New Dress Virtual Try On

CV & CS Project

2021/2022

Botti Mirco - matr: 160220

Querzoli Giulio - matr: 161654

{242649, 245058}@studenti.unimore.it

Abstract

One of the biggest fears of buying clothes online is the lack of trying them before the purchase. This project aims to help customers to perform a virtual try on before the purchase and help stylists to create and try new clothes on virtual models starting with a simple silhouette.

We created a linear pipeline, all the steps aim to keep code the simplest possible, trying to generalize and increase the situation in which the system can perform, especially from the customer point of view.
We want to provide good results even without powerful cameras and with photos taken from amatour.

We used a first version of the Unimore dataset *Dress Code: High-Resolution Multi-Category Virtual Try-On*, in fact the dataset version provided to us does not have all the feature of the original one but we take the challenge to create a good system with limited resources.

Project's source code:
https://github.com/JCobot/CVandCSproject

I. Introduction

We divided our pipeline in 4 steps. In the first step we preprocessed the images and added some features to the existing data using different classical computer vision algorithms in order to create a garment mask.

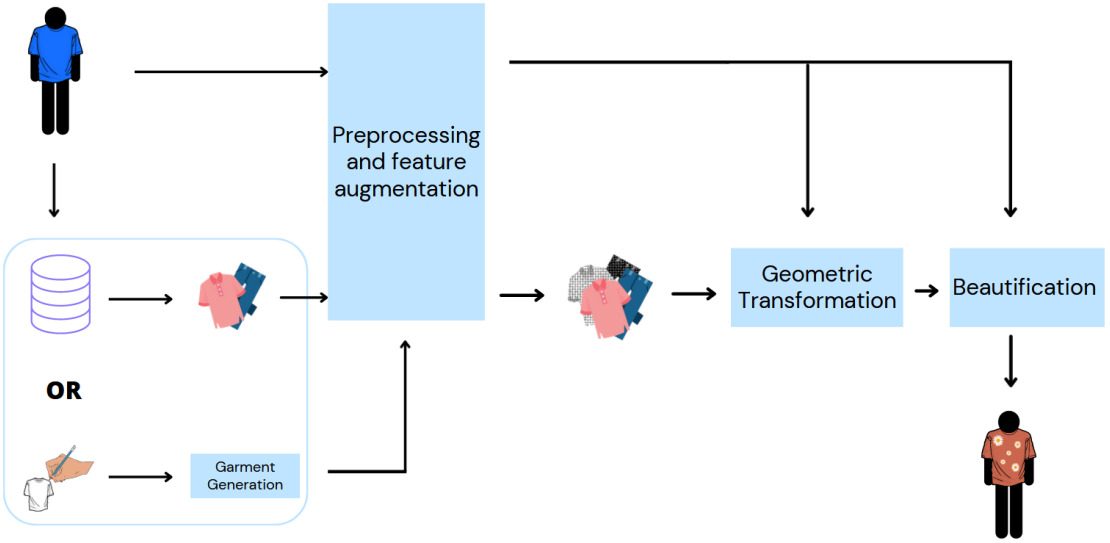
The second step can be divided in two main parts depending on what users want:

- retrieval of clothes that match with the user *season* and simulate armocromy
- creation of new garments.

Both parts give as output a cloth to fit into the model.

