Style Transfer for 2D Talking Head Generation

Trong Thang Pham[2], Tuong Do[1,3], Nhat Le[1],
Ngan Le[2], Hung Nguyen[1], Erman Tjiputra[1], Quang Tran[1], Anh Nguyen[3]
[1]AIOZ, Singapore, [2]University of Arkansas, USA, [3]University of Liverpool, UK

1. Detailed Architectures of Style Mapping

The implementation details of the networks in our Style Mapping module are described as follows.

Keypoint Extractor Given the input neutral image $\boldsymbol{I}_s$, a set of k number of keypoints $\mathrm{c_k}$ is extracted using a 3D U-Net encoder-decoder [5]. First, we project the encoded feature maps onto 3D volumes using 1×1 convolution. The encoder has 5 down-sampling layers. The decoder part of the network has 5 up-sampling layers. Finally, the keypoints are predicted from the final 3D convolution layer. For more details, please visit Table 1.

Layer	Kernel Shape	Output Shape	Input Shape
AntiAliasInterpolation2d		[1, 3, 64, 64]	[1, 3, 256, 256]
Sequential (DownBlock2d*5)		[1, 1024, 2, 2]	[1, 3, 64, 64]
Conv2d	[1024, 16384, 1, 1]	[1, 16384, 2, 2]	[1, 1024, 2, 2]
Sequential (UpBlock3d*5)		[1, 32, 16, 64, 64]	[1, 1024, 16, 2, 2]
Conv3d	[32, 15, 3, 3, 3]	[1, 15, 16, 64, 64]	[1, 32, 16, 64, 64]
KPDetector		[1, 15, 3]	[1, 15, 16, 64, 64]

Table 1. Keypoint Extractor architecture.

Pose Expression Network We extract the pose, parameterized by a translation vector $\tau \in \mathbb{R}^3$ and a rotation matrix $\mathrm{R} \in \mathbb{R}^{3\times3}$, and expression information ε_k from the image by a Pose Expression Network. We adopt the same architecture as in [3]. A sequence of ResNet blocks is followed by global pooling to eliminate the spatial dimension. The rotation angles, translation vector, and expression information are then estimated using various linear layers. To estimate head pose, we divide the entire angle range into 66 bins of angles for yaw, pitch, and roll. Then, the network predicts the probability of the bin to which the target angle should belong. The rotation angles are then converted into a 3D rotation matrix. The final keypoints that contain the geometric and style information of the neutral image and Style Reference image are computed as in Equations **??** and **??**. Table 2 shows the architectural details of the Pose Expression Network.

Layer	Kernel Shape	Output Shape	Input Shape
Conv2d	[3, 64, 7, 7]	[1, 64, 128, 128]	[1, 3, 256, 256]
SynchronizedBatchNorm2d	[64]	[1, 64, 128, 128]	[1, 64, 128, 128]
MaxPool2d		[1, 64, 64, 64]	[1, 64, 128, 128]
Conv2d	[64, 256, 1, 1]	[1, 256, 64, 64]	[1, 64, 64, 64]
SynchronizedBatchNorm2d	[256]	[1, 256, 64, 64]	[1, 256, 64, 64]
Sequential(ResBottleneck*3)		[1, 256, 64, 64]	[1, 256, 64, 64]
Conv2d	[256, 512, 1, 1]	[1, 512, 64, 64]	[1, 256, 64, 64]
SynchronizedBatchNorm2d	[512]	[1, 512, 64, 64]	[1, 512, 64, 64]
ResBottleneck		[1, 512, 32, 32]	[1, 512, 64, 64]
Sequential		[1, 512, 32, 32]	[1, 512, 32, 32]
Conv2d	[512, 1024, 1, 1]	[1, 1024, 32, 32]	[1, 512, 32, 32]
SynchronizedBatchNorm2d	[1024]	[1, 1024, 32, 32]	[1, 1024, 32, 32]
ResBottleneck		[1, 1024, 16, 16]	[1, 1024, 32, 32]
Sequential(ResBottleneck*3)		[1, 1024, 16, 16]	[1, 1024, 16, 16]
Conv2d	[1024, 2048, 1, 1]	[1, 2048, 16, 16]	[1, 1024, 16, 16]
SynchronizedBatchNorm2d	[2048]	[1, 2048, 16, 16]	[1, 2048, 16, 16]
ResBottleneck		[1, 2048, 8, 8]	[1, 2048, 16, 16]
Sequential(ResBottleneck*3)		[1, 2048, 8, 8]	[1, 2048, 8, 8]
Reshape		[1, 2048]	[1, 2048, 8, 8]
Linear (yaw)	[2048, 66]	[1, 66]	[1, 2048]
Linear (pitch)	[2048, 66]	[1, 66]	[1, 2048]
Linear (roll)	[2048, 66]	[1, 66]	[1, 2048]
Linear (translation)	[2048, 3]	[1, 3]	[1, 2048]
Linear (expression)	[2048, 45]	[1, 45]	[1, 2048]

Table 2. Pose Expression Network.

