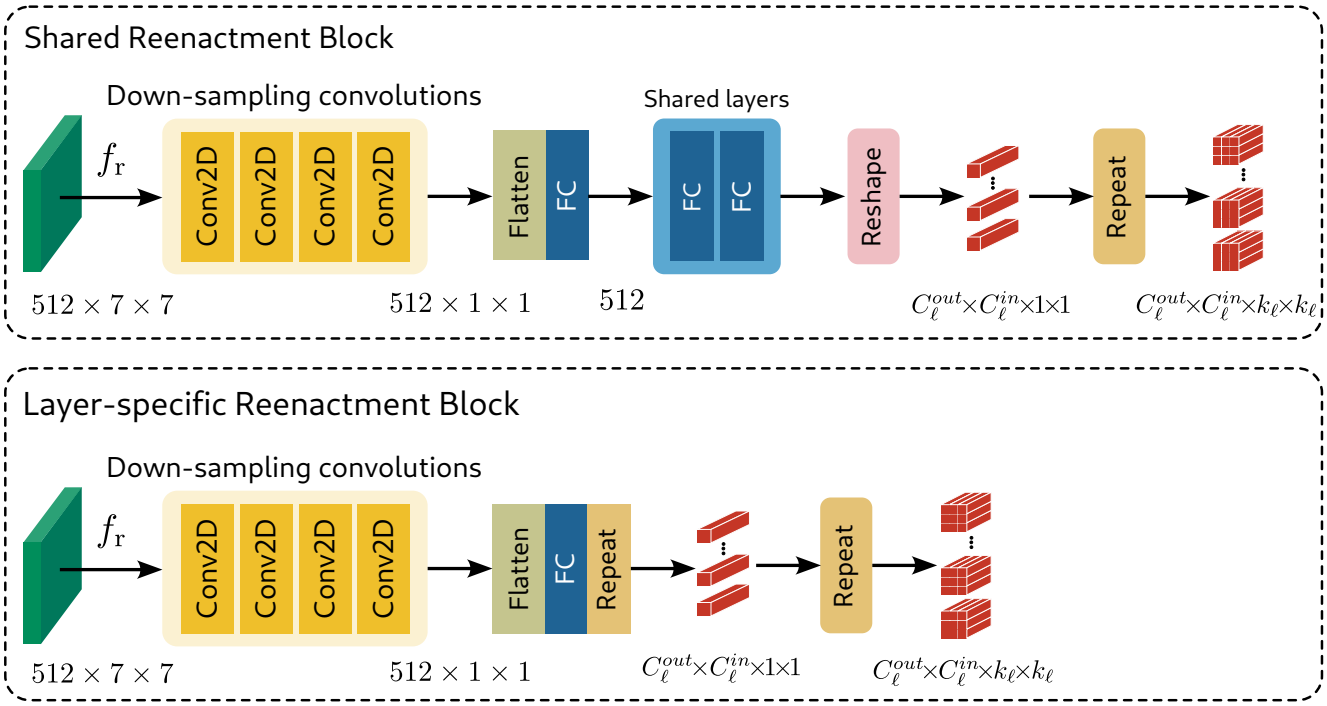


Figure 2: Detailed architecture of the two types of Reenactment Blocks, namely the Shared Reenactment Block (top row) and the Layer-specific Reenactment Block (bottom row).

Layer ℓ Index	Layer ℓ Name	Resolution	Kernel Dim. $C_\ell^{out} \times C_\ell^{in} \times k_\ell \times k_\ell$
0	Conv1	4×4	$512 \times 512 \times 3 \times 3$
1	ToRGB1	4×4	$3 \times 512 \times 1 \times 1$
2	Conv2	8×8	$512 \times 512 \times 3 \times 3$
3	Conv3	8×8	$512 \times 512 \times 3 \times 3$
4	ToRGB2	8×8	$3 \times 512 \times 1 \times 1$
5	Conv4	16×16	$512 \times 512 \times 3 \times 3$
6	Conv5	16×16	$512 \times 512 \times 3 \times 3$
7	ToRGB3	16×16	$3 \times 512 \times 1 \times 1$
8	Conv6	32×32	$512 \times 512 \times 3 \times 3$
9	Conv7	32×32	$512 \times 512 \times 3 \times 3$
10	ToRGB4	32×32	$3 \times 512 \times 1 \times 1$
11	Conv8	64×64	$256 \times 512 \times 3 \times 3$
12	Conv9	64×64	$256 \times 256 \times 3 \times 3$
13	ToRGB5	64×64	$3 \times 256 \times 1 \times 1$
14	Conv10	128×128	$128 \times 256 \times 3 \times 3$
15	Conv11	128×128	$128 \times 128 \times 3 \times 3$
16	ToRGB6	128×128	$3 \times 128 \times 1 \times 1$
17	Conv12	256×256	$64 \times 128 \times 3 \times 3$
18	Conv13	256×256	$64 \times 64 \times 3 \times 3$
19	ToRGB7	256×256	$3 \times 64 \times 1 \times 1$

