



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS**

**A Major Project Report
On
Virtual Try-On Using Generative Modeling**

Submitted By:

Prabin Bohara	(THA076BCT026)
Prabin Sharma Poudel	(THA076BCT027)
Raj Kumar Dhakal	(THA076BCT033)
Sajjan Acharya	(THA076BCT038)

Submitted To:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

March, 2024



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS**

**A Major Project Report
On
Virtual Try-On Using Generative Modeling**

Submitted By:

Prabin Bohara	(THA076BCT026)
Prabin Sharma Poudel	(THA076BCT027)
Raj Kumar Dhakal	(THA076BCT033)
Sajjan Acharya	(THA076BCT038)

Submitted To:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in
Computer Engineering

Under the Supervision of
Er. Umesh Kanta Ghimire

March, 2024

DECLARATION

We hereby declare that the report of the project entitled “**Virtual Try-On Using Generative Modeling**” which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the **Degree of Bachelor of Engineering in Computer Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Prabin Bohara (Class Roll No: 076/BCT/026)

Prabin Sharma Poudel (Class Roll No: 076/BCT/027)

Raj Kumar Dhakal (Class Roll No: 076/BCT/033)

Sajjan Acharya (Class Roll No: 076/BCT/038)

Date: March, 2024

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a major project work entitled “**Virtual Try-On Using Generative Modeling**” submitted by **Prabin Bohara, Prabin Sharma Poudel, Raj Kumar Dhakal and Sajjan Acharya** in partial fulfillment for the award of Bachelor’s Degree in Computer Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor’s degree of Electronics and Communication Engineering.

Project Supervisor

Mr. Umesh Kanta Ghimire

Department of Electronics and Computer Engineering, Thapathali Campus

External Examiner

Project Coordinator

Mr. Umesh Kanta Ghimire

Department of Electronics and Computer Engineering, Thapathali Campus

Mr. Kiran Chandra Dahal

Head of the Department

Department of Electronics and Computer Engineering, Thapathali Campus

March, 2024

COPYRIGHT

The author has agreed that the library, Department of Electronics and Computer Engineering, Thapathali Campus, may make this report freely available for inspection. Moreover, the author has agreed that the permission for extensive copying of this project work for scholarly purpose may be granted by the professor/lecturer, who supervised the project work recorded herein or, in their absence, by the head of the department. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, IOE, Thapathali Campus in any use of the material of this report. Copying of publication or other use of this report for financial gain without approval of the Department of Electronics and Computer Engineering, IOE, Thapathali Campus and author's written permission is prohibited.

Request for permission to copy or to make any use of the material in this project in whole or part should be addressed to department of Electronics and Computer Engineering, IOE, Thapathali Campus.

ACKNOWLEDGEMENT

We would like to acknowledge our Department of Electronics and Computer Engineering, Thapathali Campus, HOD Er. Kiran Chandra Dahal, DHOD Er. Umesh Kanta Ghimire and our coordinator Er. Dinesh Baniya Kshatri for providing us with this chance to do a major project.

We cannot escape from thanking our lecturers and colleagues for their ideas, suggestions, and advice. We would also like to extend our sincere gratitude to our friends for their feedback and support during the process of completion of this project.

Prabin Bohara (THA076BCT026)

Prabin Sharma Poudel (THA076BCT027)

Raj Kumar Dhakal (THA076BCT033)

Sajjan Acharya (THA076BCT038)

ABSTRACT

"Virtual Try-On Using Generative Modeling" leverages computer vision and Generative Modeling techniques to revolutionize the fashion industry. It offers users a real-time, immersive experience for trying on virtual clothing items. The person representation module is implemented with **OpenPose and DeepLabV3+** architecture for the purpose of key-point detection and image segmentation respectively. By employing geometric matching techniques, **TPS algorithms** and **cloth refinement** using Style Aggregator Modules, the system accurately captures users' body shapes, dynamically maps clothing items onto virtual representations, and simulates realistic fabric behavior. Inclusion of **Depth Estimation Module** to estimate depth of a person from their image paves the path for construction of 3D representation of try-on output. The incorporation of disentangled encodings enables personalized customization by manipulating independent clothing attributes. Through extensive evaluations, we demonstrate the effectiveness and visual fidelity of Virtual Try-On, highlighting its potential to reshape online shopping, enhance customer engagement, and transform the way fashion is experienced.

Keywords: Fabric, Fidelity, Cloth Warping Module, Semantic Segmentation, Sobel, Style Aggregator Module, Thin Plate Spline

Table of Contents

DECLARATION.....	i
CERTIFICATE OF APPROVAL	ii
COPYRIGHT.....	iii
ACKNOWLEDGEMENT.....	iv
ABSTRACT	v
List of Figures.....	x
List of Tables	xii
List of Abbreviations	xiii
1 INTRODUCTION	1
1.1 Background	2
1.2 Motivation.....	2
1.3 Problem Definition.....	3
1.4 Objectives	4
1.5 Scopes and Applications	4
2 LITERATURE REVIEW	6
3 REQUIREMENT ANALYSIS	12
3.1 Hardware Requirements.....	12
3.2 Software Requirements	12
3.3 Feasibility Analysis.....	13
3.3.1 Technical Analysis	13
3.3.2 Economic Analysis	13
3.3.3 Market Analysis	13
4 SYSTEM ARCHITECTURE AND METHODOLOGY	15
4.1 Data Collection	16
4.2 Data Augmentation	16
4.3 Data Preprocessing.....	18

4.3.1	Data Integration and Aggregation.....	18
4.3.2	Data Cleaning.....	18
4.4	Person Representation.....	19
4.4.1	Body parts Segmentation.....	20
4.4.2	Person Representation Aggregation.....	23
4.5	Layout Matcher.....	23
4.6	Geometric Transformer.....	24
4.6.1	Feature Extraction.....	24
4.6.2	Feature Normalization	25
4.6.3	Feature Correlation	25
4.6.4	Feature Regression.....	25
4.6.5	TPS Network.....	25
4.7	Style Aggregator Module.....	29
4.7.1	UNET.....	29
4.7.2	VGG19.....	30
4.8	3D Try-On Module	30
4.8.1	3D Depth Estimation from 2D Images	31
4.8.2	Sobel Operation	33
4.8.3	Initial depth estimation	34
4.8.4	Final Depth Estimation	35
4.8.5	3D Try-On.....	36
4.9	Model Fundamentals.....	38
4.9.1	Activation Functions	38
4.9.2	Loss Functions	39
4.9.3	Optimizer	41
4.10	Full Stack Integration.....	43
5	IMPLEMENTATION DETAILS	44

5.1	Dataset Collection	44
5.2	Clothing Images	44
5.3	Dataset Preprocessing	45
5.3.1	Binary Masks	45
5.4	Dataset Preprocessing	48
5.4.1	Dataset Augmentation.....	48
5.4.2	Photometric Augmentation	52
5.5	Keypoint Detection using OpenPose	57
5.6	Implementation of Image Segmentation	58
5.7	Implementation of TPS Transformation	60
5.8	Implementation of Cloth Warping Module.....	63
5.8.1	Feature-Extraction Layer	63
5.8.2	Feature L2-Norm Layer	64
5.8.3	Feature Correlation Layer	64
5.8.4	Feature Regression Layer	64
5.9	Implementation of Style Aggregator Module	65
5.10	Sobel Operation Implementation	66
5.11	Depth Estimation	67
5.11.1	Initial Estimation.....	67
5.11.2	Final Depth Estimation	68
5.12	3D Output Generation.....	68
6	RESULT AND ANALYSIS	70
6.1	2D Try-On.....	70
6.2	3D Try-On.....	77
6.3	Full Stack Integration.....	81
7	FUTURE ENHANCEMENT.....	85
8	CONCLUSION.....	86

9 APPENDICES	87
Appendix A: Project Schedule.....	87
Appendix B: Architecture of U-NET.....	88
Appendix C: Architecture of VGG-19.....	89
References	90

List of Figures

Figure 4-1 System Block Diagram	15
Figure 4-2 Block Diagram of Person Representation	19
Figure 4-3: Architecture of OpenPose	20
Figure 4-4 Architecture of DeepLab-V3+	22
Figure 4-5 Block Diagram for Cloth Warping and Style Aggregator Module	24
Figure 4-6 Block Diagram for 3D Try-On.....	30
Figure 4-7: Depth Information Extraction from Binocular Disparity.....	31
Figure 4-8: Flowchart for Calculation of Depth from Binocular Disparity	32
Figure 4-9 Rest API Architecture	43
Figure 5-1: Dataset Sample.....	44
Figure 5-2: Sample of Clothing Image	45
Figure 5-3: Flowchart of Binary Mask Creation.....	46
Figure 5-4: Binary mask of Person and Cloth	47
Figure 5-5: Rotation of Cloth and Person Representation (+30 degrees).....	48
Figure 5-6: Translation of Cloth and Person Representation.....	49
Figure 5-7: Scaling of Cloth Representation	50
Figure 5-8: Scaling of Person Representation	50
Figure 5-9: Shearing of Cloth and Person Representation.....	51
Figure 5-10: Flipping of Cloth and Person Representation	52
Figure 5-11: Brightness Adjustment of Cloth and Person Representation	53
Figure 5-12: Color Adjustment of Cloth and Person Representation	54
Figure 5-13: Saturation Adjustment of Cloth and Person Representation.....	54
Figure 5-14: Hue Adjustment of Cloth and Person Representation	55
Figure 5-15: Gaussian Noise Addition in Cloth and Person Representation.....	56
Figure 5-16: Salt and Paper Noise Addition in Cloth and Person Representation	56
Figure 5-17: Demonstration of Key-point Detection with OpenPose	58
Figure 5-18: Image Segmentation with DeepLabV3+.....	60
Figure 5-19: Base Grid (left) and Warped Grid (right) from trained TPS model.....	63
Figure 5-20: Demonstration of Original Image (left), Sobel_X (middle), Sobel_Y (right)	67
Figure 6-1 Inputs to CWM model.....	70
Figure 6-2: Warped Cloth and its Mask from trained CWM model.....	71

Figure 6-3 Overlay of Warped Cloth over Person's mask and on Person	71
Figure 6-4 Total Loss Curve of CWM Module	72
Figure 6-5 Warped Cloth and Warped Mask as input to SAM.....	73
Figure 6-6 Isolated image and Pose	73
Figure 6-7 Rendered image and the Final refined output	74
Figure 6-8 Total Loss Curve of SAM Module.....	74
Figure 6-9: Demonstration with Custom Cloth	75
Figure 6-10 Demonstration with Custom Image	76
Figure 6-11 Demonstration of Unsatisfactory Result with Unique Pose.....	76
Figure 6-12 Loss Curve for Initial Depth Estimator	77
Figure 6-13: Loss Curve of Final Depth Estimation Network.....	78
Figure 6-14 Person, Target Cloth and 2D Try-On Output.....	79
Figure 6-15 3D Mesh Output (Front and Back).....	79
Figure 6-16: Demonstration of Person, Target Cloth and 2D Try-On Output.....	80
Figure 6-17: Front and Back of 3D Output with Unusual Result	80
Figure 6-18 Demonstration of User Input in Web Application	81
Figure 6-19 Demonstration of Illustrated Outputs in Web Application	81
Figure 6-20 Home Screen and Try-On Screen of Mobile App.....	83
Figure 6-21 Choice of Person(left) & Choices of Clothes(right) in Application	83
Figure 6-22 Chosen Person and Cloth in Mobile Application.....	84
Figure 6-23 Module Outputs (Left) & Final Try-On Output (Right)	84
Figure 9-1: U-Net Architecture.....	88
Figure 9-2 Architecture of VGG-19	89

List of Tables

Table 4-1: Table of Activation Functions	38
Table 9-2: Gantt Chart	87

List of Abbreviations

AI	Artificial Intelligence
ASPP	Atrous Spatial Pyramid Pooling
CNN	Convolutional Neural Network
CP-VTON	Characteristic Preserving Virtual Try-On
CV	Computer Vision
FCN	Fully Convolutional Network
GIC	Geometric Invariant Cloning
CWM	Cloth Warping Module
HD	High Definition
MAE	Mean Absolute Error
ML	Machine Learning
PDE	Partial Differential Equation
RMS	Root Mean Square
SAM	Style Aggregator Module
TPS	Thin Plate Spline
UI	User Interface
VTON	Virtual Try-On
VGG	Visual Geometric Group
ELU	Exponential Linear Unit

1 INTRODUCTION

The fashion industry has witnessed a remarkable **shift towards online retail** in recent years, resulting in the need for innovative solutions that replicate the in-store try-on experience. The challenge lies in the fact that customers cannot physically interact with clothing items before making a purchase, leading to uncertainty about fit, style, and overall satisfaction. This project report introduces the development of "Virtual Try-On" a state-of-the-art virtual try-on system that combines computer vision and generative modeling. By leveraging advanced algorithms and deep learning models, this system aims to bridge this gap by providing users with a seamless and immersive experience of virtually trying on clothing items in real-time. The system's primary objective is to accurately **capture users' body shapes, dynamically map clothing items** onto virtual representations, and simulate realistic fabric behavior. The incorporation of disentangled encodings further enables personalized customization, empowering users to make informed fashion choices and transforming the way online shopping is conducted.

The increasing trend of online fashion retail has underscored the importance of addressing the limitations associated with the absence of physical interaction with clothing items. Virtual Try-On seeks to tackle these challenges head-on by utilizing cutting-edge technologies. By integrating computer vision techniques, the system can analyze users' body shapes with precision, ensuring an accurate fit for virtual garments. Augmented reality technologies further enhance the experience by overlaying the virtual clothing seamlessly onto users' images, allowing them to visualize the appearance and style on their own bodies. Additionally, the **disentangled encoding approach allows for customized attributes** such as style, color, and texture manipulation, promoting a personalized and engaging try-on experience. This project aims to empower users, enhance customer satisfaction, and revolutionize the fashion industry by bridging the gap between online shopping and the tactile experience of trying on clothes. Through Virtual Try-On, we envision a transformative shift in how consumers interact with fashion, making informed decisions, boosting online sales, and fostering a new era of digital fashion retail.

1.1 Background

Over the years, the fashion industry has witnessed a significant transformation in the behavior of customers when it comes to purchasing clothing items. Traditionally, customers relied heavily on physical stores for the tactile experience of trying on clothes, assessing their fit, style, and overall appeal. However, with the advent of online shopping, there has been a notable **shift in consumer behavior**. More and more customers are now opting to make their fashion purchases online, leading to a diminishing physical interaction with clothing items. This shift has created a gap in the shopping experience, as customers are unable to try on clothes before making a purchase, resulting in uncertainty and potential dissatisfaction. This historical trend highlights the need for innovative solutions that replicate the in-store try-on experience in the online realm.

In the current landscape of the industry, the prevalence of online retail continues to rise. E-commerce platforms have become the go-to destinations for fashion enthusiasts, offering convenience, a wide variety of choices, and the ability to shop from anywhere at any time. However, despite these advantages, the **lack of a physical try-on experience** remains a significant drawback. Customers are often unsure about the fit, style, and suitability of clothing items when browsing online, leading to higher return rates and customer dissatisfaction. This gap in the online shopping experience has a direct impact on **customer confidence, sales conversion rates, and overall customer satisfaction**. To address this challenge, there is a pressing need for virtual try-on solutions that leverage technology, such as computer vision and augmented reality, to bridge the gap between the physical and digital fashion realms. A virtual try-on project can provide customers with a realistic and immersive experience of trying on clothes virtually, enabling them to make more informed purchasing decisions and significantly enhancing the online shopping experience.

1.2 Motivation

The rising trend of online retail poses a critical challenge for customers who seek the tactile experience of trying on clothes before making a purchase. Despite the proliferation of similar projects, a significant challenge remains in providing users with a seamless, realistic, and personalized virtual try-on experience. By leveraging

advanced technologies such as computer vision, augmented reality, and disentangled encodings, our project aims to surpass the current limitations and deliver a novel approach to virtual try-on. This research endeavor seeks to revolutionize the online shopping experience by offering users the ability to virtually try on clothing items, visualize their fit and style, and make informed purchasing decisions.

The motivation behind this project stems from the inherent need to address the existing gap in the market for virtual try-on solutions in the fashion industry; to bridge the gap between physical and digital fashion realms by providing users with a virtual try-on system that accurately captures their body shapes, simulates realistic fabric behavior, and enables customization of clothing attributes. The significance of this project lies in its potential to enhance customer confidence, reduce return rates, and transform the online shopping experience. Moreover, the project addresses the need for inclusivity by providing access to virtual try-on for individuals with physical limitations or limited access to physical retail spaces. By revolutionizing the fashion industry through advanced technologies and personalized experiences, this project has the potential to reshape consumer behavior, boost sales conversion rates, and foster a more sustainable and inclusive fashion ecosystem.

1.3 Problem Definition

The problem addressed in this project is the gap between the online shopping experience and the lack of a reliable and immersive virtual try-on solution, leading to uncertainty regarding fit, style, and customer satisfaction in the fashion industry. This limitation poses a significant challenge for both customers and fashion retailers, resulting in higher return rates, decreased customer confidence, and missed sales opportunities. Addressing this problem is crucial to enhance the online shopping experience, boost customer satisfaction, and transform the way fashion is perceived and consumed in the digital era.

1.4 Objectives

The objectives of this project are as follows:

- To fit **upper body garment** to the woman's body preserving the geometric features.
- To **render accurate 3D representation** of the woman wearing the target garment

1.5 Scopes and Applications

This project encompasses the development and implementation of a comprehensive virtual try-on system for the fashion industry, utilizing computer vision, **augmented reality**, and machine learning techniques. This project aims to fit the cloth over a person's image. The system accurately **captures users' body shapes**, enabling precise mapping of virtual clothing items onto their virtual representations. It simulates realistic fabric behavior, considering factors like **draping, stretching**, and movement. The project also involves the incorporation of disentangled encodings to enable personalized customization of clothing attributes such as style and color. The pose of the person can be complex and be handled well by the system as long as the pose shows the front part of the body clearly with **camera angle directly in front of the person**. The system can take inputs as image of person and any kind of cloth for specifically upper body. The system has an intuitive user interface for seamless interaction enhancing user engagement for demonstrating the try-on outputs.

However, the system requires the image of a woman since the datasets are heavily inclined towards women. While the primary focus is on clothing items, the system may have limitations in accurately representing complex clothing designs and fabric textures due to the constraints of virtual rendering. Since the datasets utilized for this project has images of people captured in front of their body with fully visible front part of the body, there may be major **issues if the images used are of different pose or another angle**. Hidden major body parts like shoulders and images with ambiguous distinction between upper and lower body might be some instances of images that would not be handled by the system. The system is highly sensitive. However, except those rare cases, simple pictures captured straight with visible front part of the body with less ambiguous background can be easily accepted by the system.

The virtual try-on system developed in this project has widespread applications that benefit the common man in various domains. In the fashion industry, the system enables customers to virtually try on clothing items before making **online purchases**, enhancing their confidence in fit and style choices. It provides a convenient and personalized shopping experience, reducing the likelihood of returns and improving customer satisfaction. In the field of virtual reality experiences, users can try on virtual clothing within virtual environments for **gaming, social interaction, or virtual fashion shows**. Moreover, the system can be integrated into digital styling platforms, virtual wardrobe management systems, and fashion design applications, empowering individuals to experiment with different styles, colors, and textures. The versatility of the system makes it applicable in e-commerce, entertainment, and personal styling, providing users with a unique and engaging virtual try-on experience.

2 LITERATURE REVIEW

The paper titled "VITON: An Image-based Virtual Try-on Network" [1] introduces an image-based approach to virtual try-on technology. The main objective of the paper is to develop a system that accurately maps clothing items onto user images, enabling a realistic virtual try-on experience. To achieve this goal, a **two-stage** framework comprised of a Warping Refinement Network (WRN) and a Composition Network (CN) is proposed.

In the first stage, the **WRN aligns the clothing item** with the user image by generating a warped clothing image that properly fits the user's body shape. This process involves extracting key points from the user image, estimating a warping field, and warping the clothing image accordingly. Additionally, the WRN incorporates an attention mechanism to refine the warped clothing image selectively. In the second stage, the CN **blends the warped clothing image with the user image** to produce the result. The CN employs a composition module that effectively combines the clothing and user images, considering their respective appearances and structures. To enhance fine-grained details, the authors introduce a self-attention mechanism within the CN. The proposed VITON framework employs both global and local refinement strategies, addressing challenges related to alignment, deformation, and realistic clothing appearance. laying the groundwork for further research and development in this domain. This **doesn't produce accurate output when dealing with complex poses**, occlusions, or low-quality input images.

After the advancement in the field with VTON, **CP-VTON** [2] introduced the concept of transforming the dresses into the most fitting shape along with proper preservation of the clothes' identity in the newly generated image by the model. Instead of implementing shape context matching for preserving details of the clothes, the authors have proposed a fully-learnable network (CP-VTON) for addressing possible challenges in the objective. The authors have represented the pose of the person with 18-channel feature map for one pose, 1 channel feature map for the body shape and used RGB regions for colored regions. The proposed network first learns a thin-plate spline transformation for the purpose of transforming in-shop clothes into the fitting shape of the body of the target person with the use of new Cloth Warping Module

(CWM). Thus, there is no computation of the corresponding interest points in the body as the prior researches did. CWM refers to an end-to-end neural network trained directly using pixel-wise loss that is responsible for simply aligning the cloth with a person representation and then produce a warped image of the cloth. They trained a network from scratch rather than implementation of pre-trained models like VGG network and used ground truth from the real clothes rather than the simulated warped clothes.

