Supplementary Material: Contrastive Learning for Diverse Disentangled Foreground Generation

Yuheng Li[1,2], Yijun Li[2], Jingwan Lu[2], Eli Shechtman[2], Yong Jae Lee[1], and Krishna Kumar Singh[2]

[1]University of Wisconsin-Madison [2]Adobe Research

In this supplementary material, we first introduce our model architecture and implementation details in Section 1, followed by the dataset details in Section 2. In Section 3, we provide a visual example to better understand our approach in the main paper. Next, we give details about disentanglement study conducted in the main paper and also show this study on vanilla StyleGAN in Section 4. Finally, we show more studies and qualitative results in Section 5.

1 Architecture and implementation details

Our model is an encoder and decoder architecture. Figure 1 shows the encoder (left) and decoder (right) details. The input is the masked image of size 256×256. It consists of a bunch of Bi-modulated convolution and leaky relu activation functions. Inspired by [9] to preserve better spatial alignment information with the input inpainting mask, once the feature reaches to 4×4 we upsample it to 16×16 with lateral connections. In decoder, we have two bi-modulated convolution for each resolution and one *to_rgb* layer as output skip connection since it is beneficial for gradient update [5, 6]. Note we do not show two input codes for bi-modulated convolution block in both encoder and decoder for simplicity. Please refer Figure 3 in the main paper for details about bi-modulation.

We train our model for 200k iterations with learning rate of 0.002 using Adam optimizer. We set the weight of known factor loss λ_1 as 1, and weight of unknown factor loss λ_2 as 0.1 (for face and bird) and 5 (for car) for the initial 100k iterations. We decrease both λ_1 and λ_2 by 10 times for the next 100k iterations.

2 Dataset details

For face dataset, we use the official face parsing annotations from CelebA-HQ[8] to define our face region. Specifically, the following officially defined semantic regions are merged: 'skin', 'nose', 'eye_glass', 'l_eye', 'r_eye', 'l_eyebrow', 'r_eyebrow', 'l_ear', 'r_ear', 'mouth', 'u_lip', 'l_lip'. The union of these region is defined as face mask in all our experiments. We convert mask to box when we mask out inpatining region in the main paper. For the extra training data from FFHQ [5], we do the same process.

For bird and car which are obtained from the LSUN dataset [14], we first only choose images whose both sides are greater than 400 resolution and at least

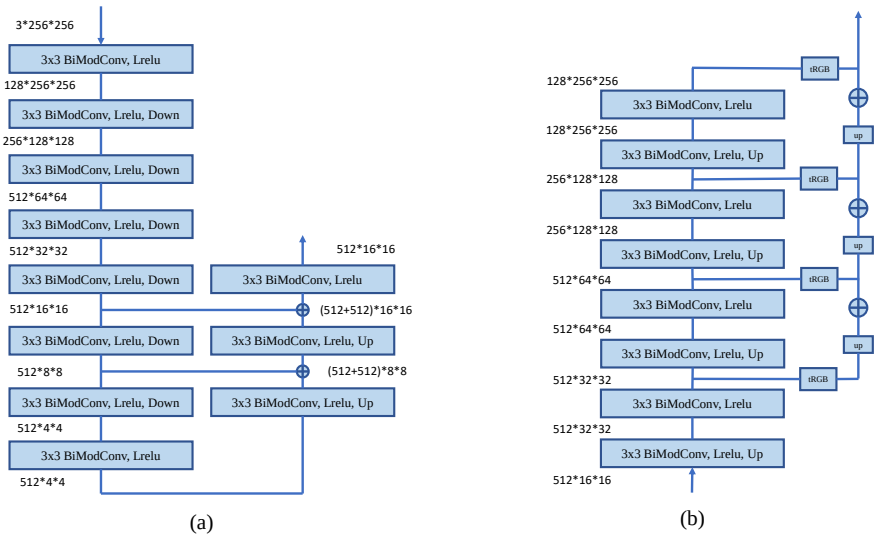

Fig. 1. Architecture details.

one side is greater than 512 resolution. Then we use a pretrained MaskRCNN [4] from Detectron2 [13] to select images. Specifically, the most confident detected object needs to be our target object ('bird' for our bird dataset, 'truck' or 'car' for our car dataset) and the most confident object also needs to be the biggest mask in its image.

