VIRTUAL SHIRT FITTING MODEL BASED ON DEEP LEARNING AND COMPUTER VISION

by

Tin Tran Ngoc Xuan
Thach Duong Vien
Duy Nguyen Thanh Anh

THE FPT UNIVERSITY HO CHI MINH CITY

VIRTUAL SHIRT FITTING MODEL BASED ON DEEP LEARNING AND COMPUTER VISION

by

Tin Tran Ngoc Xuan
Thach Duong Vien
Duy Nguyen Thanh Anh

Supervisor: Mr. Vu Hoang Anh

A final year capstone project submitted in partial fulfillment of the requirement for the Degree of Bachelor of Artificial Intelligent in Computer Science

DEPARTMENT OF ITS
THE FPT UNIVERSITY HO CHI MINH CITY
December 3, 2023

ACKNOWLEDGEMENTS

Our sincere gratitude goes to our supervisor, Mr. Vu Hoang Anh, for his invaluable support, extensive knowledge, and unwavering patience throughout our thesis journey. His expert guidance and constructive feedback were instrumental in the successful completion of this project. We also express our appreciation to the Review Committee for their meticulous review, ensuring the adherence to high standards. Mr. Vu Hoang Anh's exceptional mentorship has been a cornerstone of our academic success, and we thank him for his dedication. Our heartfelt thanks extend to all who contributed, both in significant and subtle ways, as their support was crucial to the realization of this project.

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the report as follows: Conceptualization, Duy Nguyen. and Tin Tran.; Methodology, Tin Tran. and Thach Duong.; Software, Duy Nguyen.; Formal analysis, Tin Tran. and Thach Duong.; Data curation, Duy Nguyen.; Writing-original draft preparation, Duy Nguyen.; Writing-review and editing, Duy Nguyen. and Tin Tran.; Visualization, Tin Tran.; Project administration, Thach Duong. All authors reviewed the results and approved the Final Capstone Project document.

ABSTRACT

The field of image-based virtual try-on endeavors to generate realistic images of individuals adorned in specific clothing items. Traditional approaches address this task by independently warping the clothing item to conform to the person's body and generating a segmentation map of the individual wearing the item. Unfortunately, this sequential operation often results in misalignment between the warped clothes and the segmentation map, introducing artifacts in the final synthesized image. Additionally, the lack of information exchange between the warping and segmentation generation stages leads to pixel-squeezing artifacts, particularly near clothing regions occluded by body parts. To overcome these challenges, this study introduces a novel try-on condition generator, unifying the warping and segmentation generation stages. A newly proposed feature fusion block facilitates information exchange, mitigating misalignment and pixel-squeezing artifacts. Furthermore, the incorporation of discriminator rejection filters ensures the accuracy of segmentation map predictions, enhancing the overall performance of virtual try-on frameworks. Experimental results on a high-resolution dataset showcase the effectiveness of the proposed model in addressing misalignment and occlusion, surpassing baseline models. Code is available at https://github.com/ntad27/AIP490.

Keywords: Deep Learning (DL), Computer Vision (CV), High-Resolution Virtual Try-On, Misalignment-Free, Occlusion-Handling

Contents

List of Figures

List of Tables

1 INTRODUCTION

In the ever-evolving landscape of digital commerce, the escalating significance of online shopping has propelled technological innovations aimed at enhancing the overall customer experience. A key advancement in this realm is the emergence of virtual try-on technologies, poised to revolutionize the way consumers engage with fashion and make informed purchase decisions. At the heart of this transformative experience lies the virtual try-on task, a sophisticated process designed to seamlessly transpose a chosen clothing item onto an individual, effectively simulating how the garment would appear when worn.

It's noteworthy that within the domain of virtual try-on methodologies, a distinction emerges between 3D-based approaches, relying on intricate 3D measurements of garments [1, 2, 7, 17], and the more streamlined image-based virtual try-on methods [4, 11, 13, 16, 18, 20, 22, 21], which constitute the focal point of our exploration. While both paradigms share the overarching goal of providing users with a lifelike preview of clothing items, the shift to image-based techniques introduces a layer of simplicity and accessibility. Image-based virtual try-on uniquely necessitates only a garment and a person image, eliminating the need for complex 3D garment measurements. This strategic shift towards image-centric approaches not only streamlines the virtual try-on process but also makes it more feasible for widespread and practical real-world applications.

In dissecting the intricacies of image-based virtual try-on, it becomes evident that certain stages play a pivotal role in shaping the final outcomes. Notably, the explicit consideration of segmentation maps has become a common thread among many methodologies [4, 11, 13, 16, 18, 20, 22, 21]. Previous studies underscore the importance of employing a dedicated warping module to align the clothing image with the person's body, ensuring a seamless integration that reflects a true-to-life representation. Furthermore, the predictive power of segmentation maps in guiding the generation process is crucial, as it aids in delineating regions to be generated and those to be preserved, a factor that gains increasing significance with higher image resolutions.

In essence, the fusion of these sophisticated techniques not only exemplifies the cutting-edge nature of virtual try-on technologies but also emphasizes a commitment to making these innovations practical and applicable in real-world scenarios. As researchers delve deeper into refining and optimizing image-based virtual try-on methodologies, the aim is not only to enrich the customer's online shopping experience but also to usher in a new era of interactive and immersive digital commerce, where users can confidently visualize and personalize their fashion choices before making a purchase.

In tackling the intricacies of virtual try-on tasks, preceding research endeavors have incorporated an explicit warping module designed to meticulously align the clothing image with the contours of the individual's body. This strategic inclusion is pivotal in ensuring a harmonious integration of the selected attire onto the person. Furthermore, a noteworthy advancement in the virtual try-on domain is observed through the adoption of a predictive segmentation map for the final image. This innovative approach serves a dual purpose by not only mitigating the complexities associated with image generation but also by providing a guiding framework for the layout of the person and demarcating regions to be generated versus those to be preserved [18].

The significance of the segmentation map becomes more pronounced as one delves into higher image resolutions, amplifying its role as a critical component in the overall virtual try-on pipeline. This meticulous consideration of segmentation maps is a common thread among various image-based virtual try-on methodologies, as evidenced by the incorporation of such stages in the processes outlined by notable studies [11, 13, 18, 16, 19, 20, 22]. The outputs derived from the warping and segmentation map generation modules wield substantial influence over the ultimate virtual try-on outcomes, underscoring the pivotal role played by these stages in refining and enhancing the realism and accuracy of the final results. Thus, by strategically interweaving these components into the fabric of virtual try-on frameworks, researchers aim to not only push the boundaries of technological innovation but also to yield outputs that resonate seamlessly with real-world applications and user expectations.

Within the realm of virtual try-on frameworks, a persistent challenge has emerged, particularly in systems characterized by the incorporation of warping and segmentation generation modules. This challenge manifests as misaligned regions, wherein discrepancies between the warped clothes and the segmentation map create what is commonly referred to as misalignment. The deleterious consequences of misalignment are readily apparent, with artifacts compromising the perceptual quality of the final result, a concern that becomes particularly pronounced at higher resolutions.

The adverse impact of misalignment is evident in how it gives rise to noticeable artifacts within specific regions of the virtual try-on output. These artifacts, ranging from distortions to irregularities, significantly diminish the fidelity of the simulated clothing experience, impinging on the intended immersive and realistic presentation. The ramifications are particularly noteworthy in scenarios where the virtual try-on is executed at elevated resolutions, exacerbating the perceptual detriments.

A crucial insight into the root cause of misalignment lies in the operational independence of the warping module and the segmentation map generator. Operating in silos, these components lack a seamless exchange of information, a factor that contributes significantly to the persistence of misalignment issues. Despite recent efforts, exemplified by a notable study attempting to mitigate artifacts in misaligned regions, the existing methodologies fall short of providing a comprehensive solution to the misalignment problem.

As the virtual try-on landscape continues to evolve, addressing and resolving the misalignment challenge becomes pivotal for advancing the field. A holistic approach that fosters synergy between the warping module and the segmentation map generator is imperative, necessitating a paradigm shift from their current disjointed operation. Researchers are actively engaged in exploring innovative solutions that foster information exchange between these modules, seeking to eradicate misalignment and enhance the perceptual quality of virtual try-on results across varying resolutions.

In summary, the persistent misalignment issue within virtual try-on frameworks underscores the ongoing quest for refinement and optimization in this dynamic field. While recent studies have made strides in mitigating artifacts, a comprehensive resolution to the misalignment problem remains an elusive goal, motivating researchers to delve deeper into collaborative strategies between key modules to usher in a new era of seamless and artifact-free virtual try-on experiences.

In the intricate landscape of virtual try-on frameworks, the challenges persist, and a noteworthy issue stemming from the information disconnection between two pivotal modules has garnered significant attention. This particular concern gives rise to what is termed as pixel-squeezing artifacts, which have notable ramifications for the overall quality of the virtual try-on output.

Pixel-squeezing artifacts come to the forefront, prominently visible in scenarios where body parts occlude the intended garment. The detrimental impact on the results is evident in the form of distortions and visual inconsistencies, leading to a compromised virtual try-on experience. The root cause of pixel-squeezing artifacts lies in the excessive warping of clothes near occluded regions, stemming from the lack of effective information exchange between the warping and segmentation map generation modules.

The absence of a seamless communication channel between these critical components hinders the ability to adapt the virtual try-on system to complex real-world scenarios, limiting its applicability and practicality. This issue becomes particularly pronounced when attempting to simulate varied poses and configurations of the person images. The constraints introduced by pixel-squeezing artifacts impede the versatility of virtual try-on applications, making it challenging to deploy these systems seamlessly in real-world contexts.

Efforts to address the pixel-squeezing artifact challenge are integral to advancing the efficacy of virtual try-on frameworks. Researchers are actively engaged in exploring innovative solutions that foster a more cohesive exchange of information between the warping and segmentation map generation modules. The aim is to rectify the distortions introduced near occluded regions, thus broadening the spectrum of possible poses and scenarios that can be accommodated within the virtual try-on paradigm.

In the grander scheme of virtual try-on research and development, acknowledging and tackling issues like pixel-squeezing artifacts signifies a commitment to enhancing the realism, adaptability, and utility of these technologies. As these challenges are navigated, the potential for virtual try-on to seamlessly integrate into diverse real-world applications becomes more tangible, fostering a future where virtual fashion exploration aligns seamlessly with the dynamic and ever-evolving landscape of fashion and e-commerce.

In our continuous pursuit to address and overcome the challenges inherent in virtual try-on frameworks, we introduce a pioneering solution in the form of a novel try-on condition generator. This innovative module is designed to harmonize the warping and segmentation generation components, offering a unified approach to seamlessly predict both the warped garment and the accompanying segmentation map. The key breakthrough lies in the simultaneous prediction of these elements, ensuring perfect alignment and eliminating misalignment issues that have plagued previous methodologies.

