Design a new pipeline to make a virtual try on from a deep learning perspective

Riccardo Agazzotti
244836@studenti.unimore.it

Eros Bignardi
240696@studenti.unimore.it

Federico Cocchi
289842@studenti.unimore.it

November 3, 2021

Abstract

Nowadays E-commerce is growing faster than ever and COVID-19 emergency increased this trend even more. Our work aims to improve the e-commerce user experience while buying fashion products online. The problem that always occurs in this case is not being able to try clothes and choose the size that fits better.
Fashion companies are looking for innovative methods to help their costumers to choose their clothes correctly thus improving the online shop experience and avoid items to be returned.
In order to develop this system we have build a pipeline in which a user can do a cloth segmentation on a picture of himself.
The system can also help the user to choose the best type of cloth using a recommendation system which highlights you similar items to the ones you have recently viewed. For an architectural point of view the last step is a geometrical transformation that fits the cloth in the best way possible on the picture uploaded by the user. The process I have just described should improve sales.
To reach our goal we used neural networks trained on the dataset 'YNAP-Unimore Virtual Try-On Dataset' (Yoox)', available to us through the university.

Link to the material and the code used:
https://drive.google.com/drive/folders/1BVmLv3FtQsoEdpYxrf7PEyIrWB6BWqGx?usp=sharing

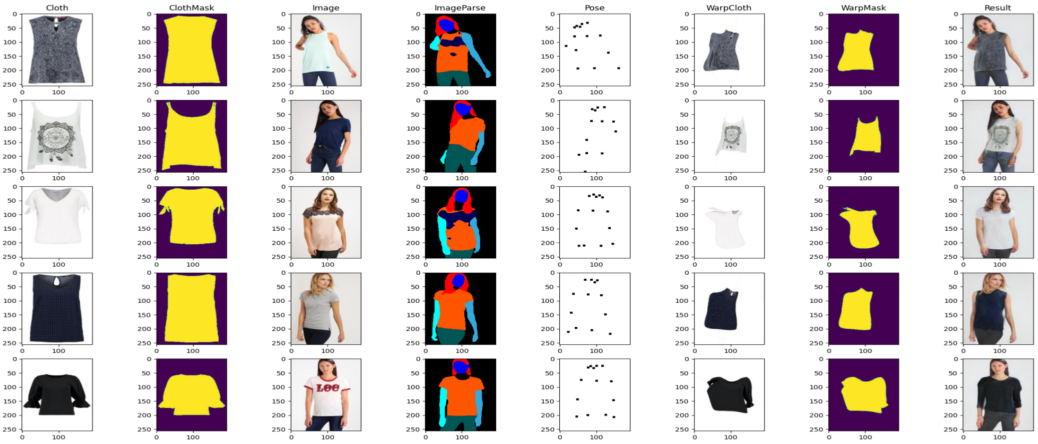


Figure 1: Virtual Try On results

1 Related works

This report lays its foundations on several works carried out in the Computer Vision literature, following the main ones.

1.1 Segmentation

The segmentation task can done in many different ways also according to the context of the data you decide to use. The main methods are shown in the survey [9] and in particular we can find thresholding techniques, color k-means, edge detection and template matching.
In literature we can find also some works related on people segmentation like 'Self-correction' [4], using semantic segmentation. This problem can be approached by using neural network architectures, as in the net U-net [5], firstly it was used in medial images analysis but can be also trained on different data.

1.2 Retrieval

Retrieval Systems on images are based on finding the main features to describe them as color, text descriptions or key-points extraction. For this reason these systems are called Content-based image retrieval (CBIR). CBIRs systems extract features according to the whole batch of data we are considering and then compares them through similarity measures with the input image. As shown in the picture. 2.

The image retrieval system is based on the recognition of the main features in the image; For example, color, text description or extraction of key points. Therefore, they are called content-based image retrieval (CBIR) and consist of a series of blocks. CBIR extracts the features of all the images in the considered set (offline), and then compares them with the features extracted from the images selected from the input (online) through a similarity measure. As the picture shows. 2. Keypoints extraction can be done through images' detector or descriptor as SIFT or ORB [1]. In literature have also been studied systems which uses feature extraction through neural networks [6], the same idea we used in this project. As almost always happens when using images and neural networks the architecture involves a CNN so convolution operation [3].

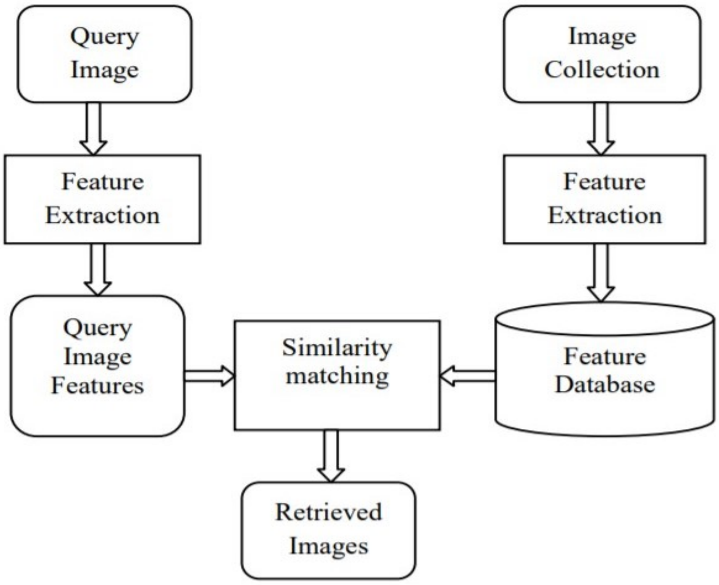


Figure 2: Blocks of a Retrieval system

1.3 Geometric Transform

As described in the abstract, the last point of our pipeline concerns a geometric transformation to get a virtual try on effect of the chosen cloth on the starting image.
In order to reach our goal in geometric transformation part we started as done in the paper 'Toward Characteristic-Preserving Image-nased Vrtual Try-On Network' presented in the ECCV 2018 [7]. The part we studied most of this paper is the one which concerns the GMM(Geometric Maching Module). From a geometric point of view we developed two ways to reach the transformation, the first one using an affine transformation and a second one using a neural network trained to learn a particular and more accurate transformation calles TPS(Thin plate spline). TPS transformation has the goal to find a poli-harmonic spline function which maps a set of points (x,y) on their correspondences (x',y') sampled from input images.

In section 3.3 geometric transform at page. 11 to localize images keypoints to find correspondences we used a neural network called Open Pose. [2].

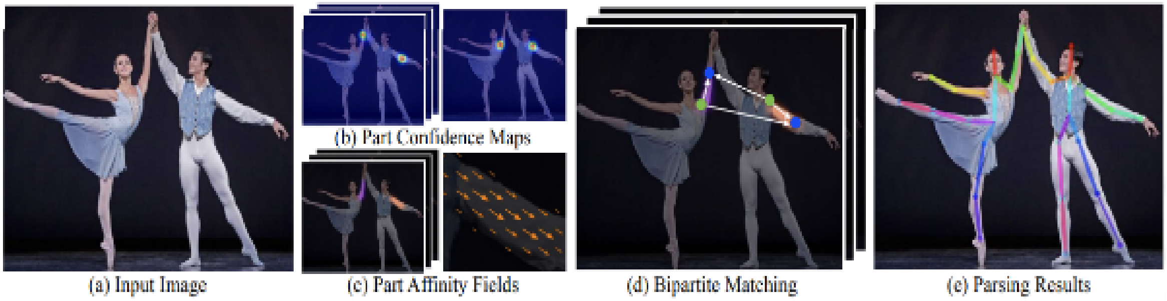


Figure 3: Open Pose Flow

This network is composed of many steps as shown in the picture 3.

