A MACHINE LEARNING POSE TRANSFER SYSTEM

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Sergi Izquierdo Bas

In partial fulfillment

of the requirements for the degree in

Telecommunications Technologies and Services Major in Audiovisual Systems **ENGINEERING**

Advisors: Javier Ruiz Hidalgo and Marta Cosano Serra

Barcelona, June 2022

Abstract

In recent years, the introduction of Machine Learning together with the use of conventional image processing techniques has given a boost to the generation of images. However, as these models are getting better their applications are getting very varied being applied in complex fields but also in daily people's life.

One of the main fields in which these new technologies are starting to be developed is fashion, especially on virtual try-on. The StageInHome company is trying to develop a virtual try-on system where people will be able to see how clothes fit them from only one single picture or selfie.

This study attempts to provide an end-to-end framework based on Generative Adversarial Networks (GANs) that can provide different perspectives from a picture of a person. Specifically, it intends to provide a different human pose given a front pose of a human.

Resum

En els darrers anys, la introducció del Machine Learning juntament amb l'ús de tècniques convencionals de processament d'imatges ha donat un impuls a la generació d'imatges. Tot i això, a mesura que aquests models van millorant les seves aplicacions són cada vegada més variades aplicant-se en camps complexos però també en la vida quotidiana de les persones.

Un dels principals camps en què s'estan començant a desenvolupar aquestes noves tecnologies és el de la moda, especialment a les proves virtuals. L'empresa StageInHome està tractant de desenvolupar un sistema de prova virtual on les persones podran veure com els queda la roba a partir d'una sola foto o selfie.

Aquest estudi intenta proporcionar un marc integral basat en xarxes generatives antagòniques (GAN) que pugui proporcionar diferents perspectives a partir d'una foto d'una persona. En concret, pretén proporcionar una posició humana diferent atesa una posició frontal d'un humà.

Resumen

En los últimos años, la introducción del Machine Learning junto con el uso de técnicas convencionales de procesamiento de imágenes ha dado un impulso a la generación de imágenes. Sin embargo, a medida que estos modelos van mejorando sus aplicaciones son cada vez más variadas aplicándose en campos complejos, pero también en la vida cotidiana de las personas.

Uno de los principales campos en los que se están empezando a desarrollar estas nuevas tecnologías es el de la moda, especialmente en las pruebas virtuales. La empresa StageInHome está tratando de desarrollar un sistema de prueba virtual en el que las personas podrán ver cómo les queda la ropa a partir de una sola foto o selfie.

Este estudio trata de proporcionar un marco integral basado en redes generativas antagónicas (GAN) que pueda proporcionar diferentes perspectivas a partir de una foto de una persona. En concreto, pretende proporcionar una pose humana diferente dada una pose frontal de un humano.

Acknowledgements

First of all, I would like to thank Javier Ruiz Hidalgo for his supervision of this project, for answering my questions and for his patience with this project, for resolving any doubts and for getting the project back on track when I got lost.

I would also like to thank Marta Cosano and all the StageInHome team for allowing me to work for their company and for providing me with the tools to develop my work as well as providing me excellent motivation to ensure work.

Of course, my sincere thanks go to my parents, who have always trusted blindly in me and in my possibilities, many times when I didn't. Above all, they have put up with me during exam periods, they have always encouraged me and tried to help me in any way they could and trying to help me as much as they could although they didn't really understand what the project was about

Also, for the financial effort they have made all this time so that I could study at the university.

Revision history and approval record

Revision	Date	Purpose
0	01/05/2022	Document creation
1	25/05/2022	Document revision
2	10/05/2022	Document revision
3	14/05/2022	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Sergi Izquierdo Bas	
Javier Ruiz Hidalgo	
Marta Cosano Serra	

Written by:		Reviewed and approved by:	
Date	01/05/2022	Date	14/05/2022
Name	Sergi Izquierdo Bas	Name	Javier Ruiz Hidalgo
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

Machine learning together with conventional image processing techniques have created a revolution in technology in recent years that can affect in a very satisfactory way the quality of our daily lives.

This project aims to apply Machine Learning and Deep Learning to our daily life by transferring one person from one pose to another in a realistic manner. It is a challenge due to the fact that the system has to generate different body parts and views from only one image, it has to generate perspectives it has never seen.

When a person tries on a piece of cloth in a real shop it can see itself reflected in the mirrors of the fitting rooms, but this doesn't happen with virtual try-on models. These systems produce images where people see themselves with other pieces of cloth that has been changed so the person will only be able to see itself from one perspective.

We propose a system that will be able to take the output image from the try-on virtual model and extract the pose to transform the person from one pose to another. That would make the customer see itself from different sides and not only form the perspective of the input image of the try-on virtual model.

1.1. Project goals

The main objective of this project is to learn and develop a Pose Transfer system using GANs and being able to good create a good database through pre-processing.

In order to achieve this a number of objectives have been set:

1. Process and analyze image databases with image processing techniques
2. Estimate the pose of a person
3. Improve the results by training the model using more dataset
4. Make inference on Pose Transfer model
5. Evaluate the results obtained

1.2. Specifications and requests

The requests of the project are the following ones:

- Implement a base system that detects the human pose

- Implement a base model that changes the pose of a person
- Be able to choose an image target to generate the pose as on the image

The project specifications are:

- Design neural networks with low error rate
- Create a database for training
- Achieve a mean opinion score (MOS) greater than 3.5

1.3. Gantt diagram

<table>
<tr><td colspan="2">Project: Pose Transfer Model</td><td colspan="2">WP ref: WP1</td></tr>
<tr><td colspan="2">Major constituent: Previous research</td><td colspan="2">Sheet 1 of 4</td></tr>
<tr><td colspan="2" rowspan="2">Short description: Previous research of older projects in which they change the pose of a person using IA, better if I can find code in to implement and make inference on them obtaining what I want.</td><td colspan="2">Planned start date: 2/03/2022
Planned end date:12/04/2022</td></tr>
<tr><td colspan="2">Start event: 29/03/2022
End event: 29/04/2022</td></tr>
<tr><td>Internal task T1: Search for Pose Estimation systems
Internal task T2: Implementation of the Pose Estimation model
Internal task T3: Search for Pose Transfer systems
Internal task T4: Implementation of the Pose Transfer model</td><td>Deliverables:
T2: Results of model inference
T4: Results of model inference</td><td>Hours spent:
T1: 18h
T2: 36h
T3: 30h
T4: 66h</td></tr>
</table>

Table 1: Work Package 1

<table>
<tr><td>Project: Pose Transfer Model</td><td>WP ref: WP2</td></tr>
<tr><td>Major constituent: Dataset</td><td>Sheet 2 of 4</td></tr>
<tr><td>Short description: Find/Create a dataset by scrapping and filtering/classifying the images to obtain best possible database for training the Pose Transfer model founded.</td><td>Planned start date: 12/04/2022
Planned end date:
02/05/2022</td></tr>
</table>

	Start event: 27/04/2022 **End event**: 10/05/2022	
Internal task T1: Research **Internal task T2**: Database creation **Internal task T3**: Database for testing	**Deliverables**:	**Hours spent**: T1: 18h T2: 18h T3: 15h

Table 2: Work Package 2

Project: Pose Transfer Model	WP ref: **WP3**	
Major constituent: Software	Sheet 3 of 4	
Short description: Create and adapt a training code to apply transfer learning/train the Pose Transfer model and obtain better results than the previous ones using the pretrained one.	**Planned start date**: 02/05/2022 **Planned end date**: 29/05/2022	
	Start event: 11/05/2022 **End event**: 03/06/2022	
Internal task T1: Code for training **Internal task T2**: Code for testing **Internal task T3**: Code documentation **Internal task T4**: Tests and results	**Deliverables**: T1: Training code T4: Conclusions of the obtained results	**Hours spent**: T1: 36h T2: 42h T3: 12h T4: 24h

Table 3: Work Package 3

Project: Pose Transfer Model	WP ref: **WP4**	
Major constituent: Documentation	Sheet 4 of 4	
Short description: The objective is to deliver the documentation on time: 1. Project Proposal and Work plan 2. Project Critical Review 3. Final Report template	**Planned start date**: 29/05/2022 **Planned end date**: 21/05/2022	
	Start event: 05/05/2022 **End event**: 10/05/2022	

Internal task T1: Project Proposal and Work plan writing **Internal task T2**: Project Critical Review writing **Internal task T3**: Final Report template writing	**Deliverables**: T1 T2 T3	**Hours spent**: T1: 10h T2: 15h T3: 66h