The proposed try-on condition generator stands as a transformative advancement, not only eradicating misalignment but also adeptly handling occlusions introduced by body parts in a natural and intuitive manner. By predicting the warped garment and segmentation map in tandem, our framework introduces a level of coherence that significantly enhances the overall fidelity and realism of virtual try-on outputs.

A cornerstone of our approach is the capacity to effectively handle occlusions, a persistent challenge in virtual try-on scenarios, especially when the clothing is partially obscured by various body parts. The unified predictions of the try-on condition generator empower the system to navigate and adapt seamlessly to such occlusions, preserving the accuracy and visual appeal of the virtual try-on results.

Extensive experimentation validates the efficacy of our proposed framework, showcasing its ability to not only handle occlusions and misalignment but also to deliver state-of-the-art results, particularly on high-resolution datasets such as 1024×768. This comprehensive evaluation, conducted both quantitatively and qualitatively, underscores the robustness and superiority of our solution in comparison to existing methodologies.

As the virtual try-on landscape continues to evolve, our contribution marks a significant step towards a more refined and realistic virtual try-on experience. The integration of the proposed try-on condition generator not only mitigates existing challenges but also sets a new standard for the capabilities of virtual try-on frameworks, demonstrating their potential to provide highly accurate, visually compelling, and versatile outputs. Through our innovative approach, we aspire to pave the way for a future where virtual try-on technologies seamlessly integrate into diverse applications, enhancing the overall user experience in the realms of fashion, e-commerce, and beyond.

2 RELATED WORK

2.1 Image-based Virtual Try-On

Embarking on the intricacies of image-based virtual try-on, the overarching goal is to generate a visually coherent person image adorned with a specified target clothing item, leveraging a pair of input images comprising clothes and a person. The evolution of recent virtual try-on methodologies [11, 13, 16, 18, 19, 20, 22] unveils a common structural framework, typically comprising three distinct modules: 1) the segmentation map generation module, 2) the clothing warping module, and 3) the fusion module. This three-tiered architecture collaboratively orchestrates the transformation from input images to a seamlessly integrated virtual try-on output.

The initial stage, the segmentation map generation module, serves as the foundation for subsequent processes. This module generates a segmentation map that delineates different regions of the person image, providing a structural roadmap for the subsequent stages of the virtual try-on pipeline. Following this, the clothing warping module takes center stage, leveraging the segmentation map to guide the transformation of the target clothing onto the person image. This strategic alignment ensures that the virtual try-on results maintain a realistic and anatomically accurate representation, laying the groundwork for a compelling user experience.

The fusion module, a pivotal component in the virtual try-on methodology, emerges as the synthesis hub where intermediate representations, such as the warped clothes and segmentation maps generated in previous stages, are harnessed to produce the final photo-realistic images. This critical stage involves the amalgamation of these representations, orchestrating a harmonious blend that captures the nuances of texture, shading, and contours, ultimately yielding a visually convincing virtual try-on outcome.

While the existing virtual try-on methods have made significant strides, the emphasis on these three modular stages underlines the importance of a systematic and coherent approach to tackle the complexities inherent in the task. As technology continues to advance, researchers are actively engaged in refining and optimizing each module, exploring innovative techniques to enhance the realism, accuracy, and applicability of virtual try-on systems across diverse scenarios.

By dissecting the virtual try-on pipeline into these distinct modules, the research community aims to not only address current challenges but also to pave the way for future advancements. The quest for more sophisticated, adaptive, and user-centric virtual try-on solutions remains ongoing, fueled by the collective endeavor to seamlessly integrate this technology into the fabric of everyday experiences, revolutionizing the way we interact with and perceive fashion in the digital realm.

2.2 High-Resolution Virtual Try-On with Misalignment and Occlusion-Handled Conditions

Given a reference image-person with dimension (3 x H x W) of a person and a cloth image also in the same dimension as the image-person (H and W denote the image height and width, respectively), the goal is to synthesize an image (3 x H x W) of a person in person-image wearing the shirt in cloth-image, where the pose and the body shape of person-image maintained. Following the training procedure of VITON [11], the author trained the model to reconstruct image-person from a clothing-agnostic person representation and cloth-image that person is wearing already. The clothing-agnostic person representation eliminates any clothing information in image-person, and it allows the model to generalize at test time when an arbitrary clothing image is given.

The framework is composed into two stages: 1) a *try-on condition generator*, 2) a *try-on image generator*. Given the clothing-agnostic person representation and cloth-image and produces the segmentation map simultaneously. The generator deforms does not create any misalignment or pixel -squeezing artifacts. Afterward, the try-on image generator synthesizes the final try-on result using the outputs of the try-on condition generator. At test time, the author applied discriminator rejection that filters out incorrect segmentation map predictions.

HR-VITON with Misalignment and Occlusion-Handled Conditions

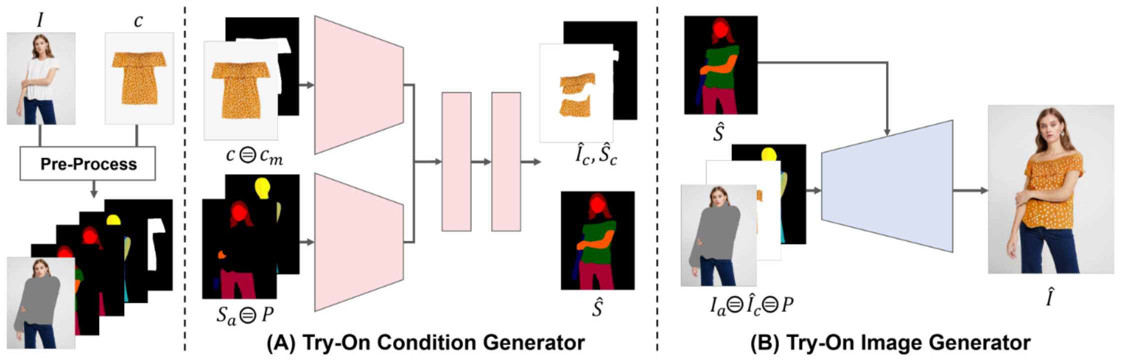


Figure 1: Overview of the author's framework (HR-VITON)

Pre-Processing. In the pre-processing step, the author obtained a segmentation map $S \in L^{H \times W}$ of the person, a clothing mask $c_m \in L^{H \times W}$, and a pose map $P \in \mathbb{R}^{3 \times H \times W}$ with the off-the-shelf models [9, 10], where L is a set of integers indicating the semantic labels. For the pose map P, they utilized a dense pose [10], which maps all pixels of the person regions in the RGB image to the 3D surface of the person's body. For the clothing-agnostic person representation, they employed a clothing-agnostic person image I_a and a clothing-agnostic segmentation map S_a as those of VITON-HD [19].

2.2.1 Try-On Condition Generator

In this stage, the author aim to generate the segmentation map $\hat{S}$ of the person wearing the target clothing item c and deform c to fit the body of the person. A warped clothing image $\hat{I}_c$ and a generated segmentation map $\hat{S}$ are used as the conditions for the try-on image generator.

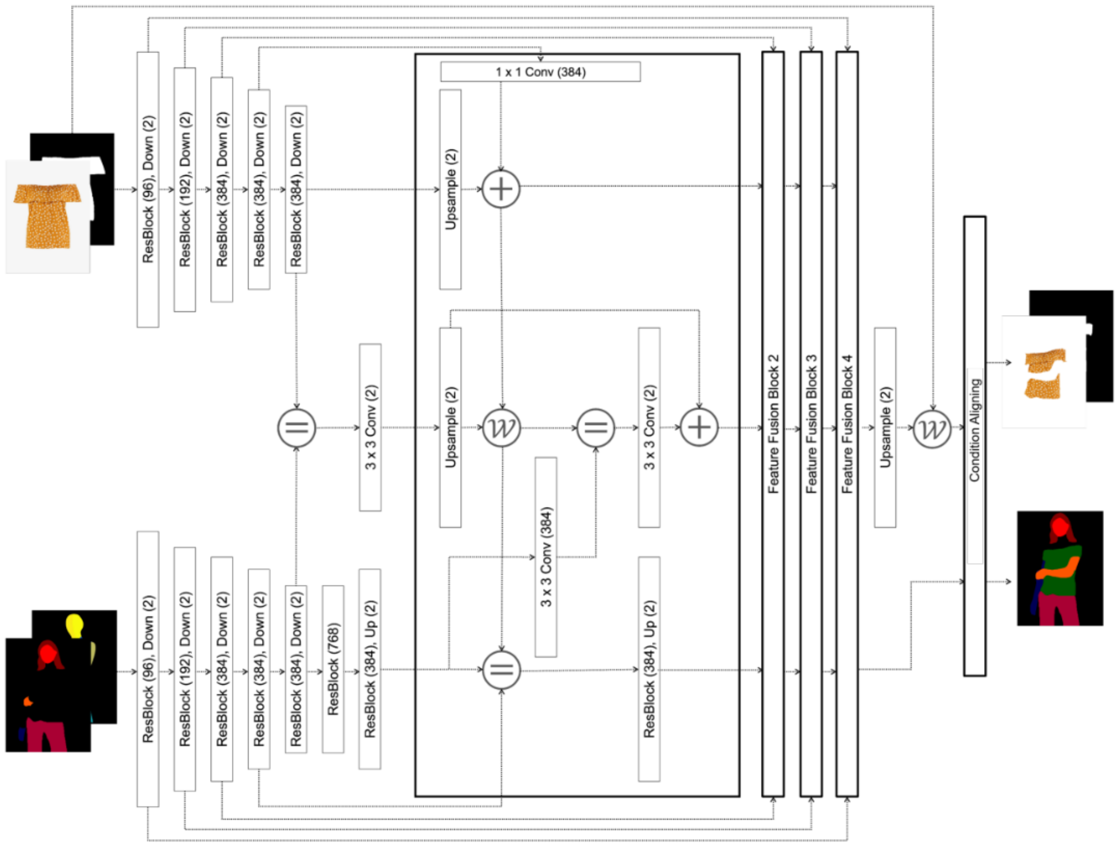


Figure 2: The detailed architecture of the try-on condition generator

The try-on condition generator consists of two encoders and four fusion blocks, and each encoder is composed of five residual blocks. The features of the last residual blocks are concatenated and passed to a 3x3 convolutional layer, which generates the first flow map of the flow pathway. Also, the last feature of the segmentation encoder is used as the input of the segmentation pathway (i.e., seg pathway) after passing through two residual blocks. The author employed two multi-scale discriminators for the conditional adversarial loss. The visualization of the try-on condition generator architecture is in Figure 2.

