## Contents

## A Poisson blending

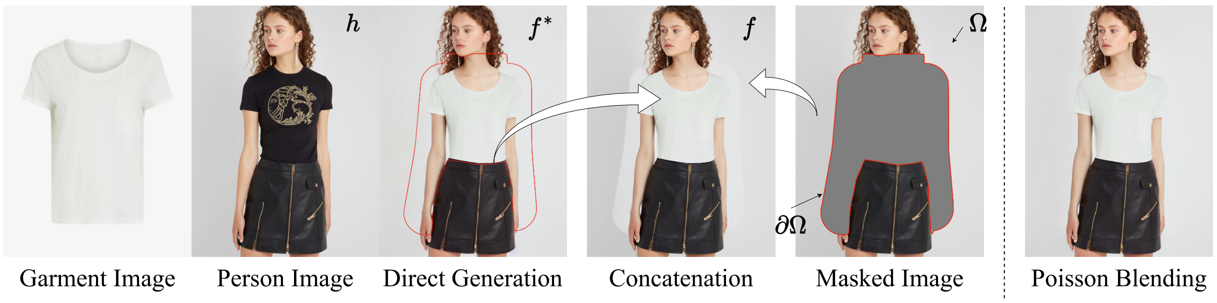


Figure 1: The image on the left provides a detailed explanation of Poisson blending in our model. The image on the right shows the result of using Poisson blending.

Since our model is trained in a latent space, it requires converting images from the latent space back into pixel space when generating try-on images. In this process, information loss, which is frequently encountered especially in complex regions like the human face, tends to result in the generation of images with noticeable artifacts.

As shown in Figure 1, we can concatenate the directly generated try-on image $f^*$ with the input person image $h$ by using the mask $m$ to address the issue of changes in the image outside the garment area. The non-clothing region is represented as $\Omega$. The image $f$ obtained through this method can be represented as follows:

$$f = \Omega + f^*(1-m) = h * m + f^*(1-m) \tag{1}$$

However, this method often results in noticeable discontinuities at the boundary. To address this, we employ Poisson blending to achieve a smoother and more coherent transition at the boundary. The core idea of this process is to modify the pixel values in the non-clothing region $\Omega$ so that the gradient of $f$ at the boundary of $\Omega$ closely approximates the gradient of $h$ at the boundary of $\Omega$.

$$\min_f \iint_\Omega \|\nabla f - \nabla h\|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}, \tag{2}$$

where $\nabla = [\frac{\partial}{\partial x}, \frac{\partial}{\partial y}]$ is the gradient operator, $\partial\Omega$ is the boundary of the non-clothing region $\Omega$.

For images, the problem can be discretized using the underlying discrete pixel grid to obtain a quadratic optimization problem.

$$\min_{f|\Omega} \sum_{\langle p,q\cap\Omega\neq\emptyset\rangle} (f_p - f_q - v_{pq})^2, \text{ with } f_p = f_p^* \text{ for all } p \in \partial\Omega, \tag{3}$$

where $N_p$ is the set of 4-connected neighbors for pixel $p$, $\langle p, q\rangle$ denote a pixel pair such that $q \in N_p$, $f_p$ is the value of $f$ at $p$ and $v_{pq} = h_p - h_q$ for all $\langle p, q\rangle$.

For discrete system, the solution can be converted into the following simultaneous linear equations. For $p \in \Omega$, we have

$$|N_p|f_p - \sum_{q\in N_p} f_q = \sum_{q\in N_p} v_{pq} \tag{4}$$

There is a subtle aspect that we designate the non-clothing area as the blending region $\Omega$. This is because we want the generated clothing area to remain unaffected.

## B Additional experimental details

### B.1 Model architecture

The architecture of GC-DM is shown in Figure 2. GC-DM comprises a fixed-parameter PBE and a trainable ControlNet. PBE utilizes an U-Net architecture, comprising multiple SD Encoder Blocks, several SD Decoder

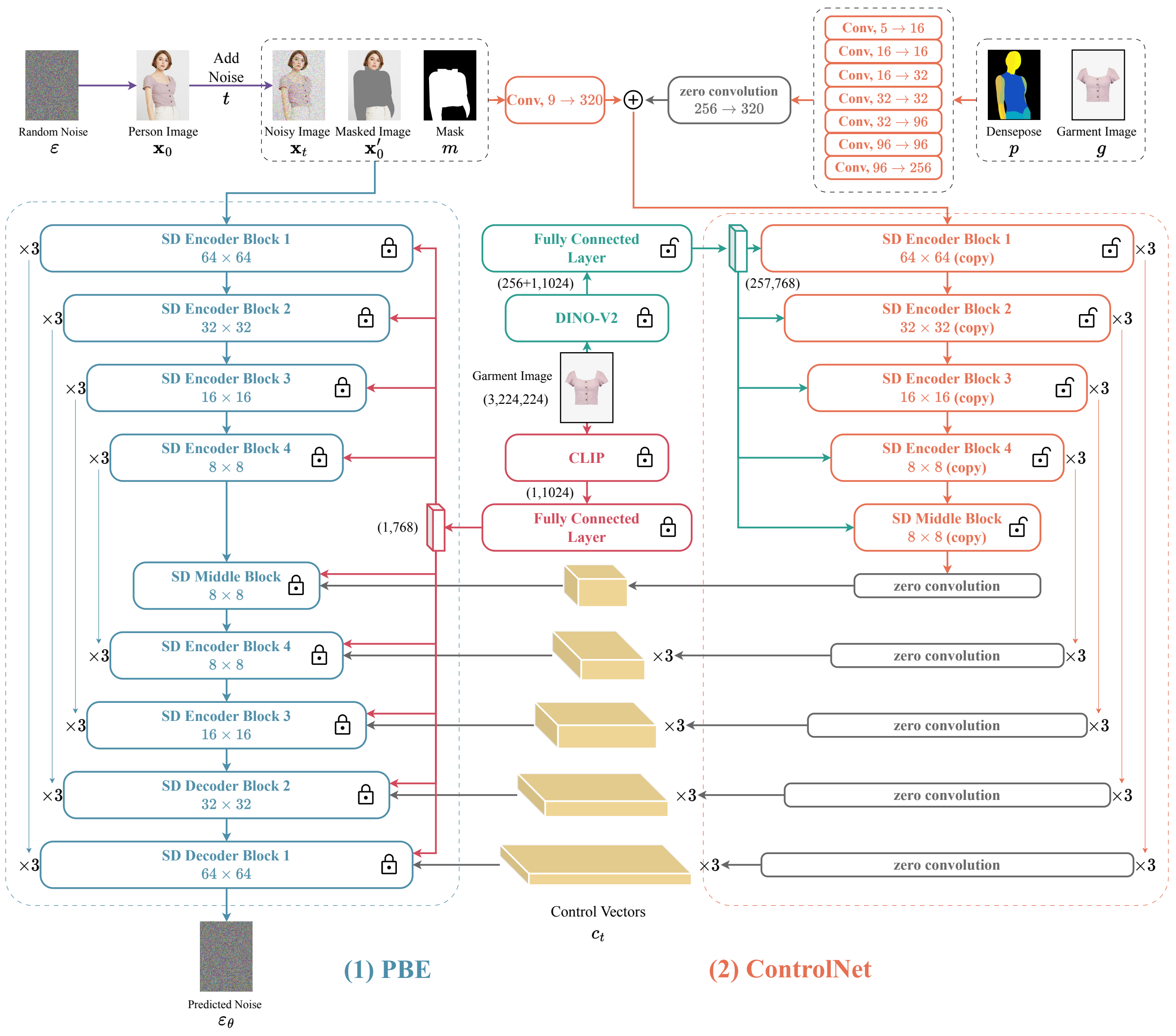

Random Noise $\varepsilon$
Person Image $\mathbf{x}_0$
Add Noise $t$
Noisy Image $\mathbf{x}_t$
Masked Image $\mathbf{x}_0'$
Mask $m$
Conv, 9 → 320
zero convolution 256 → 320
Conv, 5 → 16
Conv, 16 → 16
Conv, 16 → 32
Conv, 32 → 32
Conv, 32 → 96
Conv, 96 → 96
Conv, 96 → 256
Densepose $p$
Garment Image $g$
SD Encoder Block 1 64 × 64
SD Encoder Block 2 32 × 32
SD Encoder Block 3 16 × 16
SD Encoder Block 4 8 × 8
SD Middle Block 8 × 8
SD Decoder Block 2 32 × 32
SD Decoder Block 1 64 × 64
Fully Connected Layer
(256+1,1024)
DINO-V2
Garment Image (3,224,224)
CLIP
(1,1024)
(1,768)
(257,768)
SD Encoder Block 1 64 × 64 (copy)
SD Encoder Block 2 32 × 32 (copy)
SD Encoder Block 3 16 × 16 (copy)
SD Encoder Block 4 8 × 8 (copy)
SD Middle Block 8 × 8 (copy)
zero convolution
×3
Control Vectors $c_t$
Predicted Noise $\varepsilon_\theta$
(1) PBE
(2) ControlNet