Table 4: Work Package 4

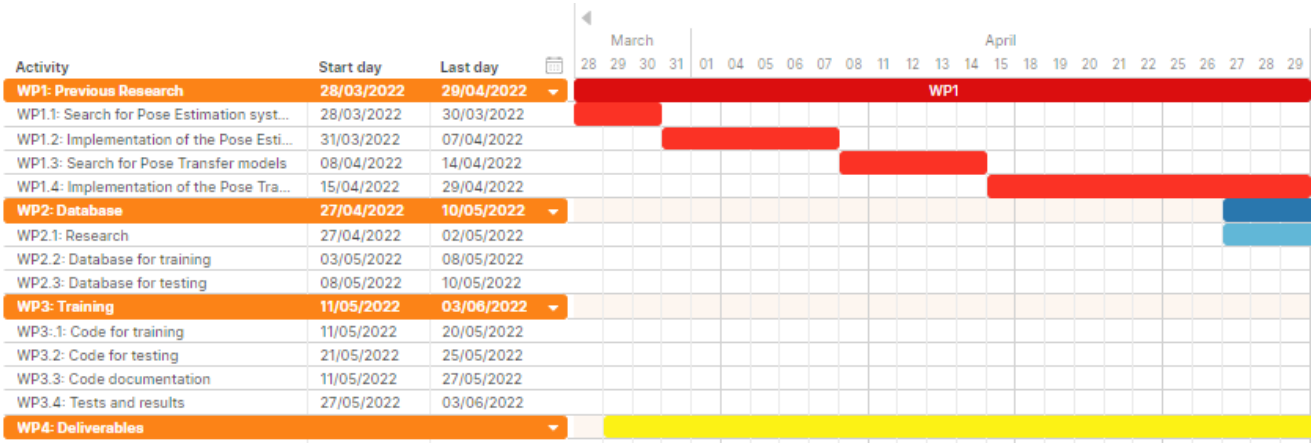


Figure 1: GANTT diagram 1

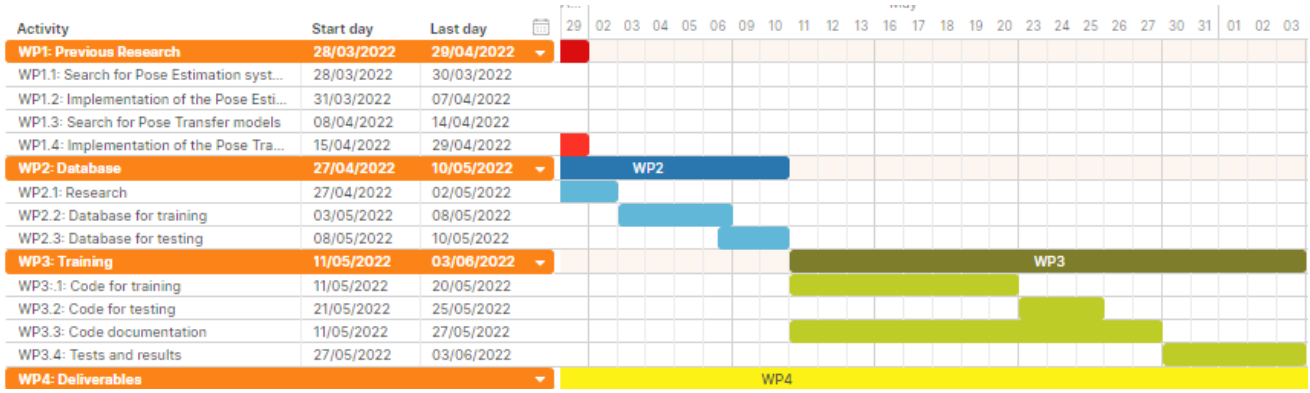


Figure 2: GANTT diagram 2

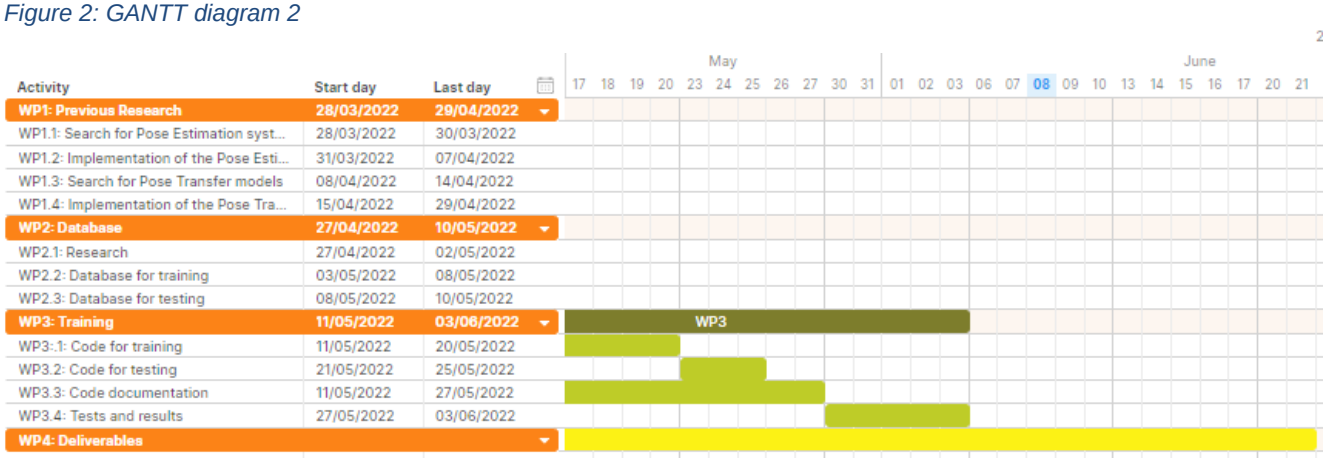


Figure 3: GANTT diagram 3

The first Gantt diagram done in March was quite different from the one above.

The first main difference is the main objectives of the project. In March the project was focused on the generation of human models, this would be very useful for small fashion shops that want to sell and promote their clothes but do not have the purchasing power to pay for a photo shoot with models. Although this big difference, the work packages are very similar between the planned for the human generation and the pose transfer final project.

In the beginning the lack of experience I had in large projects and in the use of Deep Learning networks made me prolong the project. I had several problems that would be easier to solve now, but at that moment I wasted time.

On the other hand, I initially planned the inference and the implementation of the found repository would take me less time than it does. It was planned to be finished on the 12th of April but the code was confusing and difficult to transform on what I really wanted.

Luckily the paper I found mentioned the dataset they used for training and that made the WP2 took much less time. I initially thought I would have to create all the database on my own but finally I only had to filter the database and adapt it to the training model I finally implemented.

2. State of the art of the technology used or applied in this thesis:

In this section, we will provide a summary of the actual researches and technology used for Pose Estimation and Pose Transfer. Some of them, are useful systems to approach our objectives and therefore explained in our methodology chapter of this project.

We will first start with a little bit of basics to understand all the concepts mentioned in this work. First we will explain some of the current revolutionary technology that is in a deep research phase for its successful outcomes, these are Machine and Deep Learning techniques and Generative Adversarial Networks. Following, we will see how those techniques have been applied on both systems we want to develop.

2.1. Machine and Deep Learning introduction

Deep learning is a Machine Learning technique that predicts and classifies information.

It consists of networks of interconnected neurons, organised in layers. The way in which these layered neurons are organised makes up the different architectures, they are generally classified into two big different groups, feedforward and feedback networks.[1]

On feedforward networks the signals travel in one only direction, this signal passes through the different layers where calculations are performed and each processing element (neuron) computes depending on the weighted sum of the input signals. These calculated values are the output of the layer and will be the inputs of the coming layer, this process is known as feed-forward. The process continues among all the layers and finally determines the output of the neural network.

On feedback networks the signals travel in both directions, forward and backward, using loops, this fact makes connections between neural networks have different possibilities making them a continuous changing non-linear dynamic system until it reaches an equilibrium state.

The simplest version of a neural network is a perceptron or simple artificial neuron.

The function of a neuron can be expressed as:

$$y = f(\sum(x_i W_i) + b) \qquad \textbf{(2.1)}$$

Where x_i are the inputs, W_i is the weights matrix, b is the bias and sigma is an activation function. This more commons activation functions are the sigmoid, the threshold, the rectifier, or the hyperbolic tangent.

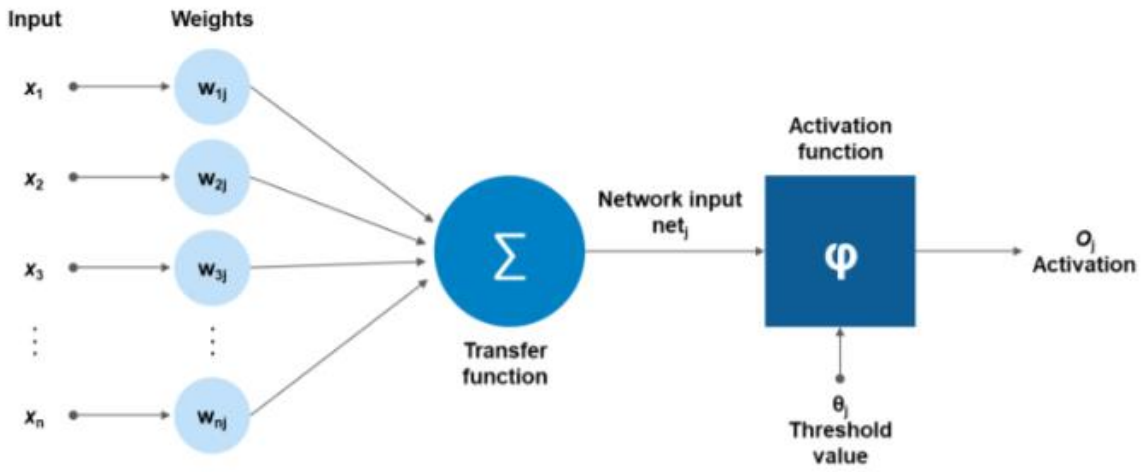


Figure 4: Artifical neuron

A multilayer perceptron (MLP) network has an input layer, one or more layers of perceptrons, called hidden layers, and a final layer with several perceptrons. [2]

The learning algorithm of a neural network consists of:

1. Start by assigning random values for the network parameters (W_i, b_i).

2. Take some first examples from the dataset and pass them through the input layer.

3. Perform forward propagation, where neurons are activated from left to right until a prediction is obtained.

4. Compare the obtained predictions with the expected labelled values and calculate the loss function.

5. Perform backpropagation. The error is propagated from right to left, and the gradient technique is used to adjust the weights of the connections between neurons and to minimise the loss function.

6. Repeat the steps: reinforced learning or batch learning (repeat the steps and update the weights after each observation). Where batch is the size of training data in one iteration of the training to update the gradient.

7. When all training data has passed through the network it is an epoch. Repeat for multiple epochs. [1] [2]

2.2. Convolutional Neural Networks

Due to the appearance of different applications and problems, different types of neural networks have appeared according to their architecture, one of them being Convolutional Neural Networks (CNNs).

CNNs are one of the most used networks for computer vision. Although there are different types of them with some differences on their architectures, they all have the same function, they take advantage of the spatial information in the images by means of e nearby relations as well as sharing the early neurons' parameters onto the following layer.

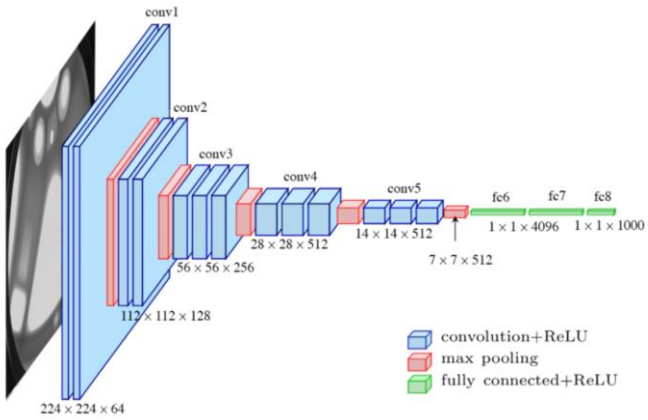


Figure 5: CNN Architecture

2.3. Generative Adversarial Networks

Generative Adversarial Networks or GANs are a revolutionary class of machine learning algorithm used to generate images and photographs that at naked eye look superficially photo realistic. GANs have the task to learn the patterns on images that will act as the dataset, this learning has to be in a way they can reproduce the patterns.

