Diagonal Attention and Style-based GAN for Content-Style Disentanglement in Image Generation and Translation : Supplementary Materials

Gihyun Kwon[1] Jong Chul Ye[1,2]
Department of Bio and Brain Engineering[1], Graduate School of AI[2], KAIST
{cyclomon,jong.ye}@kaist.ac.kr

1. Diagonal GAN Experiments

1.1. Qualitative Experiment Settting

For qualitative evaluation in the main text, our models were trained using the images from 1024×1024 CelebA-HQ [6] and 512×512 AFHQ [3].

Specifically, using full-resolution CelebA-HQ images, we trained our model by accessing 20 million training samples. For efficient training with limited GPU capacity, we started with batch size of 512 in the first 8×8 resolution, and reduced the batch size by half each time when we proceeded to a larger image size. With the aforementioned training strategy, the overall training took about one month using a single Tesla V100 GPU. For learning rate, we used 0.001 until accessing 12 million samples, then decreased the learning rate to 0.0001. To test the effect of our diagonal attention (DAT) module in the qualitative evaluation, we removed per-pixel noises at each layer. We increased the λ value of DS loss from 0.3 to 0.5 after accessing 12 million samples for the training. We used the left and right flips for data augmentation in all training procedures. For the case of full resolution AFHQ images, we used the same training settings as we did before for the CelebA-HQ dataset case, except that we used the fixed value of λ as 0.3 throughout the training.

For further qualitative evaluations, we carried out experiments with additional image data sets: Oxford Flowers 102 [9], Caltech-UCSD Birds (CUB2011) [11], and Stanford Cars [8]. For the flower data set, we first extracted the flower regions with center cropping. Then we resized the cropped images to 512×512. For the bird dataset, we extracted bird image areas using bounding box information. Then we changed the size of the extracted images to 256×256. For the car dataset, we also extracted the car image areas using bounding box information, then resized the cropped images to 384×512. We also used the left-right flips for data augmentation.

When training the models with flower and car data sets, we continued the training models with up to 12 million samples. For the bird dataset, training was continued until we accessed 10 million samples. The training settings used for the AFHQ model training was also used for the flower, car, and bird datasets.

To improve the perceptual quality, the images are generated by applying truncation trick similar to [2]. More specifically, we found that best perceptual image quality was obtained by truncating mapped style code in $\boldsymbol{W}_s$ up to 0.7, whereas no truncation was used for $\boldsymbol{W}_c$.

1.1.1 Additional Qualitative Experiment Results

AFHQ results: Fig. 3 shows the results of direct attention map manipulation of our full-resolution AFHQ model. Similar to our CelebA-HQ results, we could obtain the faces with the desired direction by manipulating the 4×4 map, and control the mouth opening by changing the values around mouth in 8×8 map. We also show the results of the interpolation of the content codes of our AFHQ model in Figs. 4, 5 and 6. When the content codes at all levels are changed, the global spatial attributes are changed, and when the 8×8 maps are changed, the lower parts of the areas change. Quantitatively, our model trained with AFHQ in full resolution achieved 10.79 in FID.

CelebA-HQ results: We also show more results of the content code interpolation with the model trained with full-resolution CelebA-HQ in Figure 7 and 8. When the content codes for all layers are changed, the global spatial attributes are changed, and when the first 4×4 codes are changed, face direction changes. When the 8×8 codes are changed, lower parts of faces change.

Comparison with editing methods: To show the advantages of our spatial attention map editing, we show the comparative experiments with the existing face editing method in Figure 9. Among various editing methods, we compared our model with GANspace [4], which controls the generated images in unsupervised way by finding principal components of style code space. For fair comparison, we brought CelebA-HQ editing results from the original script of GANspace.

Since GANspace changes the principal component in the

style code space, this often leads to the remaining entanglement between content and style attributes. In contrast, more detailed editing is possible since our model has disentangled style and content code spaces.

Flower results: To test the versatility of our proposed model, we trained our model using additional datasets. Fig. 10 illustrates the generated images from the model trained on flower dataset. When we change the content code with fixed style codes, we can change the spatial information such as flower shape, number, and location of flowers. On the other hand, if we vary style codes with fixed content codes, we can observe the changes of the global style attributes including species, flower color and background. Fig. 11 also shows the results by changing the content smoothly with interpolating the content codes. Our flower model scored 46.43 in FID.

Birds results: Fig. 12 shows the generated images from the model trained on birds dataset. Fig. 12 (b) shows samples with varying style codes and the fixed content code. We can observe the changes of the global style attributes including species, feather colors and patterns. Fig. 12 (c) illustrate samples generated with varying content codes and the fixed style. We can observe that the content attributes including location, rotation and global shapes change with different content codes. Fig. 13 also shows the content interpolation results, which shows that birds smoothly change the head orientation. Our model on birds dataset scored 14.27 in FID.

Cars results: Figure 14(a) shows the sample images from random content and style code. Then, Figure 14(b) shows the samples with varying style codes and the fixed content code. We can observe the changes of the global style attributes including car type, colors and background. Samples generated with varying content codes and the fixed style are provided in Figure 14(c). We can observe that the content attributes including rotation and global shapes change with different content codes. In Fig. 15, we can see the effect of content interpolation in terms of rotation angle. Our model on car dataset scored 8.96 in FID.

1.2. Quantitative Experiments

For quantitative evaluation, we compared our method with the baseline SNI model, SNI with DS loss, and the original StyleGAN. In order to carry out extensive comparative studies with various models, we trained the models with the reduced resolution of 256×256 using 500,000 iterations (total of $\sim$4.7 million samples). We also used batch-size scheduling for efficient training. It took four days for training each model with a single NVIDIA RTX2080Ti GPU. For a fair comparison, we used the same non-saturating loss with R_1 regularization in all the experiments. The same settings are also used in all models for our ablation studies and additional disentanglement studies.

For training the baseline SNI model and SNI with DS loss, we implemented the models on PyTorch based on official source code[1]. To follow the best settings in the original paper [1], we used the input tensor of 8×8 resolution for training the models. For the DS loss, λ is set to 0.3, which shows the best results.

In order to quantitatively evaluate the image quality of the generated samples, we calculated the FID values [5]. For CelebA-HQ with 30,000 training images, we computed the FID values with 50,000 generated samples. For the AFHQ and other data sets with relatively fewer training images, we calculated the FID values with 20,000 generated samples.

To calculate the total PPL of the W space, we follow the same calculation proposed in StyleGAN[7]. If we sample the two style codes $s_1, s_2 \in W_s$ and two contents codes $c_1, c_2 \in W_c$, the PPL score is calculated as

$$\begin{aligned}&\text{PPL}_W\\ =&\frac{1}{\epsilon^2}\mathbb{E}\left[d(G(ts_1+(1-t)s_2, tc_1+(1-t)c_2)),\right.\\ &\left.G((t+\epsilon)s_1+(1-t-\epsilon)s_2, (t+\epsilon)c_1+(1-t-\epsilon)c_2))\right]\end{aligned}$$

where t is a uniformly sampled between $[0, 1]$, and $G(s, c)$ is the generator output with respect to the style code s and content code c, respectively, and $d(X, Y)$ denotes the perceptual distance between two images X and Y. We use $\epsilon = 10^{-4}$ for all the calculations and report the average values that are computed using 10,000 generated samples.