Table 1: Structure of StyleGAN2 generator trained on 256×256 image resolution.

Layer	Input shape	Output shape
Conv2D kernel=(3,3), stride=1, pad=1	$B \times 512 \times 7 \times 7$	$B \times 128 \times 7 \times 7$
LeakyReLU activation slope=0.01	$B \times 128 \times 7 \times 7$	$B \times 128 \times 7 \times 7$
Conv2D kernel=(3,3), stride=1, pad=0	$B \times 128 \times 7 \times 7$	$B \times 128 \times 5 \times 5$
LeakyReLU activation slope=0.01	$B \times 128 \times 5 \times 5$	$B \times 128 \times 5 \times 5$
Conv2D kernel=(3,3), stride=1, pad=0	$B \times 128 \times 5 \times 5$	$B \times 128 \times 3 \times 3$
LeakyReLU activation slope=0.01	$B \times 128 \times 3 \times 3$	$B \times 128 \times 3 \times 3$
Conv2D kernel=(3,3), stride=1, pad=0	$B \times 128 \times 3 \times 3$	$B \times 512 \times 1 \times 1$
LeakyReLU activation slope=0.01	$B \times 512 \times 1 \times 1$	$B \times 512 \times 1 \times 1$
Flatten	$B \times 512 \times 1 \times 1$	$B \times 512$
FC	$B \times 512$	$B \times 512$
Shared FC	$B \times 512$	$B \times (512 \times 512)$
Shared FC	$B \times (512 \times 512)$	$B \times (512 \times 512)$
Reshape	$B \times (512 \times 512)$	$B \times 512 \times 512 \times 1 \times 1$
Repeat	$B \times 512 \times 512 \times 1 \times 1$	$B \times 512 \times 512 \times 3 \times 3$

Table 2: Architecture of the Shared Reenactment Blocks (B denotes the batch size).

Layer	Input shape	Output shape
Conv2D kernel=(3,3), stride=1, pad=1	$B \times 512 \times 7 \times 7$	$B \times 256 \times 7 \times 7$
LeakyReLU activation slope=0.01	$B \times 256 \times 7 \times 7$	$B \times 256 \times 7 \times 7$
Conv2D kernel=(3,3), stride=1, pad=0	$B \times 256 \times 7 \times 7$	$B \times 256 \times 5 \times 5$
LeakyReLU activation slope=0.01	$B \times 256 \times 5 \times 5$	$B \times 256 \times 5 \times 5$
Conv2D kernel=(3,3), stride=1, pad=0	$B \times 256 \times 5 \times 5$	$B \times 256 \times 3 \times 3$
LeakyReLU activation slope=0.01	$B \times 256 \times 3 \times 3$	$B \times 256 \times 3 \times 3$
Conv2D kernel=(3,3), stride=1, pad=0	$B \times 256 \times 3 \times 3$	$B \times 512 \times 1 \times 1$
LeakyReLU activation slope=0.01	$B \times 512 \times 1 \times 1$	$B \times 512 \times 1 \times 1$
Flatten	$B \times 512 \times 1 \times 1$	$B \times 512$
FC	$B \times 512$	$B \times (C_\ell^{out} \times C_\ell^{in})$
Reshape	$B \times (C_\ell^{out} \times C_\ell^{in})$	$B \times C_\ell^{out} \times C_\ell^{in} \times 1 \times 1$
Repeat	$B \times C_\ell^{out} \times C_\ell^{in} \times 1 \times 1$	$B \times C_\ell^{out} \times C_\ell^{in} \times 3 \times 3$

Table 3: Architecture of the Layer-specific Reenactment Blocks. The input on the block has a size of $B \times 512 \times 7 \times 7$, where B is the batch size, while C_ℓ^{in} and C_ℓ^{out} are the input and output channels, respectively.

