Supplementary Materials for Virtual Try-On with Pose-Garment Keypoints Guided Inpainting

Zhi Li[1,2] Pengfei Wei[1] Xiang Yin[1] Zejun Ma[1] Alex C. Kot[2]
[1]Bytedance Ltd. [2]Nanyang Technological University
{zhi.li.2023, pengfei.wei, yinxiang.stephen, mazejun}@bytedance.com eackot@ntu.edu.sg

1. Introduction

This document is a supplement to the article Virtual Try-On with Pose-Garment Keypoints Guided Inpainting (KGI). The supplementary materials include implementation details, illustrations and visualizations that are not included in the main paper due to the page limit. The checklist is shown below:

- results of user study.
- experiments of ablation study.
- structures of KGI submodules (including the pose-oriented garment keypots prediction model, the target segmentation map generation model, and the semantic conditioned inpainting model).
- hyper-parameter configurations for the model training.
- illustrations of the person image recomposition.
- visualizations of the experimental results.

2. User Study

Following [7], we conducted a user study to compare the results of our method with two baselines: VITON-HD [1] and HR-VITON [7].

There are 30 volunteers participating in the user study. The participants are given 30 sets of image samples, and each set contains the generated results of three methods and the target garment image for reference. In different sets, the orders of the generated images are shuffled. The users are asked to answer two questions: (1) Which image is the most photo-realistic? (2) Which image preserves the details of the given clothing the most?

The statistics of the user study are presented in Figure I. In 62.2% of total cases, the result of our method is chosen as the most photo-realistic one; In 80.1% of total cases, the result of our method is chosen as the one preserves the most clothing details. The results of user study show that our method performs better than baselines.

Method	SSIM↑	LPIPS↓	FID↓	KID↓
CP-VTON+ warping [8]	0.868	0.071	10.44	0.359
VITON-HD warping [1]	0.868	0.072	9.04	0.247
HR-VITON warping [7]	0.874	0.067	8.14	0.189
Our warping	**0.878**	**0.062**	**6.38**	**0.084**

Table I. Comparison of different warping methods.

3. Ablation Study

3.1. Ablation Study on Warping Methods

To verify the necessity of the proposed keypoints guided warping method, we kept the diffusion inpainting module unchanged and replaced our keypoints guided warping with warping results of three prior methods: CP-VTON+ [8] (TPS-based), VITON-HD [1](TPS-based), and HR-VITON [7](Flow-based). We conducted experiments at paired setting with 256×192 image resolution. From experimental results shown in Table I, we observed that our keypoints warping method contributes in improving performance and consistently outperforms prior TPS-based and Flow-based warping methods in terms of various evaluation metrics.

3.2. Ablation Study on Neck Segmentation

The generated images have similar collars with the source persons because we kept neck segmentation while generating the garment-agnostic segmentation map. To analyze the impact of keeping/removing neck segmentation, we conducted experiments for ablation study at both paired and unpaired setting with 1024×768 image resolution. The results in Table II show that keeping/removing neck segmentation does NOT significantly affect the quantitative results and our method performs better than baselines with either scheme. Besides, the FID and KID scores under unpaired setting are larger than paired setting with either scheme. It is reasonable as the generated images are paired with ground truth images (the same person wearing the same garment) in the paired setting, and thus has smaller data gap than the unpaired setting where the generated images and ground truth images are the same person with different garments.

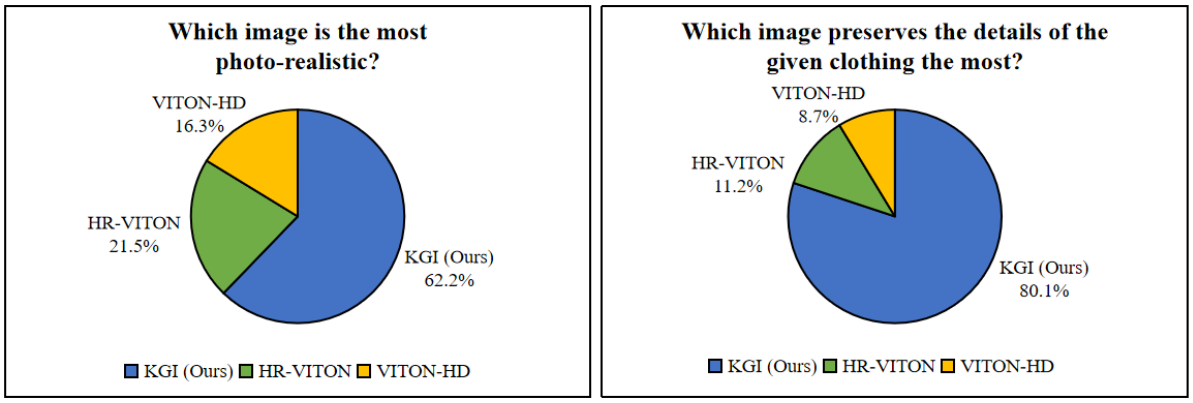

Figure I. User study results.

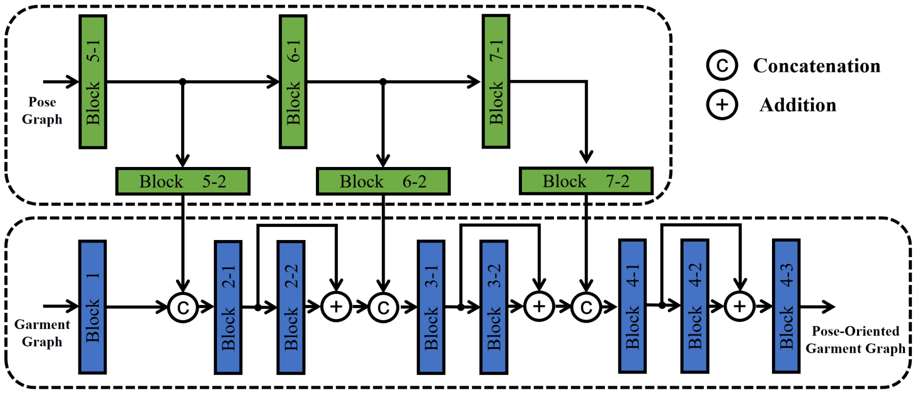

Figure II. Network Architecture of the Pose-Oriented Garment Keypoints Prediction Model.

Scheme	Paired Setting				Unpaired Setting	
	SSIM↑	LPIPS↓	FID↓	KID↓	FID↓	KID↓
keep neck	0.900	0.066	6.93	0.077	10.33	0.179
remove neck	0.900	0.067	6.96	0.078	10.00	0.137

Table II. Ablation study on neck segmentation.

4. Implementation Details of the Neural Network Models

In KGI, three neural network models are developed for pose-oriented garment keypoints prediction task, target segmentation map generation task and semantic conditioned inpainting task, respectively. We introduce the implementation details of these networks in this section.

4.1. Pose-Oriented Garment Keypoints Prediction Model

The Pose-Oriented Garment Keypoints Prediction Model is implemented with a two stream graph convolutional neural network as illustrated in Figure II, which consists of one garment graph embedding stream and one pose graph embedding stream. The garment graph embedding stream is sequentially built with 8 blocks. The skip connection is used to pass the residual information in some blocks. The blocks are constructed with SemGraphConv [17], BatchNorm [5], and ReLU [9] layers. The implementation details of the garment graph embedding blocks are provided in Table III. The pose graph embedding stream consists of network blocks. Three blocks are used for feature embedding and the others are employed for feature reshaping. The reshaped features are passed to the garment graph embedding stream to provide pose information as conditions. The implementation details of the pose graph embedding blocks are listed in Table IV.

In Tables III - IV, the parameters of the SemGraphConv layer are the number of input feature dimensions and the number of output feature dimensions, respectively. The values of the output shape are batch size, the number of nodes and the number of feature dimensions, from left to right.

