Supplementary material for Semantically Multi-modal Image Synthesis

1. Implementation details

Network architectures. In this section, we give detailed network designs for each dataset. We demonstrate the architecture of the discriminator in Fig. 1. Note the architecture of the discriminator holds the same for different datasets. In Fig. 2, we demonstrate the encoder architecture for different datasets. Fig. 3 depicts the architectures of the decoders for the DeepFashion and Cityscapes, while Fig. 4 shows the architecture of the decoder for ADE20K. Since ADE20K has so many classes, we bring down the channel number for each group to avoid massive GPU usage. In this case, the overall network capacity decreases, and we assume it's not helpful to the results. Therefore, we add some additional convolutional layers to enlarge the network capacity; thus, this makes the architecture of the decoder for ADE20K is different from those of the other two datasets.

Training details. We train all experiments on DeepFashion for 100 epochs, where in the first 60 epochs, the learning rates for both the generator and discriminator maintain the same while linearly decay to 0 in the last 40 epochs. For Cityscapes and ADE20K datasets, we follow the training settings of SPADE [7] to train 200 epochs, where the learning rates linearly decay to 0 from 100 to 200 epochs. The image sizes are 256×256, except the Cityscapes at 512×256. The batch size for DeepFashion and Cityscapes is 32 while 16 for ADE20K due to the large number of channels to meet the requirements of sufficient capacity for the 150 classes. The network weights are initialized with Glorot initialization [2] following SPADE [7].

Group number selection strategy. Actually, it is hard to devise a programmatic strategy to decide the decreasing numbers, under restrictions of the capacity of GPU memory, batch size, and the number of parameters *etc*. However, we still followed two rules to design the group numbers: 1) the numbers decrease drastically in the first several layers of the decoder to largely reduce the computational cost; 2) the group number in the previous layer is either equal or 2 times of that in the next layer.

2. Datasets

DeepFashion [5]. DeepFashion (In-shop Clothes Retrieval Benchmark) contains of 52,712 person images with fashion clothes. We select about 29,000 training and 2,500 validation images. After that, we use an off-the-shelf human parser [3] pre-trained on the LIP dataset [4] to get segmentation maps. Specifically, given an input image, we first get its segmentation map, then re-organize the map into eight categories: hair, face, skin (including hands and legs), top-clothes, bottom-clothes, socks, shoes, and background. At the same time, we filter out the images with some rare attributes like a glove, and so on. We choose DeepFashion because this dataset shows lots of diversities of all semantic classes, which is naturally suitable for assessing the model's ability to conduct multi-modal synthesis.

Cityscapes [1]. Cityscapes dataset [1] has 3,000 training images and 500 validation images, collected from German cities. The size of the images in Cityscapes are quite large, so it is proper to test the model's ability to produce high-resolution images on this dataset.

ADE20K [9]. ADE20K dataset [9] contains 20,210 training and 2000 validation images. This dataset is extremely challenging for many tasks because it contains up to 150 semantic classes. ADE20K is extremely challenging for its massive number of classes, and we find it hard to train MulNet and GroupNet on ADE20K with our limited GPUs.

3. Additional results

In Fig 5, we show more ablation qualitative results on DeepFashion. The conclusions are basically the same as we put in the main submission. One thing to note is that compared to MulNet, GroupNet, and GroupEnc, our GroupDNet has better color, style, and illumination consistencies due to its design consideration for carving the correlation among different classes. Likewise, GroupDec and VSAPDE seem to have the ability to consider class correlations just as GroupDNet, because the regular convolutions in their decoders help to discover the relationships. But they instead lose strong SMIS controllability, unlike GroupDNet. These results firmly verify the efficacy of GroupDNet and show its balanced trade-off between SMIS

Models	FID↓	mCSD↑	mOCD↓	LPIPS↑	SHE↑	FPS ↑	# Para↓
GroupDNet	**9.50**	0.0264	**0.0033**	0.228	**81.2**	12.2	109.1
w/o map	11.01	0.0253	0.0036	0.217	79.5	11.5	109.3
w/o split	10.76	0.0054	0.0189	0.216	31.7	12.1	109.1
→GroupNorm	10.33	0.0256	0.0040	0.225	77.0	12.2	109.1
w/o SyncBN	9.76	0.0251	0.0037	0.216	79.3	12.3	109.1
w/o SpecNorm	10.42	**0.0290**	0.0153	**0.231**	46.3	**13.5**	109.0

Table 1. Quantitative results of the ablation experiments on the DeepFashion dataset.

controllability and image quality.

