1. Extended UVCGAN Ablation Studies

This appendix shows the impact of the UVCGAN generator, gradient penalty (GP), and self-supervised generator pretraining (PT) on UVCGAN’s performance. Table 1 summarizes these findings. For each dataset, the bottom half of the table shows the UVCGAN performance with some of its components disabled. For example, UVCGAN no GP shows the UVCGAN performance without the gradient penalty term (but using a hybrid U-Net-ViT generator and a self-supervised pre-training). This table affords a few observations: 1. the addition of a hybrid U-Net-ViT generator alone typically produces a large degree of improvement compared to CycleGAN, even in the absence of the self-supervised pre-training and GP term; 2. the self-supervised generator pre-training without the GP term does not seem to improve the image-to-image translation performance and sometimes makes it worse; 3. the self-supervised pre-training only helps when it is used in conjunction with the GP.

Table 1. FID and KID scores. Lower is better. PT stands for the self-supervised generator pre-training, and GP means usage of the gradient penalty.

|                           | Selfie to Anime | Anime to Selfie |
|---------------------------|-----------------|-----------------|
| FID                       | KID (×100)      | KID (×100)      |
| ACL-GAN                   | 99.3            | 128.6           |
| Council-GAN               | 91.9            | 120.0           |
| CycleGAN                  | 92.1            | 127.5           |
| U-GAT-IT                  | 95.8            | 1108.8          |
| UVCGAN no GP              | 79.0            | 135 ± 0.20      |
| UVCGAN no PT              | 81.4            | 133.1           |
| UVCGAN no PT and GP       | 80.9            | 134.0           |
| Male to Female            |                 |                 |
| FID                       | KID (×100)      | KID (×100)      |
| ACL-GAN                   | 9.4             | 19.1            |
| Council-GAN               | 10.4            | 24.1            |
| CycleGAN                  | 15.2            | 22.2            |
| U-GAT-IT                  | 24.1            | 15.5            |
| UVCGAN                    | 9.6             | 13.9            |
| UVCGAN no GP              | 14.1            | 20.4            |
| UVCGAN no PT              | 11.0            | 14.7            |
| Male to Male              |                 |                 |
| FID                       | KID (×100)      | KID (×100)      |
| ACL-GAN                   | 16.7            | 20.1            |
| Council-GAN               | 37.2            | 19.5            |
| CycleGAN                  | 24.2            | 19.8            |
| U-GAT-IT                  | 23.3            | 19.0            |
| UVCGAN no GP              | 14.4            | 13.6            |
| UVCGAN no PT              | 15.8            | 14.3            |
| UVCGAN no PT and GP       | 19.7            | 16.1            |

2. Hyperparameter Tuning for Other Algorithms

This section summarizes the hyperparameter tuning results for three benchmarking algorithms: ACL-GAN, CycleGAN, and U-GAT-IT. We omit-
tuned tuning for Council-GAN because it takes too long to run (300 hours per translation).

Because none of the benchmarking algorithms use any stabilization techniques (such as the EMA of network weight [5]) beyond shrinking learning rate, we suspect the fluctuation may be at least partially due to instability of the GAN training.

We only provide hyperparameter tuning results for a data set or task if an algorithm did not work on it. We skip hyperparameter tuning if either a pre-trained model or a hyperparameter setup was provided by the author. In Table 2, the best results are marked in bold font. The default hyperparameters are highlighted in gray.

ACL-GAN worked on all three data sets studied and detailed in this paper—but all for only one direction: selfe-to-anime, male-to-female, and remove glasses. For the translation in the opposite directions, we tune three parameters concerning the focus loss: focus loss weight, focus upper, and focus lower. The results are summarized in Table 2.