Figure 2: Model architectures. GC-DM primarily consists of a locked-parameter PBE and a trainable ControlNet. Zero convolution is used in both the input and output sections of ControlNet to preserve the generative capabilities of the PBE.

Blocks, and an SD Middle Block. The SD Encoder Blocks and SD Decoder Blocks are interconnected via skip-connections. We lock all parameters of PBE and copy the parameters of SD Encoder Blocks and SD Middle Block to ControlNet. During the training process, we perform gradient updates exclusively on the parameters of ControlNet.

Assume that the parameters for PBE and ControlNet are denoted as $\theta_1$ and $\theta_2$, respectively. Given a noisy image $\mathbf{x}_t$, along with its corresponding masked image $\mathbf{x}_0'$, mask $m$ and garment image $g$, PBE is capable of predicting, to some extent, the noise added to $\mathbf{x}_t$ with

$$\varepsilon_{\theta_1} = f_{\theta_1}(\mathbf{x}_t \oplus \mathbf{x}_0' \oplus m, g, t), \tag{5}$$

where $\oplus$ denotes concatenation along the channel axis, and $\varepsilon_{\theta_1}$ represents the predicted noise by PBE.

In ControlNet, we employ a convolution layer known as "zero convolution" to process the inputs and outputs. Zero convolution is a $1 \times 1$ convolution layer with both weight and bias initialized with zeros. We denote the zero convolution in the input and output sections as $Z_1()$ and $Z_2()$, respectively. Consequently, the output of GC-DM is

$$\varepsilon_\theta = f_{\theta_1}(\mathbf{x}_t \oplus \mathbf{x}_0' \oplus m, g, t) + Z_2(f_{\theta_2}(\mathbf{x}_t \oplus \mathbf{x}_0' \oplus m + Z_1(p \oplus g))), \tag{6}$$

where $p$ represents the additional control conditions, such as densepose. The symbol $\theta$ denotes the total parameters of the GC-DM, which is $\{\theta_1, \theta_2\}$. Lastly, $\varepsilon_\theta$ signifies the noise predicted by the GC-DM.

Because both the weight and bias of a zero convolution layer are initialized as zeros, in the first training step, we have

$$\begin{cases} Z_1(p \oplus g) = 0 \\ f_{\theta_2}(\mathbf{x}_t \oplus \mathbf{x}'_0 \oplus m + Z_1(p \oplus g)) = f_{\theta_2}(\mathbf{x}_t \oplus \mathbf{x}'_0 \oplus m) = f_{\theta_1}(\mathbf{x}_t \oplus \mathbf{x}'_0 \oplus m) \\ Z_2(f_{\theta_2}(\mathbf{x}_t \oplus \mathbf{x}'_0 \oplus m + Z_1(p \oplus g))) = Z_2(f_{\theta_1}(\mathbf{x}_t \oplus \mathbf{x}'_0 \oplus m)) = 0 \end{cases} \tag{7}$$

By substituting Eq. 7 into Eq. 6, we can obtain the following result:

$$\varepsilon_{\theta_1} = \varepsilon_\theta. \tag{8}$$

Eq. 8 indicate that, in the first training step, the output of GC-DM is equivalent to the output of PBE. Compared to training the ControlNet network from scratch, this approach effectively retains the generative capabilities that PBE has learned from millions of images, and also offers a faster training speed.

It is important to note that we switch the garment feature extractor in ControlNet from CLIP to DINO-V2, while maintaining the CLIP component in PBE. This was done to prevent any disturbance to the weights of PBE. Additionally, we train a fully connected layer to encode garment features into the space where U-Net resides.

## B.2 Parameterization for GC-DM

As shown in Figure 3, our model primarily consists of five components: PBE, ControlNet, DINO-V2, CLIP, and AutoencoderKL (pre-trained encoder-decoder in LDMs). Although the model has a large number of parameters, the GPU memory usage is less compared to models like stable diffusion, because we only train the parameters of ControlNet. Our model can be trained on a graphics card like the NVIDIA GeForce RTX 4090 at a resolution of $512 \times 384$.

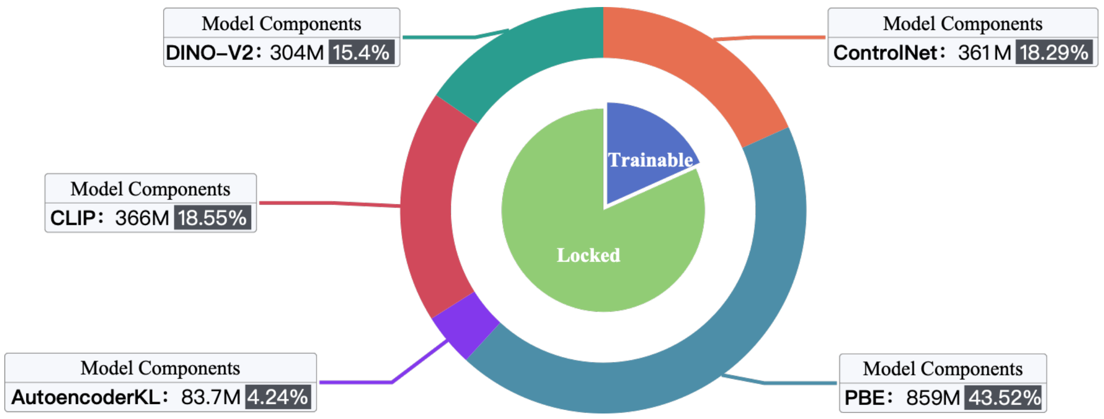


Figure 3: Parameterization for CAT-DM

## B.3 Training details

**Datasets:** In this work, we focus on evaluating virtual try-on tasks using two popular datasets: DressCode and VITON-HD. Both datasets contain high-resolution paired images of in-shop garments and their corresponding person images. For preprocessing, we employ OpenPose [1] to extract keypoints from the person images, utilize DensePose [2] for obtaining dense labels and UV mappings, and use SCHP [3] for semantic segmentation of the person images. Based on this preprocessed data, we generate a mask for each person image, allowing us to occlude the original garment without retaining any residual garment information from it. The DressCode dataset consists of 48,392 training image pairs and 5,400 test image pairs, distributed across three categories (upper-body, lower-body and dresses). On the other hand, the VITON-HD dataset contains 11,647 training and 2,032 test image pairs. The native resolution for both datasets is $1024 \times 768$, but for our experiments, we conduct them at a resolution of $512 \times 384$. In the training phase, we employe a self-supervised learning approach: given occluded person images and their corresponding in-shop garments images, the model is trained to reconstruct the original, unoccluded person images.