For the implementation of try-on, the authors used 12-layer UNet with dix 2-strided down-sampling convolutional layers as well as six sampling layers. UNet is simply a U-shaped network architecture with encoder and decoder that is designed to learn from fewer training samples. With these implementations, the authors were successful in preserving the sharp as well as the intact characteristics of the target clothes with rich details. Rather than being dependent on coarse-to-fine strategy, this model learns the coarse image of the person, align the clothes and then produce a new composition mask with rendered person and the warped cloth. VGG perceptual loss as well as misalignment between warped cloth and person are taken together for the refinement with more bias towards the composition mask of selecting the person rendered from UNet. The produced composition mask is successful in utilizing all the relevant information of the clothes aligned with the person and also plays an important role in balancing the smoothness of the image that is synthesized in the end of the process.

The paper titled "CP-VTON+: Clothing Shape and Texture Preserving Image-Based Virtual Try-On" [3] presents an image-based method of 'virtual try-on' called CP-VTON+. The objective of the paper is to address the challenges of accurately preserving clothing shape and texture during the virtual try-on process. The authors propose CP-VTON+, which is a two-stage framework consisting of a Shape Generation Network (SGN) and a Texture Generation Network (TGN). The SGN, by aligning it with the user's body shape, aims to generate the target shape of the clothing item. It has a shape generation module to estimate the deformation field between the clothing item and the user image, enabling precise shape alignment, and a shape rectification module to further refine the generated shape. Meanwhile, the TGN focuses on preserving the texture details of the clothing item. It utilizes a texture generation module that learns

the clothing texture from the reference image and applies it to the generated clothing shape.

This framework demonstrates significant improvements over previous methods in terms of preserving clothing shape and texture fidelity. The authors evaluate the performance of CP-VTON+ on various benchmark datasets, showcasing its superior capability in generating visually realistic and accurate virtual try-on results. For better output on varied cases, the authors suggest going for 3D reconstruction.

The paper “Towards Photo-Realistic Virtual Try-On by Adaptively Generating \leftrightarrow Preserving Image Content” [4] have addressed the issue of **generating** photo-realistic try-on images specially **when there is presence of large obstructions**. The authors have implemented a novel visual try-on network, Adaptively Content Generating and Preserving Network (AGCPN). The authors firstly divided the VITON dataset into 3 subsets as the ones with standard posture, the ones that has torse posture changes as the ones with medium level difficulty and lastly where the postural changes include the torso as well as the limbs of the person’s representation. The authors firstly predict the semantic layout and then adaptively determine the generation of the content.

For that purpose, the model proposed by the authors consists of three major modules, **SGM, CWM and CFM**. The SGM stands for Semantic Generation Module that implements mask generation mechanism for generating semantic representation of body parts and then to target the clothing region accurately. For this purpose, they have implemented two stage strategy with adoption of conditional GANs. The authors have trained a **body parsing GAN(G1) of UNet** [5] Architecture as the generator. For the generator part, they have deployed it in pixel-to-pixel implementation that that is responsible for distinguishing generated masks from the ground truth masks. CWM stands for Clothes Warping Module that has a second order difference constraint on the warping network for realizing geometric matching and retention of the characters. The authors believed that simple training of a Spatial Transformation Network (STN) and applying the thin-plate spline transformation **(TPS) would not enough** for the accurate transformation of the hard cases of the clothes modeling. This module imposes **collinearity of the local affine transforms**. Along with it, it is also responsible in maintaining the flexibility of TPS warping in a global manner. CFM stands for Content

Fusion Module which renders the results obtained from both SGM and CWM modules. It is responsible in adaptively preserving the fine-scale details of the body parts like gaps between fingers. The authors also claim to have solved instances of hard cases with prominent improvements with some extensive experiments.

The research paper titled "Learning to Transfer Texture from Clothing Images to 3D Humans" [6] introduces an innovative method for automatically transferring garments onto static models of human bodies. The study focuses on utilizing 3D garment images, converting them into 2D representations, and mapping them onto human body models. Notably, the **alignment of garments is based on the silhouette shape** rather than the texture, providing invariance to variations in clothing textures. To address this challenge, the authors propose a unique model called Pix2Surf, specifically designed for digitally mapping garments from online clothing images onto 3D models. A crucial aspect of their approach involves employing a **2D-UV mapping technique, facilitating the application of 2D textures onto 3D objects**. To train their models, the authors curate custom datasets by extracting data from popular clothing store websites such as Zalando, SAM Tailor, and Jack and Jones. The training process of the networks in the research paper relies on the Adam optimizer. For the segmentation network, the authors adopt a UNet architecture, incorporating instance normalization and employing color jittering techniques on input data to enhance performance. Additionally, Pix2Surf utilizes a ResNet [7] architecture with six blocks, leveraging the power of residual connections to address the issue of vanishing gradients. The UNet architecture, known for its effectiveness in image segmentation tasks, follows an encoder-decoder structure.

In summary, the research paper presents an original methodology for transferring textures from clothing images to 3D human models. The Pix2Surf model is introduced, highlighting the importance of 2D-UV mapping. The authors outline the process of creating custom datasets through web scraping and discuss the architectural choices made for both the segmentation network and Pix2Surf, including the utilization of UNet and ResNet architectures.

The paper "TryOnGAN" [8] introduces a novel deep learning approach for virtual try-on, focusing on generating realistic images of individuals wearing various clothing items. The authors acknowledge the specific challenges associated with virtual try-on,

such as accurate clothing deformation and precise alignment. To address these challenges, the proposed TryOnGAN leverages a generative adversarial network (GAN) framework. Notably, the architecture incorporates a generator network that utilizes spatial transformer and deformable skip connection modules to ensure realistic clothing deformation and alignment. The evaluation results provided in the paper demonstrate the superior performance of TryOnGAN compared to existing methods in terms of image quality, clothing deformation, and overall realism. Overall, the paper establishes TryOnGAN as an innovative and promising approach in the field of virtual try-on.

The research paper titled "Style-Based Global Appearance Flow from Virtual Try-On" [9] presents an original method for estimating style-based global appearance flow using the StyleGAN architecture. This framework enables precise control over both the overall structure and fine details of images. The primary objective of the study is to seamlessly fit in-shop garments onto human body images. To achieve this, the authors adopt a methodology involving **pre-training a parser-based model to segment** the human body into distinct parts and identify different body segments. The feature extraction process employs two convolutional encoders with identical architectures, extracting relevant feature maps from both the person image and the garment image. These feature maps are then utilized to predict the appearance flow. The proposed approach employs a global style vector method, initially estimating a rough appearance flow through style modulation, and subsequently **refining the predicted coarse** appearance flow based on local feature correspondence. The evaluation of the proposed method is performed on the VTON dataset, which consists of 14,221 pairs of training datasets and 2,023 pairs of testing datasets. Through comprehensive experimentation, the authors demonstrate the superior performance and robustness of their approach. Performance metrics such as the Frechet Inception Distance (FID) score and Structural Similarity Index (SSIM) score are used for comparisons with state-of-the-art methods, revealing that their method achieves new state-of-the-art performance, surpassing approaches like PF-AFN and Clothflow. Furthermore, a human evaluation is conducted to assess the quality of the generated try-on images, with their method obtaining a significantly higher preference rate compared to other models. The qualitative results exemplify the effectiveness of their approach in handling challenging scenarios, including difficult poses and barriers. The authors emphasize the method's robustness

in dealing with significant misalignments between person and garment images. In summary, this research paper makes a notable contribution to the field of virtual try-on by introducing a style-based global appearance flow estimation method that enhances the fidelity and realism of the virtual try-on process.

3 REQUIREMENT ANALYSIS

3.1 Hardware Requirements

- Processing power: High-performance CPU with multiple cores and high clock speed is required for intensive computation and training. The minimum bar is the Intel Core i5 processor.
- Graphics Card: A dedicated GPU with CUDA or OpenCL support, such as NVIDIA GeForce or AMD Radeon, to accelerate the rendering and visualization of virtual clothing in real-time.
- Camera: A high-resolution digital camera is required, preferably with about 12 megapixels, capable of capturing detailed images of the user's body for accurate fitting of virtual clothing.
- Display Device: A high-quality monitor with a high refresh rate and color accuracy is required to provide a realistic and immersive virtual try-on experience.

3.2 Software Requirements

- Image Processing Libraries: Popular image processing libraries are required such as OpenCV, scikit-image, or PIL to perform tasks like image segmentation, edge detection, and image manipulation.
- Machine Learning Frameworks: Deep learning framework PyTorch is necessary to train and deploy the models for tasks like cloth segmentation, pose estimation, and style code generation.
- MeshLab: It serves as a crucial software requirement for our project, facilitating the visualization and analysis of our 3D reconstructions. It functions as a versatile tool for processing and rendering PLY files, which store point cloud data, including color information. MeshLab supports tasks ranging from basic rendering to advanced mesh processing and simplification, providing an essential means for the visual representation of our RGBD data in the form of detailed and interactive 3D models.
- User Interface Design: A user-friendly interface is to be developed using web development techniques as well as mobile development framework like Flutter to enable users to interact with the system, select clothing items, and visualize the try-on results virtually.

3.3 Feasibility Analysis

3.3.1 Technical Analysis

The project requires advanced **image processing techniques**, including human pose estimation, clothing segmentation, and style code extraction. Various image-processing libraries and deep learning frameworks are evaluated and found to ensure that they can effectively handle these tasks. The hardware requirements such as high-resolution camera, GPU availability have been accessed. Careful consideration was given to optimize algorithm performance and efficiency.

3.3.2 Economic Analysis

The cost of development, including the actions of the team, and supervisors involved in the project has been evaluated and it fell within the project's estimated budget. An in-depth analysis of the cost of a high-performance CPU, GPU and processing devices fell under the allocated budget of the major project.

3.3.3 Market Analysis

“In 2023, the global e-commerce fashion industry is forecast to reach an overall market value of 821 billion U.S. dollars. According to estimates, the industry is expected to reach a value of over 1.2 trillion U.S. dollars by 2027.”, Statistica Research Development, May 30, 2023. Similarly, the global market for virtual fashion is estimated to be about **4.83 billion dollars in 2024** and is expected to grow to 15 billion dollars before 2029. This growth can be attributed to the increasing prominence of e-commerce and online shopping, where consumers are demanding and seeking more personalized and immersive experiences. Generation Z consumers express demand for virtual try-on experiences. Furthermore, with the widespread availability of smartphones and high-speed internet access, virtual try-on applications are becoming accessible to a larger audience across different age groups and regions. Likewise, Major fashion brands and retailers have begun incorporating virtual try-on functionalities into their e-commerce platforms. These trends show us that the TAM (Total Available Market) is huge in size. Furthermore, SAM (Serviceable Addressable Market) is still very huge since there still is significant room for innovation and advancement within the virtual try-on market. The SOM (Serviceable Obtainable Market) is still huge as the

national market can be realistically captured soon. Thus, our project has the potential to reshape the market and capture a substantial share of the growing virtual try-on market.

4 SYSTEM ARCHITECTURE AND METHODOLOGY

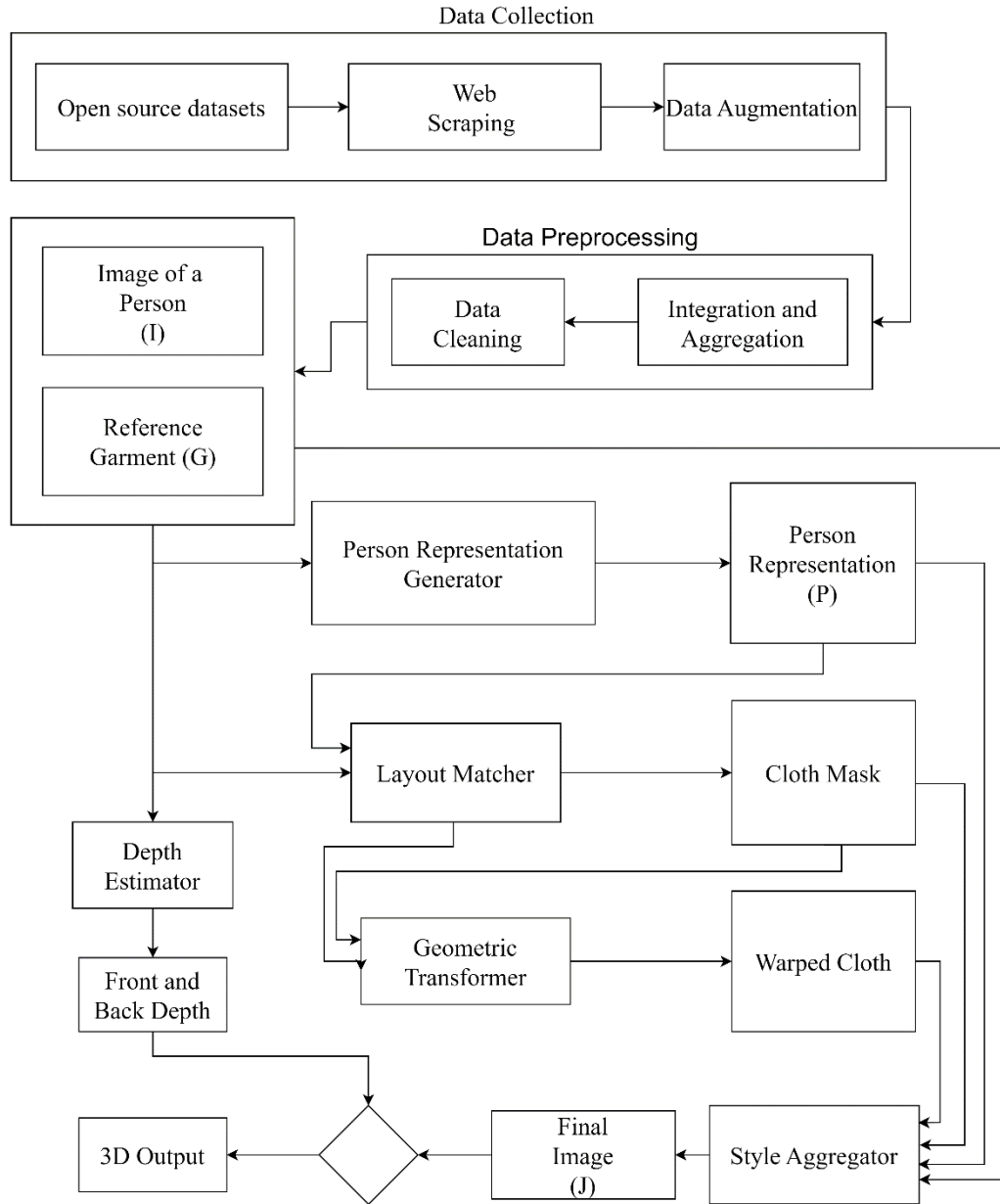


Figure 4-1 System Block Diagram

Given an Image ‘I’ of a person ‘P’ wearing garment ‘H’ and a reference garment ‘G’, the final outcome of the system would be the Image ‘J’ of person ‘P’ wearing garment ‘G’.

$$I = P + H \quad (4-1)$$

$$J = P + G \quad (4-2)$$

Thus, our system is able to approximate function (f) that maps I to J using neural network architecture.

$$f: I(P, H) \rightarrow J(P, G) \quad (4-3)$$

4.1 Data Collection

A crucial step in developing an accurate and reliable virtual try-on system is the collection of a comprehensive and diverse dataset. The dataset should consist of clothing images paired with corresponding body shape data. We have explored established fashion datasets like VITON, Deep-Fashion, VITON-HD, FashionIQ, which are well-known and widely used in the research community. More specifically, we have chosen VTON dataset for our project. It encompasses a diverse range of styles, colors, and poses, making it an excellent resource for training our virtual try-on system. The aligned images of a person and the corresponding cloth as ground truth is available in this dataset. Further processed elements of the two images are also available which have been useful in comparing processed data elements. For the generation of 3D output, half body pictures were not compatible for satisfactory result, which is why full body datasets were collected but for the purpose of 2D try-on, only upper body was considered. Full body was utilized only for the sake of 3D output and easier rotation of the obj file.

4.2 Data Augmentation

The Data augmentation process multiplies the dataset, enhancing its diversity and helps to reduce overfitting, better generalization, and in turn improves the performance of the model. Various techniques are employed such as:

- a. Geometric Augmentation: It includes Image rotation, Image flipping, Scaling, Cropping, Translation, etc. Rotation helps the model learn to handle clothing items that may appear in different orientations during virtual try-on. Image flipping allows the model to generalize better to both left-right and top-bottom orientations.

Similarly, Crop clothing images help to focus on specific regions of interest, such as the upper body or lower body, for finer-grained learning.

- b. Photometric augmentation: This augmentation technique includes applying color transformations, such as hue, saturation, and brightness adjustments, to clothing images. Adjusting the colors helps the model learn to handle variations in lighting conditions and color tones.
- c. Synthetic noise addition: Noise can be added to clothing images, such as Gaussian or salt-and-pepper noise. The statistical behavior of the present values of intensity in the noise component plays an important role in the distinction between them.
- Gaussian noise: It is a type of random noise that follows a Gaussian distribution. When adding Gaussian noise to an image, random values are sampled from the Gaussian distribution and added to each pixel. This introduces subtle variations in pixel values, simulating the random noise present in real-world images. Because of its mathematical tractability in both the spatial and frequency domains, this noise is used frequently. The PDF of a Gaussian random variable z is defined as follows:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}} \quad -\infty < z < \infty \quad (4-4)$$

- Salt-and-pepper noise: Similarly, it is a type of impulsive noise that randomly replaces some pixels in the image with either the maximum value (salt) or the minimum value (pepper). This results in isolated bright and dark pixels scattered throughout the image; helping in the simulation of sensor or transmission errors that may occur in real-world scenarios. Salt noise refers to the white spots in an image that can occur due to random occurrences in transmission errors or acquisition. Similarly, pepper noise corresponds to the black spots in the image.

The PDF of salt-and-pepper noise is given by

$$p(z) = \begin{cases} P_s & \text{for } z = 2^k - 1 \\ P_p & \text{for } z = 0 \\ 1 - (P_s + P_p) & \text{for } z = V \end{cases} \quad (4-5)$$

Where V is any integer value in the range $0 < V < 2^k - 1$.

P_s represents the probability of pixel being affected by salt noise.

P_p represents the probability of a pixel being affected by pepper noise.

‘k’ represents the total number of bits that are used to represent the values of intensity in an image. Thus, the range of the possible intensity values for that image is $[0, 2^k - 1]$. (Assuming it is an 8-bit image, range is [0-255])

4.3 Data Preprocessing

In the context of our project, which focuses on developing a virtual try-on system, the dataset preprocessing steps have been tailored to suit the specific requirements of the task.

4.3.1 Data Integration and Aggregation

We have collected the datasets from VTON source and classified them into proper hierarchy with alignment of the person image and the image of the cloth. Aggregation of the clothing data to a consistent format, ensuring compatibility and standardization across different sources is necessary for the uniform inputs.

4.3.2 Data Cleaning

We have checked for and addressed some inconsistencies or errors in the data, such as incorrect labels or inconsistent formatting. We have then applied filters as well as smoothing techniques to remove high-frequency noise in clothing images. Some filters include:

- a. Gaussian filter: This applies a weighted average to each pixel based on its neighborhood. It reduces high-frequency noise while preserving the overall structure and edges of the image.
- b. Median filter: The median filter replaces each pixel value with the median value of its neighboring pixels. It is effective in removing impulse noise or salt-and-pepper noise in the image.
- c. Bilateral Filter: The bilateral filter preserves edges while smoothing the image by considering both spatial and intensity differences. It reduces noise while preserving important image details.

- d. Mean Filter: The mean filter replaces each pixel value with the **average value** of its neighboring pixels. It is a simple and commonly used smoothing technique but may not preserve edges as effectively as other filters.
- e. Non-local Means Filter: The non-local means filter compares the similarity between patches of pixels across the entire image and performs weighted averaging. It effectively removes noise while preserving fine details.

4.4 Person Representation

The overall notation or vector obtained from this module represent the prominent semantics of a target person. This has been accomplished by different sub-modules like key-points detection, segmentation of body parts containing both Gray and RGB mapping.

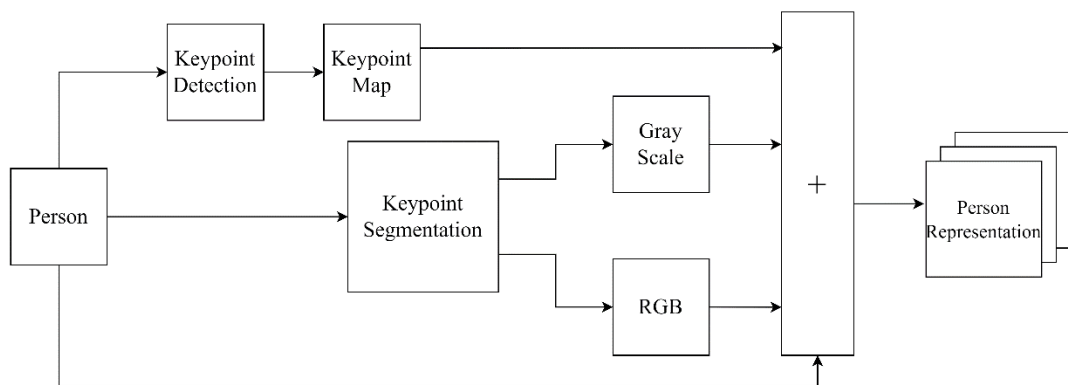


Figure 4-2 Block Diagram of Person Representation

For the purpose of key-point detection in an image of a person, we have utilized OpenPose. OpenPose [10] is a human pose detection library that has the capability to jointly detect the human body and position of body joints in the image. Since the images used in our project has the resolution of 192x256 pixels, we have implemented OpenPose of version 1 that detects major keypoints for the pose of the person in the image.

