# Virtual Try-On: Final Project Report

Francesco Martucci, Felicia Puzone, Francesco Sala
{*243215, 312722, 253589*} @studenti.unimore.it Github project repository:
https://github.com/felicia-puzone/virtual-try-on-app

**Abstract**
*Our project will be focusing on designing a 2D image-based virtual try on system. Virtual try on consists in generating an image of a reference person wearing a given try-on garment. This kind of problem is usually solved with a two-stage approach, incorporating at least both a geometric transformation module to warp the selected garment and a generative try-on module to reconstruct the realistic try-on image given the person representation and the warped cloth. We propose a complete pipeline built on this system performing in-the-wild virtual-try-on, consisting in image enhancing, background removal, a content-based retrieval system, cloth warping & try-on and a final super-resolution upscaling based on StableDiffusion.*

*The Warping and Try-On networks are trained by us through the DressCode dataset provided by Unimore. Substantial effort was put into dataset preprocessing and network adaptation. Moreover, our generative network was provided with a transformer-based block for establishing global mutual dependencies between the cloth and the person representations. We trained both our transformer-based generative module and another similar module and compared the outputs on a common test set, with results demonstrating better performances for the former.*

# Contents

# 1 Introduction

## 1.1 Problem statement

Our project aims to build a 2D image-based virtual-try-on system (VTON).

VTON consists in generating an image of a target person wearing a given try-on garment. This kind of problem has been widely investigated because of its importance in the fashion industry and its level of complexity.

The main challenges and requirements[11] needed to accomplish this task are:

- warping the garment according to the body shape and pose of the target person;
- transferring the texture of the garment on the target person without losing important details;
- merging the image of the target person with the warped result in a plausible way;
- render light and shades of the final image correctly, to ensure realism.

Due to the exam constraints, the project also includes a content-based retrieval system: given a worn cloth, the system finds similar items from a repository. Also this kind of problem is challenging:

- clothes can have very different shape, colors and intricate decorative patterns;
- the worn cloth images picture may be taken in uncotrolled setting, while in-shop clothing item pictures are taken in a clear and clean setting;
- if the repository contains a very large number of garments, it is necessary an efficient management of the data and efficient retrieval algorithm;
- relevance in the fashion market: garment retrieval can be a very useful e-commerce tool to enchance the customer experience.

## 1.2 Related works

The baseline work is the CP-VTON architecture[11]. Its main contribution was the introduction of a two-stage pipeline:

1. Warping Module: it computes a learnable Thin-Plate Spline transformation (TPS) for warping the in-shop garment in a reliable way;
2. Generative Module: it fits the warped garment on the target person.

The warping module allows the retaining of the important details of the garment, but it fails if the target person pose or the garment texture are too complex, or there are occlusions. Several works tried to improve the warping module and overcome such limitations by: integrating complementary modules; applying regularization techniques to stabilize the warping process during training; projection techniques of the garment details.

For the generative module, the classical approach is based on the U-Net architecture (feeding the person image and the warped cloth). Other works have employed a two-branch network, where one branch takes as input the person and the other the in-shop cloth and warping information. A relatively new approach applies the Transfomer-based architecture and cross-modal attention mechanisms to the inputs before feeding them to the generative network. This step allows the network to extract long-range dependencies between input person image and warped cloth, improving the generated image quality.[9] [3].

Another line of research is trying to construct better and public datasets[8]:

- increasing the total number of samples;
- increasing the image resolution: at now, the mostly used resolution is $256 \times 192$; although the processing is lighter, such images do not retain many cloth details;
- adding new garment categories, like lower-body and dresses images.

Regarding the cloth retrieval task, pioneer works utilized a fixed set of attributes (color, length, material, etc...) hand-labeled or automatically extracted (such as SIFT/ORB keypoints). Then the system compares the query-item and shop-item according to a similarity metric. The overall performance was not satisfying, since perceptual methods alone are not able to capture the higher-level depencences between clothes, that have intricate pattern and not elementary shapes. In the last years, deep neural networks have been widely applied and have pushed the research into a new phase. These new methods learns a similarity metric between real-world and shop-item images from deep features representations extracted form images; an interesting example is Exact-Street-to-Shop[4]. Like shown by[12], focusing on clothes regions and ignoring the background via a cloth parsing mechanism improves the overall performance.

# 2 Our Contribution

## 2.1 What did we do?

There are plenty of state-of-the-art 2D VTON architectures, each one focusing on the improvement of some structural detail. While most of the networks are trained using the VITON dataset[2], which contains 16.253 low-resolution images (256x192), we instead trained our networks on the DressCode dataset[8], recently introduced by the AImageLab of UNIMORE. It benefits from the following characteristics:

- High-resolution images ($1024 \times 768$).
- Very large dataset compared to publicly available ones, with approximately $50,000$ image pairs of try-on garments and corresponding catalog images, where each item is worn by a model.