Layer Index ℓ	ℓ-th Layer Name	RB Type
0	Conv1	Shared
2	Conv2	Shared
3	Conv3	Shared
5	Conv4	Shared
6	Conv5	Shared
8	Conv6	Shared
9	Conv7	Shared
11	Conv8	Layer-specific
12	Conv9	Layer-specific
14	Conv10	Layer-specific
15	Conv11	Layer-specific
17	Conv12	Layer-specific
18	Conv13	Layer-specific

Table 4: Convolutional layers of the StyleGAN2 generator that we propose to modify, altering their weights using the offsets computed by the hypernetwork.

Method	CSIM↑	APD↓	AED↓
StyleHEAT [22]	0.45	8.6	12.9
StyleMask [3]	0.47	5.3	13.2
Ours	**0.58**	**0.9**	**6.2**

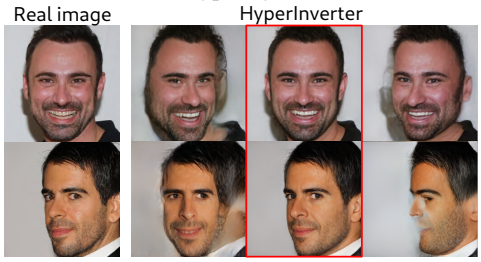
Table 5: Quantitative comparisons with StyleHEAT [22] and StyleMask [3] on the small benchmark with large head pose differences between the source and target faces.

Method	CSIM↑	APD↓	AED↓
StyleHEAT [22]	0.72	1.1	7.5
StyleMask [3]	0.66	1.6	8.8
Ours	**0.75**	**0.38**	**4.1**

Table 6: Quantitative comparisons on self-reenactment with StyleHEAT [22] and StyleMask [3] on HDTF dataset [24]

on). In Table 6, we provide quantitative results on the test videos provided by the authors of StyleHEAT [22]. Finally, Fig. 7 illustrates qualitative comparisons with both StyleHEAT and StyleMask. As shown, our method evidently outperforms both StyleHEAT and StyleMask, on identity preservation and on facial pose transfer.

(a) HyperStyle [1].



(b) HyperInverter [6].

Figure 3: Real image inversion and editing results on CelebA dataset [9] using HyperStyle [1] and HyperInverter [6]. Inside the red boxes are the inverted images, while on the right and left we show results of head pose editing using the method of InterFaceGAN [17].

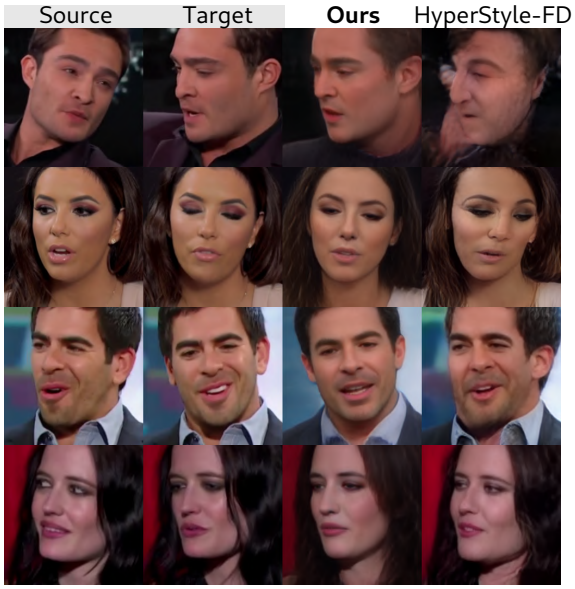


Figure 4: Qualitative comparisons against HyperStyle-FD.

5. Benchmark with extreme head pose differences

We build a small benchmark in order to evaluate our method on challenging cases where the source and target faces have large head pose differences. Specifically, consid-

Figure 5: We observe that accessory items such as hats or glasses are not fully reconstructed.

