# Acknowledgment

We would like to express our deepest gratitude to our respected supervisor, Dr. Hala Abdel-Galil, for her invaluable guidance, continuous support, and endless patience throughout the journey of this project.

Her insightful feedback, encouraging words, and dedication to our learning were key factors in shaping our progress and pushing us to reach our full potential. We are truly thankful for the time and effort she invested in us, and for always being there with constructive advice and motivating energy whenever we faced challenges.

This project would not have reached its current form without her remarkable mentorship.

# Abstract

In today's digital age, consumers increasingly rely on online shopping for convenience and accessibility. However, one significant drawback of online shopping is the inability to physically try on clothing before making a purchase.

This limitation often leads to uncertainty regarding style, resulting in customer post-purchase dissatisfaction and higher return rates. Based on research, during the pandemic, items bought online are three times more likely to be returned than those bought in-store.

FASHONISTA was conceived to bridge the gap between traditional in-store try-ons and online shopping by offering users a realistic and interactive virtual try-on experience.

Although virtual try-ons already exist, recent advancements in Artificial Intelligence have significantly enhanced their capabilities. AI has become increasingly prominent, enabling more sophisticated and realistic virtual try-on experiences than ever before.

Our approach integrates multiple deep learning techniques, including pose estimation, human parsing, and generative adversarial networks (GANs), to generate realistic images of a person wearing selected garments.

The system takes as input an image of a person and a separate image of a clothing item, and outputs a synthesized image that realistically combines both.

We aim to enhance the online shopping experience, reduce return rates, and support sustainable fashion practices by allowing users to make better-informed decisions before purchasing clothing online.

# Chapter 1

# Introduction

## 1.1 Overview

The proposed virtual try-on system is composed of four main stages, each responsible for a crucial step in the try-on pipeline:

**1. Pre-processing**

In this step, the original person image is processed to remove both the existing clothes and the arms from the segmentation and input image. Pose keypoints are extracted, and a clean version of the person (without arms and clothes) is generated. This prepares the input for more accurate garment overlay in the later stages.

**2. Segmentation Generation**

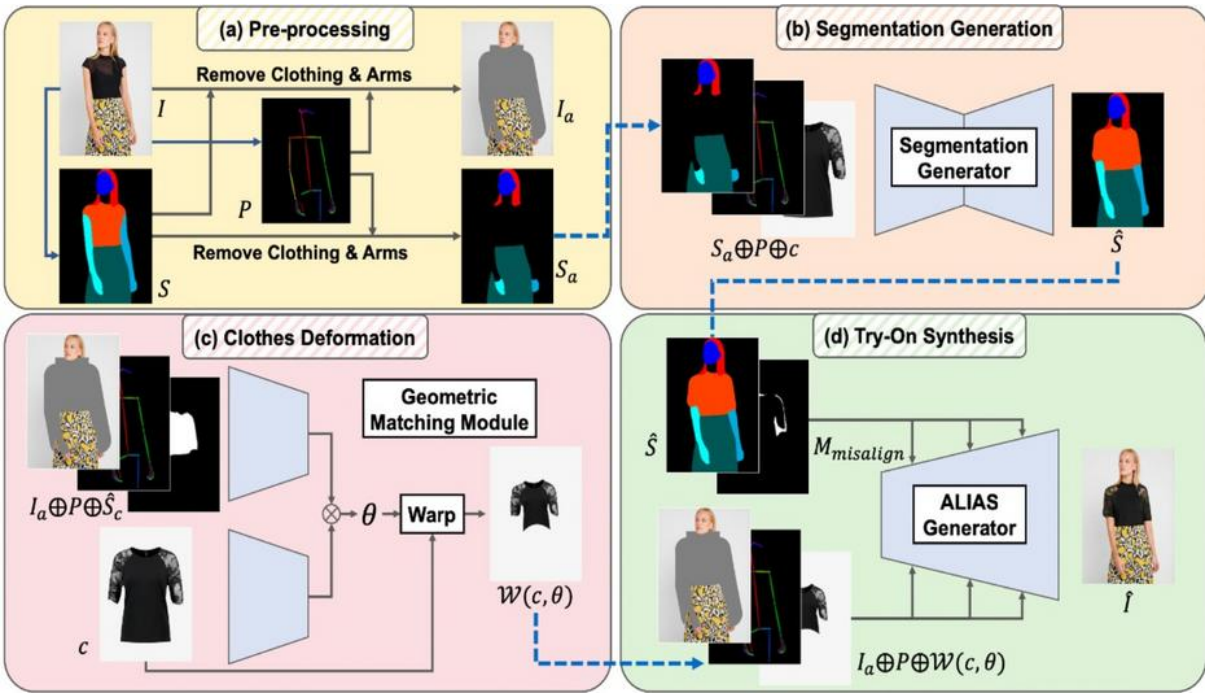
The system takes the processed segmentation map, pose keypoints, and the clothing item as input, and feeds them into the Segmentation Generator. The generator outputs a refined human parsing map that defines how the clothing should fit on the body, ensuring proper alignment of all parts.

**3. Clothes Deformation**

This module focuses on warping the garment to match the target body shape. A Geometric Matching Module computes transformation parameters (θ) to align the clothing item using pose and segmentation cues. The warped clothing is then ready to be synthesized onto the person.

**4. Try-On Synthesis**

In the final stage, all processed data—including the warped clothing, pose, segmentation map, and a misalignment mask—are passed to the ALIAS Generator. This generator synthesizes the final output image, where the person appears to be naturally wearing the target garment, with photorealistic quality and preserved identity.




**Figure#.** Overview

## 1.2 Problem Statement

Despite the growing popularity of online shopping, the inability to physically try on clothing remains a significant challenge, leading to uncertainties about fit and style and resulting in higher return rates and customer dissatisfaction.

Existing virtual try-on methods often produce low-resolution and unrealistic outputs.

This limitation requires the development of a novel solution to offer users a realistic and interactive virtual try-on experience.

FASHONISTA aims to bridge the gap between traditional in-store try-ons and online shopping by providing high-quality, lifelike representations of users wearing in-shop garments, thereby addressing the challenges associated with online apparel shopping.

## 1.3 Objective

The main objectives of this project are as follows:

1. To design and implement a functional virtual try-on system using deep learning and computer vision techniques.

2. To reduce uncertainty in online shopping by providing a realistic preview of how garments will appear on the user.
3. To minimize clothing return rates by improving user confidence in purchase decisions.
4. To enhance the user experience in online retail platforms with interactive and personalized features.
5. To explore and gain hands-on experience with modern machine learning tools such as GANs, pose estimation models, and segmentation networks.

## 1.4 Purpose

The purpose of our project is to provide a practical, AI-powered solution to a common issue in online fashion retail: the inability to try on clothes before purchasing. Many customers face challenges in selecting the right size, style, or fit when shopping online, which often leads to disappointment and product returns.

Through this system, we aim to simulate the in-store fitting room experience virtually, using advanced machine learning models to make the process accurate, quick, and user-friendly. Our goal is to make the online shopping journey more transparent and satisfying for both customers and retailers.

At the same time, this project allows us to apply what we've learned in our academic journey to solve a real problem innovatively.

## 1.5 Scope

**1. Functional Coverage**

- The system allows users to upload a frontal photo of themselves and a clothing image or take a photo using the camera.
- It generates a realistic 2D image of the user wearing the selected clothing item.
- Focus is on upper-body garments like shirts, t-shirts, and jackets.
- The final image is synthesized using deep learning models trained on public datasets.

**2. Technologies Used**

- *Pose Estimation:* To understand the user’s body position and alignment.

- *Human Parsing:* To segment different body parts for accurate clothing overlay.
- *ASGenerator:* To generate photo-realistic try-on results.
- We used Python, PyTorch, and several pre-trained models in the implementation.

**3. Output and User Experience**

- The system provides a visual-only try-on experience.
- The output is a static image that reflects the selected outfit on the uploaded photo.
- Designed to improve online shopping decisions and reduce uncertainty.

**4. Out of Scope**

- No 3D modeling or garment rotation is included.
- The system doesn’t calculate sizes or recommend fitting.
- Doesn’t support accessories or full-body clothing items.

**5. Target Users and Applications**

- Online shoppers who want to preview clothes visually.
- E-commerce platforms are looking to enhance customer experience.
- Fashion tech researchers and developers are exploring virtual try-on technologies.

# 1.6 General Constraints

**1. Dataset Limitations**

- The system was trained using publicly available datasets such as DeepFashion and VITON.
- These datasets may not cover all clothing types, skin tones, or body shapes, which limits the model's generalization ability.
- The model performs best on clean, frontal images under good lighting.

**2. Hardware Constraints**

- Training the models requires high-performance GPUs, which were limited during development.
- Some optimizations were made to run the model efficiently on medium-spec machines, but training time was still significant.
- The final application may be slow on devices with limited processing power.

### 3. Accuracy Constraints

- The system focuses on visual appearance only and does not ensure physical accuracy (e.g., fitting, size).
- Generated images might have artifacts or inconsistencies, especially with complex clothing textures or occlusions.
- The model assumes a neutral pose and doesn’t perform well with rotated or non-standard postures.

### 4. Clothing Type Constraints

- Currently supports only upper-body garments (e.g., shirts, jackets, t-shirts).
- Full-body outfits, pants, dresses, and accessories are not supported in this version.
- Clothing images need to be clear and well-aligned to work properly.

# Chapter 2

# Related Work (Literature Review)

## 2.1 Related Work:

**Image-based virtual try-on** methods focus on generating realistic visualizations of a person wearing a selected garment, ensuring that the final image maintains the original body posture, facial features, and overall identity of the user.

**Clothes Deformation:** To maintain the visual features of clothing items, many earlier virtual try-on methods, including VITON, use warping modules to adjust garments according to a person's shape. VITON, for instance, uses an encoder-decoder to generate a rough image, which is then refined using warped clothes obtained through TPS transformations. Despite continuous improvements, such warping-based systems still struggle with alignment, often leading to visual artifacts, especially around complex areas like arms. Moreover, without incorporating detailed person-specific data or accounting for deformation, these models often fail to generate accurate or realistic try-on results.

Segmentation-Based Try-On Generation: High-resolution virtual try-on systems often incorporate segmentation generation modules, as the role of segmentation maps becomes more critical with increased image quality. For instance, VITON-HD introduced a normalization technique to address misalignment, but it struggled to naturally apply clothing textures in mismatched areas.

Most of the recent approaches still rely on GANs for image generation. In HR-VITON, the researchers improved alignment by combining warping and segmentation stages into a single pipeline, which resulted in higher quality outputs.

Similarly, ACGPN uses masking techniques to remove the clothing area from the person's image and then regenerate it with the target garment. While effective in terms of structure, this approach heavily depends on accurate human parsing and often loses the fine texture details of the target clothes, as it emphasizes the silhouette over texture fidelity.

A major drawback of segmentation-heavy methods is their sensitivity to errors—small inaccuracies in parsing can produce visibly distorted try-on images.

**Generation phase and refinement of the result:** Refining the image output is another key direction in virtual try-on research. For instance, the Dress Code framework by Morelli et al. introduced a pixel-level semantic-aware discriminator to improve the quality of generated try-on images by focusing on fine-grained visual features. This method diverged from traditional image- or patch-level discrimination to allow for more precise generation. However, the system occasionally produced low-resolution outputs when facing challenging or ambiguous inputs.

Parser-free virtual try-on methods emerged to reduce dependency on human parsing, yet early approaches encountered significant visual artifacts due to training both Teacher and Student networks on identical input-output pairs. To overcome this limitation, Yuying Ge proposed PF-AFN, a knowledge distillation framework where the Teacher model's output becomes the Student's input, while the Student's output is guided by ground truth images. This technique marked a major improvement in training dynamics and set the foundation for later parser-free techniques.

In a related advancement, RMGN enhanced image generation quality by integrating SPADE blocks. However, these methods still rely—either directly or indirectly—on human representation models derived from parsing, which makes them computationally expensive.

In recent years, diffusion models have emerged as a powerful alternative to GANs for generating high-quality and realistic images. Despite their impressive results, these models are known for their heavy computational requirements. To address this, Rombach et al. introduced a latent diffusion approach that operates within the latent space of a pre-trained autoencoder, significantly reducing computational load while preserving image quality.

Building on this advancement, the LaDI-VTON model successfully applied latent diffusion models (LDMs) to the virtual try-on task, allowing for more detailed and controllable image generation.

It became the first latent diffusion-based framework specifically designed for try-on systems and achieved state-of-the-art results on standard benchmarks like Dress Code and VITON-HD.

Inspired by this work, we developed our model Our project using LaDI-VTON as a foundation, while introducing our own enhancements and novel modules to further improve realism, efficiency, and flexibility in the virtual try-on process.

# 2.2 Background:

### 2.2.1 Dataset

One of the major limitations in virtual try-on research is the scarcity of large-scale, high-resolution, and publicly available datasets. While virtual try-on has become a crucial tool in the e-commerce industry, most of the high-quality datasets remain private and unavailable for research purposes.

Public datasets that are accessible often lack diversity, contain limited examples, or do not include paired images of clothing items and models.

Moreover, the image resolution in many of these datasets is low (commonly 256×192), which negatively impacts the ability of models to generalize or produce high-fidelity results.

To address this challenge, we chose to train and evaluate our architecture using the VITON-HD dataset—a widely accepted benchmark for virtual try-on systems. VITON-HD offers a rich variety of images in terms of clothing categories, body poses, and textures, all at a significantly higher resolution (1024×768) compared to earlier datasets.

This variety allows the model to handle more complex try-on scenarios while maintaining a high level of visual realism and garment preservation.

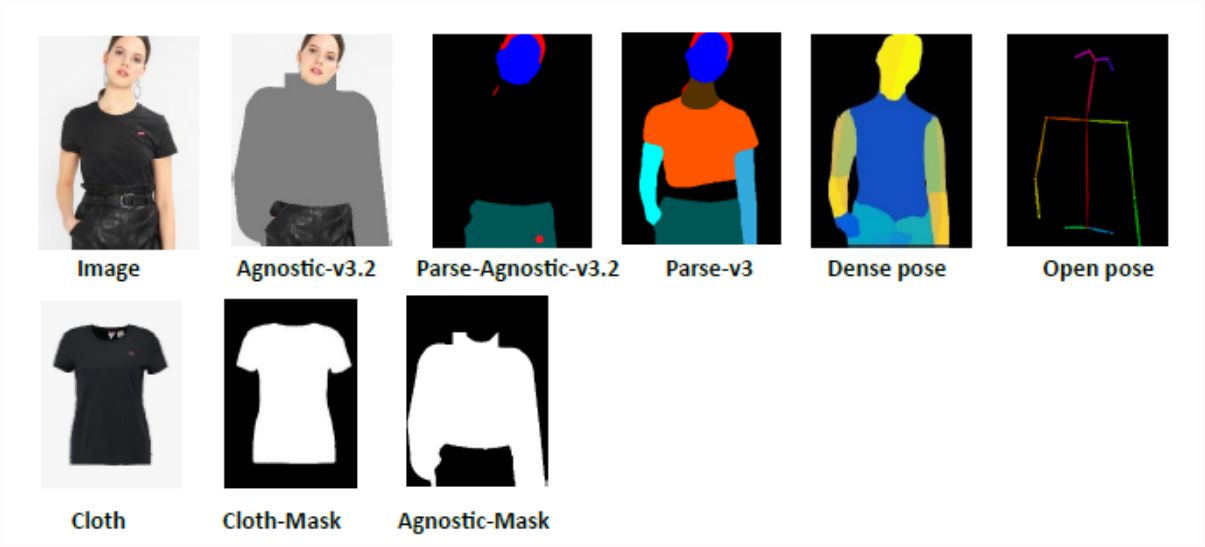
***VITON-HD Dataset***

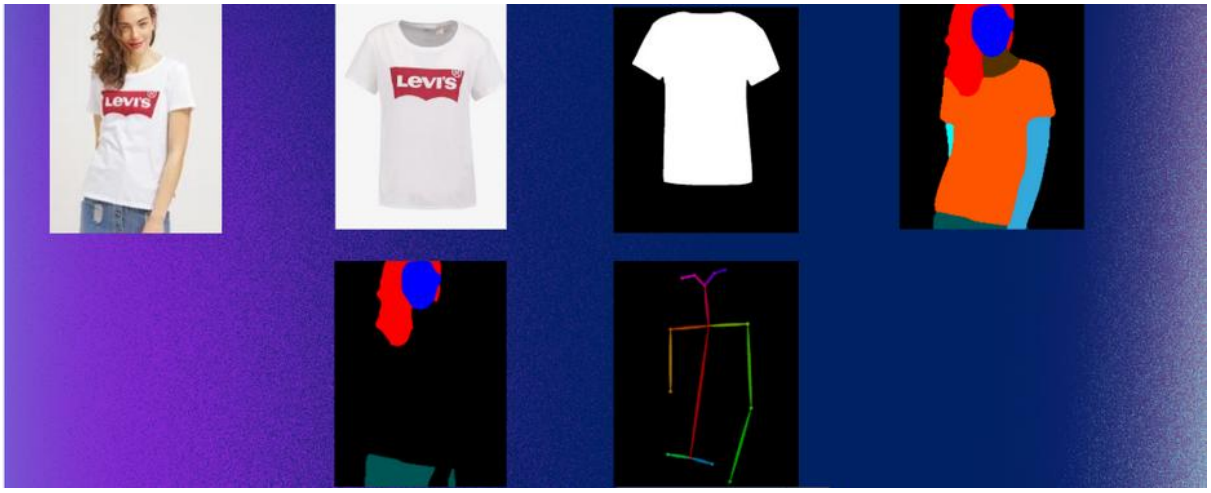
VITON-HD is a high-resolution dataset (1024×768) specifically designed for virtual try-on applications. It includes 13,679 image pairs, each consisting of a frontal-view image of a woman and a corresponding upper-body clothing item. The dataset is divided into two parts: 11,647 pairs for training and 2,032 for testing.