- Multi-category clothes: front-view and full-body of upper-body, lower-body, and full-body attire.

Given the dataset, we decided to implement a network following the classical two-step warping-generative approach. The warping module is based on the TPS spline CP-VTON architecture. The Try-On module is based on CIT[9], which implies the presence of a three transformer encoders and six cross-modal transformer encoders module built to capture long-range dependencies between input person representation, cloth image and cloth mask.

Part of the work was to adapt existing networks implementations to accept higher resolution images and a different dataset setup, as well as to re-sample and pre-process the original dataset. The main steps are described in *3.6 Data-Preprocessing and Data-Loading Adaptation*.

Given the fact that DressCode dataset lacked the required cloth masks, we implemented a completely custom algorithm to extract the masks using only Canny and other non-deep methods. The results were surprisingly accurate, and only a little portion of the masks ware discarded. More in *3.4 Cloth Mask Extraction*.

We trained the system on upper-body images both with and without the CIT block, we compared the results on the same test set, showing that cross-modal attention blocks lead to better performances. We also trained the system on full-body dress images without the CIT block (to save computational time).

One of the main goals was to make the pipeline work with in-the-wild images. For this reason, we also implemented a background removal module based on semantic segmentation and alpha transparency channel. This step is essential for the correct functioning of the system, because it makes the input image closer to the dataset distribution of sample images.

Last but not least, we performed a super-resolution upscaling based on the recently published ControlNet architecture[1]. This step drastically improves the quality of the final image, even because it removes some defects and artifacts, such as bad hands and glitches. We tuned both the positive and negative prompt commands to better fit our needs. More in *3.10 Post-Processing and Super-Resolution*

We also implemented a retrieval module following the Exact-Street-to-Shop[4] approach. The main contribution here was the introduction of a new query-feature, designed by concatenating ORB keypoints and the 256 levels grayscale histogram into a 1D feature vector. Adding the histogram to the cloth image representation vector increased the accuracy of the network. More in *3.5 Cloth Retrieval*.

# 3 Pipeline and Implementation Details

## 3.1 General approach

As depicted in the figure 1, our system will be subdivided into different modules each one designed to solve a specific step of the process:

- Pre-processing: this module handles all which regards the image enhancement (denoising, light adjustement, etc...) and performs the background removal task;

- Person Representation: this module performs pose estimation and the semantic segmentation of the person into their body parts;
- Warping module: this module implements a geometric transformation that warps the fabric of the clothing item depending of the body shape and pose of the subject;
- Try-On: this module generates a new image by composing the warped garment over the subject and should ensure the satisfaction of the requirements stated in the introduction section;
- Image Retrieval: this module perform a content-based retrieval of the in-shop clothes, given a reference image.
- Super-Resolution with Stable Diffusion: this module performs a zero-shot upscaling of the generated image of the Try-On module, based on the ControlNet architecture and some textual prompts.

Considering the e-commerce use case, during the inference, the customer inputs the target person image and the desired cloth image. The system selects the most similar in-shop cloth and then performs the virtual-try-on using these selected clothes.

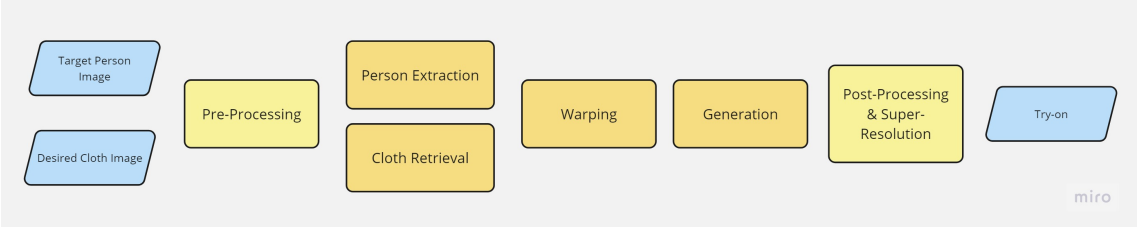



Figure 1: Overview of the pipeline

In order to get across the inner workings of each module we will separate this section into subsections related to each one.

## 3.2 Image Pre-Processing

The system expects to receive two images: target person, desired cloth (it can be worn by a person or it can be cloth-only image). As the objective of the system is to be adaptable to dirty and noisy input images (so called "in-the-wild"), great care should be taken to clean such inputs.

### 3.2.1 Image Enhancing

As the objective of the system is to be adaptable to dirty and noisy input images, in the pre-processing phase great care should be taken to clean such inputs. As such the pre-processing module applies different methods of input refinement in sequence.

Firstly the image goes through a light adjustment procedure which entails contrast stretching with the following rule:

$$I_o = (I_i - min_i) * ((max_o - min_o)/(max_i - min_i)) + min_o \tag{1}$$

then it goes through a denoising pass performed by a bilateral filter.