The base architecture of a GAN setup is very standardised, there are two different main blocks that will allow the network to generate realistic images, those are the Generator and the Discriminator, each one has different roles:

Generator: The generator will create new images and will try to make the discriminator mistakes its inputs as real when they are not. The creation of images starts with a 2 dimensions latent vector normally obtained randomly though a gaussian distribution that will become a 2D element by passing through several 2D-deconvolutions.

Discriminator: It will classify the generated images between real or fake. This will guide the generator to produce more realistic images. It will only appear on the training phase since its function is to improve the results from the generator.

In the perfect equilibrium, the discriminator will be unsure if the images are real or fake.

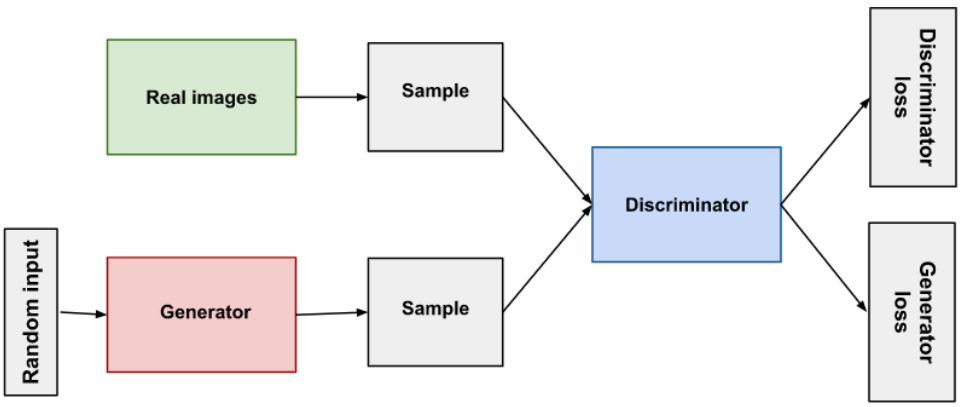


Figure 6: GANs architecture

In order for the generator to create real images and the discriminator classify between real or fake, the network must be trained by using a database from which the generator will learn the patterns that will have to reproduce while the discriminator will progressively identify better samples between real or fake images.

2.4. Pose Estimation methods

The Human Pose Estimation has been extensively studied on computer vision and deep learning methods, it aims to estimate the pose of a person (*Figure 7*), the configuration of the human body parts and how those are located, from an input data, being those an image or even a video.

Figure 7: Pose Estimation skeleton

The traditional approach to human pose estimation is to locate the different main body parts and the spatial dependencies between those on a person. The usage of Convolutional Neural Networks (CNNs) is the most accurate to carry out this task thanks to their accuracy and being used to capture global spatial dependencies [8].

One of the most recent models of Pose Estimation [16] is able not only to estimate the human body configuration, but also the human body motion information, which provides many applications in fields where there is a great deal of human interaction such as augmented reality (AR) or virtual reality (VR). But as our objective is to work with images, this is not the most accurate system to use.

One of the most extended and important models of pose estimation is OpenPose [9] where they aim to extract the different pose on a scene where different people can appear by first detecting the persons on the image and then estimating the human pose independently from the rest of the image.

The system will be explained in more detail on the methodology chapter as it is the architecture used on the project.

2.5. Pose Transfer methods

Recently there have been a lot of works on generative image modelling with deep learning techniques. These works fall into two main categories. The first line of work uses 3D models to generate new image modelling while the second line of work is more focused on using GANs and deep learning using 2D images.

The generation of human people has had a big implementation on 3D models during the last years thanks to the interest of people about the metaverse where each user has its own avatar. One of those examples would be [3] in which given a 3D and combining a VAE [4] and GAN tried to generate different image of the same person with different clothes. Others are not only focused on the human pose but also on animals and other objects, those are more aimed on animated movies or videogames design. [7]

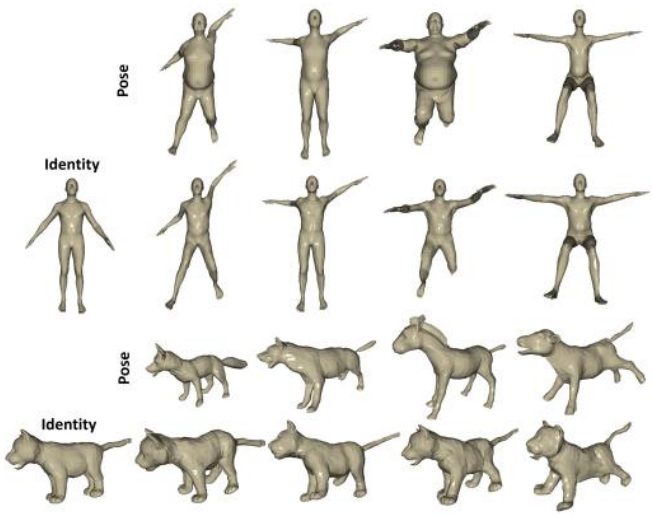


Figure 8: 3D Pose Transfer

But in our application that only consists on a person would be a high-cost process due to the fact of creating specific 3D of each user. The user should take himself more than one image to create his own 3D model when at the end, it would only be used once each time the system is used.

Leaving aside the generation of human models using 3D generation we can find more similar models aimed on pose transfer as Multi-View Image Generation from a Single-View [5] in which they try to generate realistic-looking images from different perspectives and views from the input image based on a specific type of GANs.

The model generates the target pose images in several steps instead of a single pass which could make appear several artifacts that would make the output image not as realistic as it could be.

Liqian Ma [6] implemented a very different model form the above explained by using two different stages, the first focused on the human posed generating an initial result to work on by using an adaptation of a U-Net. The second stage is a conditioned network that takes the output image from the first stage and uses it as input in another U-Net where the fully connected layer has been removed to preserve more details from the conditioning image.

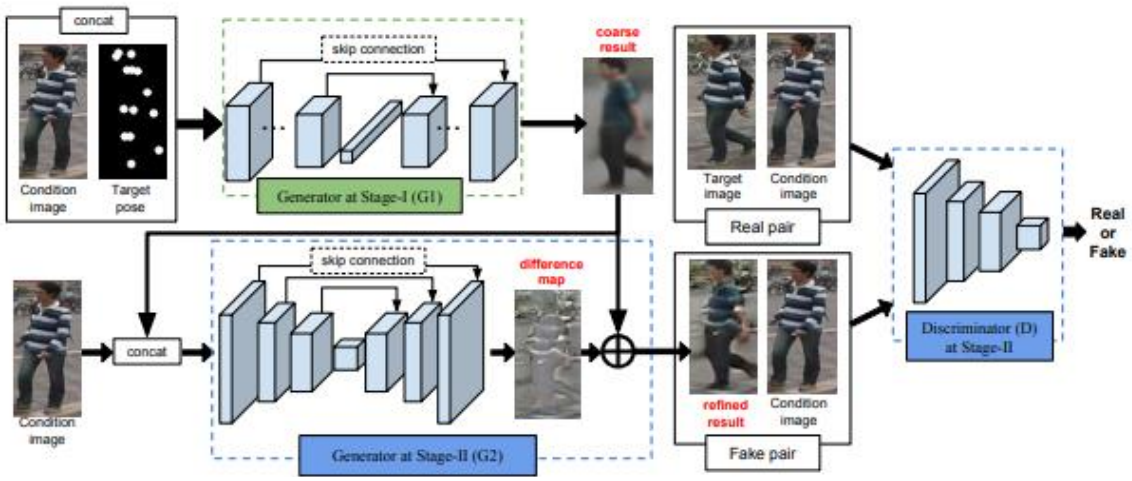


Figure 9: Pose Guided Person Image Generation model, extracted from [6][6]

The second stage contains a Discriminator whereas input has two pair of images, one made by the target image and the conditioning image while the other pair of image contains the conditioning image and the generated one.

3. Methodology / project development:

In this chapter we will detail the main methods used in the project beginning with the Pose Estimation algorithm based on the OpenPose system but modified to be implemented using Pytorch. This was our starting point, as it was necessary later to extract the pose of the input image to the Pose Transfer system and create the database for the training.

Once the Pose Estimation system is developed, we will follow with the first part of the system. We first had to create and adapt the database adjusting it to the needs and requirements of the Pose Transfer algorithm and the results we wanted to obtain.

Furthermore, we will enter on the core of the Pose Transfer model explaining the algorithm suggested by Zhen Zhu [10] understanding its architecture and how it works.

Our final project joins both systems as shown on the following diagram:

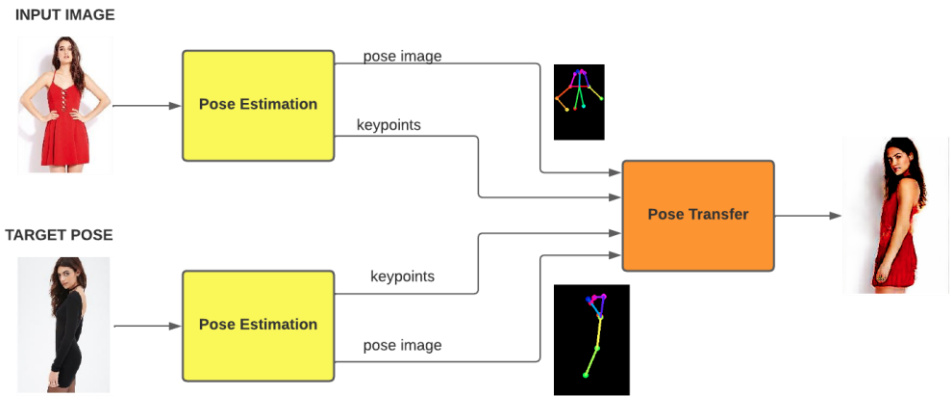


Figure 10: System scheme

3.1. Pose Estimation

Before explaining the system operation, it must be clear that we are not training any neural network with this model. Principally we have adapted it to our purpose and application by using it only for inference purposes.

3.1.1. Model used

We have used a pre-trained VGG-19 Convolutional Neural Network [11] for the human pose extraction. This network is made of 10 different layers deep trained with a wide range of images to generate a set of feature maps. The VGG-19 is a type of CNN quite famous because its developers made its architecture and weights of the trained network freely available online.