Loss Functions. The author used the pixel-wise cross-entropy loss $\mathcal{L}_{CE}$ between predicted segmentation map $\hat{S}$ and S. Additionally, L1 loss and perceptual loss are used to encourage the network to warp the clothes to fit the person's pose. These loss functions are also directly applied to the intermediate flow estimations to prevent the intermediate flow maps from vanishing and improve the performance. Formally, $\mathcal{L}_{L1}$ and $\mathcal{L}_{VGG}$ are as follows:

$$\mathcal{L}_{L1} = \sum_{i=0}^{3} w_i \cdot \left\| W\left(c_m, F_{f_i}\right) - S_c \right\|_1 + \left\| \hat{S}_c - S_c \right\|_1,$$

$$\pounds_{VGG} = \sum_{i=0}^{3} w_i \cdot \phi\left(W\left(c, F_{f_i}\right), I_c\right) + \phi\left(\hat{I}_c, I_c\right),$$

where w_i determines the relative importance between each terms.

$\pounds_{TV}$ is a total-variation loss to enforce the smoothness of the appearance flow:

$$\pounds_{TV} = \left\| \Delta F_{f_4} \right\|_1$$

They found that regularizing only the last appearance flow F_{f_4} is vital in learning the flow estimation at coarse scales. Totally, the try-on condition generator is trained end-to-end using the following objective function:

$$\pounds_{TOCG} = \lambda_{CE}\pounds_{CE} + \pounds_{cGAN} + \lambda_{L1}\pounds_{L1} + \pounds_{VGG} + \lambda_{TV}\pounds_{TV},$$

where $\pounds_{cGAN}$ is conditional GAN loss between $\hat{S}$ and S, and λ_{CE}, λ_{L1}, and λ_{TV} denote the hyper-parameters controlling relative importance between different losses. For $\pounds_{cGAN}$, we used the least-squared GAN loss [6].

During the training of the try-on condition generator, the model predicts $\hat{I}_c$, $\hat{S}_c$ and $\hat{S}$ at 256x192 resolution. In the inference phase, before forwarding the try-on image generator, the segmentation map and the appearance flow obtained from the try-on condition generator are upscaled to 1024x768. The author down-sampled the inputs for the discriminator of the try-on condition generator by a factor of 2 to increase the receptive field. In addition, the author applied a dropout [3] to the discriminator to stabilize the training. For hyper-parameters they used, λ_{CE}, λ_{VGG} and λ_{TV} are set to 10, 10 and 2, respectively. The batch sizes for training the try-on condition generator and image generator are set to 8 and 4, respectively. They trained each module for 100,000 iterations. The learning rates of the generator and the discriminator of the try-on condition generator are set to 0.0002.

2.2.2 Try-On Image Generator

In this stage, the author generate the final try-on image $\hat{I}$ by fusing the clothing-agnostic image I_a, the warped clothing image $\hat{I}_c$, and the pose map P, guided by $\hat{S}$. The try-on image generator consists of a series of residual blocks, along with upsampling layers. The residual blocks use SPADE [15] as normalization layers whose modulation parameters are inferred from $\hat{S}$. Also, the input (I_a, $\hat{I}_c$, P) is resized and concatenated to the activation before each residual block. They trained the generator with the same losses used in SPADE and pix2pixHD [14].

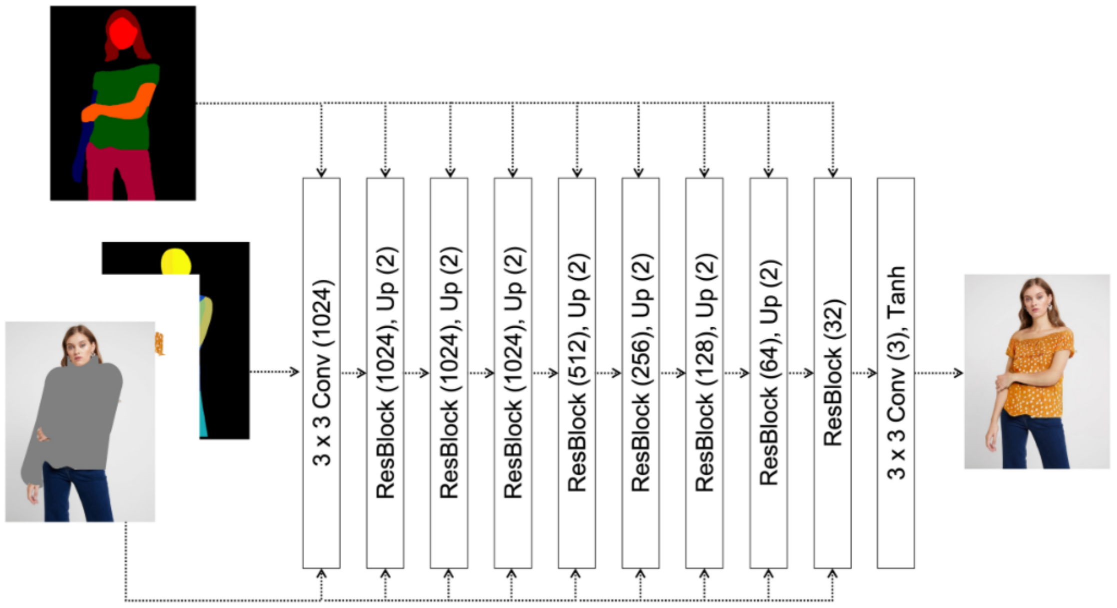


Figure 3: The detailed architecture of the try-on image generator

The detailed architecture of the try-on image generator as shown in Figure 3. The generator is composed of a series of residual blocks with upsampling layers, and two multi-scale discriminators are employed for the conditional adversarial loss. Spectral normalization [12] is applied to all the convolutional layers.

To train the try-on image generator, the author utilized the same losses used in SPADE [15] and pix2pixHD [14]. Specifically, their full objective function consists of the conditional adversarial loss, and the feature matching loss. Formally, the objective function is as follows: $\pounds_{TOIG} = \pounds_{cGAN}^{TOIG} + \lambda_{VGG}^{TOIG}\pounds_{VGG}^{TOIG} + \lambda_{FM}^{TOIG}\pounds_{FM}^{TOIG}$, where $\pounds_{cGAN}^{TOIG}$, $\pounds_{VGG}^{TOIG}$, and $\pounds_{FM}^{TOIG}$ denote the conditional adversarial loss, the perceptual loss, and the feature matching loss [14], respectively. The author used λ_{VGG}^{TOIG} and λ_{FM}^{TOIG} for hyper-parameters controlling relative importance between different losses. For $\pounds_{cGAN}^{TOIG}$, they employed the Hinge loss [5]. λ_{VGG}^{TOIG} and λ_{FM}^{TOIG} are set to 10. The learning rates of the generator and the discriminator of the try-on image generator are set to 0.0001 and 0.0004, respectively. The author adopted the Adam optimizer with β_1=0.5 and β_2=0.999 for both modules.

2.2.3 Discriminator Rejection

The author proposed a discriminator rejection method to filter out the low-quality segmentation map generated by the try-on condition generator at the test time. In the discriminator rejection sampling [8], the acceptance probability for an input x is

$$p_{accept}(x) = \frac{p_d(x)}{Lp_g(x)},$$

where p_d and p_g are the data distribution and the implicit distribution given by the generator, and L is a normalizing constant. As they used the least-squares GAN loss, the optimal discriminator is derived as follows:

$$D^*(x) = \frac{p_d(x)}{p_d(x)+p_g(x)}$$

Afterward, the acceptance probability can be represented using the discriminator D(x):

$$p_{accept}(x) = \frac{D(x)}{L(1-D(x))},$$

Where the equality is satisfied only if D = D^*. L is written as follows:

$$L = \underset{a}{max} \frac{D(x)}{(1-D(x))},$$

which is intractable. In practice, they constructed x from the segmentation map and input conditions (i.e., P, S_a, c, and c_m) and obtain L using the entire training dataset. The discriminator rejection enables us to filter out the incorrect segmentation maps faithfully.

3 PROJECT MANAGEMENT PLAN

Table 1: Project Management Plan.

Week	Task name	Owner	Note
1	- Start researching the topic - Collecting papers and projects related to the topic - Researching which try-on clothes models are having the best accuracy	All 3 members	
2	- Choose the best 3 models - Learning the overall architecture of each model - Try running 3 models again on what is provided	All 3 members	
3	- Choose HR-VITON model as the original model to improve - Rerun all HR-VITON data and evaluate it compared to the results stated in the paper	All 3 members	
4	- Find the limitations of HR-VITON model - Researching and find solutions for each of those limitations - Determine the limits of HR-VITON model	All 3 members	
5	- Research methods for preprocessing data of HR-VITON model (not provided by the author) - Research frameworks and libraries author used for the HR-VITON model	All 3 members	
6	- Upgrade the TensorFlow version of the Human Parse part from 1.15 to 2.0 - Learn other methods to improve preprocessing data	Tin Duy + Thach	

	- Add Remove background to preprocessing data	Duy + Thach
7	- Research and improve the preprocessing results of the Human Parse section - Retrain HR-VITON model over 1000 steps - Use the method of Remove background for the Cloth Mask part	Tin Thach Duy
8	- Draw the overall architectural diagram of the HR-VITON model - Visualize the results of each layer - Research and understand the mathematical formulas the author uses to process the HR-VITON model - Research and learn the metrics used to evaluate accuracy	Tin + Thach Tin + Thach Duy
9	- Together with the teacher, research methods to optimize the model - Collect additional necessary datasets	All 3 members
10	- Retrain the model over 1000 steps after applying different optimization methods - Test the model based on trained weight set - Learn how to build websites and demo apps - Come up with an idea to create a demo platform for the model	Tin Thach Duy Duy
11	- Train again on full steps model after optimization - Test the model based on the set of weights after training - Start writing reports	Tin Thach Duy
12	- Build web, app demo - Continue writing report	Tin + Thach Duy
13	- Edit and complete web, app demo	Tin + Thach

	- Edit and complete report	Duy
14	- Review knowledge and complete defense presentation slides - Present the topic to the council	All 3 members

4 MATERIALS AND METHODS

4.1 Data Preparation

For the experiments, we use the original high-resolution virtual try-on dataset introduced by VITON-HD [19] and the pre-processed one provided by HR-VITON [23]. The dataset contains 13,679 frontal-view women and top clothing image pairs with the resolution of the images being 1024x768. The dataset was split into a training and a test set with 11,647 and 2,032 pairs, respectively.

4.2 Data Pre-Processing

According to the explanation from the authors, at least 6 steps are needed for getting all the required inputs of the model. We've added one more pre-processing step and modified other steps in order to get better try-on results.

4.2.1 Remove Background

This is our additional step to improve the model's accuracy. We used the transparent-background library to remove the background of both cloth-image and person-image. Details are available at https://github.com/plemeri/transparent-background. The figure below shows some results of the original images after this step.

