Lifting 2D StyleGAN for 3D-Aware Face Generation (Supplenmentary Material)

Yichun Shi Divyansh Aggarwal Anil K. Jain

Michigan State University

{shiyichu,aggarw49}@msu.edu, jain@cse.msu.edu

1. Joint Probability as Lower Bound

Here we show the relationship between $\log p(I'_w, w')$ and the likelihood $p(I'_w) = \int_w p_{G_{2D}}(x|w)p(w)dw$. In particular, by introducing an additional encoder $q(w)$, the variational lower bound [4] is given by:

$$\log p(I'_w) \geq \mathbb{E}_{w\sim q(w)}[\log \frac{p(I'_w, w)}{q(w)}]. \tag{10}$$

In the original Variational AutoEncoder [4], q should be a conditional distribution defined as an image encoder to approximate the posterior $p(w|I'_w)$. Here, we use the manipulation network to replace the image encoder. Formally, consider q to be a Gaussian distribution conditioned on the style code $\hat{w}$, perturbation parameters V' and L', i.e. $q(w|M(\hat{w}, V', L')) = q(w|w') = \mathcal{N}(w', \sigma^2_{w'}\mathbf{I})$, if we further consider q to be a deterministic approximation, i.e. $\sigma_{w'}$ is a fixed value and $\sigma_{w'} \to 0$, the lower bound in Equation 10 becomes $\log p(I'_w, w') + c$, where c is a constant and could be omitted.

2. Additional Implementation Details

We use an Adam optimizer [3] with a learning rate of 1e-4 to train the model. The 3D generator is trained for $30,000$ steps. For the perceptual loss, we average the ℓ_2 distance between the output of `relu2_2`, `relu3_3`, `relu4_3` to compute the loss. The feature maps are ℓ_2-normalized on each pixel. Further, the images are downsampled with a scale of $\times 1$, $\times 2$, $\times 4$ and the perceptual losses are averaged for different resolutions. For the 3D renderer, we assume a fov of 10. The output depth maps are normalized between $(0.9, 1.1)$. The output viewpoint rotation and translation are normalized between (-60°,60°) and (-0.1,0.1), respectively. The lighting coefficients are normalized to (0,1). For the perturbation, we empirically sample yaw angles from a uniform distribution over $[-45°,45°]$, and pitch angles from $[-10°,10°]$. The roll angle is fixed at 0. For the lighting, we found it a difficult task for StyleGAN2 to synthesize faces with manually chosen lighting parameters. Thus, we shuffle the lighting parameters across the batch. The identity regularization loss is only applied to faces that are perturbed with a yaw angle within $[-25°,25°]$. For the identity preservation loss, we train an 18-layer ResNet [1] as the face embedding using the CosFace loss [8] on the CASIA-Webface dataset [10].

2.1. Network Architectures

The viewpoint decoder D_V and light decoder D_L are both 4-layer MLPs with LeakyReLU [6] as activation function. The latent manipulation network is composed of two parts, the first part are composed of three 4-layer MLPs to encode latent code $\hat{w}$, viewpoint V_0 and light L_0, respectively, whose outputs are summed into a feature vector for the second part. The second part is another 4-layer MLP that outputs a new style code. The hidden size of all MLPs in our work is 512, same as the style code. For shape decoder D_S and transformation decoder D_T, we mainly follow the decoder structure of Wu *et al.* [9], whose details are shown in Table 1 and Table 2, respectively. In detail, Conv(c_{in}, c_{out}, k, s) refers to a convolutional layer with c_{in} input channels, c_{out} output channels, a kernel size of k and a stride of s. Deconv is defined similarly. "GN" refers to the group normalization, where the argument is the number of groups. The "InjectConv" refers to a convolutional layer that takes the last feature map of 2D generator as input and injects its output to the decoder via summation. We found such a design helps to recover the facial details in the depth map.

