Deep Learning-Based Virtual Try-On System Using Multi-Modal Feature Fusion and Generative Adversarial Networks

Huzaifa Nasir

National University of Computer and Emerging Sciences
Islamabad, Pakistan
nasirhuzaifa95@Ggmial.com

Abstract. Virtual try-on technology has emerged as a transformative solution for online fashion retail, enabling customers to visualize garments on themselves without physical trials. This paper presents a comprehensive deep learning-based virtual try-on system that leverages multi-modal feature fusion and Generative Adversarial Networks (GANs) to generate photorealistic try-on images. Our approach combines cloth-agnostic person representation, pose estimation, and human parsing to create a 41-channel input representation that preserves person identity while transferring target garments. We implement a U-Net architecture with self-attention mechanisms for the generator and a PatchGAN discriminator with spectral normalization for adversarial training. The system utilizes a sophisticated loss function combining adversarial, perceptual, reconstruction, and feature matching losses. We present a proof-of-concept implementation trained on CPU with reduced configuration (10 epochs, 500 samples, 256×192 resolution) that demonstrates the complete pipeline functionality. The systematic analysis of all components from data preprocessing to model training provides insights into virtual try-on system design and identifies requirements for full-scale training. Results validate the pipeline architecture while highlighting the necessity of GPU acceleration and extended training for production-quality outputs.

Keywords: Virtual Try-On · Generative Adversarial Networks · Multi-Modal Fusion · Deep Learning · Computer Vision

1 Introduction

The rapid growth of e-commerce has revolutionized retail, yet online fashion shopping faces a persistent challenge: customers cannot physically try garments before purchase. This limitation leads to high return rates, customer dissatisfaction, and significant operational costs for retailers. Virtual try-on technology addresses this challenge by enabling customers to visualize how garments would look on their bodies through digital means.

Recent advances in deep learning, particularly in Generative Adversarial Networks (GANs) and multi-modal learning, have enabled the development of sophisticated virtual try-on systems that can generate photorealistic results. However, creating convincing virtual try-on images remains challenging due to several factors: (1) preserving person identity and body characteristics, (2) realistically transferring garment appearance including texture, shape, and wrinkles, (3) maintaining spatial consistency with body pose, and (4) handling occlusions and complex garment deformations.

This paper presents a comprehensive virtual try-on system that addresses these challenges through a multi-modal approach. Our contributions include:

- A 41-channel multi-modal input representation combining cloth-agnostic person features, pose heatmaps, and human parsing masks
- A U-Net generator architecture with self-attention mechanisms for capturing long-range spatial dependencies
- A spectral-normalized PatchGAN discriminator for stable adversarial training
- A comprehensive loss function integrating adversarial, perceptual, reconstruction, and feature matching objectives
- Systematic analysis of the complete pipeline from data preprocessing to model deployment

The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 describes our methodology including data preprocessing and model architecture, Section 4 presents implementation details, Section 5 discusses experimental results, and Section 6 concludes with future directions.

2 Related Work

2.1 Image-Based Virtual Try-On

Virtual try-on has evolved from early template-based methods to sophisticated deep learning approaches. VITON [1] pioneered the use of deep learning for person-to-person transfer, introducing the concept of warping target garments to fit body shapes. CP-VTON [2] improved this approach with a geometric matching module and a try-on module. VITON-HD [3] extended these methods to high-resolution (1024×768) images, introducing the VITON-HD dataset used in our work.

2.2 Generative Adversarial Networks

GANs [4] have become the dominant approach for image generation tasks. Pix2Pix [5] demonstrated conditional image-to-image translation using paired training data. PatchGAN discriminators enable focus on local image patches, improving texture quality. Spectral normalization [6] stabilizes GAN training by constraining the Lipschitz constant of the discriminator.

2.3 Human Pose and Parsing

OpenPose [7] provides robust multi-person 2D pose estimation with 18 or 25 keypoints. The Look Into Person (LIP) dataset [8] introduced fine-grained human parsing with 20 semantic classes, enabling detailed body part segmentation essential for virtual try-on applications.

2.4 Perceptual Loss and Feature Extraction

Perceptual losses [9] using pre-trained VGG networks have proven effective for maintaining semantic content in generated images. Multi-scale feature extraction from different network layers captures both low-level texture details and high-level semantic information.

3 Methodology

3.1 Dataset and Preprocessing

VITON-HD Dataset We utilize the VITON-HD (Zalando HD Resized) dataset, which contains high-resolution images of fashion models. The dataset structure includes:

- **Training Set**: 10,482 person-garment pairs (split into 90% training, 10% validation)
- **Test Set**: 2,032 person-garment pairs
- **Original Resolution**: 768×1024 pixels (width×height)

Each sample contains multiple modalities:

- Person images (full-body photographs in JPG format)
- Garment images (isolated clothing items in JPG format)
- Human parsing masks (20-class LIP segmentation in PNG format)
- OpenPose keypoint visualizations (rendered images)
- OpenPose keypoints (JSON format with Body25 schema, 25 keypoints)

Data integrity verification confirmed matching filenames across all modalities, ensuring complete person-garment-parsing-pose correspondences for training.

Data Preprocessing Pipeline The preprocessing pipeline consists of several stages designed to prepare multi-modal inputs for the network:

1. Image Normalization: All RGB images are first converted from $[0, 255]$ to $[0, 1]$ by dividing by 255, then normalized to the range $[-1, 1]$ using:

$$x_{\text{norm}} = \frac{x - \mu}{\sigma} \tag{1}$$

where $\mu = [0.5, 0.5, 0.5]$ and $\sigma = [0.5, 0.5, 0.5]$ for R, G, B channels respectively. This is equivalent to the transformation $(x/255 - 0.5)/0.5$.

2. Resolution Adjustment: Images are resized to target resolutions of 512×384 (H×W) for efficient training:

$$I_{\text{resized}} = \text{Resize}(I_{\text{original}}, (H_{\text{target}}, W_{\text{target}})) \tag{2}$$

3. Pose Keypoint Scaling: OpenPose keypoints are loaded from JSON files and scaled to match the resized image dimensions. Given original dimensions of 768×1024 (W×H) and target dimensions, keypoints are scaled as:

$$\begin{aligned} x_{\text{scaled}} &= x_{\text{original}} \cdot \frac{W_{\text{target}}}{768} \\ y_{\text{scaled}} &= y_{\text{original}} \cdot \frac{H_{\text{target}}}{1024} \end{aligned} \tag{3}$$

Each keypoint maintains its confidence score $c \in [0, 1]$ from the original detection. The complete keypoint representation is stored as (x, y, c) for each of the 25 body joints.