Table III. Implementation Details of the Pose-Oriented Garment Keypoints Prediction Model (Garment Graph Embedding Stream)

Block	Layer	Parameters	Output Shape
1	SemGraphConv	[2, 160]	[B, 32, 160]
	BatchNorm	/	
	ReLU	/	
2-1	SemGraphConv	[320, 160]	[B, 32, 160]
	BatchNorm	/	
	ReLU	/	
2-2	SemGraphConv	[160, 160]	[B, 32, 160]
	BatchNorm	/	
	ReLU	/	
	SemGraphConv	[160, 160]	
	BatchNorm	/	
	ReLU	/	
3-1	SemGraphConv	[320, 160]	[B, 32, 160]
	BatchNorm	/	
	ReLU	/	
3-2	SemGraphConv	[160, 160]	[B, 32, 160]
	BatchNorm	/	
	ReLU	/	
	SemGraphConv	[160, 160]	
	BatchNorm	/	
	ReLU	/	
4-1	SemGraphConv	[320, 160]	[B, 32, 160]
	BatchNorm	/	
	ReLU	/	
4-2	SemGraphConv	[160, 160]	[B, 32, 160]
	BatchNorm	/	
	ReLU	/	
	SemGraphConv	[160, 160]	
	BatchNorm	/	
	ReLU	/	
4-3	SemGraphConv	[160, 2]	[B, 32, 2]
	Sigmoid	/	

Table IV. Implementation Details of the Pose-Oriented Garment Keypoints Prediction Model (Pose Graph Embedding Stream)

Block	Layer	Parameters	Output Shape
5-1	SemGraphConv	[2, 512]	[B, 10, 512]
	BatchNorm	/	
	ReLU	/	
5-2	Reshape	/	[B, 32, 160]
6-1	SemGraphConv	[512, 512]	[B, 10, 512]
	BatchNorm	/	
	ReLU	/	
6-2	Reshape	/	[B, 32, 160]
7-1	SemGraphConv	[512, 512]	[B, 10, 512]
	BatchNorm	/	
	ReLU	/	
7-2	Reshape	/	[B, 32, 160]

4.2. Target Segmentation Map Generation Model

The Target Segmentation Map Generation Model is implemented with a typical U-Net architecture [12], which consists of encoder blocks, transition blocks, decoder blocks, and output blocks, as shown in Figure III.

The encoder blocks are divided into 5 groups, and each group includes a strided convolutional block for downsampling, a residual block and an activation block. The residual block [3] is constructed with sequentially connected Conv2d, InstanceNorm [13], and ReLU layers. The implementation details of the encoder blocks are provided in Table V. The transition blocks include one 1x1 convolutional block, one residual block and one activation block for latent feature transition. The details about the transition blocks are listed in Table VI. Similar to the encoder blocks, the decoder blocks are grouped in five parts. Each part consists of an upsampling block, a residual block and an activation block. The upsampling block takes the output of the previous layer and also the embedding features of the corresponding encoder blocks passed via the skip connection. In the upsampling block, an upsampling function is used to increase the size of the input features. The details about the decoder blocks are shown in Table VII. The structure of the output blocks are similar to the transition blocks, as listed in Table VIII.

In Tables V-VIII, the parameters of the Conv2d layer are input channel, output channel, kernel size, stride, and padding, from left to right. For Upsample layers, the parameter two is the scale for upsampling. sThe values of the output shape column are batch size, output channel, height, and width, respectively. B, H, and W are notations referring to batch size, the height and width of the input image.

4.3. Semantic Conditioned Inpainting Model

The Semantic Conditioned Inpainting Model is implemented with a conditioned U-Net style neural network, which takes the image, timestep, and target segmentation map as inputs. As shown in Figure IV, the input image is fed into the forward stream of the network. The timestep embedding and the segmentation maps are used as conditions.

The network is constructed with encoder blocks, transition blocks, decoder blocks, the output block and the timestep embedding block. As shown in Table IX, the timestep embedding block [4] consists of TimestepEmbed, Linear, SiLU [2] layers, which is used to embed the input timestep of the diffusion process.

The encoder blocks are divided into 7 groups. Each group consists of several timestep embedding residual blocks. As shown in Figure V, the timestep embedding residual block is further divided into four sub-blocks: A, B, C, D. Block A is constructed with GroupNorm [16], SiLU [2] and Conv2d layers for feature embedding. AverPool2d

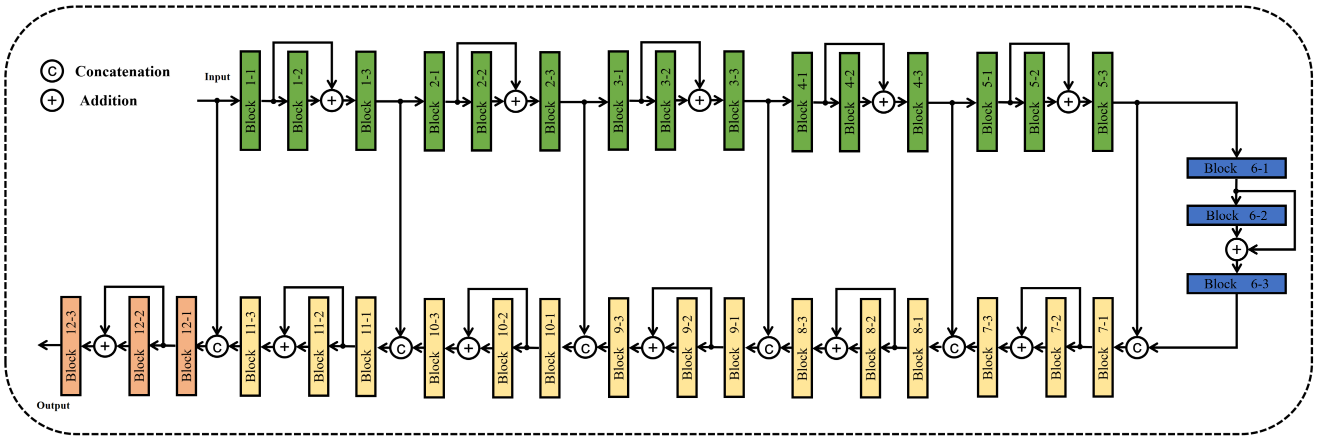

Figure III. Network Architecture of the Target Segmentation Map Generation Model.

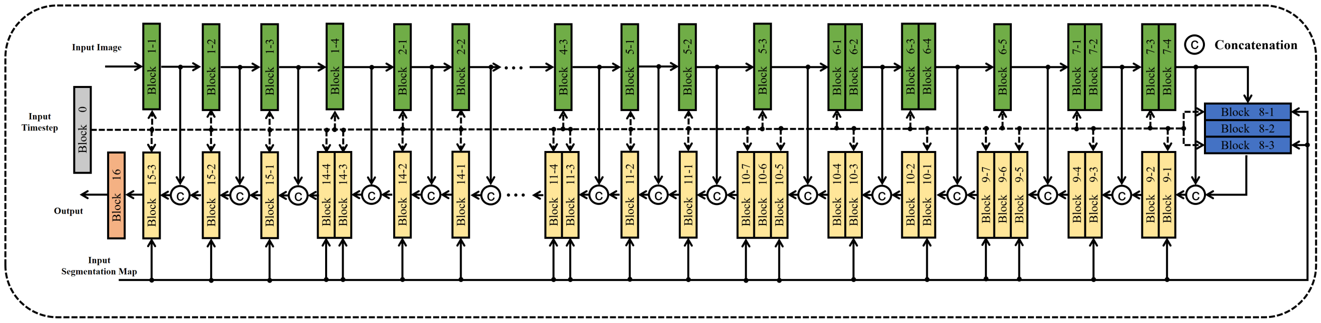

Figure IV. Network Architecture of the Semantic Conditioned Inpainting Model.