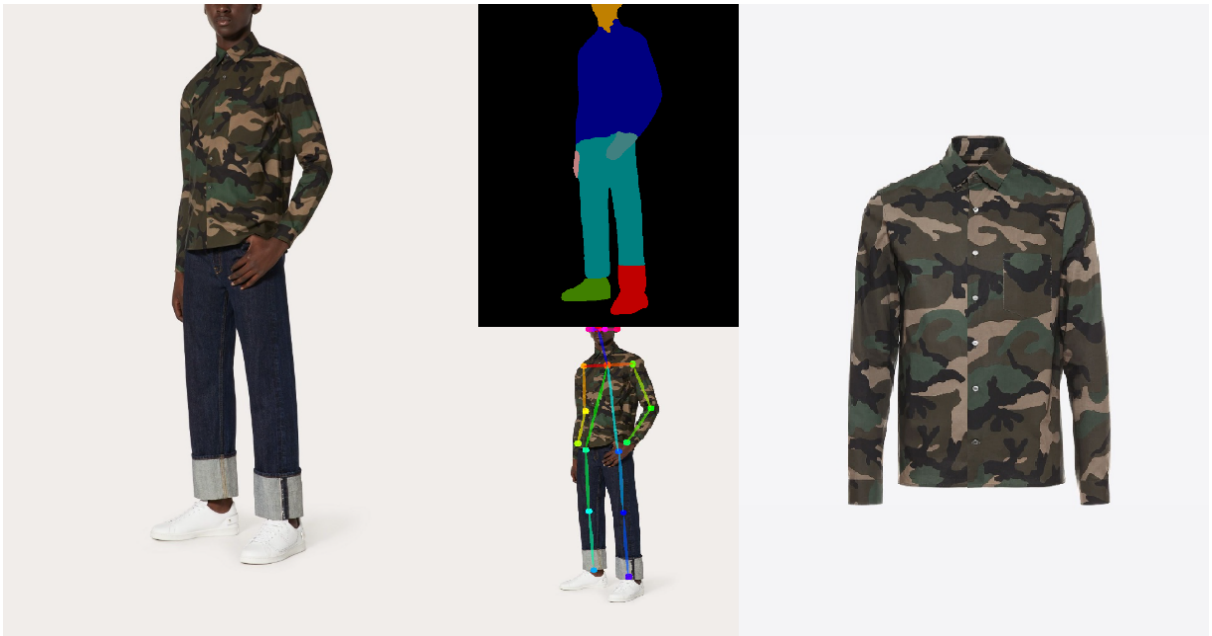
Third part of the pipeline is the warping module taken and customized from the VITON [1]. This module deforms our target cloth trying to make it wearable for the model.

Last section is called *beautification* part, in fact, thanks to a Generative Adversarial Network, starting from a raw wrapped version of the garment fit on the model we try to make as real as possible the fitted cloth.



II. Dataset

The dataset used for the project work is an initial version of *Dress Code: High-Resolution Multi-Category Virtual Try-On [2]*. We didn't use the final dataset because when we started the project the dataset wasn't already published.



The dataset contains circa 35k images. With respect to the original dataset this one contains only the upper-body and the lower-body parts, dresses were not provided.
Even in the image's feature something is missing, in fact for every image we have:

- Pose's keypoints
- Model's semantic segmentation
- Skeleton image

and not the Dense Pose.
During our initial research phase we noticed that a dense pose is widely used in the geometric transformations phase but we take this lack as a challenge and we tried to implement the network

without this feature. Even if the images were in high resolution (768x1024) we decided to work with a smaller resolution (192x256) in order to perform better with our 4GB RAM GPU (NVIDIA GTX 1050).
Another fundamental file for the training phase is the *yu-vton.cvs* that links all the features of the same cloth together.