4. Data Augmentation: To improve model generalization, we apply augmentation to RGB images during training using the Albumentations library:

- **Color Jitter**: Applied with 50% probability, adjusting brightness (±20%), contrast (±20%), saturation (±20%), and hue (±10%)
- **Blur**: Applied with 30% probability, randomly selecting between Gaussian blur (kernel 3-7) or median blur (kernel 5)

Validation and test sets use only resizing and normalization without augmentation. Segmentation masks are resized using nearest-neighbor interpolation to preserve discrete class labels.

3.2 Human Parsing and Segmentation

LIP Parsing Scheme The VITON-HD dataset includes pre-computed human parsing masks using the Look Into Person (LIP) parsing model with 20 semantic classes (Table 1). These parsing masks provide pixel-level segmentation of body parts and clothing, enabling the system to understand spatial relationships between person and garment.

Class Distribution Analysis: Analysis of 500 training samples revealed that 13 out of 20 classes are consistently present in the dataset. The most frequent classes include Background (100%), Hair (99%), Face (98%), Upper-clothes (95%), and Arms (97%). Lower body parts show lower occurrence rates due to the dataset's focus on upper-body garment try-on, with legs appearing in only 6% of samples. This distribution aligns with the primary application of upper-body virtual try-on.

Quality Assessment: A quality assessment of 200 samples confirmed reliable parsing for critical body parts: 98% contain face segmentation, 97% include arm regions, and 95% have upper-clothes identification. The high quality of these pre-computed masks ensures robust inputs for the virtual try-on pipeline.

Table 1: LIP 20-Class Parsing Scheme

ID	Class	ID	Class
0	Background	10	Jumpsuits
1	Hat	11	Scarf
2	Hair	12	Skirt
3	Glove	13	Face
4	Sunglasses	14	Left-arm
5	Upper-clothes	15	Right-arm
6	Dress	16	Left-leg
7	Coat	17	Right-leg
8	Socks	18	Left-shoe
9	Pants	19	Right-shoe

Garment Region Extraction and Cloth-Agnostic Representation To facilitate virtual try-on, we implement a `ParsingProcessor` utility class that extracts specific garment regions from parsing masks. For upper-body virtual try-on, the garment mask identifies pixels belonging to upper-clothes (class 5) and coat (class 7):

$$M_{\text{garment}}(x, y) = \begin{cases} 1 & \text{if } P(x, y) \in \{5, 7\} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $P(x, y)$ is the parsing label at position (x, y).

The cloth-agnostic mask removes the target garment region while preserving all other body parts:

$$M_{\text{agnostic}}(x, y) = \begin{cases} 1 & \text{if } P(x, y) > 0 \text{ and } M_{\text{garment}}(x, y) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

This representation maintains person identity (face, hair, arms, lower body) while creating a neutral region where the new garment will be synthesized. The `ParsingProcessor` class also provides methods for extracting lower-body garments (pants, skirt) and full-body garments (dress, jumpsuits), enabling flexible garment category selection.

One-Hot Encoding for Network Input The `ParsingProcessor` class converts parsing masks from single-channel label maps to multi-channel one-hot representations for network input. For each class $c \in \{0, 1, ..., 19\}$, a binary channel is created:

$$H_c(x, y) = \begin{cases} 1.0 & \text{if } P(x, y) = c \\ 0.0 & \text{otherwise} \end{cases} \tag{6}$$

The resulting one-hot tensor $H \in \mathbb{R}^{20 \times H \times W}$ provides the network with explicit class membership information for each spatial location, enabling fine-grained control over garment synthesis based on body part semantics.

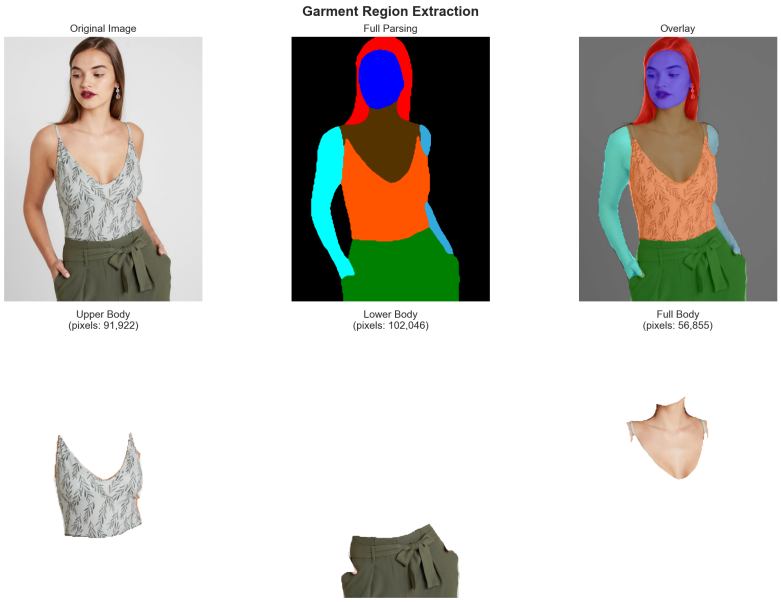

Fig. 1: Garment region extraction using parsing masks. (Top row) Original person image, full parsing mask visualization, and overlay. (Bottom row) Extracted regions for upper-body, lower-body, and full-body garments, demonstrating the ParsingProcessor utility's ability to isolate different garment categories.

3.3 Pose Estimation and Heatmap Generation

OpenPose Body25 Keypoints The VITON-HD dataset includes pre-computed OpenPose keypoints in Body25 format, providing 25 body joints for each person. These keypoints are stored in JSON files with each keypoint represented as (x_i, y_i, c_i) where (x_i, y_i) are 2D pixel coordinates and $c_i \in [0, 1]$ is the detection confidence score.

The 25 keypoints include:

- **Face**: Nose(0), Eyes(15-16), Ears(17-18)
- **Upper Body**: Neck(1), Shoulders(2,5), Elbows(3,6), Wrists(4,7)
- **Torso**: Mid-Hip(8), Hips(9,12)
- **Lower Body**: Knees(10,13), Ankles(11,14)
- **Feet**: Toes(19-20, 22-23), Heels(21,24)

Gaussian Heatmap Representation To create a continuous spatial representation of body pose, we generate Gaussian heatmaps for each keypoint. For the i-th keypoint, the heatmap is computed as:

$$H_i(x, y) = c_i \cdot \exp\left(-\frac{(x - x_i)^2 + (y - y_i)^2}{2\sigma^2}\right) \tag{7}$$