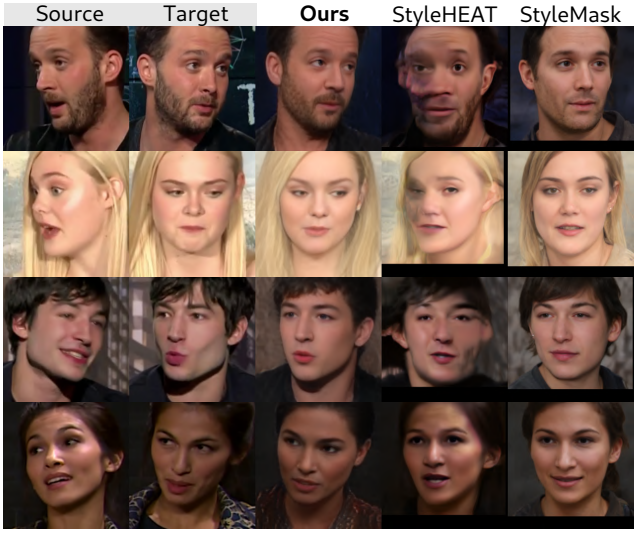


Figure 6: Qualitative comparisons with StyleHEAT [22] and StyleMask [3] on VoxCeleb dataset [13].

ering the VoxCeleb1 [13] test dataset, we pick 1000 pairs of images with large head pose distance (measured as the average of the absolute differences between the 3 Euler angles). In Fig. 8 we show the distribution of the absolute pose differences for each Euler angle, namely yaw, pitch and roll. This benchmark allows us to obtain deeper insights on the behavior of reenactment methods on challenging conditions.

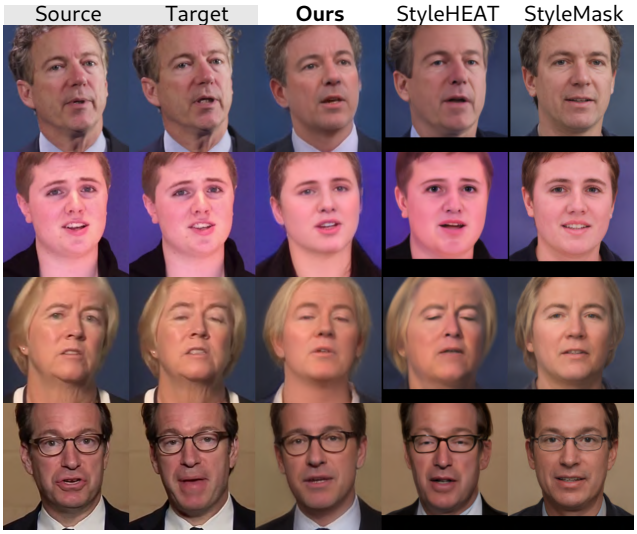


Figure 7: Qualitative comparisons with StyleHEAT [22] and StyleMask [3] on HDTF dataset [24].

In Fig. 9, we show comparisons of our method against the two state-of-the-art face reenactment methods, namely Fast BL [23] and Rome [11], on image pairs selected from the small benchmark described above. As shown, the source and target images have extreme head pose differences which makes it more challenging for face reenactment methods to generate realistic images. Nevertheless, our method is able to synthesize realistic faces even on extreme head poses. In Fig. 9 we highlight (red boxes) details on the human faces, such as areas around the mouth and eyes, where our method creates artifact free results, while Fast BL [23] and Rome [11] generate blurry unrealistic images.

6. Additional quantitative & qualitative results

In this section, we present additional quantitative and qualitative results of the proposed method in comparison to state-of-the-art works. We compare all methods in terms of their inference time and their overall performance on the two tasks, namely, self and cross-subject reenactment. Specifically, in Table 7, we demonstrate the inference time of each method while reenacting a video of 200 frames. To help drawing connections between the inference time and the performance of each method, we also report the performance ranking in Table 7 (referred to as "Perf. Rank"). Specifically, we consider the evaluation metrics that are reported in Table 1 of the main paper, and rank all methods with respect to each metric. Then, we average the ranking positions across all metrics on both self and cross-subject reenactment, for each method, to obtain its overall performance ranking. In Fig. 10, we present a plot of the two