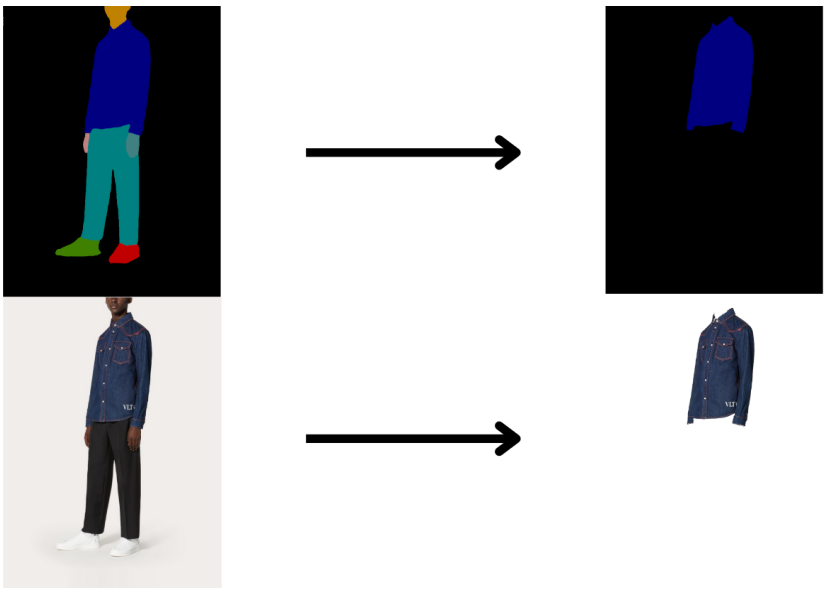
Data Augmentation

Customers do not have perfect pictures like the one in the dataset so we decided to edit it with the python library Albumentation.
Target images are too much standard and without particular variation so we decided to augment the dataset by scaling these images to different levels and cropping them to desired size. After this step we also applied gaussian noise.

III. Preprocessing

We resized the images from 768x1024 pixels to 192x256, we normalized their rgb values into the range [0,1] and changed the shape using channel first modality to make it easily managed by the networks in Pytorch.

We also added additional features like the source and target cloth's mask. Given the semantic segmentation of the model it is to extract its mask simply taking the part of the image that was labeled as our target cloth.



The garment mask was not easy to compute like the above one, there was not a pre segmented image. In order to extract the desirable feature we firstly thought to use a specialized NN for foreground-background extraction but we decided to use a classical algorithm because a NN for this task is like using a sledgehammer to crack a nut. We mainly used 3 approaches, one domain specific and the other 2 more general, both give us good results.

Domain specific solution

We noticed that in every image the background is uniform and almost white, the foreground image is always centered. With these information we designed a pipeline that apply a gaussian filter, convert the image to grayscale, apply a Sobel operator and add it to the image (this because some garments colors was very similar to the background and border where not strong enough), after that we apply two different GrabCut algorithm.

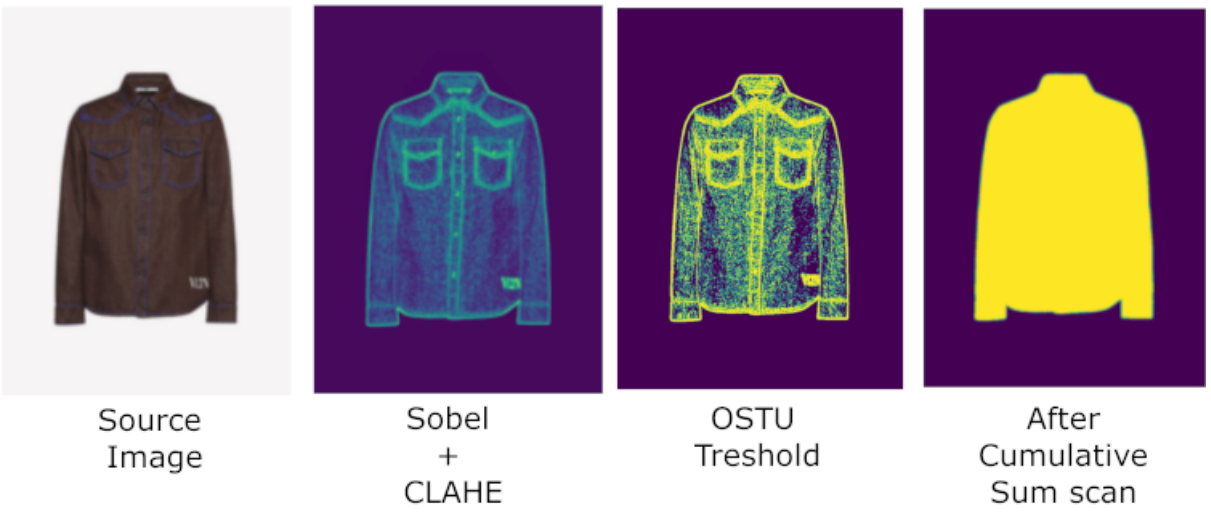
The first one uses a rectangle with fixed dimensions that point the center of the image, this gives us an initial bg-fg proposal, after that we set the given *sure background* as *probably background* and apply the second GrabCut algorithm using this new mask as initialization instead of the previous rectangle. We decided to set all the pixels classified as background to *probably background* because the initial rectangle could remove a small part of the garment outside of its bounds (like in the example below).