### 3.2.2 Background Removal

Given a person segmentation mask obtained by other means (see 3.3.2) we perform background removal creating a four-channel RGBA image and using the alpha channel and the mask to obtain only the foreground person. The algorithm can be found in *preprocessing_background_removal.py*. In following example the background was designed to be salmon pink.

(a) Image (b) Mask (c) No background Image

Figure 2: Background Removal

## 3.3 Person Representation Extraction

This module performs pose estimation and the semantic segmentation of the person into their body parts. It is foundamental that the person representation is cloth-agnostic: the try-on task has to preserve the target person's information (face, hair, body shape and pose), while ignoring the worn cloth. We followed the CP-VTON[11] approach, so the person representation contains three components:

- Pose Keypoints: the pose estimation contains information about the person pose. It follows the keypoints representation in OpenPose format. From the keypoints list, it is computed an 18-channel feature map with each channel corresponding to one human pose keypoint, drawn as an $11 \times 11$ white rectangle;
- Body Shape: a 1-channel feature map of a blurred binary mask roughly covering different parts of human body;
- Reserved Regions: an RGB image that contains the reserved regions that are not involved in thewarping or try-on phase (for instance feet, face, hair), detected using SCHP[5] that generates the segmentation mask representing the human parsing of model body parts and clothing items. Our system was adapted to dynamically build different reserved regions images, depending on the different type of garment and the network module.

### 3.3.1 Keypoints Extraction

In order to extract pose-keypoints from a person image, we first attempted to use Openpose. Unfortunately, it did not seem to be compatible with our current environment. Model servers were down so the build continued to fail. As such, we

(a) GMM Module Reserved Regions

(b) TOM Module Upper Body Reserved Regions

(c) TOM Module Dresses Reserved Regions

Figure 3: Different types of reserved regions

adopted a solution based on the Detectron2[13] keypoints extraction architecture (Model: *COCO-Keypoints/keypoint_rcnn_R_50_FPN_3x.yaml*. The keypoints were mostly compatible, but they were 17 instead of 18, and the indexes were placed in a different order. To solve this, we interpolated the missing keypoint and re-ordered them accordingly. This procedure can be found in *detectron2_scripts/d2.py*.

### 3.3.2 Body-shape Mask Extraction

A simple script (*preprocessing/mask_generator.py*) extracts the binarized mask from the SCHP segmentation label map, flattening all body-parts labels into a unique person mask, separated from the background.

## 3.4 Cloth Mask Extraction

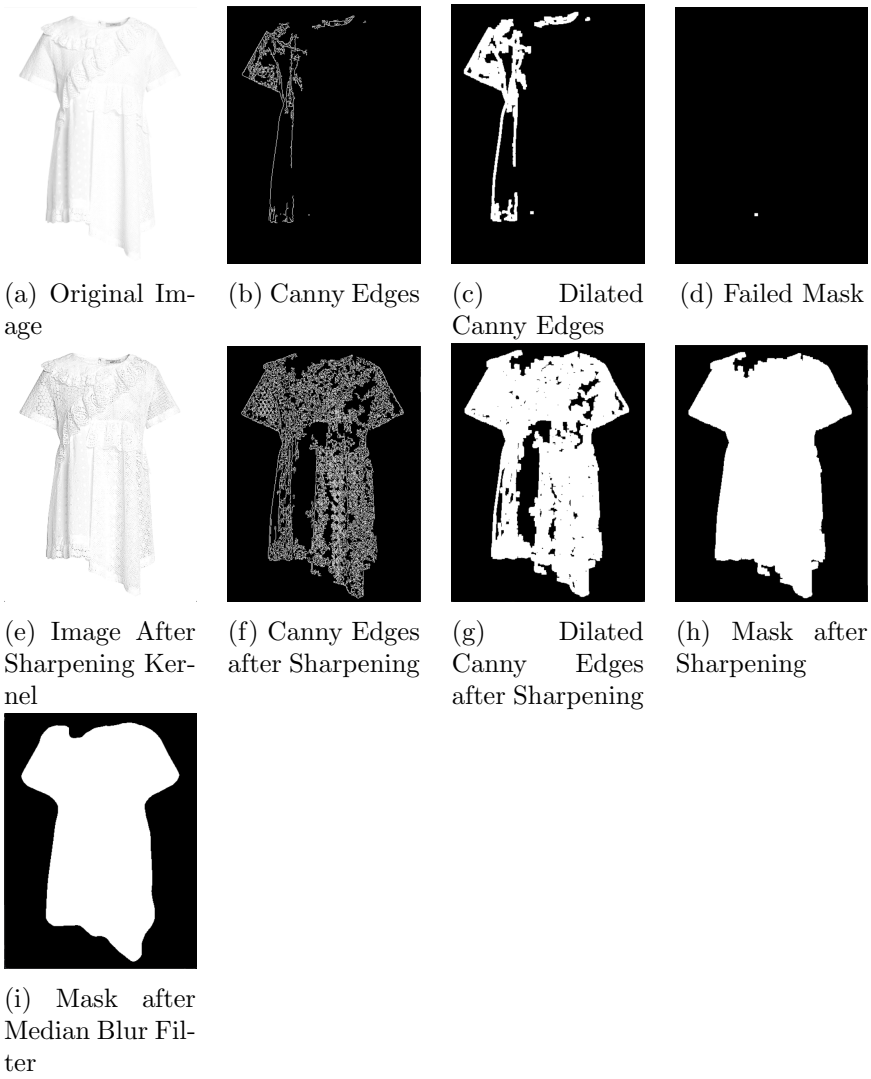
As the DressCode dataset did not include the required cloth masks for the training and inference pipeline, we provided an iterative method to extract masks using Canny algorithm and other non-semantic image processing adjustments. Given a garment cloth image, the steps are the following:

- Apply Canny algorithm to extract edges with upperthreshold = 100 and lowertreshold = 200.
- Apply a 5x5 squared Dilation kernel in order to concatenate disconnected lines.
- Apply findContours and drawContours methods, extracting only the most external outline of the image and filling it with white pixels.
- Checking if the mask is acceptable computing a white/black pixel ratio: if it is lower than a threshold, the mask generation has failed.
- If the mask has failed, a sharpening kernel k is applied to the original image and a second attempt is made.

$$k = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- If the image has been sharpened, the final mask can appear to have rough borders. In order to improve the quality, a Median Blur filter is applied to the final mask image.

Eventually, this second-pass allowed to save about 20% of the failed masks. The complete script can be found in *preprocessing/cloth_mask_generator.py*.



(a) Original Image (b) Canny Edges (c) Dilated Canny Edges (d) Failed Mask

(e) Image After Sharpening Kernel (f) Canny Edges after Sharpening (g) Dilated Canny Edges after Sharpening (h) Mask after Sharpening

(i) Mask after Median Blur Filter

Figure 4: Cloth Mask Extraction Pipeline

## 3.5 Cloth Retrieval

The goal of this module is to match a real-world example of a garment item to the same or similar item in an online shop. We take inspiration from Exact-Street-to-Shop approach that makes two interesing points:

- mixing up the geometric/perceptual and the deep-learning works pretty well;
- tackle the cloth-matching problem as a binary classification (match or not).

inserire immagine

As depicted in the figure.., we built a reference repository of in-shop cloth items, using the cloth-only images in DressCode. For each cloth item we extract a new feature representation following these steps:

- for each image channel (DressCode images are in the RGB format):
  - run ORB algorithm to compute the keypoints and the corresponding descriptors;
  - compute the histogram on the 256 different color levels;
  - concatenate all the values in a 1D tensor;
- concatenate all the channel tensors in a 1D tensor and save it.

The noise-clean desidered cloth image is fed to a sub-module that detects the cloth area (as a bounding box), using SCHP. From this area is extracted a new feature representation (query-feature) following the same steps presented for the cloth-only images. Then, the comparisons are performed:

- for each in-shop cloth:
  - concatenate the query-feature with the in-shop cloth feature in a 1D tensor;
  - fed the tensor to the deep neural network;
- sort and select the $k$-best matched in-shop clothes according to the matching-score given by the network.

inserire l'immagine della rete

As depicted in the figure..., the deep neural network we designed is composed by a sequence of Fully-Connected and Normalization layers, that outputs two numbers representing, after softmax normalization, the probability of no-matching and the probability of matching. The higher one is considered the final classification result.

sistemare la formula

To train this network, we used the regularized cross-entropy loss:

$$L(\theta) = \lambda_1 \cdot L_1(c_{warped}, I_{c_t}) + \lambda_{reg} \cdot L_{reg}. \tag{2}$$

in Experiments, accennare al dropout

During training, the network tried to classify both positive examples (the paired couples of worn-cloth and cloth taken) and negative examples (not paired couples). Since the number of potential negative examples was far more the number of positive examples, we used data augmentation to creating synthetic positive examples from existing one: we applyed gaussian noise to the feature representations.

## 3.6 Data-Preprocessing and Data-Loading Adaptation

In orderd to perform the virtual-try-on task we were inspired by CP-VTON+[7] public repository, but it was deprecated. Thankfully, we found a fixed and updated version.

Once we checked the proper functioning of the architecture on the VITON dataset we had to adapt the process of dataloading because the Dresscode dataset was different:

- it lacked binary cloth masks and binary person masks;
- the body parts segmentation on which the module was originally trained used different labeling technique: the body parts were mapped in different classes;
- the size of the dataset was way too big (70 GB) for our purposes, therefore we could not use the official training-test split list, and we had to re-sample and resize the images.