The dataset supports both paired and unpaired evaluation settings. In the paired setting, the model is expected to reconstruct the original person image using the same clothing item. In the unpaired setting, the model replaces the original clothing with a new one to test generalization capabilities.

VITON-HD uses a rich combination of tools and techniques to prepare the data. It leverages masks to separate the person from the background, uses both DensePose and OpenPose for detailed pose estimation, and applies human parsing to segment body parts.

It also includes agnostic representations—masks and parsing that ignore clothing details—so the model focuses on body shape rather than existing apparel. A specific mask for garments is also included, helping the network isolate and manage clothing regions more effectively.



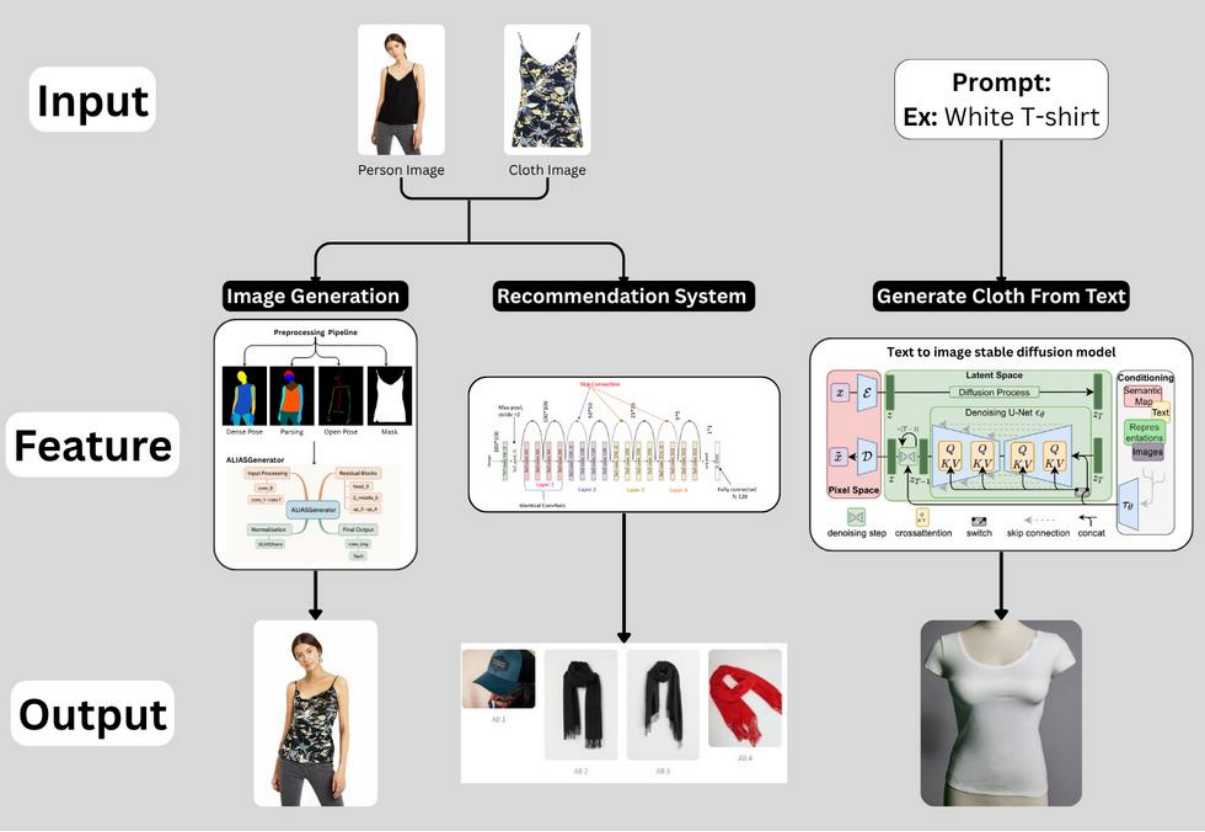
**Figure#.** VITON-HD dataset categories

# Chapter 3

# The Proposed Solution

## 3.1 Proposed System / System Architecture

### 3.1.1 Proposed System




**Figure#.** Proposed System

In our proposed system, the architecture is divided into three main components that work together to deliver a complete virtual try-on experience: **(1) Image Generation, (2) Recommendation System, and (3) Text-to-Cloth Generation**. Each of these components plays a specific role in producing personalized, realistic, and interactive results for users.

**1) Image Generation Module**

This module is the core of our virtual try-on system. It begins with a comprehensive preprocessing pipeline, where the original person image is

passed through several steps, including DensePose for 3D surface mapping, human parsing for semantic body segmentation, OpenPose for skeletal keypoint detection, and mask generation to remove the original clothing. These steps produce a clean, clothing-free representation of the person, ready for try-on.

After preprocessing, the **ALIAS Generator** receives the parsed body, pose map, and warped clothing image as inputs. It then synthesizes a photorealistic image showing the user wearing the selected garment. This generator is designed to maintain fine details such as facial identity, body shape, and garment texture, making the output visually convincing.

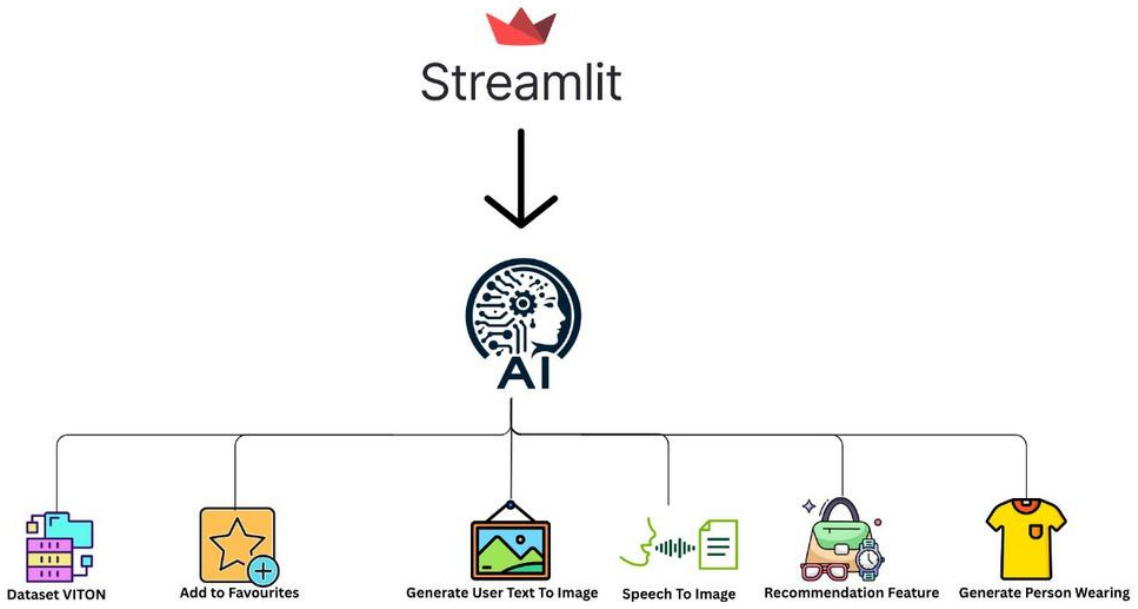
**(2) Recommendation System**

This part of the system enhances the user experience by offering fashion suggestions that go beyond individual garments. We use a CNN-based embedding extractor **(ResNet18)** to generate feature representations of clothing items. Based on these embeddings and user preferences, the system recommends complementary items such as scarves, outerwear, and shoes to complete the outfit. The recommendations are gender-aware and style-sensitive, ensuring they match the user's intended look.

**(3) Generate Cloth from Text (Prompt-Based Synthesis)**

To offer even more customization, our system includes a text-to-image generation module powered by **Stable Diffusion.** Users can input a text description, such as "White oversized hoodie," and the system will generate a realistic image of that garment. The process uses a **CLIP-based** text encoder and a **U-Net** denoising model to convert the prompt into a detailed image. This image can then be tried on virtually using the same pipeline as above.

### 3.1.2 System Architecture




System Architecture Explanation for Virtual Try-On Application

Based on the provided diagram and the previous flowchart, here's an explanation of the system architecture for this virtual try-on application:Core Components

1. **Streamlit Frontend**

- Serves as the web application interface
- Handles user interactions and input collection
- Displays virtual try-on results and recommendations
- Includes:
  - Image upload interfaces
  - Text input for clothing descriptions
  - Favorites management
  - Recommendation displays

2. **Dataset (VITON)**

- Likely uses the VITON (Virtual Try-On Network) dataset or similar
- Contains:
  - Clothing item images
  - Pre-processed garment data
  - Possibly user try-on history for recommendations
- **3.Speech-to-Image Generation for Virtual Try-On**

- This feature would convert spoken clothing descriptions into visual garment representations, expanding accessibility and user experience. Here's how it compares to text-to-image generation:
- **Core Components**
- **Speech Recognition (Speech-to-Text)**
- Converts user's spoken words into text descriptions
- Recommended models/tools:
  - **Whisper** (OpenAI's open-source ASR)
  - **Google Speech-to-Text**
  - **Mozilla DeepSpeech**
-

4. **Text-to-Image Generation**

- Converts textual clothing descriptions into visual representations
- May use models like:
  - Stable Diffusion
  - DALL-E
  - CLIP-guided diffusion models

5. **Virtual Try-On Engine**

- "Generate Person Wearing" component
- Likely uses a GAN (Generative Adversarial Network) architecture
- Potentially based on:
  - U-VTON (from your previous diagram)
  - CP-VTON
  - ACGPN architectures

6. **Recommendation System**

- Suggests clothing items based on:
  - User preferences
  - Past selections
  - Current fashion trends
  - Similar users' choices

7. **Favorites Management**

- Stores and retrieves user-preferred items

- May integrate with recommendation system

### Workflow Architecture

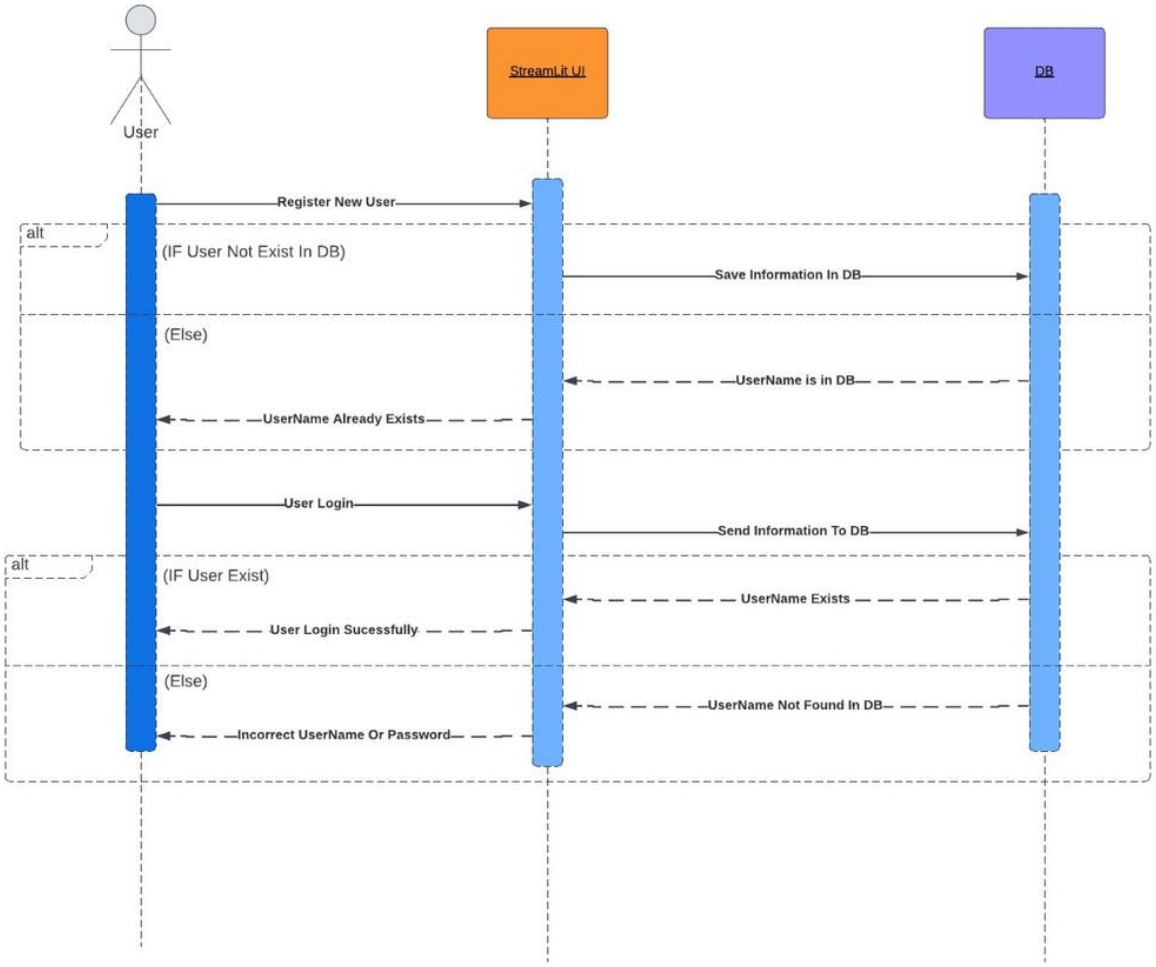
1. **Input Layer** (Streamlit):
    - Receives either:
        - Image uploads (person + clothing)
        - Text descriptions
        - Camera input
1. **Processing Layer**:
    - Image preprocessing (background removal, pose detection)
    - Text-to-image conversion when needed
    - Feature extraction from input garments
1. **Try-On Generation**:
    - Warps clothing to match user's pose
    - Generates realistic try-on images
    - Handles occlusions and lighting adjustments
1. **Output Layer**:
    - Displays virtual try-on results
    - Shows recommendations
    - Manages favorites
1. **Data Storage**:
    - User profiles
    - Favorites lists
    - Interaction history for recommendations

### Technical Stack (Inferred)

- **Frontend**: Streamlit (Python-based web framework)
- **Backend**: Likely Python with:
    - PyTorch/TensorFlow for deep learning models
    - OpenCV for image processing
- **Models**:
    - U-VTON/CP-VTON for virtual try-on
    - Stable Diffusion variants for text-to-image
    - Recommendation algorithms (collaborative filtering/content-based)
- **Storage**:
    - VITON dataset

- User data database (SQL/NoSQL)

### 3.1.3 Sequence Diagram




**Figure#.** Sequence Diagram for login

The sequence diagram shows the User, StreamLit UI, and DB interaction for registration and login:

registration:User sends "Register New User" to StreamLit UI.

If username doesn't exist, UI saves info to DB.