**Optimization:** All experiments are conducted using two NVIDIA GeForce RTX 4090 GPUs with image resolutions of $512 \times 384$. We use the AdamW optimizer, set the learning rate to $2 \times 10^{-5}$, batchsize to 4, and perform gradient accumulation every eight batches. In our experiments, we utilize xFormers [4] to optimize GPU memory utilization and accelerate computation. On the VITON-HD dataset, we train 200 epochs, each requiring half an hour of training; on the DressCode dataset, we train 80 epochs, each requiring two and a half hours of training.

**Diffusion schedule:** Our model is trained with $T = 1000$ noising steps and a linear noise schedule for $\overline{\alpha}_t$ ($t \in [1, ..., T]$). Our model employs a pre-trained encoder-decoder to transform images from pixel space to latent space, reducing both the width and height of the images by a factor of eight.

## B.4 Evaluation details

Fréchet Inception Distance (FID) and Kernel Inception Distance (KID) both employ deep neural networks to extract high-level features from the set of generated images and the set of real images, respectively. They then compute metrics by measuring the differences between these high-level features from both sets. During the metric calculation process, we use person images from the test set as the set of real images, and the generated try-on images constitute the set of generated images. Due to the variation in metric values with changes in image size [5], we uniformly use real images with a resolution of $1024 \times 768$. Besides, the KID is scaled by 1000.

For Learned Perceptual Image Patch Similarity (LPIPS) and Structural Similarity Index Measure (SSIM), we calculate these metrics exclusively in the paired setting. LPIPS and SSIM are both full-reference metrics. We calculate the scores between each generated image and its corresponding ground truth with a resolution of $512 \times 384$ individually. Then, we compute the average of these scores across the entire test set to obtain the final metric values.

The specific implementation of evaluation metrics can impact the final metric values. Therefore, we will provide details of the implementation we use.

- FID: we utilize the "pytorch-fid" implementation [6],
- KID: we utilize the "torch-fidelity" implementation [7];
- LPIPS: we utilize the "PerceptualSimilarity" implementation [8];
- SSIM: we utilize the "pytorch-ssim" implementation [9].

We use the official checkpoints to generate qualitative results. However, the method used to calculate evaluation metrics can significantly impact their final values. Taking $\text{FID}_\text{u}$ as an example, we use three different implementations, set the resolution of real images to $512 \times 384$ and $1024 \times 768$, and the results are shown in Table 1. Original papers use different methods for this calculation, leading to inconsistencies. To ensure a fair comparison, we adopt a uniform approach to calculate these metrics like previous methods.

| Method | $1024 \times 768$ | | | $512 \times 384$ | | | Original value | Original implementation |
|---|---|---|---|---|---|---|---|---|
| | Pytorch-FID | Clean-FID | Torch-Fidelity | Pytorch-FID | Clean-FID | Torch-Fidelity | | |
| VITON-HD | 14.64 | 17.07 | 14.74 | 11.81 | 11.82 | 11.86 | 14.05 | Pytorch-FID |
| HR-VTON | 12.15 | 12.45 | 12.15 | 12.06 | 12.25 | 12.07 | 9.90 | Unknown |
| GP-VTON | 10.49 | 9.68 | 10.54 | 10.92 | 11.00 | 11.28 | 9.20 | Unknown |
| PBE | 15.77 | 15.88 | 15.82 | 16.19 | 16.08 | 16.55 | - | Pytorch-FID |
| MGD | 13.34 | 12.95 | 13.29 | 11.69 | 11.68 | 11.60 | 12.81 | Clean-FID |
| LaDI-VTON | 12.33 | 9.32 | 12.49 | 13.12 | 12.94 | 13.71 | 9.41 | Clean-FID |
| DCI-VTON | 11.14 | 8.83 | 11.28 | 11.25 | 11.00 | 11.80 | 8.09 | Unkonwn |
| GC-DM | 9.67 | **8.63** | 9.75 | 10.94 | 10.79 | 11.13 | 9.67 | Pytorch-FID |
| CAT-DM | **8.93** | 8.65 | **9.08** | **9.52** | **9.76** | **10.07** | 8.93 | Pytorch-FID |

Table 1: $\text{FID}_\text{u}$ values with different implementations.

## C Algorithm

**Algorithm 1:** Training of GC-DM

```
for $k = 1$ to $K$ do
    Sample $\mathbf{x}_0 \sim \mathcal{X}$
    $\mathbf{x}'_0 = \mathbf{x}_0 \cdot m$
    $\mathbf{x}_0 \leftarrow \mathcal{E}(\mathbf{x}_0)$
    $\mathbf{x}'_0 \leftarrow \mathcal{E}(\mathbf{x}'_0)$
    $t \sim \text{Uniform}(\{1, ..., T\})$
    $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$
    $\mathbf{x}_t = \sqrt{\alpha}_t \mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon$
    $c_t = f_{\theta_2}(\mathbf{x}_t, \mathbf{x}'_0, m, g, t, p)$
    $\varepsilon_\theta = f_{\theta_1}(\mathbf{x}_t, \mathbf{x}'_0, m, g, t, c_t)$
    $l = \|\varepsilon - \varepsilon_\theta\|_2^2$
    $\theta_2 \leftarrow \text{optimizer}(\theta_2, l, \eta)$
end
Output: $f_{\theta_1}$ and $f_{\theta_2}$
```

**Algorithm 2:** Sampling of GC-DM

```
$\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
$\mathbf{x}'_0 = \mathbf{x}_0 \cdot m$
$\mathbf{x}'_0 \leftarrow \mathcal{E}(\mathbf{x}'_0)$
$t = T$
repeat
    $c_t = f_{\theta_2}(\mathbf{x}_t, \mathbf{x}'_0, m, g, t, p)$
    $\varepsilon_\theta = f_{\theta_1}(\mathbf{x}_t, \mathbf{x}'_0, m, g, t, c_t)$
    $s = t - T/N$
    $\mathbf{x}_s \leftarrow \sqrt{1-\overline{\alpha}_s}\varepsilon_\theta + \sqrt{\overline{\alpha}_s}\frac{\mathbf{x}_s - \sqrt{1-\overline{\alpha}_t}\varepsilon_\theta}{\sqrt{\overline{\alpha}_t}}$
    $t = s$
until $t < 1$
$\mathbf{x}_0 \leftarrow \mathcal{D}(\mathbf{x}_0)$
Output: $\mathbf{x}_0$
```

**Algorithm 3:** Training of CAT-DM

```
for $k = 1$ to $K$ do
    Sample $\mathbf{x}_0 \sim \mathcal{X}$
    $\mathbf{x}'_0 = \mathbf{x}_0 \cdot m$
    $\mathbf{x}_0 \leftarrow \mathcal{E}(\mathbf{x}_0)$
    $\mathbf{x}'_0 \leftarrow \mathcal{E}(\mathbf{x}'_0)$
    $t \sim \text{Uniform}(\{1, ..., T_{\text{trunc}}\})$
    $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$
    $\mathbf{x}_t = \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon$
    $c_t = f_{\theta_2}(\mathbf{x}_t, \mathbf{x}'_0, m, g, t, p)$
    $\varepsilon_\theta = f_{\theta_1}(\mathbf{x}_t, \mathbf{x}'_0, m, g, t, c_t)$
    $l = \|\varepsilon - \varepsilon_\theta\|_2^2$
    $\theta_2 \leftarrow \text{optimizer}(\theta_2, l, \eta)$
end
Output: $f_{\theta_1}$ and $f_{\theta_2}$
```

