Supplementary Material for Talking Head Anime 4: Improved Model and Its Disilltation

1 Full System's Architecture Details

1.1 U-Net with Attention

The new backbones of the half-resolution rotator and editor are U-Net with attention [2], which are frequently used in diffusion models. We base our architecture on conditional U-Nets in the diffusion autoencoder paper by Preechakul et al. [5]. There, the U-Net takes as input a feature tensor, a time value, and a 1D conditioning vector. The time value and the conditioning vector are mingled tensors derived from the input feature tensor through adaptive instance normalization (AdaIN) units [3] that are parts of residual blocks [1]. A residual block would have two AdaIN units that are applied in succession: the first for time and the second for conditioning vector. In the diffusion autoencoder paper, the conditioning vector is a 512-dimensional vector. In our case, the conditioning vector is the 6-dimensional pose vector. (While the full pose vector has 45 parameters, only 6 that concern the movement of the body are relevant to the networks that we modify.) For our networks, the time value is always 0 and is totally redundant. We kept the code related to time embedding in place in order to reduce implementation effort.

The configurations for the backbone networks are given in Table 1. Both networks scale down the feature tensors to 16×16 before scaling them back up to the original resolution. Attention blocks are only present at the 16×16 resolution. The bottleneck part of each network has 4 residual blocks alternating with three attention blocks. (In other words, there are $3 + 2 = 5$ attention blocks in total.) Each attention block has 8 attention heads.

Hyperparameter	**HRR**	**Editor**
image resolution	256×256	512×512
# base channels	64	32
channel multipliers	1, 2, 4, 4, 4	1, 2, 4, 8, 8, 8
# residual blocks per level	1	1
# bottleneck residual blocks	4	4
resolution with attention blocks	16	16
# attention heads	8	8

Table 1: Configurations of the U-Net with attention backbones for the half-resolution rotator (HRR) and editor.

The half-resolution rotator and the editor differ not only on the configurations of their backbones but how the backbones are "wrapped" by extra units to that they conform to the networks' interfaces. We discuss these extra units in the two following subsections.