General Solution

The domain specific solution works really well but if our garments change position or are a long lower-body part the algorithm can fail. That's why we also implemented a second version for cloth's masking.

As a first step we always apply a gaussian filter to the image, convert it from rgb to gray level and apply the Sobel operator. This time we don't add the result to the original image but we work on Sobel's result applying an adaptive histogram equalization with the CLAHE (Contrast Limited Adaptive Histogram Equalization) algorithm to enhance edges.

After this step we use an Otsu threshold, the result obtained is an image clearly divided in 2 regions but the internal part of the cloth isn't filled yet. In order to fill it we scanned the image in every direction (from top to bottom, left to right and vice versa) , computing the cumulative sum of the rows and setting it as a background until the cumsum doesn't become greater than 0 and foreground the remaining part.



Grana mask version

We also implemented a variation of the previous algorithm using the Grana mask instead of the cumulative sum method. The mask works really well but it fails, like the other method, when the garment has a color similar to the background. In order to select the correctly labeled group of pixels (the garment) we put to 0 all the pixels with the same label of the top left corner assuming that there's not a garment there, all the other labels become foreground.

Masking Results

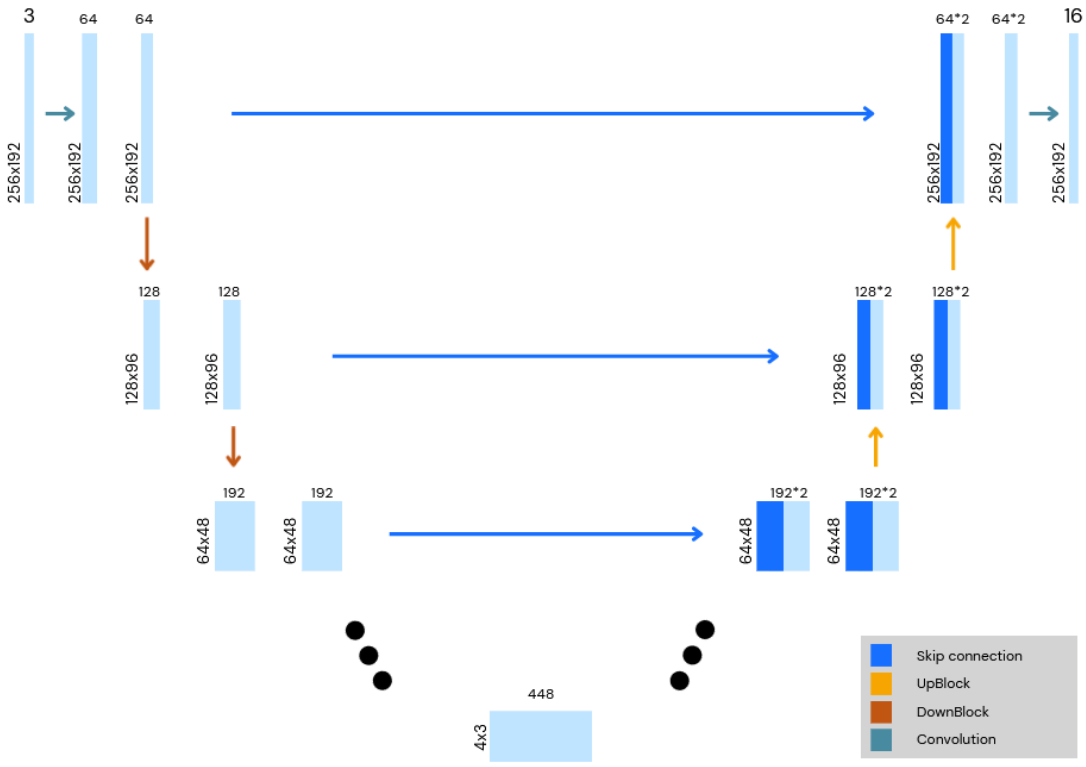
The VITON dataset contains also the garments mask's images so we used its first 1000 images as a test for our methods. We used as a measure the intersection over union and after that we compute the mean and variance of the results.