2.2. Additional Results

In Figure 1, we show more results of our method and other baselines for rotating faces into different yaw angles. Figure 2 shows the results of rotating faces into different pitch angles. Note that for HoloGAN [7], we use the officially released code and train the model on our dataset. For CONFIGNet [5] and DiscoFaceGAN [2], since they involve additional synthesized data for training, we use their pretrained models. Although most baselines are able to gener-

Shape Decoder	Output size
4-layer MLP	512
Deconv(512,512,4,1) + ReLU	512×4×4
Conv(512,512,3,1) + ReLU	512×4×4
Deconv(512,256,4,2) + GN(64) + ReLU	256×8×8
Conv(256,256,3,1) + GN(64) + ReLU	256×8×8
Deconv(256,128,4,2) + GN(32) + ReLU	128×16×16
Conv(128,128,3,1) + GN(32) + ReLU	128×16×16
Deconv(128,64,4,2) + GN(16) + ReLU	64×32×32
Conv(64,64,3,1) + GN(16) + ReLU	64×32×32
Deconv(64,32,4,2) + GN(8) + ReLU	32×64×64
Conv(32,32,3,1) + GN(8) + ReLU	32×64×64
Deconv(32,16,4,2) + GN(4) + ReLU	16×128×128
Conv(16,16,3,1) + GN(4) + ReLU	16×128×128
Upsample(2) + InjectConv(32,161,1)	16×128×128
Conv(16,16,3,1) + GN(4) + ReLU	16×256×256
Conv(16,16,5,1) + GN(4) + ReLU	16×256×256
Conv(16,1,5,1)	1×256×256

Table 1: The architecture of shape decoder.

Transformation Map Decoder	Output size
4-layer MLP	512
Deconv(512,512,4,1) + ReLU	512×4×4
Deconv(512,256,4,2) + GN(64) + ReLU	256×8×8
Deconv(256,128,4,2) + GN(32) + ReLU	128×16×16
Deconv(128,64,4,2) + GN(16) + ReLU	64×32×32
Deconv(64,32,4,2) + GN(8) + ReLU	32×64×64
Deconv(32,16,4,2) + GN(4) + ReLU	16×128×128
Upsample(2) + Conv(16,16,3,1) + GN(4) + ReLU	16×256×256
Conv(16,1,5,1)	1×256×256

Table 2: The architecture of transformation map decoder.

ate high-quality face images under near-frontal poses, clear content change could be observed in larger poses. In particular, HoloGAN and CONFIGNet generates blurred faces for larger poses, while the expression changes in the results of DiscoFaceGAN. In comparison, our method, though also suffer from quality degradation in larger poses, has a stricter control in terms of the content. We also provide the FID score of different methods in Table 3, where our method achieves second best image quality, even though our images are re-rendered from 2D generated images.

In Figure 3 and Figure 4, we show additional examples of rotating generated faces with different yaw and pitch angles, respectively. In brief, we observe that the pre-trained StyleGAN2 (with our style manipulation network) is able to change poses to a certain degree, but fails to generate faces with larger pose angles that do not exist in the training data. Further, we see a similar trend of expression change when changing pitch angles as in DiscoFaceGAN, possibly due to the intrinsic bias in the FFHQ dataset. In contrast, the 3D generator distilled from StyleGAN2 extends this rotation capability to a larger degree with better content preservation.

Method	FID↓
HoloGAN *et al*. [7]	100.28
DiscoFaceGAN *et al*. [2]	12.90
CONFIGNet *et al*. [5]	43.05
LiftedGAN (proposed)	29.81

Table 3: Quantitative Evaluation of the generated image quality between this work and baselines. The scores of "DiscoFaceGAN" and "CONFIGNet" are reported in their original papers. The scores of HoloGAN and our method are computed from sampled images of random poses.

Figure 1: Qualitative comparison with state-of-the-art methods on 3D-controllable GANs. Note that all these faces are generated by randomly sampling from latent space, therefore we cannot compare the manipulation over the same face. The example faces are supposed to have a yaw degree of -60,-45,-30,15,0,15,30,45,60.

Figure 2: Qualitative comparison with state-of-the-art methods on 3D-controllable GANs. Note that all these faces are generated by randomly sampling from latent space, therefore we cannot compare the manipulation over the same face. The example faces are supposed to have a pitch degree of -30,-20,-10,0,10,20,30.