- First step: Obtaining feature maps using the first ten layer of VGG-19.

- Second step: use feature maps to generate two elements: ”Part Confidence Maps” and ”Part Affinity Fields”, described as:
  - Part Confidence Maps: A 2D representation of the confidence with which a particular body part is localized in the considered pixels.
  - Part Affinity Fields: A map composed of a cluster of 2D vectorial fields which encodes limbs position and orientation. Encode is expressed as a connection between pairs of body parts.

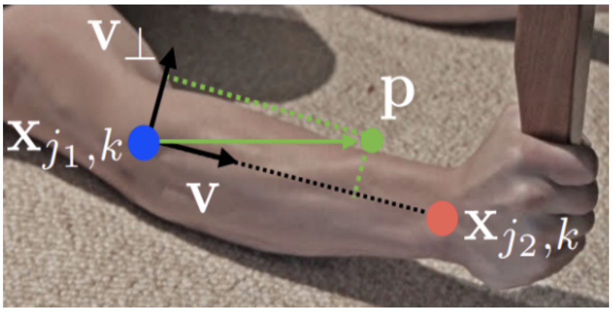


Figure 4: Part Affinity Fields

- Third step: Results obtained in the last step are computed by an algorithm ”greedy bipartite matching” in order to get the pose of a person in an input picture.

This network uses an L2-Loss function, computed between Part Confidence Maps and Part Affinity Fields compared to the ground truth of maps and field values.

2 Dataset

The dataset used is a cluster of image taken from many catalogs of YOOX NET-A-PORTER Group. This dataset is divided in 3 main categories (dresses, upper-body clothes and lower-body clothes), we decided to analyze just the upper body cloth part.
’Upper-body clothes’ is splitted in 4 under categories as follows:

- 15k images RGB of 1024x768 as dimensions of clothes with white background. Those clothes are splitted under 84 folders which are the given 84 classes of clothes (shirts_men-shirts, shirts_tops, shirts_casual-shiirt, . . .).
- 15k RGB 1024x768 of people, with the face partially hidden to avoid privacy problems, which are wearing the clothes. We have a 1:1 ratio between dress and person which wears it.
- For each image of people wearing clothes we have (15K), we have a color mask which gives at each pixels a specific dress cathegory thus obtaining a segmentation mask. In total we have 18 cathegories of dataset dresses (pants,skirt,dress, . . .). To compute those masks we used the SCHP model trained on the dataset ATR, al large single person human parsing dataset [4].
- For each image of people wearing dresses(15K), we extract keypoints of the given pose through openpose [2], those points are saved in a json as a dictionary which hlods the coordinates of joint points.

The just described dataset contains 75K images, with a rough total dimension of 4GB.

3 Pipeline

The project was carried out following several phases. The Segmentation phase is used to identify the various kinds of clothes worn and their location in the image. The Retrieval allows you to identify similar clothes, which belong to the same class. Finally, the geometric transformations build the final image with the new dress worn. These steps are described below.

3.1 Segmentation

The segmentation was performed on two types of images from the dataset.

1. On the pictures of the clothes only:
   the goal is to identify the edges of the dress and then segment it with respect to the white background. To do this, image processing algorithms relating to the field of computer vision were used. Different thresholds were tested (Otsu threshold, Adaptive threshold, Canny) which highlighted the ability to achieve the goal, as seen in Fig. 5.

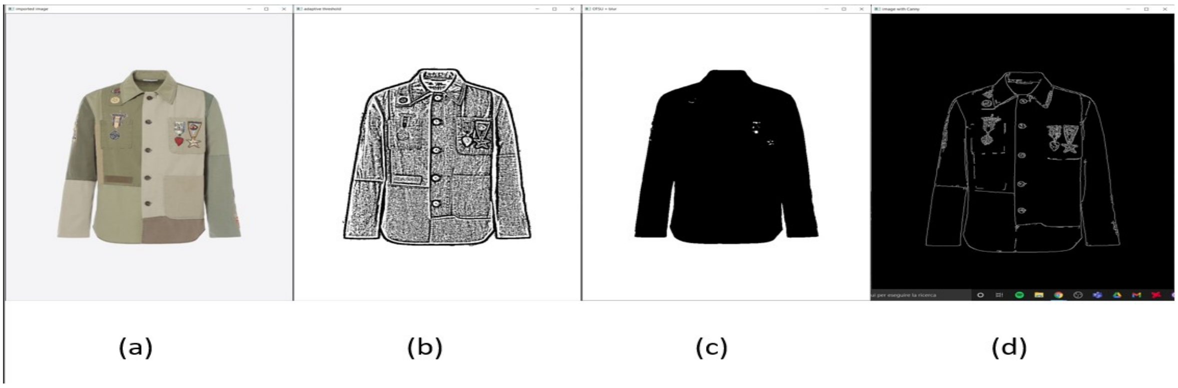


Figure 5: Image processing over picture of the dress
(a): input image (b): adaptive threshold (c): OTSU threshold + blur filter (d): Canny

2. On the pictures of the models wearing the full look:
   the goal is to recognize and identify the various types of clothes worn by each model. First we tried the same approach seen in the previous point, but such techniques as seen in Fig. 6 have the ability to identify only the person with respect to the background, without distinguishing the various clothes worn. Approaches such as template matching, Felsenszwalb's efficient graph, K-means over colors, were therefore used, which improve the results but which bring with them a high variability based on the colors present in the analyzed image Fig. 7.

3.1.1 Semantic Segmentation

To overcome this critically, Semantic Segmentation was used, using the masks present in the dataset, calculated with Human Parsing methodologies [4]. The architecture model described in 'Self-correction for human parsing' is not used [4], but implementing a U-net type architecture as shown on page. 17, which comes from the biomedical field [5], with the aim of training her to learn how to predict the masks of the clothes worn from the models, based

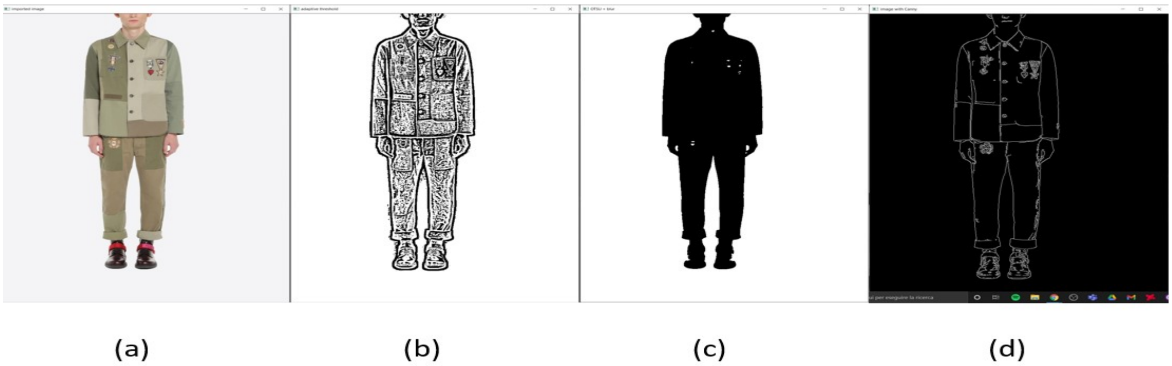


Figure 6: Image processing over picture of the dress
(a): input image (b): adaptive threshold (c): OTSU threshold + blur filter (d): Canny