We use the pretrained ArcFace [2] as the known factor encoder for face dataset, and train a classifier [3] on CUB dataset for bird known factor encoder. For car, we first group images in the Stanford car dataset [7] into 8 classes based on their shape and then train a classifier [3]. We choose the following shapes: 'SUV', 'Sedan', 'Hatchback', 'Coupe', 'Convertible', 'Wagon', 'Cab', 'Van' from their annotation names.

3 Visual explanations of our approach

To understand our approach more clearly, we use a visual example to demonstrate key notations and equations defined in the main paper. Note Figure 2 to Figure 5 are describing different terms in the same example.

Suppose we only have 4 known codes (warm colors: red yellow, pink and orange) and 4 unknowns codes (cold colors: blue green, purple and cyan) as shown in Figure 2. Like mentioned in the main paper, we ignore the input context image I as it is irrelevant to our contrastive learning, then each combination (in the middle of Figure 2) represents one resulting image S. And each parenthesis at the bottom means image pair (Eq1 in the main paper). In this case, there are $\binom{16}{2}$ image pairs in total.

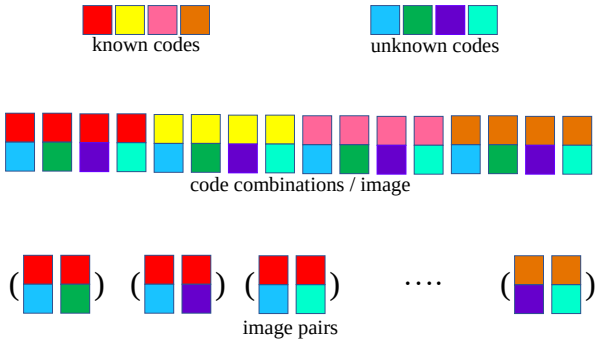

Fig. 2. 4 warm/cold colors represent known/unknown codes. Totally, there are 16 code combinations in the middle. Bottoms shows any two different combinations can be form as an image pair (Eq1 in the main paper).

Again we only consider the case in the known space for simplicity. The positive pair refers to two images sharing the same known code (e.g., top part color in the code combination should be same). In the main paper, we define $P_{\mathbf{k},u}$ as a set of all positive pairs *associated* with the code combination (k, u) in the known space. Here is how to understand our notation: P means positiveness, (k, u) indicates which code combination (or image), the bold letter defines we are considering from known space perspective. In Figure 3, we show one example of $P_{\mathbf{k},u}$ for the image whose known code is red and unknown code is blue.

In the main paper, $P_{\boldsymbol{K}}$ is defined as *all* the positive pairs in the known space. In this visual example, there are $4 * \binom{4}{2}$ pairs in total. Here is how to break it down: consider only one color in the known space, say red, and there are four images with red code. Among these four, we can choose $\binom{4}{2}$ pairs that are positive. With the same strategy applied to the other three colors, $|P_{\boldsymbol{K}}|$ is $4 * \binom{4}{2}$ in this case. To emphasize their differences: the $P_{\mathbf{k},u}$ is *associated* with one particular code combinations (or image) we are considering, whereas $P_{\boldsymbol{K}}$ consists of *all* positive pairs in the known space.

For the negative pair, we consider two images not sharing the same known code (top part color is different). Similarly, we also define $N_{\mathbf{k},u}$ as a set of all negative pairs *associated* with the code combination (k, u) in the known space. Figure 4 demonstrate 12 negative pairs for the image with red known code and blue unknown code. Among those 12 negative pairs, there are 3 hard negative pairs (parentheses are highlighted) which has the same unknown codes. Since in the hard negative pairs, features in the unknown space is the same, thus the network has to synthesize different factors according to the different known codes such that they can be recognized differently by the known factor encoder (orange one in the main paper Figure 2).

We show one example of loss function defined in the main paper Eq3. The Eq3 in the main paper is defined for *one* positive pair. Suppose we consider the image (red+blue) and image (red+green), the loss function calculated for

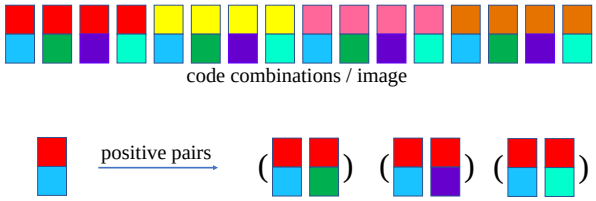

Fig. 3. This is an example for the notation $P_{\mathbf{k},u}$: among 16 code combinations (or image), for the given image (red code + blue code), we have three positive pairs for it.