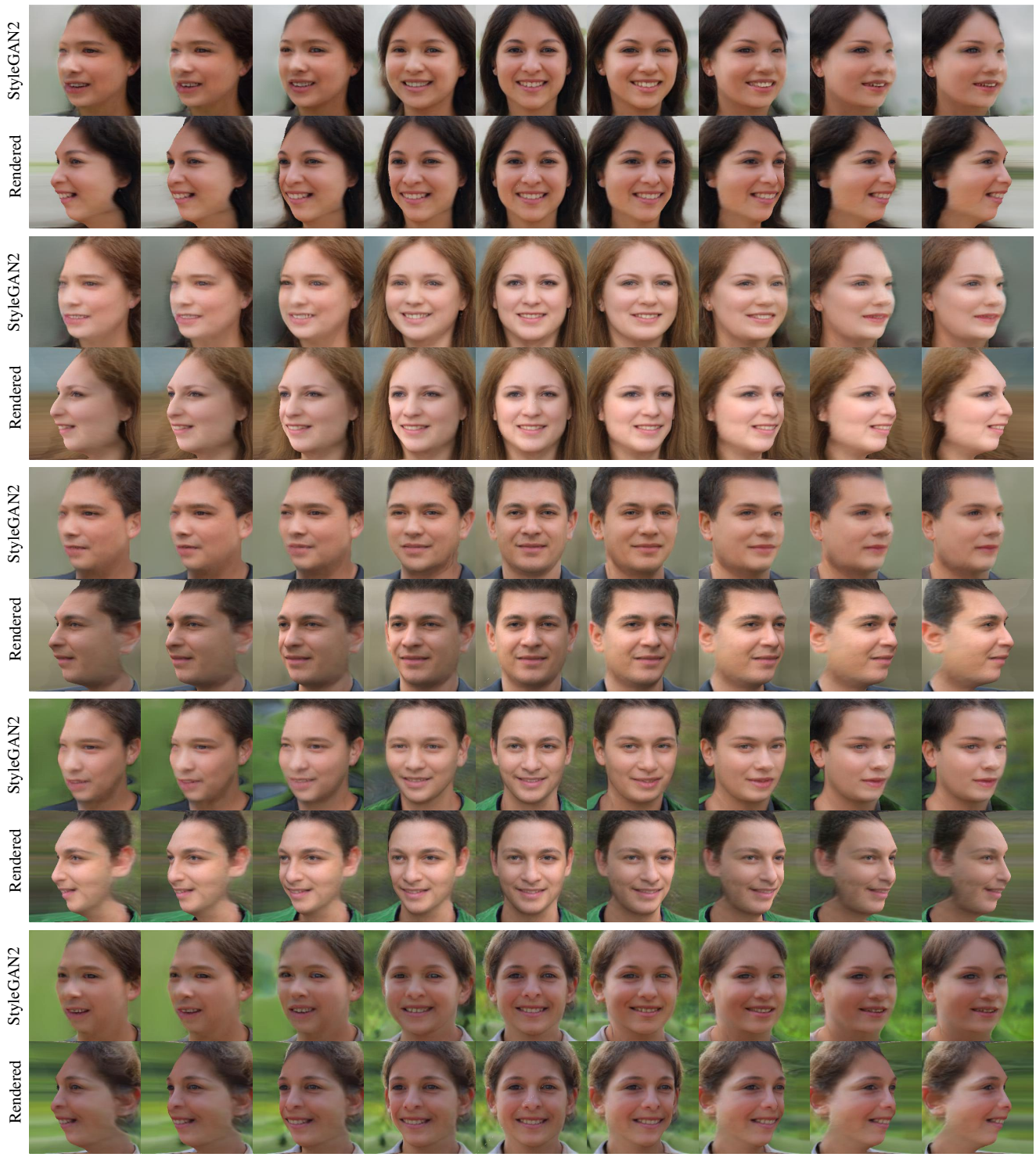

Figure 3: Results of rotating faces to yaw angles of -60,-45,-30,-15,0,15,30,45,60. For each two rows, the first row shows the results of generating by manipulating the StyleGAN2 latent style code, while the second row shows the results of 3D rendered faces.