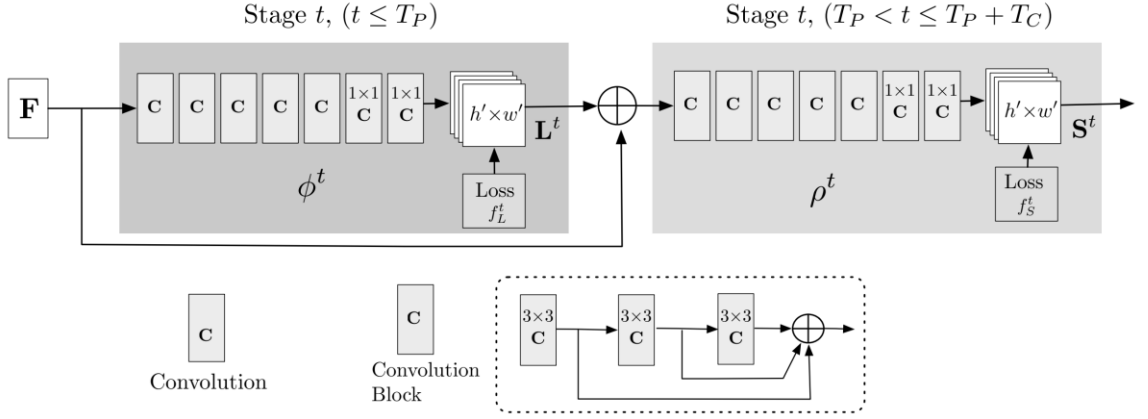


Figure 4-3: Architecture of OpenPose

OpenPose utilizes a novel approach called Part Affinity Fields (PAFs) to detect and connect body parts which enables robust estimation of the position of body parts and keypoints. Its architecture employs a multi-stage Convolutional Neural Network (CNN) to perform detection and association of the body keypoints. It repeatedly enables prediction of affinity fields in the image which then encode the part-to-part association along with the detection confidence maps.

The feedforward network of the architecture predicts a set of 2D confidence maps for specific locations of body parts. Along with it, it also predicts a set of 2D vector fields of part affinity fields (PAFs). The parted pairs of image locations are then matched by parsing confidence maps and PAFs that in return gives the set of 2D keypoints for the person in the image. The keypoints can be utilized to draw over the image for visual representation or the JSON output of the points can be utilized as network parameters as is done in our project.

OpenPose's methodology revolves around a bottom-up strategy, wherein it detects and associates body keypoints without relying on explicit person detection. This approach allows it to handle occlusions and crowded scenes more effectively, providing robustness in estimating poses of multiple individuals simultaneously.

4.4.1 Body parts Segmentation

The process of segmenting the target person's body into high quality segments is essential to figure out the existing garment location of the person's body. Various

architecture and models can be used for this task. For the segmentation of the person and generation of the image mask, we used Deeplabv3. DeepLabv3 is an advanced Convolutional Neural Network (CNN) model developed by Google for semantic segmentation. It represents an incremental improvement over earlier versions (v1 and v2) of DeepLab and achieves superior performance. In contrast to its predecessors, DeepLabv3 no longer relies on the "DenseCRF" (Dense Conditional Random Fields), which is a non-trainable post-processing module for accuracy enhancement. It is trained in a single step without the need for additional post-processing. DeepLabv3 utilizes the ResNet models as the backbone of its architecture. Atrous Convolution, also known as "dilated convolutions," is a modification of the standard convolution operation that introduces gaps or dilation rates between kernel elements. This allows the network to have a larger receptive field without increasing the number of parameters. ASPP (Atrous Spatial Pyramid Pooling) is an extension of atrous convolutions that involves applying multiple atrous convolutions with different dilation rates in parallel. Each atrous convolution branch captures features at a different scale, allowing the network to gather multi-scale contextual information effectively. ASPP is particularly useful for semantic segmentation tasks, where accurately identifying objects of different sizes in an image is crucial.

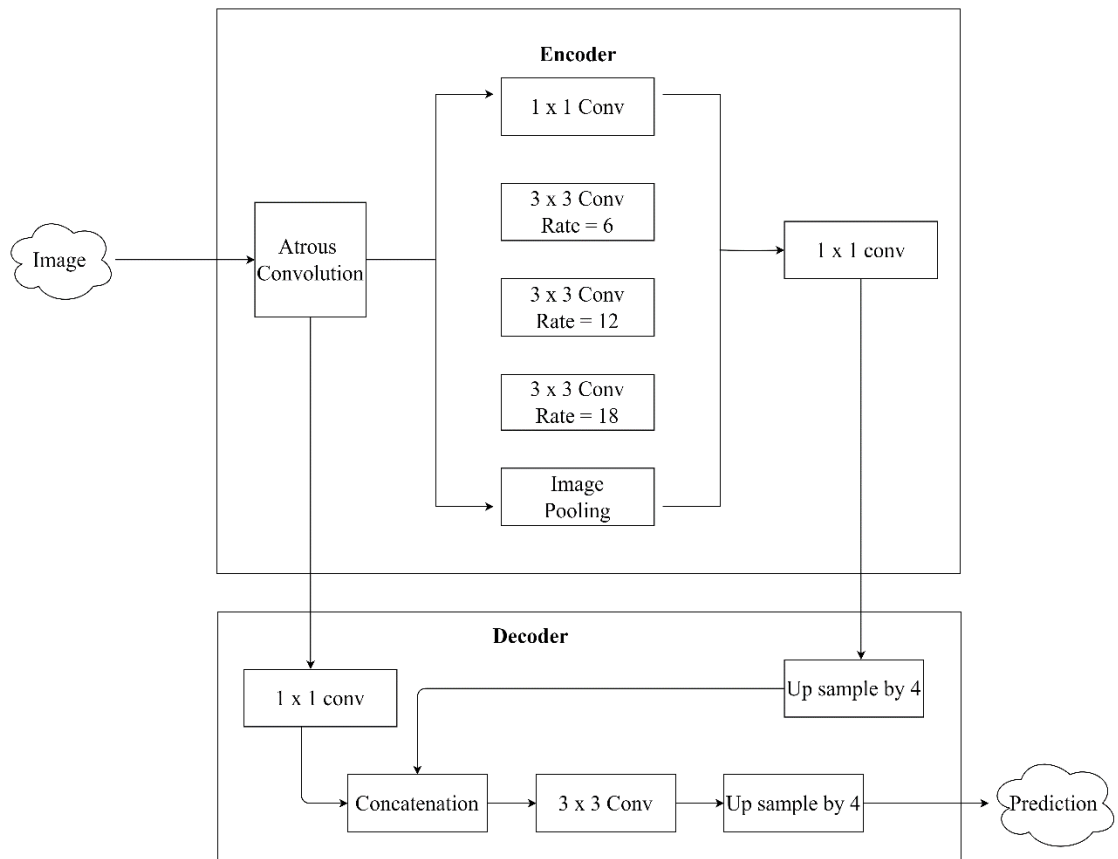


Figure 4-4 Architecture of DeepLab-V3+

In segmentation tasks, down sampling causes the feature resolution to decrease as we move deeper into the network, causing the loss of spatial information. Deeper layers in CNNs have **larger receptive fields, making them better at detecting larger objects** but compromising the ability to identify smaller objects. As a result, the network faces difficulty in simultaneously capturing objects of different scales, leading to potential inaccuracies in segmentation predictions, especially for smaller-sized objects. By introducing atrous rates for convolution, DeepLabv3 better understands "where in the image?" objects are located. ASPP extracts features at multiple rates of dilation, allowing the network to capture information from various receptive fields, covering both large and small objects. By fusing these multi-scale features, DeepLabv3 gains a better understanding of objects of different sizes, enhancing the segmentation performance, even when objects appear at various scales within the image.

The atrous convolution is given by

$$y[i] = \sum_{k=1}^K x[i + r \cdot k]w[k] \quad (4-6)$$

Where y = output feature map

x = input feature map

w = Convolution filter

r = atrous rate (=1 for standard convolution)

DeepLabv3+ further introduces the encoder-decoder like structure, where the DeepLabv3 (or encoder) captures the rich semantic information, and the next module **decoder captures the sharp recovery of boundary**. The major advantage of using this encoder-decoder-like model is faster computation and that spatial information is captured much better.

4.4.2 Person Representation Aggregation

Finally, the obtained key-points from first sub-module along with the gray and RGB segmentation map from second sub-module are **concatenated** to gain the semantic representation of a target person. The quality of this description has high influence over the generation of clothing mask, warped cloth and final output eventually. The outcome from this overall module is notated as ‘P’ referred to person representation in further modules.

4.5 Layout Matcher

From the obtained person representation ‘P’ and a reference garment ‘G’, this module **generates a cloth mask for reference cloth** whose outline synchronizes well with a target person. Firstly, **P and G matrix are parsed to the encoder block** after which respective entangled vectors will be generated to preserve individual essence of the person and garment. Then both vectors are combined to form a general tangled representation which later can be used generate the required result.

We then fed the entangled vector to the generative auto-encoder block. The intermediate latent vector after passing through the encoder is crucial to gain the high-

quality outcome that matches with the cloth mask. L1/L2 loss is utilized to capture the deflection from the ground truth of out generated matrix.

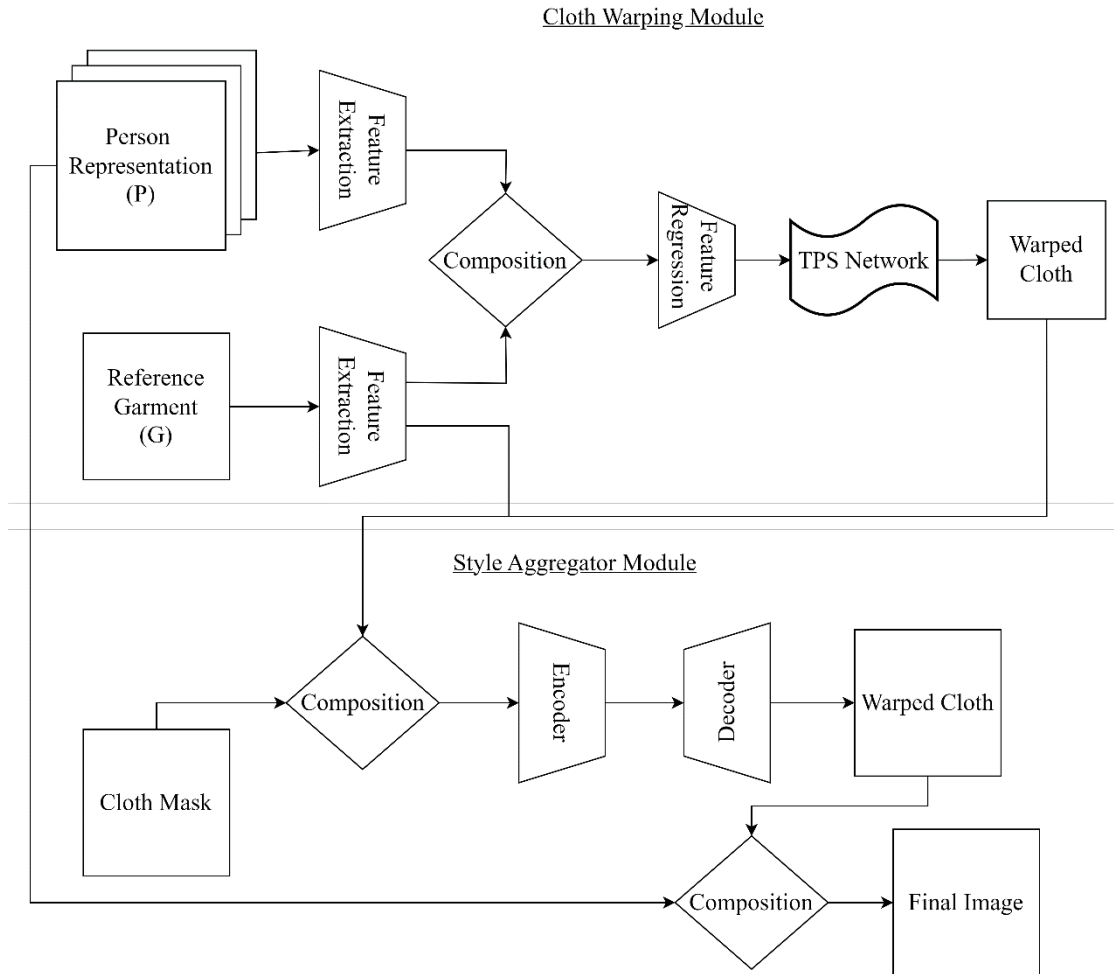


Figure 4-5 Block Diagram for Cloth Warping and Style Aggregator Module

4.6 Geometric Transformer

The Cloth Warping Module (CWM) is a crucial component for fitting the clothes with person representation. Its main purpose is to explicitly align the target clothes with the person representation to produce a warped clothes image, which can be further used for virtual try-on synthesis.

4.6.1 Feature Extraction

CWM begins by extracting high-level features from both the input person image (I) and the target clothes image 'C'. Two separate feature extraction networks are used for this

purpose. The person feature extraction network, denoted as "extractionA" takes the person image (I) as input and generates feature maps that capture the person's body shape, pose, and other relevant information. The clothes feature extraction network, denoted as "extractionB" takes the clothes image 'G' as input and generates feature maps that represent the characteristics of the clothes.

4.6.2 Feature Normalization

After feature extraction, the L2 normalization (FeatureL2Norm) is applied to the feature maps of both the person and the cloth. L2 normalization is a process that scales each feature vector to have a unit Euclidean length, which helps in reducing feature discrepancies and improving matching accuracy.

4.6.3 Feature Correlation

Once the normalized feature maps of the person and clothes are obtained, the next step is to calculate their correlation. The correlation operation is performed to find the similarities between the feature maps of the person and the clothes. It measures how each pixel in one feature map relates to the corresponding pixels in the other feature map. The output of the correlation process is a correlation tensor, which provides information about how well the person and clothes features match at each spatial location.

4.6.4 Feature Regression

The correlation tensor is then fed into the feature regression network to predict the spatial transformation parameters (θ). The spatial transformation parameters represent the geometric deformation required to align the clothes with the person's body shape and pose. The regression network learns to estimate these parameters by taking advantage of the correlation information between the person and clothes features.

4.6.5 TPS Network

TPS [11] stands for Thin Plate Spline that is a spline interpolation method that is used to fit a smooth function to a set of data points. A spline is a piecewise polynomial function that is defined by a set of control points. The TPS transformation is a non-

linear interpolation method, which means that it is able to fit a smooth function to a set of data points even if the data points are not evenly spaced.

Mathematically, TPS transformation is defined by the following equation:

$$f(x, y) = \sum_{i=0}^n w_i * \log(|x - q_i| * |y - r_i|) \quad (4-7)$$

Where, $f(x, y)$ represents the value of the warping field at point (x, y) .

w_i denotes the weights associated with the control points.

q_i and r_i denote the coordinates of the control points.

$|x - q_i|$ represents the distance between the point (x, y) and the control point q_i

$y - r_i \vee$ represents the distance between the point (x, y) and the control point r_i .

The objective of TPS is to determine the optimal weights w_i such that the smoothness of the resulting curve is maximized while maintaining the consistency with the given control points. The weights are obtained by minimizing a cost function, which is defined as follows:

$$J = \sum_{i=1}^n w_i * |f(q_i, r_i) - 1|^2 \quad (4-8)$$

Where J denotes the cost function.

$f(q_i, r_i)$ represents the value of the warping field at the control point (q_i, r_i) .

$|f(q_i, r_i) - 1|$ signifies the error between the warping field's value at the control point and 1.

For implementing the mathematics of TPS transformation, control points and the translation value is required. The fundamental steps are given as follows:

1. Computation of kernel matrix K

Given a set of control points, the kernel matrix (K) is constructed, where each element (K_{ij}) represents the pairwise Euclidean distance between two control points (i and j). This kernel matrix K forms the core of the TPS algorithm and serves as a crucial component in calculating the warping weights.

2. Computation of K-inverse

This step is vital in computation of the weights for the warping points effectively. It simply means the inverse of the K matrix computed above.

3. Computation of weights for non-linear deformation

Using the inverse kernel matrix K_{inv} , calculation of the weights associated with the non-linear part of the grid warping is implemented. These weights determine how each control point contributes to the non-linear deformation of the grid. The objective is to achieve local smoothness while preserving the original shape of the grid.

4. Computation of weights for linear deformation

Similarly, the above computed matrix is used to compute the weights for the affine part of the grid warping. These weights represent an affine transformation that applies uniformly to the entire grid. The affine component ensures global alignment between the original and warped grids.

5. Grid Warping with TPS

With the warping weights for both linear and non-linear deformation calculated, the objective is to transform the grid. For each control point, combined non-linear and affine transformations are applied to determine the corresponding point in the warped grid. This process ensures that the grid is smoothly deformed to align with the desired configuration.

The non-linear and affine transformations both are equally important in Thin Plate Spline Transformation. Non-linear deformation allows individual points in the grid to move independently, facilitating local changes. This ability to stretch, squeeze, or bend the grid at different points is particularly valuable when dealing with complex shapes or images with varying curvatures. The main objective of non-linear deformation is to achieve local smoothness while preserving the original shape of the grid. The weights for the non-linear deformation signify the magnitude of deformation for each control point.

Unlike non-linear deformation, affine transformation involves a more straightforward deformation of the entire grid. It applies the same transformation uniformly to all points, ensuring that the entire grid moves cohesively. Common types of affine transformations include rotation, scaling, and translation. The weights associated with the affine transformation represent the same transformation for each control point that can be applied uniformly to all the points in the grid.

The key distinction between non-linear deformation and affine transformation lies in their scope and purpose. Non-linear deformation allows for localized adjustments, enabling the grid to adapt to intricate shapes and contours. It preserves the fine details of the grid but may become computationally intensive for large grids. In contrast, affine transformation provides an efficient and straightforward way to deform the entire grid uniformly. It ensures that the grid moves as a cohesive unit, preserving the overall alignment and structure. By incorporating both components, the TPS algorithm achieves the best of both worlds. It allows for **fine-grained adjustments** at specific points (non-linear deformation) while simultaneously preserving **global alignment** and structure (affine transformation). This combination ensures that the warped grid smoothly adapts to the desired configuration, capturing both local and global changes.

Overall, the Cloth Warping Module enables the method to explicitly align the target clothes with the person representation. By warping the clothes image based on the learned parameters, it reduces the spatial misalignment between the clothes and the person, which is a common challenge in virtual try-on systems. The warped clothes image can then be further used in the Style Aggregator Module to synthesize the final virtual try-on result by fusing it with the rendered person image.

4.7 Style Aggregator Module

The Style Aggregator Module (SAM) is a critical component designed to seamlessly merge the warped clothing representation with the target person's body image, resulting in a realistic and visually convincing virtual try-on experience. Cloth Warping Module is responsible for creating the warped cloth based on the pose of the person and utilizing the thin plate spline transformation for the proper computation of surface areas of the clothes and preserve the characteristics. With the warped cloth created which is aligned properly with the representation of the person, it can be utilized in this part for proper fusion in the target person's image.

The approach of directly passing the warped cloth to the target representation can be an easier method of achieving the objective. However, occlusion of the body parts along with different looks around the boundary regions pose an issue for efficient final result. There is an implementation of an encoder-decoder structure of a network to match the passed inputs (warped cloth and target representation) for the proper output which is fitting the cloth into the target person. This network architecture is similar U-Net architecture but the lack of proper alignment of the cloth and the person in terms of body parts boundary and the distinct clothing regions deem this method as a failure as well. Thus, both of these structures are combined in the system for achieving the best alignment of the warped cloth over the person.

4.7.1 UNET

This architecture serves as a pivotal component in our pipeline, enabling the generation of high-quality output. It comprises multiple layers, each responsible for different aspects of the synthesis process. The architecture is structured to accommodate various input and output specifications, making it versatile for a range of image synthesis tasks.

Skip Connection Blocks are incorporated within the UNet [12] for establishing the connections between down sampling and up sampling paths for the seamless transfer of information and features. When information flows through these skip connections, it ensures that fine-grained, local details from the input image are preserved and carried forward in the synthesis process. This enables the generator to effectively capture and preserve both global as well as local features in the synthesized images. Convolutional

layers are strategically placed to capture and encode both local and global features. The convolutional operations are adept at recognizing the local patterns which are vital for preserving fine details in the output. Simultaneously, the deeper layers of the network focus on capturing more global features, like the overall structure and context of objects in the image. Employment of batch normalization as well as activation functions allow the network to stabilize and efficiently enhance the learning process. This gives a composition mask (M) and a rendered person (Ir) which when fused together can give the final output image (Io).

4.7.2 VGG19

The VGG19 [13] architecture is a convolutional neural network (CNN) model renowned for its ability to capture hierarchical features from input images. For the purpose of ensuring the synthesized output (Io) closely aligns with the ground truth, VGG19 acts as a potent visual perception network with the capability to extract hierarchical features from the images. Implementation of VGG Perceptual Loss for the alignment of output with ground truth allows significant improvements at multiple layers of the network in different levels. This flexibility allows the fine-tuning of the loss to specific requirements.

4.8 3D Try-On Module

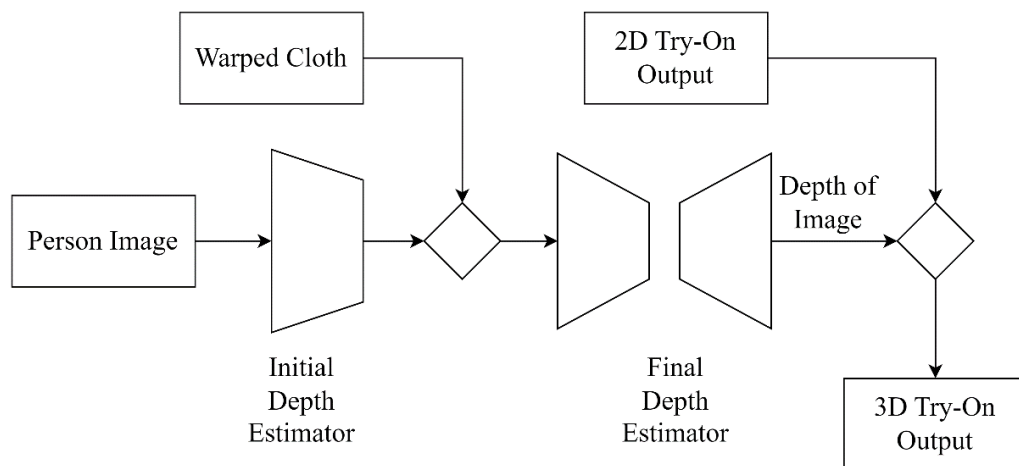


Figure 4-6 Block Diagram for 3D Try-On

The 2D Virtual Try on provides faster and efficient solution to modify the target garments onto a person but it lacks realistic representation. Most 3D models provide

realistic view but are computationally expensive. The datasets for most 3D approaches requires annotated images which is a difficult task. Therefore, the intuitive solution lies somewhere in between. 2D images contain RGB information but lacks information about the depth (unless captured from a depth-sensing camera). Proper estimation of the depth in an image can lead to manageable and efficient 3D reconstruction.