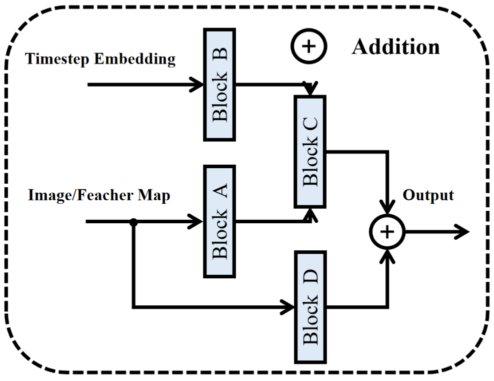

Figure V. Illustration of the Timestep Embedding Residual Block.

is used in the last timestep embedding residual block of each group for downsampling. Block B is constructed with SiLU [2] and Linear layers for timestep embedding. Block C is constructed with GroupNorm [16], SiLU [2], and Conv2d layers to fuse feature and timestep embedding. Block D is used for skip connection. If the input channel and output channel of the Block A are different, the Block D is implemented with 1x1 Conv2d layer. If the input size and output size of the Block A are different, the Block D is implemented with AverPool2d. In the group 6 and 7, attention blocks are added after the timestep embedding residual blocks. The details of encoder blocks are listed in Table (X, XI, XII, and XIII).

The transition blocks include two timestep embedding residual blocks and one attention block. Different from encoder blocks, the timestep embedding residual blocks take the target segmentation map as the condition and replace the conventional GroupNorm [16] layer with the SPADE-GroupNorm [10, 15] layer. The details are listed in Table XIV.

The decoder blocks are divided into 7 groups like encoder blocks. Each group contains several timestep embedding residual blocks and attention blocks [14] (group 9 and 10 only). The timestep embedding residual block takes the embedding of corresponding encoder blocks via the skip connection; the last timestep embedding residual block of each group contains upsampling layers in Block A and Block D to increase the size of feature maps. Simi-

Table V. Implementation Details of the Target Pose Generation Model (Encoder Blocks)

Block	Layer	Parameters	Output Shape
1-1	Conv2d	[19, 64, 3, 2, 1]	[B, 64, H/2, W/2]
1-2	Conv2d	[64, 64, 3, 1, 1]	[B, 64, H/2, W/2]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[64, 64, 3, 1, 1]	
	InstanceNorm	/	
1-3	ReLU	/	[B, 64, H/2, W/2]
2-1	Conv2d	[64, 128, 3, 2, 1]	[B, 128, H/4, W/4]
2-2	Conv2d	[128, 128, 3, 1, 1]	[B, 128, H/4, W/4]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[128, 128, 3, 1, 1]	
	InstanceNorm	/	
2-3	ReLU	/	[B, 128, H/4, W/4]
3-1	Conv2d	[128, 256, 3, 2, 1]	[B, 256, H/8, W/8]
3-2	Conv2d	[256, 256, 3, 1, 1]	[B, 256, H/8, W/8]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[256, 256, 3, 1, 1]	
	InstanceNorm	/	
3-3	ReLU	/	[B, 256, H/8, W/8]
4-1	Conv2d	[256, 256, 3, 2, 1]	[B, 256, H/16, W/16]
4-2	Conv2d	[256, 256, 3, 1, 1]	[B, 256, H/16, W/16]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[256, 256, 3, 1, 1]	
	InstanceNorm	/	
4-3	ReLU	/	[B, 256, H/16, W/16]
5-1	Conv2d	[256, 256, 3, 2, 1]	[B, 256, H/32, W/32]
5-2	Conv2d	[256, 256, 3, 1, 1]	[B, 256, H/32, W/32]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[256, 256, 3, 1, 1]	
	InstanceNorm	/	
5-3	ReLU	/	[B, 256, H/32, W/32]

Table VI. Implementation Details of the Target Segmentation Generation Model (Transition Block)

Block	Layer	Parameters	Output Shape
6-1	Conv2d	[256, 512, 1, 1, 0]	[B, 512, H/32, W/32]
6-2	Conv2d	[512, 512, 3, 1, 1]	[B, 512, H/32, W/32]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[512, 512, 3, 1, 1]	
	InstanceNorm	/	
6-3	ReLU	/	[B, 512, H/32, W/32]

lar to the transition blocks, the SPADEGroupNorm [10, 15] is adopted for segmentation map conditioning. The implementation details about the decoder blocks are listed in Table (XV, XVI, XVII, and XVIII).

The output block is constructed with sequentially connected GroupNorm [16], SiLU [2], and Conv2d layers, as presented in Table XIX.

In Tables IX-XIX, the parameters of the Conv2d layer are input channel, output channel, kernel size, stride, and padding, respectively. The parameters of the Linear layer are the input and output feature dimensions, respectively.

Table VII. Implementation Details of the Target Generation Model (Decoder Blocks)

Block	Layer	Parameters	Output Shape
7-1	Upsample	[2]	[B, 256, H/16, W/16]
	Conv2d	[512, 256, 1, 1, 0]	
7-2	Conv2d	[256, 256, 3, 1, 1]	[B, 256, H/16, W/16]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[256, 256, 3, 1, 1]	
	InstanceNorm	/	
7-3	ReLU	/	[B, 256, H/16, W/16]
8-1	Upsample	[2]	[B, 256, H/8, W/8]
	Conv2d	[512, 256, 1, 1, 0]	
8-2	Conv2d	[256, 256, 3, 1, 1]	[B, 256, H/8, W/8]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[256, 256, 3, 1, 1]	
	InstanceNorm	/	
8-3	ReLU	/	[B, 256, H/8, W/8]
9-1	Upsample	[2]	[B, 128, H/4, W/4]
	Conv2d	[512, 128, 1, 1, 0]	
9-2	Conv2d	[128, 128, 3, 1, 1]	[B, 128, H/4, W/4]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[128, 128, 3, 1, 1]	
	InstanceNorm	/	
9-3	ReLU	/	[B, 128, H/4, W/4]
10-1	Upsample	[2]	[B, 64, H/2, W/2]
	Conv2d	[256, 64, 1, 1, 0]	
10-2	Conv2d	[64, 64, 3, 1, 1]	[B, 64, H/2, W/2]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[64, 64, 3, 1, 1]	
	InstanceNorm	/	
10-3	ReLU	/	[B, 64, H/2, W/2]
11-1	Upsample	[2]	[B, 64, H, W]
	Conv2d	[128, 64, 1, 0, 0]	
11-2	Conv2d	[64, 64, 3, 1, 1]	[B, 64, H, W]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[64, 64, 3, 1, 1]	
	InstanceNorm	/	
11-3	ReLU	/	[B, 64, H, W]

Table VIII. Implementation Details of the Target Generation Model(Output Block)

Block	Layer	Parameters	Output Shape
12-1	Conv2d	[83, 64, 1, 1, 0]	[B, 64, H, W]
12-2	Conv2d	[64, 64, 3, 1, 1]	[B, 64, H, W]
	InstanceNorm	/	
	ReLU	/	
	Conv2d	[64, 64, 3, 1, 1]	
	InstanceNorm	/	
12-3	ReLU	/	[B, 13, H, W]
	Conv2d	[64, 13, 1, 1, 0]	

The parameters of the AvePool2d layer are kernel size and stride, respectively. For Upsample layers, the parameter 2 is the scale for upsampling. For GroupNorm layer, the parameter 30 is the number of channels. For SPADEGroupNorm, the parameters are the number of channels in group and the