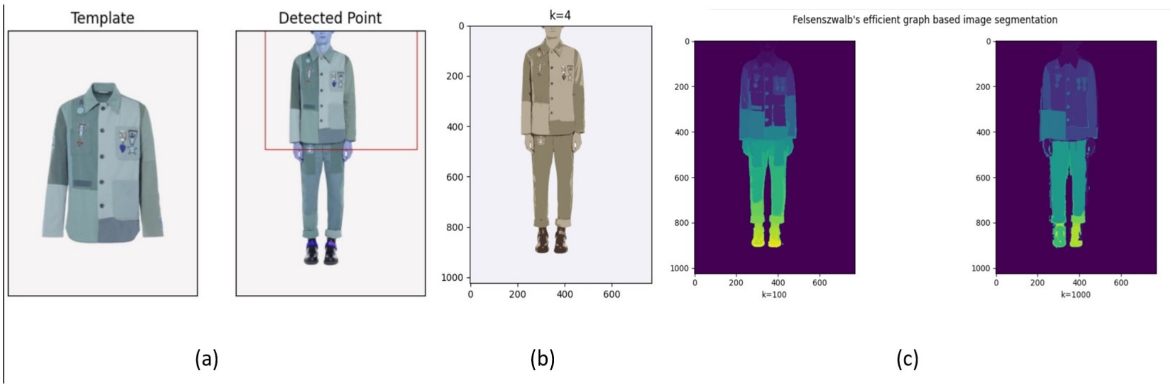


Figure 7: Image processing over picture of the dress
(a): template matching (b): k-means with colors (c): Felsenszwalb's efficient graph

on the input images.
To do this we used the Segmentation models [8] library, which allowed to carry out various experiments by importing pre-trained models on different networks.
Semantic Segmentation made it possible to obtain a Segmentation mask for each input image, assigning a specific category of clothing to each pixel. To improve the results, starting from the imported weights with respect to the backbone, a training phase was carried out for fine tuning with the data belonging to the Yoox dataset. The dataset was divided into 3 parts for training, validation, testing and trained the network for 50 eras.

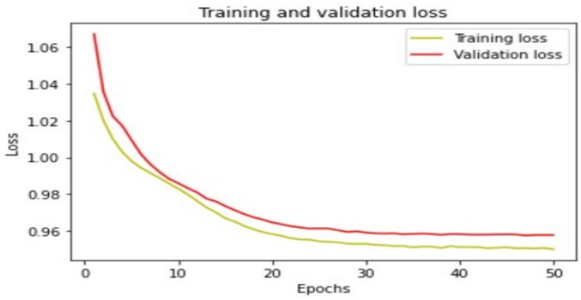


Figure 8: SemSeg, training and evaluation loss

As an implementation choice it was decided to create the masks in grayscale and not in color and with a size of 256x256, this allowed to reduce the total computational load.

Furthermore, the 18 types of clothes present in the dataset have been indicated with a value from 0 to 17 in the images; to prevent the network can learn a distance relationship between them one has applied OneHotEncoder by defining each pixel with a vector of dimension 18 made up of all 0's and a 1 which represents the class to which it belongs.

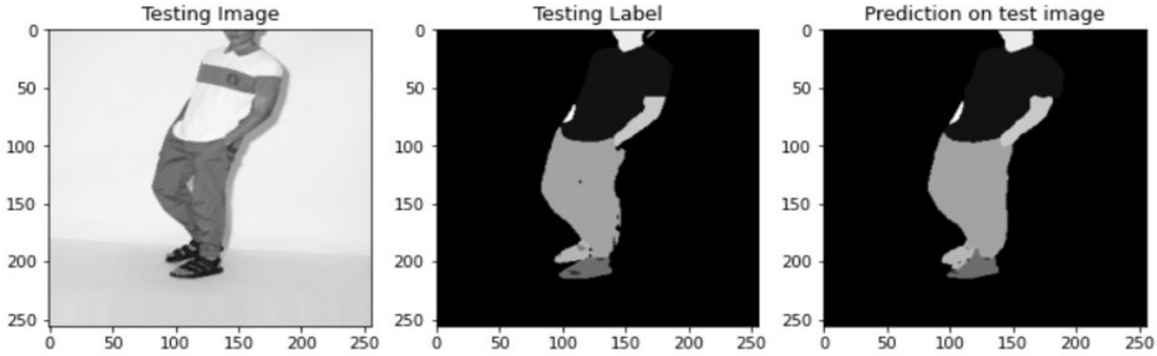


Figure 9: Visual results of Semantic Segmentation
Using a U-net, trained over 50 epochs with backbone=resnet34

To complete the analysis, several experiments were carried out with different backbones on which fine tuning was carried out, to evaluate the best choice among the results obtained on the Yoox data. Given the unbalanced presence of the different classes, with the background always very represented, the measure of accuracy is not effective in representing the quality of the segmentation. In fact, from the first tests we get:

$$accuracy_{resnet34} = 0.981 \tag{1}$$

Therefore, the average of the various Intersection over Union (IoU) of the classes present in the masks was introduced and used:

$$IoU = \frac{AreaofOverlap}{AreoofUnion} \tag{2}$$

calculated on the results obtained with the different backbones on the part of the dataset used for the test set.
The results obtained with IoU, using different backbones, are shown in Fig. 10.

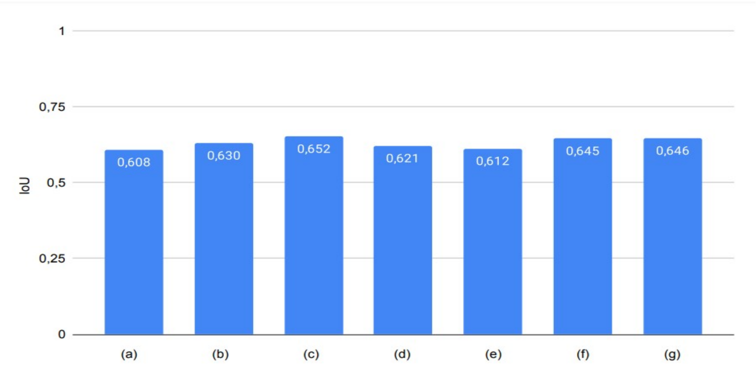
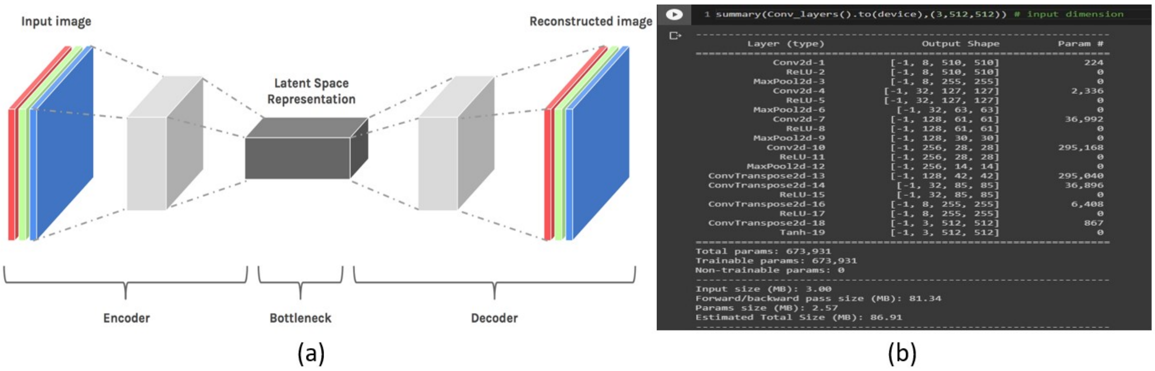