For calculation of PPL for W_s, we use fixed content code $c_{fix} \in W_c$ and paired style codes $s_1, s_2 \in W_s$:

$$\begin{aligned}&\text{PPL}_{W_s}\\ =&\frac{1}{\epsilon^2}\mathbb{E}\left[d(G(ts_1+(1-t)s_2, c_{fix})),\right.\\ &\left.G((t+\epsilon)s_1+(1-t-\epsilon)s_2, c_{fix})\right]\end{aligned}$$

To calculate PPL_{W_c}, we use fixed style code $s_{fix} \in W_s$ and sampled content codes $c_1, c_2 \in W_c$:

$$\begin{aligned}&\text{PPL}_{W_c}\\ =&\frac{1}{\epsilon^2}\mathbb{E}\left[d(G(s_{fix}, tc_1+(1-t)c_2),\right.\\ &\left.G(s_{fix}, (t+\epsilon)c_1+(1-t-\epsilon)c_2)\right]\end{aligned}$$

For the computation of PPL for W_s (resp. W_c), for each fixed code, the content codes (resp. style codes) are sampled fifty times, and the average value was calculated by repeating this 200 times. Therefore, the final PPL value is calculated using 10,000 samples.

1.3. Ablation studies

In our ablation study, we compared the quantitative performance of models trained with different settings. In all

[1] https://github.com/yalharbi/StructuredNoiseInjection

Varying λ	CelebA-HQ		AFHQ	
	FID	W PPL	FID	W PPL
0.2	11.65	49.83	13.27	70.78
0.3	**10.90**	**48.12**	**11.73**	**63.44**
0.4	11.69	62.44	13.32	80.41
0.5	11.51	68.75	13.87	73.12

Table 1: Quantitative results of ablation study. We investigate the effect of λ value in the diversity-sensitive loss.

DAT network	CelebA-HQ		AFHQ	
	FID	W PPL	FID	W PPL
2×MLP-256	14.51	53.99	16.85	70.02
CNN-256	11.18	83.37	14.28	91.88
single MLP-32	11.74	60.87	13.26	118.21
single MLP-64	**10.66**	61.98	12.50	87.89
single MLP-256	10.90	**48.12**	**11.73**	**63.44**

Table 2: Quantitative results of ablation study. We compare different attention mapping networks and the maximum resolution for the DAT layers.

of the experiments, we used models trained with per-pixel noises.

In order to validate the choice of the value of λ for the diversity-sensitive loss, we first show the results of various models that were trained with different λ. In Table 1, the models trained with $\lambda = 0.3$ show the best performance in the disentanglement capability, exhibiting the lowest PPL scores; furthermore, they showed the best image quality with the lowest FID. The models trained with lower or higher λ values show degraded disentanglement performance. The results show that we can achieve the most balanced content-style control with both codes when we set $\lambda = 0.3$.

We also investigated the effect of different network architectures for the attention mapping, and show the results in Table 2. To ensure stability in training, we use attention mapping with a single layer MLP followed by a sigmoid applied to layers with a resolution of up to 256×256. To verify our choice of mapping network architecture, we implemented two additional networks for ablation study: 2×MLP-256 and CNN-256. Here, 2×MLP-256 represents a model which has attention mapping of 2-layer MLP instead of a single MLP. The model CNN-256 uses CNN layer-wise upsampling network to generate the diagonal attention. In all the experiments, we fixed $\lambda = 0.3$ and used mapping network up to 256×256 layers.

Table 2 shows that when using 2-layer MLP, we can obtain a well-disentangled model with relatively low PPL scores, but it still cannot achieve the best performance. In case of using CNN as an attention network, the disentanglement scores are severely degraded, which may be due to the imbalance between the simple AdaIN network and CNN-based attention mapping networks.

Dataset	Methods	Per-pixel Noise			w/o Per-pixel Noise		
		content	style	all	content	style	all
CelebA-HQ	SNI [1]	0.338	0.499	0.532	0.348	0.496	0.534
	SNI+DS	0.392	0.472	0.531	0.378	0.489	0.535
	Ours	0.411	0.469	0.532	0.413	0.472	0.533
AFHQ	SNI [1]	0.344	0.588	0.605	0.411	0.583	0.610
	SNI+DS	0.399	0.585	0.607	0.407	0.589	0.608
	Ours	0.412	0.582	0.607	0.443	0.578	0.608

Table 3: Comparison of LPIPS scores of models trained using CelebA-HQ and AFHQ datasets at 256×256 resolution. Our model has more balanced content-style control than that of the baseline models.

Then, we carried out comparative study by changing the maximum resolution of the diagonal attention (DAT) layer. In Table 2, the use of DAT up to 32×32 (single MLP-32) and 64×64 (single MLP-64) have relatively high PPL values, suggesting that DAT layers at the lower levels only result in a limited expressiveness for the various content information. The disentanglement quality is particularly impaired in the models trained with AFHQ data set. We suspect that limited capacity in content control makes it more difficult to cover the variations of images in AFHQ that are more diverse than those in CelebA-HQ. Therefore, we use DAT layers up to 256×256 resolution (single MLP-256), which is our default model.

In evaluating image quality in terms of FID scores, our default model showed better performance than most of baseline settings except for single MLP-64. However, in the case of single MLP-64, the disentanglement performance is relatively poor. Therefore, we can obtain best result when using single MLP-256.

1.4. Disentanglement Experiments

To further quantify the disentanglement performance, we additionally measure the content and style diversity in the image generation. As a measurement of image diversity, we use Learned Perceptual Image Patch Similarity (LPIPS) [12]. Since our model and the baseline SNI have two independent content and style codes, we can compare their diversity of style and content. To measure the diversity score of both codes, we compute the average value of LPIPS of 40,000 images sampled with arbitrary content and style codes. On the other hand, to measure the style and content diversity separately, we calculate the LPIPS of 40 images sampled by varying one code with another code fixed, which is repeated for 10,000 times to calculate the averaged LPIPS. This makes the the total number of images for LPIPS calculation equal to 40,000 for both cases.

Table 3 is the result of LPIPS scores. When we compare

the LPIPS by varying both content and style codes, both SNI and our model show similar diversity in the generated images. However, when looking at the diversity of style and content separately, the baseline SNI shows that the diversity of content is much lower than that of the style. On the other hand, our model has more balanced diversity in style and content. With SNI trained with DS loss on content code, the model shows slightly better diversity in content code than that of the baseline SNI. However, the model still shows lower content diversity than our model as it is not able to overcome the capacity limit of content control with input tensor.