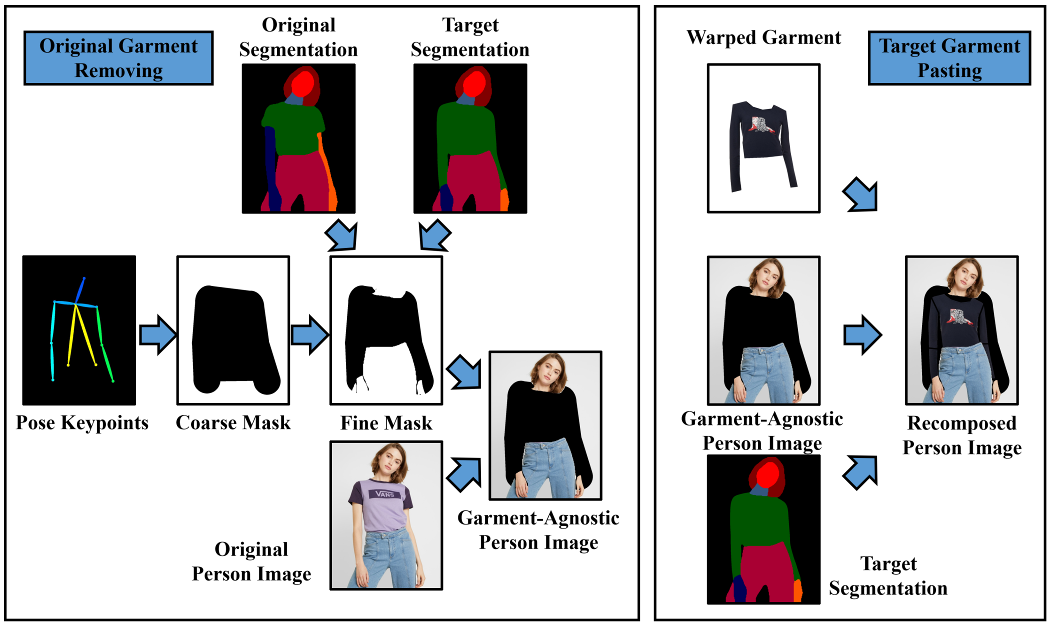

Figure VI. Illustration of the Person Image Recomposition.

number of classes in segmentation map. The values of the output shape column are batch size, output channel, height, and width, respectively. B, H, and W are notations referring to batch size, the height and width of the input image.

5. Hyper-Parameter Configurations for the Model Training

In experiments, we implement all the models using PyTorch [11] with version 1.10.0. The hyper-parameter configurations for model training are set as below.

For the Pose-Oriented Garment Keypoints Prediction model, we train the neural network for 300,000 iteration steps with a learning rate 1×10^{-3}, using the Adam optimizer [6]. The batch size is set as 16. The weights for the nodes term λ_N and edges term λ_E are set as 1 and 1, respectively.

For the Target Segmentation Generation model, we train the neural network for 2,000 iteration steps with a learning rate 2×10^{-4}, using the Adam optimizer [6] for optimization. The batch size is set as 4.

For the Semantic Conditioned Inpainting model, the network is trained for 300,000 iteration steps with a learning rate 1×10^{-4}. The Adam optimizer [6] is adopted for optimization. Due to the limitation of GPU resources, the batch size is set as 2, 2, and 1 for the experiments at 256×192, 512×384, and 1024×768 image resolution settings, respectively. The number of diffusion steps is set as $1,000$.

All the models are trained with the training split of the VITON-HD dataset [1]. No data augmentations or additional data are used for the model training.

6. Illustration of the Person Image Recomposition

As introduced in the the main paper, after the pose-oriented garment keypoints prediction and the target segmentation map generation, we recompose the given person image with the warped garment. Afterwards, the recomposed person image is treated as the incomplete try-on result to be inpainted.

As illustrated in Figure VI, the person image recomposition consists of two steps: original garment removing and target garment pasting.

The original garment removing step aims to remove the garment from the given person image and get a garment-agnostic person image. To ensure the original garment is fully removed, we follow [7] and use the pose keypoints extracted from the person image to draw a coarse mask where black regions refer to the regions to be removed. As shown in Figure VI, the coarse mask covers the region of human body, garments and the background. Since the task of virtual try-on is to replace the specific garment region only, a fine mask is then produced by pruning the coarse mask according to the original and target segmentation maps. Specifically, the regions have the consistent semantics (expect for the upper body garment and the background) in both segmentation maps are clipped from the coarse removing mask. The garment-agnostic person image is generated with the guide of the fine mask.

The target garment pasting step, just as its name implies, is to paste the warped garment onto the garment-agnostic person image. Similar to our TPS scheme, the garment pasting is performed in a divide and conquer manner. Five segments of garment image are sequentially pasted onto the garment-agnostic person image if the corresponding pixels in the target segmentation map are predicted as the target garment.

7. Visualizations of the Experimental Results

To demonstrate the effectiveness of our proposed method and the superiority compared to prior methods [1, 7], we include more visualization results for supplementary. Figure VII shows the examples generated with baselines and our KGI method. For each row, the images are given garment, given person, result of VITON-HD [1], result of HR-VITON [7], result of KGI (Ours), and a real image for reference. The results show that the try-on results generated with our KGI method have higher fidelity with more pattern details kept and less color and shape distortions.

For the ablation study experiments to study the necessity of different conditions and the impact of the number of diffusion step, we only include 1 example in the main paper due to the page limit. More visualizations of the ablation study experiments are provided in this Figure VIII and Figure IX. The Figure VIII illustrates the contributions of the warped garment and the target segmentation map. We can observe that the garment region of the results, without the warped garment as condition, are arbitrary and the textures are in monotonous color. Without the target segmentation map, the generation results present some semantic errors and the garment shapes are inconsistent with the ground truth. Figure IX shows the examples of generation results with different length of diffusion steps. With the increase of the number of the diffusion steps, the generation results are more realistic. Specifically, the textures of the inpainted garment region are more delicate and harmonious with the known pixels of the input image.

Figure X presents the visualization results of the pose-oriented garment keypoints prediction. From the left to right, the illustrations are given garment images, garment keypoints, pose keypoints, the predicted pose-oriented garment keypoints, the ground truth pose-oriented garment keypoints, and the given person images, respectively.

Since the semantic conditioned inpainting with diffusion models includes a random sampling process. To verify the stability of the generation, we visualize the generation results with different sampling noises, as shown in Figure XI. We find that our method can successfully and stably generate try-on results. The differences between the examples generated using different sampling noises are subtle.

References

[1] Seunghwan Choi, Sunghyun Park, Minsoo Lee, and Jaegul Choo. Viton-hd: High-resolution virtual try-on via misalignment-aware normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14131–14140, 2021. 1, 6, 7

[2] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. 3, 4, 5

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 3

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 2

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[7] Sangyun Lee, Gyojung Gu, Sunghyun Park, Seunghwan Choi, and Jaegul Choo. High-resolution virtual try-on with misalignment and occlusion-handled conditions. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pages 204–219. Springer, 2022. 1, 6, 7

[8] Matiur Rahman Minar, Thai Thanh Tuan, Heejune Ahn, Paul Rosin, and Yu-Kun Lai. Cp-vton+: Clothing shape and texture preserving image-based virtual try-on. In *CVPR Workshops*, volume 3, pages 10–14, 2020. 1