Figure 10: SemSeg comparison results
(a): fix. weights (b): resnet34 with 50 epochs (c): resnet34 with 150 epochs
(d): resnet18 with 50 epochs (e): vgg19 with 50 epochs (f): mobilenetv2 with 50 epochs
(g): FPN architecture (resnet34 with 50 epochs)

3.2 Retrieval

A retrieval system was implemented to create suggestions for similar clothes that exist in the dataset, compared to the clothes that exist in the starting image (the clothes worn by the model). Want to achieve this goal by using neural networks, decided not to implement descriptors Analyze various images through the characteristics of SIFT and ORB. Therefore, we created our own neural network that allows us to develop a content-based image retrieval (CBIR) system.
Two different types of autoencoders were implemented, which were then trained on the data of the Yoox dataset to learn the weights and the features extraction starting from the images available to us [3], in Fig. 11 you can see the structure of one of them.



```
summary(Conv_layers().to(device),(3,512,512)) # input dimension
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 8, 510, 510]             224
              ReLU-2          [-1, 8, 510, 510]               0
         MaxPool2d-3          [-1, 8, 255, 255]               0
            Conv2d-4         [-1, 32, 127, 127]           2,336
              ReLU-5         [-1, 32, 127, 127]               0
         MaxPool2d-6           [-1, 32, 63, 63]               0
            Conv2d-7          [-1, 128, 61, 61]          36,992
              ReLU-8          [-1, 128, 61, 61]               0
         MaxPool2d-9          [-1, 128, 30, 30]               0
           Conv2d-10          [-1, 256, 28, 28]         295,168
             ReLU-11          [-1, 256, 28, 28]               0
        MaxPool2d-12          [-1, 256, 14, 14]               0
  ConvTranspose2d-13          [-1, 128, 42, 42]         295,040
  ConvTranspose2d-14           [-1, 32, 85, 85]          36,896
             ReLU-15           [-1, 32, 85, 85]               0
  ConvTranspose2d-16          [-1, 8, 255, 255]           6,408
             ReLU-17          [-1, 8, 255, 255]               0
  ConvTranspose2d-18          [-1, 3, 512, 512]             867
             Tanh-19          [-1, 3, 512, 512]               0
================================================================
Total params: 673,931
Trainable params: 673,931
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 3.00
Forward/backward pass size (MB): 81.34
Params size (MB): 2.57
Estimated Total Size (MB): 86.91
----------------------------------------------------------------
```

(a) (b)

Figure 11: Autoenconder architecture
(a): Autoenconder structure (b): Our autoencoder architecture

In the Pre-processing phase, one of the 84 gender classes was assigned to each dress image present in the upper-body dataset of clothes present in the dataset; by doing so, the conditions have been created for having a Supervised type system. Furthermore, the values of the various images were normalized and a resize was performed. The training was performed only on the images representing the clothes (15k). Going to appropriately divide the images and classes in the train part and in the test part (40 % of the total).

Using the decoder we managed to reconstruct the initial image using the features extracted in the latent space. This reconstruction depends on the latent space learned and therefore improves during the training of the network itself.

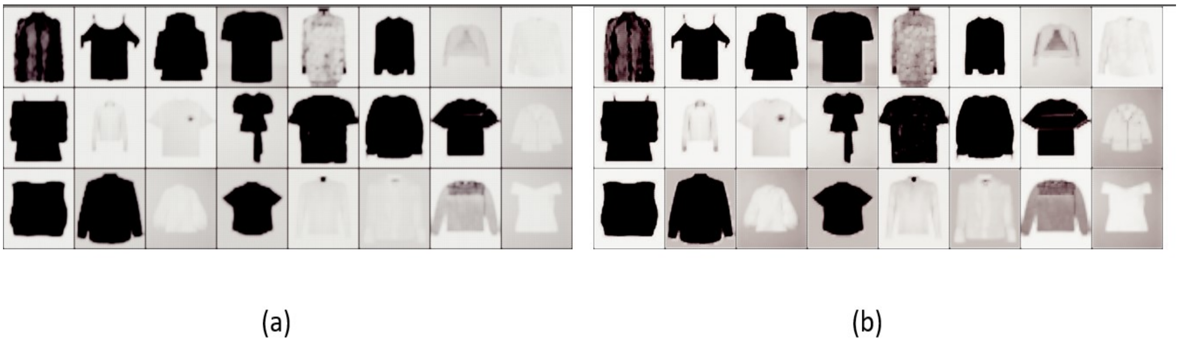


Figure 12: Reconstructed image
(a): at the beginning of training (b): during the training

By creating a vector of features that represents the image (encoder) it was possible to introduce a similarity measure in order to evaluate the distance between the different inputs, represented in the latent space. Two different measures of similarity were used and compared:

1. distanza euclidea
$$\sqrt{\sum_{i=1}^{k}(p_i - q_i)^2} \tag{3}$$

2. similarità del coseno
$$similarity = \cos(\theta) = \frac{A \bullet B}{\parallel A \parallel \parallel B \parallel} =$$
$$= \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

This made it possible to identify the closest and therefore more similar elements, sorting them through a ranking list, compared to the input dress.

To evaluate the capacity of the CBIR system, three different accuracy measures were used to return images of clothes belonging to the same class as the input dress:

- TOP 3 accuracy
- TOP 5 accuracy
- TOP 10 accuracy

These metrics were tested on both networks implemented and trained.
Through these metrics we have respectively evaluated whether in the first 3 or 5 or 10 suggested results there is an element of the input class and considered this as a positive case, on which we then calculate the accuracy with respect to the different inputs of the test set. It was also decided to admit the input dress within the suggestions, which however was excluded during the evaluation phase on everything the test set for calculating the metrics.

$$accuracy(y, y_f) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(y_f = y)$$