To support the above quantitative comparison in terms of LPIPS scores, we also qualitatively compared the effect of various content controlling methods: per-pixel noises of StyleGAN, input tensor of SNI and our DAT mapping. Since the code spaces of the baseline models are slightly different, we compare the images generated from the mean style code by varying the content codes. In Figure 16, we can see that StyleGAN with different per-pixel noises only results in the minor spatial variations such as curls of hair and fur. As for the baseline SNI, it can only change the simple geometrical information such as rotation. For SNI trained with DS loss, we can see that the generated samples have more diversity than basic SNI, but still the variation is limited to geometry similar to basic SNI. In contrast, our model allows more diverse changes on spatial information including geometry, hairstyle (fur pattern), facial expression, etc., by controlling specific DAT layers.

2. Diagonal GAN Inversion

2.1. Methods

As discussed in the main text, our diagonal GAN can be easily incorporated with GAN inversion. Specifically, our inversion model consists of two steps as shown in Figure 1. The details of each step are as follows.

Step 1: The first step is to train our proposed Diagonal GAN. For domain-aware (i.e females, males) image generation, we train a multi-domain Diagonal GAN in which the style mapping network SM can sample multiple style codes with a multi-head structure. Specifically, we use two types of style codes that represent males and females domains (see Fig. 1(a)). On the other hand, the content mapping generates a unified content code $c = CM(z_c)$ that can be used for both style domains. We used mapped style codes $s \in W_s$ which have the dimension of 512, and mapped content codes $c \in W_I$ with the dimension of 512. Our discriminator D also has a multi-head structure to simultaneously enable realistic generation and domain classification.

Step 2: After pre-training our generator network (Fig. 1(a)), we invert the real images into latent spaces with our inversion network in Step 2 (Figs. 1(b)(c)). In this step, we

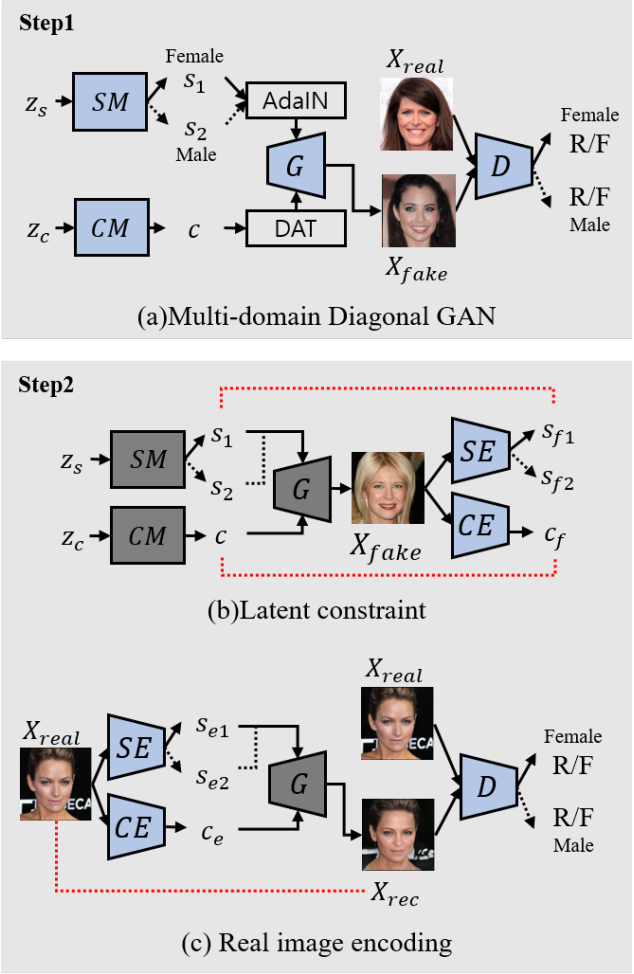

Figure 1: Detailed training procedure for GAN inversion task. Step 1: Training our Diagonal GAN with multi-domain style codes. Step 2: With pre-trained Diagonal GAN in Step 1, we introduce style encoder SE and content encoder CE to find style and content codes for real input images. The networks with gray color are fixed, and with blue color are trainable. Red dotted lines indicate supervision losses.

use the modified version of state-of-the-art GAN inversion method IDinvert [13]. To encode the real images into style and content code spaces, we introduced the style encoder SE and the content encoder CE networks. Similar to the style mapping in Step 1, our style encoder has a multi-head structure after the last convolution layer.

The main idea of IDInvert is that when encoding a real image into a latent space, realistic reconstruction is possible only when the encoded latent code is constrained within the learned latent space. To achieve this goal, as shown in Fig. 1(b), we first sampled the random style code s with random domain label y, and the random content code c using

the pre-trained mapping networks SM and CM, and generated a fake image X_{fake}. Then by putting the generated X_{fake} into the encoders, we can obtain the encoded style and content codes s_f and c_f, respectively. Then, our loss for latent codes is given by

$$L_{latent} = ||s - s_f||_2 + ||c - c_f||_2 \tag{1}$$

which reduces the mean-squared error (MSE) between the encoded codes and the learned codes so that our encoder networks can generate the codes within the learned latent spaces.

Additionally, in Fig. 1(c), we put the real image X_{real} with the corresponding domain label $\hat{y}$ into the style and content encoders to get the content code c_e and style code s_e, respectively. Then, the codes c_e and s_e are used in DAT and AdaIN layers, respectively, of the pre-trained generator to obtain the reconstructed image X_{rec}. The goal of this step is to make X_{rec} as close as possible to X_{real}. For realistic reconstruction, we reduce the distance between X_{real} and X_{rec} by using a MSE loss, a LPIPS [12] loss that reduces the perceptual distance, and an adversarial loss using a new discriminator D which also has a multi-head structure. For adversarial loss, we used the same loss function as StyleGAN [7], which is composed of the non-saturating Softplus, $f(t) = \text{softplus}(t) = \log(1 + \exp(t))$, with R_1 regularization.

Accordingly, our total loss function for the content and style encoder is given by

$$\begin{aligned} L_E =& ||X_{real} - X_{rec}||_2 + LPIPS(X_{real}, X_{rec}) \\ &+ \lambda_{lat} L_{latent} + \lambda_{adv} f(-D_{\hat{y}}(X_{rec})) \end{aligned}$$

where λ_{lat} and λ_{adv} are weight parameters. On the other hand, the loss for the discriminator is

$$\begin{aligned} L_D =& f(D_{\hat{y}}(X_{rec})) + f(-D_{\hat{y}}(X_{real})) \\ &+ \frac{\gamma}{2}\mathbb{E}[||\nabla D_{\hat{y}}(X_{real})||_2^2] \end{aligned}$$

where $D_{\hat{y}}(\cdot)$ denotes the output of the discriminator D corresponding to the domain $\hat{y}$, γ is a weight parameter for gradient R_1 regularization (the last term in L_D).