Warping Network To reconstruct the output Intermediate Style Pattern image $\boldsymbol{I}_o$, we use a Warping Network to warp the feature volume of the neutral image using the two extracted keypoints set C and $\bar{\mathrm{C}}$ and their geometry information. Specifically, following first-order approximation [4], we estimate a warping function w_k for each k-th keypoints of the Style Reference image and the keypoints of the neutral image.

$$w_k : R\bar{R}^{-1}\left(\bar{p} - \bar{\mathrm{C}}_k\right) + \mathrm{C_k} \mapsto p \tag{1}$$

where p and $\bar{p}$ are the 3D voxel location of the feature volume of the neutral image and Style Reference.

For each kth keypoint , we apply w_k on every location of the neutral feature volume $w_k(\boldsymbol{F}_s)$ to obtain the kth warped volume. Then, we concatenate all the warped volumes and pass them into a Warping Network to predict K composition maps $m = \{m_1, m_2, ..., m_K\}$, which contains the composition weights to aggregate the warping functions. In particular, we apply the softmax function at each location so that the composition weights can satisfy the condition:

$$\sum_k (m_k)(\bar{p}) = 1 \quad \& \quad 0 \leq m_k(\bar{p}) \leq 1, \qquad \forall \bar{p} \tag{2}$$

The final warped volume $w(\boldsymbol{F}_s)$ is calculated as the linear combination of the K warped volumes $w(\boldsymbol{F}_s) = \sum_{k=1}^{K} m_k w_k(\boldsymbol{F}_s)$. To handle occlusions caused by the warping, the network also predicts a 2D occlusion mask o, which is used as input of the Intermediate Generator in addition to the final warped volume. The detail of the Warping Network is described in Table 3.

Layer	**Kernel Shape**	**Output Shape**	**Input Shape**
Conv3d	[32, 4, 1, 1, 1]	[1, 4, 16, 64, 64]	[1, 32, 16, 64, 64]
SyncBatchNorm3d	[4]	[1, 4, 16, 64, 64]	[1, 4, 16, 64, 64]
Hourglass		[1, 112, 16, 64, 64]	[1, 80, 16, 64, 64]
Conv3d	[112, 16, 7, 7, 7]	[1, 16, 16, 64, 64]	[1, 112, 16, 64, 64]
Conv2d	[$112 * 16$, 1, 7, 7]	[1, 1, 64, 64]	[1, 112, 16, 64, 64]

Table 3. Warping Network.

Intermediate Generator The network details can be found in Table 4. This network takes the warped feature volume $w(\boldsymbol{F}_s)$ of the neutral image and projects it back to 2D dimensions. Then, the input feature is multiplied with the occlusion mask o obtained from the Warping Network. Finally, a 2D residual block series (6 blocks in total), 2 upsampling layers, and a convolution layer are applied to construct the final Intermediate Style Pattern image. Since Intermediate Generator is an image generator, it contains LSGAN loss [2] that stabilizes the training process by adopting least squares. To achieve high fidelity, we minimize differences at the pixel-wise level and feature level as well as ensure the consistency of estimated keypoints. We also minimize high-level differences of style discrepancies through perceptual loss [1].

2. Demonstration

The demo of our proposal can be view in this link: https://drive.google.com/file/d/16cjbJc9ZFMl6jnFTnJe0J2grhaTKA2y-/view?usp=sharing

Layer	**Kernel Shape**	**Output Shape**	**Input Shape**
Conv2d	[3, 64, 3, 3]	[1, 64, 256, 256]	[1, 3, 256, 256]
SyncBatchNorm2d	[64]	[1, 64, 256, 256]	[1, 64, 256, 256]
DownBlock2d		[1, 128, 128, 128]	[1, 64, 256, 256]
DownBlock2d		[1, 256, 64, 64]	[1, 128, 128, 128]
Conv2d	[256, 512, 1, 1]	[1, 512, 64, 64]	[1, 256, 64, 64]
ResBlock3d*6		[1, 32, 16, 64, 64]	[1, 32, 16, 64, 64]
WarpingNetwork		[1, 1, 64, 64]	[1, 32, 16, 64, 64]
Conv2d	[512, 256, 3, 3]	[1, 256, 64, 64]	[1, 512, 64, 64]
SyncBatchNorm2d		[1, 256, 64, 64]	[1, 256, 64, 64]
Conv2d	[256, 256, 1, 1]	[1, 256, 64, 64]	[1, 256, 64, 64]
Conv2d	[256, 512, 3, 3]	[1, 512, 64, 64]	[1, 256, 64, 64]
Upsample		[1, 512, 128, 128]	[1, 512, 64, 64]
Upsample		[1, 256, 256, 256]	[1, 256, 128, 128]
Conv2d	[64, 3, 3, 3]	[1, 3, 256, 256]	[1, 64, 256, 256]

Table 4. Intermediate Generator.

References

[1] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *ECCV*, 2016. 2

[2] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. In *ICCV*, 2017. 2

[3] Nataniel Ruiz, Eunji Chong, and James M Rehg. Fine-grained head pose estimation without keypoints. In *CVPR*, 2018. 1

[4] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *NIPS*, 32, 2019. 1

[5] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-Shot Free-View Neural Talking-Head Synthesis for Video Conferencing. In *CVPR*, 2021. 1