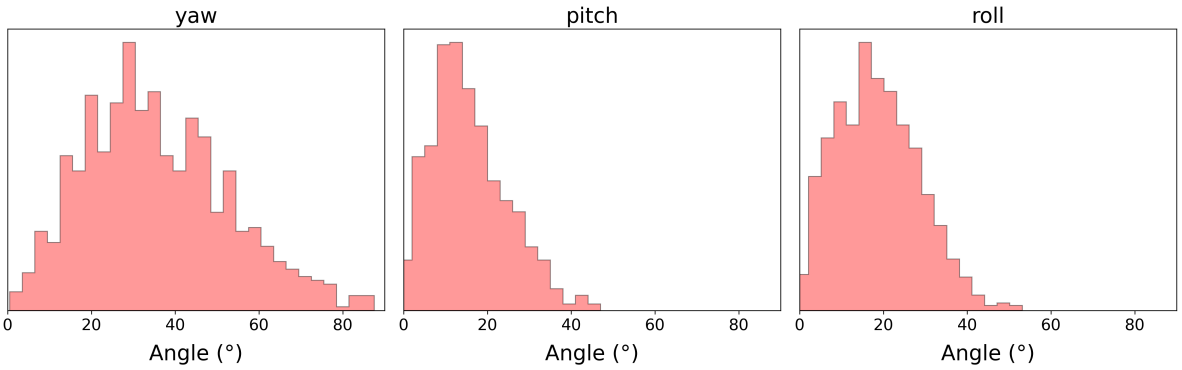


Figure 8: Distribution of the absolute pose differences for each Euler angle (yaw, pitch and roll) in our small benchmark dataset built from VoxCeleb1 test dataset.

Figure 9: Qualitative results on the small benchmark with large head pose differences between the source and target faces.

Method	Inf. time (sec)↓	Perf. Rank ↓
X2Face [20]	**11.0**	3.5
FOMM [18]	**11.0**	6.3
Neural [4]	115.0	5.6
Fast BL [23]	61.0	4.0
PIR [14]	54.0	4.3
LSR [12]	110.0	4.2
FD [2]	40.0	2.7
LIA [19]	23.0	5.3
Dual [8]	23.0	6.9
Rome [11]	70.0	3.1
Ours	37.0	**1.1**

Table 7: Quantitative comparisons on inference time and overall performance ranking (Perf. Rank) of all methods on self and cross-subject reenactment tasks (inference time is calculated while reenacting a video of 200 frames).

metrics (Inference time and Overall Performance Ranking). We note that our method achieves the best overall performance ranking, while also remaining competitive on the inference time. Additionally, while the inference time of X2Face [20] and FOMM [18] is low compared to the other methods, as shown from the qualitative results they generate images with several visual artifacts.

Moreover, we present further qualitative comparisons on the VoxCeleb1 [13], as well as quantitative and qualitative results on VoxCeleb2 [5]. Specifically, in Figs. 11 and 12 we show results on self reenactment, in Figs. 13 and 14 we show results on cross-subject reenactment, and in Fig. 15 we report results on self reenactment on the benchmark with extreme head pose differences described in Section 5.

Additionally, we quantitatively compare our method on the task of self reenactment on the VoxCeleb2 dataset [5] with the 10 state-of-the-art methods, namely, X2Face [20], FOMM [18], Neural [4], Fast BL [23], PIR [14], LSR [12], FD [2], LIA [19], Dual [8], and Rome [11]. In Table 8, we present the quantitative results on self reenactment. As shown, our method outperforms all other methods both on identity preservation (CSIM) and on head pose (APD) and expression (AED) transfer (similarly to Section 4.1).

Finally, in Figs. 16, 17, we demonstrate results of our method, both on self and on cross-subject reenactment, on additional video datasets, namely, FaceForensics [15], 300-VW [16], and CelebV-HQ [25], showing that the proposed method can generalise well on different video benchmarks.

Method	CSIM↑	APD↓	AED↓
X2Face [20]	0.60	2.4	10.6
FOMM [18]	0.57	5.1	13.6
Neural [4]	0.39	1.4	9.1
Fast BL [23]	0.57	1.1	8.6
PIR [14]	0.57	2.8	10.8
LSR [12]	0.61	1.0	7.5
FD [2]	0.59	1.3	7.3
LIA [19]	0.64	2.5	8.7
Dual [8]	0.15	3.7	12.7
Rome [11]	0.63	1.3	5.9
Ours	**0.65**	**0.5**	**5.2**

Table 8: Quantitative results on the task of self reenactment on VoxCeleb2 dataset [5].