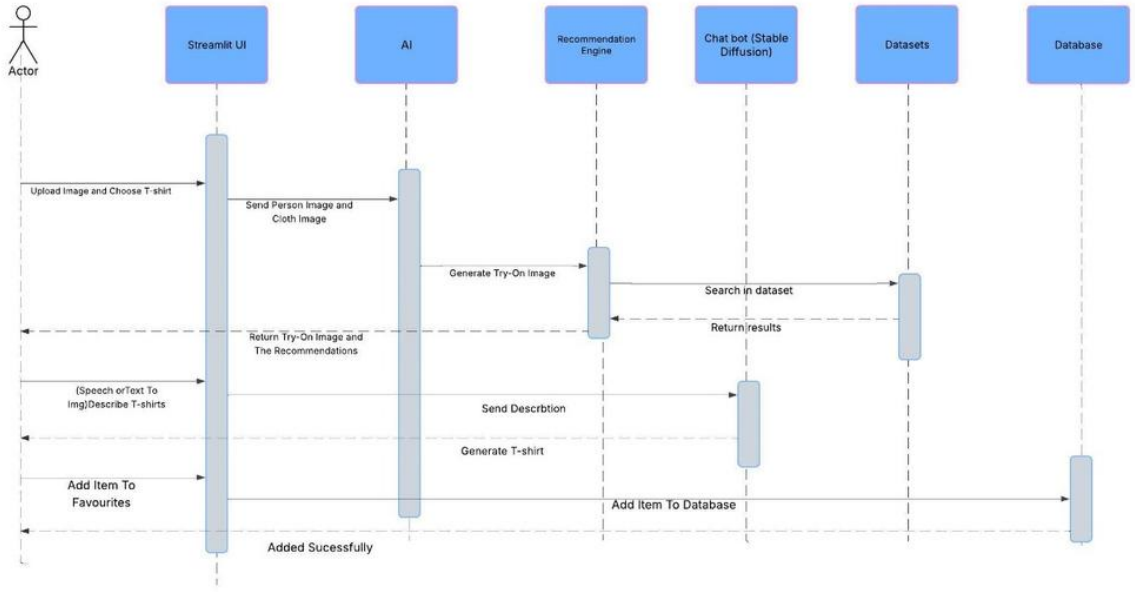
If username exists, UI returns "UserName Already Exists."

Login:

User sends "User Login" to StreamLit UI.

UI checks DB; if username exists, returns "User Login Successfully."

If username not found, returns "Incorrect UserName Or Password."




**Figure#.** Sequence Diagram

Sequence Explanation:

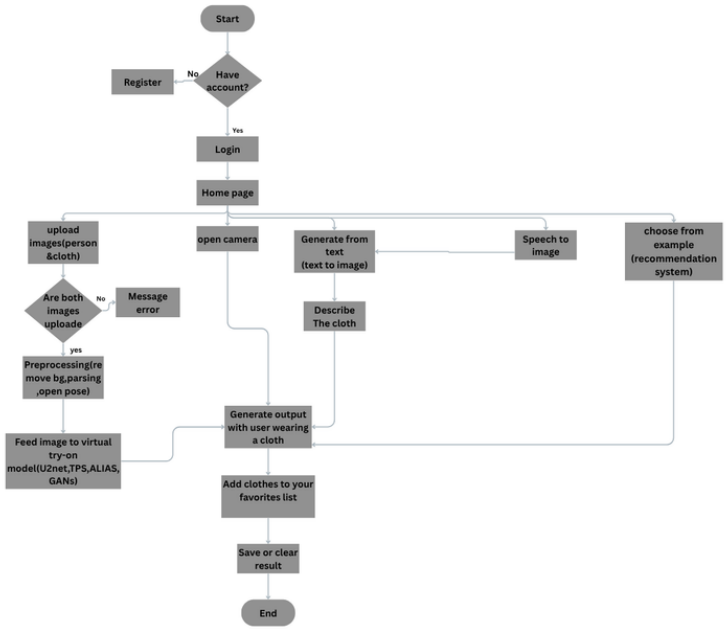
The sequence diagram illustrates the interaction flow of the virtual try-on system implemented using this approach.

Streamlit. The system supports two primary features: virtual try-on with accessory recommendation and T-shirt search via a chatbot powered by Stable Diffusion.

In the virtual try-on process, the user uploads an image or takes the photo from camera and uploads/selects a T-shirt through Streamlit UI, which forwards the person and cloth images to the AI component. The AI generates a try-on image,while the Recommendation System concurrently searches the dataset for matching accessories. Both results are returned to the Streamlit UI for display to the user.

For the T-shirt generation, the user provides a description through the Streamlit UI, which is sent to the Chatbot (Stable Diffusion). The chatbot generates a T-shirt image directly from the model and returns it to the Streamlit UI for presentation,The User can put their favorite T-shirts in Add to favourite, and it will be added to the DB

### 3.1.4 Flowchart Diagram




**Figure#.** Flowchart Diagram

**Flowchart Explanation: Virtual Try-On Process**

This flowchart outlines the user journey for a virtual clothing try-on application. Below is a detailed explanation of each step:

Start to Login

1. **Start**: The entry point of the application.
2. **Register/My/Next account?**:
   - New users can register for an account
   - Returning users can access their account ("My")
   - Option to check if another account exists ("Next account?")
1. **Login**: Users authenticate to access the system.

### Core Functionality

1. **Home page**: The main interface after successful login.
2. **Upload image (person & cloth)**:
   - Users can upload two images: one of themselves and one of the clothing item they want to try on.
1. **Excessive image uploads**:
   - System checks for upload limits and shows error messages if exceeded.
1. **Processing**:
   - Background removal ("remove bg")

- Pose estimation ("pacing open-pass")

1. **Feed image to virtual try-on model**:
   - Processed images are sent to the AI model (UZnet_TP5_ALIAS_GAMS) for virtual try-on simulation.

Alternative Options

1. **Open camera**: Option to use live camera instead of uploading images.
2. **Generate from text**: Alternative input method where users can describe clothing items textually.
3. **Describe The cloth**: Interface for entering clothing descriptions.

Output Handling

1. **Generate output**: System displays the virtual try-on result showing the user wearing the selected clothing.
2. **Add to favorites**: Option to save liked clothing items to a favorites list.
3. **Save or clear result**: Final options to either save the result or clear and start over.
4. **End**: Terminates the session or returns to home page.

This workflow supports multiple input methods (image upload, camera, text description) and provides options for saving preferred results, offering a comprehensive virtual try-on experience.

# 3.2 Details of the model and technical aspects

### 3.2.1 Model

Our proposed virtual try-on system leverages a combination of advanced deep-learning architectures tailored specifically for garment transfer onto target human models. The model integrates key innovations across **pose estimation**, **semantic segmentation**, **geometric warping**, and **image synthesis** stages, drawing inspiration from recent developments in virtual try-on technologies. The core modules of the system are carefully orchestrated to maintain human body integrity, garment texture, and spatial alignment, culminating in a realistic try-on experience.

**1.Backbone Network: ResNet-101 for Deep Feature Extraction**

At the foundation of the system lies a **ResNet-101** model, which serves as the primary **feature extraction backbone**. ResNet-101, a 101-layer deep residual network, has been selected due to its robustness in learning hierarchical feature

representations through **residual learning**. This architecture alleviates vanishing gradient issues by employing **skip connections**, allowing earlier layer activations a1 to be directly added to deeper layer outputs z2 before non-linear activation:

$$a2=g(z2+a1)$$

Such connections enable the network to retain low-level spatial features alongside high-level semantics, crucial for preserving facial structure, body pose, and background consistency during the try-on process.

**2.Segmentation and Parsing with U²-Net**

To effectively separate the human body from garments and background, the system employs **U²-Net**, a nested U-Net architecture originally designed for salient object detection. U²-Net provides **fine-grained human parsing**, producing **semantic masks** that segment body parts, including arms, torso, and legs.

This segmentation enables the creation of an **agnostic person representation** by excluding the original clothing, allowing the model to recompute the final composition based solely on the new garment. U²-Net's multiscale fusion structure ensures that even small details such as fingers or shoe contours are accurately parsed—an essential requirement for natural and artifact-free try-on synthesis.

**3.Garment Warping via Thin Plate Spline (TPS) Transformation**

To address shape deformation and alignment challenges, we integrate a **Thin Plate Spline (TPS)** module to warp the clothing image. TPS is a spatial transformation technique that uses a set of control points to deform a source image (garment) into a target configuration (body shape and pose).

This warping mechanism adapts the 2D flat image of the clothing to fit naturally onto the 3D body structure of the person image. It facilitates smooth deformations, preserving clothing details such as folds, textures, and symmetry, while matching the target pose geometry. This module is pivotal in aligning the sleeves, collar, and hem of garments to their respective body parts.

**4.ALIAS: Alignment-Aware Feature Normalization**

An additional innovation in the architecture is the incorporation of **ALIAS (Alignment-Aware Normalization)**, which refines feature fusion between the warped garment and the human representation. ALIAS adjusts normalization statistics based on spatial correspondence between inputs, improving the integration of warped garments into the person's silhouette.

Unlike conventional batch or instance normalization, ALIAS modulates feature maps using pose-conditioned alignment, which maintains edge sharpness and texture integrity. This is especially crucial in regions with sharp transitions, such as the neckline, shoulder, and sleeves, where conventional normalization may blur fine details.

**5.Image Synthesis with Warp-GAN Generator**

The final image is synthesized using **Warp-GAN**, a Generative Adversarial Network specifically designed for virtual try-on applications. Warp-GAN incorporates the outputs from previous modules—the segmented human image, warped clothing, and alignment-aware features—and synthesizes the final try-on image.

- **Generator**: Combines warped clothing with body features to generate the composite image.
- **Discriminator**: Evaluates the realism of generated outputs, guiding the generator to produce sharper, more plausible results.

The adversarial training process ensures that the synthesized try-on image maintains the **pose, identity, and background** of the person while replacing the original garment with the new one.

### 3.2.2 Preprocessing

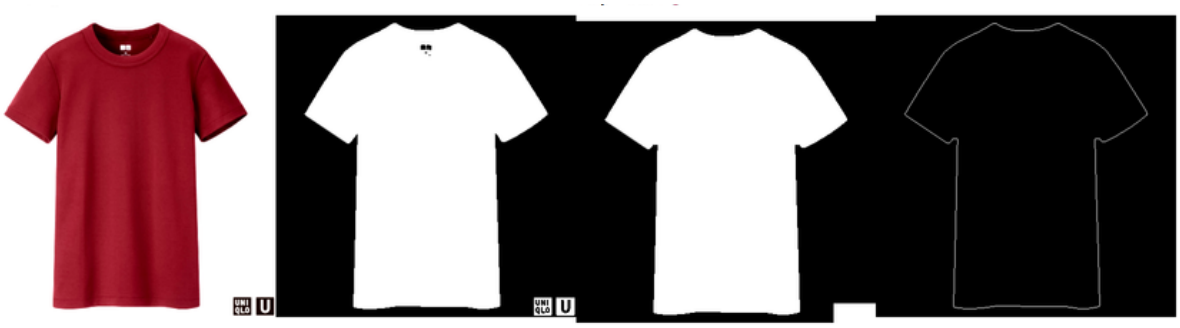
**Background removal mask:**

For the background removal process, we have adopted a streamlined and effective approach to enhance the clarity and focus of our dataset by eliminating visual noise from the background. Specifically, we utilize the Rembg module [source], a robust tool known for its high-performance background removal capabilities. Within our implementation, we make use of the remove function from the Rembg library inside the remove_bg.py script to process input images and generate clean foreground outputs.

A key component of our strategy is the use of **images with plain white backgrounds**, which improves the accuracy of background segmentation and results in cleaner separations between the subject and the background. This decision has proven essential in achieving higher fidelity in downstream tasks, such as garment segmentation and virtual try-on synthesis.

Additionally, we exploit the **optional "alpha matting" and "mask" capabilities** offered by Rembg to refine the extraction process. These options enable the generation of high-quality **binary masks**, which isolate the garment and person from the background with greater precision. The resulting output includes both the transparent-background image and the corresponding binary mask, which are crucial for guiding subsequent modules in our pipeline.

This preprocessing step ensures that only the relevant parts of each image, specifically the person and their clothing, are retained, thereby increasing the overall accuracy and efficiency of our virtual try-on system.



**Figure#.** Background removal-mask

**Parsing:**

Parsing, also referred to as semantic segmentation, is a fundamental technique used to segment the human body into multiple regions, each associated with specific semantic labels such as face, arms, clothing, and background. In our project, we used in our approach the Self-Correction for Human Parsing model, a high-performance architecture specifically tailored for fine-grained human segmentation tasks. This model introduces a self-correction mechanism that refines predictions by leveraging both supervised and unsupervised signals during training. For training and evaluation, the model makes use of benchmark datasets like the **LIP (Look Into Person)** and ATR datasets, each providing richly annotated human-centric labels. The LIP dataset offers **20 detailed semantic labels** such as 'hat', 'hair', 'gloves', 'coat', and 'pants', while the ATR dataset provides 18 similar fashion-oriented categories, enabling the model to parse complex clothing patterns and body regions accurately. These semantic labels are essential for isolating specific garment areas from the rest of the body, which is critical for downstream tasks such as virtual try-on, body modeling, and fashion recommendation systems. By employing this parsing technique, our system gains a structured understanding of the human form, thereby

enabling highly contextual and visually accurate processing of fashion-related visual data.


**Figure#.** Parsing

**Open pose:**

OpenPose is employed in our pipeline to provide accurate human pose estimation and detailed insights into key body joints. Its primary role is to ensure that the human body structure is consistently and faithfully preserved throughout the processing stages.

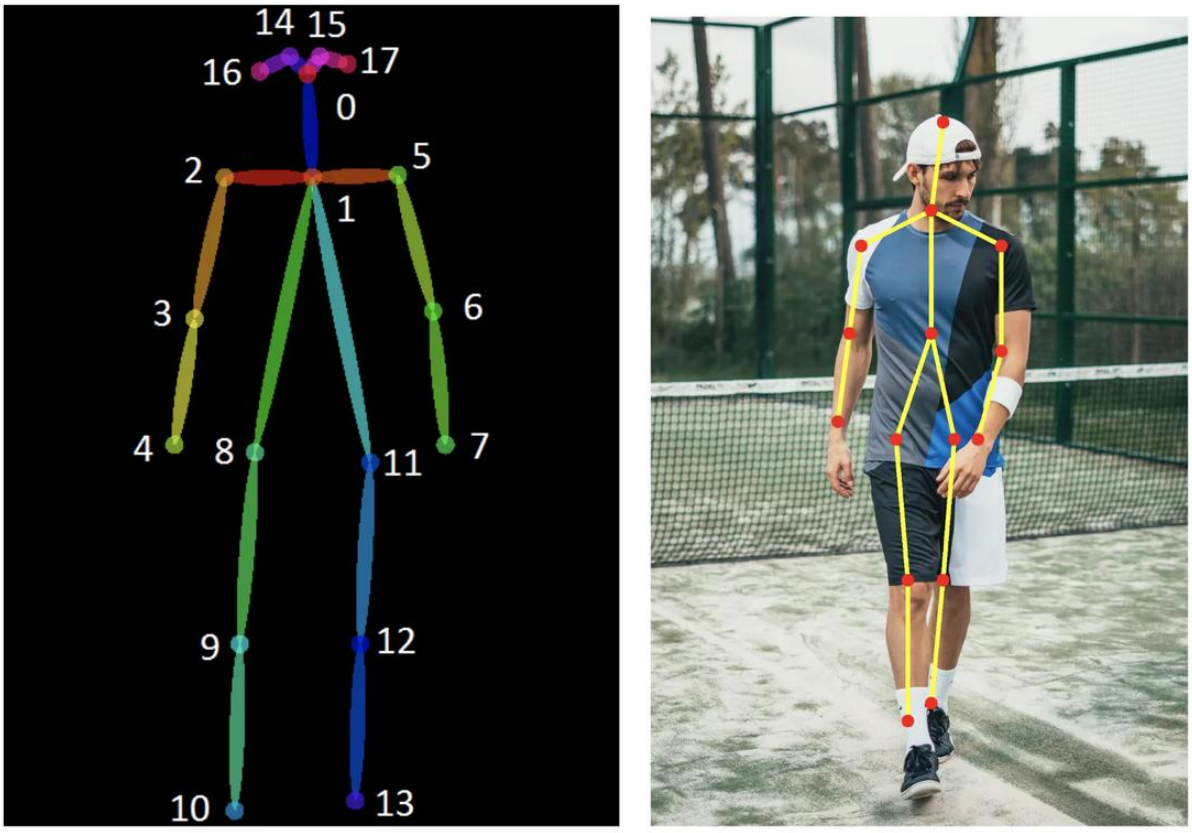
For seamless integration, we adopt the PyTorch implementation of the OpenPose model due to its compatibility with our system and its efficiency in extracting key pose-related information. Specifically, OpenPose outputs two critical components:

1. **Keypoints** – which represent the coordinates of various body joints, and
2. **JSON files**, which store these coordinates and associated confidence scores in a structured format.

The extracted keypoints are used to generate an **18-channel pose heatmap**, where each channel corresponds to a specific body part (such as nose, neck, elbows,

knees, etc.). These heatmaps serve as a comprehensive spatial representation of the human body.

This detailed pose information is crucial in preserving body alignment and form across subsequent stages of the system, making it a foundational component for tasks such as virtual try-on, pose-guided image generation, and animation.




**Figure#.** OpenPose

**Conclusion:**

Our preprocessing pipeline is meticulously designed to optimize data quality and ensure meaningful input for the virtual try-on model. Beginning with background removal using the **Rembg module,** we isolate the subject and garment to reduce noise and improve focus. Parsing, powered by the **Self-Correction Human Parsing model,** segments the human body into fine-grained semantic regions, enabling garment-aware understanding. **OpenPose** further enriches this by estimating body keypoints, capturing essential structural pose data. These are complemented by DensePose, which maps image pixels to 3D human surface coordinates, enhancing alignment between garments and body surfaces.