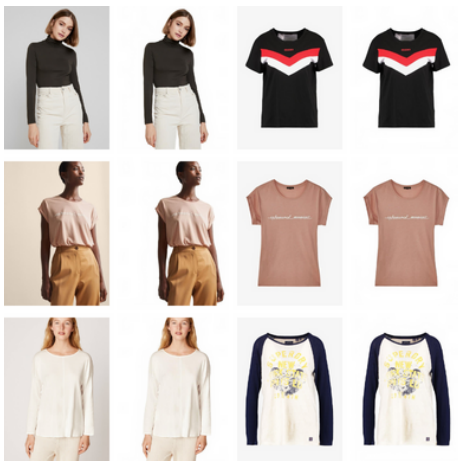


Figure 4: Remove Background Results

4.2.2 OpenPose

OpenPose is an open-source library for real-time multi-person keypoint detection and pose estimation. It was developed by the Carnegie Mellon University Perceptual Computing Lab and later supported by the OpenPose project. The primary goal of OpenPose is to detect the 2D positions of body joints (keypoints) in images or video frames, representing the human pose.
We used this library to get the keypoints of person's left, right hand in the person-image; and a skeleton map of that person. See the below figure for some results.

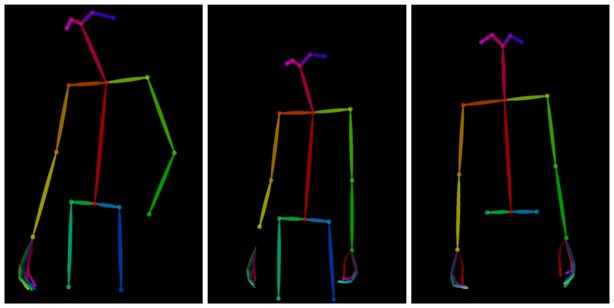


Figure 5: OpenPose Results

4.2.3 DensePose

DensePose is a computer vision project and model developed by Facebook AI Research (FAIR) that focuses on dense human pose estimation. Dense pose estimation involves mapping every pixel in an image to a corresponding 3D surface on the human body. In simple terms, it aims to provide a detailed understanding of the pose and surface geometry of the human body in images or videos.

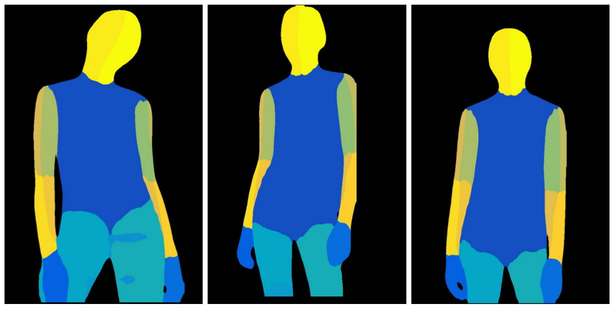


Figure 6: DensePose Results

4.2.4 Cloth Mask

At this step, the author used a model from *CarveKit* to get the shape of the clothing item from cloth-image. After checking the author's pre-processed dataset again, we found out that the CarveKit model didn't get the right shape of the clothing item. So we decided to use the same library as the *Remove Background* step to process. The figure below shows the difference between our method and the authors.

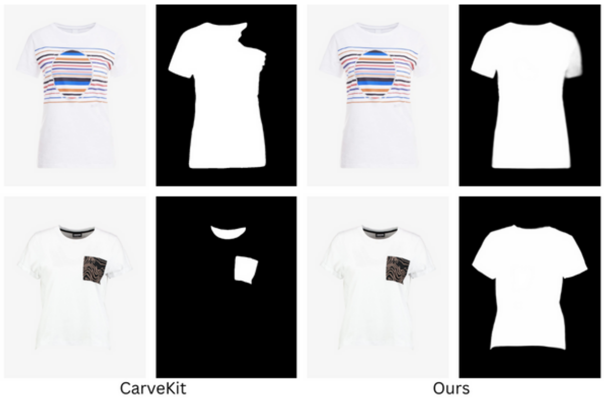


Figure 7: Cloth Mask Comparison

4.2.5 Human Parse

Just like the author, we also used CIHP_PGN to get the parse map from image-person which contains 20 unit values from 0 to 19. However, the author used an older version using *TensorFlow 1.15* running on *Python 3.7*, so we updated it to *TensorFlow 2.0* running on *Python 3.8*.

CIHP_PGN stands for Context-aware Instance Part Grouping Network. It is a deep learning method for semantic part segmentation, instance-aware edge detection, and instance-level human parsing. It was developed by Engineering-Course and presented at the European Conference on Computer Vision (ECCV) in 2018.
CIHP_PGN is built on top of *TensorFlow* and is designed to be efficient and accurate. It uses a novel context-aware instance part grouping mechanism to group together related parts of an object. This mechanism helps to improve the accuracy of the segmentation and edge detection tasks.
CIHP_PGN has been evaluated on several benchmark datasets and has achieved state-of-the-art results. It is a promising method for human parsing and has the potential to be applied to a wide range of other computer vision tasks.

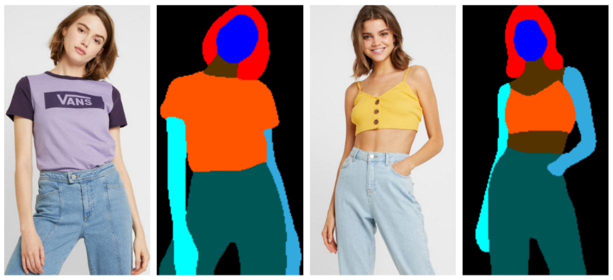


Figure 8: Human Parsing Results

4.2.6 Parse Agnostic

In this step and the *Human Agnostic* we used the original image-person, the parse map from *Human Parse* and the person's keypoints to process.

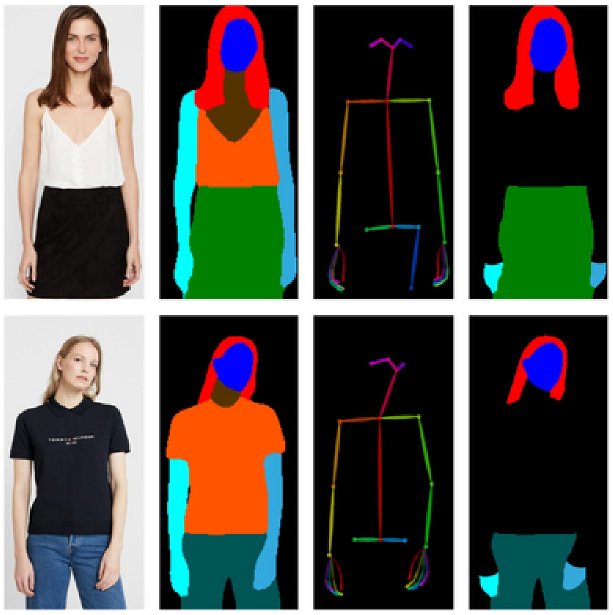


Figure 9: Parse Agnostic Results

4.2.7 Human Agnostic

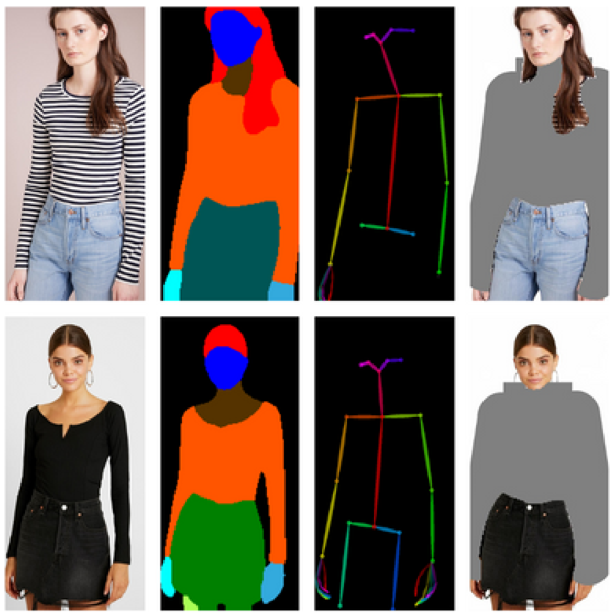


Figure 10: Human Agnostic Results

4.3 Training Metrics

4.3.1 Generator Loss

"Generator Loss" or "G Loss" in a Generative Adversarial Network (GAN) model is the loss associated with the generator. The goal of the generator is to create new data that closely resembles real data. To measure the discrepancy between real and generated data, a loss function is employed.
The Generator Loss typically consists of multiple components, such as Adversarial Loss, Image Loss, and other terms depending on the model design. Here is a general representation of the Generator Loss:

$$\text{Generator Loss} = \text{GAN Loss} + \text{Other Loss Terms}$$

GAN Loss. The GAN Loss often relies on the Cross Entropy Loss, measuring the generator's ability to "fool" the discriminator. Specifically, if D is the probability that a sample is real, and G is the probability that a sample is generated by the generator, the GAN Loss can be expressed as: GAN Loss = -log(D(G(z))), where z is a random input vector for the generator.

Other Loss Terms. Other components of the Generator Loss may include Image Loss to ensure similarity to real data, regularization components like L1 or L2 loss to control model complexity, and other terms depending on specific problem requirements.

In summary, the Generator Loss sets up an optimization problem where the generator produces data that deceives the discriminator while keeping this data close to real data. Optimizing the Generator Loss leads to an effective generator model that generates high-quality new data.

4.3.2 L1_cloth Loss

The L1_cloth Loss, similar to many loss functions in machine learning, is a part of the model training process designed to measure the difference between the model's predictions and the actual data in the context of clothing or fashion images.

L1 Norm. L1_cloth Loss typically uses the L1 norm, also known as the Manhattan Norm. The formula for calculating the L1 norm between two vectors a and b is:

$$\text{L1 norm(a, b)} = \sum_i \left| a_i - b_i \right|,$$

in the case of L1_cloth loss, a and b could be two vectors containing pixel values of the actual and predicted images, respectively.

L1_cloth Loss Formula. The formula calculates the average absolute difference between the pixel values of the actual and predicted images over the entire image.

$$\text{L1\_cloth Loss} = \frac{1}{N} \sum_{i=1}^{N} \left| I_{real_i} - I_{fake_i} \right|,$$

where N is the number of pixels in each image, I_{real_i} is the corresponding pixel value in the actual image at position i, I_{fake_i} is the corresponding pixel value in the predicted image at position i, the $\left| ... \right|$ denotes the absolute value.

Significance. L1_cloth Loss measures the "magnitude" of the difference between the actual and predicted images. In this way, the model is trained to minimize this difference during training, leading to the generation of predicted images that closely resemble the actual images. L1_cloth Loss is often used in generative models for fashion or clothing images to ensure that the generated images match well with the real images.