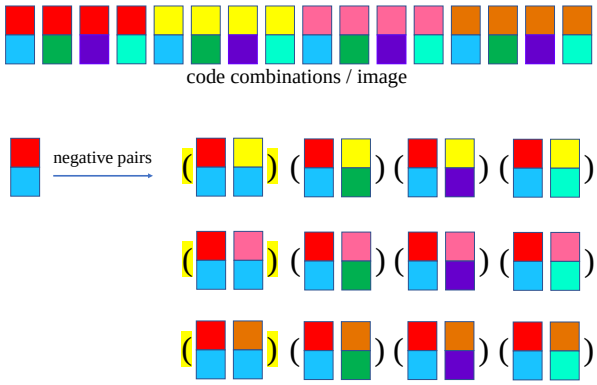

Fig. 4. This is an example for the notation $N_{\mathbf{k},u}$: among 16 code combinations (or image), for the given image (red code + blue code), we have 12 negative pairs for it. Parentheses of hard negative pairs are highlighted.

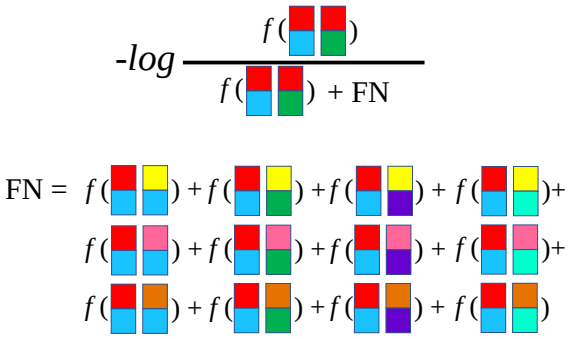

Fig. 5. Visual example of one case of Eq3 in the main paper.

this pair is visually shown in Figure 5. The $f(\cdot,\cdot)$ means the similarity between two images defined as in Eq2 in the main paper. Conceptually, we push the numerator high and FN in the denominator low. Since we have $4 * \binom{16}{2}$ positive pairs in this visual example, the final known space loss (Eq4 in the main paper)

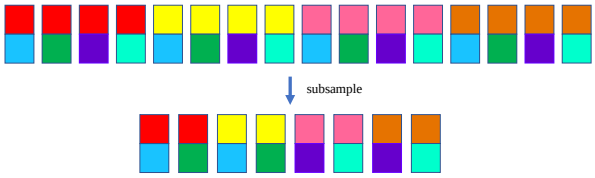

Fig. 6. Subsample 8 code combinations from 16.

is averaged across all positive pairs. Situation in the unknown space is the same which only requires viewing codes from a different perspective.

As mentioned in the main paper, in practice, we subsample code combinations (or images) such that each image has one hard negative pair in both spaces due to GPU memory constraint. Figure 6 shows the subsample result for our visual example. In practice, to increase code diversity in each space, we use 8 known and 8 unknown codes in our implementation, which will have 64 code combinations (or images) in total. We use the same subsample strategy as shown in Figure 6 resulting batch size of 16 during training.

In the main paper, we also evaluate one variant of our approach: ContrasFill-1 which only has unknown code. We apply contrastive loss for this model to increase the general diversity. This variant can be served as a baseline, which can be used to evaluate the effectiveness of contrastive loss on the diversity. And our final model, ContrasFill can be used to compare with it to study the benefit of two explicit disentangled spaces. To train such variant model, we only need to: (1) sample one set of code from unknown space each time (images with the same/different codes defined as positive/negative pairs), and apply contrastive loss in this space only. (2) replace all bi-modulated convolution in our model with normal modulated convolution proposed in the StyleGAN2 [6]. It is worth mentioning that although in this case we only have one set of code from unknown space, images with the same code do not mean that they are identical, since we always randomly sample the input image I during training as mentioned in the main paper.

4 Known factor direction study

In this section, we first provide more details about section 4.3 of the main paper and then discuss identity direction in unconditional StyleGAN.