Together, these modules form a cohesive preprocessing framework that extracts precise visual, spatial, and semantic features. This foundational data enables the

virtual try-on system to generate realistic and well-aligned outputs, preserving garment detail and body conformance. The synergy of these advanced techniques ensures robustness, accuracy, and visual coherence throughout the try-on pipeline, ultimately elevating the user experience and performance of the model.

### 3.2.3 Generate Your Cloth Feature

This section presents a detailed overview of the **"Generate Your Own Cloth"** functionality, which leverages the **DreamLike model**, a fine-tuned variant of **Stable Diffusion**, to generate high-quality and semantically accurate clothing designs from user-provided prompts. This feature combines advanced **text-to-image synthesis**, **latent space conditioning**, and **semantic guidance** to deliver compelling, personalized garment imagery.

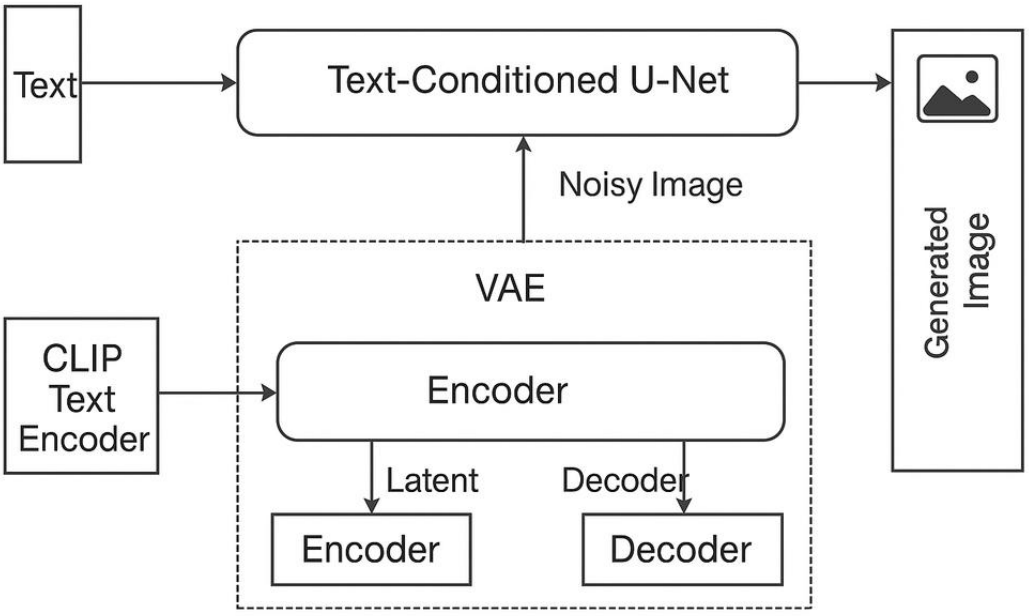
At the core of this module lies a powerful denoising diffusion architecture. The pipeline begins on the left with an initial **noisy latent image representation**, denoted by $x_0$. This noise-injected input undergoes a multi-step **denoising process**, wherein a series of learned transformations gradually convert $x_0$ into a coherent image representation x. This operation is governed by a **sigma-controlled noise schedule**, which dynamically guides the strength of noise reduction over multiple iterations, improving clarity and feature alignment.

Following denoising, the encoded representation is passed into the **latent space encoder E**, which compresses the image into a more compact and information-rich **latent vector z**. This latent embedding captures structural and semantic attributes necessary for high-quality cloth synthesis. Within the latent space module, the **U-Net-based architecture**—central to the DreamLike model—applies a series of **cross-attention mechanisms**, utilizing **Query (Q)**, **Key (K)**, and **Value (V)** interactions. These mechanisms enable the model to integrate contextual information from textual prompts and auxiliary inputs (e.g., reference cloth or style attributes), ensuring that generated outputs are not only realistic but also aligned with user intent.

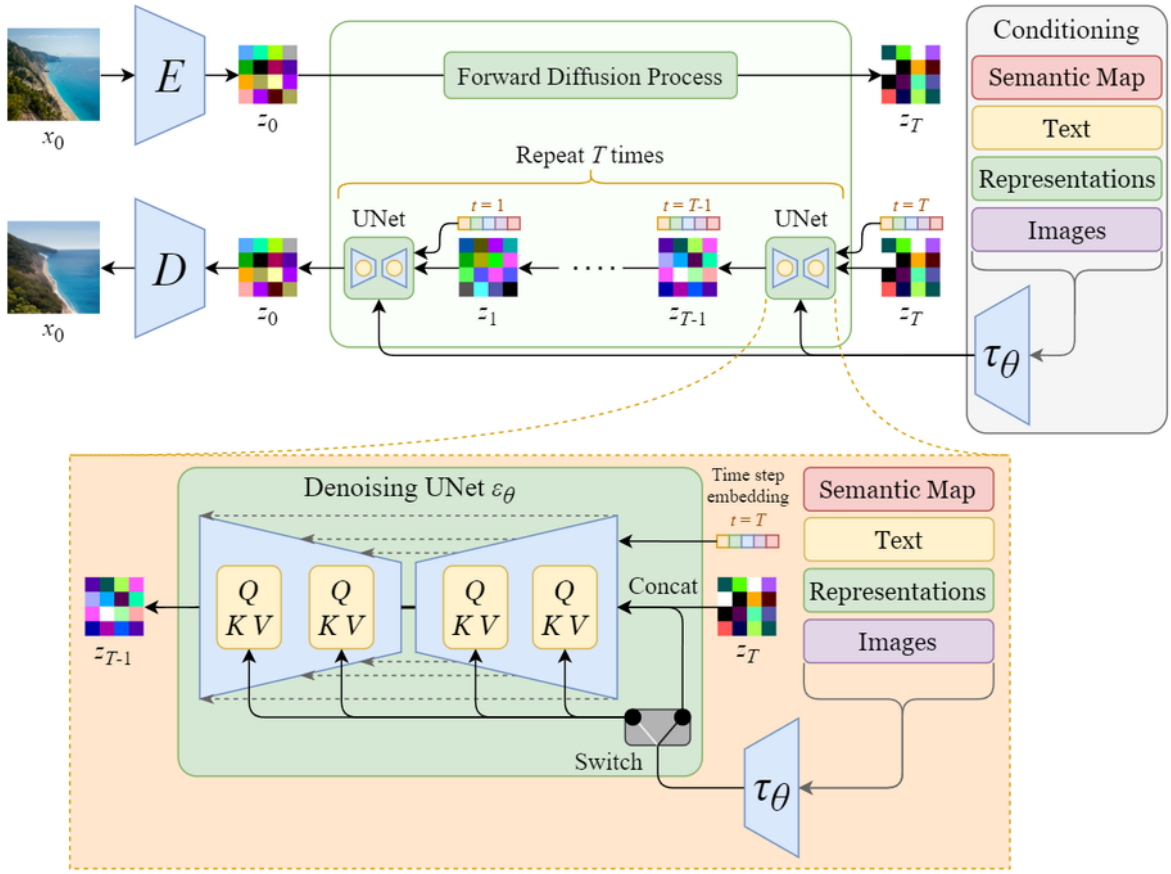
Simultaneously, a conditioning branch feeds in the **prompt embeddings**, encoded using a powerful transformer-based **text encoder**. These conditioning vectors, ΣT, represent the semantic context of user inputs and serve as guiding signals during the generation process. The cross-attention layers in the latent space align z with ΣT, producing a **conditioned latent vector zT**. This interaction allows for fine control over garment attributes such as texture, color, style, and structure.

The conditioned latent vector is then decoded via a **decoder module D**, which reconstructs the final output image xT—a detailed and high-resolution visualization of the generated garment. The decoder integrates skip connections from the U-Net's encoding layers, preserving fine-grained spatial details throughout reconstruction.

This integrated pipeline—enabled by the DreamLike model's fine-tuned diffusion capabilities—supports the **"Generate Your Own Cloth"** feature by allowing users to transform text prompts into visually coherent, fashion-relevant images. The result is a customized clothing design generation experience, where users can explore diverse apparel aesthetics tailored to their preferences, underpinned by the power and flexibility of a cutting-edge generative model.




**Figure#.** Dream-like model




**Figure#.** stable-diffusion model

The depicted model is a sophisticated architecture derived from **Stable Diffusion** and enhanced through fine-tuning into the **DreamLike model**, specifically adapted for high-fidelity clothing generation based on text prompts. It is designed to generate realistic and fashion-relevant garments by combining **denoising**, **latent-space modeling**, and **semantic conditioning**.

On the left side of the flowchart, the process begins with the **initial noisy latent input**, denoted as $x_0$, which represents a random noise vector. The first stage involves **progressive denoising**, where the model refines this noise through multiple steps to approach a visually coherent image. Each step of denoising, guided by a **learned noise prediction function** and regulated by a **noise scheduler (σ)**, transforms $x_0$ into a more defined latent representation x, reducing randomness and enhancing semantic coherence.

After denoising, the denoised latent representation is passed through an **encoder E**, which transforms the image into a latent embedding z. This **latent space** efficiently captures crucial features and structure while enabling lower-dimensional manipulation. Within this latent space lies a **Denoising U-Net**, the architectural core

of the DreamLike model, where the latent representation is processed using **cross-attention blocks** composed of **Query (Q)**, **Key (K)**, and **Value (V)** components. These attention layers allow the model to align internal visual features with external textual conditions through mechanisms such as **skip connections**, **concatenations**, and **switching operations**—all working to iteratively refine and guide the latent vector z.

On the right side of the architecture, a separate conditioning pathway introduces **semantic conditioning text embeddings ΣT**, derived from the user's prompt using a **text encoder** (typically a transformer-based module). This semantic information is injected into the latent U-Net through **cross-attention**, producing a conditioned latent representation zT that merges both visual priors and textual intent.

Finally, this conditioned representation zT is passed into a **decoder D**, which reconstructs the final high-resolution image xT. The decoder integrates information from earlier layers (via U-Net skip connections) to preserve details and improve realism. The output image xT is a **semantically guided, denoised rendering** of the user's prompt—a high-quality visualization of a personalized clothing design.

This refined architecture underpins the **"Generate Your Own Cloth"** feature, enabling the transformation of user prompts into rich, custom garment images. By leveraging **latent space attention**, **semantic conditioning**, and **DreamLike's fine-tuned diffusion weights**, this feature delivers an intuitive and visually compelling design experience aligned closely with individual user preferences.

## 3.3 Recommendation System Based on ResNet-18

To enhance personalization and user engagement in our virtual try-on system, we integrated a **recommendation module** powered by a **ResNet-18-based deep learning model**. This module is designed to analyze garment images and recommend visually or semantically similar clothing items to users based on their selected preferences.

**ResNet-18 Architecture:**

ResNet-18 is a convolutional neural network that employs **residual learning**, allowing the model to train deeper architectures effectively by mitigating the vanishing gradient problem. It consists of 18 layers, including **four residual blocks**, each containing **shortcut connections** that enable the direct propagation of feature maps. These residual connections facilitate the learning of identity mappings,

allowing the network to focus on learning fine-grained differences between garment styles, colors, and textures.

**Feature Extraction:**

In our system, ResNet-18 is **pretrained on ImageNet** and fine-tuned using our curated fashion dataset. The model processes input images (e.g., clothing items selected or worn by users) and transforms them into **high-dimensional feature vectors** by removing the final classification layer and retaining the penultimate fully connected layer as a **feature extractor**. These embeddings encapsulate key visual characteristics of the garments, such as:

- Color distribution
- Texture patterns
- Silhouette and shape
- Localized details (e.g., buttons, prints, stitching)

**Similarity Matching:**

Once feature vectors are obtained from ResNet-18, the recommendation system computes **pairwise similarities** using a distance metric such as **cosine similarity** or **Euclidean distance**. Given a query image selected by the user, the system retrieves the **top-N most similar garments** from the database by identifying feature vectors with the smallest distance to the query.

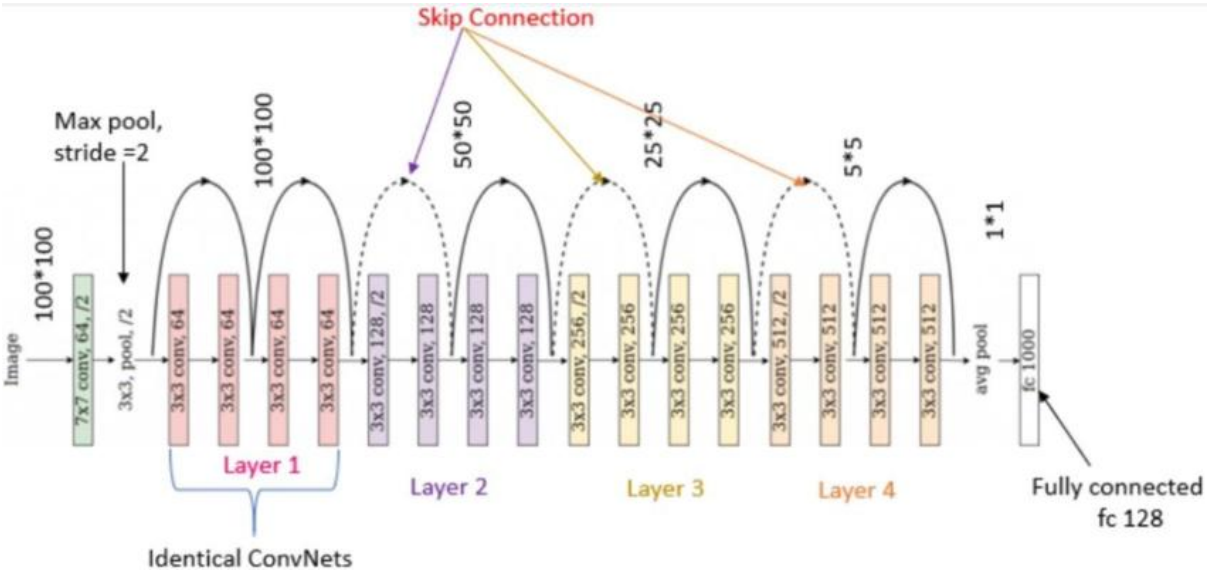
**Recommendation Pipeline:**

The pipeline consists of the following steps:

1. **Input**: A clothing image selected by the user (e.g., a generated cloth or a worn garment).
2. **Preprocessing**: Resize and normalize the image to match ResNet-18 input requirements (224×224, RGB).
3. **Embedding Generation**: Feed the image through the modified ResNet-18 to extract its feature vector.
4. **Database Embedding Lookup**: Compare the query embedding to a precomputed set of embeddings from the garment database.
5. **Top-K Retrieval**: Return the top-K garments with the highest similarity scores as recommendations.

**Benefits and Integration:**

This recommendation module not only offers **personalized outfit suggestions** but also complements the **"Generate Your Own Clothes"** feature by presenting related styles that users might prefer. By leveraging **deep visual understanding** through ResNet-18, the system adapts to user preferences in a **data-driven** and **aesthetically coherent** manner.




**Figure#.** ResNet-18

Voice-Based Garment Generation and Virtual Try-On Integration

**Introduction:**As part of enhancing the user experience in virtual try-on systems, we implemented a speech-enabled garment recommendation and generation module. This feature allows users to describe garments using natural language through speech. The system interprets the speech input, converts it to text, and then generates a visual representation of the described garment using a text-to-image model. This image can be passed through the try-on pipeline for visualization on a human figure.

System Architecture and Workflow
The implementation consists of three main stages:

1. **Speech-to-Text Conversion**
   The user's speech input is captured through a microphone and processed using automatic speech recognition (ASR). We use the SpeechRecognition library, which interfaces with the Google Web Speech API to transcribe spoken input into text. This forms the prompt for the next stage.
2. **Text-to-Image Generation**
   The textual description generated in the first stage is used as a prompt for a

generative model, specifically **Stable Diffusion**. This model produces a high-resolution image of the garment that closely reflects the input description. The resulting image represents the hypothetical clothing item.

3. **Integration with the Virtual Try-On Pipeline**

   The image generated by the diffusion model is preprocessed and passed into the virtual try-on pipeline. This typically includes:

   - Background removal (e.g., using U2-Net or Rembg),
   - Garment warping (via models like GMM or TPS),
   - Try-on synthesis using modules such as ALIASGenerator.

1. The generated clothing is aligned with the target body pose and segmentation map to create the final try-on output.

Example Workflow

- **Input Voice:** “A red velvet evening gown with long sleeves”
- **Speech-to-Text Output:** "A red velvet evening gown with long sleeves"
- **Text-to-Image Output:** Realistic rendered image of a red velvet gown
- **Try-On Result:** Synthesized image showing the user wearing the described gown

Technical Stack

- **Speech Recognition:** SpeechRecognition, pyaudio
- **Text-to-Image Generation:** Stable Diffusion via Hugging Face diffusers or InvokeAI
- **Preprocessing:** U2-Net for segmentation and background removal
- **Try-On Models:** TPS, GMM, ALIAS for warping and synthesis

Use Case Benefits

- Eliminates the need for typing, making the system more accessible.
- Allows personalized garment generation without manual image input.
- Enables creative garment recommendations and supports non-standard descriptions.