4.3.3 Visual Geometry Group Loss

Visual Geometry Group (VGG) Loss is frequently utilized in the training process of machine learning models, particularly in generative image models, to quantify the disparity between the features of generated images and real images.

VGG Loss. In the context of VGG Loss, a specialized model known as the VGG model, initially trained for image classification into different classes, is employed. However, in this scenario, we leverage it as a feature extractor. Crucially, we do not use the VGG model for classification, instead we utilize the averaged features extracted from selected layers.

VGG Loss Computing Formula. The formula for VGG Loss is often expressed as follows:

$$\text{VGG Loss} = \sum_i w_i \cdot \frac{1}{N_i} \sum_j \left\| \phi_i \left(I_{real_j} \right) - \phi_i \left(I_{fake_j} \right) \right\|_2^2,$$

where i is the index of the layers we are interested in from the VGG model, w_i is the weight applied to each layer, N_i is the number of features in layer i, $\phi_i \left(I_{real_j} \right)$ and $\phi_i \left(I_{fake_j} \right)$ are features extracted from layer i of the actual images and the corresponding predicted image, the $\left\| ... \right\|_2^2$ is the square of the L_2 norm between the features.

Significance. VGG Loss gauges the similarity between features extracted from real and predicted images. Utilizing features at a high level ensures that the most crucial features for image recognition are preserved and optimized during model training. VGG Loss is commonly employed in generative models for fashion or clothing images to guarantee that the generated images closely resemble the real images.

4.3.4 Total Variation Loss

Total Variation (TV) Loss is often employed during the training of generative image models to reduce noise and enhance the smoothness of generated images.

TV Loss. TV Loss is designed to diminish variability within an image, promoting the creation of smoother and more continuous images. This helps to mitigate noise and generate smoother images.

TV Loss Computing Formula. The formula for TV Loss for an image I can be expressed as follows:

$$\text{TV Loss(I)} = \textstyle\sum_{i,j} \left(\|I_{i+1,j} - I_{i,j}\|_2 + \|I_{i,j+1} - I_{i,j}\|_2 \right),$$

where $I_{i,j}$ denotes the pixel value at position (i,j) in the image, the $\|...\|_2$ represents the L_2 norm, or Euclidean norm, signifying the Euclidean distance, *i+1, j* and *i,j+1* correspond to neighboring positions in the horizontal and vertical directions.

Significance. TV Loss helps reduce noise in images by aiming to minimize the variability between adjacent pixels. This facilities the creation of images with continuous and uninterrupted characteristics, suitable for applications requiring smooth and continuous image generation.

4.3.5 Cross Entropy (CE)

In the context of the clothing swapping problem, Cross Entropy (CE) Loss is commonly used to measure the level of "surprise" between the real image of the model and the image generated by the model. Below is a detailed analysis and explanation of how Cross Entropy Loss is applied in this problem.

CE Loss. The CE Loss between the real image (I_{real}) and the generated image (I_{fake}) can be expressed by the following formula:

$$\text{CE Loss}(I_{real}, I_{fake}) = -\textstyle\sum_{i,j} I_{real(i,j)} - log \left(I_{fake(i,j)} \right),$$

where i and j represent pixel positions in the image, $I_{real(i,j)}$ is the pixel value at position (i,j) in the real image, and $I_{fake(i,j)}$ is the corresponding pixel value in the generated image.

Significance. The goal of the model is to generate an image (I_{fake}) that closely resembles the real image (I_{real}). CE Loss measures the difference between the real pixel values and the predicted pixel values. If a pixel has a high value in the real image but is predicted with low probability by the model, the CE Loss will be high. In the clothing swapping problem, CE Loss helps the model learn how to generate clothes with colors, styles, and details that closely match the real images. This loss supports the training process by optimizing the model parameters to minimize the discrepancy between the real and generated images.

4.3.6 G_{GAN}

In the context of a Generative Adversarial Network (GAN) model, G_{GAN} is commonly understood as a component of the loss function for the generator (G).

Adversarial Loss. In a GAN model, there are two main components: the generator (G) and the discriminator (D). G_{GAN} is a part of the optimization objective for the generator. The task of G is to generate new data to closely resemble real data, thereby deceiving the discriminator.

Adversarial Loss Computing Formula. The Adversarial Loss function often relies on CE Loss and measures the ability of the discriminator to be fooled by the generator. If D is the probability that a sample is real, and G is the probability that a sample is generated by G, the formula for G_{GAN} can be expressed as follows:

$$G_{GAN} = \text{-log(D(G(z)))},$$

where z is a random vector used to generate new data by G, and *D(G(z))* is the probability that the discriminator evaluates the sample as real.

Significance. G_{GAN} measures the ability of the generator to produce data that the discriminator cannot distinguish from real data. The goal is to make G generate data that "deceives" D effectively. G_{GAN} is a crucial part of the generator's loss function during GAN training. Optimizing G_{GAN} implies improving the ability to generate data that the discriminator cannot differentiate effectively. The formula above can be integrated into the overall loss function for the generator during GAN training.

4.3.7 Discriminator Loss

In the context of a GAN model, "loss D" refers to the loss associated with the discriminator. The main task of the discriminator is to distinguish between real and generated data by the generator.

Loss D. In GANs, the discriminator loss typically consists of two components: the loss from discriminating real data and the loss from discriminating generated data. The goal of the discriminator is to maximize this loss to accurately distinguish between real and generated data.

Loss D Computing Formula. The general formula for the discriminator loss if often expressed as follows:

$$\text{Loss D} = \text{-(log(D(data)) + log(1-D(G(z))))},$$

where *D(data)* is the probability that the discriminator evaluates a sample as real, *G(z)* is the data generated by the generator from a random vector *z*, *1-D(G(z))* is the probability that the discriminator evaluates a sample as fake (generated data).

Significance. The discriminator loss measures its ability to distinguish between real and generated data. The objective is to maximize this loss to make the discriminator increasingly accurate. The formula above represents the general discriminator loss during GAN training.

4.3.8 D_{real} in GAN

"D_{real}" in a GAN model represents the probability that the discriminator evaluates a sample as real data, indicating that it belongs to the real dataset.

What is D_{real}? "D_{real}" is a representation of the probability that the discriminator considers a sample as real, and it is a crucial part of the GAN training process. In the context of the GAN problem, this is the probability that the discriminator correctly evaluates a sample as belonging to the real dataset.

D_{real} **Computing Formula.** The probability "D_{real}" is typically calculated using the probability provided by the discriminator for a sample from real-world data (D(data)). The detailed formula is:

$$D_{real} = \mathrm{D(data)},$$

where D_{real} is the probability that a sample is evaluated as real by the discriminator, *D(data)* is the probability that the discriminator evaluates a sample as real from real-world data.

Significance. The probability "D_{real}" plays a crucial role in GAN training as it serves as a quality indicator for the discriminator. Maximizing this probability helps ensure that the discriminator can effectively recognize and distinguish real data from data generated by the generator. The equation above represents the probability of real data (D_{real}) in the GAN model and is a critical factor in the training process to enhance the model's performance.

4.3.9 D_{fake} in GAN

What is D_{fake}? "D_{fake}" is a representation of the probability that the discriminator considers a sample as fake, meaning it is generated by the generator rather than belonging to the real dataset. In the GAN model, "D_{fake}" evaluates the degree of fakeness of the generated data.

D_{fake} **Computing Formula.** The probability "D_{fake}" is typically calculated using the probability provided by the discriminator for a sample generated (D(G(z))), where z is the random vector used to generate data. The detailed formula is:

$$D_{fake} = \mathrm{D(G(z))},$$

where D_{fake} is the probability that a sample is evaluated as fake by the discriminator, *D(G(z))* is the probability that the discriminator evaluates a sample as fake from generated data.

Significance. "D_{fake}" is a crucial indicator in GAN training as it assesses the discriminator's ability to distinguish between real and generated data.

4.4 Evaluation Metrics

4.4.1 Structural Similarity Index Measurement (SSIM)

The Structural Similarity Index Measurement (SSIM) is a method for analyzing image quality, designed to measure the structural similarity between two images. The main formula for SSIM is:

$$\text{SSIM(x,y)} = \text{l(x,y)} \cdot \text{c(x,y)} \cdot \text{s(x,y)},$$

where *l(x,y)* is the luminance factor, *c(x,y)* is the contrast factor, *s(x,y)* is the structure factor.

Specifically, each factor is calculated as follows:

Luminance (l(x,y)):

$$\text{l(x,y)} = \frac{2\cdot\mu_x\cdot\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1},$$

where μ_x and μ_y are the mean values of x and y respectively, and C_1 is a constant to avoid division by zero.

Contrast (c(x,y)):

$$\text{c(x,y)} = \frac{2\cdot\sigma_x\cdot\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2},$$

where σ_x and σ_y are the standard deviations of x and y respectively, and C_2 is a constant.

Structure (s(x,y)):

$$\text{s(x,y)} = \frac{\sigma_{xy}+C_3}{\sigma_x\cdot\sigma_y+C_3},$$

where σ_{xy} is the covariance between x and y, and C_3 is a constant.

The constant C_1, C_2, and C_3 are used to avoid division by zero and adjust the sensitivity of the SSIM index. Typically, their values are chosen to ensure the sensitivity of SSIM to small changes in images.

4.4.2 Mean Squared Error (MSE)

The Mean Squared Error (MSE) is a common metric used to measure the difference between predicted and actual values in machine learning and statistics. The primary significance of MSE lies in quantifying the magnitude of errors and providing an overall view of a model's predictive performance.

Squared Difference. MSE measures the squared difference between each predicted and actual value. Squaring enhances the influence of large errors, particularly useful when concerned with significant deviations.

Meaningful Average. MSE averages the squared difference across the entire dataset, producing a single value representing the average difference between predicted and actual values.

Square Unit. MSE has units in the square of the unit of the measured variable (e.g., if the measurement is in thousands of dollars, MSE will be in thousands of square dollars).

MSE Computing Formula. The formula to calculate MSE between two datasets Y and $\hat{Y}$ is:

$$\mathrm{MSE}(\mathrm{Y},\hat{Y}) = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2,$$

where n is the number of samples in the dataset, Y_i is the actual value at sample i, $\hat{Y}_i$ is the corresponding predicted value at sample i.

Pros and Cons.

- Pros: MSE focuses on large errors, making it suitable for problems where significant deviation matter.
- Cons: MSE is sensitive to outliers, as squaring them increases their contribution to the overall value.

4.4.3 Learned Perceptual Image Patch Similarity (LPIPS)

Learned Perceptual Image Patch Similarity (LPIPS) is a metric designed to measure the similarity between two images based on human perception. It focuses on comparing small patches of images rather than the overall similarity of the entire image.

Patch-wise Comparison. LPIPS concentrates on comparing each patch of two images instead of comparing the entire images. This approach simulates how humans typically perceive and compare images.