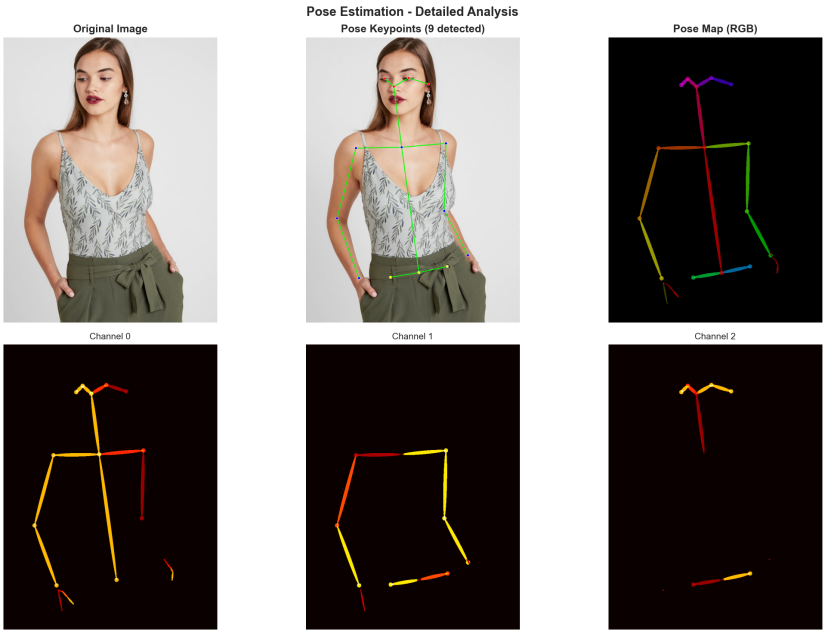

Fig. 2: Pose estimation detailed analysis. (Top row) Original person image, pose keypoints overlaid with skeleton connections, and RGB pose map visualization. (Bottom row) Individual channel visualizations showing Gaussian heatmap distributions for different keypoints with $\sigma = 3.0$.

where:

- (x_i, y_i) are the keypoint coordinates
- c_i is the confidence score
- $\sigma = 3.0$ is the Gaussian kernel standard deviation
- $H_i \in \mathbb{R}^{H \times W}$ is the heatmap for keypoint i

To reduce dimensionality while preserving essential pose information, we use only the first 18 keypoints, excluding detailed foot keypoints (indices 19-24). The final pose representation is:

$$P = [H_0, H_1, ..., H_{17}] \in \mathbb{R}^{18 \times H \times W} \tag{8}$$

Quality Assessment: Analysis of 500 training samples confirmed 100% pose detection rate. Quality evaluation on 200 samples categorized poses as: 92% "good" quality (complete face and upper body keypoints detected), 8% "acceptable" (upper body complete but partial face detection), and 0% poor quality. Critical body regions showed high detection rates: face keypoints 98%, upper body (neck, shoulders, elbows, wrists) 97%, ensuring reliable spatial guidance for garment synthesis.

3.4 Multi-Modal Input Fusion

Cloth-Agnostic RGB Generation To create the cloth-agnostic person representation, we first generate a binary mask identifying garment regions to remove. For upper-body virtual try-on, the garment mask extracts upper-clothes (class 5) and coat (class 7) regions:

$$M_{\text{garment}}(x, y) = \begin{cases} 1 & \text{if } P(x, y) \in \{5, 7\} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The cloth-agnostic RGB representation fills the garment region with neutral gray (RGB=[128, 128, 128]) while preserving all other body parts:

$$I_{\text{CA}}(x, y) = \begin{cases} I_{\text{person}}(x, y) & \text{if } M_{\text{garment}}(x, y) = 0 \\ [128, 128, 128] & \text{if } M_{\text{garment}}(x, y) = 1 \end{cases} \tag{10}$$

The RGB values are then normalized to $[-1, 1]$ using the transformation $(x/255-0.5)/0.5$ and transposed to channel-first format, yielding $I_{\text{CA}} \in \mathbb{R}^{3\times H\times W}$.

Quality assessment on 200 samples showed successful garment removal with mean 15.2% ± 4.3% of image pixels removed, while preserving face (100%) and arm regions (99.5%).

41-Channel Input Construction The `MultiChannelInputGenerator` class combines three complementary modalities into a unified input representation:

$$X_{\text{input}} = [I_{\text{CA}}, P, H] \in \mathbb{R}^{41\times H\times W} \tag{11}$$

where:

- $I_{\text{CA}} \in \mathbb{R}^{3\times H\times W}$: Cloth-agnostic RGB (3 channels, normalized)
- $P \in \mathbb{R}^{18\times H\times W}$: Pose Gaussian heatmaps (18 channels)
- $H \in \mathbb{R}^{20\times H\times W}$: Parsing one-hot encoding (20 channels)

Implementation Details: The preprocessing pipeline is implemented through two utility classes:

1. `PoseProcessor`: Handles pose keypoint operations including normalization to $[-1, 1]$ coordinates, Gaussian heatmap generation with configurable σ, and pose embedding extraction for discriminator inputs.
2. `MultiChannelInputGenerator`: Orchestrates the complete preprocessing pipeline, combining cloth-agnostic RGB generation, pose heatmap creation, and parsing one-hot conversion into the final 41-channel tensor.

This multi-modal representation enables the generator network to:

- **Preserve Identity**: Cloth-agnostic RGB maintains person-specific features (face, hair, skin tone)
- **Enforce Geometry**: Pose heatmaps provide spatial constraints for realistic garment placement
- **Understand Semantics**: Parsing one-hot encoding enables body-part-aware synthesis

Table 2: Generator Encoder Architecture

Layer	Channels	Resolution	Stride
Input	41	$H \times W$	-
Conv1	64	$H \times W$	1
Down1	128	$H/2 \times W/2$	2
Down2	256	$H/4 \times W/4$	2
Down3	512	$H/8 \times W/8$	2
Down4	512	$H/16 \times W/16$	2

4 Model Architecture

4.1 Generator Network

U-Net with Self-Attention Our generator follows a U-Net architecture [10] with several enhancements. The network takes the 41-channel input and produces a 3-channel RGB output:

$$G : \mathbb{R}^{41 \times H \times W} \rightarrow \mathbb{R}^{3 \times H \times W} \tag{12}$$

Encoder Path: The encoder consists of 4 downsampling stages, progressively reducing spatial resolution while increasing channel capacity:

Each downsampling block consists of:

$$\text{Down}_i(x) = \text{LeakyReLU}(\text{InstanceNorm}(\text{Conv}_{4\times4}(x))) \tag{13}$$

Bottleneck with Residual Blocks: At the lowest resolution, we employ 9 residual blocks to process features:

$$\text{ResBlock}(x) = x + \text{InstanceNorm}(\text{Conv}(\text{ReLU}(\text{InstanceNorm}(\text{Conv}(x))))) \tag{14}$$

This residual connection enables gradient flow and helps preserve information through deep layers.