[9] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 2

[10] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019. 4, 5

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 3

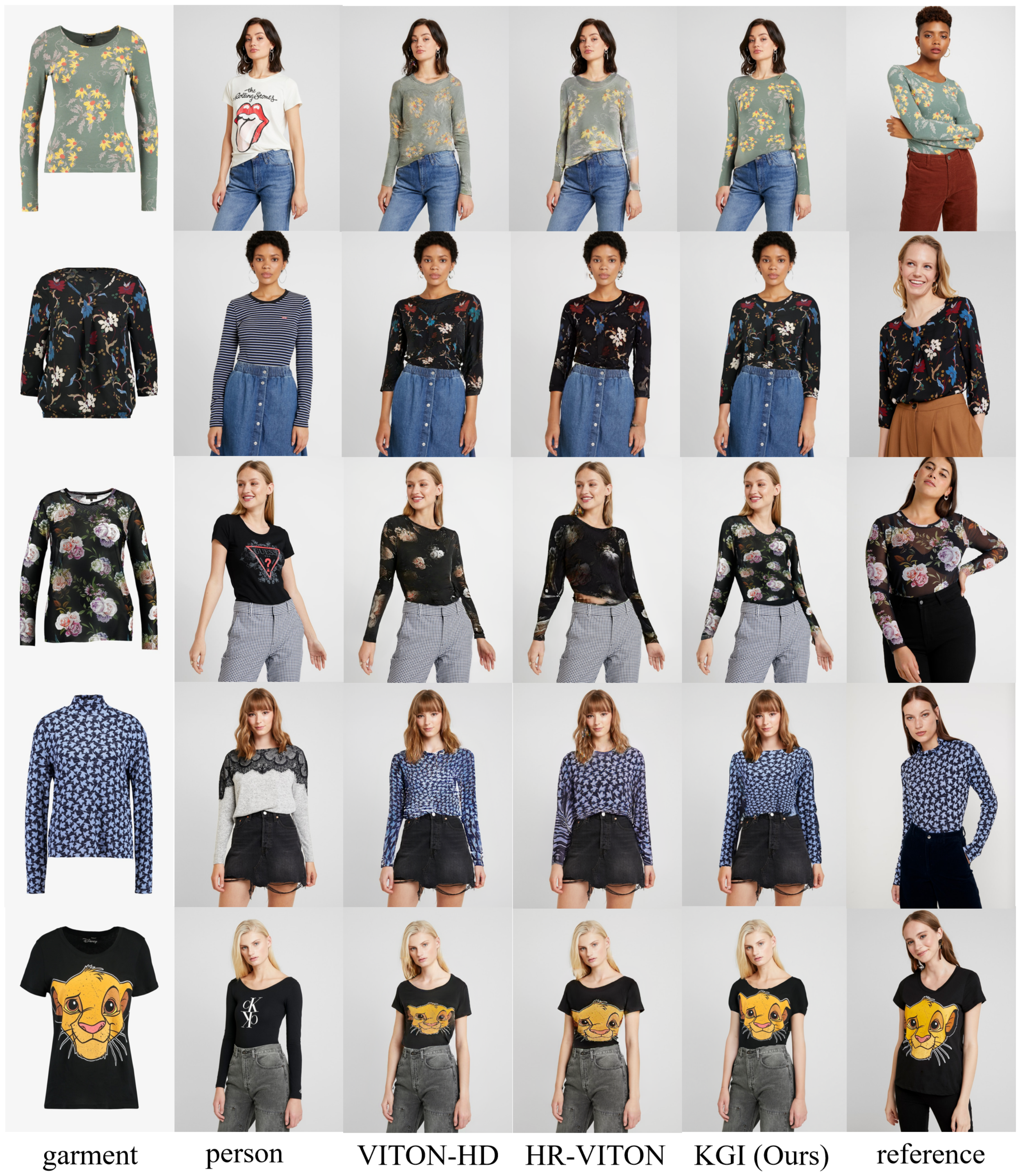

Figure VII. Examples of the Comparisons with Prior Methods. The examples are given garment images, given person images, the results of VITON-HD, the results of HR-VITON, the results of KGI (Ours), and a real image for reference.

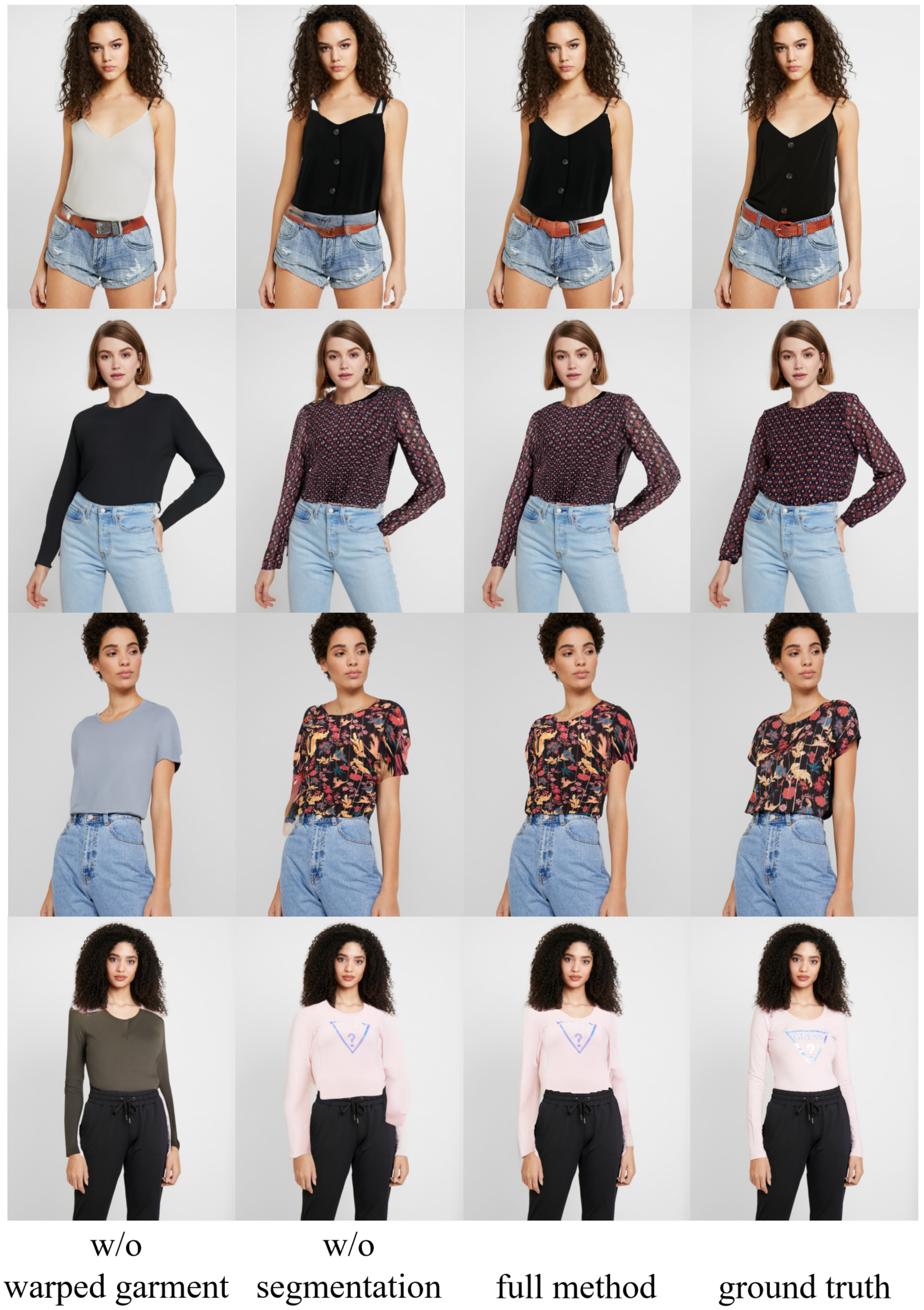

Figure VIII. Illustration of the Contributions of Warped Garment and Target Segmentation Map.

