Supplementary Material for GP-VTON

Anonymous CVPR submission

Paper ID 3366

1. Architecture Details

1.1. Local-Flow Global-Parsing Warping Module

Our Local-Flow Global-Parsing Warping Module is composed of two Feature Pyramid Network (FPN) [10] (i.e., $\mathcal{E}_p$ and $\mathcal{E}_g$) and a cascade flow estimation module which consists of five LFGP blocks. We provide the detailed architecture of the FPN and the LFGP block in Tab. 1 and Tab. 2, respectively. Note, for $\mathcal{E}_p$, it takes as inputs a 25-channel densepose [5], a 25-channel 2D pose map [2], and a 1-channel preserved mask, resulting in a 51-channel input tensor. For $\mathcal{E}_g$, it takes as inputs a 3-channel garment image and a 1-channel garment parsing, resulting in a 4-channel input tensor. For the LFGP block, we take the block with lowest resolution as example, which receives the incoming feature with the resolution 16×12, and outputs the local flow with resolution 16×12 and garment parsing with the resolution 32×24.

1.2. Generator

Our try-on generator $\mathcal{G}$ inherits the Res-UNet [12] architecture. We provide the architecture details in Tab. 3. Note, the output of $\mathcal{G}$ is a 4-channel tensor, which is further split into a 3-channel coarse try-on result I'_c and a 1-channel alpha mask M_c. The final try-on result I' is obtained by using M_c to fuse I'_c and the warped garment G', which can be formulated as:

$$I' = G' \odot M_c + I'_c \odot (1 - M_c). \tag{1}$$

2. Experiments Details

2.1. Loss Functions

During training, we train the LFGP warping module and the generator separately. For the LFGP warping module, we calculate the l_1 loss $\mathcal{L}_1$ and perceptual loss [7] $\mathcal{L}_{per}$ between the local warped parts $\{G'^k\}_{k=1}^3$ and their corresponding ground truth $\{G_{gt}^k\}_{k=1}^3$, which can be formulated as:

$$\mathcal{L}_1 = \sum_{k=1}^{3} \|G'^k - G_{gt}^k\|_1, \tag{2}$$

$$\mathcal{L}_{perc} = \sum_{k=1}^{3}\sum_{j=1}^{5} \lambda_j \left\|\phi_j(G'^k) - \phi_j\left(G_{gt}^k\right)\right\|_1, \tag{3}$$

where $\phi_j(*)$ denotes the j-th feature map in a pre-trained VGG network [13]. We also utilize the l_1 loss $\mathcal{L}_m$ for the local warped masks, and the pixel-wise cross-entropy loss $\mathcal{L}_{ce}$ and the adversarial loss $\mathcal{L}_{adv}$ for the global garment parsing. Besides, we follow PFAFN [4] and employ the second-order smooth loss $\mathcal{L}_{sec}$ on the local flows $\{f^k\}_{k=1}^3$. The total loss for the LFGP module can be formulated as:

$$\begin{aligned}\mathcal{L}^w =& \mathcal{L}_1^w + \lambda_{per}^w \mathcal{L}_{per}^w + \lambda_m^w \mathcal{L}_m^w \\ &+ \lambda_{ce}\mathcal{L}_{ce} + \lambda_{adv}^w \mathcal{L}_{adv}^w + \lambda_{sec}\mathcal{L}_{sec},\end{aligned} \tag{4}$$

where λ_{per}^w, λ_m^w, λ_{ce}, λ_{adv}^w and λ_{sec} are the trade-off hyper-parameters, which are set to 0.2, 2.0, 0.5, 0.1, and 6.0, respectively.

For the generator, we utilize l_1 loss $\mathcal{L}_1$, the perceptual loss [7] $\mathcal{L}_{per}$, and the adversarial loss for the try-on result I', and also utilize the l_1 loss $\mathcal{L}_m$ for the alpha mask M_c. The total loss is defined as follows:

$$\mathcal{L}^g = \mathcal{L}_{per}^g + \lambda_{\mathcal{L}_1}^g \mathcal{L}_1^g + \lambda_{adv}^g \mathcal{L}_{adv}^g + \lambda_m^g \mathcal{L}_m^g, \tag{5}$$

where $\lambda_{\mathcal{L}_1}^g$, λ_{adv}^g and λ_m^g are the hyper-parameters, which are set to 5.0, 0.5, and 5.0, respectively.