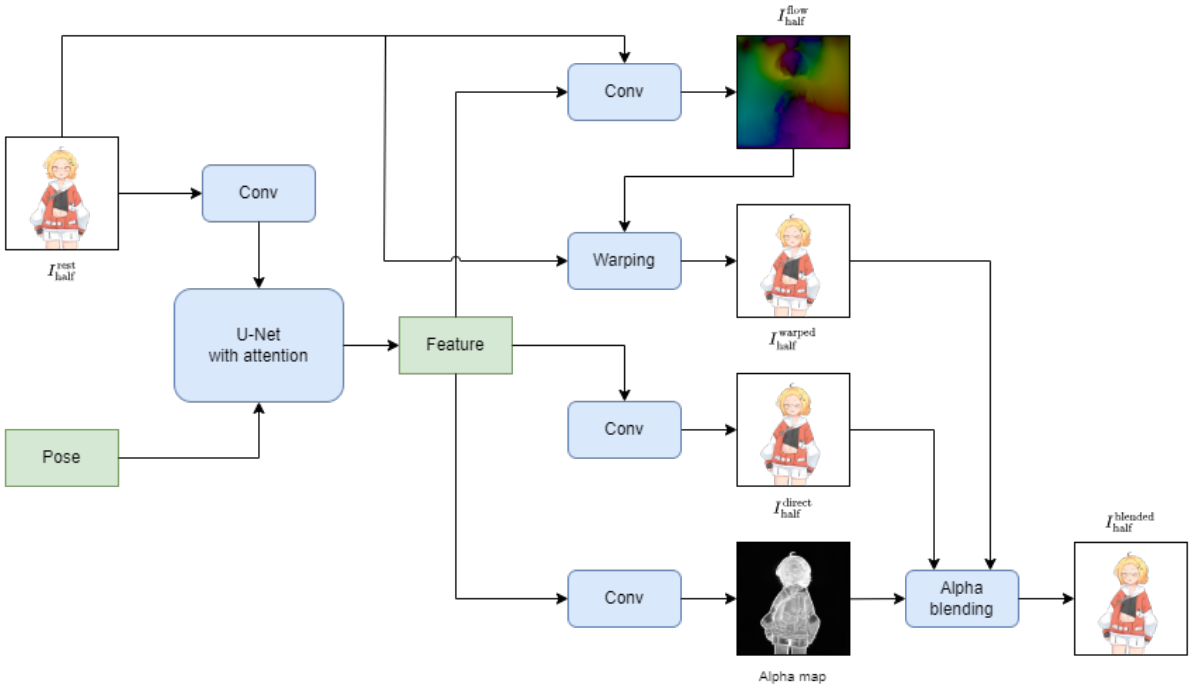

Figure 1: The new half-resolution rotator.

1.2 Half-Resolution Rotator

The half-resolution rotator is depicted in Figure 1. It takes as input:

1. $I^{\text{rest}}_{\text{half}}$, a $4 \times 256 \times 256$ RGBA image of the character in rest pose obtained by downscaling the original input image, and
2. $\mathbf{p}$, a 6-dimensional pose vector.

Because the U-Net with attention backbone takes in a $64 \times 256 \times 256$ tensor as input, $I^{\text{rest}}_{\text{half}}$ must be converted to this shape with a convolution layer. The backbone produces another $64 \times 256 \times 256$ feature tensor, which is then used to perform several image processing operations. (See Section 3 of the paper.)

- *Warping.* The feature tensor is passed to a convolution layer to produce an appearance flow $I^{\text{flow}}_{\text{half}}$ of size $2 \times 256 \times 256$. It is then used to warp the input image ($I^{\text{rest}}_{\text{half}}$) to produce a warped image $I^{\text{warped}}_{\text{half}}$.
- *Direct generation.* The feature tensor is converted to a $4 \times 256 \times 256$ RGBA image, denoted by $I^{\text{direct}}_{\text{half}}$.
- *Blending.* The feature tensor is converted to a $1 \times 256 \times 256$ alpha map, which is then used to blend the warped image $I^{\text{warped}}_{\text{half}}$ and the directly generated image $I^{\text{direct}}_{\text{half}}$ together. The result is called $I^{\text{blended}}_{\text{half}}$.

The half-resolution rotator outputs $I^{\text{flow}}_{\text{half}}$, $I^{\text{warped}}_{\text{half}}$, $I^{\text{direct}}_{\text{half}}$, and $I^{\text{blended}}_{\text{half}}$. These image are used for training. However, at test time, $I^{\text{flow}}_{\text{half}}$ and $I^{\text{blended}}_{\text{half}}$ are used by the next network, the editor.