4.8.1 3D Depth Estimation from 2D Images

It is far-fetched to know the depth from a 2D image that has no other information except the RGB values. However, the major methods for the estimation of the depth can be as follows:

4.8.1.1 Binocular Disparity

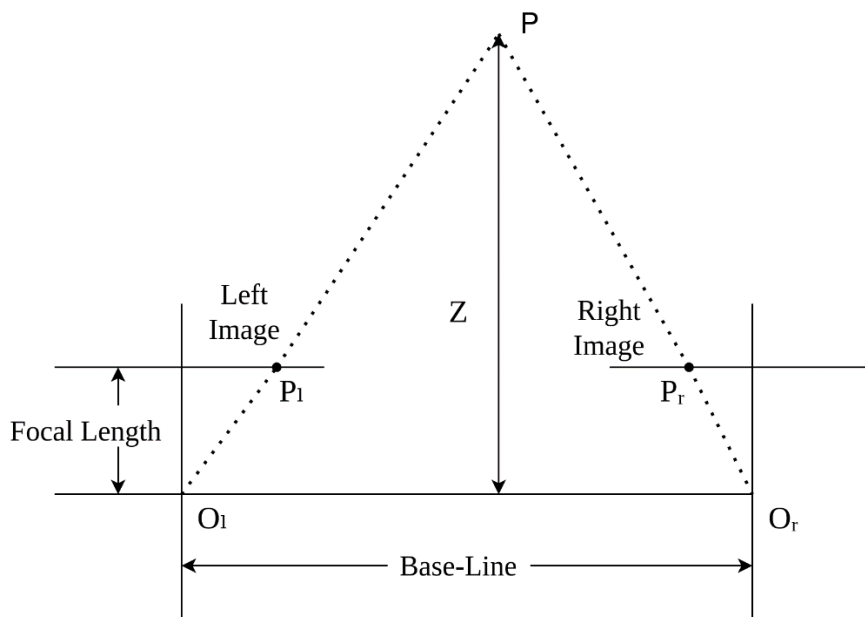


Figure 4-7: Depth Information Extraction from Binocular Disparity

Binocular disparity relies on the principle of stereopsis, which is the brain's ability to perceive depth based on the different views from each eye, thereby mimicking human visual system. The key challenge to this approach is that it may require **two images** of a same scene, and that the calculation is sensitive to calibration errors. The steps involved in the calculation of depth from binocular disparity can be shown with the flowchart as:

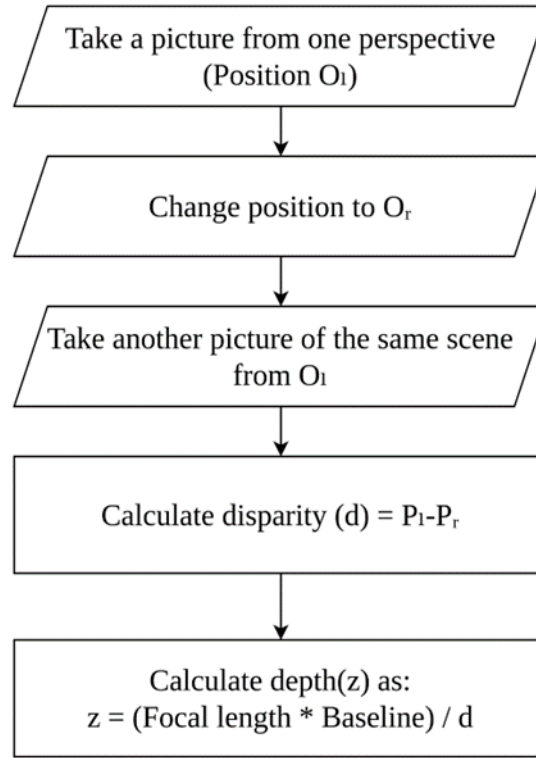


Figure 4-8: Flowchart for Calculation of Depth from Binocular Disparity

4.8.1.2 Photometric Stereo

The primary motivation behind photometric stereo is to infer surface orientation and depth information by analyzing variations in brightness across these images. The method relies on the Lambertian reflection model, which states that the observed radiance of a surface is directly proportional to the cosine of the angle between the incident light and the surface normal. For a Lambertian surface, the output radiance is expressed as

$$L_r = \rho * L_i \cdot N \quad (4-9)$$

where L_r is the observed radiance, ρ is the albedo (reflectance) of the surface, L_i is the incident radiance, and N is the surface normal. In photometric stereo, we need to capture multiple images of an object illuminated from different directions while keeping the viewing direction constant. By analyzing how the brightness of each pixel varies across these images, one can obtain valuable information about the surface normal at each point on the object.

To calculate the depth, first we need to collect images from different lighting condition keeping the viewing direction constant. The illumination from the acquired images is then used to estimate the surface normal at each pixel. The estimated normal gives us the normal map which represents the surface orientation at each pixel. From this normal map, we can estimate the depth of the surface by solving linear equations derived from the dot product of normal vectors and depth vectors.

4.8.2 Sobel Operation

Estimation of depth usually relies on identifying the edges or boundaries between objects in an image. For the estimation of depth of the person or the cloth in the image, Sobel operation helps in detecting those boundaries by highlighting the areas that have significant change in their intensity. The major implementation of Sobel is to estimate the gradients in both horizontal as well as vertical direction. These gradients allow us to identify specific areas where the depth changes abruptly that corresponds to the boundaries or edges in the image.

Even though a colour image has 3 channels (blue, green and red), only one channel is needed for the operation of Sobel. Similar to most of the edge detection algorithms, the passed image should be converted to grayscale image. For an edge detector mask, the sum of the overall number in the mask should be zero. It is because, if an image with no edges is taken, it contains an image with constant pixel value. And if that image is convolved with a mask, the result should be zero.

Now, for the implementation of Sobel operator, a single pixel of an image is selected at a time and neighbourhood operation is applied. This process manipulates the value of the particular pixel based on the value of its neighbours. Usually, 8 neighbours are considered for Sobel operation.

Once we have considered 8 neighbours and a pixel, we multiply the pixel and its neighbours with the following two matrices:

$$\text{y-direction kernel} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \text{ and } \text{x-direction kernel} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}.$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A \quad (4-10)$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \quad (4-11)$$

Where A represents the 3x3 matrix of the pixel being processed with its neighbours,

“*” represents convolution operation.

‘y-direction kernel’ is responsible for approximating intensity change in the x-direction (horizontal) and similarly, ‘x-direction kernel’ is responsible for approximating intensity change in the y-direction (vertical). It particularly means it approximates the gradient of the intensity values. The overall gradient of both direction is combined as follows:

$$G = \sqrt{G_x^2 + G_y^2} \quad (4-12)$$

Which is the edge magnitude of gradient at pixel (x, y). For the direction of the edge, the following operation is carried out:

$$\theta = \tan^{-1} \frac{G_y}{G_x} \quad (4-13)$$

Where θ represents the direction of the gradient.

Computing gradients in both x and y direction yields gradient approximations in the x and y direction at each pixel. The pixels that have large value of gradients are likely part of an edge in the image. Finally, the gradients are compared to a threshold value and the ultimate sobel image is constructed.

4.8.3 Initial depth estimation

The depth is estimated initially by a depth decoder, which takes in the Person Image and corresponding sobel image as an input and tries to generate front and back depth. The background should be removed for the image of the person before feeding it to the decoder. To capture depth in the third dimension, we represent the information via front

and back depth map. The loss involved is **L1 loss**, computed between the predicted and the ground truth of the depth for both front and back depths. The loss is represented as:

$$L_z = |D_f^i - D_f^{gt}|_1 + |D_b^i - D_b^{gt}|_1 \quad (4-14)$$

Where D_f^i and is initial depth estimated for foreground.

D_b^i is initial depth estimated for background.

D_f^{gt} is ground truth for depth for foreground.

D_b^{gt} is ground truth for depth for background.

4.8.4 Final Depth Estimation

Warped cloth is needed to further improve the depth estimated because there occurs a change in brightness at certain region of pixels after the cloth is warped. The need for the multiple depth estimation is to strengthen the accuracy of the depth of an image. As it is tough to estimate depth from a static 2D image without further information such as change in camera position or change in illumination in the same picture, accurate depth information for corresponding image of a person is vital. The **change in the pixel values after warping serves as a further information** required to better estimate the depth.

For this purpose, we compute multiple losses, one of which is the depth loss. It is the Log of L1 loss which is computed as:

$$L_{depth} = \frac{1}{n} \sum_{i=1}^n \ln (\epsilon_i + 1) \quad (4-15)$$

where ϵ_i represent the L1 loss between the predicted and ground truth of the front and back depth.

The log function is used to obtain intricate local details. Since **two very close points are penalized** by the loss function, it helps us obtain better estimation of the depth.

Another such loss used to refine the depth is Grad loss, which is **Log of gradient** taken as L1 losses in both x and y direction. The L1 loss is taken for each layer in depth map separately, after which sobel operator is applied with respect to both dimensions. The intermediate result at each direction is log transformed so as to penalize the near-by points more heavily. Finally, the total grad loss is derived by summation of each log transformed result over each layer of depth map. This stage gives us the final depth, which is used in the generation of 3D images in the later stage.

$$L_{grad} = \frac{1}{n} \sum_{i=1}^n \ln(\nabla_x(\epsilon_i + 1)) + \ln(\nabla_y(\epsilon_i + 1)) \quad (4-16)$$

Where ∇ denotes the Sobel operator

ϵ_i denotes L1 loss of the i-th depth point

n denotes total number of front/back depth map points.

4.8.5 3D Try-On

The output of the 2D Try-On in combination with the final depth obtained from the previous module is used to obtain the 3D Try on output. Since the output of 2D module has R, G, B information of the foreground only, there is no 3D information available. The depth information for each pixel is required which is obtained separately from the Depth Estimation Module. The depth information can be used in the **reconstruction** of 3D mesh, which can be easily rendered in a 2D interface to produce a realistic output. For this purpose, the colour values of the back can be obtained by digital inpainting methods. These methods help in the reconstruction of the damaged portions of an image, or in propagating the colour information into the certain region of our choice. In order to fill a certain area with the known colours, the gradients in the known pixels have to be preserved, and the colour propagation should be smooth.

One such method of painting is the Fast-Marching Method, which is a **heuristic** painting method where the nearest unknown pixel is replaced by the weighted and normalized values of known neighbouring pixels. This algorithm gradually **goes inwards** from the boundary region. The Fast-Marching method, a numerical method, is used to propagate

the known image information and move to the next neighbouring pixel. This technique is much simpler and faster than other traditional PDE (Partial Differential Equation) based methods such as Total Variational (TV) model. The first order approximation for a pixel 'p' in a region given the known pixel 'q' can be established. Then the pixel value in that region can be computed by the weighted sum of all the first order approximation from the neighbouring points, which can be normalized to obtain the valid pixel value.

Navier Stokes inpainting method is another method based on Fluid Dynamics. The Newtonian flow in fluids obey the Navier-Stokes equation, and the motivation is drawn from the fact that there are parallels between the Fluid Dynamics and Image Inpainting. The stream function, which quantifies the quantity of a fluid moving across a line governed by the Navier-Stokes equation, is analogous to the intensity of the Image. Likewise, the fluid velocity is analogous to the direction of Isophotes, where the Isophote means the curve joining the pixels with equal image intensity. The vorticity, which is the whirling motion of a fluid, is analogous to the smoothness in inpainting. Similarly, the viscosity of the fluid is analogous to the Anisotropic diffusion in inpainting. Our goal is thus to solve the Navier-Stokes equation in the specified region where painting is to be done, by continuing the Isophotes and matching the gradients at the boundary. These inpainting methods thus paint the back side of a person, giving us the colour information in the back.

We now have obtained three different information, RGB colour values of front from the 2D Try On, front and back depth from the Depth Estimation Module, and the RGB color values of the back by inpainting methods. This much information is sufficient for 3D rendering of the image. In order to do so, the 3D file format has to be obtained which can be displayed. A file format called Polygon File Format (PLY) is a format to describe graphical objects as a collection of polygons and store them. It has two sub-formats: ASCII and Binary. A typical PLY file format consists of a header, vertex list, face list and edge list, terminated by the end of the header. The properties for each list are defined. For example, the properties of a vertex list are X, Y, Z, and maybe some colours. The actual vertex list is written after the end of the header line, giving us the ply file.

When the point cloud file is read, the vertex list and the face list is read and rendered as a 3D representation. Usually, the surface reconstruction methods are employed to obtain a 3D mesh from the unstructured point clouds. Various methods such as Alpha shapes, Ball pivoting, or Poisson reconstruction can be used for obtaining a triangular mesh from the points. It is important to note that for the plane reconstruction to join the points, the normals of the points must be calculated first. For that, a plane is fitted first to the local neighbourhood of each point. After the normal of the points are obtained, the surface can be simply reconstructed perpendicular to the normals. To simplify all the hassle, a third-party software can instead be used to render the point clouds, where all the normal computation and surface reconstruction can be done.

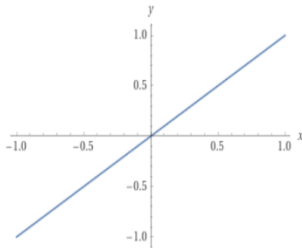
4.9 Model Fundamentals

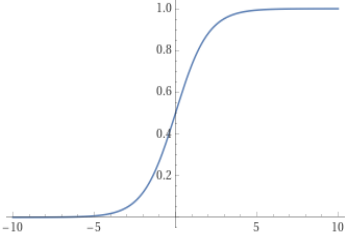
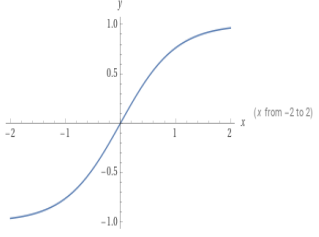
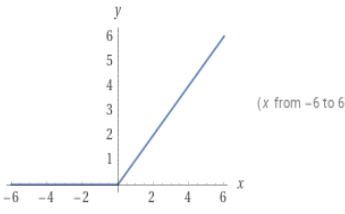
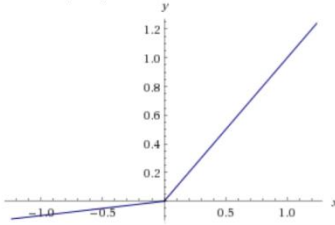
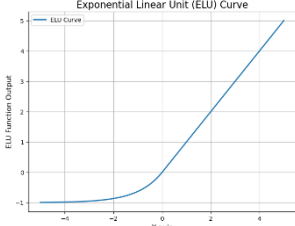
4.9.1 Activation Functions

Activation functions are the functions which help in obtaining the output of the node in the defined range for the given set of inputs. The major goal of the activation function is to introduce non-linearity and normalize the output into a range. There are linear and non-linear activation functions. Since pure linear functions don't have predefined range and don't help in capturing the complexity, we usually only talk about Non-linear activation functions.

Some of the activation functions are compared in the table below, where the graph, activation function equations and their derivatives are provided.

Table 4-1: Table of Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$

Sigmoid		$f(x) = \frac{1}{1 + e^{-x}}$	$y'(x) = \frac{e^x}{(e^x + 1)^2}$
Tanh		$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$\frac{d}{dx} \tanh(x) = \text{sech}^2(x)$
ReLU		$f(x) = \max(0, x)$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky ReLU		$f(x) = \max(0.1x, x)$	$f'(x) = g(x) = \begin{cases} 1, & x \geq 0 \\ 0.01, & x < 0 \end{cases}$
ELU		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

4.9.2 Loss Functions

4.9.2.1 L1 Loss

L1 loss, also known as Mean Absolute Error (MAE), is a loss function commonly used in various machine learning tasks, especially in regression problems. It measures the average absolute difference between the predicted values and the ground truth. The L1 loss is less sensitive to outliers compared to other loss functions like L2 loss.

Mathematically, for a set of N data points, L1 loss is given as:

$$L1\ Loss = \frac{\sum |y_{true} - y_{pred}|}{N} \quad (4-17)$$

4.9.2.2 GIC (Geometric Invariant Cloning) Loss

The GIC loss aims to enforce geometric consistency between the warped cloth and the person's pose during the virtual try-on process. Geometric consistency ensures that the clothing items align well with the person's body shape and pose after being transferred onto the target image.

This loss is implemented by Cloth Warping Module. Firstly, displacement field tensor is generated as 'grid'. It represents how each pixel in the clothing image should be warped and aligned to fit the person's body pose. If 'Gx' and 'Gy' are the x and y components of the grid, further components of the grid for neighbouring elements ('Gxcenter', 'Gxleft', and 'Gxright' as X components, and 'Gycenter', 'Gyup', 'Gydown' as Y components), are defined in the height and width dimensions.

GIC loss is then computed by measuring the similarity between neighbouring elements in both X and Y directions of the grid.

$$dt_{left} = |Gx_{left} - Gx_{center}| \quad (4-18)$$

$$dt_{right} = |Gx_{right} - Gx_{center}| \quad (4-19)$$

$$dt_{up} = |Gy_{up} - Gy_{center}| \quad (4-20)$$

$$dt_{down} = |Gy_{down} - Gy_{center}| \quad (4-21)$$

$$Loss_{GIC} = \sum (|dt_{left} - dt_{right}| + |dt_{up} - dt_{down}|) \quad (4-22)$$

The GIC loss encourages the model to generate accurate displacement fields that preserve key structural features of the clothing and align them with the corresponding body regions, resulting in geometrically consistent and visually appealing virtual try-on results.

4.9.3 Optimizer

4.9.3.1 Adam Optimizer

Adam [14], known as Adaptive Moment Estimation, is an optimization algorithm widely used for training artificial neural networks whose primary objective is to efficiently **update a model's parameters during training to minimize the loss function** and enhance overall performance. Adam distinguishes itself from other optimization methods by combining the strengths of two prominent techniques: Momentum and Root Mean Square Propagation (RMSprop).

In essence, Adam operates as follows:

1. Momentum

Momentum is a technique that accelerates the learning process by **considering past gradient information**. It introduces a momentum term that adds a fraction of the previous gradient to the current gradient. This allows the optimizer to gain momentum in relevant directions, facilitating faster convergence and helping to escape potential local minima.

2. RMSprop

Adam also incorporates the concept of RMSprop, which addresses the limitation of fixed learning rates for all parameters. RMSprop **adapts the learning rate** for each parameter by dividing it by the Root Mean Square (RMS) of recent gradients. This adaptivity allows for quicker convergence and better handling of varying learning rates across parameters.

3. Adaptive Learning Rates

The key innovation of Adam lies in its adaptive learning rates, calculated individually for each parameter. It computes the **first and second moments of the gradients** and uses them to update the learning rates accordingly. By adapting learning rates during training, Adam ensures that each parameter receives a suitable rate, catering to its specific requirements.

Adam combines momentum and RMSprop in an adaptive manner, dynamically adjusting learning rates for different parameters. This adaptive approach leads to faster convergence and improved performance compared to traditional optimization methods.

The Adam algorithm revolves around the central idea of calculating exponentially moving averages of gradients and also the squared gradients. The name Adam is derived from adaptive moment estimation. The general procedure of the Adam optimizer is given as follows.

- 1) Update biased first moment estimate

$$m_t = \gamma_1 m_{t-1} + (1 - \gamma_1) g_t \quad (4-23)$$

- 2) Update biased second moment estimate

$$v_t = \gamma_2 v_{t-1} + (1 - \gamma_2) g_t^2 \quad (4-24)$$

- 3) Compute bias corrected first moment estimate

$$\hat{m}_t = \frac{m_t}{(1 - \gamma_1^t)} \quad (4-25)$$

- 4) Compute bias corrected second moment estimate

$$\hat{v}_t = \frac{v_t}{(1 - \gamma_2^t)} \quad (4-26)$$

- 5) Update parameters

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (4-27)$$

where m and v are moving average, g is the gradient on current mini batch and gammas are the hyperparameters of the algorithm. The values of the hyperparameters are usually kept 0.9.