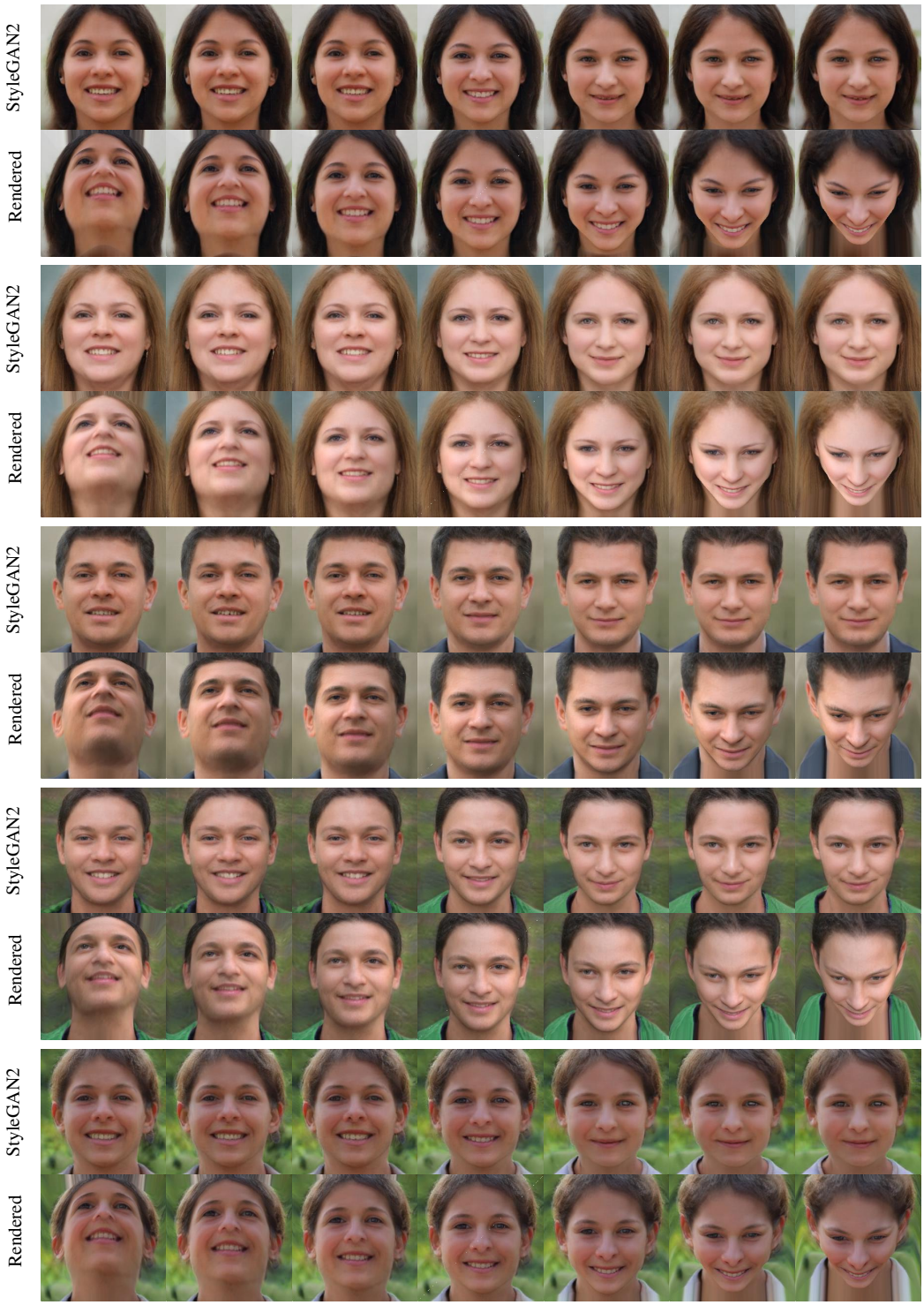

Figure 4: Results of rotating faces to pitch angles of -30,-20,-10,0,10,20,30. For each two rows, the first row shows the results of generating by manipulating the StyleGAN2 latent style code, while the second row shows the results of 3D rendered faces.

References

[1] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 1

[2] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3d imitative-contrastive learning. In *CVPR*, 2020. 1, 2

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 1

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 1

[5] Marek Kowalski, Stephan J Garbin, Virginia Estellers, Tadas Baltrušaitis, Matthew Johnson, and Jamie Shotton. Config: Controllable neural face image generation. In *ECCV*, 2020. 1, 2

[6] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. 1

[7] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *CVPR*, 2019. 1, 2

[8] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, 2018. 1

[9] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *CVPR*, 2020. 1

[10] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint:1411.7923*, 2014. 1