We first resized the images using *preprocessing/dataset_resize.py* script. After that, we ran the cloth mask generation script, saving only the samples with an acceptable mask (*preprocessing/cloth_mask_generator.py*). We then ran the body shape mask extractor (*preprocessing/mask_from_dataset.py*). We also needed a script that adapted DressCode keypoints format to the Openpose json format accepted by the original dataloader. (*preprocessing/json_conversion.py*) The last step was to subsample the original dataset and re-create the train-test list text file used to load it. All the procedure is in (*preprocessing/dataset_subsample.py* and *preprocessing/dress_code_train_test_txt_gen.py*).

The resulting dataset contained 11.959 upper-body samples and 24.309 dresses samples, in 512x384 resolution.

We also modified the Dataset class used by the dataloader, changing some file names conventions and creating a new label map dictionary to adapt the DressCode SCHP based body parts segmentation masks to the system. See more in *network/cp_dataset_modified.py* (Dictionary name: *DressCode_labelmap_g*).

## 3.7 Warping Module

We follow the warping module proposed in CP-VTON+[7]. This module transforms the input try-on garment $c$ into a warped image of the same item that matches the body pose $p$ and shape $m$. As warping function we use a thin-plate spline (TPS) geometric transformation, which is commonly used in virtual try-on models *qua va la cit.33*. Inside this module, we aim to learn the correspondence between the inputs $(c, p, m)$ and the set of parameters $\theta$ to be used in the TPS transformation. Intuitively, the TPS fits a surface (try-on garmet) through points (feature regression output points) minimizing an energy function. Formally, it is a poli-harmonic spline function which maps a set of points $(x, y)$ on their correspondences $(x', y')$ sampled from input images. Under these terms, the warping module aims to learn how to perform such transformation.

As depicted in figure blabla, the module extracts an encoded representation of the try-on garment $c$ and an encoded representation of the person representation through two separate convolutional networks. Then, it is computed a correlation map between these representations. This correlation map is used to predict the spatial transformation parameters $\theta$ corresponding to the $(x, y)$-coordinate offsets of TPS anchor points.

To train this network, we used the following loss:

$$L(\theta) = \lambda_1 \cdot L_1(c_{warped}, I_{c_t}) + \lambda_{reg} \cdot L_{reg}; \tag{3}$$

$$L_{reg}(G_x, G_y) = \sum ii \sum ii \sum iii; \tag{4}$$

where $L_1$ indicates the pixel-wise $L_1$-loss between the warped result $c_{warped}$ and the ground truth $c_t$. $L_{reg}$ indicates the grid regularization loss.

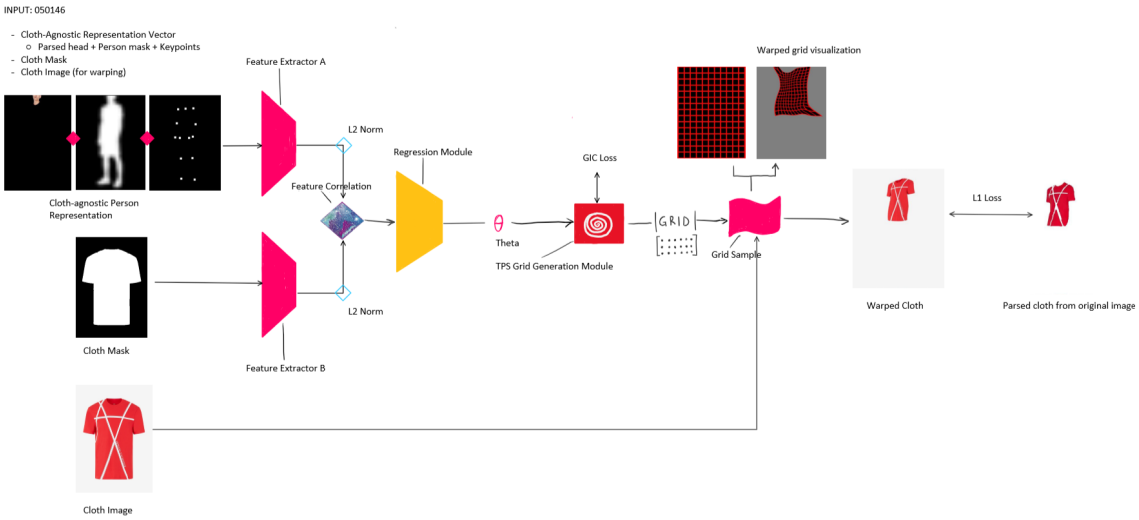



Figure 5: Geometric Warping Module Overall Architecture

## 3.8 Generative Module

This module aims at fitting the warped garment on the target person. Previous works like CP-VTON and CP-VTON+ directly concatenate the person image $p$, the warped clothing image $\hat{c}$, and the warped clothing mask image $\hat{cm}$ .