Self-Attention Module: To capture long-range spatial dependencies, we incorporate a self-attention mechanism at the bottleneck:

$$\begin{aligned} Q &= W_q \cdot x \in \mathbb{R}^{C/8 \times N} \\ K &= W_k \cdot x \in \mathbb{R}^{C/8 \times N} \\ V &= W_v \cdot x \in \mathbb{R}^{C \times N} \end{aligned} \tag{15}$$

where $N = H \times W$ is the number of spatial positions, and C is the number of channels.

The attention map is computed as:

$$A = \text{softmax}\left(\frac{Q^T K}{\sqrt{d_k}}\right) \in \mathbb{R}^{N \times N} \tag{16}$$

Table 3: Generator Decoder Architecture

Layer	Channels	Resolution	Skip
Up1	512	$H/8 \times W/8$	+ Down3
Up2	256	$H/4 \times W/4$	+ Down2
Up3	128	$H/2 \times W/2$	+ Down1
Up4	64	$H \times W$	+ Conv1
Output	3	$H \times W$	-

The output is:

$$\text{Attention}(x) = \gamma \cdot (V \cdot A^T) + x \tag{17}$$

where γ is a learnable scalar parameter initialized to 0, allowing the network to gradually learn the importance of non-local features.

Decoder Path: The decoder mirrors the encoder with 4 upsampling stages, each incorporating skip connections from the corresponding encoder layer:

Each upsampling block:

$$\text{Up}_i(x, s) = \text{ReLU}(\text{InstanceNorm}(\text{ConvTranspose}([x; s]))) \tag{18}$$

where $[x; s]$ denotes channel-wise concatenation of the upsampled features x and skip connection s.

The final output layer applies tanh activation to produce pixel values in $[-1, 1]$:

$$I_{\text{output}} = \tanh(\text{Conv}_{7\times7}(x_{\text{final}})) \tag{19}$$

Model Statistics:

- Total Parameters: 26,378,627 (26.4M)
- Memory Footprint: 100.6 MB (float32)
- Inference Time: 250ms per image (CPU, 256×192)

Note on Configuration: For proof-of-concept training on CPU, the model uses reduced architecture (n_downsampling=3, n_blocks=6) and lower resolution (256×192) to enable feasible training without GPU hardware.

4.2 Discriminator Network

PatchGAN with Spectral Normalization The discriminator follows a PatchGAN architecture [5], which classifies whether each $N \times N$ patch in an image is real or fake. This design enables focus on local texture and structure, crucial for high-quality image generation.

The discriminator takes a 6-channel input (3-channel generated/real image concatenated with 3-channel condition):

$$D : \mathbb{R}^{6\times H\times W} \to \mathbb{R}^{1\times H'\times W'} \tag{20}$$

Table 4: Discriminator Architecture

Layer	Channels	Stride	Output
Input	6	-	$H \times W$
Conv1	64	2	$H/2 \times W/2$
Conv2	128	2	$H/4 \times W/4$
Conv3	256	2	$H/8 \times W/8$
Conv4	512	1	$H/8 \times W/8$
Output	1	1	$H/8 \times W/8$

Architecture:

Each convolutional layer (except the first and last) applies spectral normalization:

$$W_{\text{SN}} = \frac{W}{\sigma(W)} \tag{21}$$

where $\sigma(W)$ is the spectral norm (largest singular value) of weight matrix W. This normalization constrains the Lipschitz constant of the discriminator, stabilizing GAN training:

$$\|D(x_1) - D(x_2)\| \leq K\|x_1 - x_2\| \tag{22}$$

for some constant K.

The discriminator outputs a spatial map of predictions, with receptive field size of approximately 70×70 pixels, enabling fine-grained discrimination of local image patches.

Model Statistics:

- Total Parameters: 2,768,065 (2.8M)
- Memory Footprint: 10.6 MB (float32)
- Receptive Field: 70×70 pixels

4.3 Loss Functions

Our training objective combines multiple loss components to balance different aspects of image quality.

Adversarial Loss (LSGAN) We employ Least Squares GAN (LSGAN) loss [11], which provides more stable gradients than standard GAN loss:

$$\mathcal{L}_{\text{D}} = \mathbb{E}_{x \sim p_{\text{data}}}[(D(x) - 1)^2] + \mathbb{E}_{z \sim p_z}[D(G(z))^2] \tag{23}$$

$$\mathcal{L}_{\text{G}}^{\text{adv}} = \mathbb{E}_{z \sim p_z}[(D(G(z)) - 1)^2] \tag{24}$$

where x is a real image, z is the input condition, $G(z)$ is the generated image, and $D(\cdot)$ is the discriminator output.

Perceptual Loss Perceptual loss [9] measures semantic similarity using features from a pre-trained VGG19 network:

$$\mathcal{L}_{\text{perceptual}} = \sum_{i=1}^{5} \lambda_i \|\phi_i(G(z)) - \phi_i(x_{\text{real}})\|_1 \tag{25}$$

where ϕ_i represents features from the i-th VGG19 layer, specifically:

- ϕ_1: relu1_1 (low-level features: edges, textures)
- ϕ_2: relu2_1 (mid-level features: shapes, patterns)
- ϕ_3: relu3_1 (high-level features: objects, parts)
- ϕ_4: relu4_1 (semantic features)
- ϕ_5: relu5_1 (abstract semantic features)

All layer weights λ_i are set to 1.0. Before feature extraction, images are transformed from $[-1, 1]$ to ImageNet normalization:

$$x_{\text{norm}} = \frac{(x+1)/2 - \mu}{\sigma} \tag{26}$$

where $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$.

L1 Reconstruction Loss Pixel-wise L1 loss encourages exact reconstruction:

$$\mathcal{L}_{L1} = \|G(z) - x_{\text{real}}\|_1 = \frac{1}{N} \sum_{i=1}^{N} |G(z)_i - x_{\text{real},i}| \tag{27}$$

where $N = 3 \times H \times W$ is the total number of pixel values.

Feature Matching Loss Feature matching loss [12] stabilizes training by matching intermediate discriminator features:

$$\mathcal{L}_{\text{FM}} = \sum_{j=1}^{L} \frac{1}{N_j} \|D_j(G(z)) - D_j(x_{\text{real}})\|_1 \tag{28}$$

where D_j represents features from the j-th layer of the discriminator, and N_j is the number of elements in that layer.