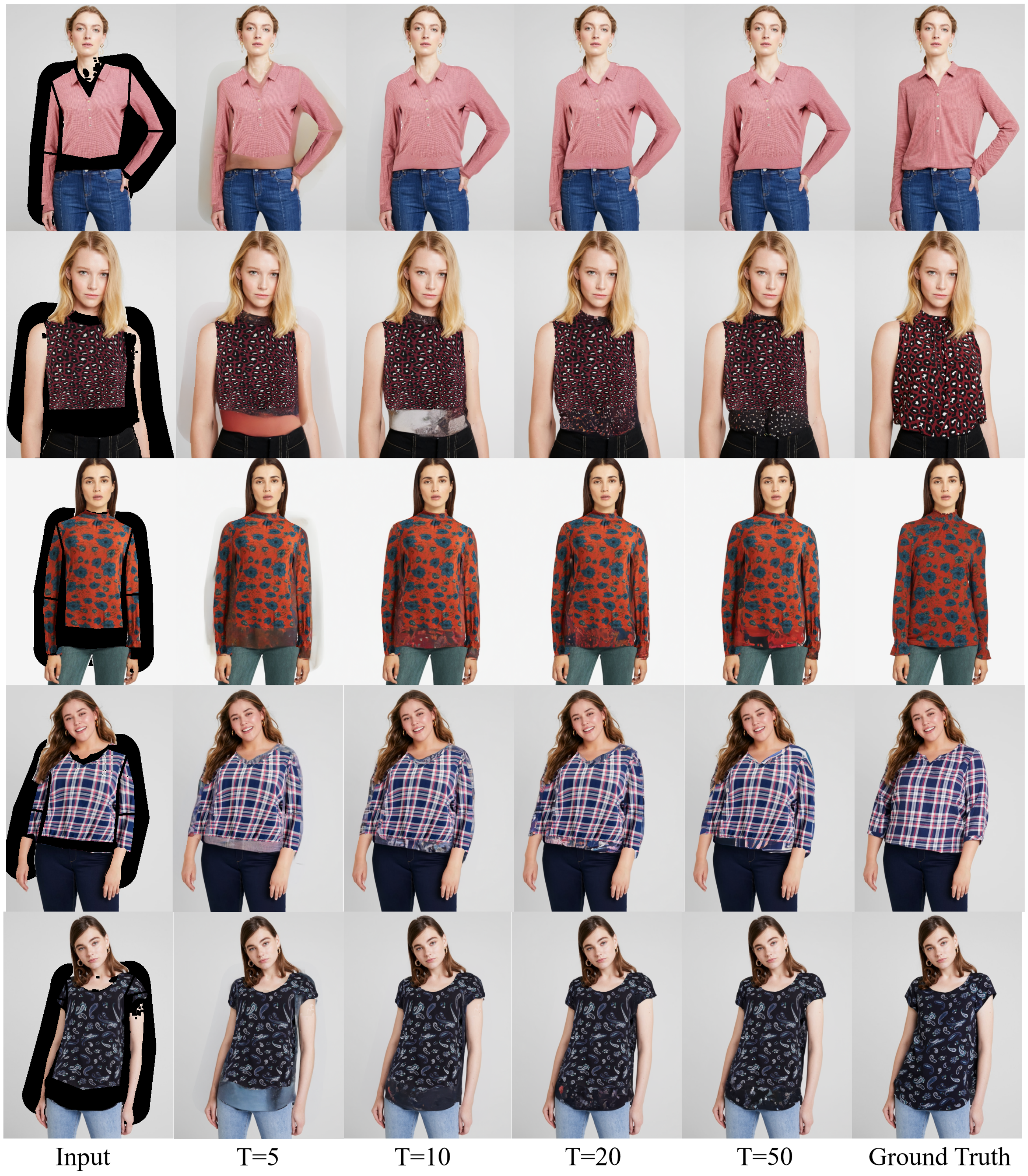

Figure IX. Examples of the Generation Results with Different Number of Diffusion Steps. The examples in Column 1 are input recomposed person images. The examples in Column 2-5 are generation results with 5, 10, 20, and 50 diffusion steps, respectively. The examples of the Column 6 are ground truth images.

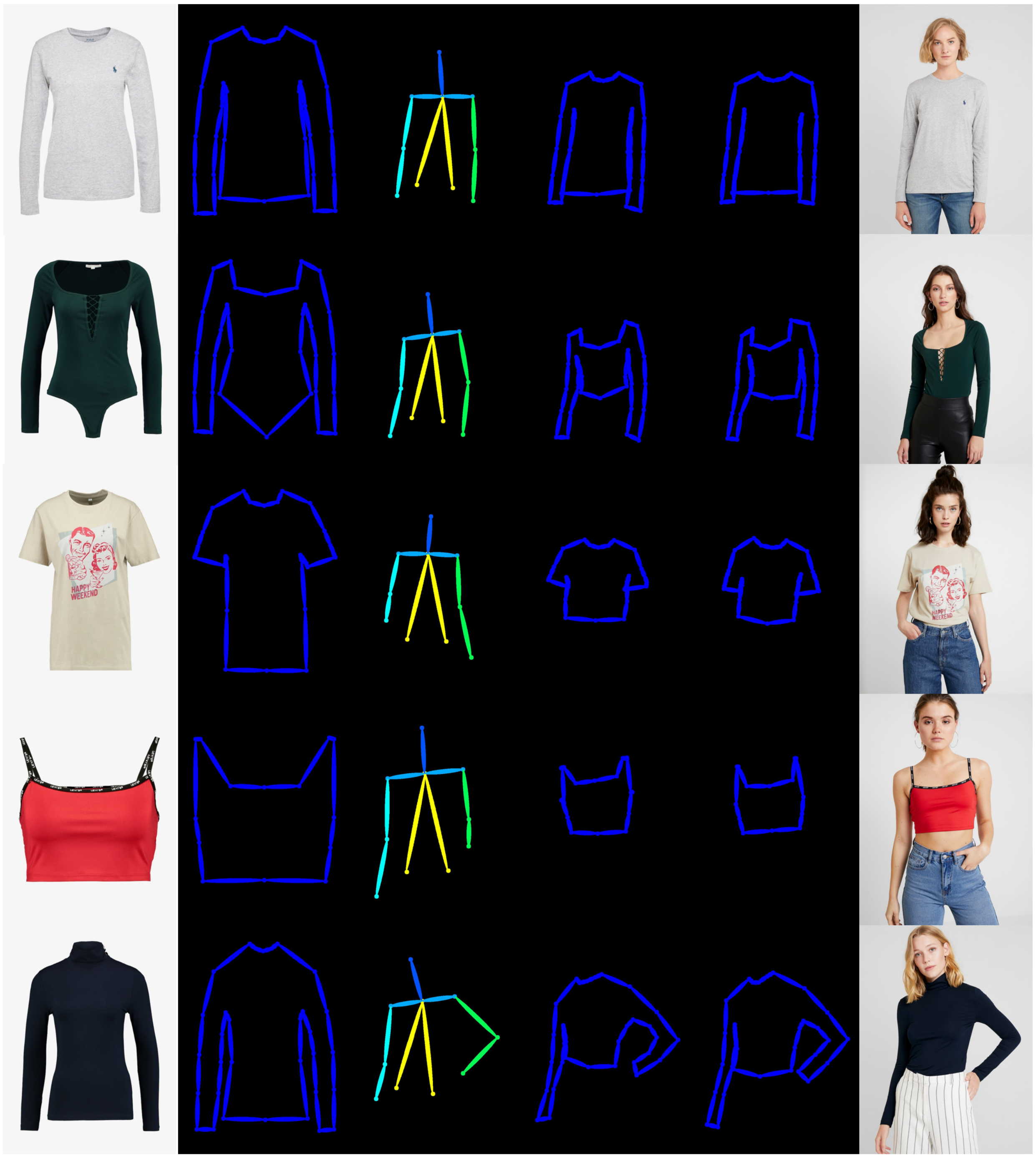
Garment Image
Gament Keypoints
Pose Keypoints
Predicted Pose-Oriented Garment Keypoints
Ground Truth Pose-Oriented Garment Keypoints
Person Image

Figure X. Visualizations of the Predicted Pose-Oriented Garment Keypoints.