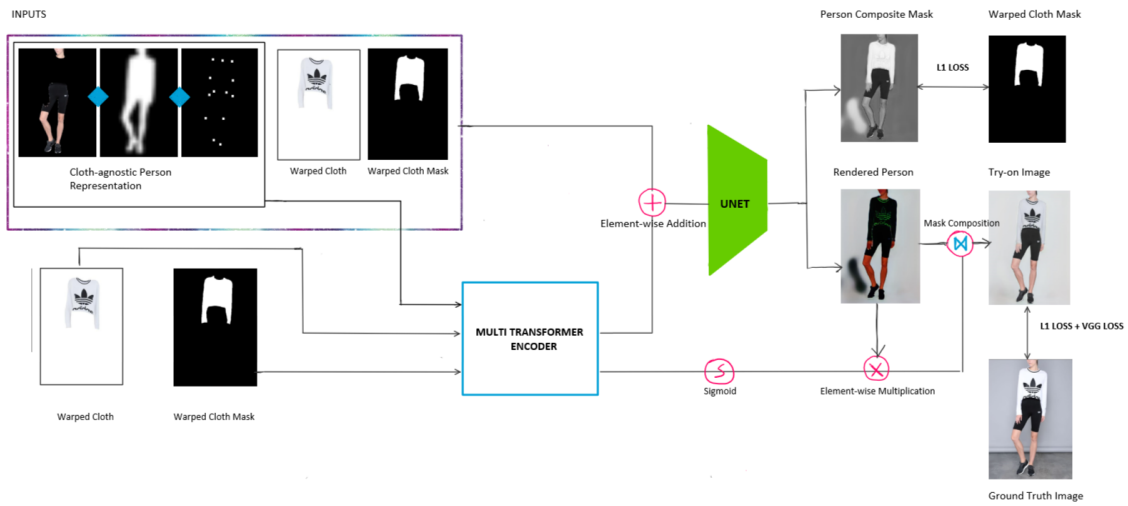



Figure 6: Try-On Module Overall Architecture

Then the concatenated input is sent to a UNet[10] to generate a composition mask $M_o$ and a rendered person image $I_R$. The main limitation of this approach is the inability of the convolution operator to model global long-range dependences. For this reason, we follow the CIT[9] approach that leverage the Transfomer-based architecture and cross-modal attention mechanism that are able to capture the dependence among these three input images.

As shown in the figure 7, the three inputs goes through the patch embedding, for making the image data compatible. Then each goes through a 1D temporal convolution to ensure the relation modeling of each element with its neighbor elements. Then the Interactive-Transformer II is utilized for modeling the global long-range correlation. The output $X_{out-II}$ of Interactive Transformer II is obtained after a linear projection and a reshape operation. Then $X_{out-II}$ is utilized for two proposes; one is to activate the important region of the overall input by adding $X_{out-II}$ to $I(p, \hat{c}, \hat{cm})$; another is to guide the final mask composition as follows:

$$\begin{aligned} I_R^{global} &= sigmoid(X_{out-II}) \times I_R, \\ I_o &= M_o \times \hat{c} + (1 - M_o) \times I_R^{global} \end{aligned} \tag{5}$$

where $\times$ represents the element-wise multiplication and *sigmod* indicates the sigmoid activation function.

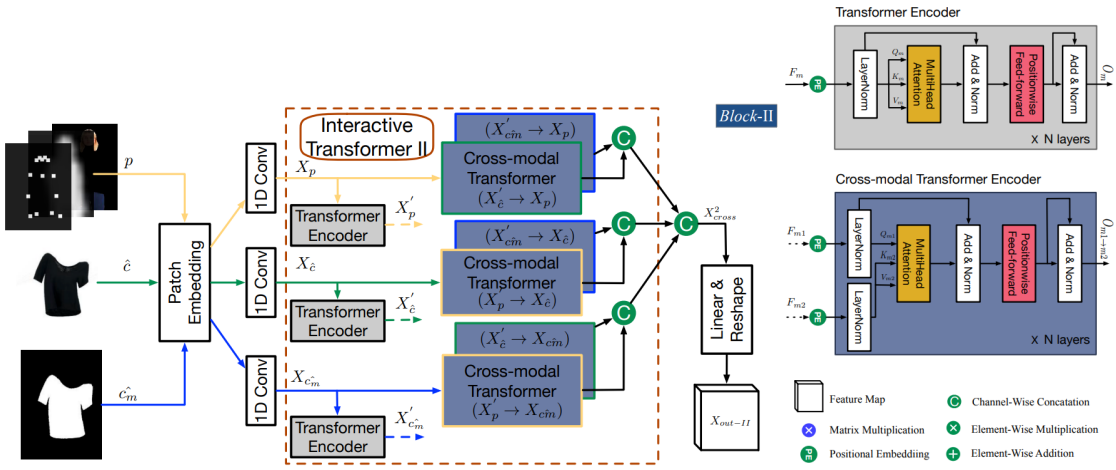



Figure 7: On the left: CIT Reasoning Block architecture, also called Interactive-Transformer II. On the right: the Transformer Encoder and Cross-Modal Transformer Encoder architetures and a legend of symbols.