2.2. Implementation Details

The training process for both dataset are the same, which include a two-stage training procedure and are trained on 8 Tesla V100 GPUs. During training LFGP warping module, the batch size is set to 2 for each GPU and the model is trained for 120 epochs with learning rate 5e-5, in which the DGT strategy is only employed for the last 50 epochs. During training the generator, the batch size is set to 16 for each GPU and the model is trained for 200 epochs with learning rate 5e-4. Both LFGP warping module and the generator employ the Adam optimizer [8] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

2.3. Human Evaluation Details

For human evaluation, we separately design two questionnaires for the VITON-HD dataset [3], and DressCode dataset [11]. Specifically, for the VITON-HD dataset, 40 volunteers are invited to complete the questionnaire which is composed of 25 assignments. For the DressCode dataset, 20 volunteers are invited to complete the questionnaire which contains 15 assignments for each garment category (i.e., upper-, lower-garment, dresses), namely, 45 assignments in total. For each assignment in the questionnaire, given a person image and a garment image, the volunteers are asked to select the most realistic and accurate try-on result out of five options, which are generated by out GP-VTON and the baseline methods (i.e., PF-AFN [4], FS-VTON [6], HRVITON [9], SDAFN [1]). Besides, the order of the generated results in each assignment are randomly shuffled. Fig. 2 shows the interface of the questionnaire for the VITON-HD dataset. The interface for the DressCode dataset is identical. Please refer to Sec.4.2 in the main text for the detailed quantitative results of the human evaluation.

3. Additional Results

Visual Comparisons with SOTAs on VITON-HD dataset [3]. Fig. 3 displays additional visual comparisons among GP-VTON and the baseline methods on the VITON-HD dataset.

Visual Comparisons with SOTAs on DressCode dataset [11]. Fig. 4, Fig. 5, and Fig. 6 display additional visual comparisons among GP-VTON and the baseline methods on the upper, lower and dresses subset of the DressCode dataset, respectively.

Virtual Try-on for FIFA World Cup Qatar 2022. We also test our GP-VTON in the jersey try-on scenario for FIFA World Cup Qatar 2022, where the garment images are jerseys from different countries that we can acquire in the Internet, and the person image are from the VITON-HD dataset [3]. Please refer to Fig. 7 for the visual results.

4. Potential Social Impacts and Limitations

Potential social impacts. As with most generative models, our GP-VTON might be applied to malicious image manipulations, such as transferring weird garment onto specific person without permission. Nevertheless, such negative impact could be alleviated via forensics analysis and other manipulation detection methods.

Limitations. Since our GP-VTON conducts local warping for different garment parts individually, it would fail to obtain accurate warped result when the input in-shop garment is incomplete. As shown in Fig. 1 (A), the right sleeve of the input garment is invisible, GP-VTON fails to generate compelling result in the arm region. Besides, GP-VTON is unable to address the parsing error. As show in Fig. 1 (B), the wrong human parsing result of the lower body leads to the incorrect preserved region, which further influences the shape of the warped garment and the visual quality of the try-on results. To alleviate the influence of the parsing error, we could resort to the knowledge distillation mechanism, which is commonly used in [4, 6], to obtain a parsing-free model.

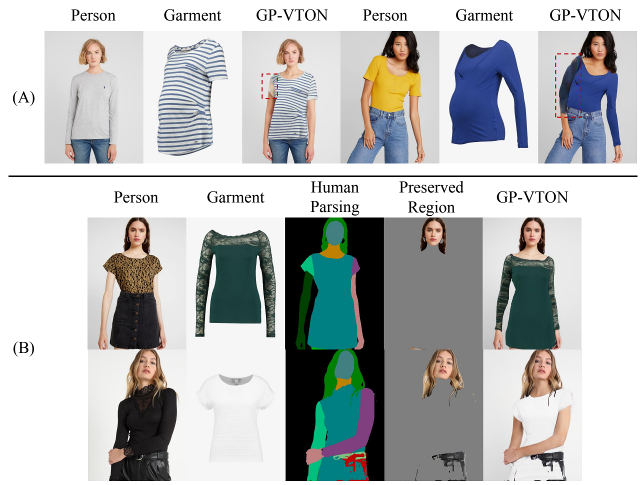

Figure 1. Failure cases of our GP-VTON. Please zoom in for more details.