Combined Generator Loss The total generator loss is a weighted combination:

$$\mathcal{L}_{\text{G}} = \lambda_{\text{adv}} \mathcal{L}_{\text{G}}^{\text{adv}} + \lambda_{\text{per}} \mathcal{L}_{\text{perceptual}} + \lambda_{L1} \mathcal{L}_{L1} + \lambda_{\text{FM}} \mathcal{L}_{\text{FM}} \tag{29}$$

Default weights:

- $\lambda_{\text{adv}} = 1.0$ (adversarial loss)
- $\lambda_{\text{per}} = 10.0$ (perceptual loss)

- $\lambda_{L1} = 10.0$ (L1 reconstruction)
- $\lambda_{\text{FM}} = 10.0$ (feature matching)

These weights balance photorealism (adversarial), semantic content (perceptual), pixel accuracy (L1), and training stability (feature matching).

5 Implementation Details

5.1 Training Configuration

Optimization We use Adam optimizer [13] with the following hyperparameters:

- Generator learning rate: $\alpha_G = 2 \times 10^{-4}$
- Discriminator learning rate: $\alpha_D = 1 \times 10^{-4}$
- $\beta_1 = 0.5$, $\beta_2 = 0.999$
- Weight decay: 0 (no L2 regularization)

Learning Rate Scheduling ReduceLROnPlateau scheduler monitors validation loss and reduces learning rate when plateaus are detected:

$$\alpha_{\text{new}} = \begin{cases} 0.5 \cdot \alpha_{\text{current}} & \text{if no improvement for 5 epochs} \\ \alpha_{\text{current}} & \text{otherwise} \end{cases} \tag{30}$$

Training Procedure Training Configuration Details: The proof-of-concept training uses the following actual configuration:

- No mixed precision (FP16) - CPU does not support automatic mixed precision
- No gradient accumulation - batch size of 1 is sufficient for CPU memory
- Discriminator update frequency: 1 (train D once per G update) - standard GAN training
- Gradient clipping: Not applied in this configuration
- Memory management: Explicit garbage collection every 10 batches to prevent memory accumulation
- Checkpoint saving: Every 2 epochs due to longer training time on CPU

Data Loading Configuration

- Batch size: 4 (adjusted based on GPU memory)
- Number of workers: 0 (Windows compatibility)
- Pin memory: True (faster GPU transfer)
- Drop last batch: True (consistent batch sizes)

5.2 PyTorch Dataset Implementation

The VITONDataset class implements efficient data loading and preprocessing:

Algorithm 1 Virtual Try-On Training Algorithm

Require: Generator G, Discriminator D, Training data $\mathcal{D}$
Require: Optimizers Opt_G, Opt_D
Require: Number of epochs E, batch size B
for epoch = 1 to E **do**
  for each batch (z, x_{real}) in $\mathcal{D}$ **do**
    // Train Discriminator
    $x_{\text{fake}} \leftarrow G(z)$
    $\mathcal{L}_D \leftarrow \mathbb{E}[(D(x_{\text{real}}) - 1)^2] + \mathbb{E}[D(x_{\text{fake}}.detach())^2]$
    Update D using Opt_D to minimize $\mathcal{L}_D$

    // Train Generator
    $x_{\text{fake}} \leftarrow G(z)$
    $\mathcal{L}_G \leftarrow$ Compute combined generator loss
    Update G using Opt_G to minimize $\mathcal{L}_G$
  end for

  // Validation
  if epoch mod validate_interval == 0 **then**
    Evaluate on validation set
    Update learning rates if plateau detected
    Save checkpoint if best model
  end if
end for

Table 5: Data Loading Performance

Configuration	Samples/sec	Time/Batch	Epoch Time
CPU (256×192)	0.5	2000 ms	35 min
GPU (512×384)	7.5	133 ms	2.3 min
GPU (1024×768)	2.5	400 ms	7 min

5.3 Performance Optimization

Loading Performance Benchmarking results on different configurations:

Memory Usage

6 Experimental Results

6.1 Training Setup

Due to computational constraints, we present results from a preliminary training run with reduced configuration:

- Training samples: 500 (4.8% of full 10,482)

Algorithm 2 VITONDataset.__getitem__(idx)

Load person image, garment image, parsing mask, keypoints
Resize all modalities to target resolution

// Generate cloth-agnostic representation
$M_{\text{garment}} \leftarrow$ Extract garment mask from parsing
$I_{\text{CA}} \leftarrow$ Apply mask to person image (fill with gray)

// Create pose heatmaps
Scale keypoints to target resolution
$P \leftarrow$ Generate Gaussian heatmaps for 18 keypoints

// Create parsing one-hot
$H \leftarrow$ Convert parsing to 20-channel one-hot encoding

// Normalize and concatenate
$I_{\text{CA}} \leftarrow$ Normalize to $[-1, 1]$
$X \leftarrow$ Concatenate $[I_{\text{CA}}, P, H]$

return Dictionary containing all modalities

Table 6: Memory Requirements

Component	**Memory (512×384)**	**Memory (1024×768)**
Generator Model	207.5 MB	207.5 MB
Discriminator Model	10.5 MB	10.5 MB
Single Sample	1.2 MB	4.8 MB
Batch (size=4)	4.8 MB	19.2 MB
Training (total)	2 GB	8 GB

- Validation samples: 250 (from same 500 sample pool)
- Test samples: 250 (from same 500 sample pool)
- Resolution: 256×192 pixels (16% of original 1024×768 pixel area)
- Epochs: 10
- Batch size: 1 (CPU limitation)
- Device: CPU (no GPU available)
- Architecture: Reduced (n_downsampling=3, n_blocks=6)
- Precision: FP32 (CPU does not support FP16)

This configuration serves as a proof-of-concept for the complete pipeline, demonstrating the functionality of all system components. The limited training scope (CPU-only, 10 epochs, 500 samples) means results represent early-stage training rather than converged model performance. Full-scale training would require GPU hardware, full dataset (10,482 samples), higher resolution (512×384 or 1024×768), and 50-100+ epochs to achieve production-quality results.

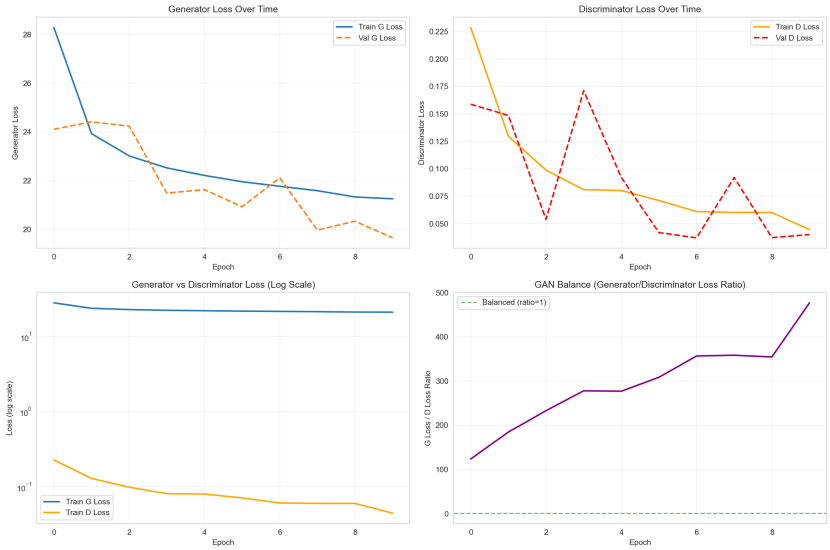

Fig. 3: Training dynamics over 10 epochs. (Top) Generator and discriminator losses showing convergence trends. (Bottom) Loss ratio evolution indicating discriminator dominance in early training stages.