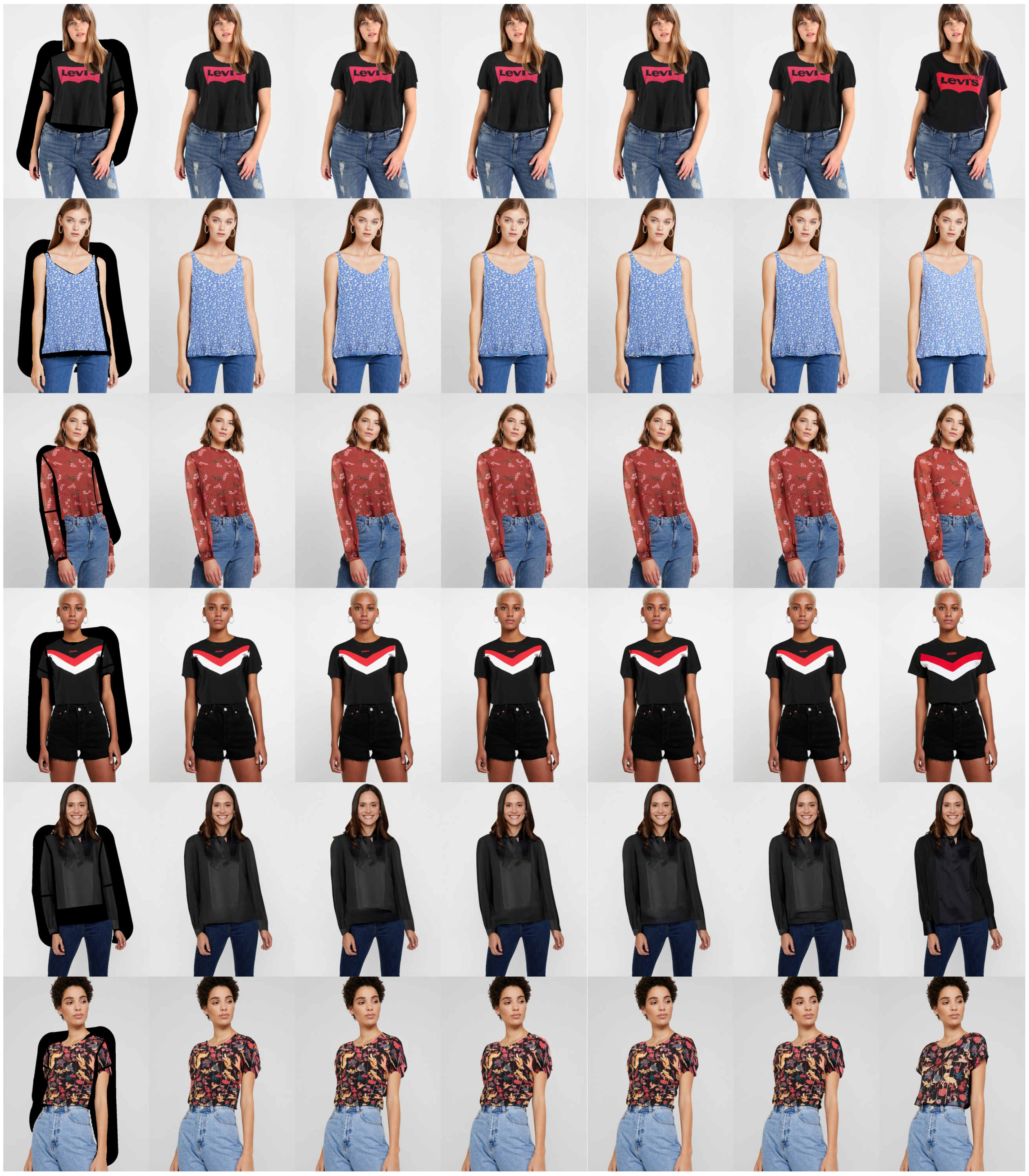

Figure XI. Illustration of the Stability of the Semantic Conditioned Inpainting.

Table IX. Implementation Details of the Semantic Conditioned Inpainting Model (Time Embedding Block)

Block		Layer	Parameters	Output Shape
0		TimestepEmbed	[128]	[B, 512]
		Linear	[128, 512]	
		SiLU	/	
		Linear	[512, 512]	

Table X. Implementation Details of the Semantic Conditioned Inpainting Model (Encoder Blocks Part I)

Block		Layer	Parameters	Output Shape
1-1		Conv2d	[3, 64, 3, 1, 1]	[B, 64, H, W]
1-2	A	GroupNorm	[32]	[B, 64, H, W]
		SiLU	/	
		Conv2d	[64, 64, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 128]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[64, 64, 3, 1, 1]	
	D	Identity	/	
1-3	A	GroupNorm	[32]	[B, 64, H, W]
		SiLU	/	
		Conv2d	[64, 64, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 128]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[64, 64, 3, 1, 1]	
	D	Identity	/	
1-4	A	GroupNorm	[32]	[B, 64, H/2, W/2]
		SiLU	/	
		AvePool2d	[2, 2]	
		Conv2d	[64, 64, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 128]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[64, 64, 3, 1, 1]	
	D	AvePool2d	[2, 2]	

[13] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 3

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

[15] Weilun Wang, Jianmin Bao, Wengang Zhou, Dongdong Chen, Dong Chen, Lu Yuan, and Houqiang Li. Semantic image synthesis via diffusion models. *arXiv preprint arXiv:2207.00050*, 2022. 4, 5

[16] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 3, 4, 5

[17] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. Semantic graph convolutional networks for 3d human pose regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3425–3435, 2019. 2

Table XI. Implementation Details of the Semantic Conditioned Inpainting Model (Encoder Blocks Part II)

Block		Layer	Parameters	Output Shape
2-1	A	GroupNorm	[32]	[B, 128, H/2, W/2]
		SiLU	/	
		Conv2d	[64, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Conv2d	[64, 128, 1, 1, 0]	
2-2	A	GroupNorm	[32]	[B, 128, H/2, W/2]
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Identity	/	
2-3	A	GroupNorm	[32]	[B, 128, H/4, W/4]
		SiLU	/	
		AvePool2d	[2, 2]	
		Conv2d	[128, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	AvePool2d	[2, 2]	
3-1	A	GroupNorm	[32]	[B, 128, H/4, W/4]
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Identity	/	
3-2	A	GroupNorm	[32]	[B, 128, H/4, W/4]
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Identity	/	
3-3	A	GroupNorm	[32]	[B, 128, H/8, W/8]
		SiLU	/	
		AvePool2d	[2, 2]	
		Conv2d	[128, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	AvePool2d	[2, 2]	

Table XII. Implementation Details of the Semantic Conditioned Inpainting Model (Encoder Blocks Part III)

Block		Layer	Parameters	Output Shape
4-1	A	GroupNorm	[32]	[B, 256, H/8, W/8]
		SiLU	/	
		Conv2d	[128, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Conv2d	[128, 256, 1, 1, 0]	
4-2	A	GroupNorm	[32]	[B, 256, H/8, W/8]
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Identity	/	
4-3	A	GroupNorm	[32]	[B, 256, H/16, W/16]
		SiLU	/	
		AvePool2d	[2, 2]	
		Conv2d	[256, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	AvePool2d	[2, 2]	
5-1	A	GroupNorm	[32]	[B, 256, H/16, W/16]
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Identity	/	
5-2	A	GroupNorm	[32]	[B, 256, H/16, W/16]
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Identity	/	
5-3	A	GroupNorm	[32]	[B, 256, H/32, W/32]
		SiLU	/	
		AvePool2d	[2, 2]	
		Conv2d	[256, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	AvePool2d	[2, 2]	

Table XIII. Implementation Details of the Semantic Conditioned Inpainting Model (Encoder Blocks Part IV)

Block		Layer	Parameters	Output Shape
6-1	A	GroupNorm	[32]	[B, 512, H/32, W/32]
		SiLU	/	
		Conv2d	[256, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Conv2d	[256, 512, 1, 1, 0]	
6-2		Attention	/	[B, 512, H/32, W/32]
6-3	A	GroupNorm	[32]	[B, 512, H/32, W/32]
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Identity	/	
6-4		Attention	/	[B, 512, H/32, W/32]
6-5	A	GroupNorm	[32]	[B, 512, H/64, W/64]
		SiLU	/	
		AvePool2d	[2, 2]	
		Conv2d	[512, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	AvePool2d	[2, 2]	
7-1	A	GroupNorm	[32]	[B, 512, H/64, W/64]
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Identity	/	
7-2		Attention	/	[B, 512, H/64, W/64]
7-3	A	GroupNorm	[32]	[B, 512, H/64, W/64]
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	GroupNorm	[32]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Identity	/	
7-4		Attention	/	[B, 512, H/64, W/64]