# Chapter 4

# Implementation

## 4. Implementation, Experimental Setup, & Results

## 4.1 Implementation Details

Our implementation focuses on a custom virtual try-on pipeline comprising several deep learning modules, each responsible for a distinct phase of the try-on process. Without requiring any additional training, we utilize pretrained components across all modules, ensuring efficient evaluation without the need for end-to-end retraining. The system is designed to generate high-resolution try-on outputs at a resolution of 512×384 pixels, preserving garment detail and body integrity.

At the core of our feature extraction stage lies the ResNet-101 architecture, a 101-layer deep residual network pretrained on the ImageNet dataset. This backbone is responsible for learning hierarchical visual representations from both the person and garment images. All layers of ResNet-101 remain frozen during testing to retain their generalization capabilities and prevent overfitting.

For **human parsing and semantic segmentation,** the system integrates a **pretrained U²-Net model**, which provides fine-grained parsing of human body parts. This enables the construction of an agnostic person representation by excluding original garments while retaining body structure, essential for accurate try-on alignment.

**Garment warping** is achieved through the **Thin Plate Spline (TPS)** transformation module. It uses control points derived from pose estimation results to deform the garment image, aligning it with the target human figure. The TPS component operates in a spatially adaptive manner, ensuring that the warped garment adheres naturally to the body shape and posture of the target model.

To enhance feature integration between warped garments and the human representation, we incorporate the **ALIAS (Alignment-Aware Normalization)**

module. This alignment-sensitive normalization layer adjusts feature maps based on spatial correspondence derived from pose estimation, refining edge sharpness and texture blending during synthesis.

The final try-on image is synthesized using a **pretrained Warp-GAN** generator. This generative module takes the warped garment, parsed body representation, and aligned features as inputs and produces a photorealistic composite image. The generator operates under the supervision of an adversarial discriminator, which guides it to produce visually plausible and identity-preserving outputs.

In addition, we apply a mask-aware fusion mechanism that selectively enhances high-frequency details in garment-sensitive regions such as sleeves, collars, and hands. This fusion process is conditioned on semantic and garment masks, ensuring precise reconstruction of fine-grained textures and contours.

Enhancing the proposed model, we customized a preprocessing pipeline tailored specifically to the Viton-HD dataset, which focuses on upper-body garments in high-resolution settings. The dataset undergoes a standardized preprocessing procedure to ensure compatibility with downstream modules in the virtual try-on pipeline.

Initially, both the garment image and the person image are processed through the **Rembg** module [6] to remove the background, isolating the foreground regions critical for accurate alignment and synthesis. The output of this stage is then forwarded sequentially through three pretrained models: a **human parsing network**, **OpenPose** for keypoint detection, and **DensePose** for fine-grained pose estimation.

For human parsing, we employ a model pretrained on the **LIP dataset**, which provides 20 semantic body part labels. These parsed maps are used to create the agnostic person representation by masking out regions such as upper clothes, arms, and hair, thereby retaining only structural components needed for garment overlay. In the DensePose model, we utilize the dp_segm parameter to generate high-resolution segmentation masks for the annotated human figures. This enhances pose accuracy and supports better alignment in the garment warping phase.

Despite relying solely on the Viton-HD dataset, all subsequent stages of the pipeline—warping, feature alignment, and image synthesis—remain consistent and agnostic to dataset-specific differences. The processed garment and person features are subsequently passed into the pretrained modules, ultimately producing a realistic output of the person wearing the newly selected garment.

To complement the virtual try-on system, we integrate a **fashion recommendation module** based on both garment texture and color features. Additionally, a **complementary item recommendation** system is developed, focusing on personalized suggestions based on inferred user preferences (e.g., gender). Both systems utilize deep learning-based feature extraction and similarity matching techniques.

At the core of this recommendation engine is a **ResNet-1** model pretrained on the ImageNet dataset [20], which serves as the visual encoder. The top classification layer is excluded, allowing the model to function purely as a feature extractor. Each garment and accessory image in our curated recommendation dataset is passed through this network to obtain a compact, high-dimensional feature embedding. These embeddings encode semantic and visual characteristics such as pattern, color, and shape.

To maintain uniformity across feature representations, the resulting vectors are normalized to unit length. A **Nearest Neighbors** algorithm is then employed to perform similarity search based on Euclidean distance, identifying garments that are visually closest to the input image. The number of neighbors and distance metric are configurable, allowing fine-tuning of recommendation precision. This framework enables the system to deliver visually relevant and personalized garment and accessory suggestions to users.

## 4.1.1 Data Processing for Virtual Try-On System

The virtual try-on system relies on a robust data processing pipeline to prepare input images for effective virtual garment fitting. This section outlines the methodology for processing input data, including background removal, cloth segmentation, and dataset preparation. The pipeline ensures that both the person's image and the garment are appropriately preprocessed to facilitate accurate virtual try-on rendering.

**1.1 Background Removal:**

The initial step in the data processing pipeline involves removing the background from input images of individuals to isolate the person for virtual try-on. This is implemented using the preprcessInput class, which leverages the rembg library to perform background removal. The process begins by loading an input image (in JPEG format) using the PIL library. The image is converted to a NumPy array to

extract its dimensions (width, height, and channels). The remove function from rembg is then applied to eliminate the background, producing an image with a transparent background saved in PNG format. The original JPEG file is removed to optimize storage. Following background removal, the image is resized to a standardized resolution of **768×1024** pixels to ensure compatibility with downstream processes. A white background is added to the resized image by creating a new RGBA image and pasting the transparent image onto it, using the alpha channel as a mask. The final output is converted to RGB format and saved as a JPEG file. This step ensures that the person's image is consistently formatted for further processing, with the background removed to focus on the subject.

### 1.2 Cloth Segmentation:

To isolate the garment from its background, a cloth segmentation process is employed using the **U2-Net** model, a deep learning architecture designed for salient object detection. The implementation is encapsulated in the cloth_mask script. The U2-Net model, pretrained on cloth segmentation tasks, is loaded with weights from a checkpoint file (cloth_segm_u2net_latest.pth) and executed on a CUDA enabled GPU for efficiency. Input garment images (in RGB format) are resized to a uniform 768×768 resolution using bicubic interpolation to preserve quality. The images are then normalized using a mean and standard deviation of 0.5, transforming pixel values to the range [−1,1]. The normalized images are fed into the U2-Net model, which outputs a segmentation map with four classes. A log-softmax operation is applied to the output, and the class with the highest probability is selected to generate a binary mask. The mask is converted to a grayscale image, resized back to the original dimensions, and saved as a JPEG file in the cloth-mask directory. A color palette is applied to ensure consistent visualization of the segmented garment. This segmentation step is critical for isolating the garment from its background, enabling precise overlay onto the person's image during the virtual try-on process.

### 1.3 Dataset Preparation:

The VITON Dataset class, built upon PyTorch's data.Dataset, orchestrates the preparation of a compre hensive dataset for training and testing the virtual try-on system. The dataset includes person images, garment images, cloth masks, pose data, and semantic parsing maps, all aligned to ensure compatibility with the model's requirements.

#### 1.3.1 Data Loading and Preprocessing:

The dataset is initialized with a configuration object specifying parameters such as image resolution (e.g., load_width and load_height) and the number of semantic classes (semantic_nc). A list of image pairs (person and garment) is loaded from a text file, specifying filenames for person images and corresponding garments. The images are resized to a consistent width using bilinear interpolation for RGB images and nearest-neighbor interpolation for masks to preserve discrete labels.

### 1.3.2 Pose and Semantic Parsing:

Pose information is extracted from OpenPose-generated JSON files containing 2D keypoints of the per son's body. These keypoints are used to create an agnostic representation of the person's image and parsing map. The get_img_agnostic method masks out specific body parts (e.g., arms, torso, neck) based on pose keypoints, preserving the head and lower body while filling masked regions with a gray color. Similarly, the get_parse_agnostic method generates a semantic parsing map by masking upper body regions (e.g., upper clothing, neck) and arms, using pose data to guide the masking process. The parsing map is encoded as a one-hot tensor with 20 channels, later reduced to the specified number of semantic classes.

### 1.3.3 Cloth and Mask Processing:

Garment images and their corresponding masks (generated from the cloth segmentation step) are loaded and resized to match the person's image dimensions. The garment images are normalized to [−1,1], and the masks are binarized (values ≥ 128 mapped to 1, otherwise 0) to create a single-channel tensor. These processed inputs are critical for training the virtual try-on model to align garments with the person's body.

### 1.3.4 Data Loader:

The VITONDataLoader class wraps the dataset in a PyTorch DataLoader, enabling batch processing with configurable batch size, shuffling, and parallel data loading. This ensures efficient data retrieval during training, with options to shuffle the dataset for randomized sampling or maintain a fixed order for testing.

## 1.4 Summary

The data processing pipeline is a cornerstone of the virtual try-on system, enabling the transformation of raw images into structured inputs suitable for model training

and inference. Background removal isolates the person, cloth segmentation extracts the garment, and dataset preparation integrates pose and semantic information to create a cohesive dataset. These steps collectively ensure that the system can accurately render garments onto a person's image, accounting for body pose and shape.

## Human Parsing Using Pretrained Models

Human parsing plays a critical role in the proposed object detection system, particularly in cases involving clothing and body-related object detection. It refers to the process of segmenting a human figure in an image into semantically meaningful regions such as body parts (e.g., arms, legs) and clothing items (e.g., dress, pants, coat). This segmentation provides a foundation for subsequent tasks, including fine-grained object identification, virtual try-on systems, and mask-based manipulation of images.

To implement this stage, the project integrates the **Self-Correction for Human Parsing (SCHP) framework.** This framework is known for its high performance in semantic segmentation and is based on a ResNet-101 backbone architecture. **SCHP** utilizes self-correction mechanisms during training and testing phases to refine segmentation masks and improve parsing accuracy.

**AugmentCE2P Module — Enhancing Parsing Robustness**

- Implements **image-aware data augmentation** (e.g., solarize, equalize, posterize) to simulate diverse real-world conditions
- Integrates **model-based augmentation** using Res2Net blocks to improve edge and scale-sensitive features
- During training, it wraps around the CE2P parser to **enhance accuracy and edge fidelity**—critical for realistic virtual try-on segmentation
- Though not used directly at inference time, its impact underlies the pretrained SCHP model's resilience to real images

The SCHP model is available with several pretrained variants, each trained on a specific dataset designed for human parsing. These datasets include **LIP** (Look Into Person), **ATR**, and **PASCAL-Person-Part**. Each dataset varies in terms of annotation granularity and the semantic classes it provides. For instance, PASCAL-

Person-Part focuses primarily on coarse body parts such as torso, arms, and legs, offering a limited number of classes. ATR extends this by including additional facial and clothing attributes, but remains constrained in terms of full-body parsing. In contrast, the LIP dataset provides the most comprehensive set of annotations, including both body parts and a wide range of clothing categories, making it particularly suitable for applications requiring detailed human understanding.

## Justification for Using the LIP Dataset

Among the supported datasets, the LIP (Look Into Person) dataset was selected for integration into this project due to several compelling reasons. First and foremost, the LIP dataset offers a rich and detailed annotation schema comprising twenty distinct semantic labels. These labels cover not only key body regions such as the face, arms, and legs but also a diverse set of clothing items, including hat, coat, dress, upper-clothes, pants, and shoes. This level of detail is essential for object-level analysis within human figures, especially in applications like cloth classification and virtual try-on.

Furthermore, the LIP dataset is large-scale and composed of over fifty thousand real-world images featuring humans in various poses, lighting conditions, and degrees of occlusion. This diversity ensures that the trained model generalizes well to unseen images, providing robust parsing results across a variety of scenarios.

Another critical factor in choosing the LIP dataset is its compatibility with virtual try-on and clothing manipulation applications. Since the dataset includes fine-grained clothing categories, it becomes feasible to isolate or remove specific garments from the image using their respective semantic labels. For instance, to create an agnostic person mask, labels corresponding to upper-clothes, dress, or coat can be selectively excluded. This enables the creation of clean inputs for downstream components that require images with certain clothing items removed or replaced.

Lastly, the availability of a high-quality pretrained SCHP model trained on the LIP dataset significantly reduces the complexity and time required to build and train a new model from scratch. The pretrained model uses a ResNet-101 backbone and delivers reliable segmentation results, making it ideal for integration into the pipeline.

## 4.1.2 Semantic Labels in LIP

The LIP dataset includes the following semantic classes: background, hat, hair, glove, sunglasses, upper-clothes, dress, coat, socks, pants, jumpsuits, scarf, skirt,

face, left-arm, right-arm, left-leg, right-leg, left-shoe, and right-shoe. Each of these categories is represented as a unique class in the segmentation output. The segmentation model assigns one of these class labels to every pixel in the input image, effectively dividing the human figure into meaningful parts.

These segmentation maps are then utilized in subsequent stages of the system, where specific classes can be retained, removed, or highlighted based on the application requirements. For example, when generating an agnostic human image for virtual try-on, clothing classes such as coats or dresses can be excluded from the image while preserving body parts and facial features.

## 4.2 Implementation and Inference Process

The implementation of the SCHP-based human parsing step relies on the official repository provided by the original authors. The simple_extractor.py script is used to perform inference on a set of input images. During execution, the input images are first resized and normalized to match the expected model input size. Then, the pretrained SCHP model predicts the segmentation mask by performing pixel-level classification. The output is subsequently resized back to the original image dimensions to maintain spatial consistency.

The final segmentation results are saved in PNG format, where each pixel's color corresponds to a specific class label. Additionally, the raw logits can be saved in .npy format for further processing if needed.

These segmentation masks form an essential component of the proposed object detection pipeline and serve as input for further operations such as segmentation refinement, mask overlay, object classification, or visual presentation in the graphical user interface.

### 4.2.1 U-Net Architecture

U-Net is a kind of neural network mainly used for **image segmentation**, which means dividing an image into different parts to identify specific objects. The name "U-Net" comes from the shape of its architecture, which looks like the letter "U" when drawn. It is widely used in medical imaging because it performs well even with a small amount of labeled data..

U-Net has proven to be highly effective in other image-based tasks, including **Virtual Try-On systems**. In the context of our project, U-Net is used for **human parsing**,

specifically segmenting a person's body and clothing items from an image, which is crucial for accurately placing garments during virtual try-on.

**U-Net Architecture Components:**

1. **Contracting Path (Encoder)**

- Applies 3×3 convolutions to extract features from the input image.
- Uses **ReLU** activation functions to introduce non-linearity and allow the model to learn complex patterns.
- Applies **2×2 max pooling** to reduce the spatial dimensions, helping the network focus on larger features such as body parts and clothing boundaries.
- The output of each level is passed forward to the decoder through **skip connections**.

2. **Bottleneck**

- The deepest part of the network is where the feature maps are most compressed.
- This layer captures high-level semantic features necessary for accurate segmentation of regions like the shirt, pants, arms, and background.