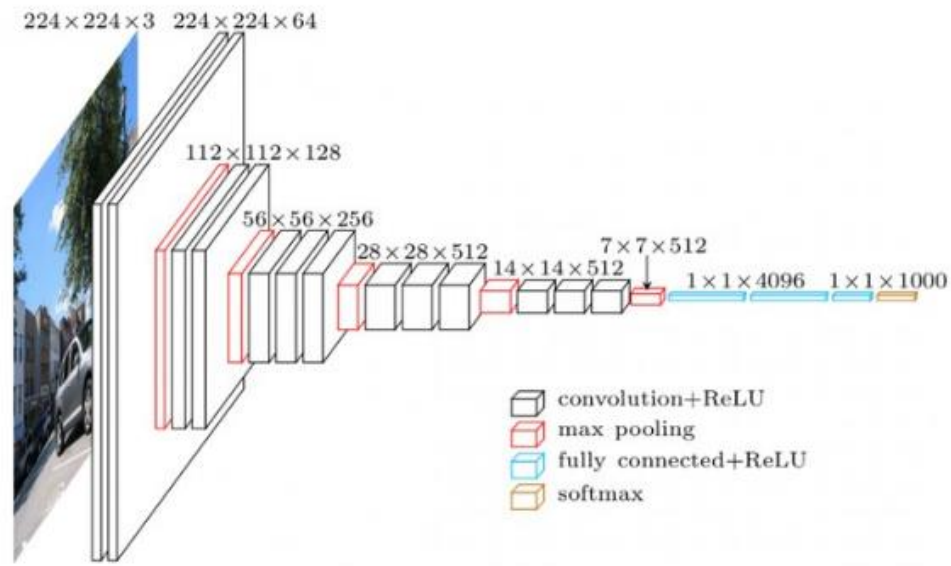


Figure 11: VGG-19 architecture

On the one hand, as said in [11] the first convolutional layers of the network take much attention to small characteristics of the image extracting its information, contours for example. On the other hand, big features as shapes and objects are extracted on the last layers of the network. Whilst the pose of a person is related to big features, as shapes, extracted on the last layers, the first layers can contain information about the edges of the person, the more information the better. That's the reason because of extracting different features of each layer.

3.1.2. Pose extraction

The model will has as input an image of a person and it will have as output a set of 18 different keypoints which represent different parts of the body. Those keypoints are given in a format in which each 18 different keypoint index is related the position of those given by its coordinates x and y.

The global structure of the system is displayed in *Figure 12* and the main key of the Pose Estimation is using the feature maps extracted by the VGG-19 to produce a set of confidence maps (CMs) and part affinity fields (PAFs).

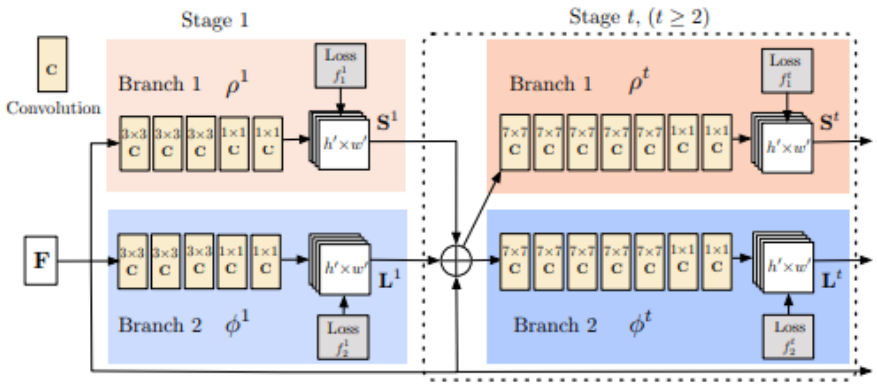


Figure 12: System overview, extracted from [9]

As shown on the image the system is divided into two branches, both taking the feature map as input. And while one branch produces the CMs ($\boldsymbol{S}$) the other branch produces a set PAFs ($\boldsymbol{L}$) . In each stage, the features of the original image and the predictions from the previous ones are concatenated to produce the new predictions.

The confidence maps and part affinity fields are calculated as,

$$\boldsymbol{S^t = \rho^t\left(F, S^{t-1}, L^{t-1}\right), \quad t \geq 2} \qquad \textbf{(3.1.2.1)}$$

$$\boldsymbol{L^t = \phi^t\left(F, S^{t-1}, L^{t-1}\right), \quad t \geq 2} \qquad \textbf{(3.1.2.2)}$$

being $\boldsymbol{\rho^t}$ and $\boldsymbol{\phi^t}$ are the CNNs for inference on the previous stage **t**.

CONFIDENCE MAPS

The confidence map is a 2D representation of how sure the system is of a particular body part being at each pixel location. The CMs for each body part should only have one peak on it if the corresponding part is visible on the image.

The trained confidence maps were calculated by using the groundtruth position of each body part. The value at the location **p** of $\boldsymbol{S_j}$ being **j** the different body parts of a person is,

$$\boldsymbol{S_j^*(p) = \exp\left(-\frac{||p - x_j||_2^2}{\sigma^2}\right)} \qquad \textbf{(3.1.2.3)}$$

being **p** the location, $\boldsymbol{x_j}$ the groundtruth position of the **j** body part and $\boldsymbol{\sigma}$ controls the spread of the gaussian peak.

The final CM of the person is calculated as an aggregation of all the individual confidence maps of the different body parts with a max operator.

$$\boldsymbol{S_j(p) = \max\left(S_j^*(p)\right)} \qquad \textbf{(3.1.2.4)}$$

It is used the max function in order to distinguish possible close peaks, that wouldn't happen if instead of taking the maximum of the CMs we take the average of them.

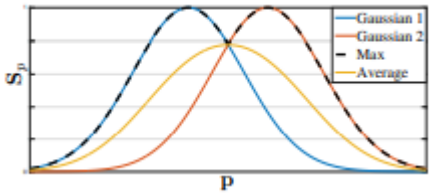


Figure 13: Max function of the confidence maps, extracted from [9]

As said previously on the state of the art, this architecture also works with multi-person pose estimation. In that case the confidence maps of each body part will be the similar for each person on the image.

PART AFFINITY FIELDS

The part affinity fields are used for associate different body parts detected and create the human pose estimation we aim to create. PAFs are aimed to preserve location and orientation information, they are a 2D vector field for each pixel belonging to a particular limb. The 2D vector encodes the direction from one part to another from a particular limb, for example, the direction from the elbow to the wrist.

Let's consider $\boldsymbol{x_{j1}}$ and $\boldsymbol{x_{j2}}$ two different groundtruth positions of body parts $\boldsymbol{j_1}$ and $\boldsymbol{j_2}$ that joining them a limb **c** it's created, for example an arm. If a point **p** is on the limb, it will be located on a unit vector $\boldsymbol{L_c^*(p)}$ that points from $\boldsymbol{j_1}$ to $\boldsymbol{j_2}$**.** For all other points the vector will be zero. It is defined the groundtruth PAF vector field in an image point **p** as:

$$\mathbf{L}_c^*(\mathbf{p}) = \begin{cases} \mathbf{v} & \text{if } \mathbf{p} \text{ on limb } c \\ \mathbf{0} & \text{otherwise.} \end{cases} \qquad \textbf{(3.1.2.5)}$$

being $v = \frac{x_{j2} - x_{j1}}{|x_{j2} - x_{j1}|}$ the unit vector that indicates the direction of the limb. The points that will be considered part of the limb are defined as those within a distance threshold from the segment, $0 \leq v(p - x_{j1}) \leq l_c$ and $|v_N(p - x_{j1}) \leq \sigma_l$ being $l_c = |\boldsymbol{x_{j2}} - \boldsymbol{x_{j1}}|$ the limb **c** length, $\boldsymbol{\sigma_l}$ the limb width measured in pixels and $\boldsymbol{v_N}$ a perpendicular vector to $\boldsymbol{v}$.

Finally, the groundtruth PAF averages the affinity fields of the limb.

$$\boldsymbol{L_c^*(p) = \frac{1}{n_c(p)} L_c^*(p)} \qquad \textbf{(3.1.2.6)}$$

where $\boldsymbol{n_c(p)}$ are the non-zero vectors at point **p**. They tested alignment of the PAFs predicted with the candidate limb that would have been created when connecting two different points $\boldsymbol{x_{j2}}$ and $\boldsymbol{x_{j1}}$. For two candidate locations d_{j1} and d_{j2} they sampled the predicted PAF $\boldsymbol{L_c}$ all along the line of the segment to measure the confidence in their association:

$$E = \int_{u=0}^{u=1} \boldsymbol{L_c(p(u))} \frac{d_{j2} - d_{j1}}{||d_{j2} - d_{j1}||_2} \qquad \textbf{(3.1.2.7)}$$

where $\boldsymbol{p(u)}$ interpolates the position of the two body parts d_{j1} and d_{j2}.

$$\boldsymbol{p(u) = (1 - u)} d_{j1} + \boldsymbol{u} \cdot d_{j2}. \qquad \textbf{(3.1.2.8)}$$

LOSS FUNCTION

The overall objective is to predict as better as we can the pose of the person. To achieve this we calculate the loss function as.

$$\boldsymbol{f} = \sum_{t=1}^{T} (\boldsymbol{f_S^t} + \boldsymbol{f_L^t}) \qquad \textbf{(3.1.2.9)}$$

As having two different branches, we will have two different loss functions at the end of each stage to guide the network to iteratively predict the PAFs and CMs, those are $\boldsymbol{f_L^t}$ and $\boldsymbol{f_S^t}$ respectively for each stage **t**, being **C** the different joins, **J** the ground truth positions of the body parts and **p** the different points on the limb. Both losses are calculated between the estimated predictions and the groundtruth maps and fields. The loss functions of both confidence maps and part affinity vector fields are:

$$\boldsymbol{f_S^t} = \sum_{j=1}^{J} \sum_{p} W(p) \cdot ||\boldsymbol{S_j^t(p)} - \boldsymbol{S_j^*(p)}||_2 \qquad \textbf{(3.1.2.10)}$$

$$\boldsymbol{f_L^t} = \sum_{c=1}^{C} \sum_{p} W(p) \cdot ||\boldsymbol{L_c^t(p)} - \boldsymbol{L_c^*(p)}||_2 \qquad \textbf{(3.1.2.11)}$$

being $\boldsymbol{S_j^*(p)}$ and $\boldsymbol{L_c^*(p)}$ the groundtruth for CMs and PAFs. Also, a binary mask $\boldsymbol{W(p)}$, which value is 0 when the annotation is missing on $\boldsymbol{p}$ pixel, is used to avoid penalizing the true positive predictions when training.

The PAFs and CMs will have detected the different keypoints that appear on the image, then we will return the coordinates of the keypoints by referring them to the keypoint they belong to.