Inference time Latent-Regularized Optimization: Additionally, we try to find better latent codes at the test time by additionally optimizing the latent code for better reconstruction. In this process, we use the latent optimization method proposed by IDinvert [13]. Specifically, as an initialization for the style and content codes s and c, respectively, we use codes s_e and c_e from pre-trained encoders in Step 2. In addition to reducing the distance between the input and reconstruction, we include latent regularization loss to make the latent vectors s,c lie within the learned space of the encoders and the generator. The resulting loss function for optimization is:

$$\begin{aligned} L_{code}(s, c) =& ||X_{real} - G(s, c)||_2 + LPIPS(X_{real}, G(s, c)) \\ &+ \lambda_{reg}||s - SE_{\hat{y}}(G(s, c))||_2 \\ &+ \lambda_{reg}||c - CE(G(s, c))||_2 \end{aligned} \tag{2}$$

where $G(s, c)$ denotes the generator output with style and content codes s and c, respectively, $SE_{\hat{y}}$ refers to the style encoder on the domain $\hat{y}$, and CE is the content encoder.

2.2. Method Details

In GAN inversion experiments, we used 256×256 resolution CelebA-HQ dataset. Total of 30,000 images, 28,000 are used as a training set, and 2,000 are used as a test set. The test set consists of 1000 male and 1000 female face images.

When training our Diagonal GAN network in Step 1, we trained the model until we access a total of 10 million training samples, which took about a week with a single RTX-2080Ti GPU. Except for the maximum resolution, other training settings are the same as our full-resolution CelebA-HQ experiments.

Our style encoder model is a CNN with multi-head fully-connected layers, which has the same structure as the discriminator. The content encoder has the same architecture, except that it has a single-head structure. In Step 2 training, we trained the model with the batch size of 2 for 200,000 iterations. We used Adam optimizer, initially using a learning rate of 0.001, and then decreased the learning rate to 0.0001 after 100,000 iterations. For weight parameters, we set $\lambda_{lat} = 1$, $\lambda_{adv} = 0.1$. This took 2 days with a single NVIDIA RTX2080Ti GPU.

For our inference time latent-regularized optimization using (2), both style and content latent codes were optimized using 100 iterations per a single input image. We used Adam optimizer with learning rate of 0.01, and set the loss weights as $\lambda_{reg} = 2$. Optimization process took 4 seconds per each input image.

2.3. Experiment Details

In order to show the superior disentanglement performance of our model, we compared our method with state-of-the-art diverse image translation model, StarGANv2 [3]. Note that our inverted model can control both content and style spaces using DAT and AdaIN layers, whereas StarGANv2 can only convert styles of input images due to the exclusive use of AdaIN layers. For a fair comparison, we used the pre-trained StarGANv2 that can be downloaded from the official GitHub repository [2].

[2] https://github.com/clovaai/stargan-v2

For quantitative evaluation, we measured the quality in terms of FID and diversity through LPIPS. Since StarGANv2 can only convert the style of the images, we only consider the style conversion by two methods for this quantitative comparison. We consider both image translation scenario: 1) latent-based image translation, which converts the style of input image to a random style by sampling the style codes, and 2) reference-based image translation, in which we convert the style of inputs to that of the reference images. At this time, we measured the performance by converting a single image of one domain into 10 different target domain images. In our GAN inversion, the experiment was conducted by varying the style codes while using the same content code of the input image. As mentioned before, this is to compare the image quality during style translation, as StarGANv2 is only for the style translation. Since the test set contains 1,000 images for each domain (female, male) and 10 target styles are used for each image, 10,000 synthesized images can be obtained. Furthermore, we consider the domain conversion scenario (i.e. females $\leftrightarrows$ males), which doubles the number of synthesized images. Therefore, for each latent and reference based experiment, we measured metrics on 20,000 generated images. For more details, please refer to the original StarGANv2 paper [3], as we use the same evaluation process.

In all the experiments, we used style and content codes obtained using inference time latent-regularized optimization process, except for the qualitative experiment of reference-based style synthesis, where style codes without latent-regularized optimization still provide better perceptual quality.

2.4. Inversion Experimental Results

2.4.1 Multi-domain Diagonal GAN Inversion

Auto-encoder Reconstruction Results: Our inversion results on multi-domain Diagonal GAN showed satisfactory reconstruction performance by extending the state-of-the-art inversion model. To evaluate the reconstruction performance, we measured the distance between input and the reconstructed image with MSE and LPIPS. When we reconstruct the images without inference time latent-regularized optimization, we could obtain MSE of 0.095 and LPIPS of 0.246. Furthermore, with additional latent-regularized optimization process, the model showed improved performance with MSE of 0.042 and LPIPS of 0.155. The results confirmed that our model shows good reconstruction performance, and more accurate reconstruction is possible when latent-regularized optimization is additionally used.

Qualitative comparison: In the main script, we have already compared the performance of our model and baseline StarGANv2 to show that our model outperforms the generation quality. Here, we provide more extensive qualitative comparison results to highlight the advantages of our model.

Figure 17 shows the generated samples synthesized from input image to follow the styles of reference images. Although StarGANv2 shows good performance in style synthesis from typical images (see Fig. 17(a)), it is still a conventional image translation model that uses spatial information of the image as it is. Therefore, as shown in Fig. 17(b), when the content information of the input is complicated or rare, we can observe that the generation performance is often severely degraded. In contrast, our model finds the content code that can best express the input content in the pre-trained space, so that it can generate realistic images even with complex or rare input contents (see Fig. 17(b)). Figs. 18(a)(b) show the result of converting the input image to follow random styles. Again, our model could generate more realistic images even if the input content is complex (see (see Fig. 18(b)).

To clearly show the strength of our model, in Figure 19, we show the results from our image translation results with content manipulation through the content code interpolation. As explained before, StarGANv2 can change the contents only by using different input images similar to Figs. 19(a)(c). In contrast, as our model can control the content code space, Figure 19(b) shows that the content information of the translated images can change smoothly with interpolating the content codes. Furthermore, Figure 20 shows the image translation experiment by changing the hierarchical content code in addition to style synthesis. Unlike StarGANv2, which can only change the style of the input, we can further change the the specific content attributes such as: face geometry by changing the 4×4 content codes, the hair shape by changing the 8×8 codes, and the mouth expression by changing the 16×16 codes.

The results clearly show that our model is capable of flexible content control in addition to the style control, which is not possible with the existing image translation models such as StarGANv2.

2.4.2 GAN Inversion comparison results

Results on another inversion method: We also show that our proposed model can be easily integrated to various GAN inversion methods by incorporating our DAT in another state-of-the-art GAN inversion method, pSp [10]. In contrast to the original pSp which uses single encoder that predicts multiple style codes, we trained two pSp encoders that predict style and content codes, respectively. We also used the identity matching loss proposed by pSp. In Table 4 and Fig 2, again we could obtain good reconstruction results similar to those of IDInvert. The results confirm that our model can be easily integrated to various GAN inversion frameworks.