Perceptual Features. LPIPS models often use Convolutional Neural Networks (CNNs) to extract perceptual features from small image patches. These features are often related to the structure, color, and other factors in the image that humans commonly recognize.

Learned Weights. The model's weights are trained to optimize the similarity between patches. These weights are used in the computation process to evaluate the similarity between patches. Typically, these weights are learned from training data to reflect human perception.

Applications and Usage.

- Image Quality Assessment: LPIPS is commonly used to assess image quality and compare the performance of different image generation models.
- Image Generation Models: In image generation tasks, LPIPS can be used to measure the perceptual similarity between generated images and real images.
- Model Comparison and Selection: LPIPS helps in comparing and selecting models based on their ability to reproduce important perceptual features recognized by humans.

LPIPS Computing Formula. The main formula of LPIPS can be expressed as the weighted sum of the differences between perceptual features of two patches:

$$\text{LPIPS(x,y)} = \textstyle\sum_i w_i \cdot \phi_i(x) \cdot \phi_i(y),$$

where x and y are two patches to be compared, $\phi_i(x)$ and $\phi_i(y)$ are the perceptual features of patch x and y, w_i is the weight of the *i-th* feature.

Summary. LPIPS is a multidimensional and perceptual metric that measures the similarity between images from a human perspective. Focusing on small image patches helps it authentically reflect how we perceive and evaluate images.

4.4.4 Inception Score Mean (IS_mean)

Inception Score (IS) is a metric commonly used to evaluate the quality of generated images in the field of generative models, especially in the context of Generative Adversarial Networks (GANs).

Inception Score Mean (IS_mean). The IS_mean, or Inception Score Mean, is a variation of the Inception Score that provides a mean value across multiple generated samples. The Inception Score is commonly defined as follows:

$$\text{IS(G)} = exp\left(E_{x \sim G}\left[D_{KL}\left(P\left(y|x\right)||P\left(y\right)\right)\right]\right),$$

where G is the generative model, x is a generated image, y is the class label predicted by the Inception model, $P\left(y|x\right)$ is conditional class distribution, and *P(y)* is the marginal class distribution.

Individual Image Quality. The quality of an individual image is often measured by the maximum predicted class probability. Let $P_{max}(y|x)$ be the maximum class probability for an image x. Higher $P_{max}(y|x)$ indicates higher quality.

Diversity Across Images. The diversity is assessed by calculating the entropy of the class probabilities across all generated images:

$$\text{H(P(y—G))} = -\sum_x \sum_y P(y|x) log_2(P(y|x))$$

IS_mean Computing Formula. For N sets of generated images G_1, G_2,..., G_N, the IS_mean is calculated as the mean of the individual Inception Scores:

$$IS_{mean} = \frac{1}{N} \sum_{i=1}^{N} IS(G_i)$$

Interpretation of IS_mean. A higher IS_mean indicates that the generative model consistently produces sets of images that are both of high quality and diverse.

4.4.5 Inception Score - Standard Deviation (IS_std)

For IS_std, we incorporate the measure of uniformity (variability) of the Inception scores. It provides information about the level of fluctuation or non-uniformity among the Inception Scores for each image.

IS_std Computing Formula. To calculate IS_std, first compute the Inception Score for each image in the generated dataset. Then, calculate the standard deviation of these Inception scores. The detailed formula can be expressed as follows:

$$\text{IS\_std} = \text{std}(\text{IS}(\text{generated\_images})),$$

where *IS_std* is the Inception Score - Standard Deviation, *std* is the standard deviation function, *IS(generated_images)* is the Inception Score calculated for the entire set of generated images.

Significance. IS_std measures the level of variability in quality and diversity among the generated images. If IS_std is low, it may indicate that the generator produces images with relatively stable quality and diversity. Conversely, a high IS_std may indicate unevenness in quality and diversity among the images.

5 RESULTS

5.1 Experiments

As mentioned before, we tried running the model again based on the data that was processed by our method. There have been a few cases with better results when running on data provided by the author. The figures below show the difference between the original HR-VITON [23] model and ours.

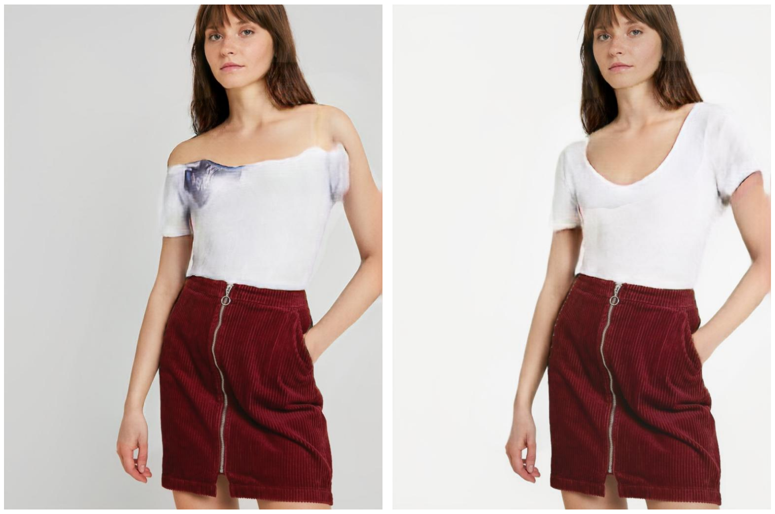


Figure 11

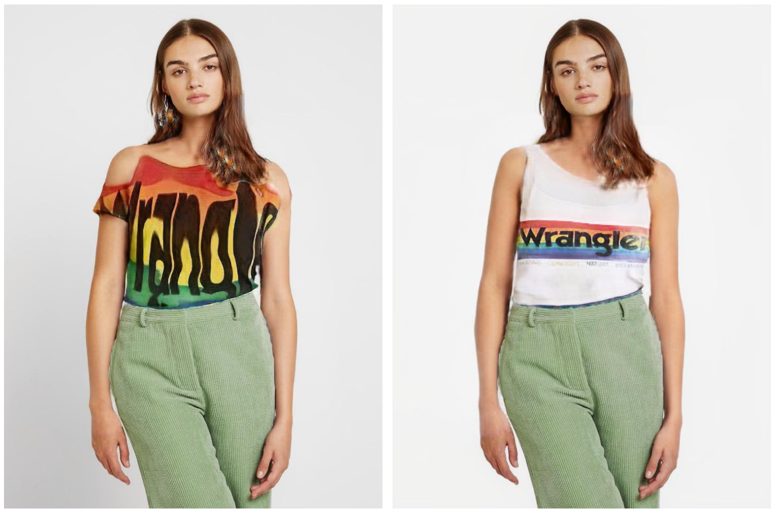


Figure 12

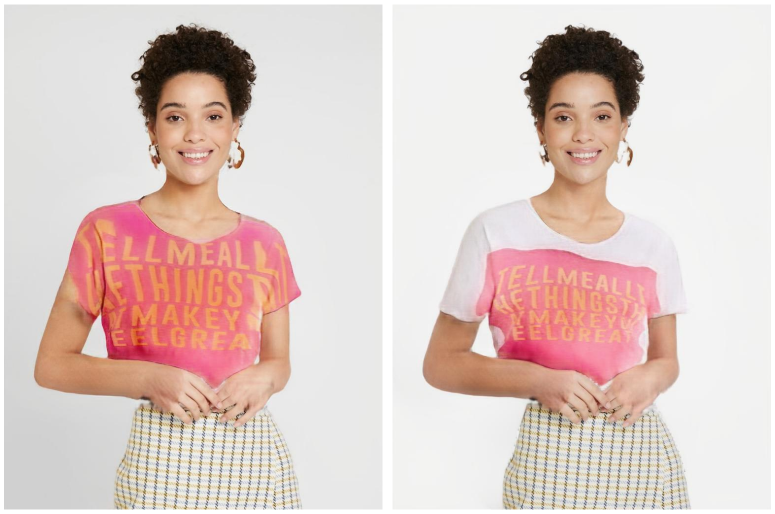


Figure 13

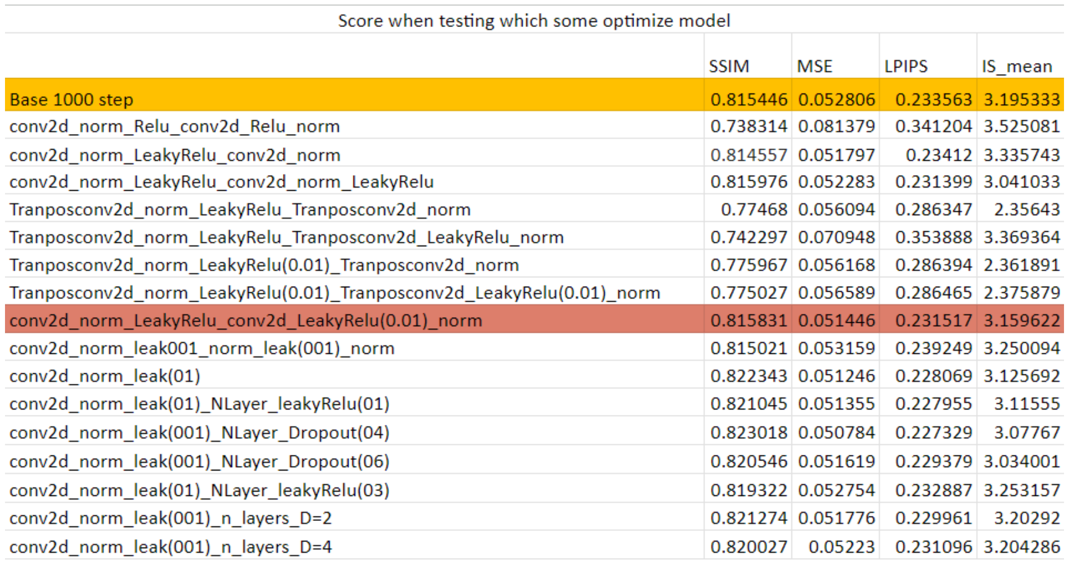
We also re-ran the evaluation metrics and got results approximately the same as the author's.