3.2. Pose adaptation for Pose Transfer

As explained on the previous point we have a set of 18 different keypoints representing the different the different parts of the body and whit a format in which it gives us the coordinates of each keypoint.

The output previously explained had not the format that the Pose Transform system needed to process the pose information. With the pose of the input image being unreadable by the model would be impossible to transpose the pose from the image to the one of the target image.

To adapt the output list of the Pose Estimation system to the needs of the Pose Transfer we pre-processed the data using different methods.

3.2.1. Pose adaptation method

When we have passed the input image through the Pose Estimation system, and it has given us the 18 different coordinates (x, y) of each one we have to pre-process them as we will explain before passing the pose information to the Pose Transfer model.

We firstly need to separate the coordinates of the different keypoints into two different vectors, one for the x coordinates and another one for the y coordinates of the keypoints. This objective is easily accomplished thanks to the structure of the format in which the Pose Estimator has given us the location of the keypoints.

Keys	Values
1	X coordinate, Y coordinate
2	X coordinate, Y coordinate
3	X coordinate, Y coordinate
…	…
17	X coordinate, Y coordinate

Table 5: Pose Estimation output, keypoints location format

Once the two vectors have been created, we need to modify the keypoints that haven't been detected or that don't appear on the image. We need to modify their value from 0 to -1. This is how the Pose Transfer system will understand that the keypoints does not exist on the image, if the value had continued being 0, it would have thought that the keypoint was at the top left corner of the image.

Afterward, we will generate an array of dimension 256 x 176 x 18. Each matrix of 2 dimensions will contain the information of each keypoints as a mask, all values shall be 0 minus the points were the keypoint has been detected which value will be 1.

Once all this process has been done, the estimated pose by the Pose Estimation system fits the demands of the Pose Transfer one and will be able to understand the pose of the person on the input image.

3.3. Pose Transfer

Unlike in the previous point where we have not trained any neural network, we have just made inference on the Pose Estimation model, we will train a Pose Transfer Network to obtain better results and focus more on the objective of the project by processing the database. This pre-process will be explained on point 4.1.

3.3.1. Model Used

The model used doesn't transfer the pose of the input image to the one of the target image directly by using only one image. It uses different block to create intermediate images in order to obtain better results.

The block by which the generator creates the image from the input image and the target image is a Pose-Attentional Transfer Network (PATN) which is made by different Pose-Attentional Transfer Blocks (PATBs) [10] that will generate those different intermediate images.

The main objective of this model is to move the joints and body parts indicated by the input image to the locations indicated by the target pose.

3.3.2. Generator

The generator consists basically of the PATN block which has as input the conditioning image or input image $\boldsymbol{P_c}$ and both poses, the one from the target pose image $\boldsymbol{S_t}$ and the one corresponding to the conditioning image $\boldsymbol{S_c}$. The generator is trained to transfer the pose $\boldsymbol{S_c}$ of the input image $\boldsymbol{P_c}$ to the pose $\boldsymbol{S_t}$, this will generate a final image $\boldsymbol{P_g}$ being the output of the PATN. $\boldsymbol{S_c}$ and $\boldsymbol{S_t}$ can be also called pose heat maps which are arrays of dimension 256 x 176 x 18 containing 0s and 1s.

Before entering to the PATN, the conditioning image is down-sampled using 2 convolutional layers creating $\boldsymbol{F_0^P}$, and both pose heat maps $\boldsymbol{S_c}$ and $\boldsymbol{S_t}$ are mixed in one array preserving the information and dependencies before also being down-sampled by 2 convolutional layers, this mix of both pose heat maps will be called $\boldsymbol{F_0^S}$.

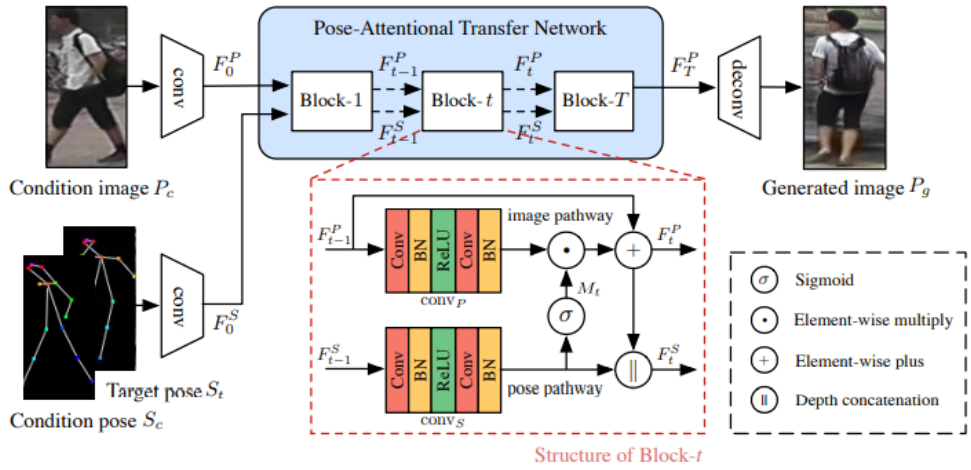


Figure 14: Pose Transfer Generator architecture, extracted from [10]

Pose-Attentional Transfer Network

As said before, the PATN block contains several PATBs that will progressively update the pose and the image to finally generate $\boldsymbol{P_g}$ by taking the final image and pose of the last PATB.

Pose-Attentional Transfer Blocks

All PATBs have the same architecture and structure. As input, they have the outputs from the previous PATB block, in the case of the first one, it will have the down-sampled conditioning image $\boldsymbol{F_0^P}$ and the mixture of pose heat maps $\boldsymbol{F_0^S}$.

Considering one PATB **t,** the inputs will firstly pass through 2 CNNs with a normalization layer and a ReLU function (the output will be directly the input if positive, otherwise the output will be 0) between them.. This process will be expressed as $\mathrm{conv}_s(\boldsymbol{F_{t-1}^S})$ for the branch having as input the pose heat map and $\mathrm{conv}_p(\boldsymbol{F_{t-1}^P})$ for the branch having as input the image information, being $\boldsymbol{F_{t-1}^P}$ and $\boldsymbol{F_{t-1}^S}$ the outputs of the previous PATB.

As output, they will have similar outputs to the input ones, with the same formats, but updated thanks to the indications given by the pose. The indications are given by the attentional masks (AMs) $\boldsymbol{M_t}$, which has values between 0 and 1 depending on its importance. $\boldsymbol{M_t}$ is calculated by using $\mathrm{conv}_s(\boldsymbol{F_{t-1}^S})$, which contains information form the conditioning and target pose thanks to being dependant on $\boldsymbol{F_0^S}$, and will finally be mapped from 0 to 1 using an element-wise sigmoid function. Its expression is:

$$\boldsymbol{M_t = \sigma \cdot conv_s(F^S_{t-1})} \quad \textbf{(3.3.2.1)}$$

The image code will be updated thanks to $\boldsymbol{M_t}$ as said before, by using the output image code from the previous block:

$$\boldsymbol{F^P_T = M_t \cdot (conv_p(F^P_{t-1})) + F^P_{t-1}} \quad \textbf{(3.3.2.2)}$$

This is done because when multiplying the transformed image codes by $\boldsymbol{M_t}$, which indicates the importance of each element, will make $\boldsymbol{F^P_T}$ be 0 or conserve the elements of $conv_p(\boldsymbol{F^P_{t-1}})$, the result is added to $\boldsymbol{F^P_{t-1}}$ being this las one a residual connection that will help us to avoid exploding gradients o vanishing gradients, this phenomena happens when gradients tend to the infinity. Residual connections act as reminder of the previous information.

Once the model has updated the image it also has to update the pose to match each other and synchronize the updates. In order to adapt the new pose, it is concatenated the new pose code with the new image code maps along the depth axis.

$$\boldsymbol{F^S_T = conv_s(F^S_{t-1})||F^P_T} \quad \textbf{(3.3.2.3)}$$

Once the information has passed along all 9 PATBs the generated image will finally be decoded by taking $\boldsymbol{F^P_9}$ and will pass through 2 deconvolutional layers by finally obtaining $\boldsymbol{P_g}$. The pose information $\boldsymbol{F^S_9}$ will be discarded.

3.3.3. Discriminator

Due to the fact of having two different elements that update the information, the image and the pose, each one treated by one different branch, we will use two different discriminators, one for each element. The discriminator $\boldsymbol{D_A}$ will be the one judging the appearance while $\boldsymbol{D_S}$ will judge the pose of the generated image $\boldsymbol{P_g}$. Both discriminators will have similar architectures where the generated image will be concatenated with $\boldsymbol{P_c}$ or with $\boldsymbol{S_t}$, the appearance consistency and the shape consistency respectively. This concatenation will be done along the depth axis and before passing through a CNN. We will have two different scores, $\boldsymbol{R^A}$ for the appearance consistency and $\boldsymbol{R^S}$ for the shape consistency. Those scores are the probabilities calculated by the last layer of the CNN using a Softmax function. Both, $\boldsymbol{R^A}$ and $\boldsymbol{R^S}$, are used to calculate the final score $\boldsymbol{R = \frac{R^A + R^S}{2}}$. The discriminator finally has 3 residual blocks after two down-sampling convolutional networks because if this would have low capacity it wouldn't be able to differentiate between real and fake images.

Loss function

As we have two different scores, one $\boldsymbol{R^A}$ judging the appearance consistency and $\boldsymbol{R^S}$ the pose consistency will make the loss function depend on both parameters calculated by the discriminator. The final loss function is calculated as:

$$\mathcal{L}_{final} = \arg\left(\min_{G}\left(\max_{D}(\alpha \cdot \mathcal{L}_{GAN}\right) + \mathcal{L}_{combL1} \qquad \textbf{(3.3.3.1)}$$

being $\mathcal{L}_{GAN}$ the adversarial loss depending on the discriminator and the generator, therefore, from $\boldsymbol{D_A}$ and $\boldsymbol{D_s}$, and $\mathcal{L}_{combL1}$ the combined $\boldsymbol{L_1}$ loss while the parameter $\boldsymbol{\alpha}$ is the contribution of the loss predicted from the GAN. As said, $\mathcal{L}_{GAN}$, depends on $\boldsymbol{D_A}$ and on $\boldsymbol{D_s}$ as follows:

$$\begin{aligned}\mathcal{L}_{GAN} = \mathbb{E}_{S_t \in \mathcal{P}_s(P_c,P_t)\in\mathcal{P}}[\log(D_A(P_c,P_t)\cdot\ D_S(S_t,P_t))] + \\ + \mathbb{E}_{S_t \in \mathcal{P}_s(P_c,P_t)\in\widehat{\mathcal{P}}}\{\log[(1 - D_A(P_c,P_t))\cdot(1 - D_S(S_t,P_t)\}\end{aligned} \qquad \textbf{(3.3.3.2)}$$

denotating $\widehat{\mathcal{P}}, \mathcal{P}\ and\ \mathcal{P}_s$ the distribution of the fake person images, real person images and person poses respectively. It has a typical loss function structure of $\log(\mathcal{D}(I, I_C)) + \log(1 - \mathcal{D}(I, I_C))$.