| task            | weight | upper | lower | FID   | KID(×100) |
|-----------------|--------|-------|-------|-------|-----------|
| 3×anime-to-selfe | 0      | −     | −     | 18.6  | 3.49 ± 0.33 |
|                 | 0.25   | 0.5   | 0.3   | 205.3 | 11.0 ± 1.01 |
|                 | 0.25   | 1.0   | 0.05  | 250.3 | 18.6 ± 1.19 |
| 3×male-to-female| 0      | −     | −     | 46.0  | 3.39 ± 0.13 |
|                 | 0.25   | 0.5   | 0.3   | 19.1  | 1.38 ± 0.09 |
|                 | 0.05   | 0.5   | 0.2   | 3.63  | 2.94 ± 0.13 |
| 3×add glasses   | 0      | −     | −     | 29.0  | 1.77 ± 0.12 |
|                 | 0.25   | 0.4   | 0.05  | 20.1  | 1.35 ± 0.14 |

Table 2. ACL-GAN hyperparameter tuning results.

We tune three hyperparameters related to the focus loss: weight of the focus loss, focus upper, and focus lower.

CycleGAN did not work on any of the three data sets. We search a grid on two hyperparameters: type of generator (Gen.) and weight (Wt.) of cycle-consistency loss. We also try two GAN modes: IsGAN and wgangp. However, because CycleGAN did not implement GP properly, wgangp did not work. The results are summarized in Table 3.

In addition to hyperparameter tuning for U-GAT-IT, we also correct the aspect ratio problem of U-GAT-IT in this revised version as the original U-GAT-IT implementation cannot handle images with different height and width. We implement the rescaling in the preprocessing stage, so a

| gen.  | Wt. | FID   | KID(×100) | FID   | KID(×100) |
|-------|-----|-------|-----------|-------|-----------|
|       |     | selfe-to-anime | anime-to-selfe |
| ResNet | 5   | 92.1  | 2.72 ± 0.29 | 127.5 | 2.52 ± 0.34 |
|       | 10  | 93.4  | 2.96 ± 0.27 | 129.4 | 2.91 ± 0.39 |
| UNet  | 5   | 121.9 | 6.21 ± 0.32 | 134.3 | 2.96 ± 0.30 |
|       | 10  | 286.0 | 27.0 ± 0.87 | 155.8 | 3.32 ± 0.32 |
|       |     | male-to-female | female-to-male |
| ResNet | 5   | 21.9  | 2.00 ± 0.12 | 33.6  | 2.82 ± 0.14 |
|       | 10  | 15.2  | 1.29 ± 0.11 | 22.2  | 1.74 ± 0.11 |
| UNet  | 5   | 45.5  | 4.55 ± 0.17 | 50.8  | 4.86 ± 0.16 |
|       | 10  | 47.4  | 4.82 ± 0.19 | 47.5  | 4.57 ± 0.17 |
|       |     | remove glasses | add glasses |
| ResNet | 5   | 27.7  | 2.08 ± 0.16 | 26.0  | 1.77 ± 0.11 |
|       | 10  | 24.2  | 1.87 ± 0.17 | 19.8  | 1.36 ± 0.12 |
| UNet  | 5   | 32.2  | 2.52 ± 0.19 | 37.3  | 2.90 ± 0.14 |
|       | 10  | 32.2  | 2.52 ± 0.19 | 44.9  | 3.63 ± 0.20 |

Table 3. CycleGAN hyperparameter tuning results.

CelebA image of width 178 and height 218 is resized to have width 256 and height 313. As we did for CycleGAN and UVCiGAN, we take a random 256 × 256 crop from a training image and a central 256 × 256 crop from a test image.

U-GAT-IT studied the selfie-to-anime data set. For the two CelebA data sets, we try three levels of weight of cycle-consistency loss: (5, 10, and 20) and summarize the results in Table 4.