In Fig 6, Fig 7 and Fig 8, we show additional comparison results from the proposed method on the DeepFashion, Cityscapes and ADE20K datasets with pix2pixHD [8] and SPADE [7]. These results show that the image quality of GroupDNet is slightly better than the other two methods, especially in terms of keeping the object structures ordered and regular in the Cityscapes dataset (See the buildings and cars in these pictures).

In the accompanying video attached to our code base[1], we demonstrate more results of our model on all datasets. Besides, we give a more straightforward demonstration of our exemplary applications. The video shows more results and detailed instructions of our exemplary applications in the main submission. From these videos, we exhibit the SMIS performance of GroupDNet on all three datasets and the potential applications of models designed for SMIS task. However, our model trained on the Cityscapes dataset seems to lose semantic controllability. For example, when altering the latent code for the buildings, other parts follow to change with the buildings. We are not willing to regard this phenomenon as a significant flaw. In some cases, we do hope the whole image can change alongside the change of a class-specific latent code because it strengthens the fidelity of the generated images. For example, the discordance between the overall illumination and illuminations on some objects could make the image unrealistic and unnatural. Another problem is the diversity of the generated results on Cityscapes seems quite limited. It is because this dataset is restricted initially to the scenes of German cities. Moreover, the images inside the dataset were shot during short intervals; hence, they exhibit no diversity of illumination from daylight to darkness. Seeing these results, we firmly believe semantically multi-modal image synthesis has more applications and intrinsic scientific values that deserve to explore given suitable datasets. In the future, we'll investigate more into GroupDNet and try to improve its performance in SMIS.

4. Additional ablation study

To support for the SMIS task and improve the quality of generated images, we made several minor modifications to GroupDNet, including: 1) splitting the original input image to different images of different semantic classes, as mentioned in our main text; 2) enforcing the encoder to produce a mean map and variance map rather than a mean vector and variance vector that are used in SPADE. To validate the effects of these strategies, we conduct ablation experiments by not using them in the networks, thus we have the results of the model without splitting the original images (**w/o split**) and the model without producing a mean and variance map (**w/o map**). Note for the latter one, we use add global average pooling at the last of the encoder to compress maps into vectors, instead of using the routine fully connected layers to produce the vectors, which could cripple the individuality of each class. Results shown in Tab. 1 indicates: splitting the input image or producing a mean and variance map are necessary strategies for the SMIS task, otherwise the model will suffer from a degraded performance of FID, mCSD and mOCD.

Besides, considering the vast use of group convolution in GroupDNet, it is very interesting to know what the effect would be if we apply group normalization as the main normalization layer in our model because group normalization also operates separately on different groups of feature channels. Therefore, we conduct another experiment by changing all normalization layers in our original model to group normalization layers and set their group numbers equal to their previous convolutional layers (→**GN**). We also conduct experiments to investigate the impacts of several normalization layers we use in our model by discarding the use of them. Basically, we then have the model without using synchronized batch normalization (**w/o SyncBN**) and without using spectral normalization [6] (**w/o SpecNorm**). Results are reported in Tab. 1. Changing the normalization layers to group normalization or removing the synchronized batch normalization have slightly degraded the performance on most metrics. However, removing the spectral normalization layers in GroupDNet will largely increase mOCD, indicating spectral normalization helps for the SMIS task. The LPIPS and mCSD metrics of the model without SpecNorm are even higher than GroupDNet, which hints that spectral normalization may have a negative effect on the diversity of the model's generated images.