On the other side we've got the combined $\boldsymbol{L_1}$ loss calculated as:

$$\mathcal{L}_{combL1} = \lambda_1\mathcal{L}_{L1} + \lambda_2\mathcal{L}_{perL1} \qquad \textbf{(3.3.3.3)}$$

where $\mathcal{L}_{L1} = ||P_g - P_t||_1$ is the pixel-wise loss computed between the generated image $\boldsymbol{P_g}$ and the target image $\boldsymbol{P_t}$. The $\mathcal{L}_{perL1}$ factor is used to make de generated image more realistic by reducing the pose distortion by integrating a perceptual $\boldsymbol{L_1}$ loss.

$$\mathcal{L}_{perL1} = \frac{1}{W_\rho H_\rho C_\rho}\sum_{x=1}^{W_\rho}\sum_{y=1}^{H_\rho}\sum_{z=1}^{C_\rho}||\phi_\rho(P_g)_{x,y,z} - \phi_\rho(P_t)_{x,y,z}||_1 \qquad \textbf{(3.3.3.4)}$$

being ϕ_ρ the outputs of the ρ layer of a pretrained VGG-19. W_ρ, H_ρ, C_ρ are the spatial dimensions of ϕ_ρ, width, height and depth.

The generated image will be more realistic because the features extracted from a pretrained a VGG (already trained to detect shapes and details of images) of the real image and compare them with the features of the image generated by the model, we can know how the 2 images (real and generated) are different between them. Lowest layers of the

VGG will compare stylish and details while higher layers will compare shapes. So if we take the outputs of the higher layers and we make an optimization process the generated and real image will be more similar and distortion will be reduced.

3.3.4. Training

When training GANs both generator and discriminator should be trained at the same time, in this case, we are training the generator and both discriminators $\boldsymbol{D_A}$ and $\boldsymbol{D_S}$ at the same time.

The training is done by creating a csv file in which there are pairs of images in which they appear the same person wearing the same cloth but in two different poses, the creation of the csv file will be explained on the Dataset chapter. The generator will take the pose conditioning image $\boldsymbol{P_c}$, the conditioning pose $\boldsymbol{S_c}$ and the target pose $\boldsymbol{S_t}$ as explained and it will use those inputs to generate the image $\boldsymbol{P_g}$ as explained before. The conditioning pose will pass through the image pathway while the concatenations of the conditioning and target pose will pass through the pose pathway.

For the discriminators, images $\boldsymbol{P_c}$, $\boldsymbol{P_t}$ and $\boldsymbol{P_g}$ are inputs of $\boldsymbol{D_A}$ using them to obtain the best appearance while $\boldsymbol{S_t}$, $\boldsymbol{P_t}$ and $\boldsymbol{P_g}$ are used for training the discriminator $\boldsymbol{D_S}$ judging the shape consistency.

4. Results

In this chapter we are going to firstly see the dataset used for training and testing the Pose Transfer model and its pre-process. Afterwards we will see the different results obtained with the pretrained Pose Transfer system with the objective of detecting the upper keypoints of the person on image, as we do not reallly care about the lower of the human body because as said before, the model is focused for images on images in which the person is shown kneeling upwards.

Subsequently we will show the improves when using the Pose Transfer system by training the model changing different parameters from the ones described on the respectively paper of the model [10]. The results will be shown by using test images to evaluate them subjectively and with other different metrics.

Finally, in order to show the results and that people could interact with the project we decided to implement a demo using Streamlit where the results are more flashy and much easy understandable how the whole system works.

4.1. Dataset

The most relevant part for training a neural network is the database. If it doesn't have enough images to learn from or without good quality, the training won't be able to transfer the pose from the input image to the pose of the target image and neither reproduce its visual quality or appearance.

The database I used to train the Pose Transfer system was the public DeepFashion dataset [15], this is one of the main datasets used for training Pose Transfer models together with the Market-1501. I decided to use the DeepFashion dataset because of the use of case, being my project more focused on fashion I thought it would be more appropriate to use images with white background since the output image of the try-on system being developed by StageInHome has white background.

The DeepFashion dataset contains images from both gender, women and men, but there are much more from women.

Raw DeepFashion database	
WOMEN	52019 images
MEN	8798 images

Table 6: Raw DeepFashion database

The DeepFashion dataset is organized in different subfolders as shown on the next scheme being TRAIN / TEST the first folder:

TRAIN / TEST	GENDER (Women/Men)	Type of cloth	ID number	Images: XX_Y_pose.jpg

Figure 15: Database organization

Due to the fact that the try-on being developed by StageInHome is focused on the upper clothes, they only accept images in which the person's lowest part that appears are the knees. To adapt to their model, I filtered the database by the type of clothes, eliminating the images in which only the legs of the model appeared.

Once I only had the folders of the upper parts I filtered them by the names of the images, the image names format is "XX_Y_pose.jpg" being XX a piece of cloth, Y a number and pose the pose in which the model appears on the image with the possibilities of: full, side, front, additional, back or flat.

Figure 16: Database image front type

02_1_front.jpg

Figure 17: Database image side type

02_2_side.jpg

Figure 18: Database image back type

02_3_back.jpg

Figure 19: Database image full type

02_4_full.jpg

The generation of the images with the target pose will be more accurate if the dataset is the most stable as possible, with the fewer variations on the image. Taking into account the latter factor on the previous point about the images accepted by the try-on system of StageInHome I decided to remove all the images of the dataset where the models didn't have a front nor a side pose. Using as example the images from above, images with names as 02_3_back.jpg and 02_4_full.jpg had been removed.

After applying that filter, I decided to augment the database by flipping all the images I had left. Flipping the images not only made my data base greater but also would help to adjust the number of images facing each side in the case there were more from one side than the other.

After all the database pre-processing, the images remaining really fit the aim of the project being more aimed to the images accepted by the StageInHome virtual try-on and by reducing the variability of the images at the same time we have augmented the database by flipping all the images useful for the training.

I finally divided the database on two different databases, one for training and another one for testing the model. From the 45422 images I had I used 39436 images for training and 5986 images for test. Once the databases were separated, I wrote two different csv containing pairs of images of the same person wearing the same cloth but in two different poses. Those are the datasets that I later used for training and for testing the trained model.

4.2. Experiments

In this section we will report all the results obtained with the Pose Estimation system and the Pose Transfer model having this last one as input the output of the former.

4.2.1. Pose Estimation

As said on the methodology, I only made inference on the Pose Estimation model. I change different parts of the system and the program to fit what I wanted from it, extract an image of the keypoints and a dictionary in which the keypoints were represented.

The system needs as input an image of a person and will return as output the image of the skeleton and the dictionary of keypoints.

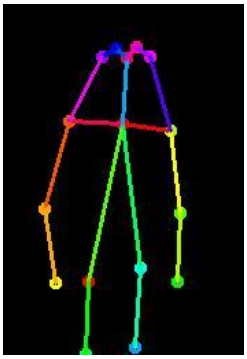
Input image	Output image	Output dict
		{0: (90, 39), 1: (87, 87), 2: (48, 85), 3: (30, 149), 4: (38, 203), 5: (122, 92), 6: (129, 152), 7: (127, 202), 8: (62, 202), 9: (60, 255), 11: (100, 192), 12: (96, 250), 14: (82, 32), 15: (97, 31), 16: (72, 38), 17: (107, 38)}

Figure 20: Pose Estimation example 1 a,b,c

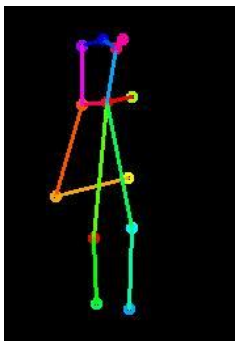
		{0: (85, 32), 1: (78, 73), 2: (59, 75), 3: (39, 145), 4: (94, 131), 5: (97, 69), 8: (68, 177), 9: (70, 227), 11: (97, 169), 12: (95, 231), 14: (75, 25), 15: (90, 25), 16: (59, 30)}

Figure 21: Pose Estimation example 2 a,b,c

I adapted the model in order to see if the person on the input image has the head turned around if the person is on his back by joining the keypoints of the left shoulder with the ones of the left ear and the same on the right side. *Figure 22*

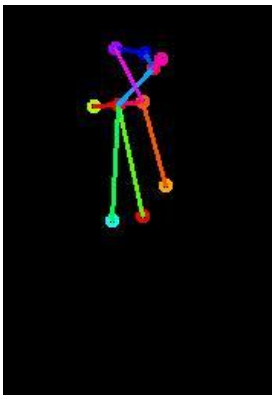
Input image	Output image	Output dict
		{0: (98, 41), 1: (75, 65), 2: (91, 63), 3: (106, 118), 5: (59, 66), 8: (91, 138), 11: (71, 141), 14: (92, 31), 15: (103, 35), 16: (73, 28)}

Figure 22: Pose Estimation example 3 a,b,c

Although the system works properly for many input images it has some difficulties when the person on the image has strange poses or very difficult ones. It will obviously not detect the joints and human body parts that don't appear on the image as happens with the left arm of the image. *Figure 21*

Most of these problems are solved thanks to the classification of the dataset by keeping only the images in which people appear in a front or side pose which are the most realistic ones as seen on *Figure 20*.

Although we didn't have the groundtruth of the images we did a little test by passing trough the system 50 different images and on 42 of them the model detected all the keypoints that appeared on it.

4.2.2. Pose Transfer

In this chapter we will show the different results obtained from the Pose Transfer system and we will see how they have evolved during all the different types of training we have done.

I firstly ran inference on the base model to see what results it gave and how similar they were in terms of quality to the ones shown on the paper [10]. In order to obtain an output you have to pass the system two different images as input. The first one will contain the person you want to change the pose of and the other one will be the image containing the target pose, the pose you want to have the first image.