4.10 Full Stack Integration

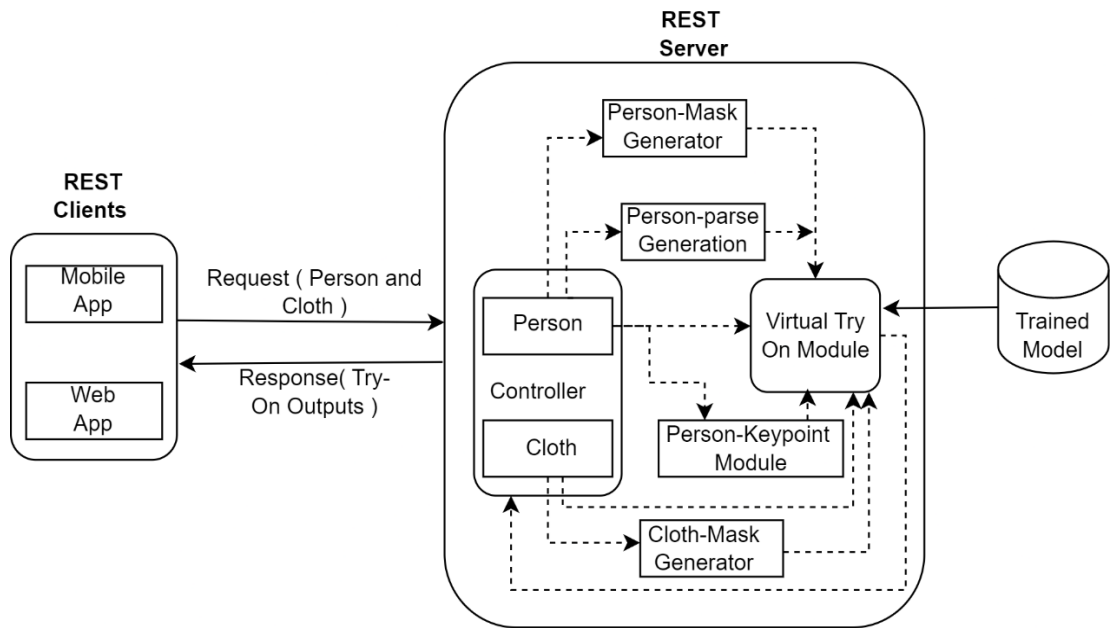


Figure 4-9 Rest API Architecture

REST (Representational State Transfer) API architecture is a set of guidelines and best practices for creating scalable web services. It is an architectural style for designing networked applications, rather than a standard or protocol. The request received at the back-end endpoint is **unpacked into person and cloth images**. The controller is then responsible for distributing the person's image and cloth's image to multiple sub-modules. The person's image is sent to the person-mask generator, person-parse, person key-point, and final virtual Style Aggregator Modules, while the cloth's image is sent to the cloth-mask generator and Style Aggregator Module. The intermediate results from the **sub-modules are then passed to the Style Aggregator Module** to generate the final output of our model. The Style Aggregator Module loads different trained models, and inference is performed on the inputs obtained from multiple intermediate results from the sub-modules. The final result is obtained by running the Cloth Warping Module and Style Aggregator Module trained models and is sent to the front-end as a response. For the working of the web application, **trained models are loaded into the Django framework** and processed inputs are passed. The obtained outputs are displayed via HTML templates.

5 IMPLEMENTATION DETAILS

5.1 Dataset Collection

The VTON dataset contains persons' images showing individuals wearing various clothing items, with their full or upper body captured from multiple angles and under different lighting conditions. For full body datasets, similar sets of images with full body were considered to be trained on the same architecture as upper body datasets. To create the dataset, they initially crawled around 19,000 pairs of frontal-view woman images along with corresponding images of the top 2 clothing items. After removing noisy images without parsing results, the images left were 16,253 image pairs. The resolution of the images in the VTON dataset is 192x256 pixels. For the full body dataset for 3D image, the resolution is 320x512.



Figure 5-1: Dataset Sample

5.2 Clothing Images

The clothing images in the VTON dataset are standalone representations of clothing items, not featuring any individuals wearing them. These images are specifically captured to facilitate the generation of virtual try-on outcomes. They share the same dimensions as the person images, ensuring compatibility and accurate alignment during the virtual try-on process. This consistent sizing enables seamless overlaying of clothing items onto person images, allowing researchers and developers to experiment with various combinations of garments and achieve realistic virtual try-on results.

Moreover, the number of clothing images in both the test and train datasets corresponds to the number of person images. This balanced distribution ensures fairness in the dataset, providing equal opportunities for training and evaluating virtual try-on algorithms and models. The uniform image sizing and equitable data division enable reliable comparisons among different methods, contributing to the advancement and evaluation of virtual try-on systems.



Figure 5-2: Sample of Clothing Image

5.3 Dataset Preprocessing

5.3.1 Binary Masks

A binary mask is a type of image mask that consists of **only two pixel values**, typically representing a binary decision or segmentation. In this context, "binary" refers to the two possible states: on or off, foreground or background, or presence or absence of an object or region. To convert an image to its mask, we first convert the given image into **grayscale image**. To convert an RGB image to grayscale in Python, you can use the following formula:

$$\text{Grayscale} = 0.2989 * \text{Red} + 0.5870 * \text{Green} + 0.1140 * \text{Blue} \quad (5-1)$$

After converting the RGB image to grayscale, we can create a mask by setting the pixels that correspond to the cloth segment to keep (foreground) to 255 (white), and setting the rest of the pixels to 0 (black).

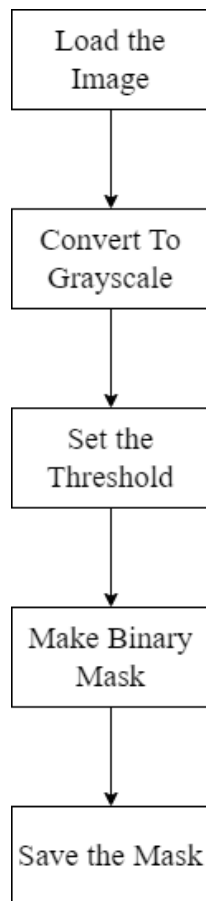


Figure 5-3: Flowchart of Binary Mask Creation

The inclusion of segmentation masks, such as cloth mask and person mask, enhances the capability to manipulate and overlay clothing items during virtual try-on. These masks **enable precise alignment** and realistic visualization of virtual try-on outcomes by providing detailed information about clothing and person regions.

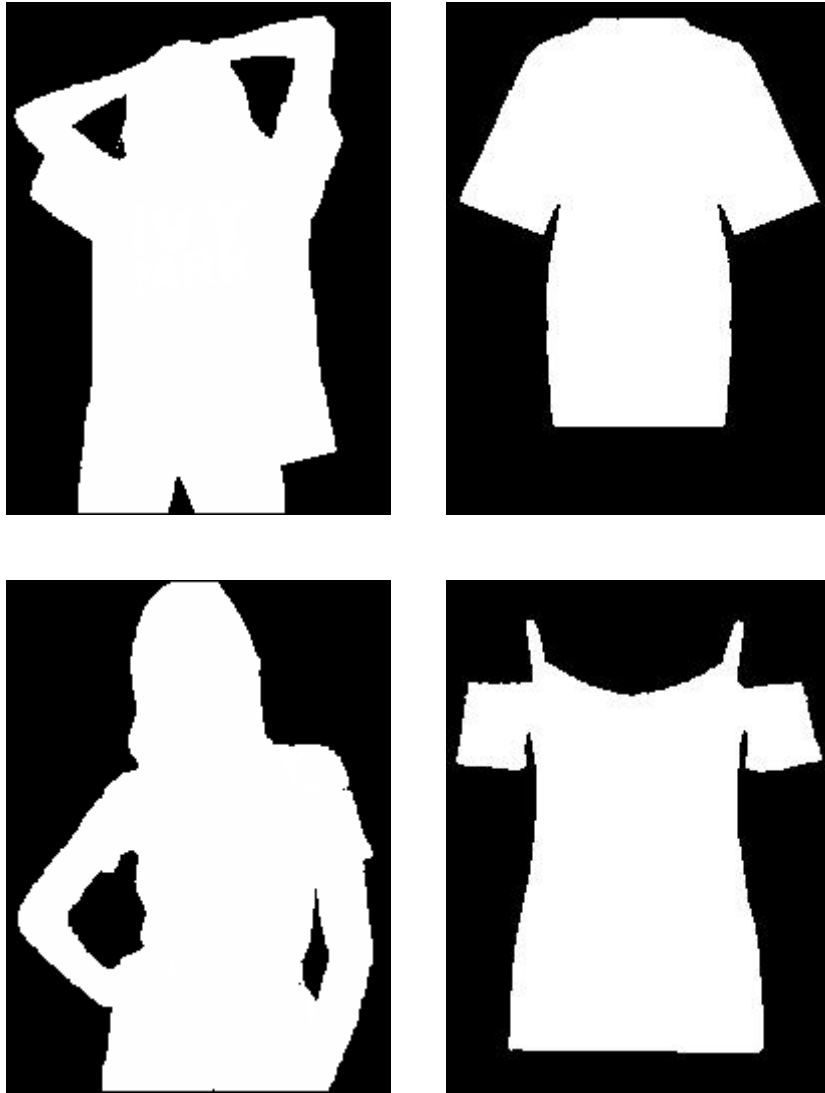


Figure 5-4: Binary mask of Person and Cloth

The cloth masks in the dataset are binary masks that distinguish the clothing items from the background. They accurately outline the boundaries of the clothing, enabling a clear separation between the foreground (clothing) and background pixels.

Similarly, the person masks are binary masks that highlight the person in the image, differentiating them from the background. These masks precisely identify and isolate the person's region.

5.4 Dataset Preprocessing

5.4.1 Dataset Augmentation

The images of person as well as the clothing may not be optimal for the purpose of the system to be designed. Thus, multiple processes are carried out in finessing and producing suitable form of datasets. Some of the basic geometric functions in images are implemented to the images of person and clothes.

a. Rotation

Rotation augmentation involves rotating an image by a certain angle around its center. The rotation angle is usually measured in degrees. The formula to perform a rotation around the center of the image is given by:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (5-2)$$

Where θ represents the rotation angle in radians or degrees.

x and y represent the translation parameters to adjust centre of rotation.

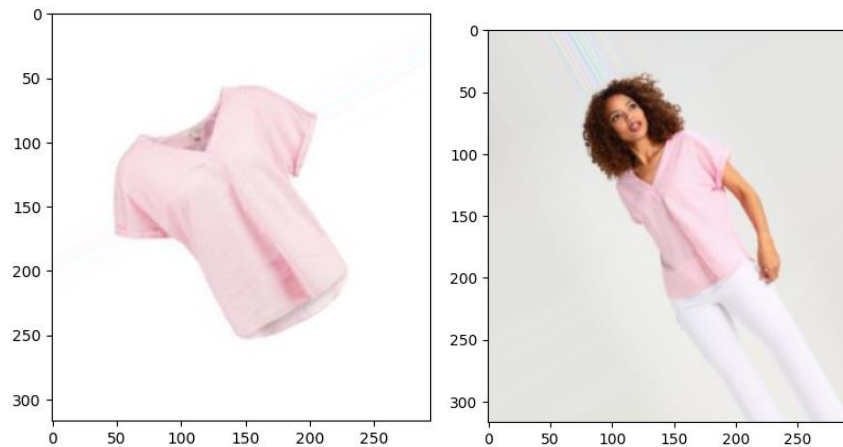


Figure 5-5: Rotation of Cloth and Person Representation (+30 degrees)

By applying rotation augmentation, the model becomes more invariant to the orientation of objects in the images and can recognize them regardless of their orientation.

b. Translation

Translation augmentation shifts an image along the x and y axes by specified amounts. The translation matrix for this transformation is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (5-3)$$

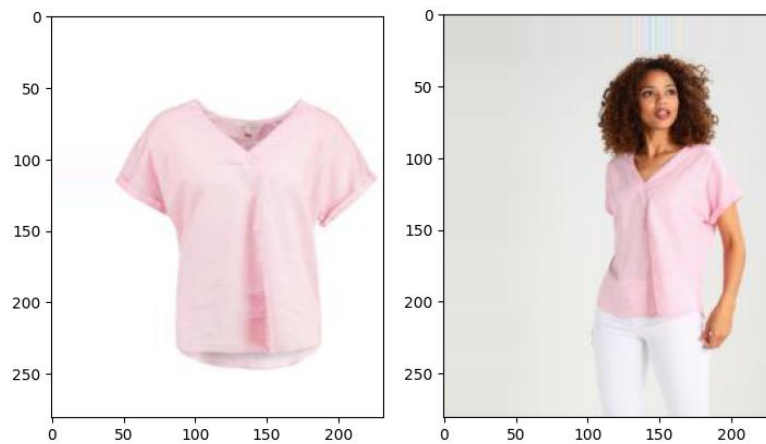


Figure 5-6: Translation of Cloth and Person Representation

This transformation introduces positional variance in the dataset, making the model more tolerant to minor translations of objects. It also helps the model understand that objects can appear in different positions in the image.

c. Scaling

Scaling augmentation resizes an image by applying a scale factor along the x and y axes. The scaling formula is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (5-4)$$

Where s_x and s_y are the scaling factors in X and Y axes respectively.



Figure 5-7: Scaling of Cloth Representation



Figure 5-8: Scaling of Person Representation

This transformation improves the model's ability to handle objects at different scales and distances from the camera, enhancing its scale invariance.

d. Shearing

Shearing augmentation skews the image along the x or y-axis. The shearing matrix for the x-axis and y-axis shearing, respectively, are:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \text{shear_factor}_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (5-5)$$

And,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \text{shear_factor}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (5-6)$$

Where, shear_factor represents the amount of shearing applied along the X or Y-axis.

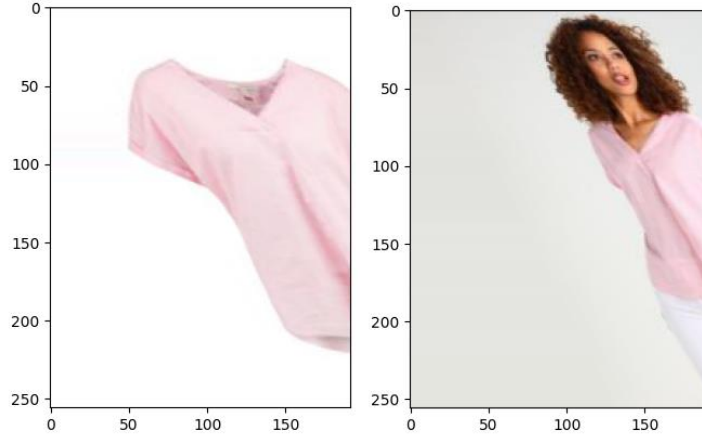


Figure 5-9: Shearing of Cloth and Person Representation

By introducing perspective variations through shearing, the model becomes more robust to real-world scenarios where perspective distortion might occur.

e. Horizontal and Vertical Flipping

Flipping augmentation involves flipping an image horizontally, vertically, or both. The flipping formula is straightforward. To flip an image horizontally, we reverse the order of its columns. To flip an image vertically, we reverse the order of its rows. The horizontal and vertical flipping augmentations make the model more invariant to the orientation of objects and help it learn symmetrical patterns. Some images may have been flipped as some cameras store the images in the flipped version of the original one.

In horizontal flipping, we reverse the order of columns in the image matrix. For an image with width W and height H, the formula is as follows:

$$Image_{flipped}[i, j] = Image_{original}[i, W - 1 - j] \quad (5-7)$$

In vertical flipping, we reverse the order of rows in the image matrix. For an image with width W and height H, the formula is as follows:

$$Image_{flipped}[i, j] = Image_{original}[H - 1 - i, j] \quad (5-8)$$

Where, $Image_{flipped}[i, j]$ is the pixel value at row i and column j in the flipped image.

$Image_{original}[i, j]$ is the pixel value at row i and column j in the original image.

H is the height and W is the width of the image.



Figure 5-10: Flipping of Cloth and Person Representation

Thus, these techniques of geometric augmentation are carried out in handling the datasets to fit into a common standard and bring uniformity in them.

5.4.2 Photometric Augmentation

Some of the photometric augmenting techniques implemented while processing the dataset, along with their theoretical background and formulae are given as follows.

a) Brightness Adjustment

Brightness adjustment involves adding or subtracting a constant value to all pixel values in the image. This technique effectively changes the overall brightness without affecting the color information. The brightness adjustment is performed in the HSV color space, where the V (Value) channel represents the brightness.

For each pixel value V in the image, the brightness adjustment is given by:

$$V' = V + value \quad (5-9)$$

where V' is the new pixel value after brightness adjustment and "value" is the brightness adjustment factor.

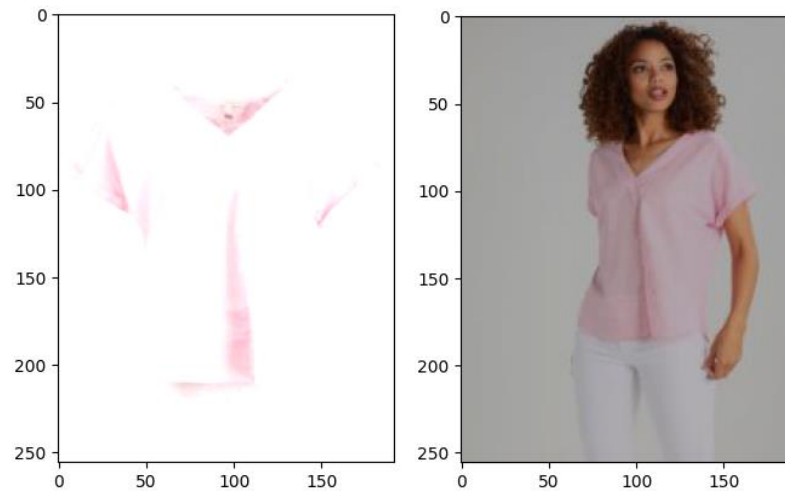


Figure 5-11: Brightness Adjustment of Cloth and Person Representation

b) Contrast Adjustment

Contrast adjustment aims to enhance or reduce the difference between the pixel intensities in an image. It is achieved by scaling the pixel values using a contrast factor.

For each pixel value V in the image, the contrast adjustment is given by:

$$V' = (V - mean) * alpha + mean \quad (5-10)$$

where V' is the new pixel value after contrast adjustment, "alpha" is the contrast factor, and "mean" is the mean pixel value of the image.

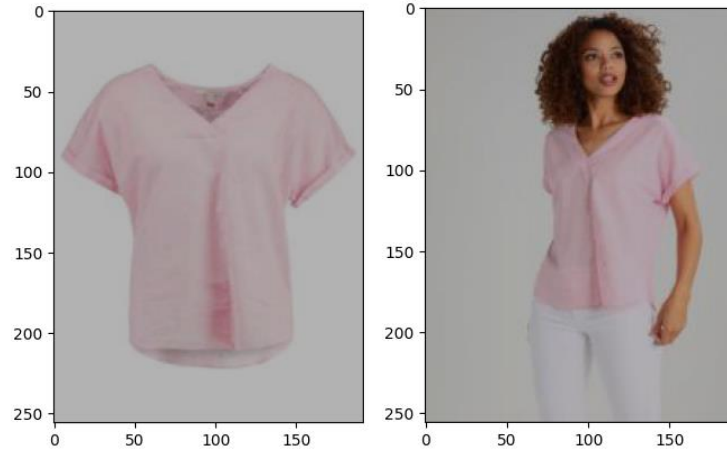


Figure 5-12: Color Adjustment of Cloth and Person Representation

c) Saturation Adjustment

Saturation adjustment alters the intensity of colors in the image while maintaining the brightness. It is achieved by **scaling the saturation channel in the HSV color space.**

For each pixel value S in the saturation channel of the image in the HSV color space, the saturation adjustment is given by:

$$S' = S * \text{scale_factor} \quad (5-11)$$

where S' is the new saturation value after adjustment, and "scale_factor" is the adjustment factor.

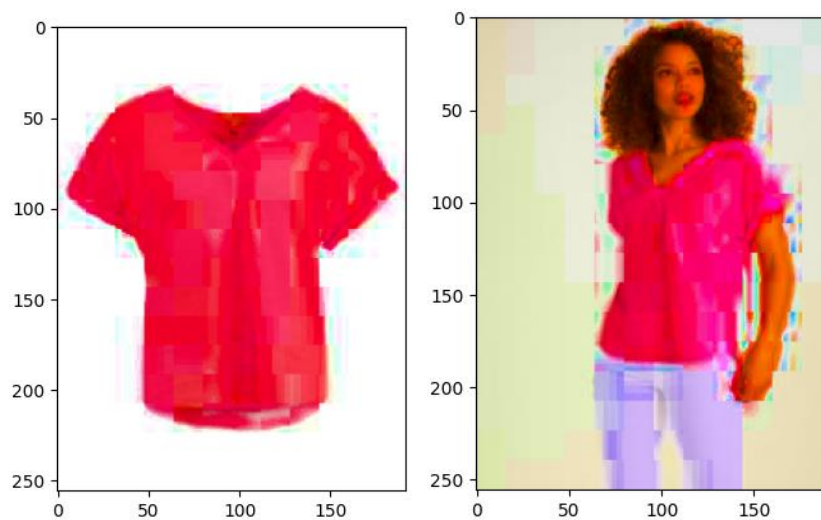


Figure 5-13: Saturation Adjustment of Cloth and Person Representation

d) Hue Adjustment

Hue adjustment involves **shifting the color** hue of the image without affecting brightness or saturation. It changes the overall color tone.

For each pixel value H in the hue channel of the image in the HSV color space, the hue

$$H' = (H + \text{hue_shift}) \bmod 360 \quad (5-12)$$

where H' is the new hue value after adjustment, 'hue_shift' is the shift value in degrees, and the result is wrapped around 360 to keep it within the valid range.

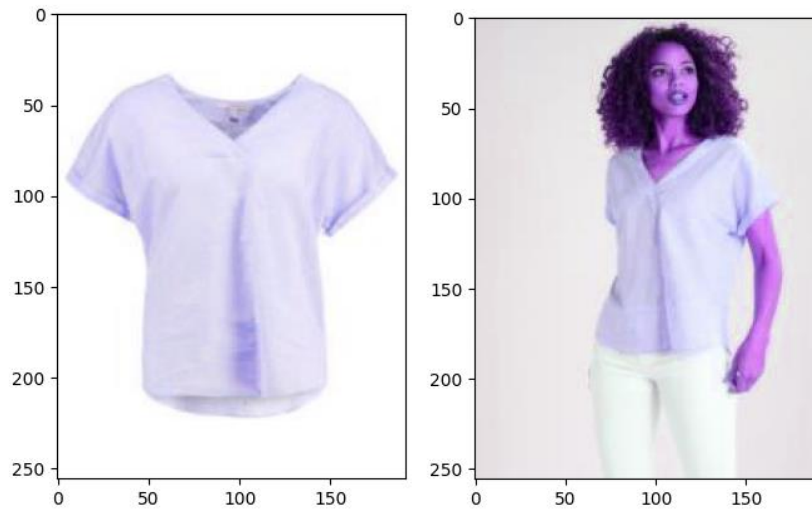


Figure 5-14: Hue Adjustment of Cloth and Person Representation

e) Gaussian Noise

Gaussian noise is a type of additive noise that introduces random variations in pixel values, simulating real-world noise present in images.