References

[1] Shuai Bai, Huiling Zhou, Zhikang Li, Chang Zhou, and Hongxia Yang. Single stage virtual try-on via deformable attention flows. In *ECCV*, 2022. 2

[2] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *TPAMI*, 43(01):172–186, 2021. 1

[3] Seunghwan Choi, Sunghyun Park, Minsoo Lee, and Jaegul Choo. Viton-hd: High-resolution virtual try-on via misalignment-aware normalization. In *CVPR*, pages 14131–14140, 2021. 1, 2, 3, 4

[4] Yuying Ge, Yibing Song, Ruimao Zhang, Chongjian Ge, Wei Liu, and Ping Luo. Parser-free virtual try-on via distilling appearance flows. In *CVPR*, pages 8485–8493, 2021. 1, 2

[5] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *CVPR*, pages 7297–7306, 2018. 1

[6] Sen He, Yi-Zhe Song, and Tao Xiang. Style-based global appearance flow for virtual try-on. In *CVPR*, pages 3470–3479, 2022. 2

[7] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016. 1

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[9] Sangyun Lee, Gyojung Gu, Sunghyun Park, Seunghwan Choi, and Jaegul Choo. High-resolution virtual try-on with misalignment and occlusion-handled conditions. In *ECCV*, 2022. 2

[10] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid

Figure 2. Interface of the questionnaire used to evaluate the final try-on results on the VITON-HD dataset [3].

Figure 3. Qualitative comparison on the VITON-HD dataset [3].Please zoom in for more details.

Figure 4. Qualitative comparison on the upper subset of the DressCode dataset [11].Please zoom in for more details.

Figure 5. Qualitative comparison on the lower subset of the DressCode dataset [11].Please zoom in for more details.

Figure 6. Qualitative comparison on the dresses subset of the DressCode dataset [11].Please zoom in for more details.

Figure 7. Virtual try-on results for FIFA World Cup Qatar 2022. Please zoom in for more details.

networks for object detection. In *CVPR*, pages 2117–2125, 2017. 1

[11] Davide Morelli, Matteo Fincato, Marcella Cornia, Federico Landi, Fabio Cesari, and Rita Cucchiara. Dress Code: High-Resolution Multi-Category Virtual Try-On. In *ECCV*, 2022. 2, 5, 6, 7

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 1

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015. 1

$\mathcal{E}_p$ ($\mathcal{E}_g$)			
Layer		**Operation**	**Output Size**
Feature Encoder	Input	-	(512,384,51(4))
	Conv 1-1	InstanceNorm, ReLU, Conv2d 3×3	(256,192,64)
	Res 1-1	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(256,192,64)
	Res 1-2	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(256,192,64)
	Conv 2-1	InstanceNorm, ReLU, Conv2d 3×3	(128,96,128)
	Res 2-1	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(128,96,128)
	Res 2-2	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(128,96,128)
	Conv 3-1	InstanceNorm, ReLU, Conv2d 3×3	(64,48,256)
	Res 3-1	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(64,48,256)
	Res 3-2	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(64,48,256)
	Conv 4-1	InstanceNorm, ReLU, Conv2d 3×3	(32,24,256)
	Res 4-1	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(32,24,256)
	Res 4-2	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(32,24,256)
	Conv 5-1	InstanceNorm, ReLU, Conv2d 3×3	(16,12,256)
	Res 5-1	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(16,12,256)
	Res 5-2	Residual Block (InstanceNorm, ReLU, Conv2d 3×3)	(16,12,256)
Pyramid Encoder	Conv 6-1	Conv2d 1×1	(16,12,256)
	Conv 6-2	Conv2d 3×3	(16,12,256)
	Upsample 7-1	Interpolation(scale-factor=2)	(32,24,256)
	Skip Connection 7-1	Skip Connection from Res 4-2 (Conv2d 1×1, Addition)	(32,24,256)
	Conv 7-1	Conv2d 3×3	(32,24,256)
	Upsample 8-1	Interpolation(scale-factor=2)	(64,48,256)
	Skip Connection 8-1	Skip Connection from Res 3-2 (Conv2d 1×1, Addition)	(64,48,256)
	Conv 8-1	Conv2d 3×3	(64,48,256)
	Upsample 9-1	Interpolation(scale-factor=2)	(128,96,256)
	Skip Connection 9-1	Skip Connection from Res 2-2 (Conv2d 1×1, Addition)	(128,96,256)
	Conv 9-1	Conv2d 3×3	(128,96,256)
	Upsample 10-1	Interpolation(scale-factor=2)	(256,128,256)
	Skip Connection 10-1	Skip Connection from Res 1-2 (Conv2d 1×1, Addition)	(256,128,256)
	Conv 10-1	Conv2d 3×3	(256,128,256)

Table 1. The architecture details of the FPN ($\mathcal{E}_p$ ($\mathcal{E}_g$)).