3. **Expanding Path (Decoder)**

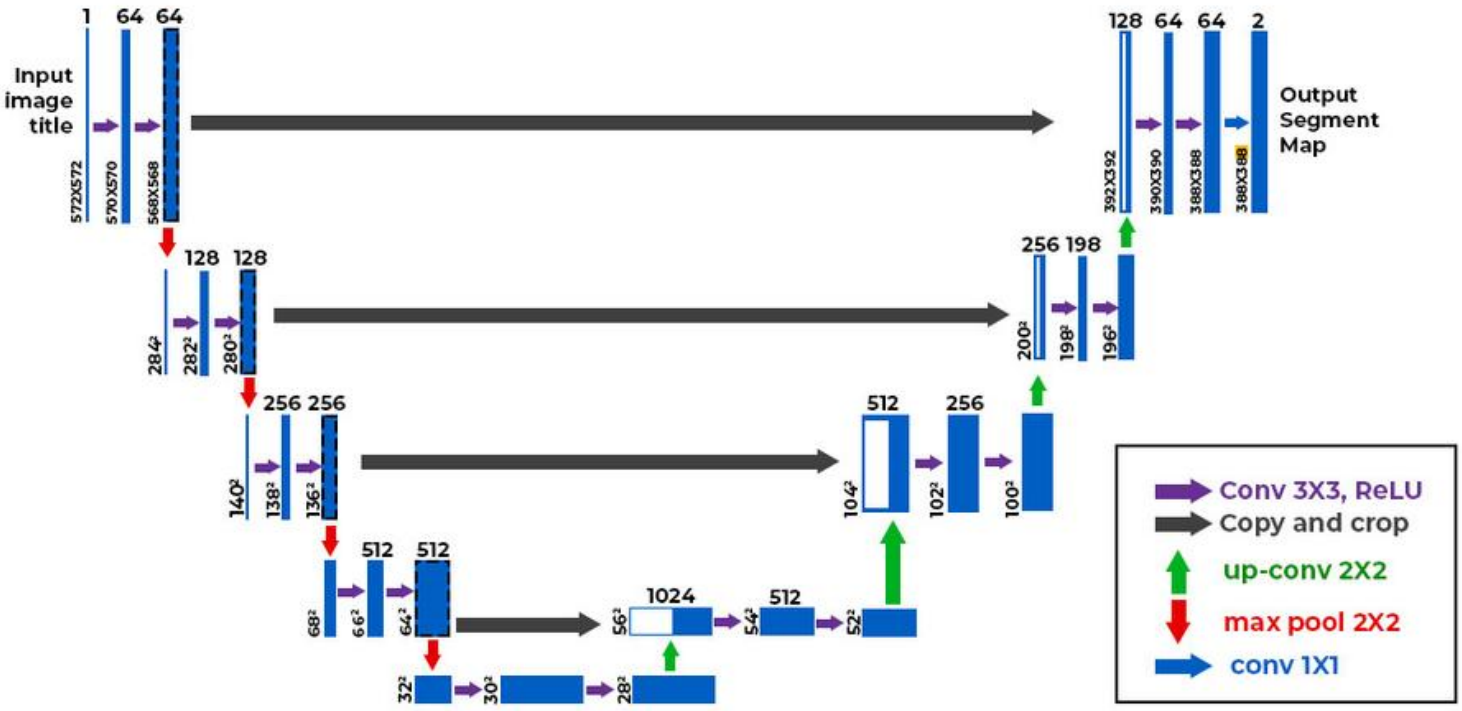
- Performs **upsampling** (increasing the resolution) using transposed convolutions.
- **Skip connections** from the encoder are concatenated here, allowing the decoder to use detailed spatial features that were lost during downsampling.
- More convolutional layers refine the output and correct object boundaries.

**U-Net in Virtual Try-On:**

In this project, U-Net is used to **segment human body parts and clothes** from input images. This segmentation allows us to:

- Identify **regions where new clothes can be overlaid** (e.g., shirt, pants).
- Remove or mask existing clothing without affecting body parts like arms or faces.
- Provide a **clean and accurate input** for the virtual try-on module to place garments naturally and realistically.

This process is often referred to as **"Human Parsing"**, and U-Net is the backbone that performs this task.




**Figure#.**

**How U-Net Works in Our System:**

1. **Input Image**

- The input is a photo of a person, usually taken from the front view.

2. **Feature Extraction (Encoder)**

- The encoder extracts patterns such as edges, textures, and shapes to detect different body parts and clothing items.

3. **Compression (Bottleneck)**

- The network encodes this information into a compact feature representation, isolating the essential visual features of the image.

4. **Reconstruction (Decoder)**

- The decoder reconstructs the segmented image by upsampling and re-integrating details from the encoder using skip connections.

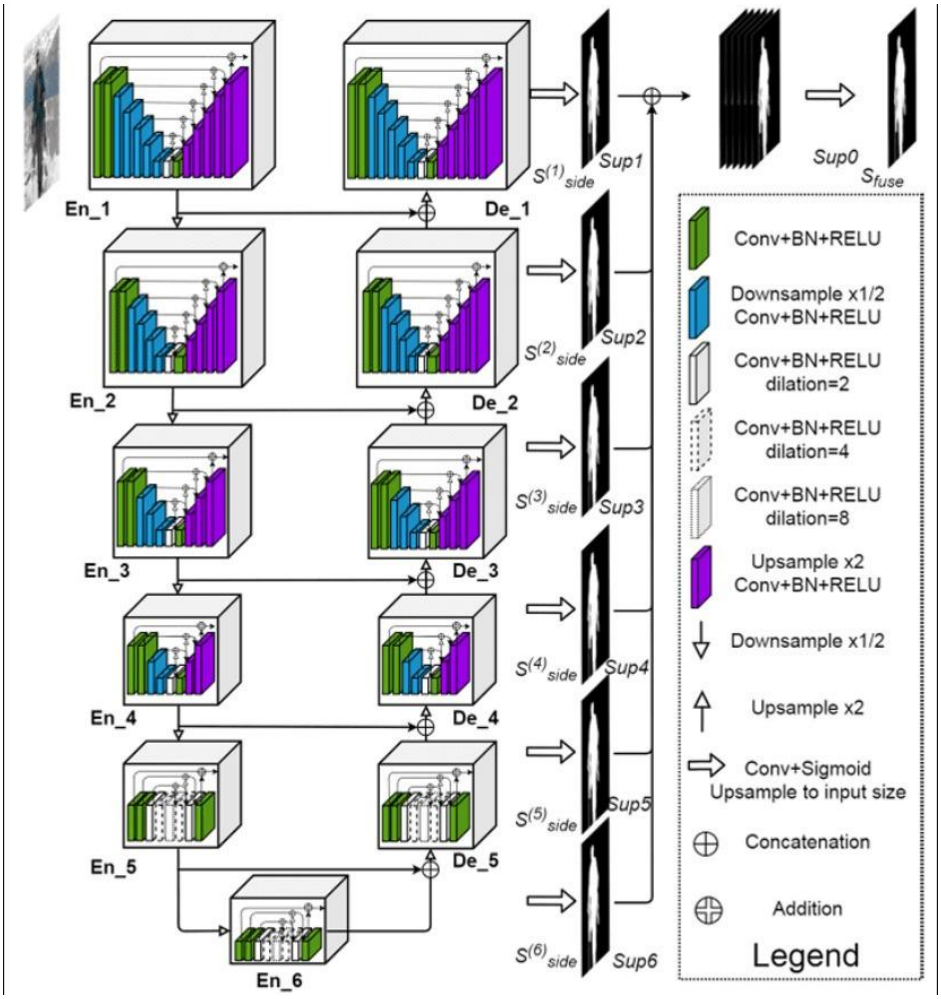
5. **Output Segmentation Map**

- The final output is a pixel-wise mask that classifies each pixel as part of a clothing item, body, or background.

- This mask is used to remove the existing clothing or to locate where the new garment should be overlaid.

## 4.2.2 U2-Net Architecture

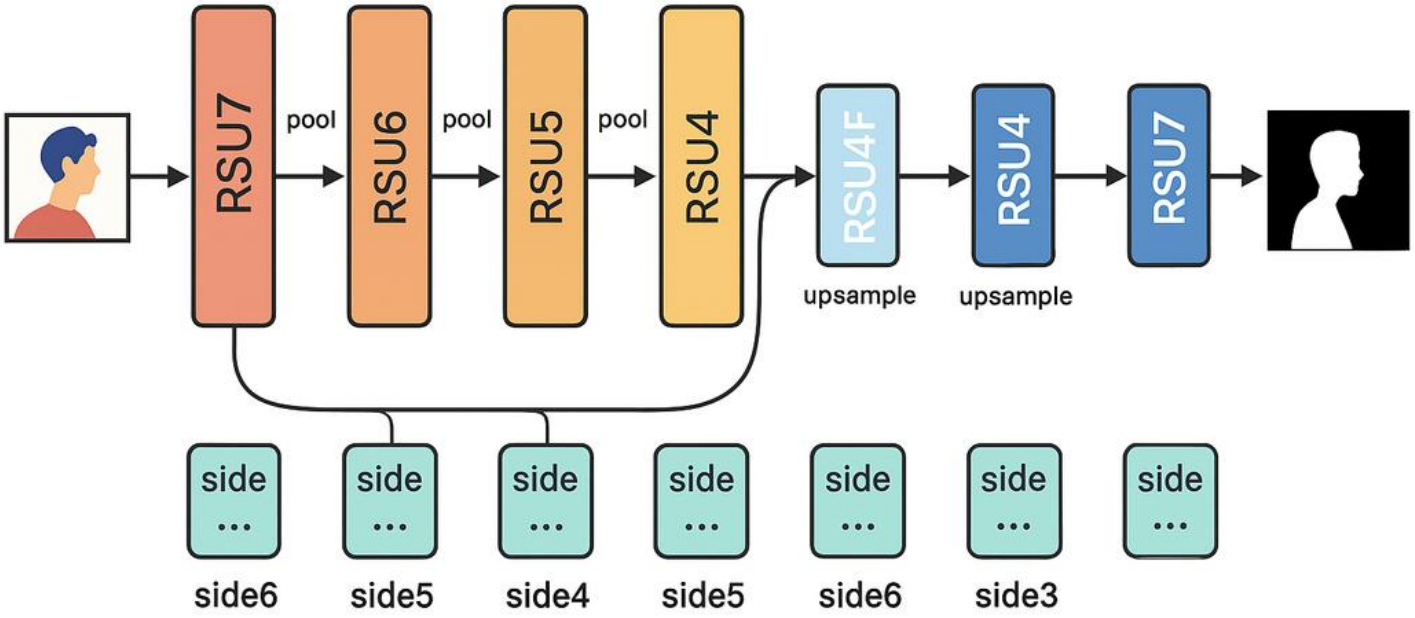
The U2-Net architecture can handle **multi-level deep feature extraction** and **multi-scale information** across local and global contexts. This two-level nested modified U-Net-like structure enables training without significant memory consumption and computation costs. The core of each level in the architecture is built upon the Residual U-block (RSU), which incorporates the properties of a residual block and a .U-Net-like symmetric encoder-decoder structure




**Figure#.**

U^2-Net is typically used to generate precise **segmentation masks for clothes** or **the human body**, required for compositing garments onto model images, swapping clothes, or removing the background.

Feed in an image, get a **mask (output)**, which is then used to extract or manipulate garments/person in later steps of the pipeline.




**Figure#.**

## 4.2.3 Geometric Matching Module (GMM) in Virtual Try-On

The **Geometric Matching Module (GMM)** is responsible for aligning the 2D clothing image with the target person's pose and body structure before final synthesis. It utilizes a Thin Plate Spline (TPS) transformation, which allows flexible and smooth warping of the clothing image.

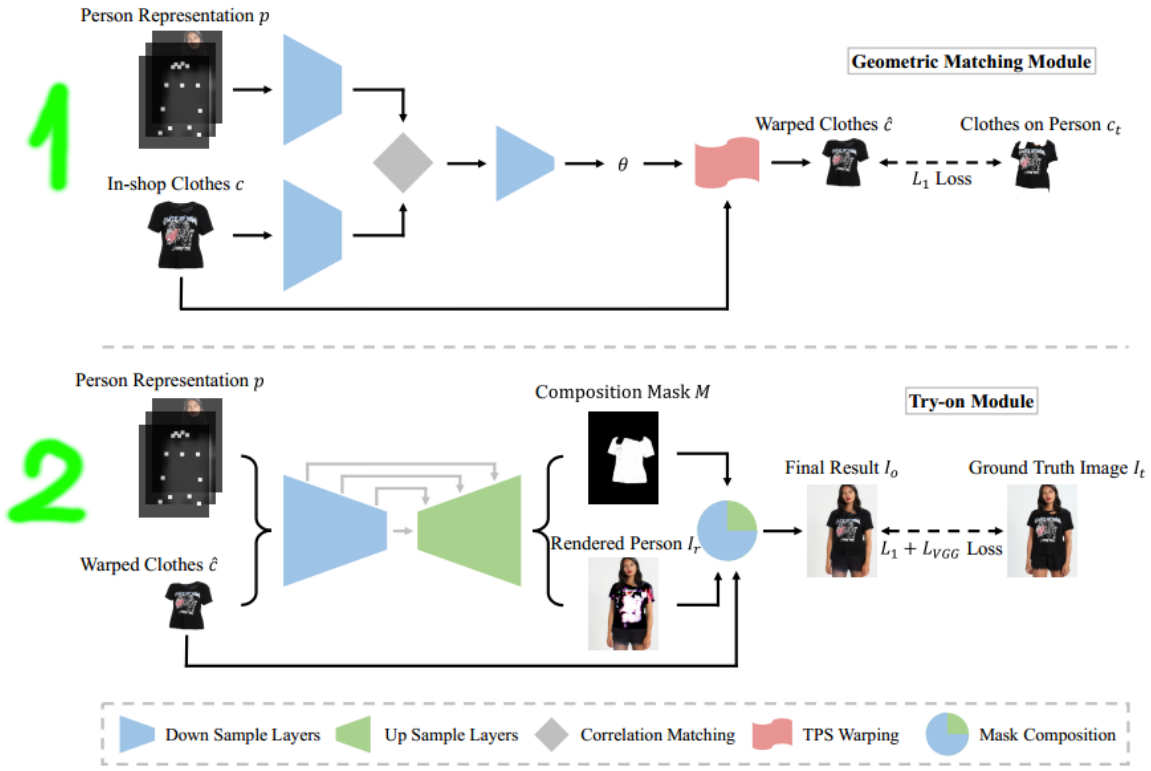
The first stage trains a module called Geometric Matching Module to predict a thin plate spline deformation that warps the in-shop cloth image to match the corresponding cloth on the model image. After training is done, the model is used to deform/warp all the input in-shop clothes images, which will be used as the training data for the next stage.

The second stage trains another module called Try-On to put the warped clothes on the human subject. Instead of predicting directly the final result, this module predicts a rendered person image and a composition mask separately and then use the mask to blend the rendered person image with the input warped image to form the final result.

**Step-by-Step Process of GMM:**

1. **Input Images**:
    - Two images are used as input:
        - A **person image** (usually with the clothes removed or masked).
        - A **clothing image** (usually a flat image of a shirt, dress, etc.).
1. **Feature Extraction**:
    - Separate convolutional neural networks extract important features from both the person image and the clothing image.
    - These features include pose, shape, and texture information.
1. **Feature Correlation**:
    - The system compares the extracted features from the person and the clothing.
    - A **correlation map** is generated to find how well each part of the clothing aligns with the person's body.
1. **Predict Transformation Parameters**:
    - A regression network uses the correlation map to estimate how the clothing should be warped or adjusted.
    - This step predicts the **geometric transformation** needed to reshape the clothing image to match the person's body.
1. **Warp the Clothing**:
    - Using the predicted transformation, the clothing is **warped** (bent, resized, repositioned) to fit the target body.
    - This step ensures the clothing aligns with arms, shoulders, and torso areas in a realistic way.
1. **Segmentation Assistance** (optional enhancement):
    - A helper network may generate a segmentation map to guide the transformation process.
    - This ensures that specific parts like sleeves, collar, etc., are accurately positioned.
1. **Training the GMM**:
    - During training, the system compares the warped clothing with the expected output (ground truth).
    - The model learns to improve accuracy using losses like image difference and segmentation accuracy.

From two input data: standard in-shop clothes image cc and a person representation p, the module GMM learns to warp the clothes image so that it aligns with the target pose in the person image.




**Figure#.**

**Calculate person representation:** p

The person image is not passed to the model directly because in the test time this image is not available. Therefore, the input person is image is transformed to another person representation to get rid of information about old clothes, color, texture and shape and still preserves face, hair and general body shape of the target. As described in the section 3.1 in the paper, the human representation consists of

- A pose heat map is a 18-channel image where each slice encodes the heat map of a skeleton joint.
- A body shape map is 1-channel image which describes the blurred shape of the person.
- An RGB image that contains the facial and hair region of the human subject.