Gaussian noise is added to each pixel value V in the image using a random sample from a Gaussian distribution with mean "mean" and standard deviation "std":

$$V' = V + \text{Gaussian}(\text{mean}, \text{std}) \quad (5-13)$$

where V' is the new pixel value after adding Gaussian noise.



Figure 5-15: Gaussian Noise Addition in Cloth and Person Representation

f) Salt and Pepper Noise

Salt and pepper noise randomly replace some pixels in the image with either the **maximum (white) or minimum (black)** pixel value. For each pixel value V in the image, it is replaced by 255 (white) with probability "prob" for salt noise, or by 0 (black) with probability "prob" for pepper noise.

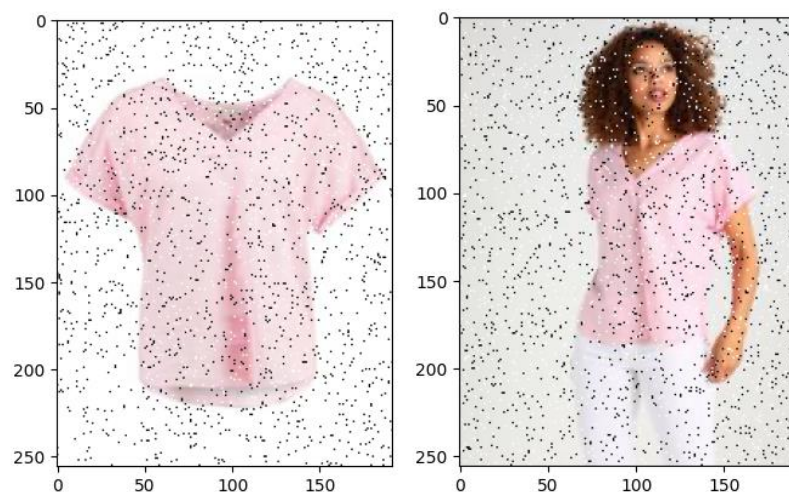


Figure 5-16: Salt and Paper Noise Addition in Cloth and Person Representation

By applying photometric augmentation techniques during the training process, we can augment the dataset and increase its diversity, leading to improved model

generalization and performance in various real-world scenarios. The choice and combination of these techniques should be carefully considered based on the dataset characteristics and the specific needs of the machine learning task at hand.

5.5 Keypoint Detection using OpenPose

We have implemented OpenPose to detect the keypoints in the image of the person in our VTON dataset. Since we only require the pose information and basic body parts, complex and time-consuming detection models are not required. In a simple manner, we can pass our image to the OpenPose library which returns the image with detected keypoints in them. Presence of a deep CNN allows it to extract high-level features from the preprocessed image and allows the localization of the body joints with produced keypoints. For associating and parsing detected keypoints, there is the implementation of greedy parsing algorithm. It involves a **heuristic-based approach to connect the keypoints** based on the proximity and information extracted from the affinity fields. The iterative process forms the keypoints that are associative and closest in nature, thus in return making them the most compatible pairs of keypoints with associated affinity fields. The final skeletal structure is formed by linking the detected keypoints which ultimately represents the human pose.



Figure 5-17: Demonstration of Key-point Detection with OpenPose

5.6 Implementation of Image Segmentation

The DeepLabV3+ model with ResNet-50 backbone is loaded for the case of Segmentation. A list of labels is defined, representing different classes for 21 classes in the segmentation output. For the given input images residing in a folder, ‘cv2.imread’ is used to load them. The necessary RGB conversion is done as before and resizing is also done to 192x192 size. The image is converted to tensor and normalized using ‘transforms.ToTensor’. The preprocessed image is passed through the DeepLabV3+ model and the model return the segmentation output, where the index of the class with the highest probability is obtained using ‘torch.max’. When the input is passed to the model, logits is obtained as output. The output is a tensor containing the logits for each pixel across all the classes in the segmentation task. In the context of machine learning and deep learning, logits are the raw, unnormalized outputs of a model before they are

transformed into probabilities using a SoftMax function. They are a measure of the model's confidence in its predictions for each class in a classification task or for each category in a multi-class problem.

A mask for human regions is created by checking if the segmentation mask is equal to the required label index. The mask for human regions is resized to match the original image size using `cv2.resize`. The mask is applied to the original image using `cv2.bitwise_and` to obtain the segmented human regions. For the binary mask of the human, a binary mask is created and applied to the original image. The tensor of class indices (segmentation mask) is to be converted to a NumPy array of bytes. The byte format is used because each pixel index is an integer representing the class label, and byte format helps to reduce memory usage.

We can manipulate the segmentation mask to obtain the binary mask as well as the human segmented image of the original image. For the experimentation, we have compared four types of manipulations on the original image. The first type is the segmentation of human regions from background regions. The second type is the color enhanced over the first type of image to distinguish even more between the segmentation classes. The third type is the binary mask image. Note, that for multi-person images, all the people have the same segmentation color in this case, since the goal is to differentiate person class from background. The fourth type segments the different classes within the person classes, thus distinguishing between various human parts and the clothes. Some of the samples are provided below:



Figure 5-18: Image Segmentation with DeepLabV3+

5.7 Implementation of TPS Transformation

The mathematical steps for the transformation were implemented as follows:

1. Construction of Kernel Matrix (K)

$$P_{distance}^2 = (x - P_{i(x)})^2 + (y - P_{i(y)})^2 \quad (5-14)$$

For each control point P_i and each point in grid (x, y). 1 is added to the diagonal elements of $P_{distance}^2$ for the purpose of avoiding any issues that might arise while taking logarithm of zero.

$$K = P_{distance}^2 * \log (P_{distance}^2) \quad (5-15)$$

2. Construction of Extended matrix L

A matrix P of size (N, 3) was created where N is the total number of control points. P consists of the homogeneous coordinates (1, x, y) of the control points. An extended matrix L is formed by concatenating K with P and its transpose $[P^T, Z]$ where Z is a 3x3 zero matrix.

3. Computation of Inverse of L (Li)

A normal matrix inversion of L is computed and stored as Li.

4. Computation of Weights for Non-Linear Portions

Non-linear part of the transformation is extracted as Q_x and Q_y . For the actual control points, the base coordinates ($P_{x_{base}}, P_{y_{base}}$) are added to Q_x and Q_y .

$$W_{x_{NL}} = Li \times Q_x \quad (5-16)$$

$$W_{y_{NL}} = Li \times Q_y \quad (5-17)$$

5. Computation of Weights for Affine Parts

The matrix multiplication of Li and Q_x as well as Q_y separately give the required weights for the affine portions similar to the non-linear portions.

$$W_{x_L} = Li \times Q_x \quad (5-18)$$

$$W_{y_L} = Li \times Q_y \quad (5-19)$$

6. Computation of Distance between Points and Control Points

Distance between coordinates of each point in the grid (x, y) and each control point P_x and P_y is computed as:

$$\Delta x = x - P_x \quad (5-20)$$

$$\Delta y = y - P_y \quad (5-21)$$

7. Computation of squared distance

$$d^2 = \Delta x^2 + \Delta y^2 \quad (5-22)$$

8. Computation of U

$$U = d^2 * \log(d^2) \quad (5-23)$$

Here, U represents the distance between the warped points (grid points after the transformation) and the control points.

9. Computation of Warped Points

$$X' = W_{x_L} + W_{x_{L-1}} * x + W_{x_{L-2}} * y + \text{sum}(W_{x_{NL}} * U) \quad (5-24)$$

$$Y' = W_{y_L} + W_{y_{L-1}} * x + W_{y_{L-2}} * y + \text{sum}(W_{y_{NL}} * U) \quad (5-25)$$

Where, W_{x_L} is the constant term in affine transformation and represents translation component of the affine transformation along the X-axis.

$W_{x_{L-1}} * x$ is the linear term in the affine transformation with respect to the X-coordinate of the control points. It scales the grid vertically based on the points. It allows for a linear deformation of the grid along the Y-axis but controlled by the X coordinates of the control points.

$W_{x_{L-2}} * y$ is similar to it but with respect to Y-coordinate of the control points.

$\text{sum}(W_{x_{NL}} * U)$ is for the non-linear part of the grid warping. It involves element-wise multiplication of non-linear weights and the matrix U and the sum. It provides the non-linear deformation along the X-axis allowing the grid to smoothly bend and reform.

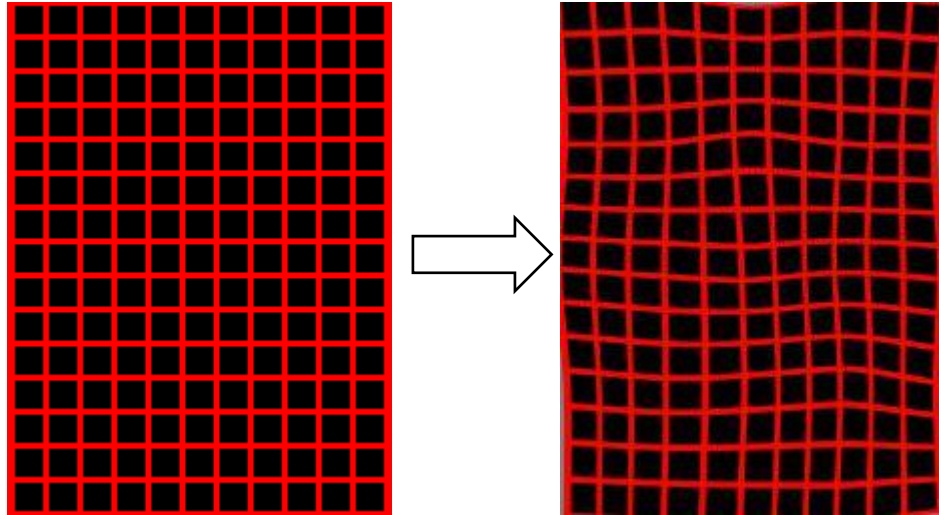


Figure 5-19: Base Grid (left) and Warped Grid (right) from trained TPS model

After progressing the training of the model with networks like feature extractors and correlation networks, a layer of TPS Grid Generation is also appended for generating corresponding grid images to the representations passing through the model. As the model attempts to recreate the warps and transformation in the images, the grid is also expected to be warped but with implementation of the formulation of TPS.

5.8 Implementation of Cloth Warping Module

The layers that define a Cloth Warping Module (CWM) is implemented using combination of network architectures, which includes several layers for feature extraction, correlation, regression, and grid generation.

5.8.1 Feature-Extraction Layer

The Feature-Extraction layer is responsible for extracting meaningful features from the input images. It consists of ‘ $n_layers + 1$ ’ blocks, each containing a series of layers. The initial Conv2d layer performs a 2D convolution on the input images, extracting low-level features. The number of **output channels** is set to **64** by default. This is followed by a ReLU activation function, introducing non-linearity to the feature maps. The **BatchNorm2d** layer then normalizes the features, which speeds up convergence and stabilizes training. The number of channels in the Conv2d layers increases progressively with each block, allowing the network to capture more high-level features

and information as it goes deeper into the network. If the number of channels exceeds 512, it is clipped to 512 to prevent excessive computational overhead.

5.8.2 Feature L2-Norm Layer

The FeatureL2Norm layer normalizes the feature maps obtained from the Feature-Extraction layer. It calculates the L2 norm of each feature vector, ensuring that all feature vectors have unit length. This normalization process helps in learning robust feature representations, making the model less sensitive to variations in lighting and scale. The normalization is performed by **dividing each element of the feature map by the norm**, which is computed as the square root of the sum of squared elements plus a small epsilon value to avoid division by zero.

5.8.3 Feature Correlation Layer

The Feature-Correlation layer takes the normalized feature maps from two inputs and calculates the correlation tensor between them. The correlation tensor represents the similarity between each pair of features from given inputs. It first reshapes the feature maps to enable matrix multiplication. The correlation tensor is computed by performing a **batch matrix multiplication between the reshaped feature maps**. The resulting tensor has elements representing the correlation between a feature from 'inputA' and all features from 'inputB'. The correlation tensor encodes spatial information about the relationship between the two feature maps.

5.8.4 Feature Regression Layer

The Feature-Regression layer performs regression on the correlation tensor to predict the parameters of the geometric transformation. It consists of several Conv2d layers with batch normalization and ReLU activation functions. These layers process the correlation tensor to extract relevant spatial information. The output of these convolutional layers is then flattened and passed through a fully connected layer with **64 output units, followed by a Tanh** activation function. The Tanh activation function ensures that the predicted transformation parameters are within a range of $[-1, 1]$, providing stability during training and preventing extreme transformations.

Thus, the CWM module performs geometric matching between inputA and inputB by first extracting relevant features using the Feature-Extraction layer. The extracted features are then normalized using the FeatureL2Norm layer, and the correlation tensor is computed using the Feature-Correlation layer. The Feature-Regression layer predicts the geometric transformation parameters based on the correlation tensor, and the ‘AffineGridGen’ and ‘TpsGridGen’ layers generate the transformation grids. These transformation grids are used to align the features from inputA to inputB, enabling the model to learn to match geometrically related image pairs.

5.9 Implementation of Style Aggregator Module

A U-Net generator and its associated building blocks for skip connections were defined, serving as fundamental components of the U-Net architecture. The core of the U-Net was formed by the `UNetGenerator` class, which accepted several key parameters such as the number of input and output channels, the depth of the network, the number of feature channels and whether dropout was used. Each block in the U-Net has convolutional layers, normalization layers (typically Batch Normalization), and optional dropout layers. The forward method of the model was used to pass input data through this constructed U-Net model, and begin the training procedure.

The `UnetSkipConnectionBlock` class defined the individual building blocks that were used within the U-Net. These blocks were responsible for conducting down-sampling and up-sampling operations, and they included skip connections to ensure the preservation of high-resolution information during the up-sampling process. It received parameters such as the number of input and output channels, whether the block was the innermost or outermost in the network, and whether dropout was employed. Depending on its position in the network, each block was constructed with specific layers, including convolutional layers, normalization, activation functions (LeakyReLU or ReLU), and upsampling layers. The forward method of `UnetSkipConnectionBlock` was employed to apply these layers to the input tensor, incorporating skip connections as needed to maintain information flow throughout the network.

The training procedure was carried out with training the model for Style Aggregator Module with required datasets obtained from CWM module. To ensure efficient GPU

usage, the model as well as input to the model were transferred to the GPU format for faster computation. Several loss functions were defined for guiding the training process, including the L1 loss (`criterionL1``), a custom VGG loss and another L1 loss. Additionally, an Adam optimizer was initialized with specific learning rate parameters.

Within each epoch, a series of operations were performed, including obtaining input data from the training loader, sending data to the GPU when necessary, and utilizing the model to generate outputs, including rendered images and composite masks based on the architecture. Activation functions, such as tanh and sigmoid, were applied to certain outputs to process them.

Loss calculations, encompassing L1 loss, VGG loss, and mask loss, were conducted, and these individual losses were aggregated to compute the overall loss. Gradients were reset, and backpropagation was executed to update the model's parameters based on the computed loss. Furthermore, the model checkpoints were saved at regular intervals.

5.10 Sobel Operation Implementation

For applying sobel operation for the image of person as well as cloth, we have implemented the function provided by OpenCV library. `'Cv2.sobel()'` takes in arguments like the image, direction of the operation and the size of the kernel.

Taking 3x3 pixels of an image with consideration of the centered pixel (x, y), it is convoluted with the 3x3 sobel mask and value of gradients denote if the pixel is a part of an edge in the image.



Figure 5-20: Demonstration of Original Image (left), Sobel_X (middle), Sobel_Y (right)

5.11 Depth Estimation

For the purpose of estimation of depth of an image of a person, two stages of depth computation have been implemented.

5.11.1 Initial Estimation

This module is designed to **decode depth information** from the feature maps generated by previous layers of the network. The architecture of this Initial Depth Decoder comprises a series of up-sampling blocks, each incorporating convolutional layers and **instance normalization**.

The first up-sampling block initiates the process by applying a convolutional layer with a kernel size of 3, generating 512 channels. Subsequently, instance normalization is employed to stabilize and accelerate the training process, followed by a Rectified Linear Unit (ReLU) activation function. The second block continues this pattern, further increasing the number of channels to 512, with additional instance normalization and ReLU activation.

The subsequent up-sampling blocks follow a similar structure, utilizing **bilinear up-sampling**, convolutional layers, and instance normalization to progressively refine the spatial resolution of the feature maps. The **final up-sampling block employs a Tanh** activation function to constrain output values within the range of $[-1, 1]$. It is trained

for around 45 epochs using the Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and the learning rate initialized to 0.0002.

5.11.2 Final Depth Estimation

The final depth estimation block is initiated by the **down-sampling phase**, encompassing four layers of convolutional operations designed to progressively reduce spatial dimensions. The initial convolutional layer within each stage has a kernel size of 3, applies padding of 1, and utilizes an Exponential Linear Unit (**ELU**) activation function. Subsequently, an additional convolutional layer, again with a kernel size of 3, is employed, along with ELU activation. Down-sampling is achieved by alternating these convolutional layers with a **2x stride convolution**, effectively reducing the feature map dimensions by half. Instance normalization is applied after the second convolutional layer to stabilize training.

The process is then followed by up-sampling of alternating 4 layers. It involves bilinear interpolation to increase spatial dimensions. Each up-sampling block comprises convolutional layers, ELU activation, and instance normalization. Notably, first two sequential layers involve a 1x1 convolutional layer prior to up-sampling, facilitating the combination of interpolated features with those from the corresponding down-sampling stages. The final layer consists of a 1x1 convolutional layer, ELU activation, and a hyperbolic tangent (**Tanh**) activation function. This configuration ensures that the output is bounded within an appropriate range, aligning with the requirements of depth prediction.

5.12 3D Output Generation

The depth of the image and the output of 2D Try-On is concatenated and the final 3D Try-On output is generated for each image. The depth information is used to convert to world coordinates for both front and back. In painting the back side of an image means to assign RGB values for each pixel in the back side of a person. For this the human parsing segmentation information is used to understand which RGB values to assign to the different parts of a person. The back side is painted using either of the Navier-Stokes-based or Fast-Marching-based methods. The painting is done with OpenCV as:

dest = cv2.inpaint(src, inpaintMask, inpaintRadius, flags)

Here the '*inpaintRadius*' means the neighborhood around a pixel to in-paint. Typically, if the regions to be painted are thin, smaller values produce better results (less blurry). The flags are set as 'INPAINT_NS' (for Navier-Stokes based method) or 'INPAINT_TELEA' (for Fast marching based method) for inpainting the back portion of the image.

Thus, the RGB values were obtained for both front and back of a person. These front and back color values, along with the front and back depth values were used to write a file in Polygon File Format (PLY). The file was written with a header specifying that the file used was ASCII, with the syntax '*format ascii 1.0*'. The vertex element along with face, and edge elements can be defined below the header but for our purpose, only the vertex element was defined. The vertex element further contained six properties X, Y, Z, R, G, and B. All these six values as a list were written after the 'end header' line. This PLY file was opened and visualized using Open3D library of Python. For the ball pivoting the function '*create_from_point_cloud_ball_pivoting*' was used where the vertices were passed as parameters. Similarly, for alpha shapes, '*create_from_point_cloud_ball_pivoting*' was used. The normals for the points were computed using '*estimate_normals*' method. Alternatively, the written PLY file was also viewed using any third-party software, for our purpose, MeshLab as well as Open3d viewer were used to view the 3D mesh, where various smoothing techniques were carried out for better visual results.

6 RESULT AND ANALYSIS

6.1 2D Try-On

The two modules, CWM and SAM, have been trained for 55 epochs by loading about 14221 images. Similarly, these models have been tested on 2032 test images to evaluate their results. First, the training of CWM required us to have cloth, cloth mask, the image of the person, image mask, parsed or segmented image, and finally the pose of the person obtained from the key-points detection. The parsed image with the pose is combinedly referred to as Person representation. The following are the sample inputs to the model:



Figure 6-1 Inputs to CWM model

These inputs were processed by the model and the training was done on batch size of 4, for 55 epochs, giving the outputs as overlaid TPS, cloth on the person mask, warped cloth, warped grid, and the warped mask. The sample outputs of the model are:



Figure 6-2: Warped Cloth and its Mask from trained CWM model



Figure 6-3 Overlay of Warped Cloth over Person's mask and on Person

The events were recorded during the training of the CWM module for 55 epochs. The overall loss recorded during its training is depicted in the figure below:

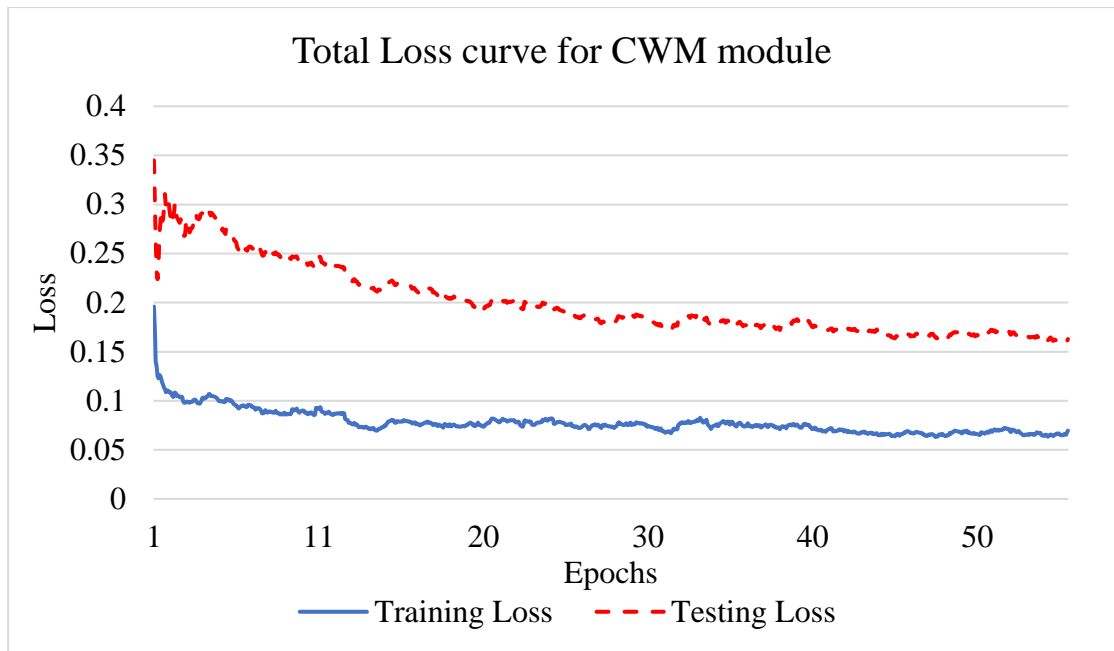


Figure 6-4 Total Loss Curve of CWM Module

The above curves depict the loss during training and testing of Cloth Warping Module over the numbers of epoch. The training loss has stabilized around 15 epochs, which started to fluctuate over the plateau afterwards. However, the loss for test data kept on declining, which indicated that the model was good enough to infer on the new data. The testing loss curve can be illustrated to have been stabilized around 40 epochs with minimal decrease in the loss afterwards. The variance of model also seemed to be low as of fifty-five epochs indicating the marginal difference between training and testing loss.