In addition to hyperparameter tuning for U-GAT-IT, we also correct the aspect ratio problem of U-GAT-IT in this revised version as the original U-GAT-IT implementation cannot handle images with different height and width. We implement the rescaling in the preprocessing stage, so a

| weight | FID   | KID(×100) | FID   | KID(×100) |
|--------|-------|-----------|-------|-----------|
|        |      | male-to-female | female-to-male |
| 5      | 39.2  | 3.86 ± 0.15 | 45.1  | 4.04 ± 0.16 |
| 10     | 24.1  | 2.20 ± 0.12 | 15.5  | 0.94 ± 0.07 |
| 20     | 32.1  | 3.09 ± 0.16 | 47.5  | 4.42 ± 0.17 |
|        |      | remove glasses | add glasses |
| 5      | 34.9  | 2.63 ± 0.15 | 50.0  | 5.08 ± 0.26 |
| 10     | 23.3  | 1.69 ± 0.14 | 19.0  | 1.08 ± 0.10 |
| 20     | 36.1  | 3.13 ± 0.19 | 36.1  | 2.67 ± 0.13 |

Table 4. U-GAT-IT hyperparameter tuning results.
3. More detail about the UNet-ViT Generator

A UNet-ViT generator consists of a UNet [6] with a pixel-wise Vision Transformer (ViT) [4] at the bottleneck (Figure 1). UNet’s encoding path extracts features from the input via four layers of convolution and downsampling. The features extracted at each layer are also passed to the corresponding layers of the decoding path via the skip connections, whereas the bottom-most features are passed to the pixel-wise ViT (Figure 2).

On UNet’s encoding path, the pre-processing layer turns an image into a tensor with dimension \((w_0, h_0, f_0)\). Each layer of the encoding path consists of a basic and downsampling block. The basic block is composed primarily of two convolutions, while the downsampling block has one convolution with stride 2. A pre-processed tensor will have its width and height halved at each downsampling block, while the feature dimension doubles at the last three downsampling blocks. Hence, the output from the encoding path will have dimension \((w, h, f) = (w_0/16, h_0/16, 8f_0)\), and it forms the input to the pixel-wise ViT bottleneck. Each layer of the UNet decoding path consists of an upsampling block followed by a basic block. A basic block on the decoding path differs from one on the encoding path in that it takes as input a concatenated tensor as input formed with the output from the upsampling layer and the tensor from the corresponding skip connection of the encoding path. The decoding path’s output will go through a post-processing layer of 1 \(\times\) 1 convolution with a sigmoid activation to produce an image.

A pixel-wise ViT is composed primarily of a stack of Transformer encoder blocks [3]. To construct an input to the stack, the ViT first flattens an encoding along the spatial dimensions to form a sequence of transformer tokens. The token sequence has length \(w \times h\), and each token in the sequence is a vector of length \(f\). It then concatenates each token with its two-dimensional Fourier positional embedding [1] of dimension \(f_p\) and linearly maps the result to have dimension \(f_v\). To improve the Transformer convergence, we adopt the rezero regularization [2] scheme and introduce a trainable scaling parameter \(\alpha\) that modulates the magnitudes of the nontrivial branches of the residual blocks. The Transformer stack output is linearly projected back to have dimension \(f\) and unflattened to have width \(w\) and \(h\). In this study, we use input raw or cropped images with \(w_0 = h_0 = 256\) and set \(f_0 = 48\). Hence, we have \(w = h = 16\) and \(f = 384\). We use 12 Transform encoder blocks in ViT and set \(f_p = f_v = f\), and \(h = 4f_v\) for the feed-forward network in each transformer encoder block.

4. Additional Sample Translations

We show a few more sample translations in Figures 3 to 5.

References

[1] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lemitsky, and Denis Korzhankov. Image generators with conditionally-independent pixel synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14278–14287, 2021.

[2] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In Uncertainty in Artificial Intelligence, pages 1352–1361. PMLR, 2021.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[5] 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.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference
Figure 1. UNet ViT Generator with Full Details

Figure 2. Vision Transformer with Full Details

on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.
| Input  | ACL-GAN | Council-GAN | CycleGAN | U-GAT-IT | UVCGAN |
|--------|---------|-------------|----------|----------|--------|
| ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |

Figure 3. Additional Sample Translations: Selfie2Anime
|                | Input | ACL-GAN | Council-GAN | CycleGAN | U-GAT-IT | UVCGAN |
|----------------|-------|---------|-------------|----------|----------|--------|
| **male to female** | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| **female to male**  | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |

Figure 4. Additional Sample Translations: GenderSwap
Figure 5. Additional Sample Translations: Eyeglasses