Comparison with baselines: To further verify the inver-

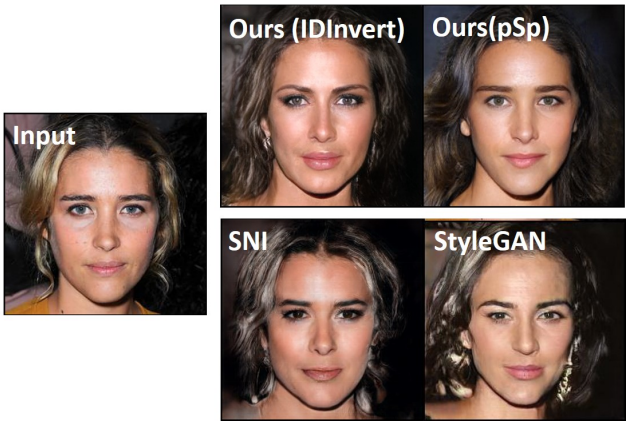

Figure 2: Qualitative comparison of various inversion experiments.

Methods	MSE↓	LPIPS↓
StyleGAN(IDInvert)	0.120	0.314
SNI(IDInvert)	0.095	0.286
Ours(pSp)	**0.074**	0.267
Ours(IDInvert)	0.081	**0.263**

Table 4: GAN inversion comparison in terms of MSE and LPIPS scores between input and reconstructed images. All models are trained on 28,000 CelebA-HQ training images. The scores are calculated on 2,000 images in CelebA-HQ validation set. Compared to other baseline models, inversion results on our proposed model shows the best performance.

sion performance of our proposed model, we show comparison experiment on inverting various baseline models. For fair comparison, we applied IDInvert to the models trained on 256×256 CelebA-HQ. In Table 4 and Fig 2, our inverted model can obtain the most accurate reconstructions by finding appropriate style and content codes. In case of baseline models of SNI and StyleGAN, the performance is degraded because the encoder failed to find right codes due to the highly entangled code spaces. The results show that our model outperforms the baseline models due to the further disentangled code spaces.

References

[1] Yazeed Alharbi and Peter Wonka. Disentangled image generation through structured noise injection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5134–5142, 2020. 2, 3

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 1

[3] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8188–8197, 2020. 1, 5, 6

[4] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9841–9850. Curran Associates, Inc., 2020. 1

[5] 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 *Adv. Neural Inform. Process. Syst.*, pages 6626–6637, 2017. 2

[6] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *Int. Conf. Learn. Represent.*, 2018. 1

[7] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4401–4410, 2019. 2, 5

[8] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 1

[9] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 1

[10] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: A stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2287–2296, June 2021. 6

[11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 1

[12] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 586–595, 2018. 3, 5

[13] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020. 4, 5

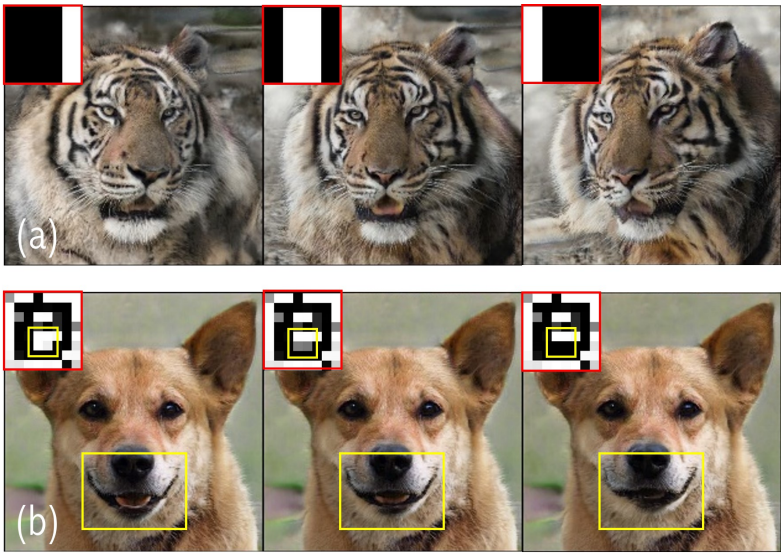

Figure 3: Direct attention map manipulation. By controlling the specific areas of attention, we can selectively change the animal facial attributes. Results from changing (a) the first 4×4 attention map, (b) the 2nd 8×8 attention map. We can manipulate the face direction with changing first 4×4 map, and change mouth expression with 2nd 8×8 map. The yellow boxes represent the edited areas.

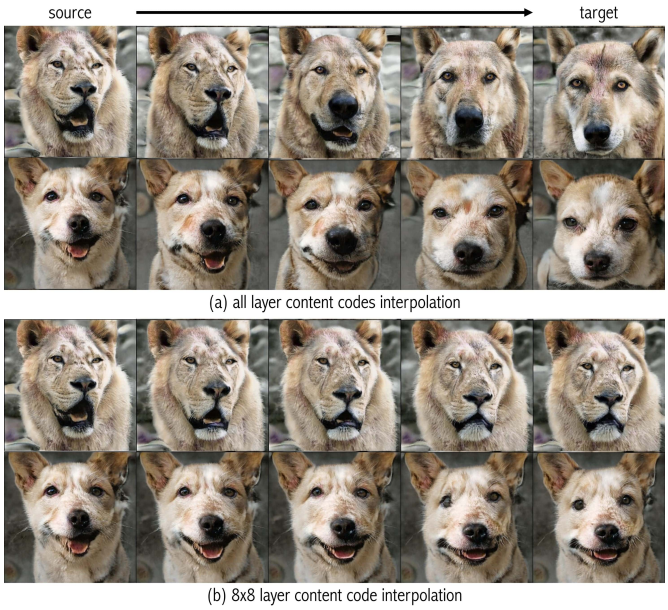

Figure 4: AFHQ results for content code interpolation. (a) Results with interpolating content codes of all layers. (b) Interpolating 8×8 layer content codes.

Figure 5: Examples from interpolated content codes. The images of each row are sampled from the same content code. (Left) Samples by varying content codes across all layers. We can observe that the global attributes such as rotation, and shape change gradually. (Right) Samples by varying 8×8 attention maps. We can observe that this layer mainly contributes on the lower part of faces (i.e mouth).

Figure 6: Examples from interpolated content codes. All images are samples by varying the attention maps across all layers. All samples show that we can control content information independently from the styles.

Figure 7: Examples from interpolated content codes. The images at each row are sampled from the same content code. (Left) Samples by varying the content codes across entire layers. We can observe that the global attributes including rotation, facial expression, and identity changes gradually. (Right) Samples with changing 8×8 resolution content codes. We can see the lower part of faces (i.e mouth) changes.