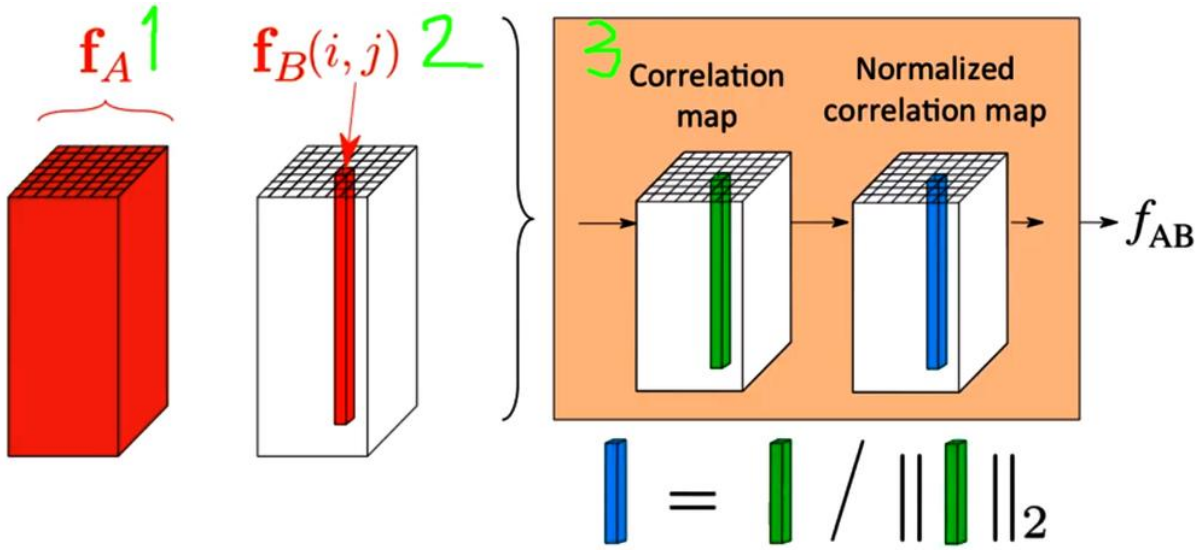
**Figure#.**

**Find feature correlation:**

After the person representation p is extracted from the input image, with the cloth image cc, they are passed to two separate feature extraction modules, each of which consists of a chain of 2-strided down-sampling convolutional, batch normalization and relu layers,Both extracted features are then passed to a correlation module that is supposed to merge them into a single tensor that encodes the correlation between the person pose and the standard in-shop clothes. Specifically, assume that

1. is the feature map from the person representation pp, and

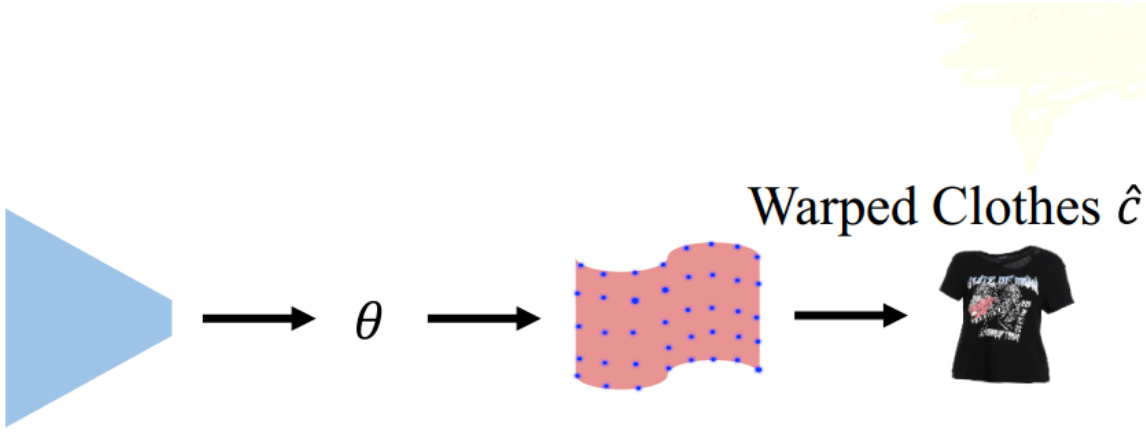
2. is the feature map from the in-shop cloth cc, each feature column in the feature map 3. encodes the similarities between the corresponding feature column in 2 and every other feature columns in 1. In other words, the correlation map 3 encodes the pairwise similarity between two feature maps 1 and 2.




**Figure#.**

## 4.2.4 Predict Thin-Plate-Spline transformation

The correlation map is then passed to a regressor (the blue trapeze block) that predicts the warped control points for the Thin-Plate-Spline stage, as depicted by the blue points in the below figure. These blue control points will be then used to solve for a smooth Thin-Plate-Spline transformation that warps the input in-shop clothes images to align with the target clothes images on the human subject. In other words, the Thin-Plate-Spline transformation is learned by minimizing the MSE loss between the warped clothes and the corresponding target clothes.




**Figure#.**

**How is the Thin Plate Spline (TPS) transformation generated?**

**General idea:**

In this part, I will go deeper into explaining how the TPS transformation is generated because it seems that it is the most important step in the Geometric Matching Module.

Given two sets of grid control points, the module **TpsGridGen** estimates a Thin Plate Spline transformation that warps the in-shop clothes to match the person pose.

**The first control point set**, as shown in the left picture, is constructed in the initialization stage and does not change during the training process. **The second control point set** in the right picture is the prediction result from the regressor, as represented by the tensor variable **theta** in the previous code.

**Summary of GMM Pipeline:**

1. Extract features from person and cloth images.
2. Compute feature correlation and regress TPS parameters.
3. Warp the cloth using TPS.
4. Optimize the warping and segmentation performance using combined loss.

**Why GMM Matters in Virtual Try-On?**

The GMM module ensures that the clothing appears naturally fitted on the user's body, maintaining alignment with the pose and preserving the realistic shape. This alignment is critical for generating convincing virtual try-on images.

- **Accurate Alignment**: A correctly warped garment aligns with body contours and posture, reducing distortions.
- **Texture Preservation**: The TPS allows smooth deformations that maintain visual details such as text, prints, and patterns.
- **Robustness**: Integrating auxiliary segmentation adds semantic awareness to the warp, helping the system handle occlusions and complex poses.
- **End-to-End Learning**: GMM is trained jointly with the auxiliary loss, improving feature representation without requiring separate steps.

**Thin Plate Spline (TPS):**

**Introduction:**

Thin Plate Spline (TPS) is a geometric transformation technique that is commonly used for non-rigid image warping. In the context of virtual try-on systems, the TPS transformation plays a critical role in aligning a clothing item with the pose and shape of a target person. It is designed to handle complex and flexible deformations,

making it particularly suitable for clothing applications where rigid transformations (such as rotation, translation, and scaling) are insufficient.

The TPS transformation is a key component of the Geometric Matching Module **(GMM)**, which is responsible for aligning the source clothing image with the body representation of the target person before merging the two in the final synthesis stage.

**Purpose of TPS in Virtual Try-On:**

The main purpose of using the Thin Plate Spline in a virtual try-on pipeline is to adapt the source garment to match the target person's body in terms of shape and pose. Since garments are deformable objects, they need to be adjusted flexibly to follow body contours, sleeve curves, and other spatial variations.

TPS allows for such non-linear warping while preserving the smoothness of the transformation. It ensures that the resulting garment overlay appears natural and realistic, without distortions or visual discontinuities.

**Working Principle of TPS in the Pipeline:**

The Thin Plate Spline module operates within the Geometric Matching Module (GMM) and performs the following key functions:

**1. Control Point Estimation**

A neural network takes both the source clothing image and the body representation (e.g., a pose map or segmentation map of the person) as input. From these inputs, it learns to predict a fixed number of control points on both the clothing and the body.

- **Source control points**: Located on the source garment.
- **Target control points**: Located on the corresponding regions of the target body.

These control points are used as anchors to determine how the clothing should be warped.

**2. TPS Transformation Estimation**

Once the control points are predicted, the TPS transformation is calculated. This transformation defines a smooth mapping from the source points to the target points.

The TPS function interpolates the movement of all pixels in the clothing image based on the displacement of these control points. The mathematical model ensures that the resulting warp is smooth and minimally distorted, spreading the deformation evenly across the image.

**3. Image Warping**

The TPS parameters are applied to the entire clothing image to warp it. The result is a transformed version of the original garment that conforms to the spatial configuration of the target body.

This step ensures that features such as the sleeves, shoulders, or hem of a shirt align correctly with the person's limbs and torso in the target image.

**4. Integration into the Try-On Pipeline**

After the garment is warped using the TPS transformation, it is passed on to the next stage of the pipeline, typically a synthesis or rendering module such as the Try-On Synthesis Parser (TSP). There, it is blended with the original person image to generate the final try-on output.

**Advantages of Using TPS:**

1. **Non-Rigid Deformation**: TPS allows garments to deform in a flexible way, unlike rigid transformations which cannot simulate body-conforming fit.
2. **Smooth Warping**: The transformation generated by TPS is continuous and smooth, which helps avoid unnatural kinks or folds in the garment overlay.
3. **Preservation of Fine Details**: TPS, especially when used in conjunction with skip connections in deep networks, helps retain texture and structural details of the original garment.
4. **Pose Adaptability**: TPS enables the garment to be adjusted according to varying human poses, which is essential in real-world virtual try-on applications.

**Summary:**

In virtual try-on systems, the Thin Plate Spline transformation provides a powerful and flexible method to warp clothing images so they conform naturally to the shape and pose of a target human body. By predicting control points and estimating a smooth transformation, TPS ensures that the aligned clothing appears realistic and visually convincing when overlaid on the person. This makes TPS an essential component in achieving high-quality and personalized virtual fashion experiences.

## 4.2.5 ALIAS: Alignment-Aware Feature Normalization

**Overview:**

In Virtual Try-On systems, a key challenge is how to **seamlessly integrate the warped garment** with the person's body in a visually realistic manner. While warping the clothing (via TPS or GMM) helps match the body pose and shape, **merging the warped garment with the person's appearance** without visual artifacts or loss of detail is a complex task.

To address this, the system integrates **ALIAS (Alignment-Aware Feature Normalization)**—a specialized normalization technique designed to preserve **spatial alignment** between the person and the clothing while enhancing **feature fusion** quality. ALIAS improves the realism and continuity in the final try-on output by retaining sharp transitions and structural consistency.

**Function of ALIAS in the Pipeline:**

ALIAS is used in the **Try-On Module**, where it modulates feature maps during synthesis. Its function can be broken into the following steps:

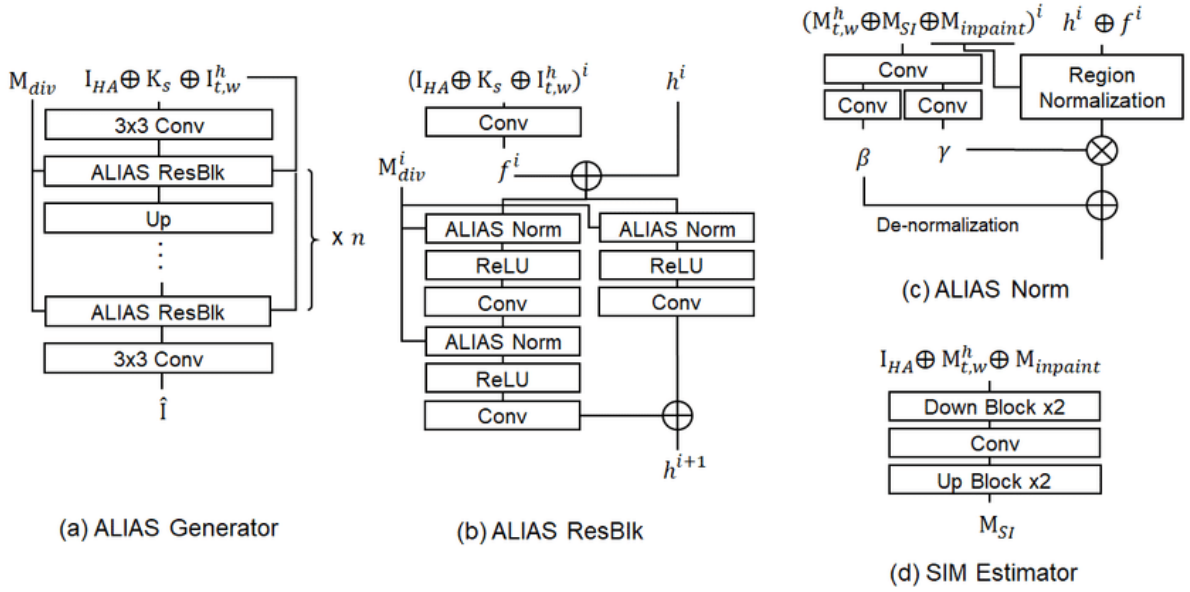
1. **Feature Encoding**:
    - The warped garment and person representation (which includes pose and segmentation maps) are passed through convolutional encoders.
    - Each encoder produces a set of intermediate feature maps for further processing.
1. **Normalization with Spatial Awareness**:
    - Instead of applying traditional normalization (like BatchNorm or InstanceNorm), ALIAS adjusts the feature statistics **conditioned on pose and spatial layout**.
    - This ensures that important spatial landmarks (e.g., necklines, shoulders, arms) retain sharp boundaries and consistent textures after normalization.
1. **Fusion through Modulation**:
    - ALIAS computes affine modulation parameters (scale and shift) from the pose-conditioned representation.
    - These parameters are then used to modulate (normalize and re-scale) the feature maps of the clothing image.
    - As a result, clothing textures are blended with the person's structure while preserving garment identity and alignment.

**Implementation in the Try-On Network:**

In our virtual try-on project pipeline, ALIAS is implemented as follows:

- It is embedded within the **Generator Network** in the Try-On Module.

- The generator receives two inputs:
  a. The **warped garment** from the TPS/GMM output.
  b. The **pose-aware person representation**, which includes segmentation masks, pose maps, and potentially the original image background.
- These inputs are processed through convolutional layers with ALIAS normalization.
- The output is a refined, high-resolution image where the garment looks **naturally worn** by the target person, respecting both shape and pose.




(a) ALIAS Generator (b) ALIAS ResBlk (c) ALIAS Norm (d) SIM Estimator

**Figure#.**

**Key Advantages in Virtual Try-On:**

- **Edge Preservation**:
  - Maintains clarity in regions like the **neckline, collar, shoulder**, and **sleeves** that typically suffer from blurring during fusion.
- **Pose-Conditioned Adaptation**:
  - Adapts normalization based on the person's pose, improving alignment between clothing and body.
- **Better Texture Integration**:
  - Helps to retain intricate garment details such as **fabric folds, prints, and seams** even after merging.
- **Smooth Blending**:
  - Produces a more **realistic and natural try-on output**, free from artifacts or texture bleeding.

**Camera Feature & Favorites List in Virtual Try-On System**

### 1. Open Camera Feature

This feature allows users to capture real-time images using their device's camera for virtual try-on instead of uploading static photos.

### Key Functionalities:

- **Live Capture Mode**:
  - Users can position themselves in front of the camera to get real-time pose estimation.
  - Enables dynamic adjustments (e.g., turning, hand placement) for better try-on accuracy.
- **Background Removal (Optional)**:
  - AI-powered background segmentation (e.g., using U²-Net or MODNet) to isolate the user.
- **Pose Detection & Alignment**:
  - Detects body landmarks (using OpenPose or MediaPipe) to align clothing realistically.
- **Instant Preview**:
  - Shows a simulated try-on before final processing.
- **Save Snapshot**:
  - Users can save the captured image for later use.

### Technical Implementation:

- **Frontend (Streamlit)**:
  - Uses streamlit-webrtc for real-time camera access.
  - Displays live feed with try-on overlay.
- **Backend Processing**:
  - Pose estimation (MediaPipe/OpenPose).
  - Garment warping (using Thin Plate Spline or GAN-based methods).
  - Final rendering (U-VTON or similar model).

## 2. Add to Favorites List

This feature allows users to save preferred clothing items for future reference.

### Key Functionalities:

- **Save Favorite Items**:

    - Users can bookmark clothing items they like (from uploads, text-to-image, or recommendations).
- **Organize Favorites**:
    - Categorize by type (e.g., shirts, dresses).
    - Filter by color, style, or brand.
- **Quick Re-Try**:
    - Users can reuse favorited items without re-uploading.
- **Sync Across Devices (Optional)**:
    - If user accounts exist, favorites are stored in a database.

## Technical Implementation:

- **Frontend (Streamlit)**:
    - "Favorite" button on each clothing preview.
    - A dedicated "Favorites" page with thumbnails.
- **Backend Storage**:
    - SQLite (for lightweight storage) or Firebase/PostgreSQL (for cloud sync).
    - Stores:
        - User ID (if logged in).
        - Clothing image/text reference.
        - Metadata (category, color, etc.).

**Integration in Workflow**

1. User opens camera → captures image → tries on clothing.
2. If satisfied, clicks "Add to Favorites" for later use.
3. Later, they can browse their Favorites list and reapply items.

This enhances user experience by:

✓ **Reducing repetitive uploads** (via camera & favorites).

✓ **Personalizing recommendations** (based on saved items).

✓ **Improving engagement** (quick access to liked clothes).

**Summary:**

The inclusion of **ALIAS normalization** represents a crucial advancement in Virtual Try-On architecture. By introducing **pose-aware spatial modulation**, it enables the system to **preserve garment fidelity** and **body alignment**, leading to more convincing and realistic synthetic images. This is particularly important for real-time

applications such as virtual fitting rooms and fashion e-commerce platforms, where visual accuracy is critical for user experience.

### 4.2.2 Web Application Deployment:

Our virtual try-on system was designed and deployed entirely on-premises, leveraging local high-performance computing resources to maintain full control over hardware, data privacy, and model customization. The solution combines a Streamlit-based web interface with SQLite database persistence, all hosted on dedicated institutional servers to ensure reliability and low-latency performance without relying on cloud services.

The system runs on powerful local hardware including NVIDIA RTX 3090 GPUs with 24GB VRAM for accelerated deep learning inference, supported by 32GB RAM and multi-core processors to handle parallel tasks efficiently. Storage is configured with RAID-enabled SSDs to optimize I/O operations during intensive image processing tasks. This local infrastructure provides the necessary computational power for real-time virtual try-on generation while keeping all data securely within institutional networks.