The cumulative form of both L1 and GIC loss has been represented in above figure, where the GIC loss is computed as the difference in the overlay of warped cloth on the intended person's mask, otherwise known as the distortion of original grid of TPS, because the TPS distortion causes the warped cloth to distort as well. Likewise, the L1 criterion in above loss indicates the Mean Absolute Error (MAE) of generated warp cloth with respect to original parsed cloth from reference person image. The technique was to minimize both the loss to strike a balance between warping cloth and retaining the original image characteristics.

The second module was called Style Aggregator Module (SAM), where the training of this module required the warped cloth and warped mask of the cloth obtained from the previous module as an input. The following are the sample inputs to the model:



Figure 6-5 Warped Cloth and Warped Mask as input to SAM

These inputs were processed by the model and the SAM training was also done on batch size of 4, for 3556 iterations for each epoch, giving the outputs as background isolated mask, person's most likely pose, cloth rendered image and the final refined output of the image. During the inferencing 2,032 items were used. The sample outputs of the trained model are:

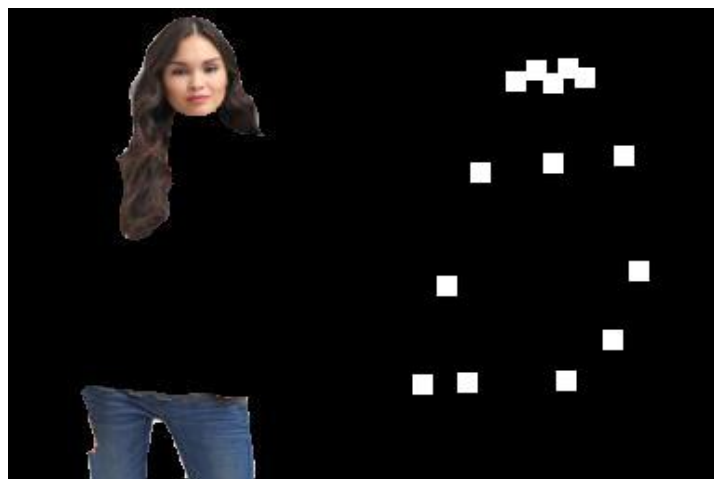


Figure 6-6 Isolated image and Pose



Figure 6-7 Rendered image and the Final refined output

The events during the training of the Style Aggregator Module for 55 epochs were recorded and the loss curve was obtained as:

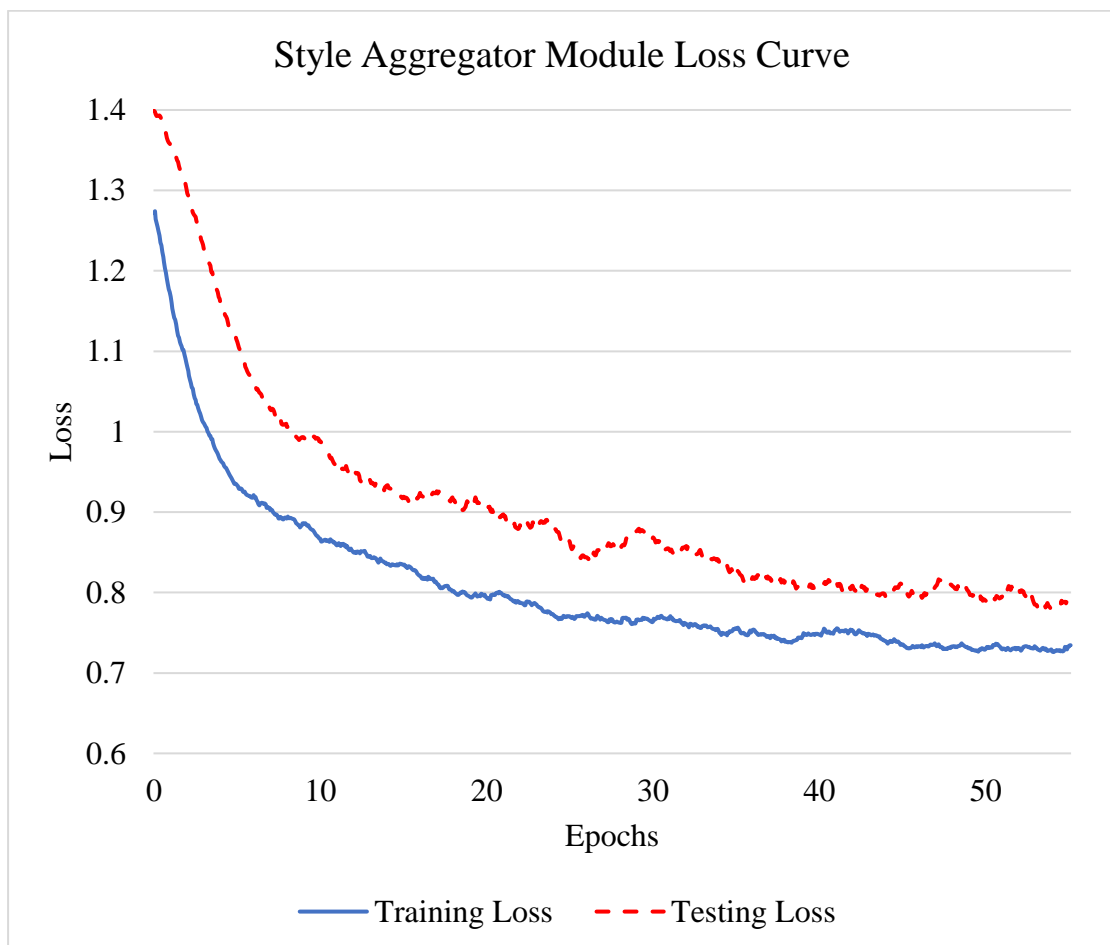


Figure 6-8 Total Loss Curve of SAM Module

The above curves illustrate the loss during training and testing of Style Aggregator Module over the numbers of epoch. Both the training and test loss kept declining in a very slow order. This could be because of the complex model with heavy input parameters as well as number of the iterations under each epoch. However, the loss curve for the test data showed low variance in the model inference, which indicated that the model was good to infer on the new data, based upon the training it received. The overall loss for the training of about 55 epochs can be stated to be low and stabilized towards the end of the training for both training and testing data. Encompassing L1 loss, VGG loss, and mask loss, the overall loss was computed which is depicted in the figure above.

For further experimentation, we also clicked a picture of cloth manually and passed the cloth and its mask to the model. Even with the imperfect capture of the cloth with less clear background, the obtained result was somewhat satisfactory.



Figure 6-9: Demonstration with Custom Cloth

We also demonstrated the system passing a custom image which has front part of the body visible with plain background with ideal hand position and camera angle. In this instance, the target cloth perfectly aligned with the person's body. The results are illustrated as follows:



Figure 6-10 Demonstration with Custom Image

Along with the satisfactory results obtained from Style Aggregator Module, there has been few instances where appropriate results were not obtained. An illustration is given below:



Figure 6-11 Demonstration of Unsatisfactory Result with Unique Pose

In the above picture, the pose of the person is not straight and the image of the person is not of complete upper body either. Since most of the images utilized during the

training has the front part of the body fully visible, the model is unsuccessful in fitting the desired cloth over the unique posed image of the person.

6.2 3D Try-On

For the initiation depth estimation, the depth network compared the predicted depths with the ground truth depths for both front and back portion of the person's body. With continuation of the training of the network and optimization, the following values of the losses were obtained.

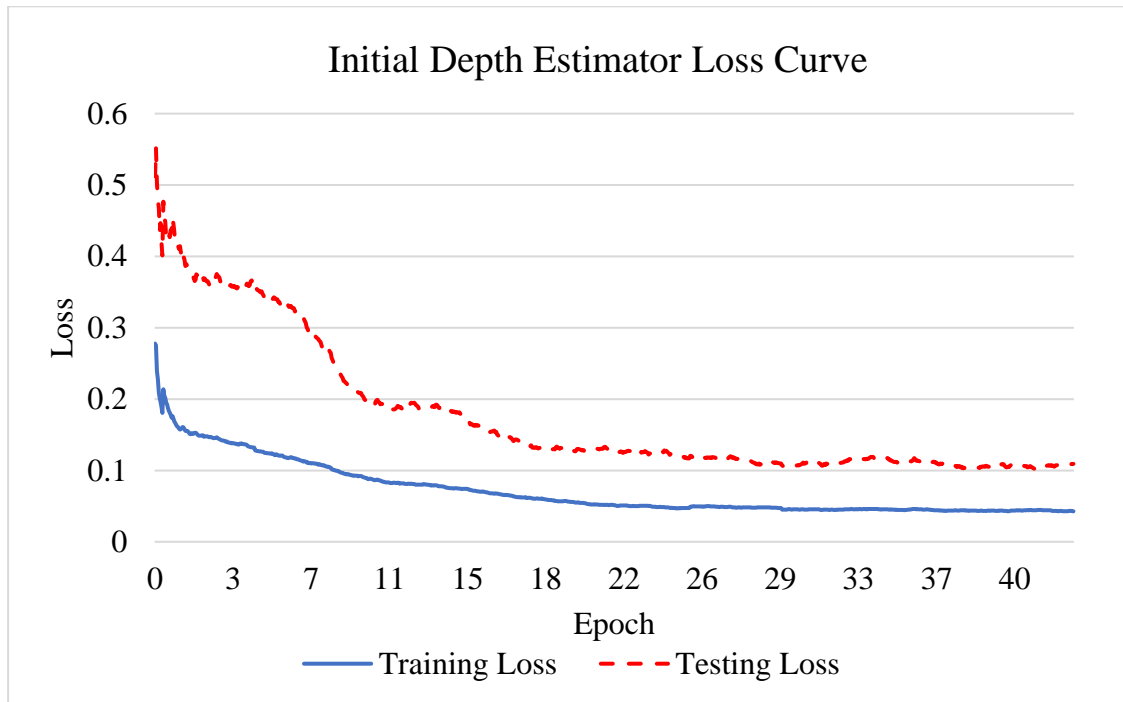


Figure 6-12 Loss Curve for Initial Depth Estimator

The above curves show the training and testing loss over the number of epochs during the experimentation. The testing loss for foreground-depth and background-depth seem to be high at the initial epoch due to lack of enough training. However, after around 18 epochs, the testing loss is minimized nearer to training loss. Since the loss is minimum and the gap between training and testing loss is least, it can be inferred that the model has low bias and low variance.

Now, for further improved depth of the images, the final depth estimation network utilizes available image of the person, sobel images in x and y direction and warped

cloth to generate realistic depth that represents the edges around the person with higher accuracy. The losses obtained during the training is illustrated as follows:

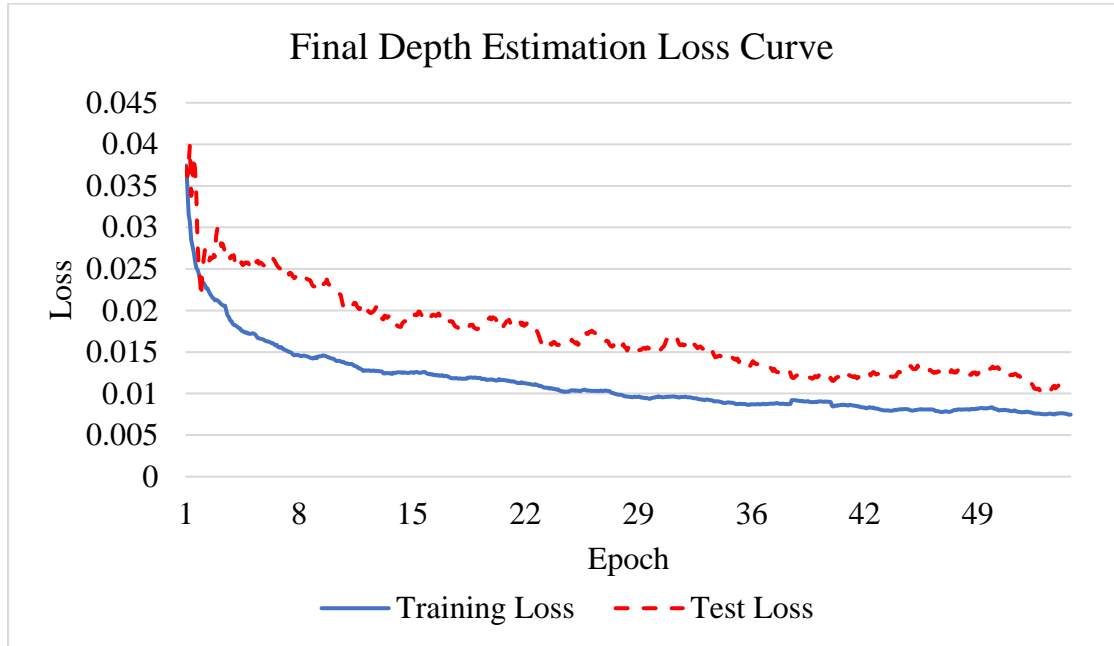


Figure 6-13: Loss Curve of Final Depth Estimation Network

The loss curve of the final depth estimator network shows the loss during training and testing of the network. From the curve, it can be inferred that the loss has been in decreasing with the increment of number of epochs. It is also visible that there is **consistent gap** between training and testing loss throughout the experiment. The depths achieved from this trained model however has been quite satisfactory for the construction of 3D mesh of the person.

Since the obtained PLY format output doesn't support animation sequences, a 3D viewer- MeshLab was used to render models in the PLY format. It even allows rotation and zooming features in the mesh. In the MeshLab itself, a mesh was imported and visualized from different angles. To enhance the generated mesh, smoothing techniques has been used such as adjusting the number of neighbors to estimate normal. The obtained outputs are illustrated as follows:



Figure 6-14 Person, Target Cloth and 2D Try-On Output



Figure 6-15 3D Mesh Output (Front and Back)

The above output shows satisfactory mesh output of the person with try-on output as illustrated above demonstrated with Open3D viewer in Python. However, when some unique pose image is passed for the construction of 3D with computation of depth, output with few errors is obtained as well as shown below:

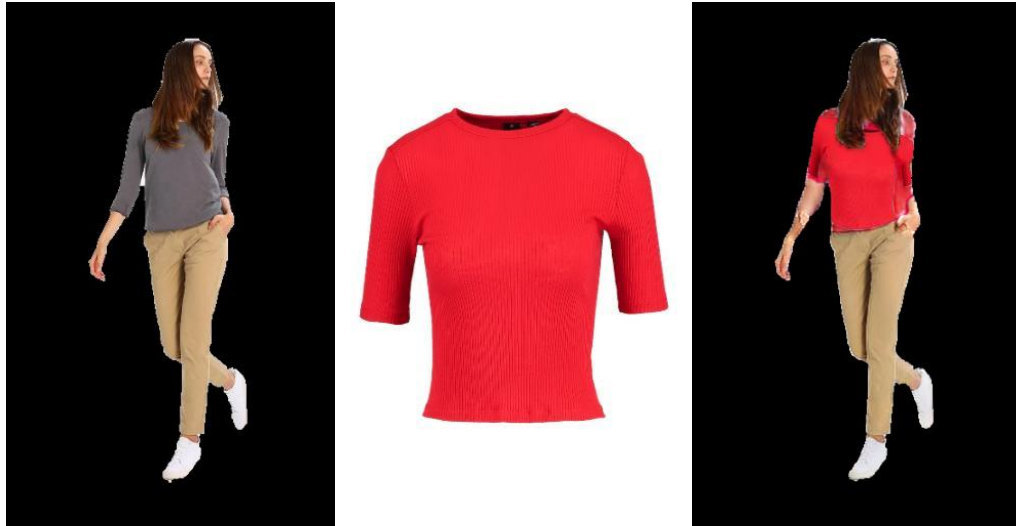


Figure 6-16: Demonstration of Person, Target Cloth and 2D Try-On Output

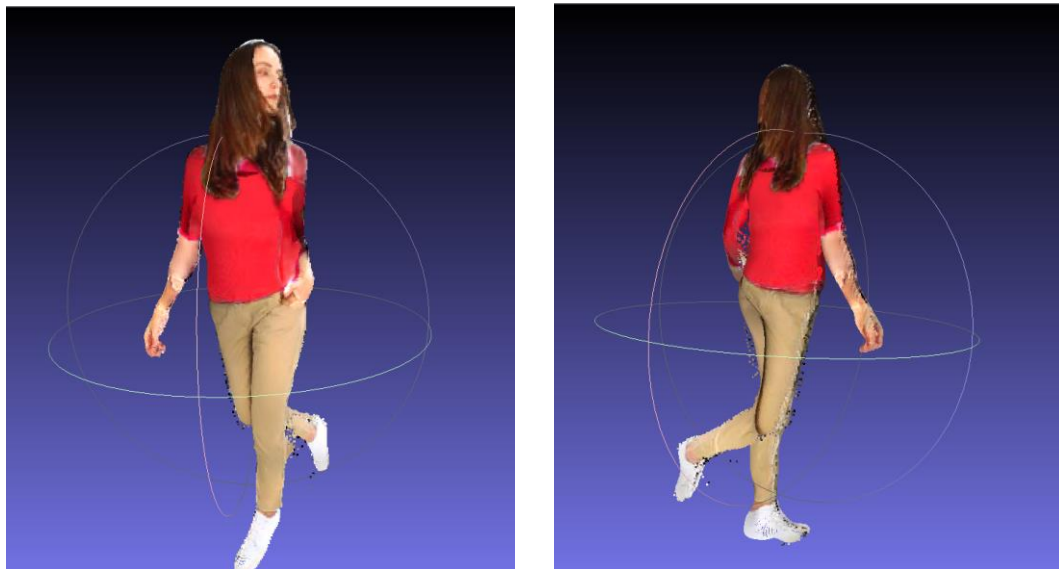


Figure 6-17: Front and Back of 3D Output with Unusual Result

In the provided figure, despite the unique pose, the generated 3D mesh appears almost satisfactory. However, the current inpainting methods used to predict the back portion of the image result in anomalies, particularly noticeable in the leg area of the back view. This is due to the frontal leg appearing in the in-painted back side, resulting in an unusual bone structure. **To address this issue, more advanced inpainting techniques** could be explored to improve the accuracy of predicting the back view of the human body.