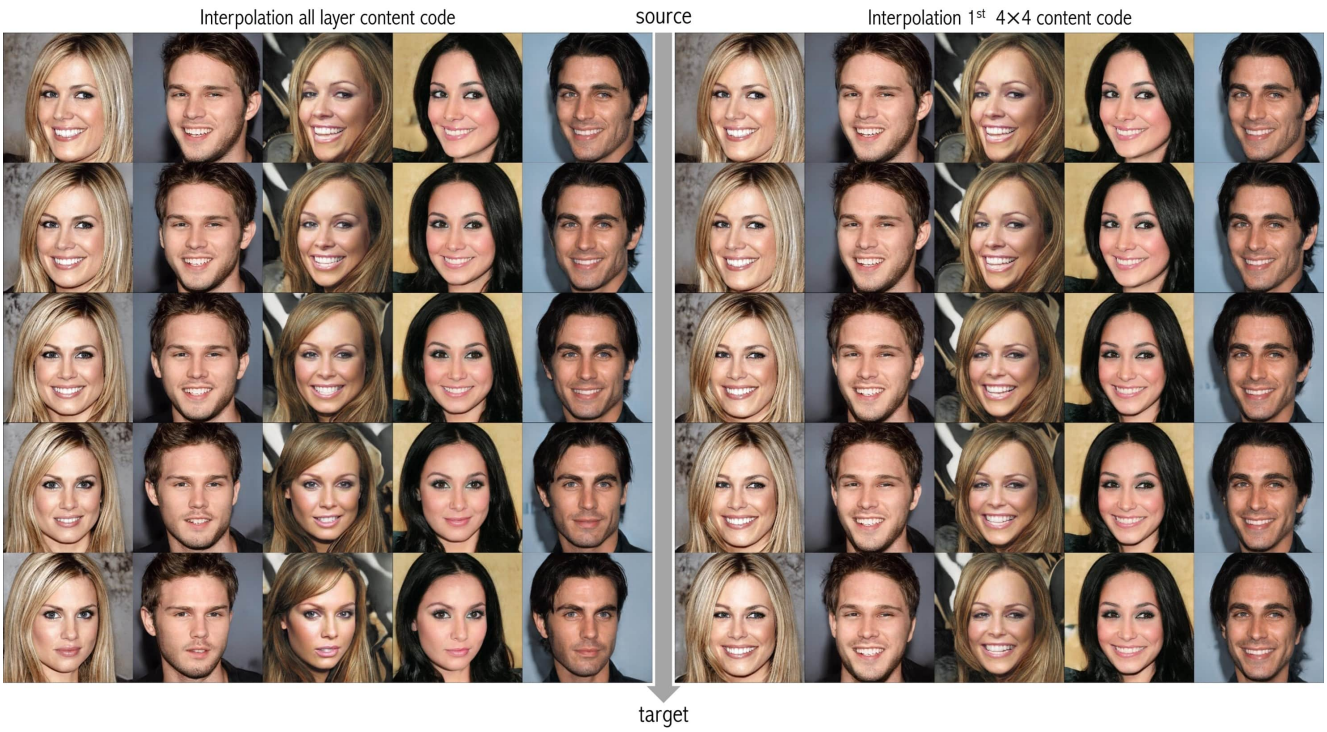

Figure 8: Examples from interpolated content codes. The images of each row are sampled from same content code. (Left) samples with changing content code across all layers. (Right) Samples by varying the first 4×4 resolution content code. We can see the global geometry (i.e rotation) changes.

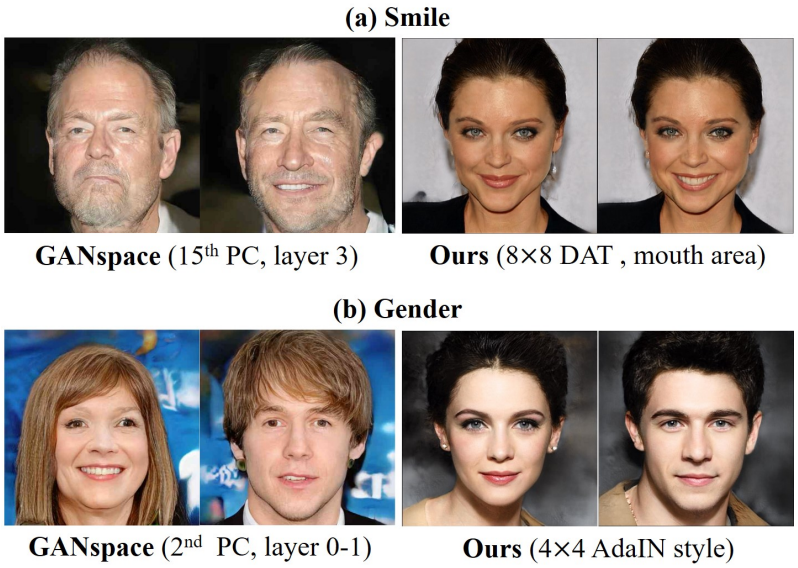

Figure 9: Comparison between our model and GANspace. (a) In controlling the mouth expression of the generated face, GANspace changes not only the targeted mouth part, but also other attributes such as nose and ears. In our method, only the mouth area can be controlled without changing other attributes by editing the attention map of the mouth area. (b) In the gender translation experiment which requires changing the overall style of the face, GANspace changes the direction or shape of the face which are irrelevant to the target attribute. However, since content and style are separated in our model, changing only the style code of low resolution can change the gender while minimizing the change of other attributes. The results show that our model is capable of more detailed attribute editing than the existing editing method.

Figure 10: Generated 512×512 images by our method trained using flower data set. (a) A source image generated from arbitrary style and content code. (b) Samples with varying style codes and the fixed content code. We can observe the changes of the global style attributes including species, flower color and background. (c) Samples generated with varying content codes and the fixed style. We can observe that the content attributes including location, global shape, and the number of flowers change with different content codes. (d) Samples generated with both varying content and style codes.

Figure 11: Examples from interpolated content codes. The images of each row are sampled from the same content code. All images are samples with changing content codes across all layers.

Figure 12: Generated 256×256 images by our method trained using bird data set. (a) A source image generated from arbitrary style and content code. (b) Samples with varying style codes and the fixed content code. We can observe the changes of the global style attributes including species, feather colors and patterns. (c) Samples generated with varying content codes and the fixed style. We can observe that the content attributes including location, rotation and global shapes change with different content codes. (d) Samples generated with both varying content and style codes.

