HeadGAN: One-shot Neural Head Synthesis and Editing (supplementary material)

Michail Christos Doukas[1,2], Stefanos Zafeiriou[1,2], Viktoriia Sharmanska[1,3]
[1]Imperial College London, UK [2]Huawei Technologies, London, UK [3] University of Sussex, UK
{michail-christos.doukas16, s.zafeiriou, sharmanska.v}@imperial.ac.uk

1. 3DMM fitting

Given a facial image $\mathbf{y}$, our 3DMM fitting stage recovers shape $\mathbf{p}$ and camera $\mathbf{c}$ parameters. It relies on dense 3D points of the face, which are regressed with RetinaFace-R501[5] network, pre-trained on WIDER FACE dataset [17]. We use Procrustes analysis to register the regressed points with the mean shape $\bar{\mathbf{x}}$ of LSFM 3DMM [3].

Algorithm 1: Fit the 3DMM to a given image.

Input: 3DMM: $\{\mathbf{U}^{id}, \mathbf{U}^{exp}, \bar{\mathbf{x}}\}$, image: $\mathbf{y}$
`/* Regress dense 3D points */`
$\mathbf{l}$ = RetinaFace($\mathbf{y}$)
`/* Align points to mean shape */`
$\mathbf{l}'$ = Procrustes($\bar{\mathbf{x}}, \mathbf{l}$)
`/* Merge id. with exp. 3DMM */`
$\mathbf{U} = [\mathbf{U}^{id\top}; \mathbf{U}^{exp\top}]^\top$
`/* Compute Moore-Penrose inverse */`
$\mathbf{U}^+ = (\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top$
`/* Recover shape parameters */`
$\mathbf{p} = \mathbf{U}^+(\mathbf{l}' - \bar{\mathbf{x}})$
`/* Compute affine camera matrix */`
$\mathbf{P}$ = Least_squares ($\mathbf{l}_{homog}, \bar{\mathbf{x}}$) $\in \mathbb{R}^{3\times4}$
`/* Recover camera parameters:`
`scale, rotation, translation */`
$\mathbf{c}$ = P_to_srt($\mathbf{P}$)
Result: shape parameters $\mathbf{p}$, camera parameters $\mathbf{c}$

2. Objective functions - Training

We train HeadGAN framework, consisting of the Generator G and the two Discriminators D and D_m, using GAN Hinge loss [8]. Therefore, the adversarial loss term for G is given by

$$\mathcal{L}_G^{adv} = -\mathrm{E}_{p_{data}}[D(\mathbf{x}_t, \tilde{\mathbf{y}}_t) + D_m(\mathbf{h}_t^{(a)}, \tilde{\mathbf{y}}_t^m)], \quad (1)$$

where $\mathbf{x}_t$ is the 3D face representation input, $\mathbf{h}_t^{(a)}$ is the input audio feature vector, $\tilde{\mathbf{y}}_t$ is the "fake" frame generated by G and $\tilde{\mathbf{y}}_t^m$ the corresponding cropped mouth area of size 64×64. Given that during training we perform self-reenactment, we have access to the ground truth frame $\mathbf{y}_t$. The image Discriminator D is optimised by minimising the loss

$$\begin{aligned}\mathcal{L}_D^{adv} = -\mathrm{E}_{p_{data}}[&\min(0, -1 + D(\mathbf{x}_t, \mathbf{y}_t) \\ &- \min(0, -1 - D(\mathbf{x}_t, \tilde{\mathbf{y}}_t)].\end{aligned} \quad (2)$$

and the mouth Discriminator D_m using a similar loss

$$\begin{aligned}\mathcal{L}_{D_m}^{adv} = -\mathrm{E}_{p_{data}}[&\min(0, -1 + D_m(\mathbf{x}_t, \mathbf{y}_t^m) \\ &- \min(0, -1 - D_m(\mathbf{x}_t, \tilde{\mathbf{y}}_t^m)].\end{aligned} \quad (3)$$