4.1 Details about Section 4.3 in the main paper

In the main paper Section 4.3, we study the disentanglement of known factor direction in the latent spaces of baselines (CoModGAN, CollageGAN, ContrasFill-1). Here we describe more details.

Following [11], we train a linear regressor based on latent codes and their identity labels. Specially, we first randomly sample 100k images using 100k latent

codes; and then use the pretrained known factor classifier to get penultimate features of these images. Then we try to group these 100k features into 1k clusters using K-means. For each cluster, we choose top 10 features which are closest to their clustering center. Since each image feature is associated with its latent code, after doing so, we have 10k codes with their group labels. We then train a linear regressor, which predicts a scalar value for each code. The objective for this regressor is to predict same/different value if codes belonging/not belonging to the same cluster. We use the contrastive loss [1] to train this regressor. After training, the weight of this regressor should indicate the latent direction for the known factor. For example, in the case of face, if one moves along this discovered direction, then the ideal change in the image should be related with identity.

After discovering the known factor directions for baselines. We randomly sample 1k images and for each image we sample 10 different results for the same mask by moving along the discovered known direction d. Since it is hard to define how far one should move along this direction, thus we use the trained discriminator to monitor this process. Specifically, for each image, we first sample 1 result and denote its latent as w_o. We obtain its realness score r (between 0 to 1) base on the trained discriminator, and then we define a lower bound l as $r - 0.1$. To sample along this direction, we randomly sample a step s from $\mathcal{U}_{[-R,R]}$, where R is a scalar. Then the new image will be synthesized by the latent code $w_o + s * d$. We only choose the image whose realness score is above than lower bound l. Otherwise, it indicates we step too far such that the image is not in the distribution anymore. If this happens (e.g., realness is lower than lower bound) we call it a miss. We empirically set R such that the chance of missing is around 10%, and this is to make sure we do not have a too small step in the direction d.

As the results shown in the main paper, the postprocessing method is not as good as having explicit disentangled latent space even if they all access to the same supervision.

4.2 Identity direction in StyleGAN

The experiment in the main paper section 4.3 shows that directions for certain factors such as human identity can not be easily disentangled in the StyleGAN based inpainting model. For completeness, we find that this is also true for the unconditional StyleGAN.

Starting with CelebA-HQ dataset where we know identity labels and a pretrained StyleGAN, we first use PSP [10], the state-of-the-art StyleGAN face inversion encoder, to encode these images into the StyleGAN latent space. Then we treat latent codes and their corresponding identity labels as training data. Similarly, following the supervised method [12], we train a linear regressor to group all latent codes. Once it is trained, its weight should indicate the direction for identity in the vanilla StyleGAN latent space.

Figure 7 shows the results where we move along the discovered identity direction for three samples (middle one in each row). This direction is able to change identity, however, the resulting images still resemble images in the middle. Also

Fig. 7. Moving along the discovered identity direction in the StyleGAN latent space.

Fig. 8. Each column has the same known factor (color), and each column has the same unknown factor which controls the object shape and pose.

certain attributes change as well such as eyes open more and makeup become heavier when moving to the right. This shows that the identity direction can not be easily disentangled even in the unconditional StyleGAN.

5 Additional qualitative results and studies

Here we provide additional qualitative and quantitative studies about our method.
Generalization across different known factors. To demonstrate the generalization of our approach on the known factor, we train a car color classifier as our known factor feature extractor. We use the Stanford car dataset [7] as training data, and average pixels of foreground object in each image and run k-means to group all averaged colors into 8 classes. Then we train our model on the same car data we have. Figure 8 shows the disentanglement results, where each column has the same known code controlling the color, and each row has the same unknown code which controls object pose and shape. Please compare