Method	**Time (s)**	**Mean**	**Variance**
Two GrabCut	233	0.783	0.068
CumSum	3	0.917	0.018
Grana Mask	3	0.805	0.065

Cumulative sum works better due to its nature, if a border is not closed Grana mask labels all its internal pixels as background while the current algorithm stops when reaching the first internal edge in the perpendicular direction.

IV. U-net variation

During the project we decided to use the same architecture for different tasks, from segmentation to generation or beautification of new garments. The architecture chosen is a variation of the U-net used in Pix2Pix [3] that uses less parameters w.r.t the original one.



The architecture of the network is composed by an initial convolutional layer that preserves the original dimensions of the image but increases its channels and, after a Batch Normalization and a Leaky ReLu, a series of DownBlock layers that reduces the image to a desired dimension and increase receptive field. In order to reconstruct the image a series of UpBlock layers with skip connections are performed and, as a final step, a convolution layer is done.

The DownBlock consist in:

```
layers = []
layers+=[nn.Conv2d(in_channels,out_channels,4,2,1,bias=False,padding_mode="reflect")]
layers += [ nn.BatchNorm2d(out_channels) ] if norm else []
layers += [ nn.LeakyReLU(0.2) ] if relu else []
self.model = nn.Sequential(*layers)
```

The UpBlock is composed by:

```
layers = []
layers += [ nn.ConvTranspose2d(in_channels, out_channels, 4, 2, 1, bias=False) ]
layers += [ nn.BatchNorm2d(out_channels) ] if norm else []
layers += [ nn.LeakyReLU(0.2) ] if relu else []
self.model = nn.Sequential(*layers)
```

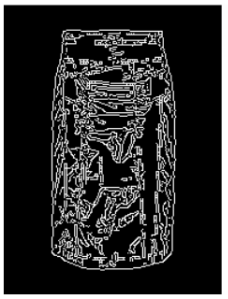
As you see, instead of using the MaxPooling layer to reduce the image dimensions like in the classic U-net we used a convolution with kernel size 4 and stride 2.
Another important thing is that our net has less parameters than U-Net (with 6 layers, 22 Millions vs 61 Millions circa), this because our scope was to implement a network capable to run even in an ordinary computer with a mainstream GPU.

It is possible to have different levels of DownBlock and UpBlock but every network used in this project uses the same number to have maximum receptive field. All tasks need different losses.

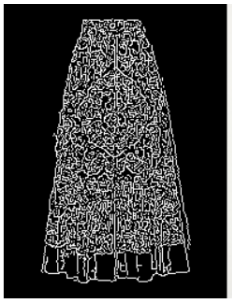
V. New Garments

At the beginning of our pipeline you can choose if you want to try a garment taken from the dataset or a brand new garment.
The process to create a new cloth star from a simple sketch or a contour of an existing cloth image (preprocessed with canny) that is passed through a GAN having as a generator our U-Net variation with 6 DownBlock and as output size an rgb image (256x192x3).

After the down phase of the U-Net we concatenate the bottleneck with a color space encoded (vector of 64 values) suggesting to the network the color of the final garment.
The color autoencoder is trained with MSE loss while U-Net tries to recreate the original dress and get evaluated by GAN loss and MSE.



from left to right: silhouette, generated and original garment



silhouette, generated and original garment

VI. Retrieval

If instead of generating a new garment the customers want to try a cloth in the e-commerce database our system, thanks to our retrieval module, will suggest the most fittable.

We used the concept of armocromia in order to suggest the most fittable cloth, basing the system choice on the model's hair and skin colors. As a feature vector we used a discretized color histogram. We computed the cosine distance between other models' feature vectors and selected the top 10 of them.

We made a survey to 5 persons interested in armocromy. They were skeptical because according to professionist in this sector there are too many things to consider. Given a first person they voted if the

following clothes would fit well to it and the retrieval performed a MAP of 0.8424 with a minimum score obtained in a query of 0.6679.

VII. Semantic Segmentation

We implemented the semantic segmentation of the target image to have the possibility to detect cloth on it and use it to distort other cloth.
In this part we tried different implementations and computed metrics in a way to compare different methods. All this method starts from our U-Net.
All these methods are trained for 50 epochs with learning rate 1e-4 and a scheduler that decreases it by factor 0.1 after a 10 epochs plateau.