The generative network G is trained by minimising also a reconstruction loss term between the generated and ground frames, in the image pixel space

$$\mathcal{L}_G^{L1} = \mathrm{E}_{p_{data}}[||\tilde{\mathbf{y}}_t - \mathbf{y}_t||_1], \quad (4)$$

as well as the feature space, using feature maps extracted by a pre-trained VGG network [7]:

$$\mathcal{L}_G^{VGG} = \mathrm{E}_{p_{data}}[\sum_l ||VGG_l(\tilde{\mathbf{y}}_t) - VGG_l(\mathbf{y}_t)||_1]. \quad (5)$$

Similarly to VGG loss, we use the two Discriminators to compute visual features from both real and synthetic frames and compute a feature matching loss $\mathcal{L}_G^{FM}$ that was originally proposed in [16] and has been proven very effective at increasing the photo-realism of generated samples.

In addition, we apply both $L1$ and VGG losses on the warped image $\bar{\mathbf{y}}_t^{ref}$, in order to force the dense flow network F to learn a correct flow from the reference image to the desired head pose, obtaining the loss terms $\mathcal{L}_F^{L1}$ and $\mathcal{L}_F^{VGG}$.

To sum up, the overall objective for G is given as:

$$\begin{aligned}\mathcal{L}_G = &\mathcal{L}_G^{adv} + \lambda_{L1}\mathcal{L}_G^{L1} + \lambda_{VGG}\mathcal{L}_G^{VGG} + \lambda_{FM}\mathcal{L}_G^{FM} + \\ &\lambda_{L1}\mathcal{L}_F^{L1} + \lambda_{VGG}\mathcal{L}_F^{VGG},\end{aligned} \quad (6)$$

with $\lambda_{L1} = 50$ and $\lambda_{VGG} = \lambda_{FM} = 10$. The Discriminators are optimised under their corresponding adversarial loss terms

$$\mathcal{L}_D = \mathcal{L}_D^{adv}, \quad \mathcal{L}_{D_m} = \mathcal{L}_{D_m}^{adv}. \quad (7)$$

3. Architecture Details

3.1. Generator G

Dense flow network F (Table 1). The dense flow network consists of an encoding and a decoding part. Its encoder is made up from three convolutional layers, each one with instance normalization units [14] and ReLU activation functions. The last two convolutions are performed with a stride of 2, for down-sampling the input twice. The decoder is equipped with SPADE blocks [11], which are used to "inject" the 3D face representation $\mathbf{x}_{t-k:t}$ (modulation input). Here we down-sample $\mathbf{x}_{t-k:t}$ to match it with the spatial size of each SPADE layer, similarly to the original work [11]. We employ two Pixel Shuffle [13] layers, for up-sampling. Finally, dense flow is calculated with a 7×7 convolutional output layer.

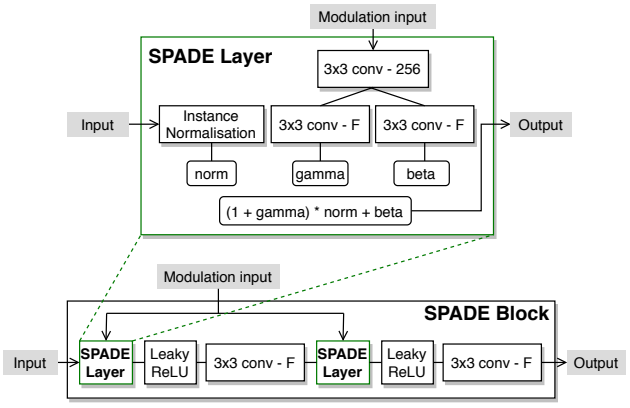
Rendering network R (Table 2). Our rendering network has an encoder-decoder architecture as well. Its encoder has a similar structure to the encoder of F. The decoder is built from alternating SPADE and AdaIN blocks, which are used to condition synthesis on our multi-scale visual feature maps and audio feature vectors respectively. We use Pixel Shuffle layers for up-sampling, since we noticed it performs better than simple up-sampling operations (e.g. nearest neighbor, linear, bi-linear). After the last decoding block, a convolutional layer is placed for the computation of the synthetic RGB image.