LFGP Block				
Layer			**Operation**	**Output Size**
Coarse Flow Block	Left Flow Block-1	Conv 1-1	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,128)
		Conv 1-2	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,64)
		Conv 1-3	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,32)
		Conv 1-4	Conv2d 3×3	(16,12,2)
	Middle Flow Block-1	Conv 2-1	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,128)
		Conv 2-2	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,64)
		Conv 2-3	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,32)
		Conv 2-4	Conv2d 3×3	(16,12,2)
	Right Flow Block-1	Conv 3-1	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,128)
		Conv 3-2	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,64)
		Conv 3-3	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,32)
		Conv 3-4	Conv2d 3×3	(16,12,2)
Fine Flow Block	Left Flow Block-2	Conv 4-1	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,128)
		Conv 4-2	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,64)
		Conv 4-3	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,32)
		Conv 4-4	Conv2d 3×3	(16,12,2)
	Middle Flow Block-2	Conv 5-1	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,128)
		Conv 5-2	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,64)
		Conv 5-3	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,32)
		Conv 5-4	Conv2d 3×3	(16,12,2)
	Right Flow Block-2	Conv 6-1	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,128)
		Conv 6-2	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,64)
		Conv 6-3	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(16,12,32)
		Conv 6-4	Conv2d 3×3	(16,12,2)
Global Parsing Block	Fusion Block	Conv 7-1	Conv2d 1×1	(32,24,256)
		Res 7-1	Residual Block	(32,24,256)
	Parsing Block	Conv 8-1	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(32,24,128)
		Conv 8-2	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(32,24,64)
		Conv 8-3	Conv2d 3×3, LeakyReLU(negative-slope=0.1)	(32,24,32)
		Conv 8-4	Conv2d 3×3, Tanh	(32,24,7)

Table 2. The architecture details of the LFGP block.

$\mathcal{G}$			
Layer		**Operation**	**Output Size**
Encoder	Input	-	(512,384,37)
	Conv 1-1	Conv2d 3×3, ReLU	(256,192,64)
	Res 1-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(256,192,64)
	Res 1-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(256,192,64)
	Conv 2-1	Conv2d 3×3, BN, ReLU	(128,96,128)
	Res 2-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(128,96,128)
	Res 2-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(128,96,128)
	Conv 3-1	Conv2d 3×3, BN, ReLU	(64,48,256)
	Res 3-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(64,48,256)
	Res 3-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(64,48,256)
	Conv 4-1	Conv2d 3×3, BN, ReLU	(32,24,512)
	Res 4-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(32,24,512)
	Res 4-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(32,24,512)
	Conv 5-1	Conv2d 3×3, ReLU	(16,12,512)
	Res 5-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(16,12,512)
	Res 5-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(16,12,512)
Decoder	Conv 6-1	Upsample, Conv2d 3×3, BN, ReLU	(32,24,512)
	Res 6-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(32,24,512)
	Res 6-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(32,24,512)
	Skip Connection 7-1	Skip Connection from Res 4-2 (Concatenation)	(32,24,1024)
	Conv 7-1	Upsample, Conv2d 3×3, BN, ReLU	(64,48,256)
	Res 7-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(64,48,256)
	Res 7-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(64,48,256)
	Skip Connection 8-1	Skip Connection from Res 3-2 (Concatenation)	(64,48,512)
	Conv 8-1	Upsample, Conv2d 3×3, BN, ReLU	(128,96,128)
	Res 8-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(128,96,128)
	Res 8-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(128,96,128)
	Skip Connection 9-1	Skip Connection from Res 2-2 (Concatenation)	(128,96,256)
	Conv 9-1	Upsample, Conv2d 3×3, BN, ReLU	(256,192,64)
	Res 9-1	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(256,192,64)
	Res 9-2	Residual Block (Conv2d 3×3, BN, ReLU, Conv2d 3×3, BN)	(256,192,64)
	Skip Connection 10-1	Skip Connection from Res 1-2 (Concatenation)	(256,192,128)
	Conv 10-1	Upsample, Conv2d 3×3	(512,384,4)

Table 3. The architecture details of the generator $\mathcal{G}$.