## 3.9 Training

We performed five training steps. GMM module for upper-body and dresses garments, TOM module with CIT and TOM module without CIT for upper-body garments, and TOM module without CIT for dresses garments. We evaluated results by IoU measure for the warping step, and by SSIM, FID and MSE for the generative step.

### 3.9.1 Setup

In all experiments, we use $L1$ =vgg = 1. We trained GMM Modules for 315.000 steps batch size 4. We use Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Learning rate is first fi

xed at 0.0001 for 100K steps and then linearly decays to zero for the remaining steps.

Due to the heavy computational load of the CIT reasoning block, the upper-body generative modules were trained for 50.000 steps only. To have an idea, without CIT block one training step required about 1 second, with CIT block it required 10 seconds. Batch size and optimizer options are the same. We were able to supervise the training procedure by Tensorboard tool.

### 3.9.2 Upper body Training

Some insight into the Geometric Matching Module.

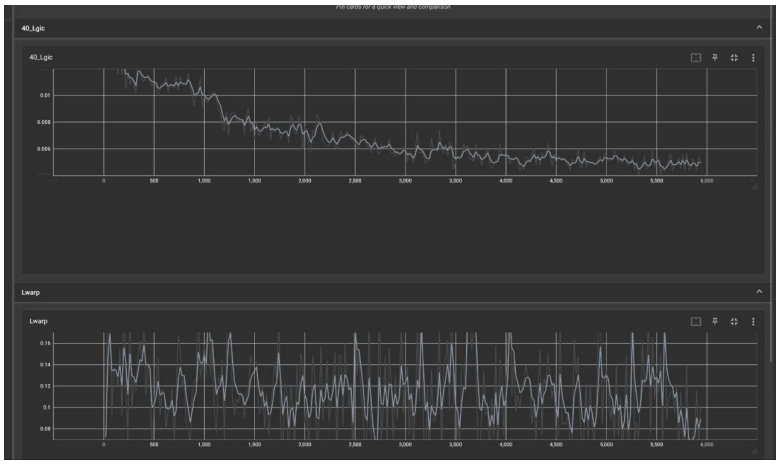
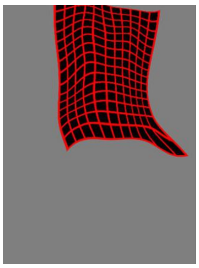



Figure 8: GMM Gic and Warp Losses evolution

(a) In-shop cloth

(b) Warped cloth

(c) Ground truth worn cloth



(d) Warped grid

(e) Warped cloth over image

(f) Ground truth person image

Figure 9: Example of the warping stage of an in-shop cloth image over a paired person image

We trained the network with 11.411 training images and tested it on 1.196 test images. Some comparison between CIT and baseline generated images on test-set:

| | | | |
|---|---|---|---|
| With CIT Block |  |  |  |
| Without CIT Block |  |  |  |

Table 1: Comparison between Generation with and without CIT block

### 3.9.3 Dresses Training

Training on the Dresses category was only performed using the baseline version of the generative network to save computational time. We trained the network on 24.309 images for 200.000 steps for the warping module and 100.000 steps for the generative module, and tested it on 2.431 images.

Figure 10: Some examples of test Dresses generated images.

### 3.9.4 Final Experiment Results

We computed some measures between our results and other available benchmarks from other papers. Due to the different test set and the different number of steps, a comparison between our work and other official publication results is misleading. The only fair comparison is between the two upper-body generative results, which are both trained on the same data and the same number of steps (50.000). The results show that the CIT block indeed increase the generative image performance. The measures are computed on paired image couples, but we also performed an

unpaired cloth-person test set generation. Scripts can be found in the *measures* folder.

| IoU Measure [↑] (Upper body) | Mean | Std |
|---|---|---|
| CP-VTON (on VITON ds) | 0.7898 | |
| CP-VTON+ (on VITON ds) | 0.8425 | |
| Ours (on Dress-Code 200.000 steps) | 0.7841 | 0.1444 |
| Ours (on Dress-Code 315.000 steps) | 0.7975 | 0.1428 |
| | | |
| FID Measure (on DressCode) [↓] (Upper body) | FID | |
| CP-VTON (trained by DressCode authors) | 46.99 | |
| CP-VTON+ (trained by DressCode authors) | 28.93 | |
| **CP-VTON+ (50.000 steps, trained by us)** | **23.95** | |
| CIT (trained by DressCode authors) | 26.41 | |
| PSAD (trained by DressCode authors) | 17.18 | |
| Dual-Branch Collaborative transformer | 17.30 | |
| **Ours (50.000 steps)** | **22.08** | |
| **Ours + Super-Resolution** | **24.61** | |
| | | |
| SSIM (on DressCode) [↑] (Upper body) | | |
| CP-VTON (trained by DressCode authors) | 0.812 | |
| CP-VTON+ (trained by DressCode authors) | 0.863 | |
| **CP-VTON+ (50.000 steps, trained by us)** | **0.894** | |
| CIT (trained by DressCode authors) | 0.860 | |
| PSAD (trained by DressCode authors) | 0.928 | |
| Dual-Branch Collaborative transformer | 0.911 | |
| **Ours (50.000 steps)** | **0.903** | |
| **Ours + Super-Resolution** | **0.875** | |
| | | |
| MSE [↓] | | |
| **Ours (50.000 steps)** | **278.17** | |
| **CP-VTON+ (50.000 steps, trained by us)** | **788.30** | |
| **Ours + Super-Resolution** | **398.85** | |