5. Discussions and future work

As mentioned earlier, a limitation of GroupDNet is the restricted power to capture the class-wise modality of images in the Cityscapes dataset. Though the dataset itself demonstrates very few variation of object appearance, we believe there remains lots of methodological and architectural modifications that could enable GroupDNet or other models to handle such difficult cases.

Besides, we also feel the necessity to design a clean and effective strategy to decide how to set the group numbers for each convolution or possibly normalization layers. Though

[1]https://github.com/Seanseattle/SMIS

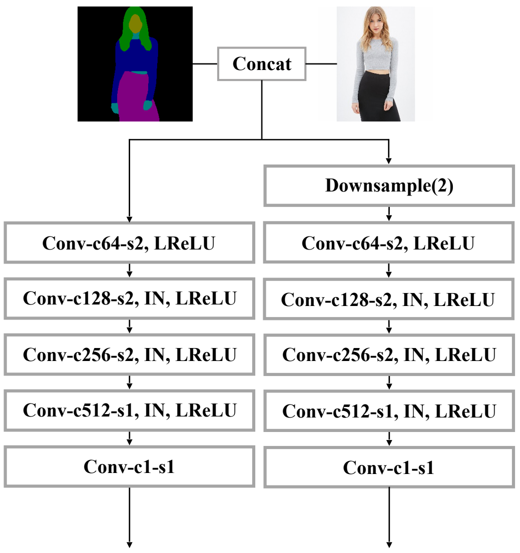
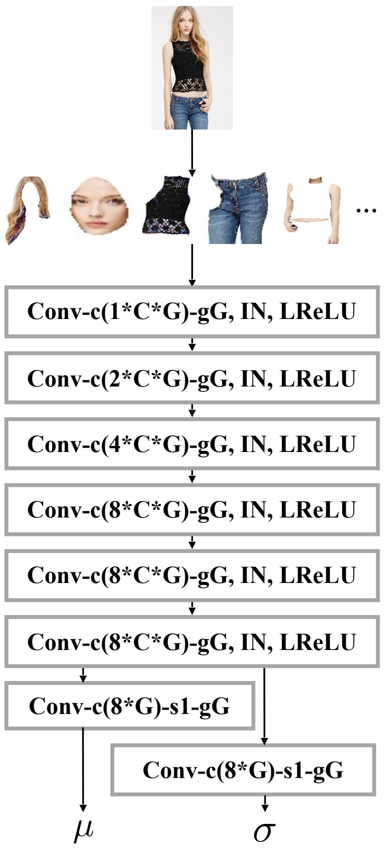


Figure 1. Architecture of our discriminator for all three datasets. Note “Conv” means convolutional layer. The numbers after “-c”, “-s” and “-g” represent the channel number, stride and group number of the corresponding convolution. If not specified, the default kernel size, stride, padding and group number of the convolutional layer are 4, 2, 2, 1, respectively. “IN” represents instance normalization layer and “LReLU” means leaky ReLU layer. “Downsample(·)” means an average pooling layer with kernel size set to the number inside the bracket.

we haven’t give clear experimental evidence, we feel that different configuration of group numbers might have some impacts on the performance. This conclusion is natural because different group number configurations determine different network structures and more often different network structures have influence on the network performance.

Moreover, it is also interesting to discover whether changing the input order of different classes could make any difference to the performance, considering now we feed the split input images to the encoder, following the order set by the dataset provider or randomly set by us. It is quite natural to reason that putting similar classes together could make the corresponding areas change harmoniously and concurrently, thus producing images with much more fidelity.