The main parameters of the training are adjusted based on the Adam optimizer [12]. The learning rate is fixed during the whole training with a value of 0.0002 and the values for the loss functions $\mathcal{L}_{final}$ and $\mathcal{L}_{combL1}$ are set to $\lambda_1 = 1$, $\lambda_2 = 1$ and $\alpha = 5$.

Figure 23: Base model example 1

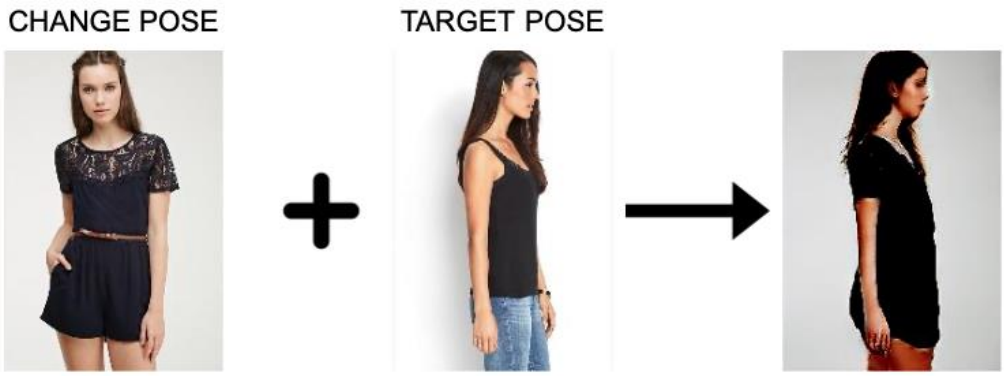


Figure 24: Base model example 2

We can evaluate the outputs of the model by the appearance and the pose as the model does. On the one hand, in reference to the appearance and quality of the image, as shown on *Figure 23* and *Figure 24* the results obtained weren't as good as expected, the person on the output image should be at least very similar to the ones on the "CHANGE POSE" image and they aren't. On the other hand the pose of the output image is very similar to the pose of the person on the "TARGET POSE" image.

IMPROVEMENTS

As the results of the base model weren't as expected I decided to train the model with the same parameters as the ones used on the base model.

Training of 70 epochs

The first training I did was on the server hosted by StageInHome and I only could ran the code during a certain period of time due to the fact that I took up all the GPU memory. During this time, I was able to train for about 10 hours, that's the equivalent of 70 epochs with a batch of 8 images. The GPU used is RTX2090Ti.

Figure 25: 70 epochs training, example 1

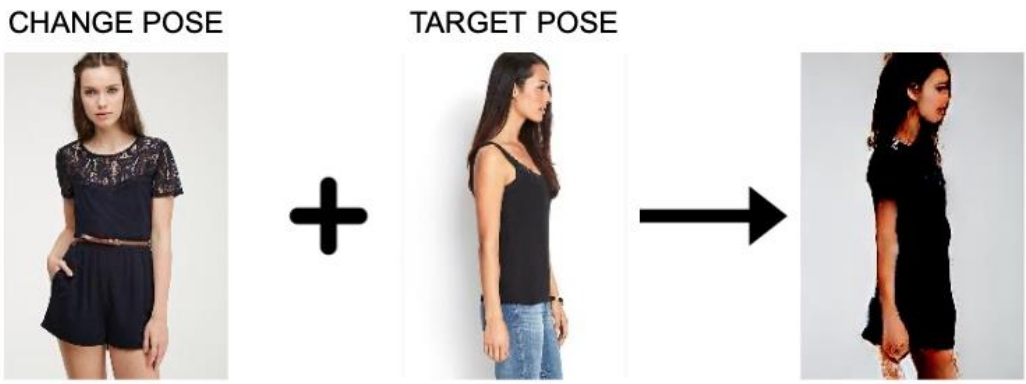


Figure 26: 70 epochs training, example 2

Figure 25 and *Figure 26* show the results obtained when the model used was trained during 70 epochs, comparing to the outputs to the ones obtained when using the base model, they are much worse in appearance and although the pose is similar to the pose on the image of "TARGET POSE" but not as good as the one obtained from the base model.

Train of 500 epochs

Trying to improve the results obtained by the training of 70 epochs I decided to let the model take much more time training, the model was trained for 3 days, and which is the equivalent to 500 epochs, with the same parameters as the base model and the model of 70 epochs.

Figure 27: 500 epochs training, example 1

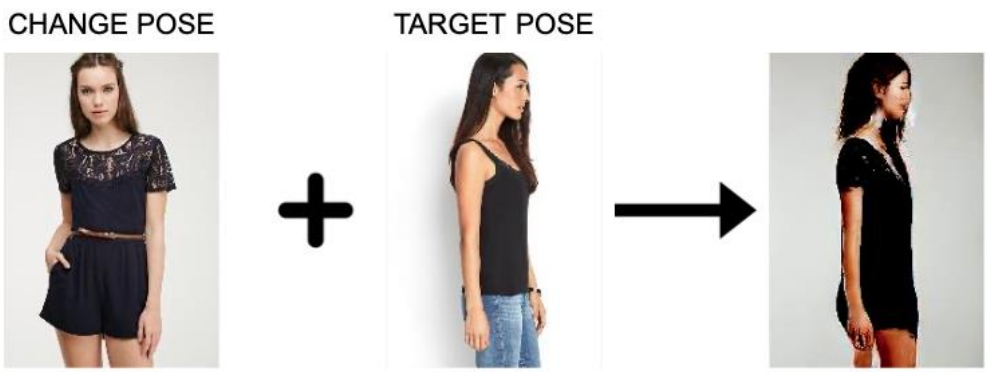


Figure 28: 500 epochs training, example 2

We can clearly see, and specially on *Figure 28*, that the results are much more realistic than the ones obtained previously on *Figure 25* and *Figure 26*. It is true that the appearance from *Figure 28* isn't as good as the one on *Figure 27* despite being outputs from the same model, but it is better than the one obtained from the model trained with 70 epochs. We can say that the model has improved its quality.
Referring to the pose, it has not improved much from the previous trained model despite the fact of training more the model.

Comparison between base model and model trained during 500 epochs

Once I had done several trainings, I wanted to compare them with the base model and see if I was on the right track to obtain better results and investigate what elements or parameters could I change in order to obtain a much better model, generating much more realistic persons.

Figure 29: Base model and 500 epochs training comparison, example 1

Figure 30: Base model and 500 epochs training comparison, example 2

Figure 31: Base model and 500 epochs training comparison, example 3

In our opinion the models are at the same level because on some results one is better than the other. For example, on *Figure 29* we think their result is better because there doesn't appear the space between the arm and the trunk of the women while on *Figure 30* the image generated by our model has more details on the eyes and nose.

With these comparisons we concluded that the pose wasn't such a problem as the appearance on the generated image. The differences between the pose on the generated image are very similar between the base model, the one trained during 70 epochs and the model trained during 500 epochs, all generated images have a very similar pose to the person on the "TARGET IMAGE".

We decided to try to improve the appearance of the generated images by modifying the loss function of the Pose Transfer system. As we wanted to generate more realistic images, we increased the weight of the appearance by giving more importance to $\boldsymbol{D_A}$, the discriminator judging the appearance of the generated image, we increased its weight by 2 while we didn't modify the score of the discriminator judging the pose accuracy.

We had two different paths, take the model trained during 500 epochs and fine-tune the model with the loss modified, redirect the training towards were we wanted, or train the model with the modified loss from 0. As we thought the model had learned much on the training to fine-tune the appearance, we finally decided to train the model from scratch.

Training of 500 epochs with modified loss

We decided to train the model from scratch with having the loss modified by doubling the weight of $\boldsymbol{D_A}$. We also wanted to augment the batch and that wasn't able on the StageInHome server due to lack of memory, so I had to start using instances on Amazon Web Services (AWS) with higher memory than the server. The instance we took was g4dn.xlarge which led as train with a batch of 14 images. Although on the AWS we could train with more batch, it took much more time than training on the StageInHome server.

Figure 32: Model with the loss modified 1

Figure 33: Model with the loss modified 2

The results have much better appearance than the ones obtained when using the model without having modified the loss function. Despite the better quality, we thought the results could be much better if we if we continued giving more importance to the appearance as the pose of the person in the generated image is still very good.

Train of 700 epochs with modified loss

We decided to give more importance to the appearance by giving it a weight of 3 instead of 2 as we had done on the previous training. We continued using a batch of 14 images, so the training had to be also done on the AWS instance. This time instead of only training during 500 epochs we decided to train during 700 which took 8 days.

Figure 34: Last trained model example

The results obtained were much more satisfying than all the previous trainings, so we decided to do much more tests with this model, not only testing with women but also testing with men and mixtures of both genders.

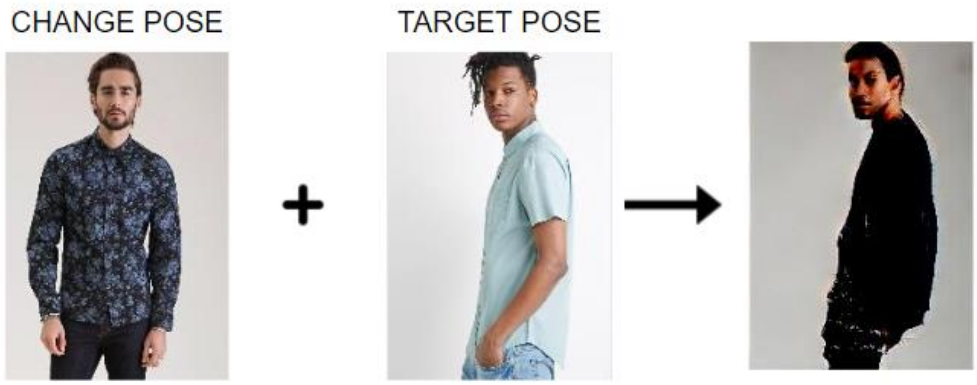


Figure 35: Last train man example

Figure 36: Last train, mixing genders example 1

Figure 37: Last train, mixing genders example 2

As shown on *Figure 35* the Pose Transfer system when using man is much worse than when using women, that's because the database has much more images from women than from men. In reference to the images *Figure 36* the result is also very bad comparing to the once using only women while on *Figure 37* the result is much realistic but not as good as on *Figure 34*.

Once the model was trained, we observed the database to see what we could improve but we saw that the database wasn't what we intended to be when doing the pre-processing explained on the Dataset chapter. Lots of images were similar to *Figure 18* and *Figure 19.*

METRICS

Until now we have been evaluating the results qualitatively, in order to give a more solid base to the results we have used the metrics to give solidity to the results obtained.. We firstly compared the structural similarity (SSIM) [14] obtained when comparing the generated images of both models with the groundtruth images, this metric will help us evaluate the pose of the images.