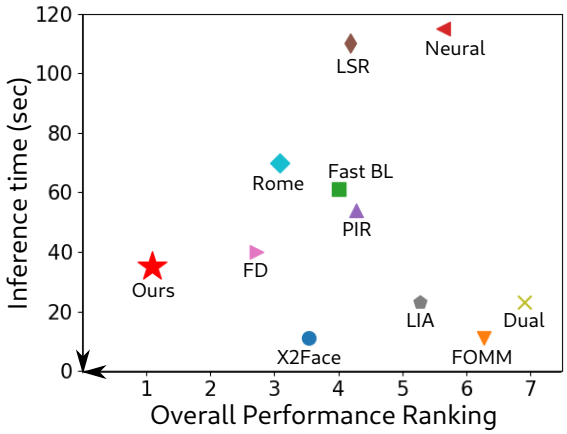


Figure 10: Comparisons in terms of the overall performance ranking of each method (presented in Table 1 of the main paper) and their inference time required to reenact a video of 200 frames. The arrows point towards the best results.

7. Ethics consideration

Neural face reenactment methods allow for the creation of realistic talking head sequences that resemble the real ones. Consequently, besides being used for benevolent and useful purposes, such as in video conferencing, film and video production, arts, and education, we acknowledge that face reenactment methods, such as the proposed one, can also be misused towards malevolent purposes, such as deepfake fraud, that can harm individuals and can pose a greater societal threat.

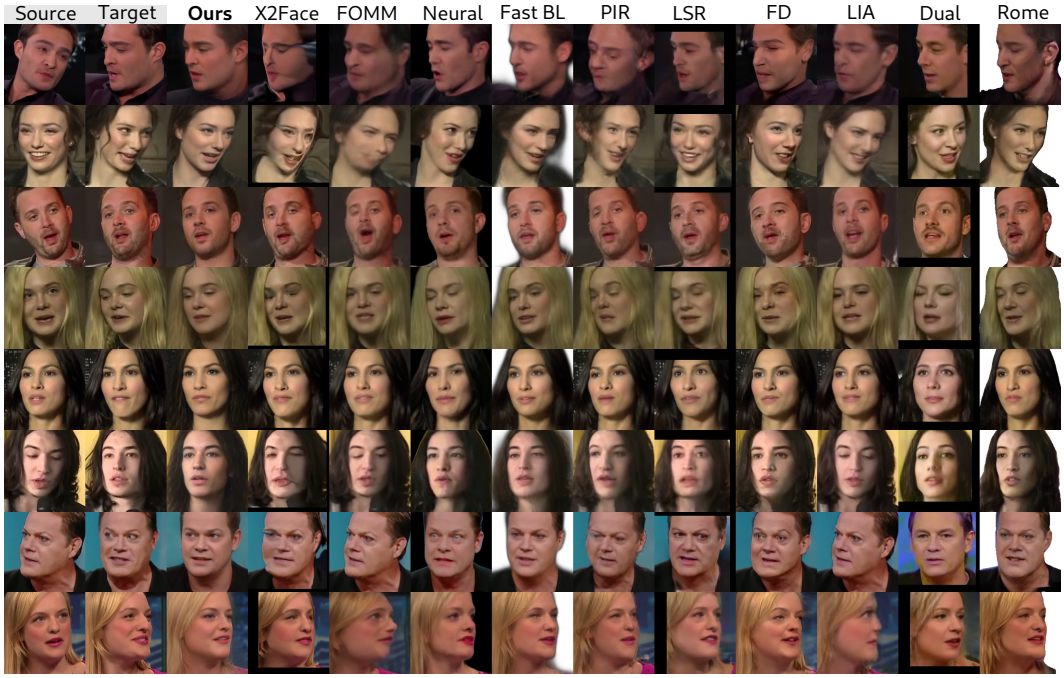


Figure 11: Additional qualitative results and comparisons on self-reenactment on VoxCeleb1 dataset [13]. The first and second columns show the source and target faces.

Figure 12: Qualitative results and comparisons on self-reenactment on VoxCeleb2 dataset [5]. The first and second columns show the source and target faces.