**Algorithm 4:** Sampling of CAT-DM

```
$\mathbf{x}'_0 = \mathbf{x}_0 \cdot m$
$\mathbf{x}_{T_{\text{trunc}}} = G_\phi(\mathbf{x}'_0, m, g, p)$
$\mathbf{x}'_0 \leftarrow \mathcal{E}(\mathbf{x}'_0)$
$\mathbf{x}_{T_{\text{trunc}}} \leftarrow \mathcal{E}(\mathbf{x}_{T_{\text{trunc}}})$
$t = T_{\text{trunc}}$
repeat
    $c_t = f_{\theta_2}(\mathbf{x}_t, \mathbf{x}'_0, m, g, t, p)$
    $\varepsilon_\theta = f_{\theta_1}(\mathbf{x}_t, \mathbf{x}'_0, m, g, t, c_t)$
    $s = t - T_{\text{trunc}}/N_{\text{trunc}}$
    $\mathbf{x}_s \leftarrow \sqrt{1-\overline{\alpha}_s}\varepsilon_\theta + \sqrt{\overline{\alpha}_s}\frac{\mathbf{x}_s - \sqrt{1-\overline{\alpha}_t}\varepsilon_\theta}{\sqrt{\overline{\alpha}_t}}$
    $t = s$
until $t < 1$
$\mathbf{x}_0 \leftarrow \mathcal{D}(\mathbf{x}_0)$
Output: $\mathbf{x}_0$
```

| Notations | Descriptions |
|---|---|
| $\mathcal{X}$ | Person image dataset |
| $\mathbf{x}_0$ | Person image |
| $\mathbf{x}'_0$ | Masked image |
| $\mathbf{x}_t$ | Noisy image |
| $m$ | Mask corresponding to the person image |
| $g$ | Garment image corresponding to the person image |
| $p$ | Densepose corresponding to the person image |
| $\mathcal{E}$ | Pre-trianed encoder of LDMs |
| $\mathcal{D}$ | Pre-trianed decoder of LDMs |
| $\theta_1$ | PBE model parameters |
| $f_{\theta_1}$ | PBE model |
| $\theta_2$ | ControlNet model parameters |
| $f_{\theta_2}$ | ControlNet model |
| $G_\phi$ | Pre-trained GAN-based model |
| $T$ | Discrete total time steps |
| $T_{\text{trunc}}$ | Truncation step |
| $N$ | Number of sampling steps |
| $N_{\text{trunc}}$ | Number of Truncated sampling steps |
| $t$ | Random time |
| $\overline{\alpha}_t$ | Pre-defined schedul |
| $c_t$ | Control vectors |
| $\varepsilon$ | Gaussian noise |
| $\varepsilon_\theta$ | Predicted noise |
| $K$ | Total number of optimization steps |
| $l$ | Loss |
| $\eta$ | Learning rate |

Table 2: Notions in GC-DM and CAT-DM

| Metrics | Steps | PBE | GC-DM with CLIP | | | GC-DM with DINO-V2 | | |
|---|---|---|---|---|---|---|---|---|
| | | | Direct Generation | Concatenation | Poisson Blending | Direct Generation | Concatenation | Poisson Blending |
| | 1 | 427.58 | 421.60 | 120.92 | 218.83 | 421.64 | **120.91** | 218.83 |
| | 2 | 304.49 | 105.20 | 75.81 | 84.51 | 100.73 | **70.69** | 78.32 |
| | 4 | 51.00 | 33.10 | 26.79 | 26.42 | 27.08 | 20.62 | **20.07** |
| FID$_u$ ↓ | 8 | 30.46 | 14.48 | 13.36 | 12.21 | 12.64 | 11.56 | **10.46** |
| | 16 | 15.77 | 11.37 | 11.22 | 10.21 | 10.76 | 10.57 | **9.67** |
| | 32 | 15.77 | 11.30 | 11.13 | 10.16 | 10.81 | 10.54 | **9.64** |
| | 64 | 16.08 | 11.58 | 11.32 | 10.36 | 10.86 | 10.56 | **9.70** |
| | 1 | 620.56 | 614.52 | **137.23** | 305.15 | 614.56 | **137.23** | 305.22 |
| | 2 | 421.93 | 111.88 | 72.08 | 84.89 | 105.60 | **65.83** | 77.39 |
| | 4 | 36.63 | 23.88 | 15.11 | 14.72 | 18.92 | 10.22 | **9.43** |
| KID$_u$ ↓ | 8 | 18.32 | 5.64 | 4.43 | 2.92 | 4.28 | 3.16 | **1.68** |
| | 16 | 6.22 | 2.97 | 3.13 | 1.77 | 2.53 | 2.59 | **1.36** |
| | 32 | 6.32 | 3.03 | 3.12 | 1.73 | 2.68 | 2.65 | **1.42** |
| | 64 | 6.57 | 3.20 | 3.20 | 1.85 | 2.72 | 2.63 | **1.42** |
| | 1 | 427.59 | 421.60 | **120.91** | 218.83 | 421.63 | **120.91** | 218.83 |
| | 2 | 305.62 | 103.37 | 74.60 | 82.34 | 99.11 | **69.57** | 76.84 |
| | 4 | 51.09 | 29.89 | 23.86 | 23.34 | 23.70 | 17.51 | **16.81** |
| FID$_p$ ↓ | 8 | 28.96 | 11.96 | 11.06 | 9.72 | 10.08 | 9.20 | **7.88** |
| | 16 | 14.32 | 9.03 | 9.02 | 7.90 | 8.25 | 8.18 | **7.11** |
| | 32 | 14.16 | 9.12 | 9.07 | 7.92 | 8.22 | 8.13 | **7.05** |
| | 64 | 14.23 | 9.18 | 9.08 | 7.95 | 8.29 | 8.12 | **7.09** |
| | 1 | 620.56 | 614.52 | **137.23** | 305.19 | 614.55 | **137.23** | 305.20 |
| | 2 | 424.76 | 109.38 | 70.45 | 82.38 | 102.10 | **64.17** | 74.73 |
| | 4 | 36.56 | 21.26 | 13.09 | 12.67 | 16.56 | 8.58 | **7.84** |
| KID$_p$ ↓ | 8 | 16.99 | 4.57 | 3.66 | 4.57 | 3.59 | 2.77 | **1.22** |
| | 16 | 5.44 | 2.40 | 2.75 | 1.38 | 2.09 | 2.42 | **1.12** |
| | 32 | 5.24 | 2.46 | 2.78 | 1.35 | 2.12 | 2.40 | **1.07** |
| | 64 | 5.31 | 2.52 | 2.79 | 1.39 | 2.21 | 2.39 | **1.09** |
| | 1 | 0.034 | 0.025 | **0.606** | 0.431 | 0.025 | **0.606** | 0.431 |
| | 2 | 0.341 | 0.641 | 0.710 | 0.709 | 0.656 | **0.729** | 0.728 |
| | 4 | 0.643 | 0.764 | 0.788 | 0.793 | 0.789 | 0.813 | **0.819** |
| SSIM$_p$ ↑ | 8 | 0.713 | 0.808 | 0.828 | 0.834 | 0.823 | 0.843 | **0.848** |
| | 16 | 0.763 | 0.829 | 0.848 | 0.853 | 0.835 | 0.854 | **0.862** |
| | 32 | 0.763 | 0.826 | 0.845 | 0.850 | 0.833 | 0.852 | **0.857** |
| | 64 | 0.762 | 0.824 | 0.843 | 0.848 | 0.831 | 0.850 | **0.855** |
| | 1 | 0.9061 | 0.8880 | **0.4191** | 0.6676 | 0.8879 | **0.4191** | 0.6675 |
| | 2 | 0.7071 | 0.4568 | 0.3230 | 0.3236 | 0.4354 | 0.2987 | **0.2964** |
| | 4 | 0.3571 | 0.2005 | 0.2008 | 0.1828 | 0.1668 | 0.1687 | **0.1490** |
| LPIPS$_p$ ↓ | 8 | 0.2785 | 0.1412 | 0.1413 | 0.1314 | 0.1209 | 0.1212 | **0.1113** |
| | 16 | 0.2254 | 0.1192 | 0.1159 | 0.1111 | 0.1069 | 0.1033 | **0.0988** |
| | 32 | 0.2263 | 0.1207 | 0.1175 | 0.1127 | 0.1077 | 0.1043 | **0.0997** |
| | 64 | 0.2255 | 0.1222 | 0.1189 | 0.1141 | 0.1090 | 0.1056 | **0.1010** |

Table 3: Statistical experimental results of GC-DM. We analyze various garment feature extractors and different generation processes, comparing them with the locked-parameter PBE model. The subscripts "u" and "p" respectively represent the unpaired setting and paired setting. Best results are reported with the highlighted sections in pink for emphasis.