...
Conv-c(1*C*G)-gG, IN, LReLU
Conv-c(2*C*G)-gG, IN, LReLU
Conv-c(4*C*G)-gG, IN, LReLU
Conv-c(8*C*G)-gG, IN, LReLU
Conv-c(8*C*G)-gG, IN, LReLU
Conv-c(8*C*G)-gG, IN, LReLU
Conv-c(8*G)-s1-gG
Conv-c(8*G)-s1-gG
μ
σ

Figure 2. Architecture of our encoder for three datasets. Note “Conv” means convolutional layer. The numbers after “-c”, “-s” and “-g” represent the channel number, stride and group number of the corresponding convolution. If not specified, the default kernel size, stride, padding and group number of the convolutional layer are 3, 2, 1, 1, respectively. “IN” represents instance normalization layer and “LReLU” means leaky ReLU layer. Here “C” and “G” are pre-defined numbers for each datasets, which are given in Tab. 2.

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. CVPR*, pages 3213–3223, 2016. 1

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, pages 249–256, 2010. 1

[3] Ke Gong, Xiaodan Liang, Yicheng Li, Yimin Chen, Ming Yang, and Liang Lin. Instance-level human parsing via part grouping network. In *Proc. ECCV*, pages 805–822, 2018. 1

[4] Ke Gong, Xiaodan Liang, Dongyu Zhang, Xiaohui Shen, and Liang Lin. Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In *Proc. CVPR*, pages 6757–6765, 2017. 1

[5] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proc. CVPR*, 2016. 1

[6] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proc. ICLR*, 2018. 2

[7] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proc. CVPR*, pages 2337–2346, 2019. 1, 2

[8] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proc. CVPR*, pages 8798–8807, 2018. 2

Models	DeepFashion	Cityscapes	ADE20K
Encoder	$C = 64, G = 8$	$C = 8, G = 35$	$C = 3, G = 151$
Decoder	$C = 160, G = \{8, 8, 4, 4, 2, 2, 1\}$	$C = 280, G = \{35, 35, 20, 14, 10, 4, 1\}$	$C = \{151, 64\}, G = \{151, 16, 16, 8, 4, 2, 1, 1\}$

Table 2. Pre-defined hyperparameters of different datasets for our encoder and decoder. Note in Fig. 2, Fig. 3 and Fig. 4, "$C\{i\}$" represent the i-th number inside the brace of C and "$G\{i\}$" likewise in the brace of G.

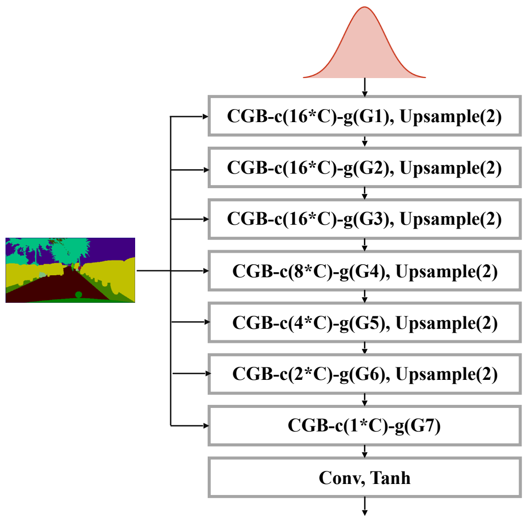

CGB-c(16*C)-g(G1), Upsample(2)
CGB-c(16*C)-g(G2), Upsample(2)
CGB-c(16*C)-g(G3), Upsample(2)
CGB-c(8*C)-g(G4), Upsample(2)
CGB-c(4*C)-g(G5), Upsample(2)
CGB-c(2*C)-g(G6), Upsample(2)
CGB-c(1*C)-g(G7)
Conv, Tanh

Figure 3. Architecture of our decoder for the DeepFashion and Cityscapes datasets. Note "CGB" means our conditional group block (CG-Block). The numbers after "-c", "-s" and "-g" represent the channel number, stride and group number of the corresponding convolution. If not specified, the default kernel size, stride, padding and group number of the convolutional layer are 3, 1, 1, 1, respectively. After each CG-Norm inside CG-Block, there follows a ReLu layer. "Upsample(·)" means a nearest neighbor upsampling layer with kernel size set to the number inside the bracket. Here "C" and "G" are pre-defined numbers for each datasets, which are given in Tab. 2.