Pixel classification

First of all we implemented the model using multiclass Dice Loss and cross entropy summed up.

Performance on test set:
DICE loss: 0.5958
CE loss: 0.3098

from left to right: input, predicted and ground truth

This network performs quite well but commits bad semantic error classifying as belt some random pixel in the t-shirt. It is appreciable that the model classifies socks as part of the shoes in this image.

GAN approach

To reduce semantic problems we also trained a discriminator to recognize fake segmentation, this yields a more uniform segmentation and less strange pattern in the prediction but has numerical performance decreased by a small factor. In the following image we can see how our model classifies the left arm of the mannequin correctly even if it were classified as head in the dataset.

from left to right: input, predicted and ground truth

Performance on test set:
DICE loss: 0.6118
CE loss: 0.3029

Weight normalization

As seen in different papers we tried weight normalization [4] to stabilize and speed up the training process. Dice loss does not decrease but multiclass cross entropy does and the result was subjectively better relative with other previous steps.

Performance on test set:
DICE loss: 0.5704
CE loss: 0.1587

from left to right: input, predicted and ground truth

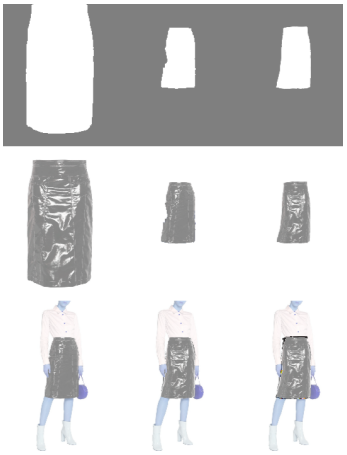
VIII. Warping Module

One of the most important parts of our pipeline is the warping module that is responsible, given an initial representation of an object *c* to warp it as close as possible to the same item *m* fitted to a model. As a warping function we used the Thin Plate Spline (TPS) [5], a common CNN for geometric matching widely used for virtual try on. Despite lots of models giving as input additional information about the target model like the agnostic representation [1], [2] we decided to keep this part as simple as possible and give as input only the target and source masks computed during the preprocessing phase. This simplification allows for a lighter inference due to the removal of additional features (like pose), only the segmentation is required. Input features are decreased from 22 to 9.

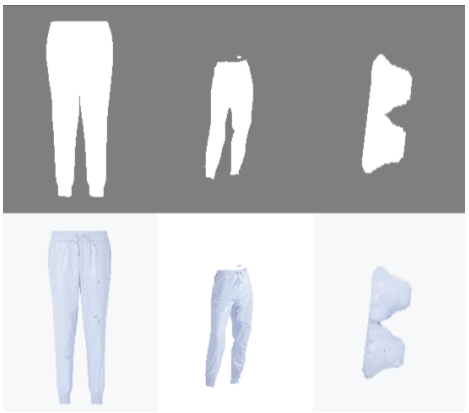
The module is divided in 3 main components: a siamese network for information extraction, a correlation part to find common characteristics and a regression network that gives as output a 50 dimensional vector for the TPS.

Module customization

Initially we tried with the original implementation by [1] using the L1 loss. We noticed that wrapped images could not overlap the original one due to partial occlusion of the garment caused by the body orientation, cloth folds and light reflection. For this reason we decided to consider only the shape and not the texture. This problem can be viewed as a pixel's binary classification, so we implemented the dice loss that can be computed also as F1 score.