## D Statistical experimental results

### D.1 Garment feature extraction

As a complement of Table 3 of the main paper, Table 3 presents the original experimental data of GC-DM using different garment feature extractors and different generation processes on the VITON-HD dataset. Under the same generation process, the GC-DM utilizing DINO-V2 shows better results across all six metrics. This suggests that providing the GC-DM with more detailed garment features can enhance its generative capabilities.
On the other hand, we can observe that compared to the original PBE model, the performance of GC-DM, even when using CLIP as the garment feature extractor, is significantly better. This indicates that ControlNet architecture is effective in adapting the PBE for virtual try-on tasks.

### D.2 Poisson blending

As shown in Table 3, compared to direct generation and concatenation, utilizing Poisson blending can effectively improve the quality of image generation. Therefore, GC-DM, which employs both Poisson blending and DINO-V2, often achieves the best performance (as shown in the last column of the table). At steps=1 and steps=2, the images obtained through concatenation are better. This is because the limited number of sampling steps results in

| Metrics | Steps | GC-DM | VITON-HD | HR-VITON | GP-VTON | GAT-DM with | | | GAT-DM with GP-VTON | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | VITON-HD | HR-VITON | GP-VTON | $T_{\text{trunc}} = 50$ | $T_{\text{trunc}} = 150$ |
| | 2 | 78.32 | | | | 11.22($\downarrow$ 3.42) | 10.18($\downarrow$ 1.97) | 8.93($\downarrow$ 1.56) | 8.52 | 8.52 |
| | 4 | 20.07 | | | | 11.42($\downarrow$ 3.22) | 11.13($\downarrow$ 1.02) | 9.39($\downarrow$ 1.10) | 8.83 | 9.46 |
| $\text{FID}_\text{u} \downarrow$ | 8 | 10.46 | 14.64 | 12.15 | 10.49 | 11.25($\downarrow$ 3.39) | 11.00($\downarrow$ 1.15) | 9.31($\downarrow$ 1.18) | 8.80 | 9.83 |
| | 16 | 9.67 | | | | 11.12($\downarrow$ 3.52) | 10.92($\downarrow$ 1.23) | 9.26($\downarrow$ 1.23) | 8.77 | 9.73 |
| | 32 | 9.64 | | | | 11.16($\downarrow$ 3.48) | 10.92($\downarrow$ 1.23) | 9.27($\downarrow$ 1.22) | 8.78 | 9.78 |
| | 2 | 77.39 | | | | 2.38($\downarrow$ 3.72) | 1.96($\downarrow$ 1.46) | 1.37($\downarrow$ 0.86) | 1.26 | 1.26 |
| | 4 | 9.43 | | | | 2.96($\downarrow$ 3.14) | 3.17($\downarrow$ 0.25) | 1.44($\downarrow$ 0.79) | 1.55 | 1.46 |
| $\text{KID}_\text{u} \downarrow$ | 8 | 1.68 | 6.10 | 3.42 | 2.23 | 2.85($\downarrow$ 3.25) | 3.10($\downarrow$ 0.32) | 1.41($\downarrow$ 0.82) | 1.53 | 1.74 |
| | 16 | 1.36 | | | | 2.75($\downarrow$ 3.35) | 3.03($\downarrow$ 0.39) | 1.38($\downarrow$ 0.85) | 1.50 | 1.69 |
| | 32 | 1.42 | | | | 2.78($\downarrow$ 3.32) | 3.05($\downarrow$ 0.37) | 1.38($\downarrow$ 0.85) | 1.52 | 1.72 |
| | 2 | 76.84 | | | | 9.14($\downarrow$ 3.67) | 7.88($\downarrow$ 2.04) | 5.60($\downarrow$ 2.11) | 5.60 | 5.60 |
| | 4 | 16.81 | | | | 9.71($\downarrow$ 3.10) | 8.88($\downarrow$ 1.04) | 6.61($\downarrow$ 1.10) | 5.94 | 6.69 |
| $\text{FID}_\text{p} \downarrow$ | 8 | 7.88 | 12.81 | 9.92 | 7.71 | 9.54($\downarrow$ 3.27) | 8.74($\downarrow$ 1.18) | 6.53($\downarrow$ 1.18) | 5.90 | 7.10 |
| | 16 | 7.11 | | | | 9.42($\downarrow$ 3.39) | 8.63($\downarrow$ 1.29) | 6.47($\downarrow$ 1.24) | 5.89 | 6.97 |
| | 32 | 7.05 | | | | 9.45($\downarrow$ 3.36) | 8.65($\downarrow$ 1.27) | 6.49($\downarrow$ 1.22) | 5.87 | 7.04 |
| | 2 | 74.73 | | | | 1.93($\downarrow$ 3.59) | 1.48($\downarrow$ 1.58) | 0.83($\downarrow$ 1.18) | 0.83 | 0.83 |
| | 4 | 7.84 | | | | 2.86($\downarrow$ 2.66) | 2.62($\downarrow$ 0.44) | 1.09($\downarrow$ 0.92) | 1.07 | 1.17 |
| $\text{KID}_\text{p} \downarrow$ | 8 | 1.22 | 5.52 | 3.06 | 2.01 | 2.76($\downarrow$ 2.76) | 2.55($\downarrow$ 0.51) | 1.06($\downarrow$ 0.95) | 1.06 | 1.40 |
| | 16 | 1.12 | | | | 2.67($\downarrow$ 2.85) | 2.48($\downarrow$ 0.58) | 1.04($\downarrow$ 0.97) | 1.04 | 1.34 |
| | 32 | 1.07 | | | | 2.71($\downarrow$ 2.81) | 2.49($\downarrow$ 0.57) | 1.04($\downarrow$ 0.97) | 1.04 | 1.40 |
| | 2 | 0.728 | | | | 0.847($\downarrow$ 0.001) | 0.863($\uparrow$ 0.003) | 0.877($\uparrow$ 0.020) | 0.877 | 0.877 |
| | 4 | 0.819 | | | | 0.858($\uparrow$ 0.010) | 0.871($\uparrow$ 0.011) | 0.882($\uparrow$ 0.025) | 0.880 | 0.882 |
| $\text{SSIM}_\text{p} \uparrow$ | 8 | 0.848 | 0.848 | 0.860 | 0.857 | 0.858($\uparrow$ 0.010) | 0.871($\uparrow$ 0.011) | 0.882($\uparrow$ 0.025) | 0.880 | 0.882 |
| | 16 | 0.862 | | | | 0.857($\uparrow$ 0.009) | 0.871($\uparrow$ 0.011) | 0.882($\uparrow$ 0.025) | 0.880 | 0.882 |
| | 32 | 0.857 | | | | 0.857($\uparrow$ 0.009) | 0.870($\uparrow$ 0.010) | 0.882($\uparrow$ 0.025) | 0.880 | 0.882 |
| | 2 | 0.2964 | | | | 0.1157($\downarrow$ 0.0059) | 0.0974($\downarrow$ 0.0064) | 0.0803($\downarrow$ 0.0094) | 0.0803 | 0.0803 |
| | 4 | 0.1490 | | | | 0.1155($\downarrow$ 0.0061) | 0.0971($\downarrow$ 0.0067) | 0.0794($\downarrow$ 0.0103) | 0.0786 | 0.0797 |
| $\text{LPIPS}_\text{p} \downarrow$ | 8 | 0.1113 | 0.1216 | 0.1038 | 0.0897 | 0.1150($\downarrow$ 0.0066) | 0.0967($\downarrow$ 0.0071) | 0.0790($\downarrow$ 0.0107) | 0.0785 | 0.0808 |
| | 16 | 0.0988 | | | | 0.1147($\downarrow$ 0.0069) | 0.0965($\downarrow$ 0.0073) | 0.0787($\downarrow$ 0.0110) | 0.0784 | 0.0802 |
| | 32 | 0.0997 | | | | 0.1147($\downarrow$ 0.0069) | 0.0965($\downarrow$ 0.0073) | 0.0787($\downarrow$ 0.0110) | 0.0784 | 0.0803 |

Table 4: Statistical experimental results of CAT-DM. We analyze various pre-trained GAN-based models and different trunncation steps on the VITON-HD dataset. We set the truncation step to 100 and conducte experiments on CAT-DM using different pre-trained GAN-based models. Then, we specifically employ GP-VTON as the pre-trained GAN-based model and compare the experimental results of CAT-DM with varying truncation steps. The subscripts "u" and "p" respectively represent the unpaired setting and paired setting.

directly generated images that are almost purely noise. Compared to concatenation, Poisson blending will affect the originally clear non-clothing areas with the generated garment areas. However, in practical scenarios, such a low number of sampling steps would not be used.