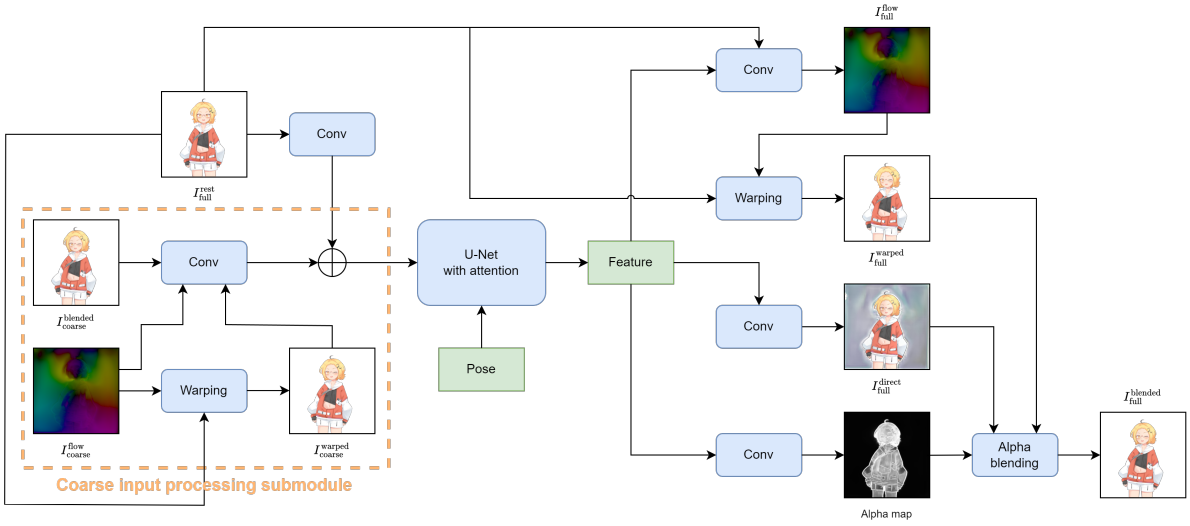

Figure 2: The new editor.

1.3 Editor

The architecture of the editor is depicted in Figure 2. It takes 4 inputs:

1. $I^{\text{rest}}_{\text{full}}$, the original character image at the 512×512 resolution,
2. $\mathbf{p}$, the 6-dimensional pose vector,
3. $I^{\text{blended}}_{\text{coarse}}$, which is $I^{\text{blended}}_{\text{half}}$ scaled up to 512×512, and
4. $I^{\text{flow}}_{\text{coarse}}$, which is $I^{\text{flow}}_{\text{half}}$ scaled up to 512×512.

The way the editor processes these input is quite similar to what the half-resolution rotator does. The pose vector is passed to the backbone directly. The input image $I^{\text{rest}}_{\text{full}}$ is convolved to create a $32 \times 512 \times 512$ tensor. The two other inputs are passed through what we call the "coarse input processing submodule," which carries out the following steps.

- First, the coarse appearance flow $I^{\text{flow}}_{\text{coarse}}$ is used to warped the original input image $I^{\text{rest}}_{\text{full}}$ to obtain the coarse warped image $I^{\text{warped}}_{\text{coarse}}$.
- Second, $I^{\text{blended}}_{\text{coarse}}$, $I^{\text{flow}}_{\text{coarse}}$, and $I^{\text{warped}}_{\text{coarse}}$ are concatenated, and the resulting tensor is convolved to form a $32 \times 512 \times 512$ tensor.
- Third, the result form the last step is added to the output of convolution layer that was applied to the original image to produce a $32 \times 512 \times 512$ tensor.

The resulting tensor is then passed to the backbone U-Net with attention. The output of the backbone is processed in the same way as what the half-resolution rotator does. This produces four tensors at full resolution: $I^{\text{flow}}_{\text{full}}$, $I^{\text{warped}}_{\text{full}}$, $I^{\text{direct}}_{\text{full}}$, and $I^{\text{blended}}_{\text{full}}$.

Note that, if we remove the coarse input processing submodule, the architecture of the editor would be exactly the same as the half-resolution rotator. Hence, the editor can be thought of as a network that also rotates the body given in the original image $I^{\text{rest}}_{\text{full}}$, but it takes the coarse inputs, $I^{\text{blended}}_{\text{coarse}}$ and $I^{\text{flow}}_{\text{coarse}}$, as hints. We will exploit this property in the training process of the editor.

2 Full System's Training Details

2.1 Half-Resolution Rotator

The half-resolution rotator is trained with the following 6-termed loss that is a combination of the L1 loss and the perceptual content loss [4]. More concretely,

$$\mathcal{L}_{\text{HRR}} = \ell_{\text{L1}}\Big(\mathtt{L}_{\text{L1}}^{\text{warped}} + \mathtt{L}_{\text{L1}}^{\text{direct}} + \mathtt{L}_{\text{L1}}^{\text{blended}}\Big) + \ell_{\text{percept}}\Big(\mathtt{L}_{\text{percept}}^{\text{warped}} + \mathtt{L}_{\text{percept}}^{\text{direct}} + \mathtt{L}_{\text{percept}}^{\text{blended}}\Big).$$