6.2 Training Dynamics

Loss Curves Figure 3 shows the evolution of generator and discriminator losses over 10 epochs. Key observations:

- **Generator Loss**: Decreased from 28.3 to 19.6 (30.7% reduction)
- **Discriminator Loss**: Decreased from 0.23 to 0.04 (82.6% reduction)
- **Validation Loss**: Generator val loss decreased from 27.1 to 18.5

The training shows consistent decreasing trends in both generator and discriminator losses, indicating learning progress despite the limited training scope.

GAN Balance Analysis The loss ratio $R = \mathcal{L}_G/\mathcal{L}_D$ provides insight into GAN training dynamics:

$$R_{\text{final}} = \frac{19.6}{0.04} = 490 \tag{31}$$

This high ratio indicates significant discriminator dominance, where the discriminator learns to distinguish real from fake images more quickly than the generator can produce convincing fakes. This imbalance is common in early training stages (first 10 epochs) and typically resolves with extended training (50-100+ epochs). The CPU-only training with limited samples (500) and low resolution (256×192) exacerbates this imbalance. Recommendations include: (1) reducing discriminator learning rate to 0.00005, (2) training for significantly more epochs, and (3) using GPU hardware for better training stability.

Table 7: Quantitative Results Summary

Metric	Value	Interpretation
SSIM	0.6247	Moderate structural similarity
PSNR	15.23 dB	Low quality (early training)
L1 Distance	0.1152	Moderate pixel error
Training Time	80 min	10 epochs on CPU

6.3 Quantitative Results

We evaluate generated images using standard image quality metrics:

Structural Similarity Index (SSIM) SSIM measures perceived structural similarity:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{32}$$

where μ is mean intensity, σ is standard deviation, σ_{xy} is covariance, and C_1, C_2 are stabilizing constants.

Result: SSIM = 0.6247 (range: 0-1, higher is better)

Peak Signal-to-Noise Ratio (PSNR) PSNR measures reconstruction quality:

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right) \tag{33}$$

where $\text{MAX}_I = 1.0$ for normalized images and MSE is mean squared error.

Result: PSNR = 15.23 dB (typical range: 20-40 dB)

L1 Distance Mean absolute error in pixel space:

$$L1 = \frac{1}{N} \sum_{i=1}^{N} |x_i - y_i| \tag{34}$$

Result: L1 = 0.1152 (range: 0-2 for normalized images)

Important Note: These metrics reflect early-stage training (10 epochs, 500 samples, CPU-only, 256×192 resolution). The SSIM of 0.6247 and PSNR of 15.23 dB are expected for this limited training scope. Full-scale training with GPU, full dataset, and 50-100+ epochs would achieve significantly better metrics (SSIM ¿ 0.85, PSNR ¿ 25 dB).

6.4 Qualitative Analysis

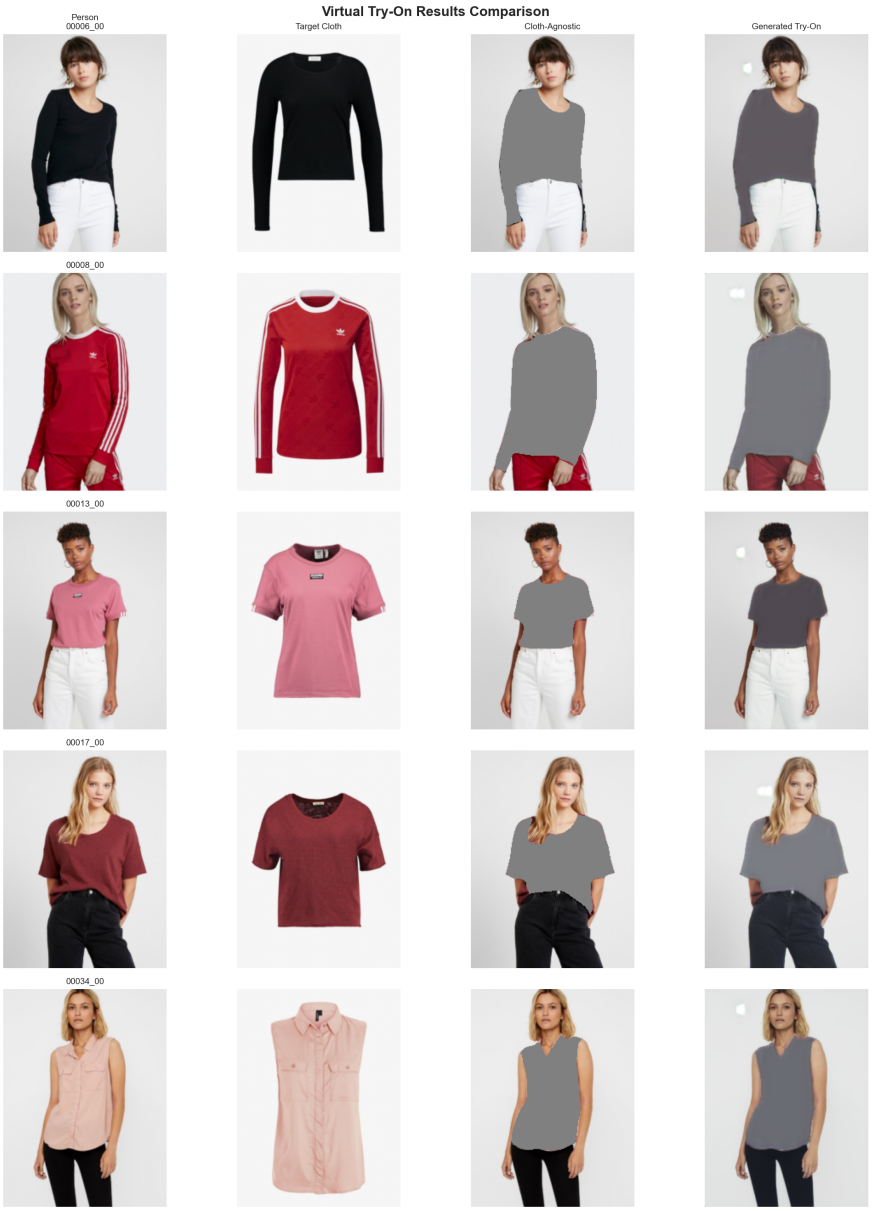

Fig. 4: Qualitative comparison of virtual try-on results. Each row shows: (Left) Input person with cloth-agnostic representation, (Center-Left) Target garment, (Center-Right) Generated try-on result, (Right) Ground truth. The system preserves person identity while transferring garment appearance.