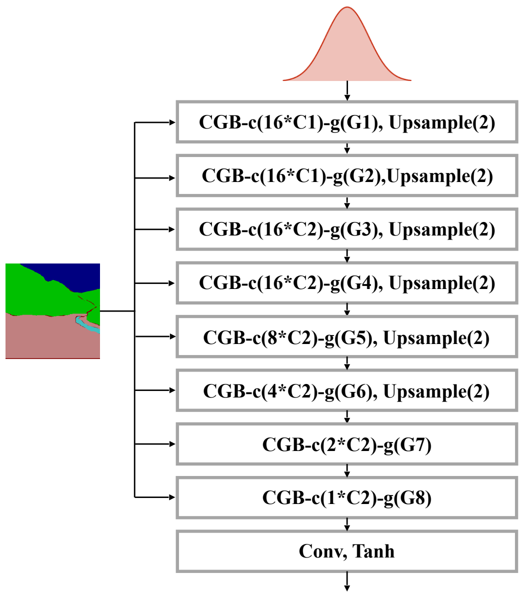

CGB-c(16*C1)-g(G1), Upsample(2)
CGB-c(16*C1)-g(G2),Upsample(2)
CGB-c(16*C2)-g(G3), Upsample(2)
CGB-c(16*C2)-g(G4), Upsample(2)
CGB-c(8*C2)-g(G5), Upsample(2)
CGB-c(4*C2)-g(G6), Upsample(2)
CGB-c(2*C2)-g(G7)
CGB-c(1*C2)-g(G8)
Conv, Tanh

Figure 4. Architecture of our decoder for the ADE20K datasets. Note "CGB" means our conditional group block (CG-Block). The numbers after "-c", "-s" and "-g" represent the channel number, stride and group number of the corresponding convolution. If not specified, the default kernel size, stride, padding and group number of the convolutional layer are 3, 1, 1, 1, respectively. After each CG-Norm inside CG-Block, there follows a ReLu layer. "Upsample(·)" means a nearest neighbor upsampling layer with kernel size set to the number inside the bracket. Here "C" and "G" are pre-defined numbers for each datasets, which are given in Tab. 2.

[9] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *Proc. CVPR*, pages 5122–5130, 2017. 1

[10] Zhen Zhu, Tengteng Huang, Baoguang Shi, Miao Yu, Bofei Wang, and Xiang Bai. Progressive pose attention transfer for person image generation. In *Proc. CVPR*, pages 2347–2356, 2019.