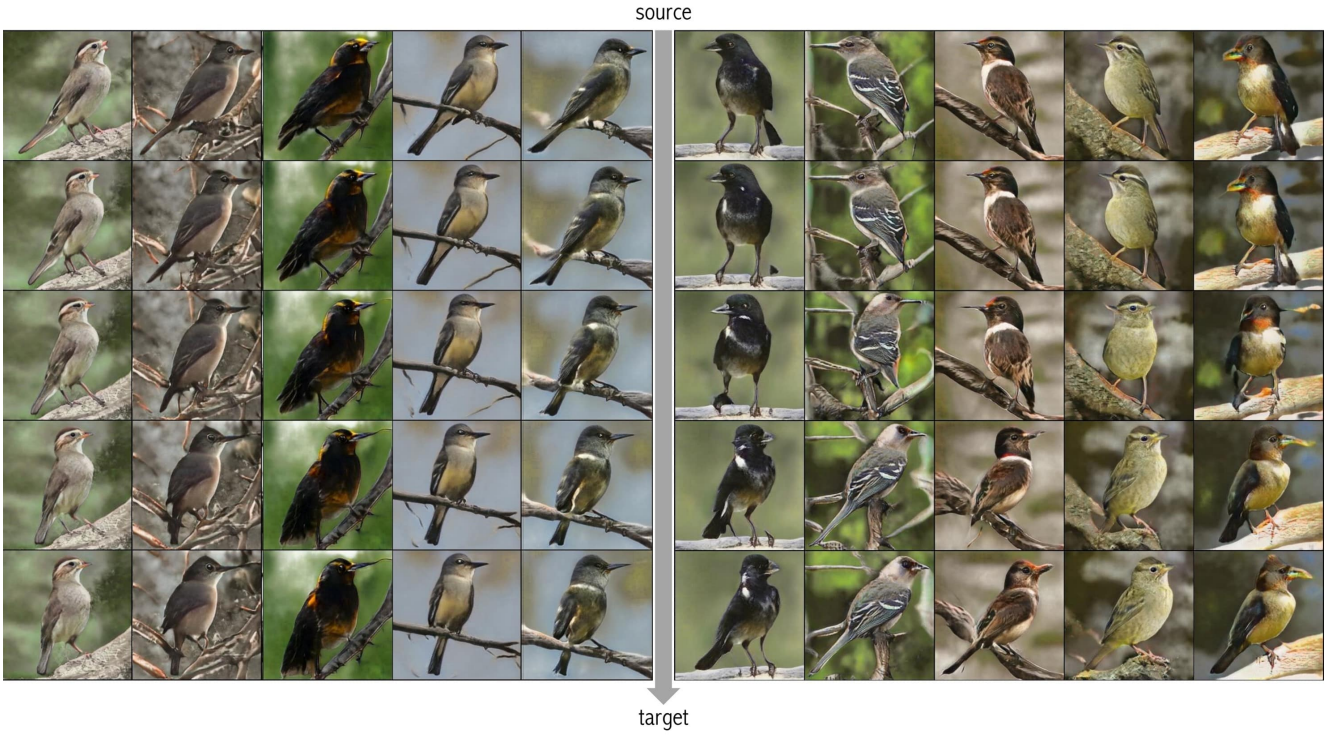

Figure 13: Examples from interpolated content codes. The images of each row are sampled from the same content code. All images are samples with changing content codes across all layers.

Figure 14: Generated 384×512 images by our method trained using car data set. (a) A source image generated from arbitrary style and content code. (b) Samples with varying style codes and the fixed content code. We can observe the changes of the global style attributes including car type, colors and background. (c) Samples generated with varying content codes and the fixed style. We can observe that the content attributes including rotation and global shapes change with different content codes. (d) Samples generated with both varying content and style codes.

Figure 15: Examples from interpolated content codes. The images of each row are sampled from the same content code. All images are samples with changing content codes across all layers.

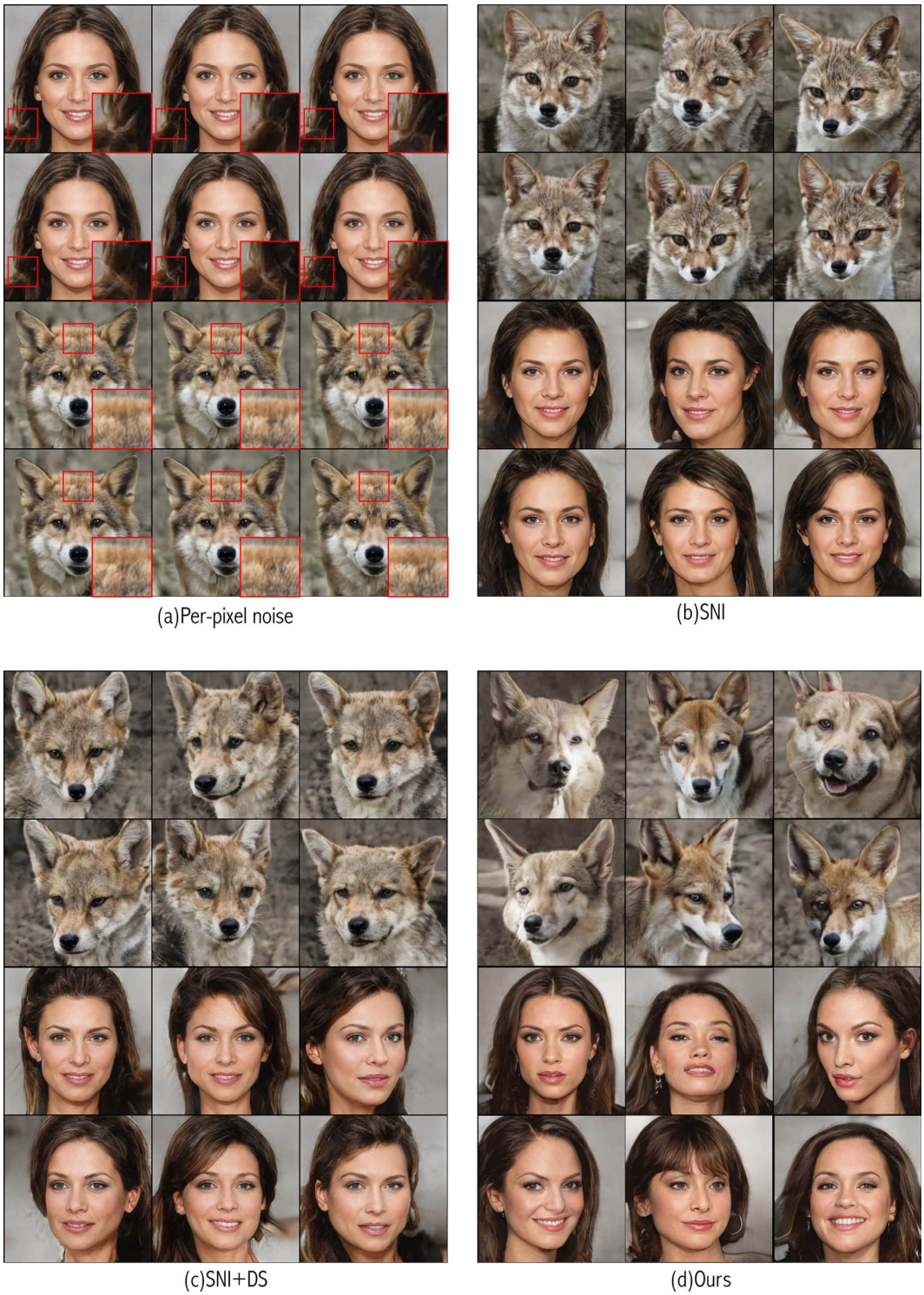

Figure 16: Comparison results of various content controlling methods. Sampled 256×256 images from (a) StyleGAN with varying per-pixel noises, varying content codes of (b) SNI , (c) SNI trained with DS loss, and (d) our model.