To evaluate the appearance of the generated images we firstly did a MOS in which we asked people to rate from 1-5 how similar the person in the generated image was to the person we wanted to change the pose by adapting to the posture of the second image. We also calculated the Fréchet inception distance (FID) [13] using the same method as the one used to calculate the SSIM.

Metric	Base model	Trained model	Best result
SSIM	0,558	0,554	1
MOS	1,99	3,27	5
FID	81,36	65,033	0

Table 7: Metrics results

One the one hand, on the SSIM metric, we see that they have a very similar rating, which means that they are very similar I regarding pose. Both, the trained model and the base model are at the half of the best results possible.

On the other hand, we can see that both on the FID and on the MOS our trained model is much better meaning that we have improved the base model regarding the appearance of the generated image. In both cases our model is much better but far from perfect, especially in terms of FID.

4.3. User Interface

In order to show the project in a very visual way and that people could have interaction we decided to do a little demo on Streamlit [17]. This is an open-source app framework which allows you to load deep learning and machine learning models as the ones used in this project.

In the demo the user will choose from three images the person he wants to apply the Pose Transfer on. Once this image is chosen, he will choose from four different poses which want he wants to transfer the pose to. Once both images have been chosen the user will be able to generate an image and see how the first image chosen looks with the pose of the person of the second image.

People can easily play with the image selection being able to generate as many images as they want but being restricted to the combinations available.

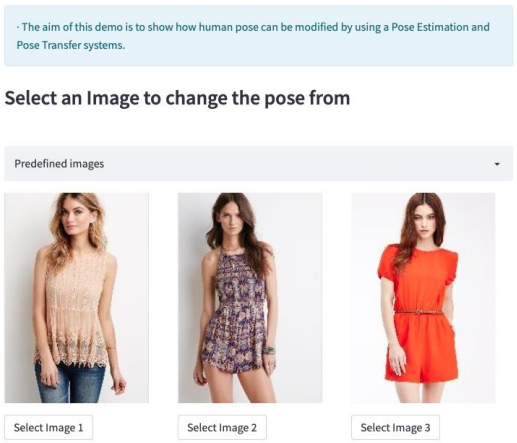


Figure 38: Demo first image selection

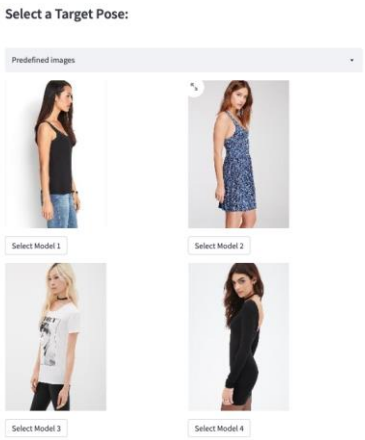


Figure 39: Demo, second image selection

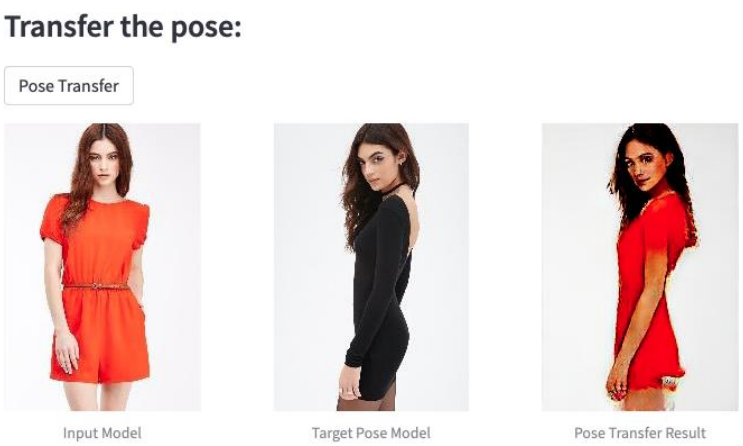


Figure 40: Demo show generated image

5. Budget

The approximate costs of this work have been calculated by considering the services used, the salary and also the time in which the different models have been training and the server in which we have worked on. The software has not been considered as we have used tools that are free of charge.

Services

A laptop is needed in order to develop the software, its approximate cost is of about $\frac{900 \cdot 0{,}9€}{5} = 162€$ per year being the depreciation of the laptop a 10%, considering it hasn't been used during all the year, only 1/3 of it, the total cost is of about $162 \cdot \frac{1}{3} = 54€$ Furthermore, a server has been needed $\frac{3500 \cdot 0{,}85€}{5} = 595€$,being its depreciation 15%, that has also only been used the same amount of time, $\frac{595€}{3} = 198{,}33€$

Trainings have been mostly done on Amazon Web Service (AWS) on a specific instance type, the g4dn.xlarge which has a price of 0,526$ per hour, the trainings have taken in total 240 hours which make a total cost of 126,24$, this is a total of 117,29€.

Description	Cost	Useful life (years)	Total cost (€)
Computer	900€	5	54€
Server	3500€	5	198,33€
AWS instance	0,526$		117,29€.

Table 8: Services costs

Wage

It has been assumed that a junior engineer's salary is 10€/hour and that he has been working from Monday to Friday 6 hours a day, from March 28th to June 17th, 13 weeks. In addition, it is taken the social security costs that the company must pay, 33% is considered.

Description	€/hour	Time	Social security costs	Total cost (€)
Wage	10€	390 hours	1287€	5187€

Table 9: Wage cost

Other costs

A part of the services and the wage, there have also been light costs and travel expenses. Taking into account that as average one person pays 73€ of electricity each two months, the electricity cost will be $73 + 73/2 = 109{,}5$€.

On the other hand, the total travel expenses have been of 105,20€.

Description	Cost
Electricity	109,5€
Travel expenses	105,20€

Table 10: Other costs

The final cost of the project is:

Description	Cost
Services	369,62€
Wage	5187€
Other costs	214,7€
Total	5771,32€

Table 11: Final budget

6. Conclusions and future development:

As we mentioned in the introduction, Deep Learning and Machine Learning are new revolutionary technologies with a lot of potential, but they are not used in human daily life when they can provide lots of commodities. One of the new applications of these technologies are virtual try-on systems in which people can see how some clothes fit them by only using an image. Traditionally people will send an image of themselves, and the system will change the cloth they are wearing, that will make them only see the result of the try-on from just one perspective, that of the image they have used.

Our research wanted to improve the virtual try-on experience. In particular, the main objective is to implement an algorithm to transfer the pose of one person to another using Machine Learning techniques and GANs.
The Pose Transfer method proposed by Zhen Zhu [10] would provide a point of departure for our objective, by adding the human pose extraction and followed by several modifications on the system parameters which have provided much better results on the appearance of the generated images. This has been achieved by pre-processing the database and reducing as much as possible the variance in dependence on the project implementation.

In reference to the system implemented on this project, we can assure that they have reached the five main objectives established at the beginning. We have created a system that will estimate the pose of a person which will be used to reach the pose of another person by using a Pose Transfer. The Pose Transfer system has been trained with pre-processed images that helped us reach better results than the base model.

Finally, we can believe that the results could be considerably improved by cleaning more exhaustively the dataset or by implementing a different manner of calculating the loss function in order to obtain better results in terms of the quality of the generated images. The part that can be improved the most is especially the face and the parts of the head like the hair, for that an implementation of face generation could be done, since there are already a lot of facts and it would only be necessary to make inference.

Bibliography:

[1] The Complete Beginner's Guide to Deep Learning: Artificial Neural Networks". (n.d.). Available: https://towardsdatascience.com/simply-deep-learning-an-effortless-introduction-45591a1c4abb [Accessed: 20 January 2020]

[2] Torres,Jordi (2018) "Deep Learning – Introducción práctica con Keras" . Avaiable: https://torres.ai/deep-learning-inteligencia-artificial-keras/ . [Accessed: 20 January 2020]

[3] Christoph Lassner, Gerard Pons-Moll, and Peter V. Gehler. A generative model of people in clothing. In Proc. ICCV2017

[4] Diederik P. Kingma and Max Welling. Auto-encoding vari-ational bayes. CoRR, abs/1312.6114, 2013.

[5] Bo Zhao, Xiao Wu, Zhi-Qi Cheng, Hao Liu, and Jiashi Feng. Multi-view image generation from a single-view. CoRR,2017.

[6] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuyte laars, and Luc Van Gool. Pose guided person image generation. In Proc. NIPS, 2017.

[7] Chaoyue Song, Jiacheng Wei, Ruibo Li, Fayao Liu and Guosheng Lin. 3D Pose Transfer with Correspondence Learning and Mesh Refinement. 2021

[8] T. Pfister, J. Charles, and A. Zisserman, "Flowing convnets for human pose estimation in videos," in ICCV, 2015.

[9] Zhe Cao, Student Member, IEEE, Gines Hidalgo, Student Member, IEEE, Tomas Simon, Shih-En Wei, and Yaser Sheikh: OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. 2019

[10] Zhen Zhu , Tengteng Huang1 , Baoguang Shi , Miao Yu , Bofei Wang , Xiang Bai Huazhong Univ. Sci. and Tech., Microsoft, Redmond, ZTE Corporation. Progressive Pose Attention Transfer for Person Image Generation. 2019

[11] K. SIMONYAN AND A. ZISSERMAN, Very deep convolutional networks for large-scale image recognition, CoRR, abs/1409.1556 (2014)

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014.

[13] M. Fréchet. Sur la distance de deux lois de probabilité. C. R. Acad. Sci. Paris, 244:689–692, 1957.

[14] Peter Ndajah, SSIM Image Quality Metric for Denoised Images. NIIG 2010.

[15] DeepFashion Database (cuhk.edu.hk)

[16] Ce Zheng, Wenhan Wu, Chen Chen, Toajianna Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, Mubarak Shah. Deep Learning-Based Human Pose Estimation: A Survey. 2022

[17] Streamlit framework 2022

Glossary

A list of all acronyms and the meaning they stand for.

- GANs: Generative Adversarial Networks
- MOS: Mean Opinion Survey
- MCs: Confidence maps
- PAFs: Part affinity fields
- AWS: Amazon Web Service
- PATN: Pose-Attentional Transfer Network
- PATBs: Pose-Attentional Transfer Blocks
- AMs: Attentional masks
- SSIM: Structural similarity
- VR: Virtual reality
- AR: Artificial reality