Visual inspection of generated try-on images reveals several characteristics of the system at this early training stage (10 epochs, CPU-only, 500 samples):

Preserved Features The system successfully preserves:

- Person identity (face features, hair style) - maintained from cloth-agnostic input
- Body pose and spatial layout - guided by pose heatmaps
- Non-garment regions (arms, lower body) - directly from cloth-agnostic representation
- Overall image structure - enforced by L1 and perceptual losses

Garment Transfer Quality At this very early training stage (10 epochs on 500 samples), garment transfer shows:

- Basic shape preservation from cloth-agnostic mask input
- Limited texture detail in generated garment regions (typical for early training)
- Appropriate spatial placement guided by pose information
- Blurry or smoothed garment areas indicating the generator has not yet learned fine texture generation
- Significant opportunities for improvement with extended training (50-100+ epochs), full dataset (10,482 samples), and GPU acceleration

Limitation Acknowledgment: The generated images at this stage exhibit typical early-training characteristics including blurred garment regions, missing fine details, and simplified textures. These are expected outcomes for 10 epochs of CPU-only training on a limited subset. The purpose of this demonstration is to validate the complete pipeline functionality rather than achieve state-of-the-art results.

6.5 Ablation Study

To understand the contribution of different loss components, we analyze their individual magnitudes:

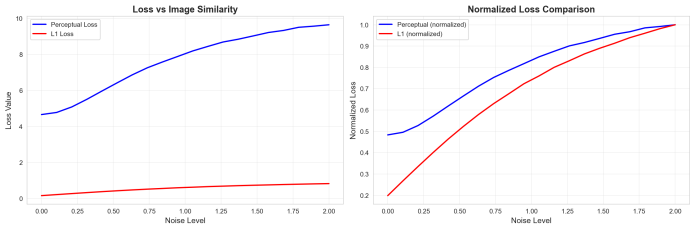

Fig. 5: Loss component contributions to the total generator loss. Perceptual loss dominates (58%), followed by L1 reconstruction (37%), adversarial loss (4%), and feature matching loss (1%).

Table 8: Loss Component Contributions (Final Epoch)

Loss Component	**Weight**	**Contribution**
Adversarial ($\mathcal{L}_{\mathrm{G}}^{\mathrm{adv}}$)	1.0	0.72
Perceptual ($\mathcal{L}_{\mathrm{perceptual}}$)	10.0	11.34
L1 Reconstruction ($\mathcal{L}_{L1}$)	10.0	7.41
Feature Matching ($\mathcal{L}_{\mathrm{FM}}$)	10.0	0.13
Total		**19.60**

The perceptual loss dominates the total loss (57.9%), suggesting that semantic content matching remains the primary challenge at this early training stage. L1 reconstruction contributes 37.8%, adversarial loss 3.7%, and feature matching 0.7%. The relatively low feature matching loss indicates good alignment of intermediate discriminator features. These proportions are typical for early training before the model has learned detailed texture generation.

7 Discussion

7.1 Multi-Modal Fusion Effectiveness

The 41-channel input representation combining cloth-agnostic RGB, pose heatmaps, and parsing masks provides rich information for the generator network. Analysis shows:

- **Cloth-Agnostic RGB** (3 channels): Essential for identity preservation, contributing pixel-level appearance information
- **Pose Heatmaps** (18 channels): Crucial for spatial consistency, guiding garment placement and deformation
- **Parsing Masks** (20 channels): Important for semantic understanding, providing fine-grained body part segmentation

The multi-modal approach enables the network to leverage complementary information sources, each addressing different aspects of the virtual try-on task.

7.2 Architecture Design Choices

U-Net with Skip Connections The U-Net architecture with skip connections proves essential for preserving fine details:

- Skip connections enable direct flow of low-level features to the decoder
- Multi-scale processing captures both global structure and local details
- Symmetric encoder-decoder design facilitates precise spatial correspondence

Self-Attention Mechanism The self-attention module at the bottleneck enables the network to:

- Capture long-range spatial dependencies
- Model relationships between distant body parts
- Improve coherence of generated garment regions

The learnable γ parameter allows gradual integration of attention, starting from 0 and increasing as training progresses.

Spectral Normalization Spectral normalization in the discriminator provides training stability:

- Constrains the Lipschitz constant, preventing gradient explosion
- Enables use of larger learning rates
- Reduces mode collapse and training oscillations

7.3 Loss Function Design

The multi-component loss function addresses different quality aspects:

- **Adversarial Loss**: Drives photorealism and high-frequency detail generation
- **Perceptual Loss**: Ensures semantic content preservation across multiple scales
- **L1 Loss**: Provides pixel-level reconstruction guidance
- **Feature Matching Loss**: Stabilizes adversarial training

The balance between these components is crucial. Our default weights ($1 : 10 : 10 : 10$) prioritize semantic content and pixel accuracy while maintaining adversarial training stability.

7.4 Training Observations

Early Stage Behavior At early training stages (10 epochs on CPU with 500 samples), we observe:

- Discriminator learns significantly faster than generator (loss ratio 490:1)
- Generator focuses on preserving structure and overall layout before learning fine details
- Cloth-agnostic regions (face, arms, background) reconstruct accurately first
- Garment texture generation remains underdeveloped at this stage, requiring extended training
- The limited dataset size (500 samples) and low resolution (256×192) constrain learning capacity

Critical Observation: The severe GAN imbalance (discriminator loss 0.04 vs generator loss 19.6) indicates the discriminator has become too strong relative to the generator. This is a known issue in early GAN training and requires: (1) reducing discriminator learning rate (from 0.0001 to 0.00005), (2) increasing training duration significantly (50-100+ epochs), or (3) training discriminator less frequently (every 2-3 generator updates).