Table XIV. Implementation Details of the Semantic Conditioned Inpainting Model (Transition Block)

Block		Layer	Parameters	Output Shape
8-1	A	SPADEGroupNorm	[32, 13]	[B, 512, H/64, W/64]
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Identity	/	
8-2		Attention	/	[B, 512, H/64, W/64]
8-3	A	SPADEGroupNorm	[32, 13]	[B, 512, H/64, W/64]
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Identity	/	

Table XV. Implementation Details of the Semantic Conditioned Inpainting Model (Decoder Blocks Part I)

Block		Layer	Parameters	Output Shape
9-1	A	SPADEGroupNorm	[32, 13]	[B, 512, H/64, W/64]
		SiLU	/	
		Conv2d	[1024, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Conv2d	[1024, 512, 1, 1, 0]	
9-2		Attention	/	[B, 512, H/64, W/64]
9-3	A	SPADEGroupNorm	[32, 13]	[B, 512, H/64, W/64]
		SiLU	/	
		Conv2d	[1024, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Conv2d	[1024, 512, 1, 1, 0]	
9-4		Attention	/	[B, 512, H/64, W/64]
9-5	A	SPADEGroupNorm	[32, 13]	[B, 512, H/64, W/64]
		SiLU	/	
		Conv2d	[1024, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Conv2d	[1024, 512, 1, 1, 0]	
9-6		Attention	/	[B, 512, H/64, W/64]
9-7	A	SPADEGroupNorm	[32, 13]	[B, 512, H/32, W/32]
		SiLU	/	
		Upsample	[2]	
		Conv2d	[512, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Upsample	[2]	
10-1	A	SPADEGroupNorm	[32, 13]	[B, 512, H/32, W/32]
		SiLU	/	
		Conv2d	[1024, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Conv2d	[1024, 512, 1, 1, 0]	
10-2		Attention	/	[B, 512, H/32, W/32]
10-3	A	SPADEGroupNorm	[32, 13]	[B, 512, H/32, W/32]
		SiLU	/	
		Conv2d	[1024, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Conv2d	[1024, 512, 1, 1, 0]	
10-4		Attention	/	[B, 512, H/32, W/32]
10-5	A	SPADEGroupNorm	[32, 13]	[B, 512, H/32, W/32]
		SiLU	/	
		Conv2d	[768, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Conv2d	[768, 512, 1, 1, 0]	
10-6		Attention	/	[B, 512, H/32, W/32]
10-7	A	SPADEGroupNorm	[32, 13]	[B, 512, H/16, W/16]
		SiLU	/	
		Upsample	[2]	
		Conv2d	[512, 512, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 1024]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[512, 512, 3, 1, 1]	
	D	Upsample	[2]	

Table XVI. Implementation Details of the Semantic Conditioned Inpainting Model (Decoder Blocks Part II)

Block		Layer	Parameters	Output Shape
11-1	A	SPADEGroupNorm	[32, 13]	[B, 256, H/16, W/16]
		SiLU	/	
		Conv2d	[768, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Conv2d	[768, 256, 1, 1, 0]	
11-2	A	SPADEGroupNorm	[32, 13]	[B, 256, H/16, W/16]
		SiLU	/	
		Conv2d	[512, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Conv2d	[512, 256, 1, 1, 0]	
11-3	A	SPADEGroupNorm	[32, 13]	[B, 256, H/16, W/16]
		SiLU	/	
		Conv2d	[512, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Conv2d	[512, 256, 1, 1, 0]	
11-4	A	SPADEGroupNorm	[32, 13]	[B, 256, H/8, W/8]
		SiLU	/	
		Upsample	[2]	
		Conv2d	[256, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Upsample	[2]	
12-1	A	SPADEGroupNorm	[32, 13]	[B, 256, H/8, W/8]
		SiLU	/	
		Conv2d	[512, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Conv2d	[512, 256, 1, 1, 0]	
12-2	A	SPADEGroupNorm	[32, 13]	[B, 256, H/8, W/8]
		SiLU	/	
		Conv2d	[512, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Conv2d	[512, 256, 1, 1, 0]	
12-3	A	SPADEGroupNorm	[32, 13]	[B, 256, H/8, W/8]
		SiLU	/	
		Conv2d	[384, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Conv2d	[384, 256, 1, 1, 0]	
12-4	A	SPADEGroupNorm	[32, 13]	[B, 256, H/4, W/4]
		SiLU	/	
		Upsample	[2]	
		Conv2d	[256, 256, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 512]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[256, 256, 3, 1, 1]	
	D	Upsample	[2]	

Table XVII. Implementation Details of the Semantic Conditioned Inpainting Model (Decoder Blocks Part III)

Block		Layer	Parameters	Output Shape
13-1	A	SPADEGroupNorm	[32, 13]	[B, 128, H/4, W/4]
		SiLU	/	
		Conv2d	[384, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Conv2d	[384, 128, 1, 1, 0]	
13-2	A	SPADEGroupNorm	[32, 13]	[B, 128, H/4, W/4]
		SiLU	/	
		Conv2d	256, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Conv2d	[256, 128, 1, 1, 0]	
13-3	A	SPADEGroupNorm	[32, 13]	[B, 128, H/4, W/4]
		SiLU	/	
		Conv2d	[256, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Conv2d	[256, 128, 1, 1, 0]	
13-4	A	SPADEGroupNorm	[32, 13]	[B, 128, H/2, W/2]
		SiLU	/	
		Upsample	[2]	
		Conv2d	[128, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Upsample	[2]	
14-1	A	SPADEGroupNorm	[32, 13]	[B, 128, H/2, W/2]
		SiLU	/	
		Conv2d	[256, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Conv2d	[256, 128, 1, 1, 0]	
14-2	A	SPADEGroupNorm	[32, 13]	[B, 128, H/2, W/2]
		SiLU	/	
		Conv2d	[256, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Conv2d	[256, 128, 1, 1, 0]	
14-3	A	SPADEGroupNorm	[32, 13]	[B, 128, H/2, W/2]
		SiLU	/	
		Conv2d	[192, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Conv2d	[192, 128, 1, 1, 0]	
14-4	A	SPADEGroupNorm	[32, 13]	[B, 128, H, W]
		SiLU	/	
		Upsample	[2]	
		Conv2d	[128, 128, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 256]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[128, 128, 3, 1, 1]	
	D	Upsample	[2]	

Table XVIII. Implementation Details of the Semantic Conditioned Inpainting Model (Decoder Blocks Part IV)

Block		Layer	Parameters	Output Shape
15-1	A	SPADEGroupNorm	[32, 13]	[B, 64, H, W]
		SiLU	/	
		Conv2d	[192, 64, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 128]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[64, 64, 3, 1, 1]	
	D	Conv2d	[192, 64, 1, 1, 0]	
15-2	A	SPADEGroupNorm	[32, 13]	[B, 64, H, W]
		SiLU	/	
		Conv2d	128, 64, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 128]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[64, 64, 3, 1, 1]	
	D	Conv2d	[128, 64, 1, 1, 0]	
15-3	A	SPADEGroupNorm	[32, 13]	[B, 64, H, W]
		SiLU	/	
		Conv2d	[128, 64, 3, 1, 1]	
	B	SiLU	/	
		Linear	[512, 128]	
	C	SPADEGroupNorm	[32, 13]	
		SiLU	/	
		Conv2d	[64, 64, 3, 1, 1]	
	D	Conv2d	[128, 64, 1, 1, 0]	

Table XIX. Implementation Details of the Semantic Conditioned Inpainting Model (Output Block)

Block		Layer	Parameters	Output Shape
16		GroupNorm	[32]	[B, 6, H, W]
		SiLU	/	
		Conv2d	[64, 6, 3, 1, 1]	