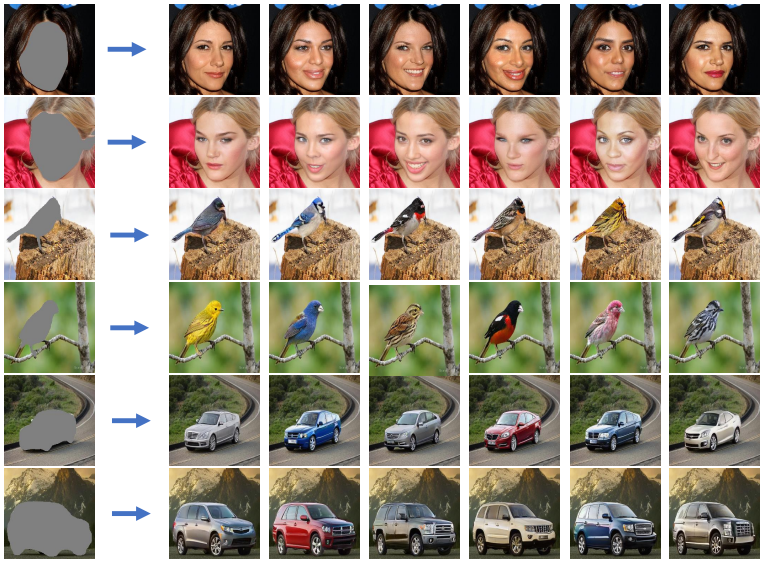

Fig. 9. We can achieve diverse samples when filling region is object semantic mask.

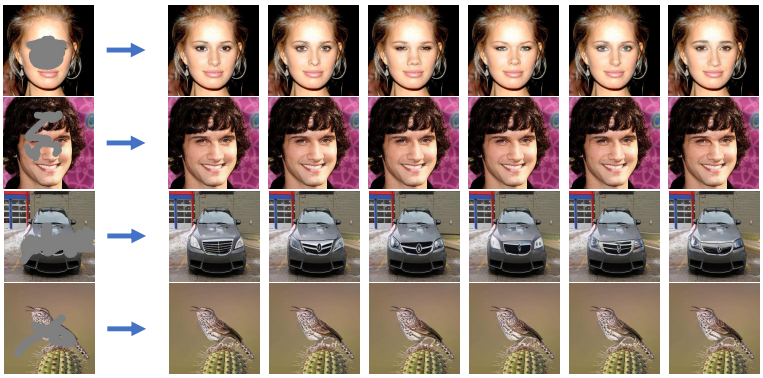

Fig. 10. If masked region allows diversity (row 1, 3), our model still can generate diverse results. Our model generate almost deterministic results when partially masked region is strongly correlated with the context regions (row,2,4)

this result with the car result in Figure 6 of the main paper where the known factor is the car shape.

Other mask regions instead of box. In the main paper figure 8, we show results where the foreground region is the object semantic mask, instead of box, for face dataset. Here we show more results on all three datasets in Figure 9. For face, we can still generate diverse identities and expressions. For bird, we can generate different textures. For car, we can generate different color, headlights

	Face			Bird			Car		
	FID	LPIPS	KFFA	FID	LPIPS	KFFA	FID	LPIPS	KFFA
CoModGAN	**5.73**	0.029	51.19	7.92	0.030	27.35	**5.77**	0.146	57.53
CollageGAN	6.07	0.029	59.73	8.99	0.023	24.70	6.02	0.171	58.88
BAT-Fill	11.97	**0.050**	72.48	31.25	0.080	55.36	19.88	0.200	60.44
ContrasFill (Ours)	5.95	0.048	**83.39**	**7.74**	**0.101**	**72.23**	5.95	**0.201**	**77.86**

Table 1. Our method has comparable image quality with the state-of-the-art, but with more diversity.

and car grills. Note that we set known code as color in car dataset as shape can not be changed when the foreground region is object semantic mask. We also quantitatively measure our generation quality (FID), overall diversity (LPIPS) and known factor diversity (KFFA) with other baselines in three datasets when the missing region is object mask, and we report numbers in Table 1. Overall, we have comparable image quality with CoModGAN [15] and CollageGAN [9] as reflected by FID, but we can generate much more diverse results, especially in terms of known factors. Although BAT-Fill has high diversity also, they suffer from image quality as reflected by the FID.

Our problem setting is foreground object generation where the entire foreground region should be masked, but we also tried to study how does our model behave if object is masked partially. As shown in the Figure 10, if there is loose correlation between masked region and context, we can still generate diverse results. E.g., if we masked both eyes (1st row) and car headlights and grills (3rd row), our model can still generate diverse results. But if masked regions have strong correlation with the context and can be inferred only in a fixed way, then our generated results are deterministic with limited variations. E.g., if we only mask one eye (2nd row) or part of the bird (4th row), then results are almost deterministic with only small variation (such as texture patterns of the birds in the 4th row if check closely). Our model behaviour is actually expected; since if a missing region is heavily dependent on the context, then results should be almost the same. Nevertheless, in our problem setting of foreground generation, we generate diverse results as entire object is masked and missing regions are not strongly correlated with the context.