## D.3 Pre-trained GAN-based model

As a complement of Figure 9 and Figure 10 of the main paper, Table 4 presents the original experimental data of CAT-DM using different Pre-trained GAN-based models and different truncation steps $T_{\text{trunc}}$ on the VITON-HD dataset. In this section, CAT-DM uniformly uses DINO-V2 as the garment feature extractor and employs Poisson blending for processing the generated images. Additionally, we set the truncation step $T_{\text{trunc}}$ to 100. As shown in Figure 4, when the performance of GAN-based models is subpar, the initially generated samples are not accurate enough, which consequently affects the outcomes produced by CAT-DM. When the number of sampling steps is sufficient, CAT-DM, utilizing GP-VTON as its pre-trained GAN-based model, not only surpasses GP-VTON but also outperforms GC-DM.

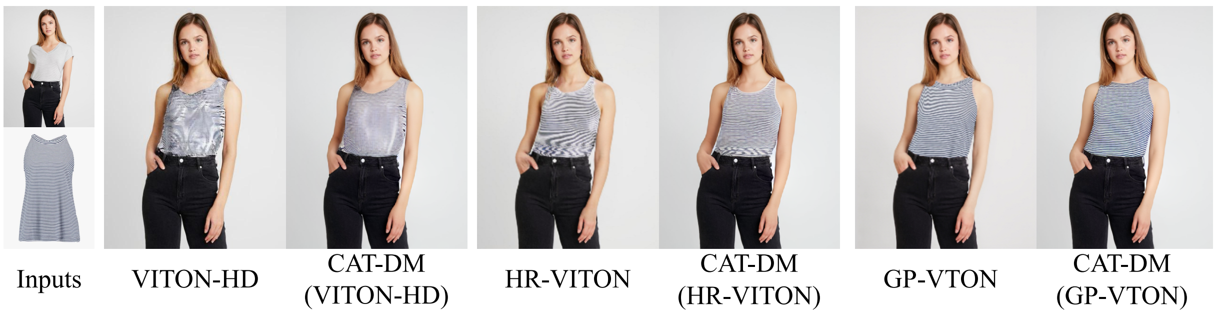


Figure 4: Visual results by integrating different pre-trained GAN-based models.

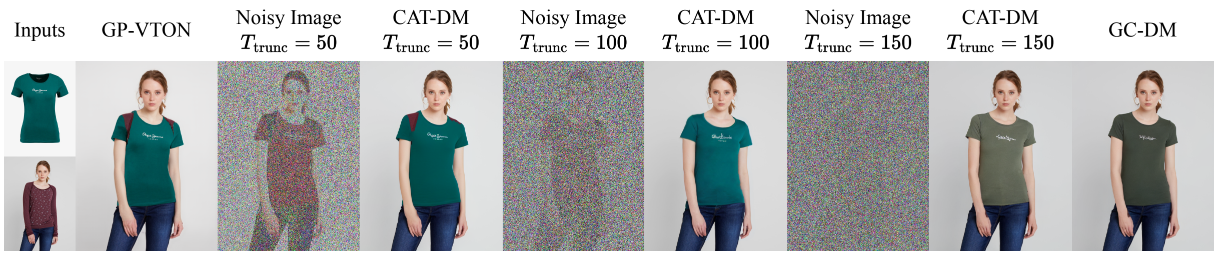

Inputs
GP-VTON
Noisy Image $T_{\text{trunc}} = 50$
CAT-DM $T_{\text{trunc}} = 50$
Noisy Image $T_{\text{trunc}} = 100$
CAT-DM $T_{\text{trunc}} = 100$
Noisy Image $T_{\text{trunc}} = 150$
CAT-DM $T_{\text{trunc}} = 150$
GC-DM

Figure 5: The impact of different truncation steps on the noisy images and generated image by CAT-DM.

In comparing performance metrics, GP-VTON typically exceeds HR-VITON, which in turn outperforms VITON-HD. Similarly, CAT-DM implementations using GP-VTON are generally more effective than those using HR-VITON, and these surpass the versions using VITON-HD. This pattern suggests that employing a higher-quality pre-trained GAN-based model in CAT-DM results in more precise initial try-on images, thus improving its ability to generate images. As shown in Figure 4, when the performance of GAN-based models is subpar, the initially generated samples are not accurate enough, which consequently affects the outcomes produced by CAT-DM. When the number of sampling steps is sufficient, CAT-DM, utilizing GP-VTON as its pre-trained GAN-based model, not only surpasses GP-VTON but also outperforms GC-DM.

Compared to GC-DM, CAT-DM requires significantly fewer sampling steps, indicating that CAT-DM can accelerate the inference speed of diffusion models. Compared to the pre-trained GAN-based model it utilizes, CAT-DM can enhance the quality of generated images by leveraging the generative capabilities of diffusion models. In Table 4, we highlight the performance gains of CAT-DM compared to the pre-trained GAN-based model it utilizes.

CAT-DM in virtual try-on tasks combines the advantages of GANs and diffusion models. We discover that providing GC-DM with a relatively accurate initial try-on result, which includes the garment pattern, effectively enhances the controllability of the diffusion model. Reflected in performance metrics, CAT-DM, which uses GP-VTON as the pre-trained GAN-based model, not only surpasses GP-VTON but also outperforms GC-DM.

### D.4 Truncation step

Furthermore, we also compare the impact of different truncation steps on CAT-DM. Our choice of truncation step is based on observations from Figure 5. When the truncation step is set to 50, the image after noise addition still retains most of the visual features of the person image. However, at a truncation step of 150, the image after noise addition loses most of the original image's information. Typically, the larger the truncation step, the smaller the impact of the pre-trained GAN-based model on the generated image; conversely, the smaller the truncation step, the greater the influence of pre-trained GAN-based model on the generated image. On one hand, when $T_{\text{trunc}}$ set to 0, CAT-DM and GP-VTON are essentially the same model. On the other hand, when $T_{\text{trunc}}$ is set to 1000, CAT-DM and GC-DM become identical models.

From Table 4, we can understand that different truncation steps do not significantly impact the quality of image generation and the model tends to perform best when the number of sampling steps is 2.

## E Additional qualitative comparison results

### E.1 Results on VITON-HD

We will present additional comparative experimental results on the VITON-HD dataset. Our comparators include GP-VTON, PBE, MGD, LaDI-VTON, and DCI-VTON. Our CAT-DM uses GP-VTON as the pre-trained GAN-based model, with the sampling step set to 2, and the truncation step set to 100.

PBE solely relies on CLIP as a garment feature extractor, thus it can only be semantically controlled. MGD utilizes multimodal data, such as text and garment sketches, to guide the generation of images. However, these data are incapable of accurately representing the visual characteristics of garments. LaDI-VTON employs textual inversion to extract visual information about garments, but this method still relies on the accuracy of the provided text and falls short in accurately capturing garment patterns. DCI-VTON involves the process of overlaying the flowed garment onto the masked image to guide the diffusion model. However, a drawback of this approach is that the generated results often retain some of the original garment's category-specific characteristics, such as its length and style. GP-VTON takes a different approach by independently warping garment using local flows and then assembling the local flow results through global garment parsing. While GP-VTON often succeeds in generating accurate garment patterns compared to other methods, its results may not always appear entirely realistic, especially when dealing with complex poses.