The ℓ_{L1} and ℓ_{percept} are loss weights, which change once during the training process. (More on this later.) The loss terms that have subscripts "L1" are given by

$$\mathtt{L}_{L1}^{\square} = \frac{\|I_{\text{half}}^{\square} - I_{\text{half}}^{\text{posed}}\|_1}{C \times H \times W}$$

where C, H, and W are the channels, height, and width of the tensors, respectively. The $I_{\text{half}}^{\text{posed}}$ is the groundtruth posed image in the training dataset scaled down to 256×256, and $I_{\text{half}}^{\square}$ are the outputs of the half-resolution rotator as defined in Section 1.2. The loss with subscripts "percept" is given by

$$\mathtt{L}_{\text{percept}}^{\square} = \Phi(I_{\text{half}}^{\square} - I_{\text{half}}^{\text{posed}})$$

and

$$\Phi(I_1, I_2) = \sum_{i=1}^{3} c_i\big(\|\phi_i(I_1^{\text{rgb}}) - \phi_i(I_2^{\text{rgb}})\|_1 + \|\phi_i(I_1^{\text{aaa}}) - \phi_i(I_2^{\text{aaa}})\|_1\big).$$

Here,

- $\phi_i(\cdot)$ denote the feature tensor outputted by the ith used layer in the VGG16 network [6], and we use the `relu1_2`,relu2_2, and `relu3_3` layers.
- c_i is the reciprocal of the number of components of $\phi_i(\cdot)$.
- I^{rgb} denotes the 3-channel image formed by dropping the alpha channel of image I.
- I^{aaa} denotes the 3-channel image formed by repeating the alpha channel of I three times.

We compute the perceptual loss as two L1 loss terms because the VGG16 network accepts an RGB image as input whereas the images outputted by the half-resolution rotator have 4 channels. To speed up the computation of $\Phi(\cdot, \cdot)$, we evaluate it stochastically. We flip a coin with head probability of 3/4. If it turns up head, we evaluate the term with I^{rgb}; otherwise, we evaluate the term with I^{aaa}. Of course, the terms are scaled with the reciprocal of the probability to make sure that the expectation has the right value.

Training is divided into two phases.

- In the first phase, only the L1 losses are used. In other words, $\ell_{\text{L1}} = 1$, and $\ell_{\text{percept}} = 0$. The first phase lasts for 1 epoch (500K training examples).
- In the second phase, all loss terms are used. In particular, we set $\ell_{\text{L1}} = 1$, and $\ell_{\text{percept}} = 0.2$. The second phase lasts for 18 epochs (9M training examples).

We used the Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate starts at 0 and linearly increases to 10^{-4} over the first 5,000 training examples. The batch size was 16.

2.2 Editor

Training has three phases. In the first two phases, the coarse input processing submodule is dropped from the editor, making it temporarily a "full-resolution rotator." The network is trained using the training process of the half-resolution rotator but now with the full resolution images instead of the half-resolution ones.

In the third phase, we added the coarse input processing submodule back and train the network to minimize the following loss:

$$\mathcal{L}_{\text{ED}} = \lambda_{\text{L1}}\Big(\mathcal{L}_{\text{L1}}^{\text{warped}} + \mathcal{L}_{\text{L1}}^{\text{direct}} + \mathcal{L}_{\text{L1}}^{\text{blended}}\Big) + \lambda_{\text{percept}}\Big(\mathcal{L}_{\text{half}}^{\text{direct}} + \mathcal{L}_{\text{half}}^{\text{blended}} + \mathcal{L}_{\text{quad}}^{\text{direct}} + \mathcal{L}_{\text{quad}}^{\text{blended}}\Big).$$

We fixed $\lambda_{\text{L1}} = 1$ and $\lambda_{\text{percept}} = 0.2$. The L1 losses are given by

$$\mathcal{L}_{\text{L1}}^{\square} = \|I_{\text{full}}^{\square} - I_{\text{full}}^{\text{posed}}\|_1$$

where $I_{\text{full}}^{\text{posed}}$ is the groundtruth posed image from the training dataset, and the $I_{\text{full}}^{\square}$ are the outputs of the editor as defined in Section 1.3. The losses $\mathcal{L}_{\text{half}}^{\text{direct}}$, $\mathcal{L}_{\text{half}}^{\text{blended}}$, $\mathcal{L}_{\text{quad}}^{\text{direct}}$, and $\mathcal{L}_{\text{quad}}^{\text{blended}}$ are perceptual losses. The superscripts indicate the outputs of the editor that we compute the losses with, and the subscripts indicate how the losses are computed. The "half" subscript indicates that the images are scaled down to 256×256:

$$\mathcal{L}_{\text{half}}^{\square} = \Phi\Big(\textsc{Down}(I_{\text{full}}^{\square}), \textsc{Down}(I_{\text{full}}^{\text{posed}})\Big)$$

where $\textsc{Down}(\cdot)$ denotes scaling a 512×512 image down to 256×256. The "quad" subscript indicates that the images are divided into four quadrants so that a 512×512 images becomes four $4 \times 256 \times 256$ images. The quadrants are then used to evaluate the perceptual losses.

$$\mathcal{L}_{\text{quad}}^{\square} = \sum_{i=1}^{4} \Phi\Big(Q_i(I_{\text{full}}^{\square}), Q_i(I_{\text{full}}^{\text{posed}})\Big)$$

where $Q_i(\cdot)$ extracts the ith quadrant from the argument. We found that evaluating the perceptual losses at 256×256 rather than 512×512 led to a network that produced sharper images.

Again, we use the same optimizers and learning rate schedule as those of the half-resolution rotator. The first phase lasts for 1 epoch (500K examples), the second 18 epochs (9M examples), and the third 18 epochs (9M examples).

3 Computers Used for Speed Measurements

We conducted experiments that measured time it took for the models to produce a single animation frame on the following 3 desktop computers.

- **Computer A** has two Nvidia Nvidia RTX A6000 GPUs, a 2.10 GHz Intel Xeon Silver 4310 CPU, and 128 GB of RAM. It represents a computer used primarily for machine learning research.
- **Computer B** has an Nvidia Titan RTX GPU, a 3.60 GHz Intel Core i9-9900KF CPU, and 64 GB of RAM. It represents a high-end gaming PC.
- **Computer C** has an Nvidia GeForce GTX 1080 Ti GPU, a 3.70 GHz Intel Core i7-8700K CPU, and 32 GB of RAM. It represents a typical (yet somewhat outdated) gaming PC.

4 Student Face Morpher's Loss Function

The student face morpher is trained to minimize the following loss:

$$\mathcal{L}_{\text{fm}} = E_{\mathbf{p}\sim p_{\text{pose}}}\Big[\big\|S^{\text{fm}}(I_{\text{in}}, \mathbf{p}) - T^{\text{fm}}(I_{\text{in}}, \mathbf{p})\big\|_1 + \lambda_{\text{fm}}\big\|M \odot (S^{\text{fm}}(I_{\text{in}}, \mathbf{p}) - T^{\text{fm}}(I_{\text{in}}, \mathbf{p}))\big\|_1\Big].$$

Here,

(a) Character face (b) Mask

Figure 3: Binary mask that covers movable facial parts of a character.

- I_{in} is the image of the character that we want to create a specialized student model of.
- $\mathbf{p}$ is a pose vector, which is sampled from the training dataset of the full system.
- p_{pose} is the uniform distribution over the poses in the training dataset.
- $S^{fm}(\cdot, \cdot)$ is the student face morpher.
- $T^{fm}(\cdot, \cdot)$ is the teacher face morpher, which comes from the full system in Section 3 of the paper.
- M is a binary mask that covers all the movable facial organs: eyebrows, eyes, mouth, and chin. This mask has to be created for each individual character. See Figure 3 for an example.
- Lastly, λ_{fm} is a weighting constant, whose value is 20 in all experiments.

References

[1] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[2] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *CoRR abs/2006.11239* (2020).

[3] Huang, X., and Belongie, S. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV* (2017).

[4] Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of European Conference on Computer Vision (ECCV)* (2016).

[5] Preechakul, K., Chatthee, N., Wizadwongsa, S., and Suwajanakorn, S. Diffusion autoencoders: Toward a meaningful and decodable representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[6] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)* (2015).