Figure 11: Experimental results

## 3.10 Post-Processing and Super-Resolution

The final step of the pipeline consists in applying an upscaling to the generated image, to obtain both an higher resolution and a correction of some generative defects. The upscaling is done through a recently released architecture named *ControlNet* [1]. ControlNet is a neural network structure to control diffusion models by adding extra conditions. You can choose a StableDiffusion base model and through some prompt an input image, the generation is forced to follow the basic structure of the input image. There are various models trained for example to generate images from edges maps, from depth maps and so on. We chose the *tile* architecture [6], trained to perform a realistic upscaling. Our positive prompt was set to

$$prompt = "bestquality, clothes, garment, model, shop, upperclothes"$$

and the negative one to

$$prompt = "blur, lowres, badanatomy, badclothes, badhands, cropped, worstquality,$$

$$fading, glitch, robot, tech, highsaturation, medieval"$$

The complete script is *super-resolution/controlNet.py*

| | | | |
|---|---|---|---|
| Original |  |  |  |
| With Super-Resolution |  |  |  |

Table 2: Comparison between generated images and generated images after Super-Resolution

# 4 Conclusion

Our work examined the 2D virtual try on problem in detail, highlighting some of its strenghts and its weaknesses. We were able to handle an heterogeneous dataset and to make it suitable for our purposes, by means of a lot of scripts and algorithms. We performed network adaptation and training on different cloth categories, and we were able to make a comparison between two approaches. We found out that the presence of a multi-modal transfomer-based feature extraction module improved the generated images. We re-implemented and re-trained a custom retrieval module, changing its feature vector composition. How can all of this be improved for future development? First of all, the computational training time prevented us to obtain a more precise final result, expecially in the generative module. A second improvement can be made in the dataset itself. Following the idea of the CP-VTON+ fix, more accuracy can be obtained if in the SCHP body segmentation images the neck label could be separated from the face label. The neck should not be included in the reserved region image, because it can be overwritten and modified by the worn cloth. Another idea can be to add a self-supervised step to disjoint the particular dataset image structure from an input image that can be taken with a different angle, a different camera perspective and so on.

(a) Input image (b) No background image (c) In-shop cloth (d) Try-on image (e) Super-resolution image

Figure 12: Final example of VTON starting from a in-the-wild image

# References


[1] Lvmin Zhang Anyi Rao Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.

[2] Xintong Han Zuxuan Wu Zhe Wu Ruichi Yu Larry S. Davis. Viton: An image-based virtual try-on network. https://arxiv.org/abs/1711.08447, 2018.

[3] Emanuele Fenocchi, Davide Morelli, Marcella Cornia, Lorenzo Baraldi, Fabio Cesari, and Rita Cucchiara. Dual-branch collaborative transformer for virtual try-on. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2246–2250, 2022.

[4] M. Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C. Berg, and Tamara L. Berg. Where to buy it: Matching street clothing photos in online shops. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3343–3351, 2015.

[5] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. Self-correction for human parsing, 2019.

[6] lllyasviel. Controlnet - v1.1 - tile version. https://huggingface.co/lllyasviel/control_v11f1e_sd15_tile, 2023.

[7] Matiur Rahman Minar, Thai Thanh Tuan, Heejune Ahn, Paul Rosin, and Yu-Kun Lai. Cp-vton+: Clothing shape and texture preserving image-based virtual try-on. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[8] Davide Morelli, Matteo Fincato, Marcella Cornia, Federico Landi, Fabio Cesari, and Rita Cucchiara. Dress Code: High-Resolution Multi-Category Virtual Try-On. In *Proceedings of the European Conference on Computer Vision*, 2022.

[9] Bin Ren, Hao Tang, Fanyang Meng, Runwei Ding, Ling Shao, Philip HS Torr, and Nicu Sebe. Cloth interactive transformer for virtual try-on. *arXiv preprint arXiv:2104.05519*, 2021.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[11] Bochao Wang, Huabin Zheng, Xiaodan Liang, Yimin Chen, and Liang Lin. Toward characteristic-preserving image-based virtual try-on network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 589–604, 2018.

[12] Zhonghao Wang, Yujun Gu, Ya Zhang, Jun Zhou, and Xiao Gu. Clothing retrieval with visual attention model, 2017.

[13] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.