	HR-VITON	Ours
SSIM	0.82591	0.821920
MSE	0.050458	0.051742
LPIPS	0.230576	0.229801
IS_mean	3.189770	3.227256

Table 2: Metrics Comparison

5.2 Optimization

The primary objective of this part is to optimize HR-VITON [23]model specifically, we perform the optimizer in try-on condition generator section. The reason why we only focus on the try-on condition generator, because during the research process, we found this part to be the most important part of the entire project cause the outputs of this step are the shirt shape after being redrawn to fit the person, so choosing this part to optimize will gave the most direct and obvious impact on the entire model architecture. We have verified the changes ideals by retraining the entire model and checking the output metrics over 1000 steps and have obtained a preliminary assessment as follows:



Score when testing which some optimize model

	SSIM	MSE	LPIPS	IS_mean
Base 1000 step	0.815446	0.052806	0.233563	3.195333
conv2d_norm_Relu_conv2d_Relu_norm	0.738314	0.081379	0.341204	3.525081
conv2d_norm_LeakyRelu_conv2d_norm	0.814557	0.051797	0.23412	3.335743
conv2d_norm_LeakyRelu_conv2d_norm_LeakyRelu	0.815976	0.052283	0.231399	3.041033
Tranposconv2d_norm_LeakyRelu_Tranposconv2d_norm	0.77468	0.056094	0.286347	2.35643
Tranposconv2d_norm_LeakyRelu_Tranposconv2d_LeakyRelu_norm	0.742297	0.070948	0.353888	3.369364
Tranposconv2d_norm_LeakyRelu(0.01)_Tranposconv2d_norm	0.775967	0.056168	0.286394	2.361891
Tranposconv2d_norm_LeakyRelu(0.01)_Tranposconv2d_LeakyRelu(0.01)_norm	0.775027	0.056589	0.286465	2.375879
conv2d_norm_LeakyRelu_conv2d_LeakyRelu(0.01)_norm	0.815831	0.051446	0.231517	3.159622
conv2d_norm_leak001_norm_leak(001)_norm	0.815021	0.053159	0.239249	3.250094
conv2d_norm_leak(01)	0.822343	0.051246	0.228069	3.125692
conv2d_norm_leak(01)_NLayer_leakyRelu(01)	0.821045	0.051355	0.227955	3.11555
conv2d_norm_leak(001)_NLayer_Dropout(04)	0.823018	0.050784	0.227329	3.07767
conv2d_norm_leak(001)_NLayer_Dropout(06)	0.820546	0.051619	0.229379	3.034001
conv2d_norm_leak(01)_NLayer_leakyRelu(03)	0.819322	0.052754	0.232887	3.253157
conv2d_norm_leak(001)_n_layers_D=2	0.821274	0.051776	0.229961	3.20292
conv2d_norm_leak(001)_n_layers_D=4	0.820027	0.05223	0.231096	3.204286

Figure 14: Score testing on 1000 steps

The figure above shows the results of testing some optimization ideals by changing a flew layers and editing some hyper parameters. After a preliminary evaluation, we conducted to see if the newly output are more beautiful and detailed than original structure and was concluded that using LeakyReLU would gave a better results and get higher evaluation metrics than the original.

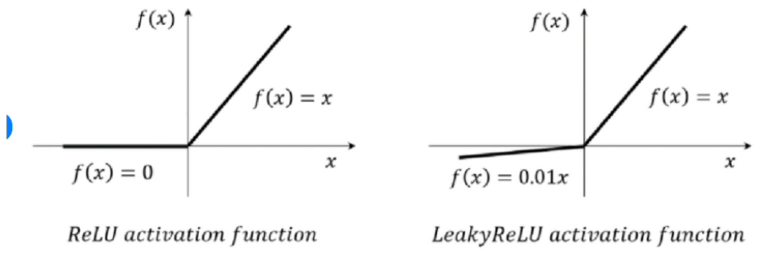


Figure 15: ReLU vs LeakyReLU

The reason choosing LeakyReLU, LeakyReLU is often chosen over ReLU in machine learning model design because it addresses the "dying ReLU" issue by allowing for small negative values. LeakyReLU is also more flexible in handling negative values and may help reducing the risk of over-fitting. That lead to the encoder blocks can extract lots of pixel than using ReLU at the activation function.

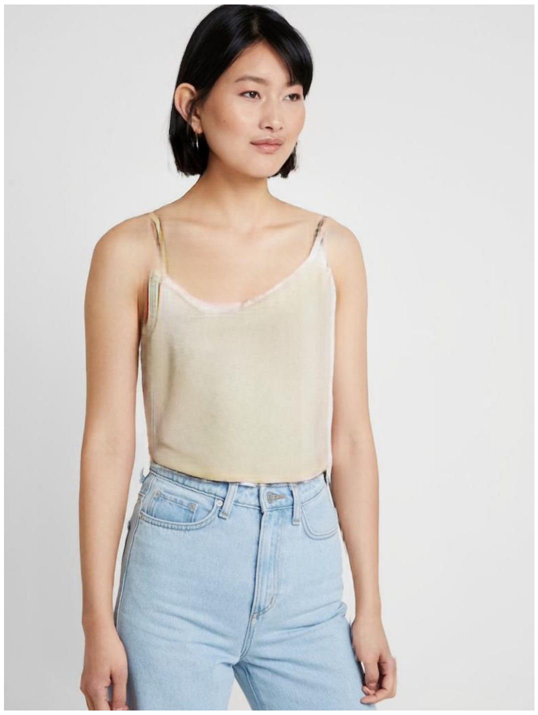


Figure 16: Retrain 1000 steps on original model

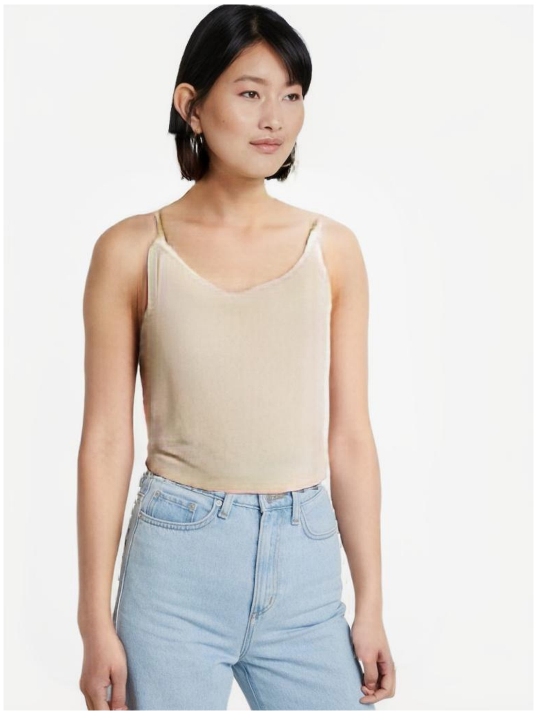


Figure 17: Retrain 1000 steps with LeakyReLU

5.3 Training on pre-train checkpoints

Initially, we planned to retrain the entire model from scratch with the input datas are image-person and cloth-image provided by the author and remaining input such as DensePose, OpenPose, etc. taken from our pre-processing steps and put all required parts and start retraining model from scratch. However, during the process we encountered a problem of not having enough hardware to be able to retrain with the same settings as the author gave us. The solution we offer to solve this problem is that instead of retraining from scratch we will train on the pre-train checkpoints provided by the author on 3000 steps per turn. The figure below shows our results.

Table 3: 30000 steps training loss

STEP	loss_G	L1_cloth	VGG	TV	CE	G_GAN	loss_D	D_real	D_fake
3000	2.5698	0.0284	0.7785	0.3755	0.0248	0.5085	1.0031	0.4998	0.5033
6000	2.2659	0.0188	0.7176	0.3109	0.0211	0.5274	1.0061	0.5272	0.4790
9000	2.2266	0.0206	0.7617	0.3033	0.0170	0.4821	1.0225	0.4819	0.5406
12000	2.1098	0.0148	0.5683	0.3503	0.0161	0.5311	1.0070	0.5249	0.4821
15000	2.2145	0.0169	0.7119	0.3292	0.0177	0.4989	1.0180	0.4979	0.5201
18000	2.1277	0.0193	0.7142	0.3147	0.0216	0.3756	1.0685	0.3825	0.6860
21000	2.2928	0.0154	0.7064	0.3628	0.0121	0.5863	1.0319	0.5852	0.4467
24000	2.3089	0.0164	0.7624	0.3172	0.0164	0.5846	1.0217	0.5815	0.4401
27000	2.0604	0.0139	0.6622	0.3350	0.0133	0.4565	1.0187	0.4618	0.5569
30000	1.9883	0.0107	0.5643	0.3667	0.0135	0.4491	1.0431	0.4471	0.5960

Table 4: Retrain 30000 steps results

STEP	SSIM	MSE	LPIPS	IS_mean
3000	0.812811	0.054265	0.241237	3.108629
6000	0.813482	0.053982	0.239779	3.124469
9000	0.813805	0.053811	0.2389	3.128264
12000	0.812789	0.05404	0.240045	3.265919
15000	0.822069	0.052957	0.229872	3.265919
18000	0.82195	0.053811	0.226971	3.293342
21000	0.815857	0.051024	0.237209	3.294366
24000	0.816077	0.052957	0.236621	3.305738
27000	0.822359	0.053132	0.236827	3.310482
30000	0.822001	0.053142	0.235957	3.311385