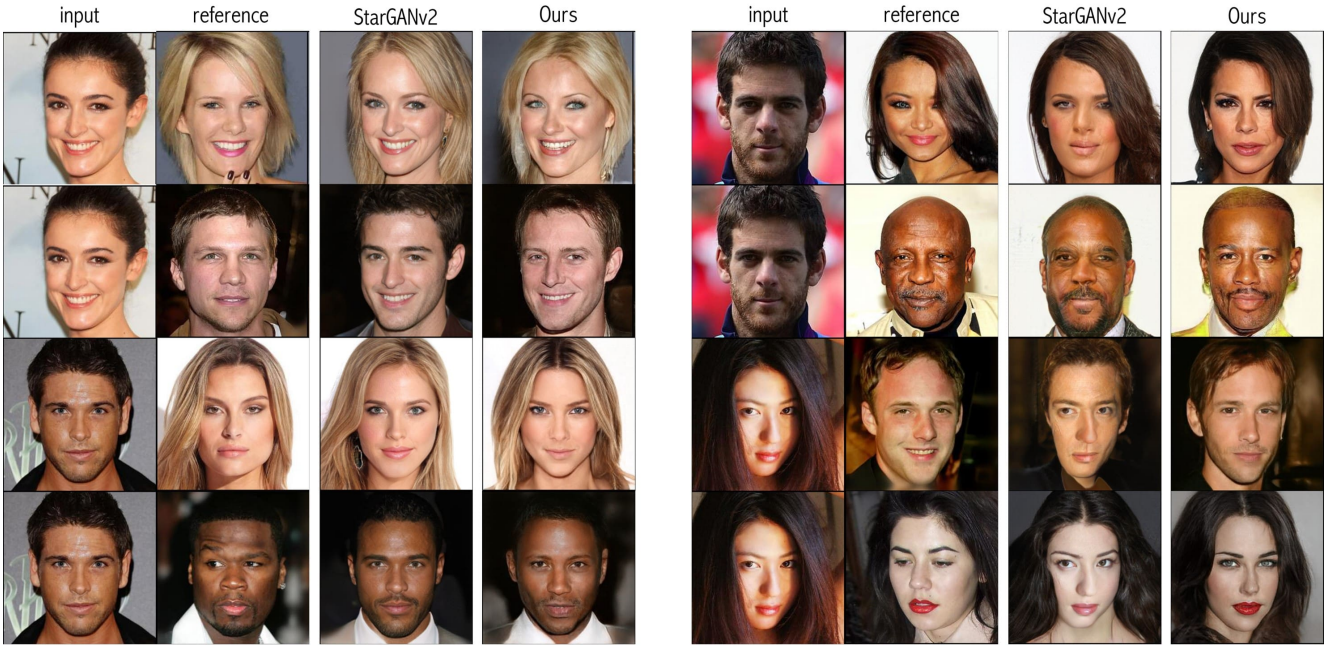

Figure 17: Comparison results of reference-based synthesis. (a) Results from typical input images. Both StarGANv2 and ours can generate realistic images with the styles of reference images. (b) Results from rare cases of input images. Our inversion model can successfully synthesize realistic images, whereas StarGANv2 fails.

Figure 18: Comparison results of latent-based synthesis. (a) Results from typical input images. Both StarGANv2 and Ours can generate realistic images with different styles. (b) Results from rare cases of input images. Our inversion model can successfully synthesize realistic images, whereas StarGANv2 fails.

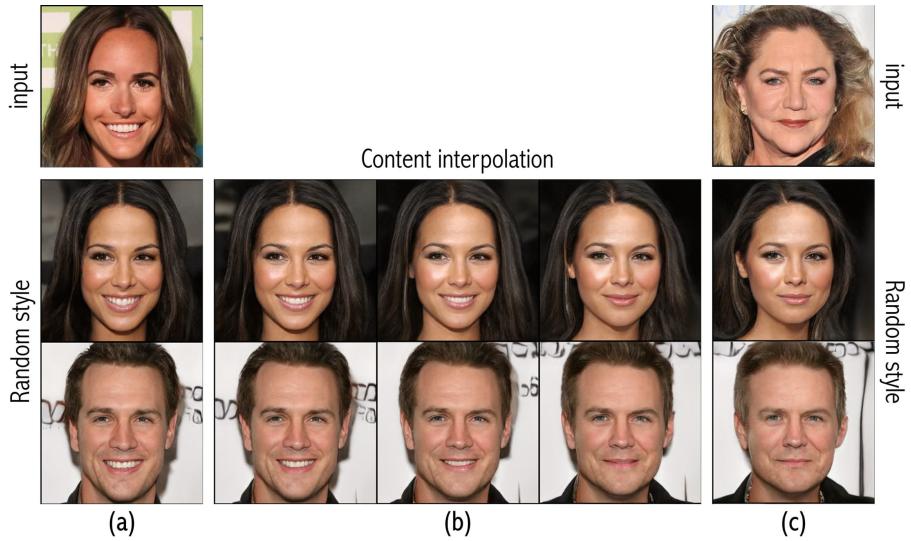

Figure 19: Results of content code interpolation from our model. (a,c) Similar to StarGANv2, our model can randomly change the style of the input images. (b) When interpolating the input content code between those of (a) and (c), we can obtain smooth content variation between two images. This variation is impossible in the existing image translation model such as StarGANv2.

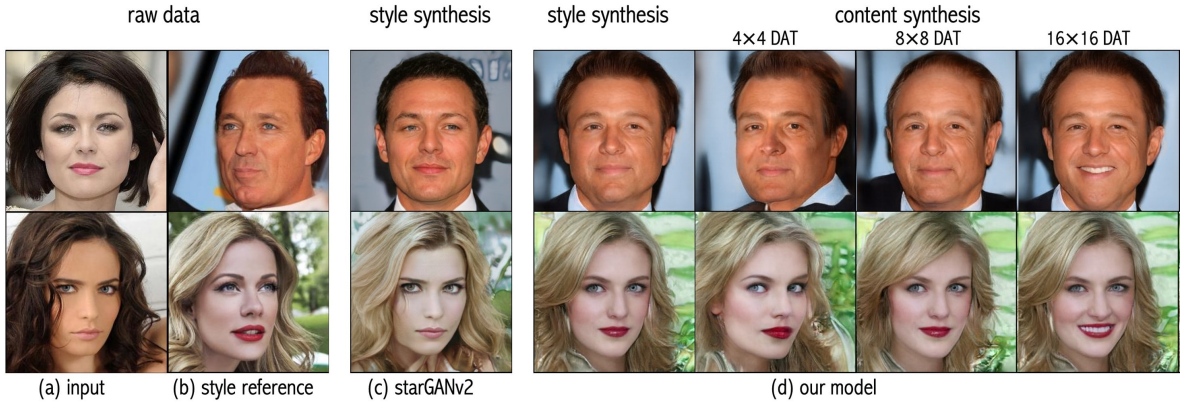

Figure 20: Comparison between StarGANv2 and our model. (a) Input image and (b) style reference images. Similar to StarGANv2 in (c), our model can synthesize the style using the style reference as shown in the leftmost column of (d). Furthermore, by changing the content codes in a hierarchical manner through DAT layers, the corresponding content attributes are selectively synthesized as shown in from the second to the fourth columns in (d). This fine content control is not possible using StarGANv2.