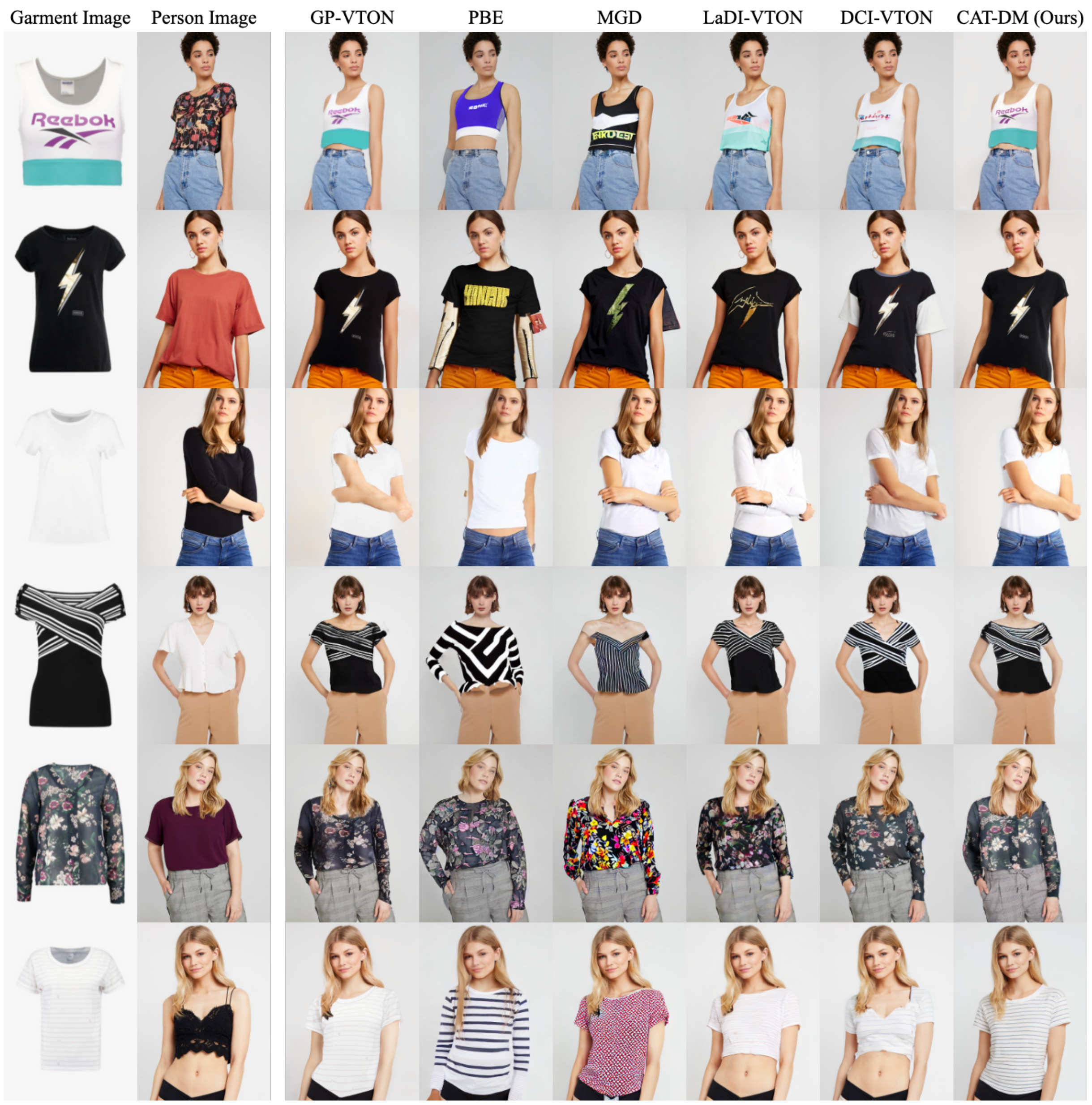


Figure 6: Additional experimental results on the VITON-HD dataset.

CAT-DM uses GP-VTON as the pre-trained GAN-based model and further leverages the diffusion model to optimize GP-VTON's generated results. As shown in Figure 6, compared to other methods, our model exhibits superior image realism and garment accuracy.

### E.2 Results on DressCode

We also present additional comparative experimental results on the DressCode dataset. Figure 7, Figure 8 and Figure 9 respectively present the comparative experimental results for DressCode's three categories: upper-body, lower-body and dresses. GC-DM achieves the best results in each category of the DressCode dataset.

## F Limitations

Although CAT-DM can efficiently and accurately generate realistic try-on images, our method still has some shortcomings. CAT-DM relies on the precision of the mask, which needs to completely cover the garment area in the input person image without obscuring unrelated regions. If the mask is too small and cannot fully cover

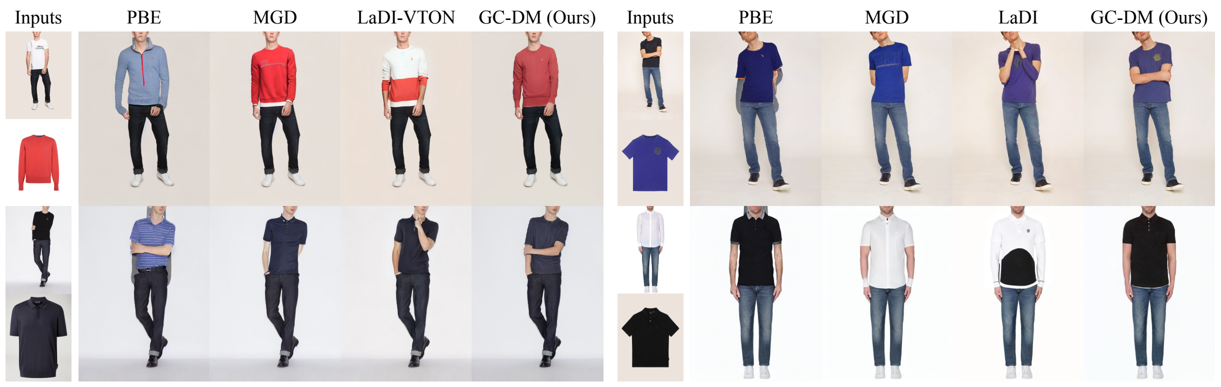


Figure 7: Additional experimental results for the upper-body category on the DressCode dataset.

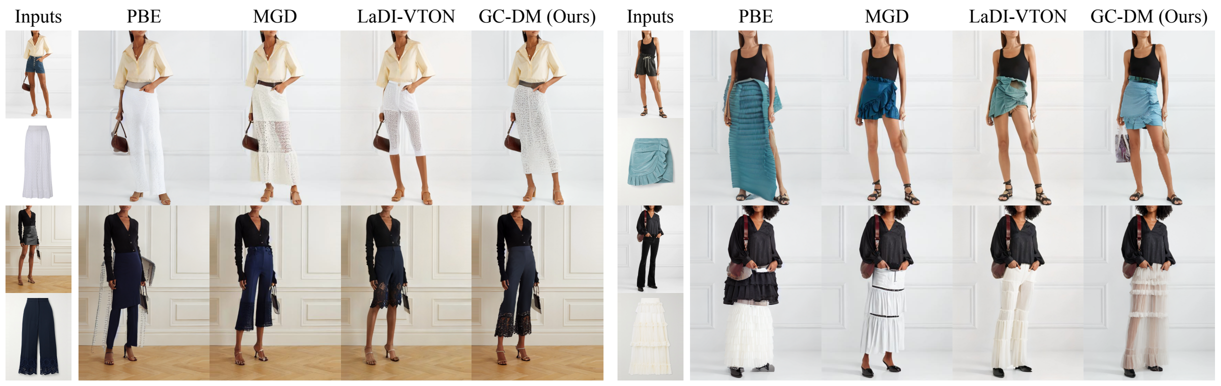


Figure 8: Additional experimental results for the lower-body category on the DressCode dataset.