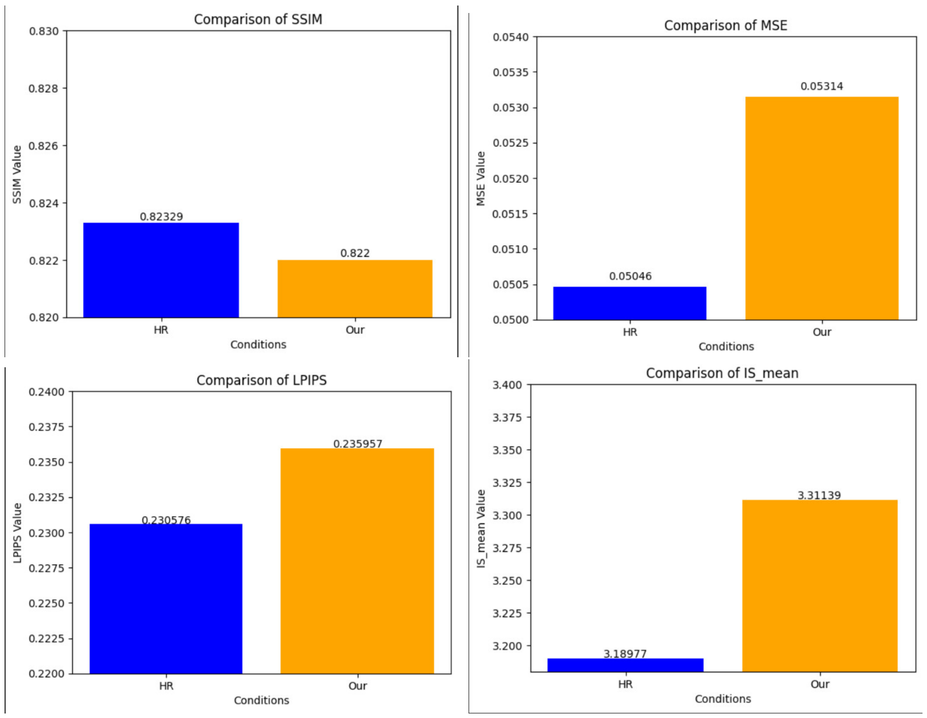


Figure 18: Compare evaluation metrics

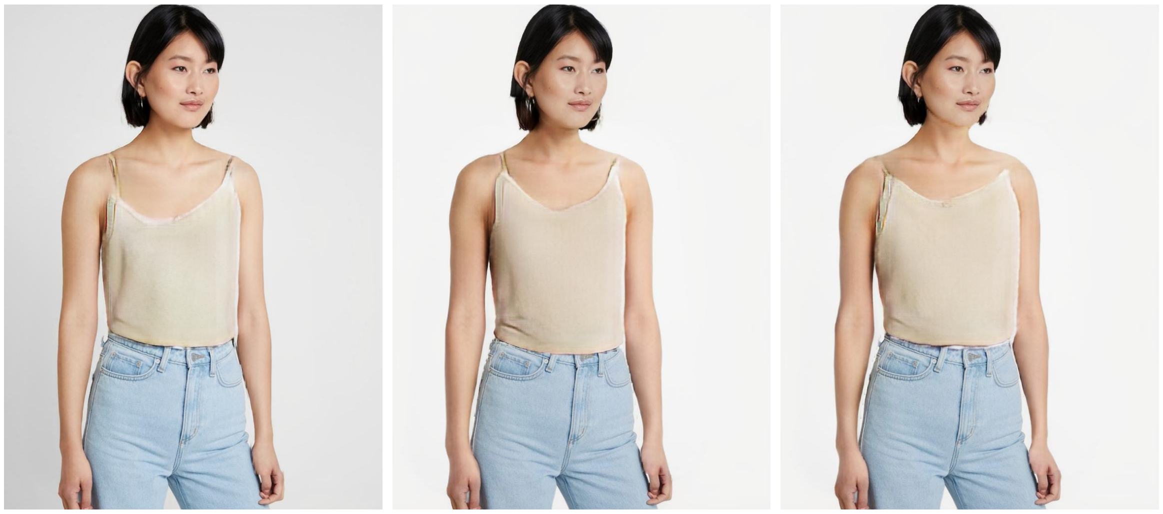


Figure 19: Compare results between the original, 1000 steps and 30000 steps

5.4 Fail case analysis

During the process of training HR-VITON [23]model to generate images of clothing item on the person's body, some specific problems appeared, causing unexpected results during the image generation process. Below is a detailed descriptions of these issues:

- **Incomplete display of both arms due to poor generation results:** One significant challenge the model is facing is its inability to generate sufficiently high-quality images that display both arms of the model. This may result from inaccuracies in the image generation process, possibly stemming from the model's failure to learn the complex relationships between different parts of the body.
- **Inability to handle the length of the try-on shirt:** Another issue is the model's incapacity to adapt to variations in the shape and length of the try-on shirt. If the model is ineffective in mapping the relationship between the model's body shape and the necessary length of the shirt, the generated results may not accurately reflect the clothing effect on the body.
- **Poor quality of input images:** An important challenge faced during the deployment of the model is the low quality of the input images. If the images are of low quality, containing noise or blur, the model will struggle to learn essential features of both the clothing and the model's body.

- Noise and blur degrade performance: If the input images contain noise or are blurry, the model will encounter difficulties in detecting and accurately reproducing the details of the clothing and the body. This can lead to the generation of low-quality images that do not reflect the real-world state accurately.
- Insufficient information: Low-quality images may also result in the loss of crucial information, reducing the model's capability to learn the complex relationships between the try-on clothes and the model.

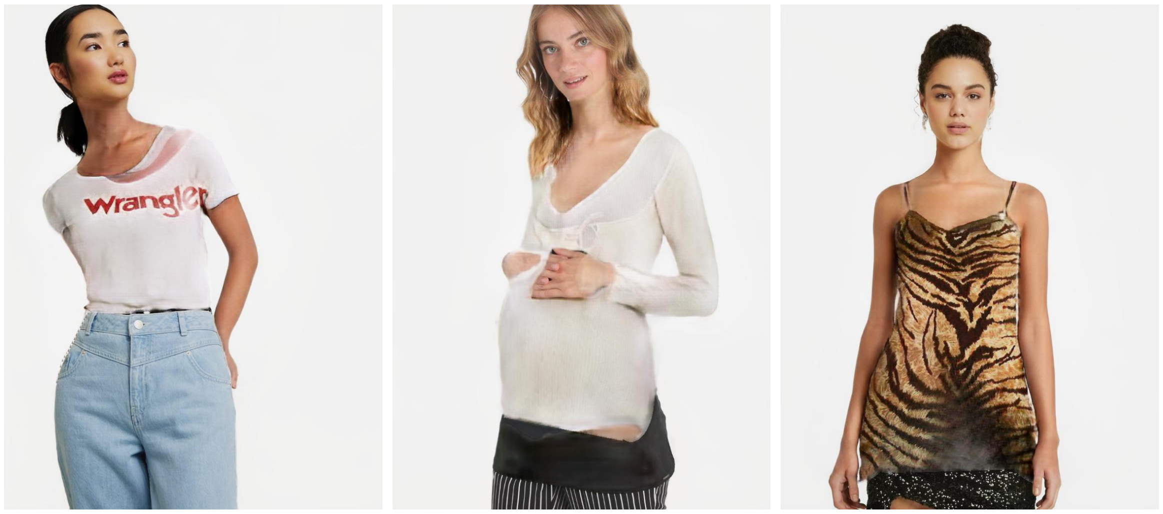


Figure 20: Bad cases: (1) Missing body parts, (2) Person's shirt too long, (3) Cannot handle dress, jacket, etc.

6 DISCUSSIONS

6.1 Experimental Results

Over the past 14 weeks, we have conducted error testing, tested the data set and the original input processing steps that the author gave us, and retested the input processing steps we gave. I did it myself then also tried optimizing the model in the try-on conditions section and got some positive results when running the test on the image created after running the test with the processing steps. Our input processing results show that the generated image has a few points better than the original result of the original image. However, because the time was only 14 weeks, our optimization was not completely completed, specifically we only had enough time to test the optimization by only being able to retrain with 30,000 steps is only 1/10 of the author's, so our optimization evaluation is not very good, but with some results compared to the author's, we also provide some positive evidence for the optimization step. ours. We also built a website for demos, but the total running time is still quite slow because to be able to run the final result, the image will have to go through a few initial processing steps.

6.2 User Interaction and Adaptability

A significant strength of our virtual try-on system lies in its natural and flexible user interaction. Users can seamlessly try on various clothing items without altering their primary image, enhancing the overall online shopping experience. The system demonstrated versatility in adapting to different types of clothing, spanning from formal wear to athletic attire, making it a comprehensive and adaptable virtual fitting room.

6.3 Challenges and Future Enhancements

Although some initial steps have been taken to improve the quality of the images produced, some challenges have also been identified, as mentioned above in some cases where the input images are bad or some If parts such as hands are obscured, the image processing steps do not provide the necessary conditions to be included in the main model, so the image created will not be of good quality. or jacket,... will also affect the performance of the entire model.

7 CONCLUSIONS

Through running and experimenting over a period of 14 weeks, we have seen the good points and limitations of the model as mentioned above. Some directions for development to improve image quality that we suggest are to continue letting the model learn with the optimization we gave above and continue to go deeper into the try-on part. conditions generator to be able to change the mathematical part that the author researched in creating the image of the shirt after being redrawn to fit the model while continuing to research on improving the application part to be able to reduce get runtime to optimize system resources. In the future, if we have the opportunity, we will continue to test models for other types of clothing such as pants, skirts, dresses, jackets, etc.

References

[1] Peng Guan et al. “Drape: Dressing any person”. *ACM Transactions on Graphics (ToG)* 31.4 (2012), pp. 1–10.

[2] Masahiro Sekine et al. “Virtual fitting by single-shot body shape estimation”. *Int. Conf. on 3D Body Scanning Technologies*. Citeseer. 2014, pp. 406–413.

[3] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[4] Nikolay Jetchev and Urs Bergmann. “The conditional analogy gan: Swapping fashion articles on people images”. *Proceedings of the IEEE international conference on computer vision workshops*. 2017, pp. 2287–2292.

[5] Jae Hyun Lim and Jong Chul Ye. “Geometric gan”. *arXiv preprint arXiv:1705.02894* (2017).

[6] Xudong Mao et al. “Least squares generative adversarial networks”. *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2794–2802.

[7] Gerard Pons-Moll et al. “ClothCap: Seamless 4D clothing capture and retargeting”. *ACM Transactions on Graphics (ToG)* 36.4 (2017), pp. 1–15.

[8] Samaneh Azadi et al. “Discriminator rejection sampling”. *arXiv preprint arXiv:1810.06758* (2018).

[9] Ke Gong et al. “Instance-level human parsing via part grouping network”. *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 770–785.

[10] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. “Densepose: Dense human pose estimation in the wild”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7297–7306.

[11] Xintong Han et al. “Viton: An image-based virtual try-on network”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7543–7552.

[12] Takeru Miyato et al. “Spectral normalization for generative adversarial networks”. *arXiv preprint arXiv:1802.05957* (2018).

[13] Bochao Wang et al. “Toward characteristic-preserving image-based virtual try-on network”. *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 589–604.

[14] Ting-Chun Wang et al. “High-resolution image synthesis and semantic manipulation with conditional gans”. *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2018, pp. 8798–8807.

[15] Taesung Park et al. “Semantic image synthesis with spatially-adaptive normalization”. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2019, pp. 2337–2346.

[16] Ruiyun Yu, Xiaoqi Wang, and Xiaohui Xie. “Vtnfp: An image-based virtual try-on network with body and clothing feature preservation”. *Proceedings of the IEEE/CVF international conference on computer vision.* 2019, pp. 10511–10520.

[17] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. “Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style”. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2020, pp. 7365–7375.

[18] Han Yang et al. “Towards photo-realistic virtual try-on by adaptively generating-preserving image content”. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2020, pp. 7850–7859.

[19] Seunghwan Choi et al. “Viton-hd: High-resolution virtual try-on via misalignment-aware normalization”. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2021, pp. 14131–14140.

[20] Ayush Chopra et al. “Zflow: Gated appearance flow-based virtual try-on with 3d priors”. *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pp. 5433–5442.

[21] Kathleen M Lewis, Srivatsan Varadharajan, and Ira Kemelmacher-Shlizerman. “Vogue: Try-on by stylegan interpolation optimization”. *arXiv preprint arXiv:2101.02285* 1.2 (2021).

[22] Kedan Li et al. “Toward accurate and realistic outfits visualization with attention to details”. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2021, pp. 15546–15555.

[23] Sangyun Lee et al. “High-resolution virtual try-on with misalignment and occlusion-handled conditions”. *European Conference on Computer Vision.* Springer. 2022, pp. 204–219.