Importance of context. Although in our setting the entire object is generated, the model still needs to use context for a correct generation. To study the importance of context, one naive baseline is to treat this as a pure generation problem: generating a foreground object without considering context and then pasting the result back. Thus we design a baseline where the model is only conditioned on the mask to confine the generation region, and after generation we copy and paste the generated result back to the masked context image. As Figure 11 shows that if we generate a foreground object without considering context information, the generated objects may not be compatible with original surroundings in terms of lighting, pose and other factors.

Fig. 11. A model will generate inconsistent foreground without considering context.

More qualitative results. Finally, we show more results of comparisons with baselines in Figure 12, Figure 13 and Figure 14. Note that for CoModGAN [15] and CollageGAN [9], they have difficulties in changing object shape and pose for bird and car datasets. Figure 15 shows disentanglement results.

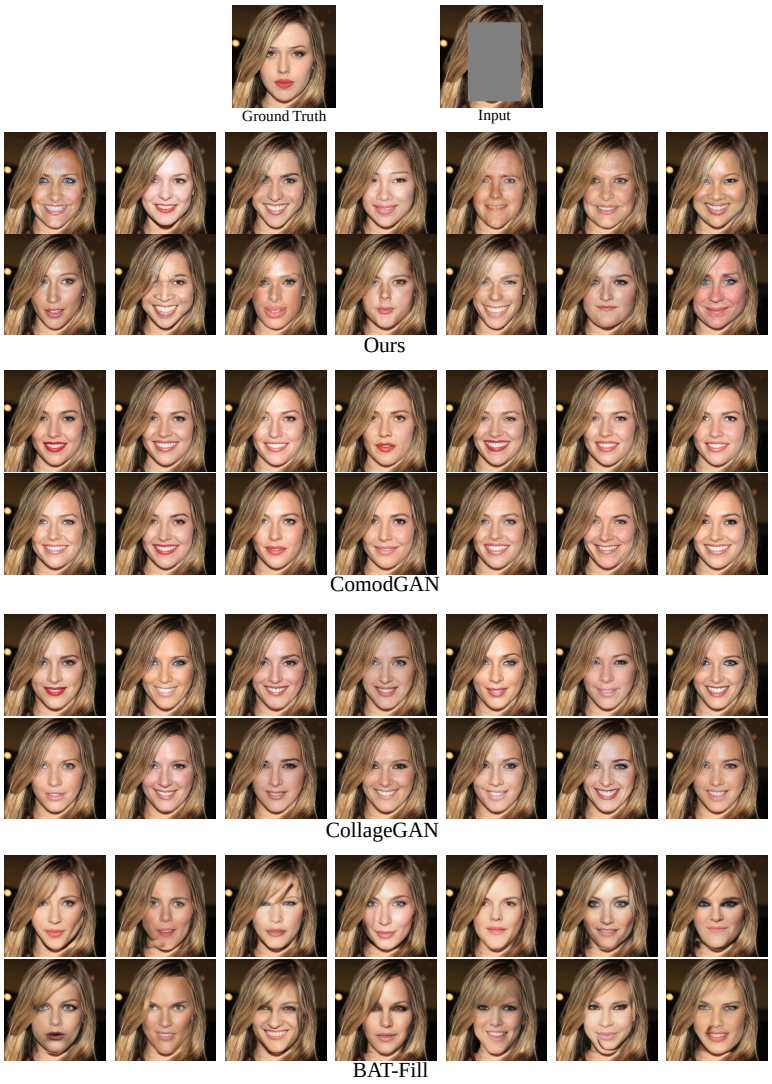

Fig. 12. Random samples on the face dataset.

Fig. 13. Random samples on the bird dataset.