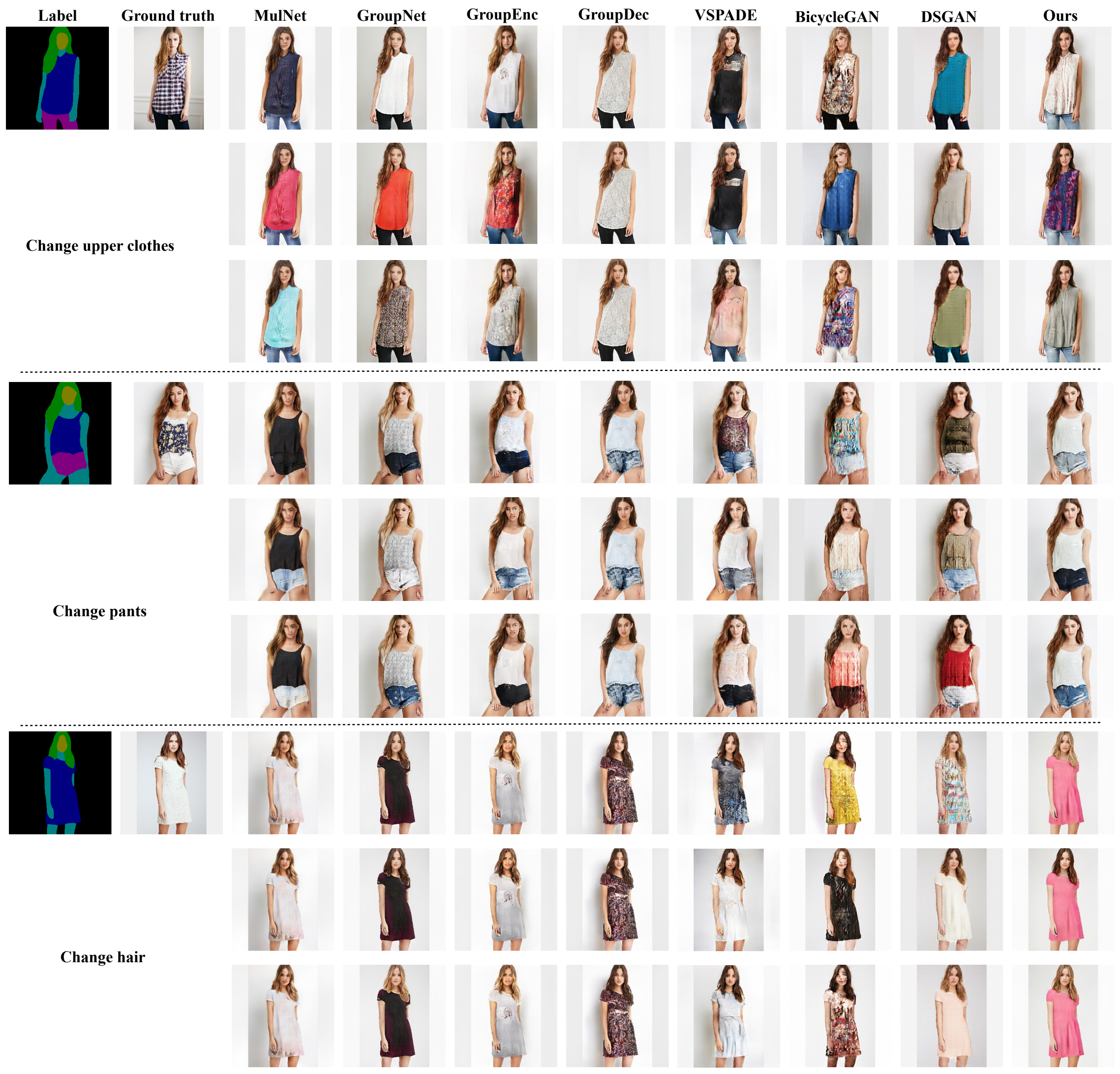


Figure 5. Qualitative comparison between GroupDNet and other baseline models. The first three rows represent the results of different models by changing their upper-clothes latent code. The middle three rows represent the results of different models by changing their pants latent code while the last three rows represent their results of changing the hair latent code. Note, for those models which have no class-specific latent code such as VSPADE, we alter their overall latent codes to generate different images.