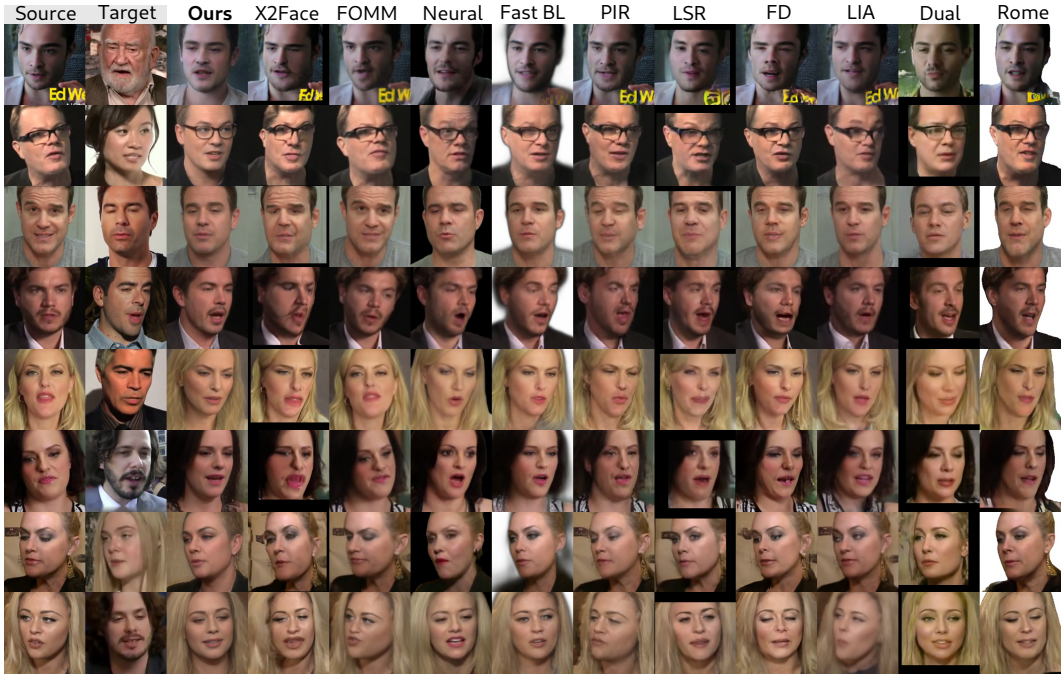


Figure 13: Additional qualitative results and comparisons on cross-subject reenactment on VoxCeleb1 dataset [13]. The first and second columns show the source and target faces.

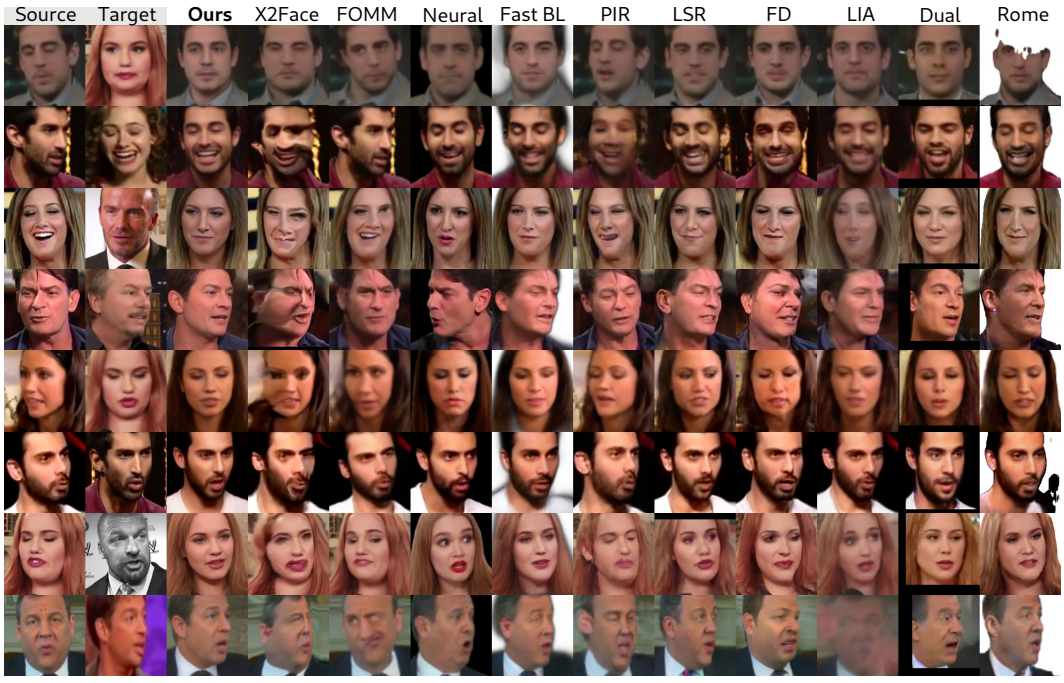


Figure 14: Qualitative results and comparisons on cross-subject reenactment on VoxCeleb2 dataset [5]. The first and second columns show the source and target faces, which are from different identities.