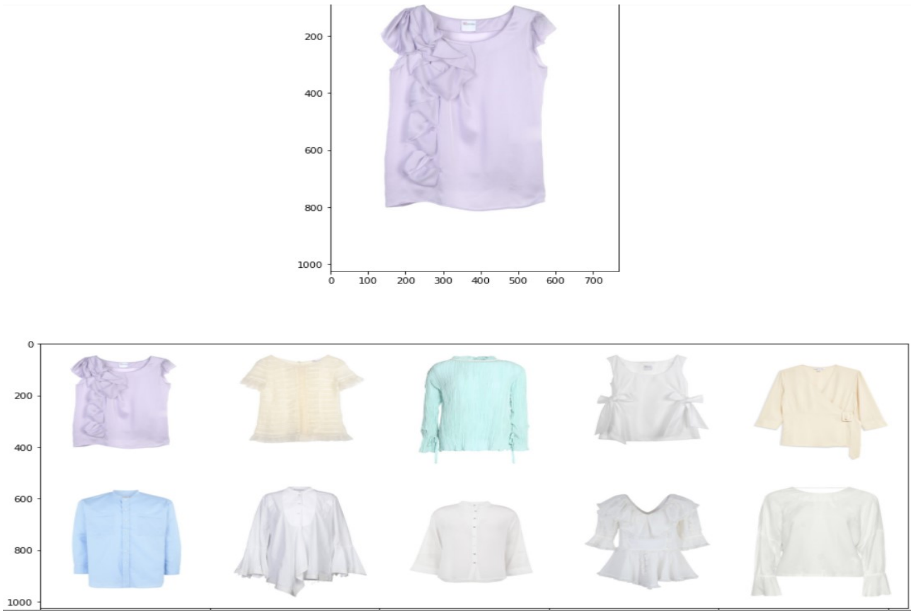


Figure 13: Example of retrieval's result

The quantitative results of the experiments carried out are reported in the table 1 on page 10; the qualitative ones are shown in Figure ref fig: qualitative1 and in the additional materials on page. 18.

Model	TOP3-accuracy	TOP5-accuracy	TOP10-accuracy
Resnet18	0.60	0.72	0.83
Autoencoder (1)	0.57	0.69	0.80
Deep Autoencoder (2)	0.61	0.73	0.84
Deep Autoencoder with 50 epochs	0.58	0.69	0.80
Deep Autoencoder with Triplet Loss	0.46	0.59	0.73

Table 1: Quantitative results of the retrieval system
Evaluation of the results of the 2 models introduced compared with the benchmark (Resnet18). To select the distance to use, experiments were made with both the Euclidean and the cosine distance. The results presented in the table refer to the ***Euclidean distance*** *which is the one that gave the best performance.*

The methodology used provides results in line with the objectives set; considering that within the classes of the dataset there is a high heterogeneity of clothes for shapes and colors, the proposals suggested by the system have in fact elements of recognizable affinity with the selected garment, as can be seen in Fig. 13. Evaluating the metrics in Table **??** we can see how the use of an autoencoder with several layers (Deep Autoencoder) allows to improve the results obtained previously; therefore hopefully leads to the extraction in latent space of features that improve the representation of the input image. Furthermore, this model achieves better results than the network taken as a benchmark (Resnet18).
To expand the experiments carried out, the Deep Autoencoder was further trained, the model that achieves the best results. The training was carried out both over a greater number of eras (50), or using the Triplet Loss as Loss function. During the evaluation phase, worse performances than those shown in the table were obtained.

3.3 Geometric transform

In this phase, the geometric transformation of the new dress on the original image was taken care of. The basic idea is to find the keypoints of the person, that is the joints of the body, then use them to know how to apply the transformation to make the new dress 'wear'.

To carry out the transformations of the dress on the person we therefore used the network proposed in "cp-vton" [7], which aims to preserve the characteristics of the initial image. The network uses a data-set of images with a 192x256 pixel format, organized in:

- cloth: where there are images of individual dresses;
- cloth-mask: containing the masks of the clothes, calculated previously with the methods previously described;
- image: where the real images of the people dressed are collected;
- image-parse: containing the representation of the previous images, to which the Semantic Segmentation described above has been applied;
- pose: inside which we find a file in json format for each image representing the person, this contains the information on the pose that has been extracted using OpenPose [2].

The network is divided into two modules, with the corresponding training and testing phases. The first module, called Geometric Matching Module (GMM), has the aim of finding the most correct geometric transformation in order to segment the clothes on the target subject, in particular the geometric transformation that is found is a Thin Plate Spline function, with the related loss function. The second module, called Try-on module, takes the image produced as input and creates a synthetic model that reflects as much as possible a real image. The GMM was trained independently using data from the Yoox dataset.

3.3.1 Geometric Matching Module

The GMM module architecture has been developed by taking inspiration form Wang et al work described in "Toward Characteristic-Preserving Image-based Virtual Try-On Network". In paticular this block is designed as follows:

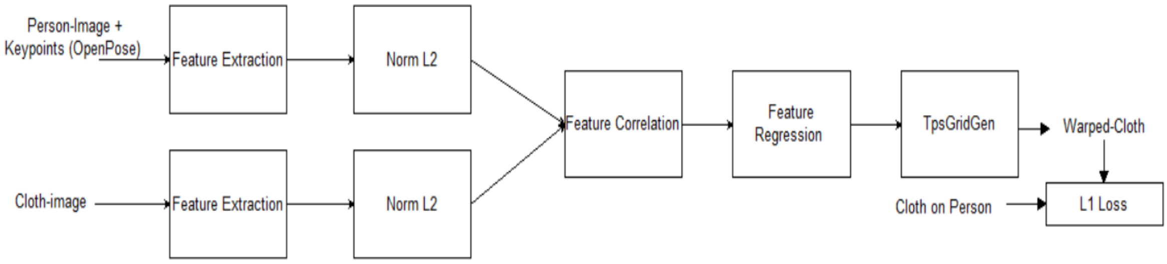


Figure 14: Geometric Matching Module

As shown in the picture four main blocks are being defined to get the required transformation, in particular it has feature extraction, feature correlation, feature regression and Tsp Grid Generator blocks while the L2 norm block just computes the L2 operation. To train this module the loss function used is L1 Loss computed between the warped cloth and the cloth worn by the person in the input image. In order to give a deeper understanding of this part of the network a brief description of each block will be provided in the following paragraphs.

3.3.2 Feature Extraction

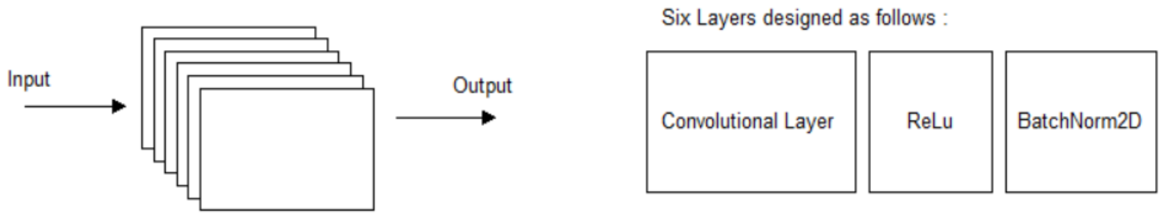


Figure 15: Feature Extraction Block

The figure above represents the feature extraction part of the system, the goal of this block is to get significant features starting form the cloth we want to model and the input image. Since our goal is to build a geometric transformation one of the main thing we need are corresponded points form a semantic point of view, to infer this kind of knowledge to the network the input image channel's gets more layers in which one of those is a point-map computed with Open pose network. At the end of this stage we have Norm L2 layer for normalization.