For the user interface, we implemented a responsive Streamlit application that handles image uploads, real-time camera processing, and interactive visualization of try-on results. The framework's simplicity allowed rapid development of an intuitive interface while supporting complex integrations with our deep learning models. Caching mechanisms ensure frequently used models remain loaded in GPU memory, reducing inference times for subsequent requests.

Data persistence is managed through a SQLite database that stores user preferences, favorited clothing items, and session histories. The lightweight database solution provides sufficient performance for our current user load while maintaining data integrity through automated backups. User information and generated content remain securely stored on local servers, addressing privacy concerns that might arise with cloud storage solutions.

Network accessibility is managed through Nginx configured as a reverse proxy, handling HTTPS encryption and load balancing to distribute requests efficiently. Systemd services monitor and maintain application uptime, automatically recovering from unexpected interruptions. This setup ensures reliable access for users within the institutional network while maintaining security through firewall protections and optional VPN access for authorized remote users.

The on-premises deployment offers several advantages including complete data control, consistent performance without cloud latency, and elimination of recurring service fees. The localized approach also allows for greater flexibility in model customization and system tuning, as we have direct access to both hardware and software components. While currently optimized for moderate user loads, the architecture is designed to support future scaling through containerization and orchestration technologies if needed.

Security measures include encrypted storage volumes, strict access controls, and comprehensive logging of system activity. These protections ensure compliance with institutional data policies while maintaining the system's responsiveness. Performance monitoring tools track GPU utilization and processing times, allowing for continuous optimization of resource allocation.

This self-contained deployment model demonstrates how advanced computer vision applications can be implemented effectively within organizational infrastructure constraints. By maintaining all components on-premises, we achieve a balance of performance, privacy, and cost-efficiency while delivering a seamless user experience. The system's design accommodates future enhancements, whether through hardware upgrades, model improvements, or expanded user capacity, all while retaining the benefits of local control and customization

## 4.3 Experimental / Simulations Setup

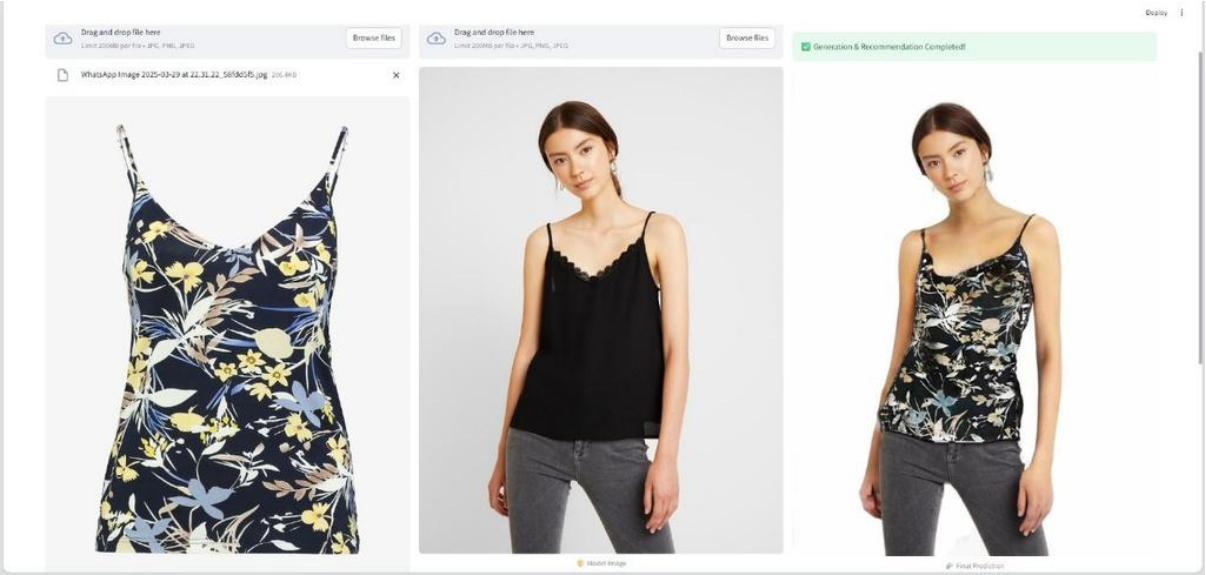
For our virtual try-on System, we leveraged one specific dataset to handle different segments of attire **VITON-HD**: This dataset is primarily used for experiments involving the upper body garments. It features high-resolution images that are crucial for maintaining the quality and realism in virtual try-ons.

We created a web app using Google **Colab** and **Streamlit** for seamless data integration and interactive visualizations.

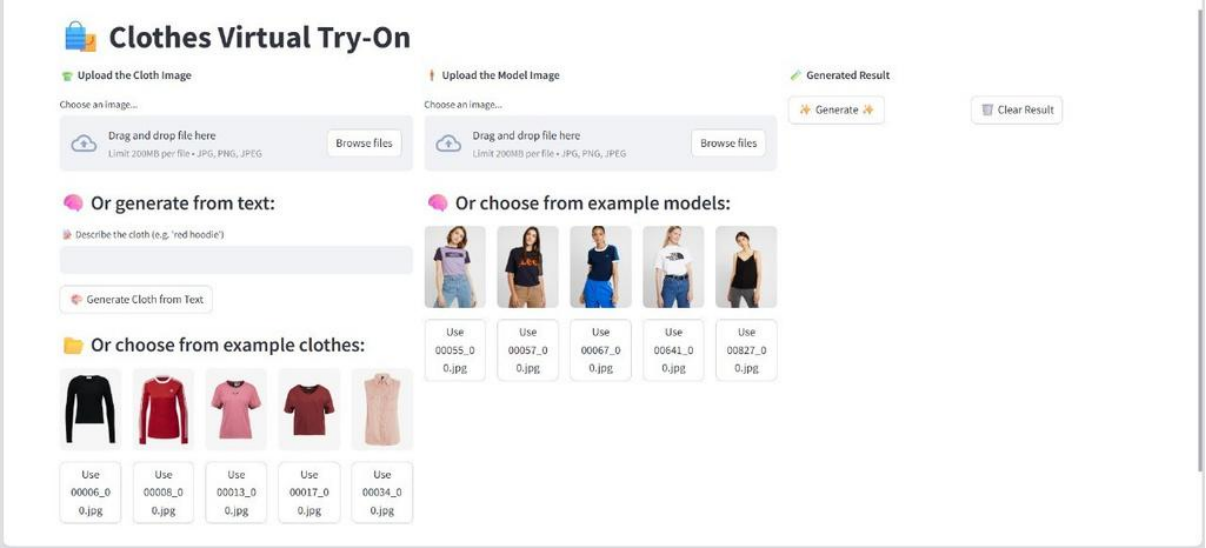
## 4.4 Conducted Results

The input garment and person undergo our preprocessing pipeline, where various transformations occur, although these intermediate preprocessing images are not directly shown to the user. Instead, they serve as inputs to the model Behind the

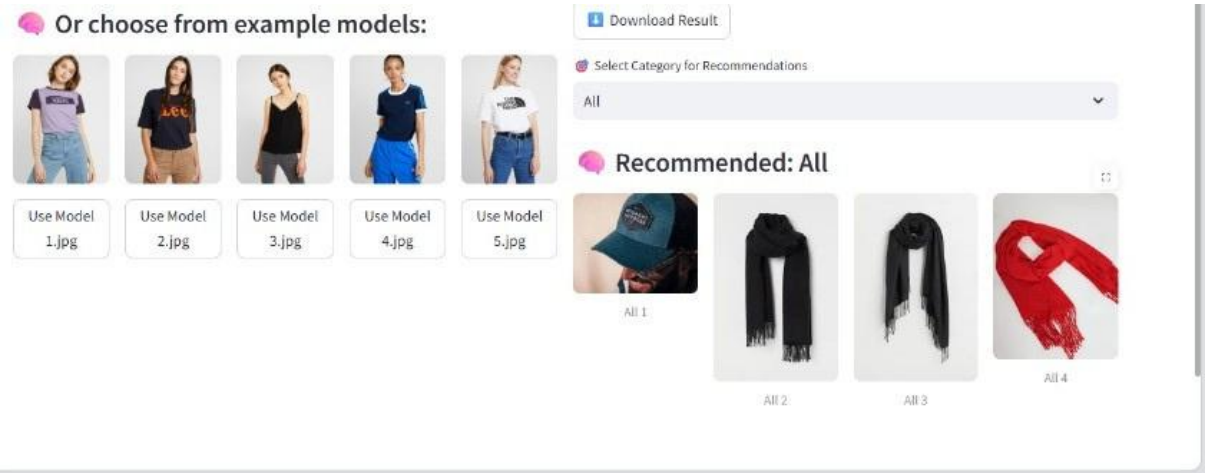
scenes, which then generates the results. Upper body We used Viton HD dataset that focused solely on upper garments.



**Figure#.**




**Figure#.**




**Figure#.**

**Overall Flow of the Virtual Try-On System:**

This virtual try-on system is implemented as a three-stage pipeline. Each stage transforms and processes the input image data to generate a realistic try-on result where a person is shown wearing new clothes.

Three-Stage Pipeline:

1. **Segmentation Prediction**
   The first stage predicts the human body segmentation map conditioned on the new clothing. This helps in understanding where each body part is located to guide garment placement.
2. **Cloth Warping**
   The second stage involves deforming or warping the clothing item to fit the body structure and pose of the target person.
3. **Try-On Synthesis**
   The final stage blends the warped garment and the person's features to generate a realistic image of the person wearing the new clothing.

**Main Inference Script: test.py**

This script handles the execution of the virtual try-on pipeline. It is the primary entry point for testing the system.

**Functions:**

- **get_opt()**
  Parses command-line arguments such as:
  - Paths to data folders

- Batch size
- Checkpoint file paths

- **main()**

  This function is responsible for:
  - Loading pretrained models: SegGenerator, GMM, and ALIASGenerator
  - Loading model weights using load_checkpoint
  - Executing the test() function
- **test()**

  Runs the core inference stages:
  - Loads the test data using VITONDataset and VITONDataLoader
  - **Stage 1**: Uses SegGenerator to predict body segmentation with new clothes
  - **Stage 2**: Uses GMM to warp the clothes to match the target body
  - **Stage 3**: Uses ALIASGenerator to produce the final output image
  - Saves results using save_images()

**Utility Functions: utils.py**

This file includes common helper functions used across the codebase.

**Functions:**

- **gen_noise(shape)**

  Generates random noise input to make segmentation output more robust and diverse.
- **save_images(tensors, names, save_dir)**

  Converts tensors to .jpg images and saves them to the output directory.
- **load_checkpoint(model, path)**

  Loads model parameters from a .pth file into a model.

# Chapter 5

# Conclusions

## 5.1 Limitations

Despite the promising results achieved by the Fashionista system, there are several limitations that we acknowledge in the current version of the project.

First, the system is currently limited to upper-body garments such as shirts, jackets, and t-shirts. Full-body try-on for pants, dresses, or layered outfits is not yet supported, primarily due to limitations in the datasets used for training and parsing accuracy for lower-body segmentation.

Second, the image synthesis relies heavily on the accuracy of segmentation and pose estimation. Minor errors in human parsing or misalignment in pose keypoints can lead to visible artifacts, such as distorted garment placement or incorrect limb coverage.

Third, our personalized recommendation system depends on image-based embeddings and user preferences but does not yet incorporate user feedback loops, purchase history, or deep behavioral learning, which could enhance recommendation quality.

Finally, the performance of the system depends on relatively powerful computing resources. While inference has been optimized, training and high-resolution generation require GPUs, making it difficult to deploy the complete pipeline on low-resource devices.

## 5.2 Future Work

Based on the current limitations of the Fashionista system, we identify several directions for future development that can improve its functionality, scalability, and user experience.

1. **Support for Full-Body Garments**
   Future work should expand the system's capabilities to handle full-body garments such as pants, dresses, and layered outfits. This would require

collecting or fine-tuning datasets that include comprehensive body annotations and improving segmentation models to handle lower-body and occluded regions more accurately.

2. **Improving Parsing and Alignment Accuracy**
   To reduce artifacts and misalignment issues, we propose incorporating hybrid models or error-correction mechanisms that refine pose keypoints and segmentation maps in real time. Adding attention mechanisms or feedback loops during image synthesis may also improve overall visual consistency.
3. **Enhancing the Recommendation Engine**
   The recommendation system can be extended to include behavioral data, such as user feedback, purchase history, and seasonality. This would enable more personalized suggestions and adaptive styling based on current trends and user interaction.
4. **Resource Optimization and Deployment**
   Future versions of the system should focus on deploying a lightweight version of the pipeline optimized for mobile and browser-based platforms. Using quantization, model distillation, or cloud inference techniques would ensure accessibility to users with low-end devices without sacrificing quality.
5. **Integration of Generated Garments**
   To complete the personalization cycle, the generated garments from text prompts can be automatically classified and integrated into the user's wardrobe, allowing seamless try-on and matching with existing outfit components.
6. **Mobile Application Development**
   To maximize accessibility and user engagement, a dedicated mobile application is planned as part of future development. This app would allow users to upload their photos, browse or generate garments, receive personalized recommendations, and perform virtual try-ons all in one streamlined interface. The mobile app will also integrate camera-based try-on features, wardrobe management, and shareable social outputs, making the experience more immersive and user-friendly.

# References

[1] Seunghwan Choi, Sunghyun Park, Minsoo Lee, Jaegul Choo, VITON-HD: High Resolution Virtual Try-On via Misalignment-Aware Normalization, South Korea, 2021.

[2] Peike Li, Yunqiu Xu, Yunchao Wei, Yi Yang, Self-Correction-Human-Parsing, Centre for Artificial Intelligence, University of Technology Sydney, 2020.

[3] Yisol Choi1, Sangkyung Kwak1, Kyungmin Lee1, Hyungwon Choi2, and Jinwoo Shin1 2403.05139v3

[4]Daniel Gatis, Rembg remove images background, 2020.

[5] D. Gatis, *rembg: Image Background Removal Library*, GitHub repository, 2020. Available: https://github.com/dinesh4444/rembg-library.

[6] D. Gatis, **rembg**: Image Background Removal for Python, Python Package Index (PyPI), 2020. Available: https://pypi.org/project/rembg/.

[7]Pillow Contributors, **Pillow**: Python Imaging Library (Fork) Documentation, 2025. Available: https://pillow.readthedocs.io/en/stable/.

[8] LearnOpenCV, **U²-Net for Image Segmentation** Tutorial, 2024. Available: https://learnopencv.com/u2-net-image-segmentation/.

[9]GeeksforGeeks, **U-Net Architecture Explained**, 2025. Available: https://www.geeksforgeeks.org/u-net-architecture-explained/.

[10] J. Kim, G. Gu, M. Park, S. Park, and J. Choo, “StableVITON: Learning Semantic Correspondence with Latent Diffusion Model for Virtual Try-On,” arXiv preprint arXiv:2312.01725v1, 2023. Available: https://arxiv.org/pdf/2312.01725v1.

[11] Papers With Code, **Geometric Matching: Latest Methods**, 2025. Available: https://paperswithcode.com/task/geometric-matching/latest.

[12]Lvmin Zhang and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. arXiv:2302.05543 [cs.CV]

[13] Yisol Choi, Sangkyung Kwak, Kyungmin Lee, Hyungwon Choi, and Jinwoo Shin, "IDM-VTON: Toward High-Fidelity Virtual Try-On with Latent Diffusion Models", 2024. Available at: https://huggingface.co/spaces/yisol/IDM-VTON

[14]Xinyu Gong, Yujun Shen, Jiapeng Zhu, Yinan He, and Bolei Zhou, "Fashion-VDM: Video Diffusion Model for Virtual Try-On", 2024. Available at: https://paperswithcode.com/paper/fashion-vdm-video-diffusion-model-for-virtual-1

[15]VITON-HD Dataset – High-Resolution Virtual Try-On Dataset. Available at: https://paperswithcode.com/dataset/viton-hd

[16]Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. 2023. Text-to-image Diffusion Models in Generative AI: A Survey. arXiv:2303.07909 [cs.CV]

[17]Hammaad ali, Real Fashion, real dataset which contains images of various kinds of clothing accessories, 2020.

[18] **Yu Liu, Yan Gao, and Tingrong Wu.** *Virtual try-on based on attention U-Net*. Available at: https://www.researchgate.net/publication/362031904_Virtual_try-on_based_on_attention_U-Net

[19][38] Chenshuang Zhang, Chaoning Zhang, Sheng Zheng, Mengchun Zhang, Maryam Qamar, Sung-Ho Bae, and In So Kweon. 2023. A Survey on Audio Diffusion Models: Text To Speech Synthesis and Enhancement in Generative AI. arXiv:2303.13336 [cs.SD]

[20] Ai model like (Chat GPT4 ,Deep seek,GROK)

[21]*"Context Encoding for Semantic Segmentation"* – CVPR 2018