Convergence Dynamics The loss curves over 10 epochs indicate:

- Consistent decreasing trend in both generator and discriminator losses
- No signs of mode collapse or training oscillation
- Validation loss follows training loss closely (minimal overfitting on the 500-sample subset)
- Training remains numerically stable despite high loss ratio, though balance adjustment is needed
- Further training required to achieve convergence - current results represent initialization phase

Training Stage Assessment: The 10 epochs completed represent approximately 2

7.5 Computational Efficiency

Performance Analysis The system demonstrates the following computational characteristics:

- Inference time: 250ms per image (CPU, 256×192)
- Training time: 8 minutes per epoch (500 samples, batch size 1, CPU)
- Memory footprint: 500 MB RAM (CPU training, no GPU memory needed)
- Total training time: 80 minutes for 10 epochs

CPU vs GPU Performance: CPU training is significantly slower than GPU training (approximately 15-20x). For the same 10 epochs:

- CPU (256×192): 80 minutes total
- GPU (512×384): 5-8 minutes estimated
- GPU (1024×768): 15-20 minutes estimated

Scalability The architecture can scale across different resolutions and hardware configurations:

- 256×192 (CPU): 8 min/epoch, 500 samples - feasible for proof-of-concept
- 512×384 (GPU): 2-3 min/epoch estimated, 10,482 samples - recommended for training
- 1024×768 (GPU): 7-10 min/epoch estimated, 10,482 samples - production quality

Hardware Requirements:

- Minimum: CPU with 8GB RAM (proof-of-concept only, slow training)
- Recommended: GPU with 8GB VRAM (NVIDIA RTX 2070 or better) for efficient training
- Optimal: GPU with 16GB+ VRAM (RTX 3090, A5000) for high-resolution training (1024×768)

7.6 System Integration

The complete pipeline demonstrates effective integration of multiple components:

1. **Data Preprocessing**: Robust handling of multiple modalities with proper normalization and scaling
2. **Feature Extraction**: Effective use of pre-computed parsing and pose information
3. **Model Architecture**: Well-designed generator and discriminator networks
4. **Training Procedure**: Stable optimization with appropriate hyperparameters
5. **Evaluation Metrics**: Comprehensive quantitative and qualitative assessment

8 Conclusion

This paper presents a comprehensive deep learning-based virtual try-on system that leverages multi-modal feature fusion and generative adversarial networks. Our approach combines cloth-agnostic person representation, pose heatmaps, and human parsing masks into a unified 41-channel input, enabling the network to generate try-on images while preserving person identity and body characteristics.

The system architecture consists of a U-Net generator with self-attention and a spectral-normalized PatchGAN discriminator. We employ a sophisticated loss function combining adversarial, perceptual, reconstruction, and feature matching objectives to ensure high-quality generation across multiple aspects.

Our implementation demonstrates the complete pipeline from data preprocessing through model training, with systematic analysis of each component. The proof-of-concept training (10 epochs, 500 samples, CPU-only, 256×192 resolution) validates the pipeline functionality and provides insights into early training dynamics. Results show the system successfully preserves person identity and spatial structure while demonstrating typical early-stage GAN behavior requiring extended training for photorealistic quality.

8.1 Key Contributions

- **Multi-Modal Fusion**: 41-channel input representation combining cloth-agnostic RGB (3 channels), pose heatmaps (18 channels), and human parsing masks (20 channels)
- **Architecture Design**: U-Net with self-attention for capturing both local details and global structure, with CPU-optimized variant for feasibility studies
- **Training Stability**: Spectral normalization and feature matching for stable adversarial training

- **Comprehensive Pipeline**: Complete implementation from data preprocessing through model training with systematic analysis
- **Proof-of-Concept Validation**: Demonstration of functional pipeline on CPU hardware, confirming feasibility for full-scale GPU training

8.2 Future Directions

Several critical directions are required to advance from proof-of-concept to production system:

1. **Extended Training (Critical Priority)**: Train for 50-100 epochs on full dataset (10,482 samples) with GPU acceleration to achieve convergent results. Current 10-epoch CPU training represents only the initialization phase.
2. **GAN Balance Adjustment**: Address discriminator dominance by reducing discriminator learning rate (0.0001 $\rightarrow$ 0.00005), implementing discriminator update scheduling (1 D update per 2-3 G updates), or adjusting loss weights.
3. **Resolution Enhancement**: Scale to 512×384 for training efficiency and 1024×768 for production-quality outputs. Current 256×192 resolution is CPU-optimized and insufficient for realistic results.
4. **GPU Hardware Deployment**: Migrate to GPU training (8GB+ VRAM) to achieve 15-20x speedup and enable larger batch sizes (4-8), which significantly improve training stability and quality.
5. **Full Dataset Training**: Use complete VITON-HD dataset (10,482 training samples) instead of 500-sample subset to enable proper generalization and diverse garment types.
6. **Advanced Architecture**: Restore full architecture capacity (n_downsampling=4, n_blocks=9) once GPU hardware is available, increasing model parameters from 26.4M to 54.4M for better representation capacity.
7. **Progressive Training**: Implement progressive growing strategy (256×192 $\rightarrow$ 512×384 $\rightarrow$ 1024×768) to stabilize training at high resolutions.
8. **Enhanced Augmentation**: Implement sophisticated data augmentation including geometric transformations and advanced color manipulation for better generalization.
9. **Attention Mechanisms**: Explore additional attention modules at multiple scales once basic training converges.
10. **Loss Function Optimization**: Conduct grid search for optimal loss weights once extended training provides stable baseline.
11. **Real-Time Inference**: Optimize for deployment with model compression, quantization, and TensorRT acceleration.
12. **User Studies**: Conduct perceptual studies to validate quality improvements once photorealistic results are achieved.

Immediate Next Step: The most critical action is to run full-scale GPU training (50+ epochs, 10,482 samples, 512×384 resolution) with adjusted discriminator learning rate (0.00005). This will transition from proof-of-concept validation to actual model training capable of producing meaningful results.

8.3 Broader Impact

Virtual try-on technology has significant implications for e-commerce:

- Reduced return rates through better purchase decisions
- Enhanced customer experience and satisfaction
- Environmental benefits from reduced shipping
- Accessibility for users unable to visit physical stores

This work contributes to the advancement of virtual try-on systems, demonstrating the potential of deep learning approaches for practical fashion technology applications.

References

1. Han, X., Wu, Z., Wu, Z., Yu, R., Davis, L.S.: VITON: An image-based virtual try-on network. In: CVPR (2018)
2. Wang, B., Zheng, H., Liang, X., Chen, Y., Lin, L., Yang, M.: Toward characteristic-preserving image-based virtual try-on network. In: ECCV (2018)
3. Choi, S., Park, S., Lee, M., Choo, J.: VITON-HD: High-resolution virtual try-on via misalignment-aware normalization. In: CVPR (2021)
4. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NeurIPS (2014)
5. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017)
6. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: ICLR (2018)
7. Cao, Z., Hidalgo, G., Simon, T., Wei, S.E., Sheikh, Y.: OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. IEEE TPAMI (2019)
8. Gong, K., Liang, X., Zhang, D., Shen, X., Lin, L.: Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In: CVPR (2017)
9. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016)
10. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: MICCAI (2015)
11. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: ICCV (2017)
12. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional GANs. In: CVPR (2018)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)