3.3.3 Feature Correlation

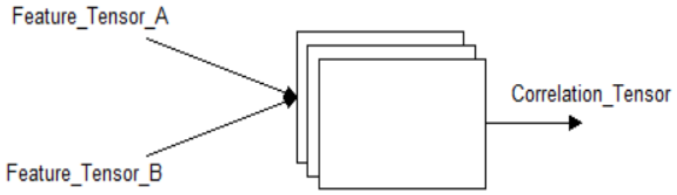


Figure 16: Feature Correlation Block

The feature correlation block is composed of 3 layers with no learned parameters thus the goal here is to perform matrix operation to get a correlated version of the the two input tensor as show in 14. To achieve our goal the work described by Wang et al. paper has been followed. From a mathematical point of view tensors have been reshaped in order to make shapes match and a batch matrix multiplication gave us the desired result.

3.3.4 Feature Regression

The Feature Regression block aims to give us the feature vector corresponding to x and y coordinates offsets for the TPS anchor point indeed the output is 2x5x5.

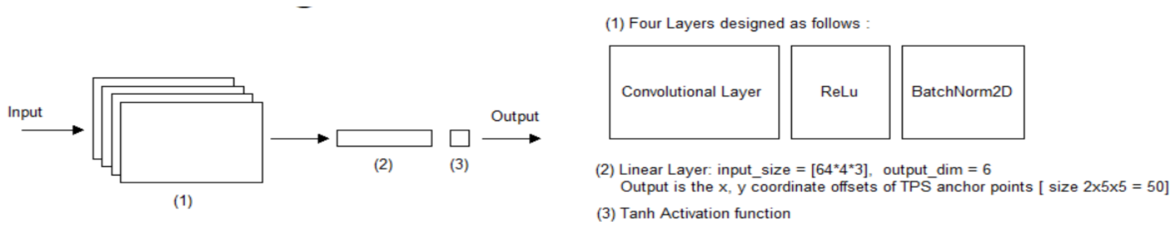

Input
(1)
(2)
(3)
Output
(1) Four Layers designed as follows :
Convolutional Layer
ReLu
BatchNorm2D
(2) Linear Layer: input_size = [64*4*3], output_dim = 6
Output is the x, y coordinate offsets of TPS anchor points [size 2x5x5 = 50]
(3) Tanh Activation function

Figure 17: Feature Regression Block

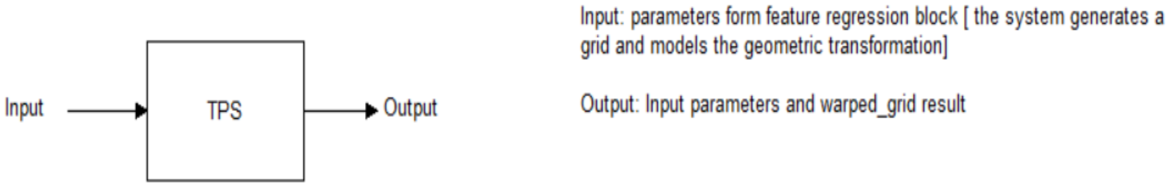

Input
TPS
Output
Input: parameters form feature regression block [the system generates a grid and models the geometric transformation]
Output: Input parameters and warped_grid result

Figure 18: TPS Grid Generator

3.3.5 TPS Gid Generator

Once we have the points computed by previous blocks the final goal is to compute the geometric transformation and this is the task of our block. The geometric transformation chosen is the thin plate spline (according to Wang et al. work), this transformation form a mathematical point of view is a way to fit a surface (our cloth image) through points (feature regression output points) the goal is reached when a second order derivative of an variational method energy function is minimized. This allows to model the input image in a non linear way thus following the person pose in the best way possible.

This GMM block is trained end-to-end by a pixel-wise L1 loss computed between the warped result and the ground truth cloth worn in the target image. The training phase always sees triplets like (p, c, ct) which are person wearing a cloth[p], a cloth warped [c] and the ground truth of the cloth [ct].

3.3.6 Try-on Module

As described above at this point of the pipeline we still miss to fit the warped cloth on the person image, thus there is a second module, separately trained, which takes warped clothes and images as input and syntheses the final image. The work underneath this module is taken form Wang et al. paper so just a brief description will be given. The main goal is to find a way to obtain a good image which is not blurred and not affected by occlusion problems. The architecture proposed is a Unet architecture with the person image and the warped cloth as input to obtain a rendered image of the person and a composition mask M. Once we have these to elements the composition mask is used to combine the just rendered image and the warped cloth. The loss used in this module is a combination of the VGG perceptual loss and an L1 loss.

3.4 Affine Tranform

In addition to the deep network described above, a model generation system has been developed using an affine geometric transformation. The pipeline followed to obtain the image with the new dress 'worn' with this second approach is as follows:

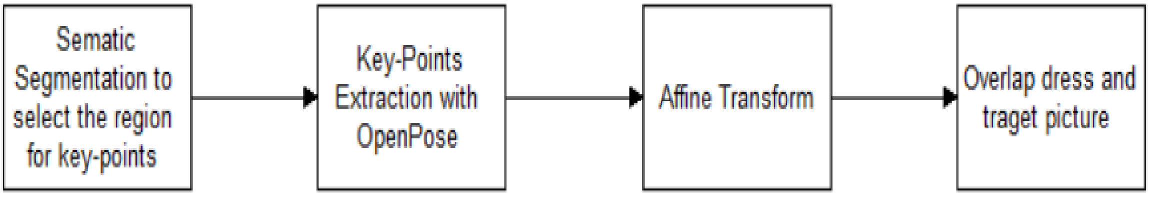

Sematic Segmentation to select the region for key-points
Key-Points Extraction with OpenPose
Affine Transform
Overlap dress and traget picture

Figure 19: Pipeline Affine Transform

As shown in result section [4] this approach is not qualitatively satisfying due to the transformation limitation caused by few degrees of freedom and the poor generative part.

4 Results

The results were then compared using the two different methodologies:

- Deep model:

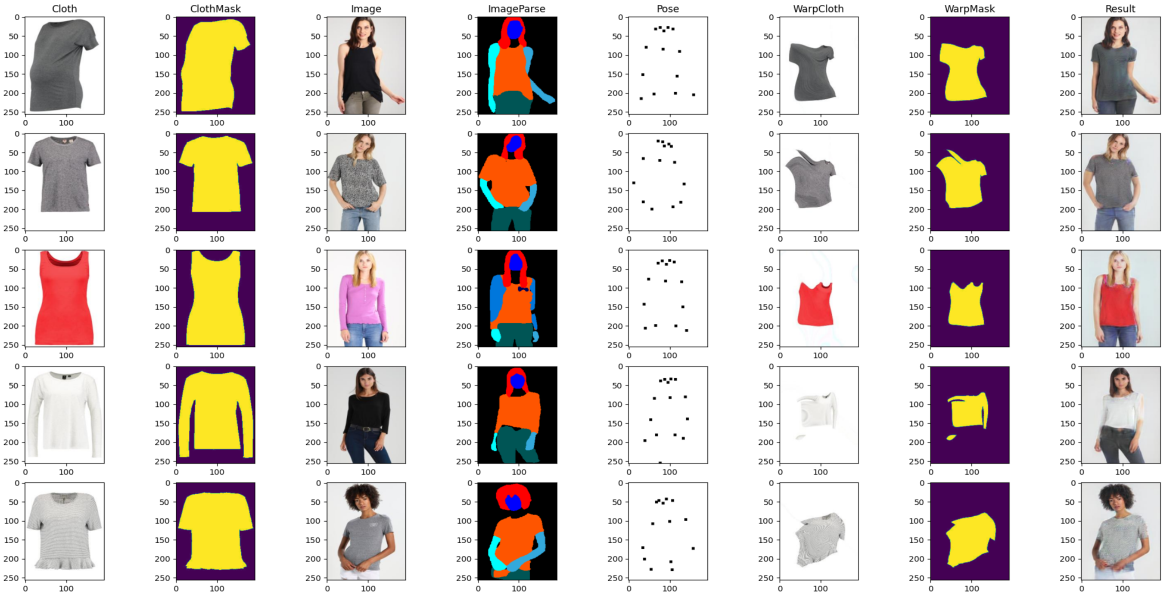


Figure 20: Results with Geometric matching module

- Affine transform:

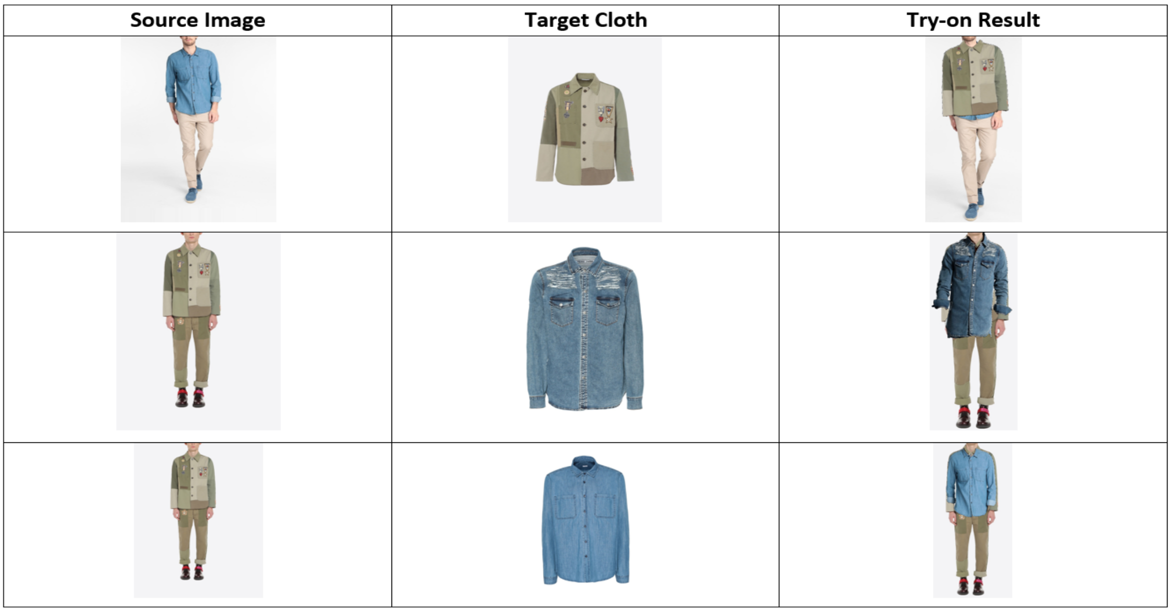


Figure 21: Results with Affine Geometric Transformation

5 Conclusion

The characterizing elements of the completed project are described. To improve the results obtained with classical computer vision algorithms with respect to segmentation, deep architectures were used, which together with the data available through the 'Yoox' dataset allowed a strong improvement on the task, as highlighted by the performances in Fig.10.
A deep architecture was also used for the implementation of the retieval system, using an autoencoder formed by convolutional layers, capable of extracting the main characteristics of each image in the latent space. To quantitatively evaluate the performance of this system, the labels of the various classes present in the dataset were used, obtaining the results in Tab. 1.
In the last block of the pipeline, two different types of geometric transformations were compared. An affine transformation [3.4] has been applied and compared with a learned transformation, on the data present in the dataset, [3.3]. This second methodology was used to simulate and better learn the transposition of the new dress in the initial image.
The main challenge of this work has been defining a way for describing a qualitative good result in terms of derivable loss to train the try-on network and that is the main reason why results sometime are blurred or have some occlusion problems. Another obstacle was in the train data since everything has been trained by trying to put the same cloth the person was wearing on itself and this approach can lead to overfitting. Some ideas to solve this problem are proposed in the section [6].
This line of research, supported by the results obtained, can be exploited to make online sales systems more impactful and propose possible alternative dresses. Especially in those cases where it is useful to carry out tests, such as for clothes.

A special thanks to Dr. Marcella Cornia who followed and supported us during the selection phases of the project.

6 Future possible improvements

In the following paragraphs we describe how this work can be developed from our point of view.

6.1 Geometric transformation

Thinking about the challenge we had to face once we wanted to measure if our results where good or not from a qualitative point of view, since the only goal of this kind of systems is making people look good on never-worn clothes to sell as much as possible, we should focus on this aspect. In order to improve under this point of view a rough solution idea can be described as follows. A new module can be added at the end of our pipeline with the goal of generating a better version of the input image, the type of architecture sholud be a GAN-Like trained on a dataset of a set of labelled (Real-Fake) images however the main challenge of this approach is to modify the image in way in which characteristic are preserved(eg. dress texture, logo and shape must be kept).

6.2 Retrival System

The retrival system can be improved changing the loss with a triplette loss and develop a different architecture that extract the characteristic features from the images.

List of Figures

References

[1] Content-based image retrieval system using orb and sift features. 2020.

[2] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[3] Manjunath Jogin, Mohana Mohana, M Madhulika, G Divya, R Meghana, and S Apoorva. Feature extraction using convolution neural networks (cnn) and deep learning. pages 2319–2323, 05 2018.

[4] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. Self-correction for human parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[6] Sumithra Devi K.A. Vijayakumar Bhandi. Image retrieval using features from pretrained deep cnn, 2020.

[7] 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.

[8] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014.

[9] Song Yuheng and Yan Hao. Image segmentation algorithms overview, 2017.