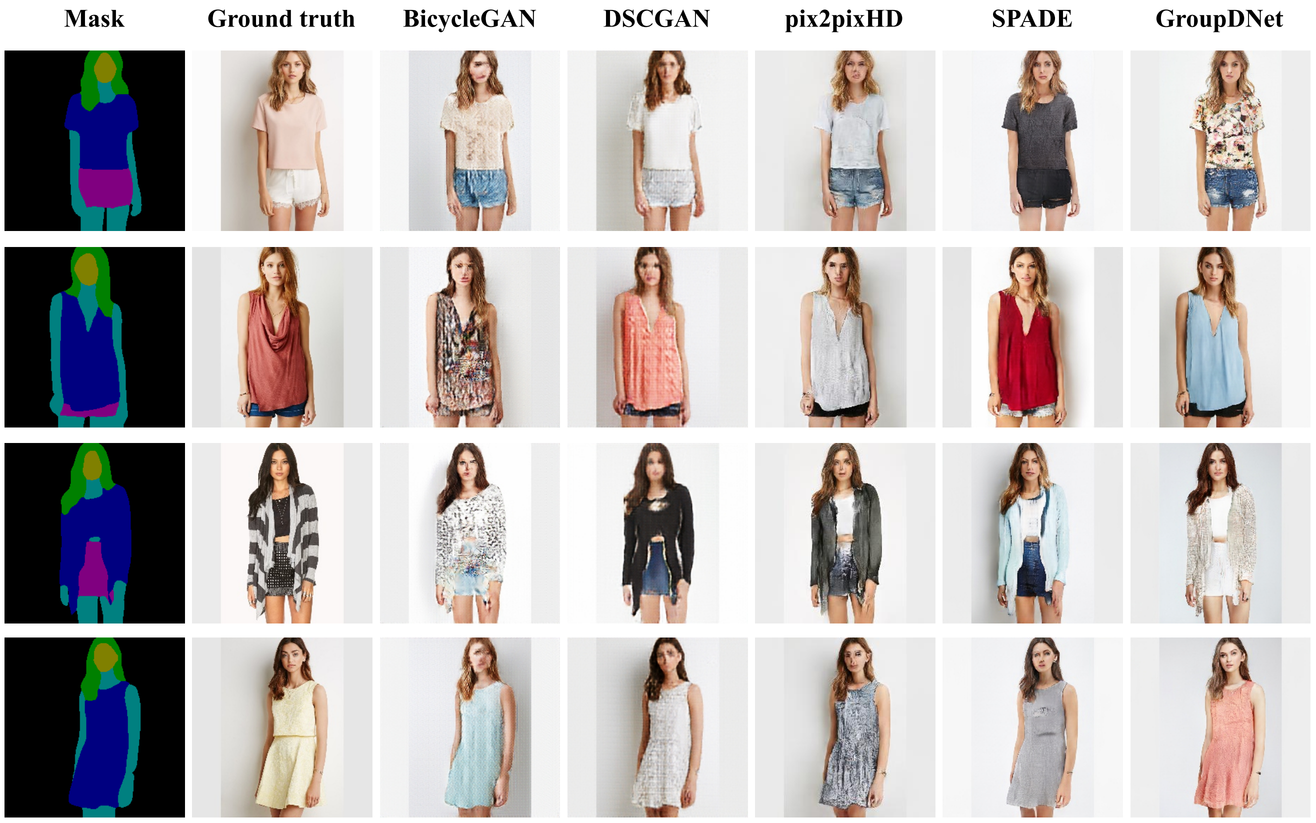


Figure 6. Qualitative comparison of our model with several label-to-image methods on the DeepFashion dataset.

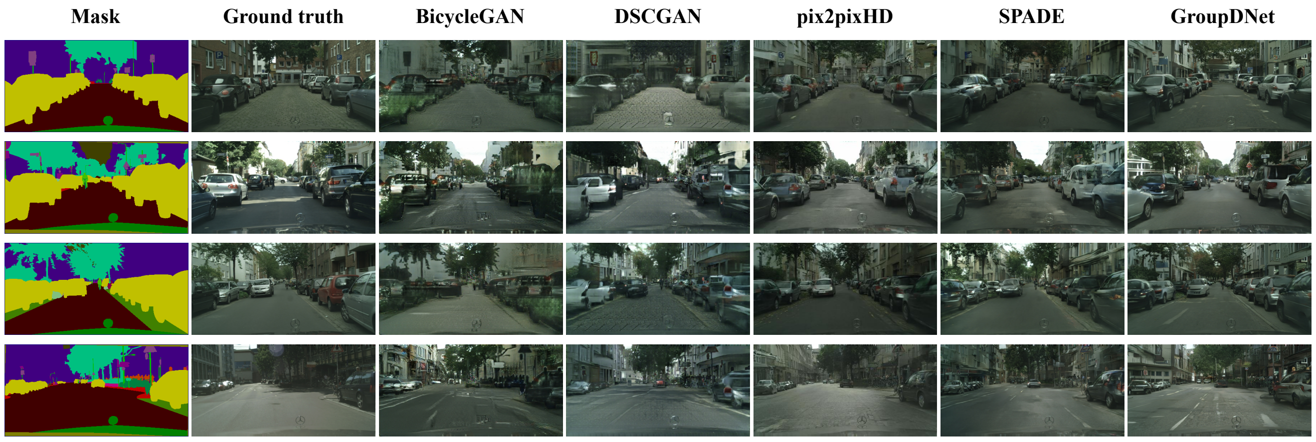


Figure 7. Qualitative comparison of our model with several label-to-image methods on the Cityscapes dataset.

