PROMO: Promptable Outfitting for Efficient High-Fidelity Virtual Try-On

Supplementary Material

A. Supplemantary Quantitative Results

The full results of ablating **Parsing Area Loss**, **Style Prompt** and **3D-RoPE**

	DressCode						VITON-HD					
	Paired				Unpaired		Paired				Unpaired	
	SSIM ↑	LPIPS ↓	FID ↓	KID ↓	FID ↓	KID ↓	SSIM ↑	LPIPS ↓	FID ↓	KID ↓	FID ↓	KID ↓
- w/o parsing area loss	<u>0.8896</u>	**0.0872**	**3.2847**	<u>0.5076</u>	**4.6390**	0.9535	0.8401	<u>0.1342</u>	<u>7.0419</u>	**1.2711**	**9.4283**	**1.2711**
- w/o style prompt	<u>0.8896</u>	0.0926	3.7178	0.8926	<u>5.3516</u>	<u>0.6180</u>	<u>0.8432</u>	0.1376	7.8145	1.5427	10.1242	<u>1.5575</u>
- w/o 3D-RoPE	0.8702	0.1304	6.7293	1.7160	7.8175	2.2801	0.8335	0.1609	10.1979	1.5219	11.8961	2.2801
PROMO (Ours)	**0.8913**	<u>0.0887</u>	<u>3.3103</u>	**0.4902**	<u>5.3516</u>	**0.4992**	**0.8619**	**0.1111**	**6.8903**	<u>1.4895</u>	<u>9.9034</u>	1.9174

Table 1. Ablation study on DressCode and VITON-HD datasets. The best and second-best results are highlighted in **bold** and <u>underlined</u>, respectively.

B. Implementation Details of TryOff Model

We trained our TryOff model based on FLUX.1 Kontext [dev] [3], using LoRA [2] with rank of 16 and alpha of 4. We employed the AdamW optimizer [4] with a learning rate of 1e-4. The training dataset consists of person-garment pairs from DressCode [5] and Viton-HD [1] datasets.

For garment extraction, we defined category-specific prompts as follows:

```
PROMPT_DICT = {
    "lower_body": "extract only the lower body garment over a white
                   background, product photography style",
    "upper_body": "extract only the upper body garment over a white
                   background, product photography style",
    "dresses": "extract only the dresses garment over a white
                background, product photography style",
}
```

C. Prompt Extraction Details

The following code is the complete pydantic schema, and we use the OpenAPI json formatted schema description generated by pydantic using `Output.model_json_schema()` to guide the LLM to output the expected result.

```python
from pydantic import BaseModel

class Person(BaseModel):
    body_shape: str = Field(description="Body type in 1 word", examples=["slim",
    ↪  "athletic", "curvy", "plus-size", "average"])
    gender: Literal["man", "woman"]
    hair_length: str = Field(description="Hair length in 2-3 words", examples=["short",
    ↪  "shoulder-length", "long", "very-long"])
    pose: str = Field(description="Overall body pose in 5-10 words")
    hand_pose: str = Field(description="Hand/arm position in 5-10 words")

class GarmentBase(BaseModel):
```

```python
    garment_type: str = Field(description="Describe specific garment type in 1 words.", examples=["t-shirt", "sweater", "jeans", "skirt", "shorts", "dress", "romper"])
    color: str = Field(description="Describe cloth color in 1 words.")
    material: str = Field(description="Describe material in 1 words.")
    fit: str = Field(description="Describe fit of the garment in 1 words.", examples=["tight", "loose", "relaxed", "fitted", "oversized"])

class GarmentUpperBase(GarmentBase):
    neckline: str = Field(description="Describe neckline in 1 word.")
    sleeve_length: Literal["sleeveless", "short sleeve (above elbow)", "elbow length", "3/4 sleeve (mid forearm)", "long sleeve (at wrist)", "extra long (below wrist, covering hand)"]
    sleeve_rolling_style: Literal["sleeveless", "rolled down", "rolled up", "rolled in the middle"]

class GarmentUpper(GarmentUpperBase):
    type: Literal["upper_body"]
    upper_garment_length: Literal["cropped length", "waist level", "below waist, covering lower garment"]

class GarmentLowerBase(GarmentBase):
    lower_garment_length: Literal["out of frame", "short (above knee)", "knee-length (precisely reaches knee)", "midi (mid-calf)", "ankle-length", "floor-length"]

class GarmentLower(GarmentLowerBase):
    type: Literal["lower_body"]

class GarmentFull(GarmentUpperBase, GarmentLowerBase):
    type: Literal["full_body"]

class GarmentShoes(GarmentBase):
    type: Literal["shoes"]
    shoes_length: Literal["above knee", "knee height", "mid-calf", "ankle height", "low-top", "no shoes"]

class GarmentSocks(GarmentBase):
    type: Literal["socks"]
    socks_length: Literal["above knee", "knee height", "mid-calf", "ankle height", "low-top", "no socks"]

class Output(BaseModel):
    person: Person
    garments: list[Annotated[GarmentUpper | GarmentLower | GarmentFull | GarmentShoes | GarmentSocks, Field(discriminator="type")]]
```

References

[1] Seunghwan Choi, Sunghyun Park, Minsoo Lee, and Jaegul Choo. Viton-hd: High-resolution virtual try-on via misalignment-aware normalization, 2021. 1

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 1

[3] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. 1

[4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 1

[5] Davide Morelli, Matteo Fincato, Marcella Cornia, Federico Landi, Fabio Cesari, and Rita Cucchiara. Dress code: High-resolution multi-category virtual try-on, 2022. 1