Figure 15: Additional qualitative results and comparisons on self reenactment using our small benchmark with image pairs that have large head pose differences. We show that our method presents robust results with artifact-free images, compared to the other state-of-the-art methods.

Figure 16: Additional qualitative results of our method on other video datasets such as FaceForensics [15] and 300-VW [16].

Figure 17: Additional qualitative results of our method on CelebV-HQ video dataset [25].

References

[1] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18511–18521, 2022. 1, 2, 5

[2] Stella Bounareli, Vasileios Argyriou, and Georgios Tzimiropoulos. Finding directions in gan's latent space for neural face reenactment. *British Machine Vision Conference (BMVC)*, 2022. 2, 7, 8

[3] Stella Bounareli, Christos Tzelepis, Vasileios Argyriou, Ioannis Patras, and Georgios Tzimiropoulos. Stylemask: Disentangling the style space of stylegan2 for neural face reenactment. *IEEE Conference on Automatic Face and Gesture Recognition*, 2023. 1, 2, 5, 6

[4] Egor Burkov, Igor Pasechnik, Artur Grigorev, and Victor Lempitsky. Neural head reenactment with latent pose descriptors. In *CVPR*, 2020. 7, 8

[5] J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. In *INTERSPEECH*, 2018. 1, 7, 8, 9, 10

[6] Tan M Dinh, Anh Tuan Tran, Rang Nguyen, and Binh-Son Hua. Hyperinverter: Improving stylegan inversion via hypernetwork. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11389–11398, 2022. 1, 2, 5

[7] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. 2

[8] Gee-Sern Hsu, Chun-Hung Tsai, and Hung-Yi Wu. Dual-generator face reenactment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 642–650, 2022. 7, 8

[9] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 2, 5

[10] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 2

[11] Taras Khakhulin, Vanessa Sklyarova, Victor Lempitsky, and Egor Zakharov. Realistic one-shot mesh-based head avatars. In *European Conference on Computer Vision*, pages 345–362. Springer, 2022. 6, 7, 8

[12] Moustafa Meshry, Saksham Suri, Larry S Davis, and Abhinav Shrivastava. Learned spatial representations for few-shot talking-head synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13829–13838, 2021. 7, 8

[13] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017. 1, 2, 6, 7, 9, 10

[14] Yurui Ren, Ge Li, Yuanqi Chen, Thomas H Li, and Shan Liu. Pirenderer: Controllable portrait image generation via semantic neural rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13759–13768, 2021. 7, 8

[15] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics: A large-scale video dataset for forgery detection in human faces. *arXiv*, 2018. 7, 12

[16] Jie Shen, Stefanos Zafeiriou, Grigoris G Chrysos, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. The first facial landmark tracking in-the-wild challenge: Benchmark and results. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 50–58, 2015. 7, 12

[17] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2, 5

[18] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in Neural Information Processing Systems*, 32:7137–7147, 2019. 7, 8

[19] Yaohui Wang, Di Yang, Francois Bremond, and Antitza Dantcheva. Latent image animator: Learning to animate images via latent space navigation. In *International Conference on Learning Representations*, 2021. 7, 8

[20] Olivia Wiles, A Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio, and pose codes. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–686, 2018. 7, 8

[21] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1

[22] Fei Yin, Yong Zhang, Xiaodong Cun, Mingdeng Cao, Yanbo Fan, Xuan Wang, Qingyan Bai, Baoyuan Wu, Jue Wang, and Yujiu Yang. Styleheat: One-shot high-resolution editable talking face generation via pretrained stylegan. *arXiv preprint arXiv:2203.04036*, 2022. 1, 2, 5, 6

[23] Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *ECCV*, 2020. 6, 7, 8

[24] Zhimeng Zhang, Lincheng Li, Yu Ding, and Changjie Fan. Flow-guided one-shot talking face generation with a high-resolution audio-visual dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3661–3670, 2021. 2, 5, 6

[25] Hao Zhu, Wayne Wu, Wentao Zhu, Liming Jiang, Siwei Tang, Li Zhang, Ziwei Liu, and Chen Change Loy. CelebV-HQ: A large-scale video facial attributes dataset. In *ECCV*, 2022. 7, 13