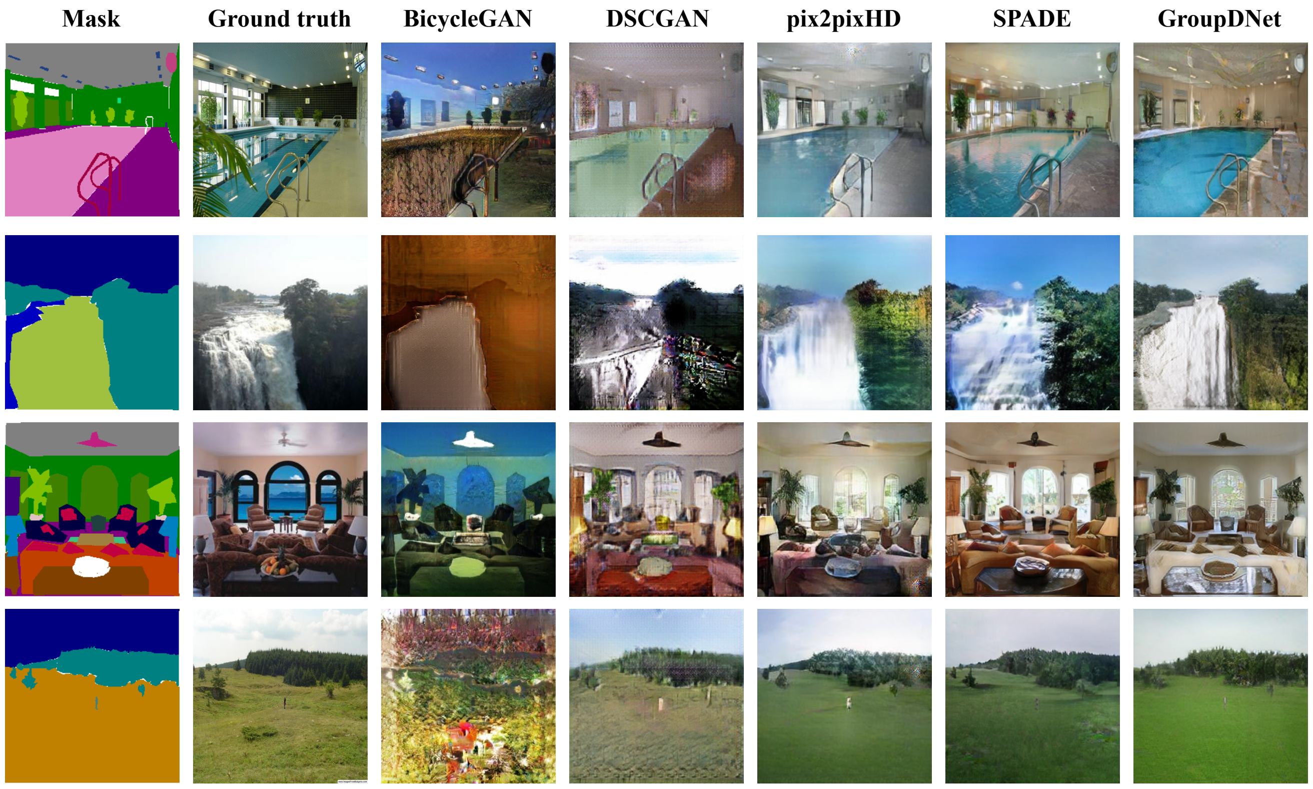


Figure 8. Qualitative comparison of our model with several label-to-image methods on the ADE20K dataset.