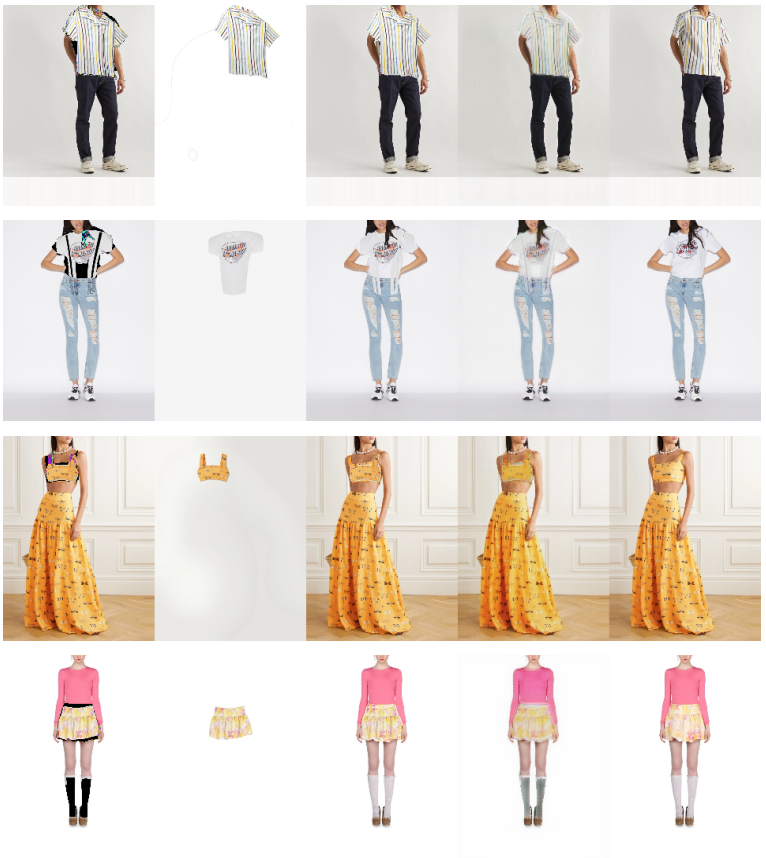
We also noticed that in most of the GMM implementations an affine grid were implemented, so we decided to implement also the homograpy one increasing the degree of freedom from 6 to 8 with respect to the affine method. The network learns to fit the destination area but unfortunately the result loses its semantic meaning in some cases while TPS is more stable.



IX. Beautification

The last step is the beautification part. This step is necessary to remove the wrong border and clear semantic problems like dress-over-arm. The module consists, like other parts of the pipeline in a GAN with 6 layer U-Net variation.
Network takes as input the target person with its original cloth segmented out, the desired gatermet distorted for a better fit, original and wrapped cloth segmentation, segmentation of arm, head and hair.

We use as loss the Gan loss and a particular mean squared error computed with the original image and a fake one recreated overlapping the wrapped cloth to the original target without segmentation to simulate better border.



from left to right: overlapped cloth given as input, wrapped cloth, cloth overlapped to the target, output, original target

Computing L1 loss with respect to the original target tends to produce blurred dress due to unoverlappable textures. A lot of the garment once worn seems completely different so we decided to minimize the difference with the original target overlapping even the wrapped cloth on it. This allows us to keep wrapped textures and original borders.

Conclusions and Future Developments

This project is just a starting point to a more complex pipeline for virtual try on. We tried to take all the steps simple and use fewer features as possible in a manner that everyone can use it. To improve the model's generalization an extra step in the data preparation can be done: modify the background in order to have a different ones for every image, that's because a general user, especially if they want to do the try on during online shopping, doesn't always have a white background.
Even in the garment generator module a lot of future improvements can be done, for instance providing an image to take inspiration or a texture to imitate.

Bibliography

[1] X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis, “VITON: An Image-based Virtual Try-on Network.” arXiv, 2017. doi: 10.48550/ARXIV.1711.08447.

[2] D. Morelli, M. Fincato, M. Cornia, F. Landi, F. Cesari, and R. Cucchiara, “Dress Code: High-Resolution Multi-Category Virtual Try-On,” *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2022. doi: 10.1109/cvprw56347.2022.00243.

[3] Isola, Zhu, Zhou, and Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” *CVPR*.

[4] T. Salimans and D. P. Kingma, “Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks,” *arXiv [cs.LG]*, Feb. 25, 2016. [Online]. Available: http://arxiv.org/abs/1602.07868

[5] I. Rocco, R. Arandjelovic, and J. Sivic, “Convolutional Neural Network Architecture for Geometric Matching,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2553–2567, Nov. 2019.