	Block		Output size
	Input		$(256, 256, 6)$
7×7 conv-32	Inst. Norm.	ReLU	$(256, 256, 32)$
3×3 conv-128	Inst. Norm.	ReLU	$(128, 128, 128)$
3×3 conv-512	Inst. Norm.	ReLU	$(64, 64, 512)$
	SPADE Block		$(64, 64, 512)$
	SPADE Block		$(64, 64, 512)$
	SPADE Block		$(64, 64, 512)$
	Pixel Shuffle		$(128, 128, 128)$
	SPADE Block		$(128, 128, 128)$
	Pixel Shuffle		$(256, 256, 32)$
	7×7 conv-2		$(256, 256, 2)$

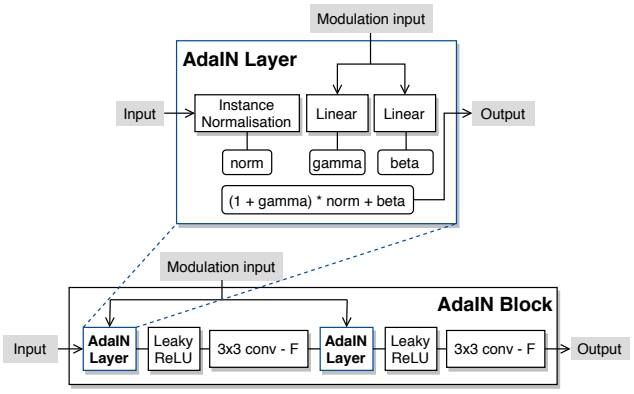
Table 1: Architecture of dense flow network F.

	Block		Output size
	Input		$(256, 256, 9)$
7×7 conv-32	Inst. Norm.	ReLU	$(256, 256, 32)$
3×3 conv-128	Inst. Norm.	ReLU	$(128, 128, 128)$
3×3 conv-512	Inst. Norm.	ReLU	$(64, 64, 512)$
	SPADE Block		$(64, 64, 512)$
	AdaIN Block		$(64, 64, 512)$
	Pixel Shuffle		$(128, 128, 128)$
	SPADE Block		$(128, 128, 128)$
	AdaIN Block		$(128, 128, 128)$
	Pixel Shuffle		$(256, 256, 32)$
	SPADE Block		$(256, 256, 32)$
	AdaIN Block		$(256, 256, 32)$
	SPADE Block		$(256, 256, 32)$
LReLU	7×7 conv-3	tanh	$(256, 256, 3)$

Table 2: Architecture of rendering network R.



(a) SPADE Block architecture.



(b) AdaIN Block architecture

Figure 1: Our SPADE and AdaIN blocks are based on the SPADE Resnet blocks proposed in [11], but without a residual component, as we always keep the same number of input channels F at the output, both on SPADE and AdaIN blocks.

3.2. Discriminators D and D_m

Both D and D_m have a similar architecture to the discriminator presented in [11]. We apply Spectral Normalisation [9] to all normalisation layers of the Discriminators.

4. Additional results

In Fig. 2 and Fig. 3 we present a few more generated samples using VoxCeleb test set [10]. Here, we also include the predicted flow and warped image in the results.

Figure 2: Reconstruction. From left to right: reference, reference 3D face, driving 3D face, flow, warped, generated, driving.

Figure 3: Reenactment. From left to right: reference, reference 3D face, driving 3D face, flow, warped, generated, driving.

5. Evaluation metrics

We quantitatively compare *HeadGAN* with the baselines, using the metrics described below.

L1 distance (L1). We evaluate the reconstructive ability of models by computing the mean $l1$-distance, between the synthesised and ground truth frames. We average the distance across channels, pixel locations and frames in the test set, to obtain the L1 metric. Please note that RGB channels are in the range [0, 255].