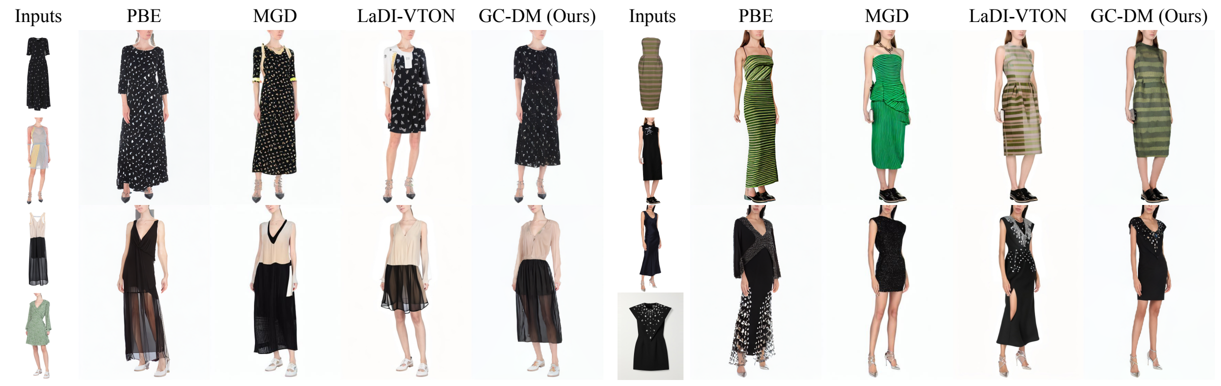
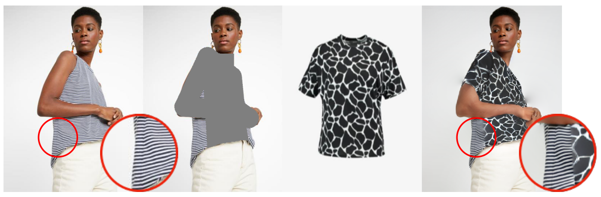


Figure 9: Additional experimental results for the dresses category on the DressCode dataset.

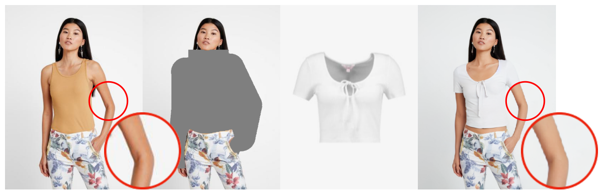
the original garment, as illustrated in Figure 10a, the generated try-on image may still show traces of the original garment. However, a larger mask is not always better. When the mask area is too large, the generated try-on image may not accurately reproduce the features of the original person image. As shown in Figure 10b, the color of the arms in the generated image has changed compared to the arm color in the input person image.

Although CAT-DM significantly enhances the controllability of diffusion models, it struggles to accurately preserve every detail on garment with relatively small and complex patterns, as shown in Figure 11. We believe this is due to CAT-DM's reliance on a pre-trained encoder-decoder of LDMs, which results in the information loss when converting images from latent space to pixel space. Although training CAT-DM directly in pixel space might improve this issue, it would require substantial computational resources.

Besides, while the try-on results generated by GC-DM is more realistic than the state-of-the-art baselines, the generated color in the try-on result is not consistent with the input clothes. As shown in Figure 12,



(a) The mask area is too small.



(b) The mask area is too large.

Figure 10: Failure cases of CAT-DM with inaccurate mask.

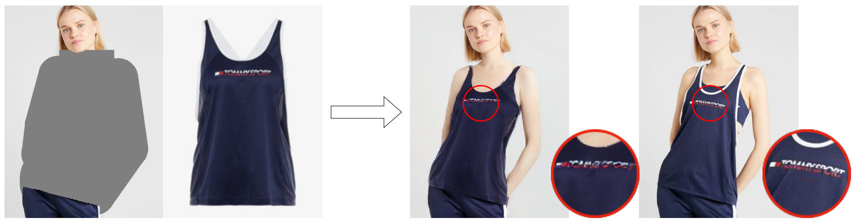


Figure 11: Failure case of CAT-DM about complex pattern.

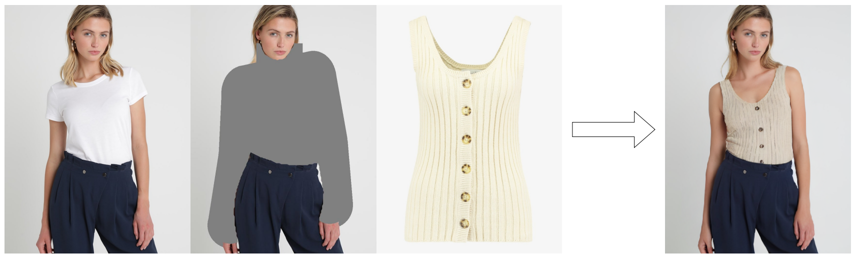


Figure 12: Color deviations of try-on image generated by GC-DM.

## G Inference time

Under our experimental conditions (i9-12900kf and RTX 3090), the inference time for each module of the model is as shown in Table 5. The table is divided into four columns, each representing a different module of CAT-DM, with the final column showing the total inference time. The modules are pre-trained GAN-based model (GP-VTON), diffusion model (GC-DM) and Poisson Blending which is performed on the CPU.
In GP-VTON, 33.4ms and 3.1ms respectively represent the inference time required for its two stages. In the diffusion model, the inference times are doubled (as indicated by "2×") because the number of sampling steps is set to 2. Furthermore, in AutoencoderKL, 2.2ms and 2.3ms represent the time required for the model to transform from pixel space to latent space and the time required to transform back from latent space to pixel space, respectively. The total inference time for all modules combined is 686.9ms.

| GP-VTON | Diffusion model (Sample steps is set to 2) | | | | | Poisson Blending (CPU) | Total |
|---|---|---|---|---|---|---|---|
| | CLIP | DINO-V2 | ControlNet | PBE | AutoencoderKL | | |
| 33.4ms+3.1ms | 2×5.9ms | 2×5.6ms | 2×25.3ms | 2×25.5ms | 2.2ms+2.3ms | 521.3ms | 686.9ms |

Table 5: Inference time of each module.

## References

[1] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7291–7299, 2017.

[2] R. A. Güler, N. Neverova, and I. Kokkinos, “Densepose: Dense human pose estimation in the wild,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7297–7306, 2018.

[3] P. Li, Y. Xu, Y. Wei, and Y. Yang, “Self-correction for human parsing,” 2020.

[4] B. Lefaudeux, F. Massa, D. Liskovich, W. Xiong, V. Caggiano, S. Naren, M. Xu, J. Hu, M. Tintore, S. Zhang, P. Labatut, and D. Haziza, “xformers: A modular and hackable transformer modelling library.” `https://github.com/facebookresearch/xformers`, 2022.

[5] G. Parmar, R. Zhang, and J.-Y. Zhu, “On aliased resizing and surprising subtleties in gan evaluation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11410–11420, 2022.

[6] M. Seitzer, “pytorch-fid: FID Score for PyTorch.” `https://github.com/mseitzer/pytorch-fid`, August 2020. Version 0.3.0.

[7] A. Obukhov, M. Seitzer, P.-W. Wu, S. Zhydenko, J. Kyl, and E. Y.-J. Lin, “High-fidelity performance metrics for generative models in pytorch,” 2020. Version: 0.3.0, DOI: 10.5281/zenodo.4957738.

[8] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

[9] Po-Hsun-Su, “Pytorch ssim.” `https://github.com/Po-Hsun-Su/pytorch-ssim`.