7 Additional materials

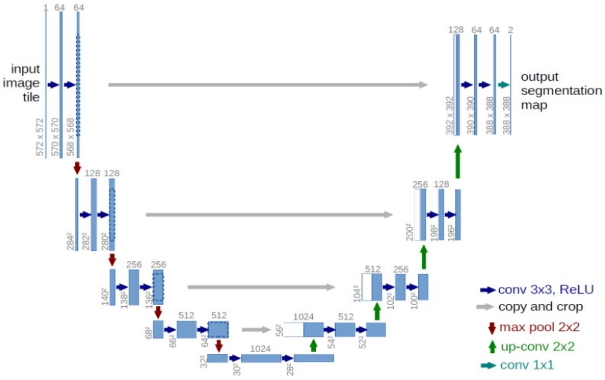
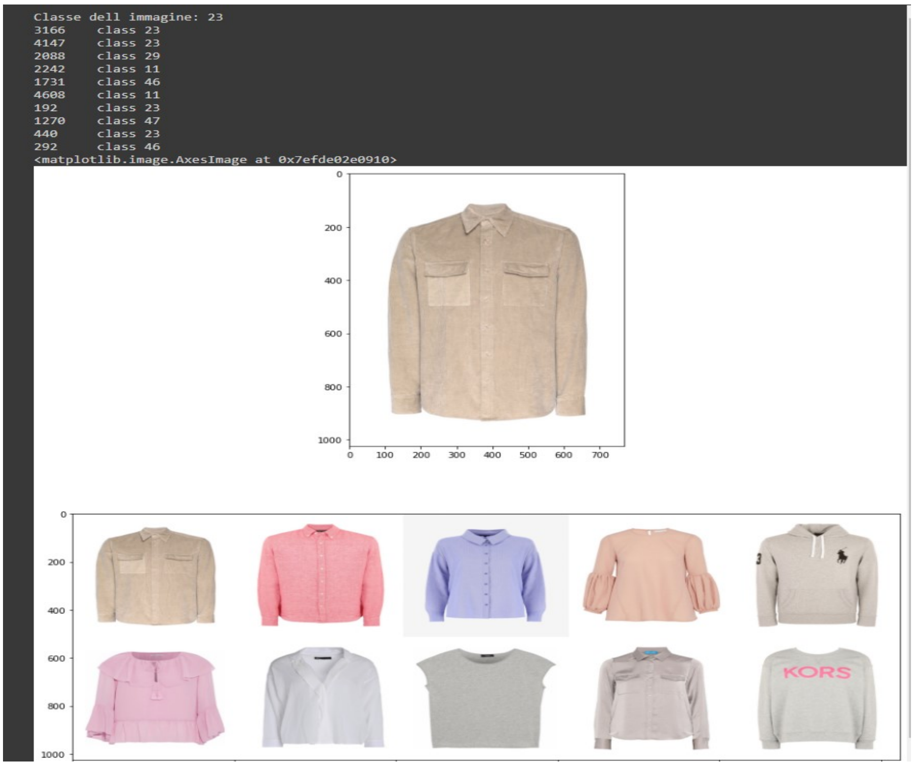


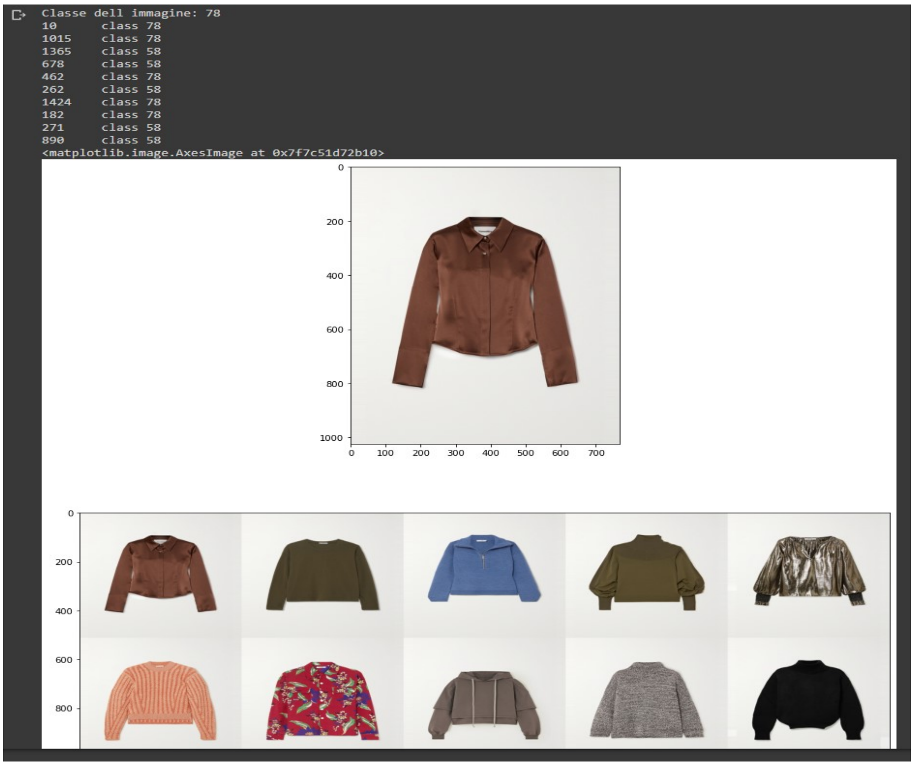
Figure 22: U-net architecture



```
Classe dell immagine: 23
3166    class 23
4147    class 23
2088    class 29
2242    class 11
1731    class 46
4608    class 11
192     class 23
1270    class 47
440     class 23
292     class 46
<matplotlib.image.AxesImage at 0x7efde02e0910>
```

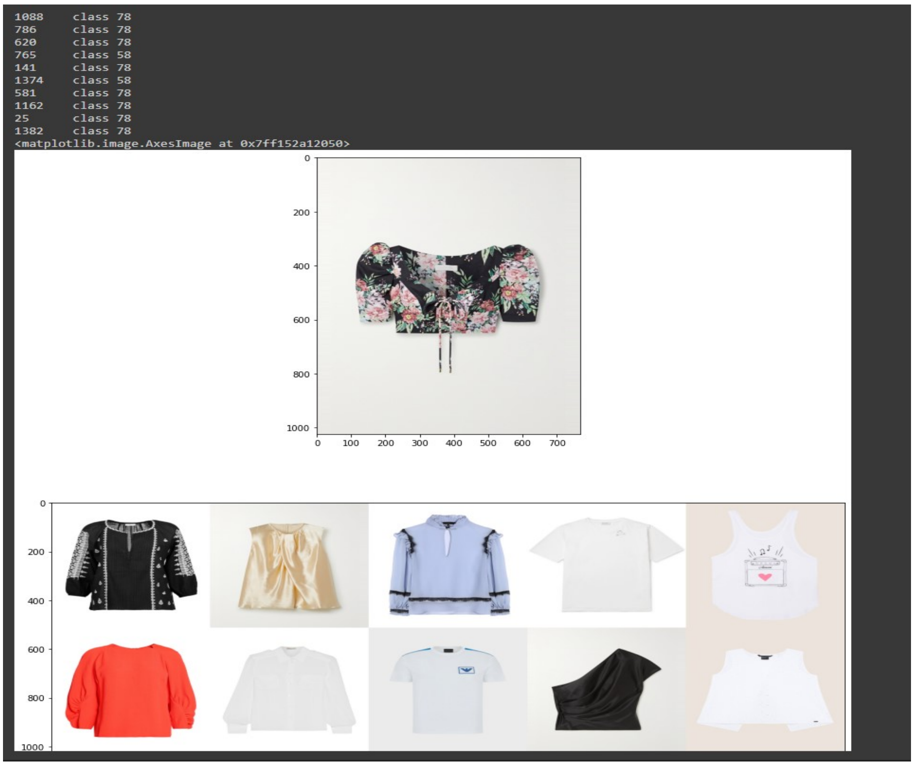
Figure 23: Example of retrieval 1

La didascalia superiore nell'immagine evidenzia nella prima riga la classe dell'abito in input ed in quelle seguenti le classi a cui appartengono i 10 capi di abbigliamento suggeriti e visualizzati.



```
Classe dell immagine: 78
10      class 78
1015    class 78
1365    class 58
678     class 58
462     class 78
262     class 58
1424    class 78
182     class 78
271     class 58
890     class 58
<matplotlib.image.AxesImage at 0x7f7c51d72b10>
```

Figure 24: Example of retrieval 2



```
1088    class 78
786     class 78
620     class 78
765     class 58
141     class 78
1374    class 58
581     class 78
1162    class 78
25      class 78
1382    class 78
<matplotlib.image.AxesImage at 0x7ff152a12050>
```

Figure 25: Example of retrieval with an input image from outside of the dataset