Peak signal-to-noise ratio (PSNR). This is another metric to measure the quality of reconstructed videos. PSNR is the ratio between the maximum possible power of a signal and the power of noise that affects the fidelity of its representation, defined as: $20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10} MSE$. Here, $MAX_I = 255$ and MSE denotes the mean squared error, computed across color channels, spatial locations and frames. PSNR is expressed in dBs.

Learned Perceptual Image Patch Similarity (LPIPS). Perceptual metrics such as PSNR are simple shallow functions that are not able to account for many nuances of human perception. LPIPS [18] uses a neural network that is trained to solve challenging visual prediction and modeling tasks as a feature extractor, since the network learns a representation that correlates well with perceptual judgments. Then, a similarity score between two images is calculated based on visual features.

Fréchet Inception Distance (FID). We employ FID [6, 12] as a measure of similarity between the dataset of real images and the dataset of images generated by the models. This score provides a useful insight into the photo-realism of synthetic frames.

Fréchet Video Distance (FVD). Given that we handle video data, it is important to evaluate the generative performance of models using a metric which takes into account the temporal coherence between frames. To that end, we calculate the FVD score [15] of generated sequences, which has shown to correlate well with the human judgment on visual quality of generated videos.

Cosine Similarity (CSIM). Cosine similarity is a widely-used metric, which measures identity preservation in synthetic frames. We use ArcFace [4] as an identity recognition network, in order to compute pairs of embedding vectors from the driving and corresponding generated images. Then, we calculate the cosine similarity between all pairs of embedding vectors in the dataset and report its average value. During reenactment, where no ground truth images are available, we extract the embedding vector from the reference image and compare it with the embeddings coming from synthetic frames. This leads to smaller CSIM values in reenactment, as the poses of the source and generated images do not match and the identity recognition network's output is not completely unaffected by a person's pose.

Action Units Hamming distance (AU-H). In order to measure the facial expression transferability of models, we use OpenFace [1] and more specifically [2], for the detection of Action Units (AU) in driving and generated images. Facial Action Coding System (FACS) is a system to taxonomise human facial movements by their appearance on the face. Using FACS it is possible to code nearly any anatomically possible facial expression, deconstructing it into the specific AUs that produced the expression. It is a common standard to objectively describe facial expressions. We use OpenFace to recognise if a set of AUs is present in a facial image or not, calculating a boolean vector of AUs. Then, we compute the Hamming distance $\in [0, 1]$ between AU boolean vectors, extracted from the corresponding driving and synthetic frames, and average across all frames.

Average Rotation Distance (ARD). This metric evaluates pose transfer. We use the camera parameters from 3D face reconstruction to compute the Euler angles that correspond to head poses in the driving and generated frames. Then, the average $l1$-distance of Euler angles across all frames is determined, in terms of degrees.

Average Rotation Error (ARE). This is a metric similar to ARD, but measures the average $l1$-distance of Euler angles from the frontal pose (zero degrees), across all images.

References

[1] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016. 5

[2] T. Baltrušaitis, M. Mahmoud, and P. Robinson. Cross-dataset learning and person-specific normalisation for automatic action unit detection. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 06, pages 1–6, 2015. 5

[3] James Booth, Anastasios Roussos, Stefanos Zafeiriou, Allan Ponniah, and David Dunaway. A 3d morphable model learnt from 10,000 faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[4] Jiankang Deng, Jia Guo, Xue Niannan, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 5

[5] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6626–6637. Curran Associates, Inc., 2017. 5

[7] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017. 1

[8] Jae Hyun Lim and Jong Chul Ye. Geometric gan, 2017. 1

[9] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 2

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

[11] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[12] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.1.1. 5

[13] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016. 2

[14] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. 2

[15] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 5

[16] Xiangyu Xu, Deqing Sun, Jinshan Pan, Yujin Zhang, Hanspeter Pfister, and Ming-Hsuan Yang. Learning to super-resolve blurry face and text images. In *Proceedings of the IEEE international conference on computer vision*, pages 251–260, 2017. 1

[17] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[18] Richard Yi Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings - 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 586–595. IEEE Computer Society, 2018. 5