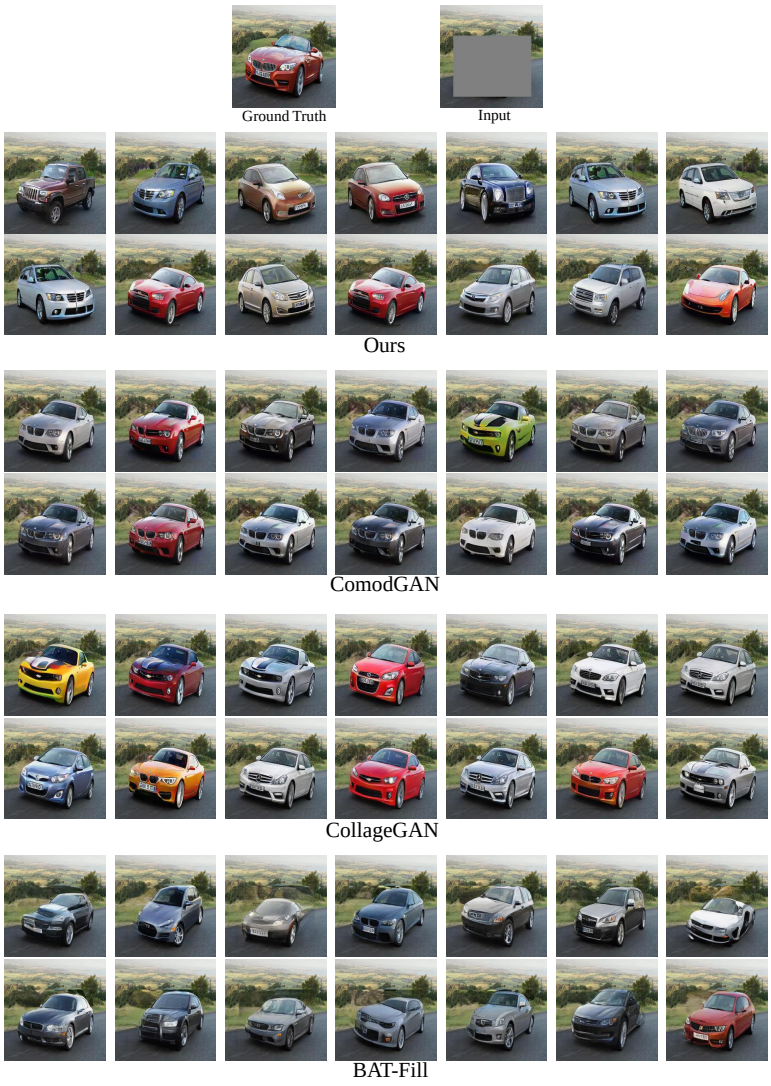

Fig. 14. Random samples on the car dataset.

Fig. 15. Disentanglement results on three different datasets.

References

1. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E.: A simple framework for contrastive learning of visual representations. ArXiv **abs/2002.05709** (2020)
2. Deng, J., Guo, J., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 4685–4694 (2019)
3. Du, R., Chang, D., Bhunia, A.K., Xie, J., Song, Y.Z., Ma, Z., Guo, J.: Fine-grained visual classification via progressive multi-granularity training of jigsaw patches. In: European Conference on Computer Vision (2020)
4. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask r-cnn. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**, 386–397 (2020)
5. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019)
6. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of StyleGAN. In: Proc. CVPR (2020)
7. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13). Sydney, Australia (2013)
8. Lee, C.H., Liu, Z., Wu, L., Luo, P.: Maskgan: Towards diverse and interactive facial image manipulation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
9. Li, Y., Li, Y., Lu, J., Shechtman, E., Lee, Y.J., Singh, K.K.: Collaging class-specific gans for semantic image synthesis. ICCV (2021)
10. Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., Cohen-Or, D.: Encoding in style: a stylegan encoder for image-to-image translation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2021)
11. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 9240–9249 (2020)
12. Shen, Y., Yang, C., Tang, X., Zhou, B.: Interfacegan: Interpreting the disentangled face representation learned by gans. TPAMI (2020)
13. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)
14. Yu, F., Zhang, Y., Song, S., Seff, A., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. ArXiv **abs/1506.03365** (2015)
15. Zhao, S., Cui, J., Sheng, Y., Dong, Y., Liang, X., Chang, E.I.C., Xu, Y.: Large scale image completion via co-modulated generative adversarial networks. ArXiv **abs/2103.10428** (2021)