6.3 Full Stack Integration

A demonstration of the website handling inputs of image and cloth with illustration of output try-on is given below:

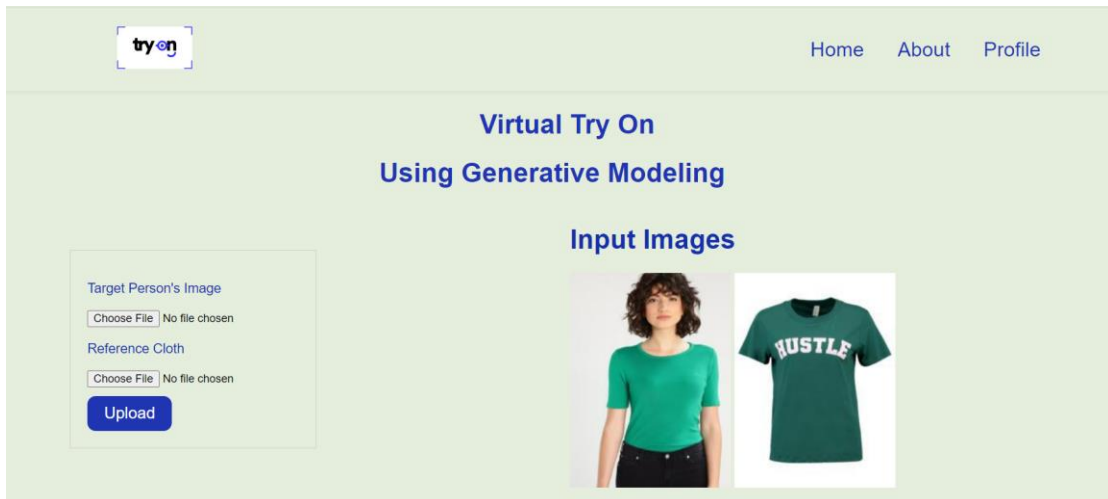


Figure 6-18 Demonstration of User Input in Web Application

With above inputs, the output image with intermediate output images of the module as well are displayed in the website as below:

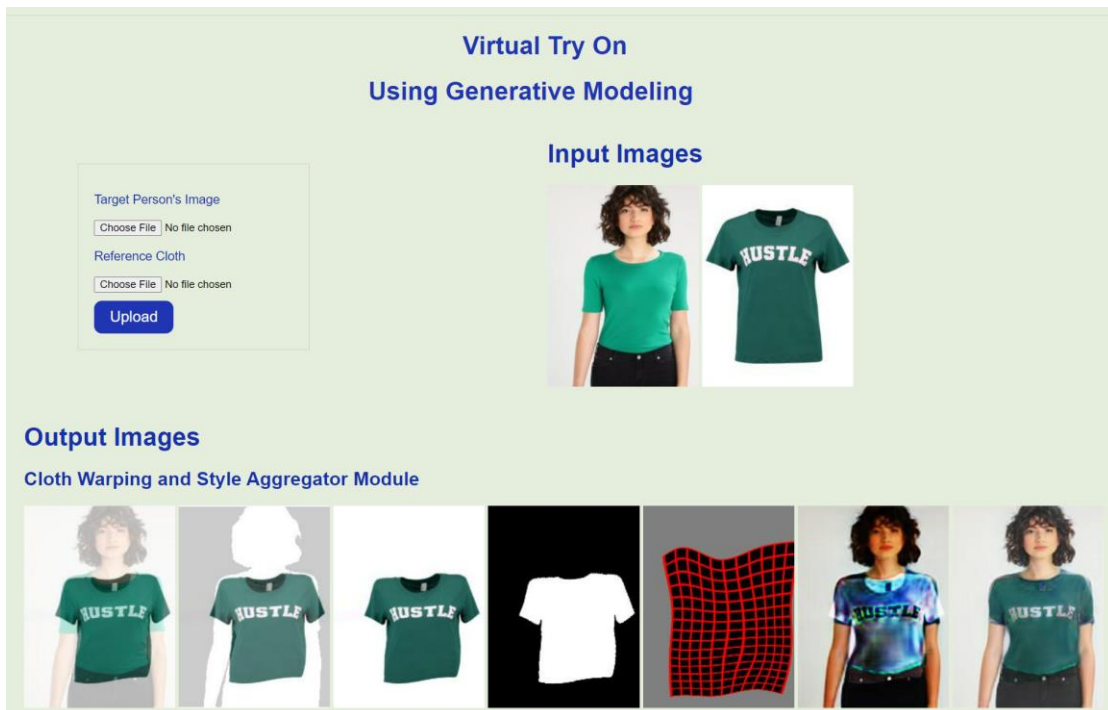


Figure 6-19 Demonstration of Illustrated Outputs in Web Application

Along with the website for the demonstration, a mobile application is developed with Flutter framework that handles the user inputs where image of the person can be clicked or chosen from the files in mobile phone. The mobile application is designed to capture an image of a person or select one from the gallery, which is then sent to the backend through Rest API. The interface includes two rectangles, each holding the image of a person and the targeted image. Users can capture an image or select one from the gallery using the *image_picker* package.

Additionally, the user has the option to select the target garment or upper body clothing for females from the cloth page, which displays images of all available garments. Upon sending the inputs, which include the person's image and the target image, the application generates an output.

Users can view the outputs on another screen by tapping on them. To enhance the viewing experience, the application incorporates the 'PhotoView' package, allowing users to zoom in on the photo. The output is displayed by scaling it two times, and users have control over both zooming and rotating the picture. Some snip-shots of the mobile application are given as follows:

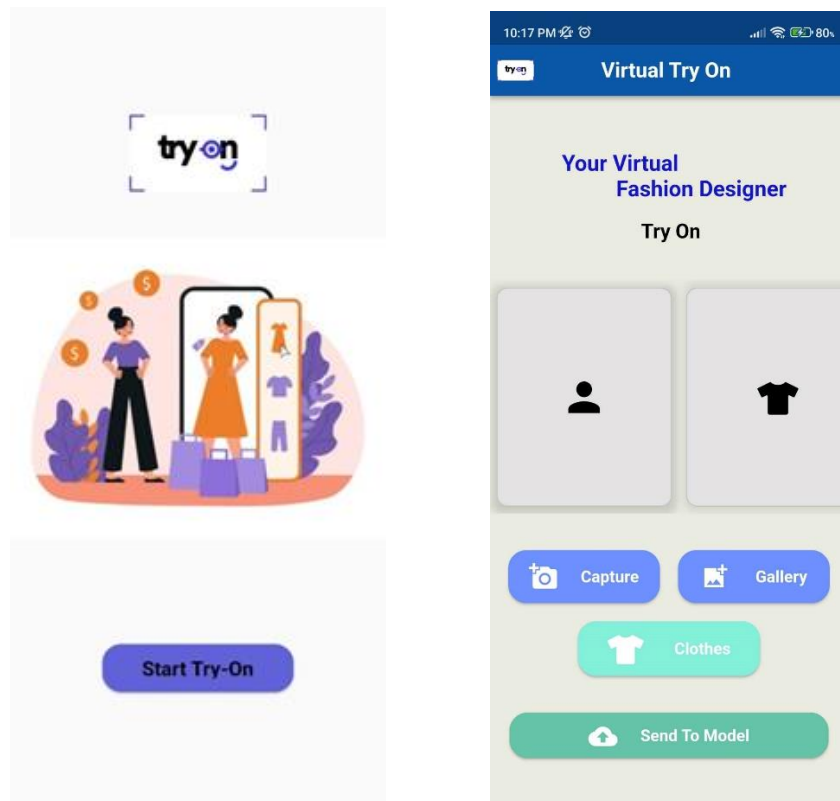


Figure 6-20 Home Screen and Try-On Screen of Mobile App

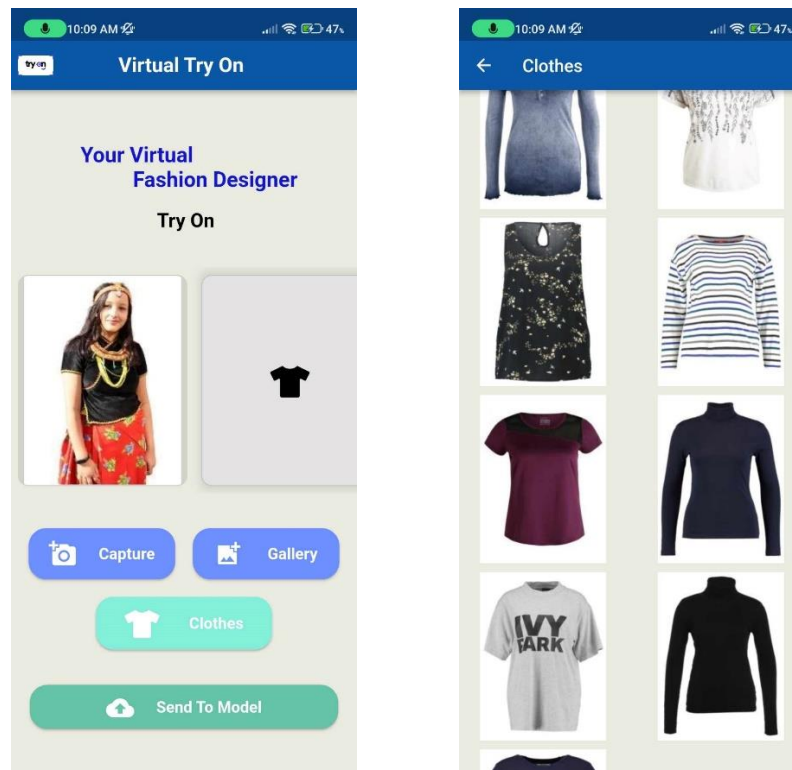


Figure 6-21 Choice of Person(left) & Choices of Clothes(right) in Application

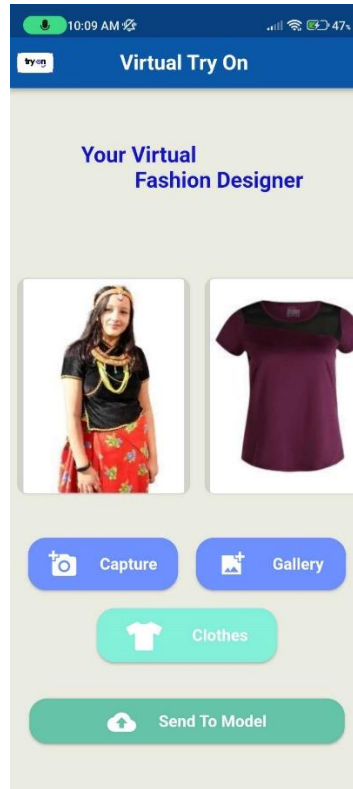


Figure 6-22 Chosen Person and Cloth in Mobile Application

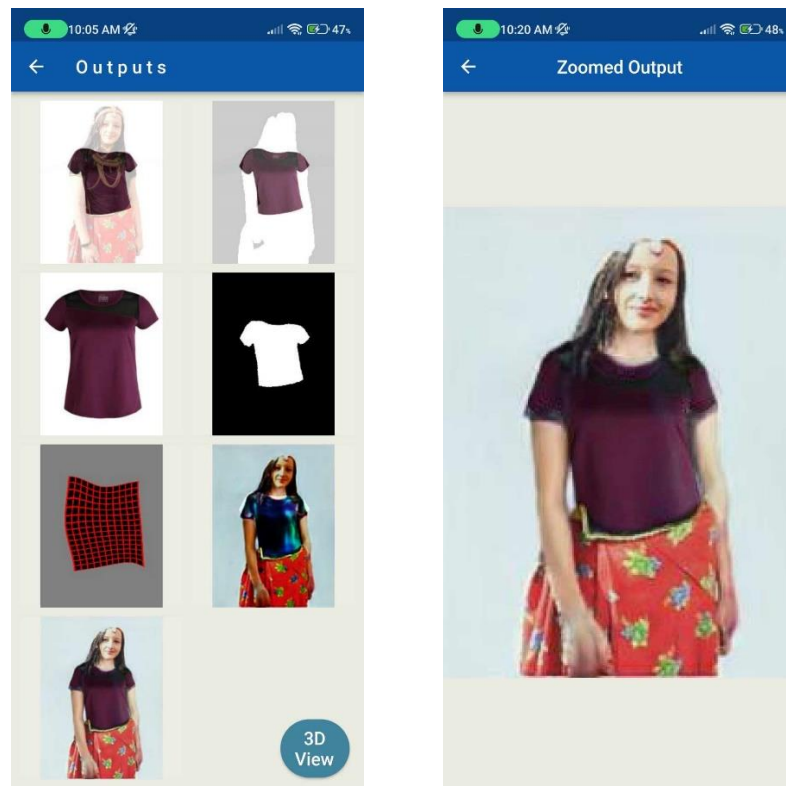


Figure 6-23 Module **Outputs** (Left) & Final Try-On Output (Right)

7 FUTURE ENHANCEMENT

Further enhancements can be made on top of this system which are mentioned as follows:

- Improvement of 3D model with 3D annotated datasets
- Use of advanced architectures like diffusion models for in-depth capture of features of the person and pose
- Addition of Input Validation for cloth and camera angle of the image passed into the system

8 CONCLUSION

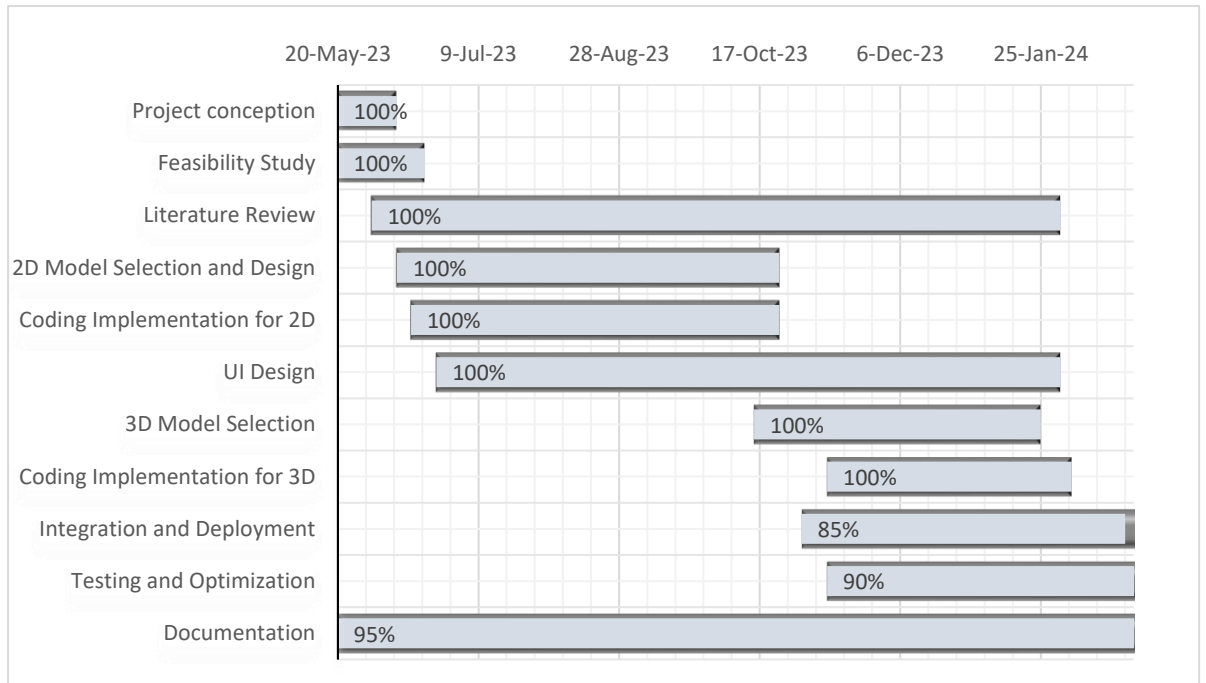
Cloth Warping Module and Style Aggregator Module were used to warp the target cloth as per the pose of the target person and fit the warped cloth over the person's body. Many varieties of cloth were demonstrated for images with different poses. Initial and Final Depth Estimator Modules were used to get the depth of the image from the 2D image. The try on output of the target cloth over the person was then used together with the final depth of foreground and background and 3D mesh was reconstructed using inpainting methods.

The project used OpenPose for the keypoint detection, DeepLabV3 for the image segmentation and Sobel operation for edge detection in the image. The output of the image is then visualized with the utilization of REST framework for enhancing the user interaction. For construction of 3D object of the person from 2D image, depth of the image has been utilized from front and back of the body. 3D rendering applications like MeshLab and 3D Viewer of Python have been used for viewing the 3D view of the person wearing the target cloth. Hence, the system successfully transfers target garment onto the target person with proper alignment. The 3d mesh of the person wearing the garment is also rendered with successful reconstruction for the most part. Thus, it can be concluded that the project successfully achieved its objectives.

9 APPENDICES

Appendix A: Project Schedule

Table 9-2: Gantt Chart



Appendix B: Architecture of U-NET

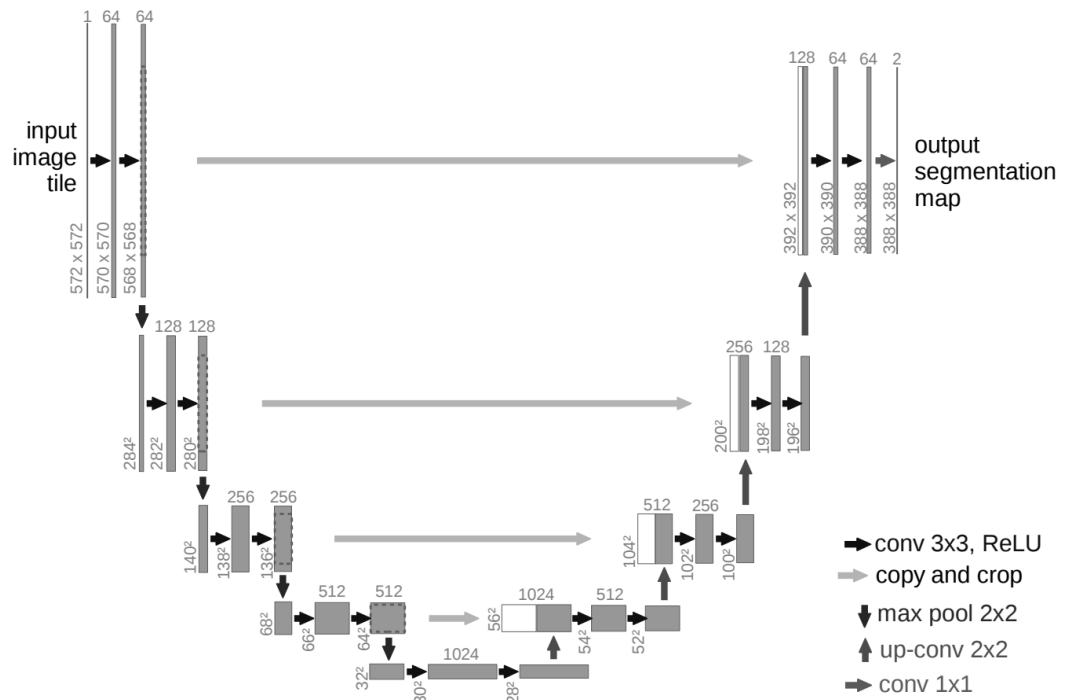


Figure 9-1: U-Net Architecture

Appendix C: Architecture of VGG-19

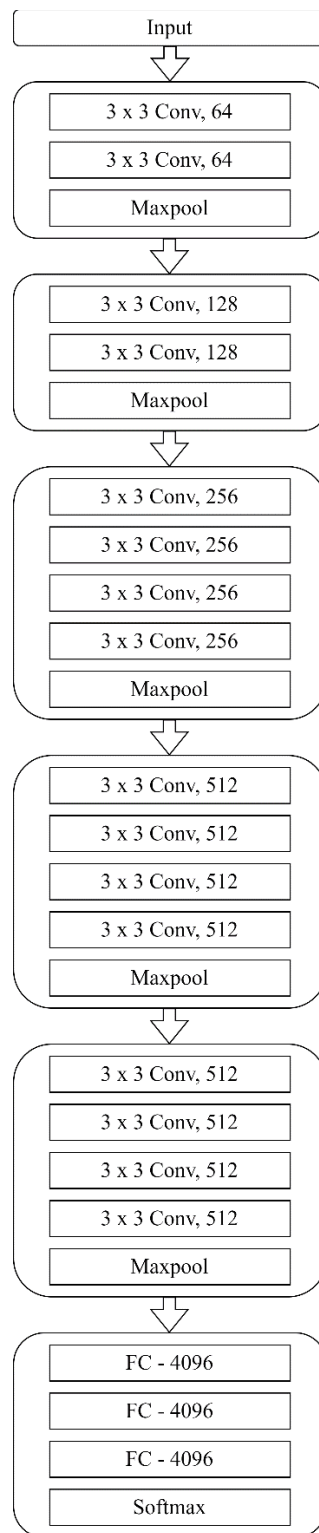


Figure 9-2 Architecture of VGG-19

References

- X. Han, Z. Wu, Z. Wu, R. Yu and L. S. Davis, "VITON: An Image-based Virtual Try-on Network," in *IEEE conference on computer vision and pattern recognition*, 2018.
- [1] B. Wang, H. Zheng, X. Liang, Y. Chen, L. Lin and M. Yang, "Toward Characteristic-Preserving Image-based Virtual Try-On Network," in *European conference on computer vision*, Munich, 2018.
- [2] M. R. Minar, T. T. Tuan, H. Ahn, P. L. Rosin and Y.-K. Lai, "CP-VTON+: Clothing Shape and Texture Preserving Image-Based Virtual Try-On," in *Conference on Computer Vision and Pattern Recognition*, Seattle, 2020.
- [3] Y. Han, Z. Ruimao, G. Xiaobao, L. Wei, Z. Wangmeng and L. Ping, "Towards Photo-realistic Virtual Try-on by Adaptively Generating-Preserving Image Content," in *IEEE/CVF conference on computer vision and pattern recognition*, 2020.
- [4] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI*, 2015.
- [5] M. Aymen, A. Thiemo and P.-M. Gerard, "Learning to Transfer Texture from Clothing Images to 3D Humans," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [6] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE conference on computer vision and pattern*, 2016.
- [7]

- [8] K. M. Lewis, S. Varadharajan and I. Kemelmacher-Shlizerman, "TryOnGAN: Body-Aware Try-On via Layered Interpolation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, p. 10, 2021.
- [9] H. Sen, S. Yi-Zhe and X. Tao, "Style-Based Global Appearance Flow for Virtual Try-On," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
- [10] Z. Cao, T. Simon, S.-E. Wei and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," in *IEEE Conference on Computer Vision and Pattern Recognition* , 2017.
- [11] M. Jaderberg, K. Simonyan and A. Zisserman, "Spatial Transformer Networks," in *Advances in Neural Information Processing Systems*, 2015.
- [12] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.
- [13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition.," in *International Conference on Learning Representations*, 2014.
- [14] D. P. Kingma and J. Lei Ba, "Adam: A method for stochastic optimization," *ICLR*, vol. 1412, no. 6980, 2015.
- [15] V. Ronny and L. Na, "TensorFlow Blog," Google, 17 May 2021. [Online]. Available: <https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html>. [Accessed 15 June 2023].

- [16] D. Kaiwen, B. Song, X. Lingxi, Q. Honggang, H. Qingming and T. Qi, "CenterNet: Keypoint Triplets for Object Detection," in *IEEE/CVF international conference on computer vision.*, California, 2019.
- [17] G. I. J., P.-A. Jean, M. Mehdi, X. Bing, W.-F. David, O. Sherjil, C. Aaron and B. Yoshua, "Generative Adversarial Nets," *Communications of the ACM*, vol. 63, no. 11, pp. 139-144, 2020.
- [18] K. Tero, L. Samuli and A. Timo, "A Style-Based Generator Architecture for Generative Adversarial Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, 2019.