UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Changing clothing on people images using generative adversarial networks

Author:
Yevhen Pozdniakov

Supervisor:
Orest Kupyn

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Yevhen Pozdniakov, declare that this thesis titled, "Changing clothing on people images using generative adversarial networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Changing clothing on people images using generative adversarial networks

by Yevhen Pozdniakov

Abstract

Generative Adversarial Networks (GANs) in recent years has certainly become one of the biggest trends in the computer vision domain. GANs are used for generating face images and computer game scenes, transferring artwork style, visualizing designs, creating super-resolution images, translating text to images, etc.
We want to present a model to solve an image problem: generate new outfits onto people images. This task seems to be extremely important for offline/online trade and fashion industry.
Changing clothing on people images isn't a trivial task. The generated part of the image should have high quality without blurring. Another problem is generating long sleeves on the images with T-shirts, for example. As a result, well-known models are not suitable for this task.
In the master project, we are going to reproduce the model for clothing changing on people images based on the existing approaches and improve it in order to get better quality of the image.

Contents

List of Figures

List of Tables

List of Abbreviations

GAN	Generative adversarial network
ML	Machine learning
DCGAN	Deep convolutional generative adversarial
CAGAN	Conditional analogy GAN
CP-VTON	Characteristic-preserving virtual try-on network
LWB	Liquid warping block **HMR**
Human mesh recovery	
SMPL	Skinned multi-person linear model
NMR	Neural mesh renderer
GMM	Geometric matching module
TOM	Try-on module

Chapter 1

Introduction

1.1 Background

Machine learning (ML) is one of the hottest topics over the last decade. ML theory based on the idea that algorithms can learn without being directly programmed. Typically ML algorithm trained on the prepared data and after it performs on the unknown data with similar or even better accuracy as people.
Many classical machine learning algorithms have been developed for different tasks. The traditional examples of such tasks are classification and regression problem. However, the rapid growth of computational power and the amount of digital information leads to the development of the more sophisticated algorithms. These algorithms play a key role in self-driving cars, recommendation systems, fraud detection, robotics, etc.
Computer vision is one of the application areas of machine learning algorithms. Computer vision history started in the 1960s. One of the first task in this domain was image recognition; in other words, people ask computers to tell us what they see. Computers "see" world as number of pixels and even first computer vision algorithms require a lot of computation power.
The rapid growth of computer vision began in the last decade. Significant increase in computational power and a large amount of digital information creates possibilities for deep learning usage. Deep learning added a huge boost to the computer vision domain. During the last years, many applications of computer vision techniques have been introduced. These applications are essential parts of our everyday lives. Typical computer vision tasks are image classification, object detection, object segmentation, image style transfer, image colorization, image reconstruction, image super-resolution, image synthesis, etc.
There are many application areas where computer vision algorithms based on deep learning are valuable. The one example that is taking advantage of such algorithms is the automotive industry. With a computer, it's possible to mitigate human error in the auto industry, assisting drivers at the wheel with tools and features that keep them from committing severe mistakes and accidents. For many years automotive sectors have created new sensors and systems to help the driver and reduce the accident rate. Computer vision acts as a combination of all such sensors, analyzes the environment around cars for potential threats, obstacles, and other relevant situations that a driver needs to react while driving [2].
Another application for computer vision is retail. One of the impressive examples is Amazon Go. It is an automated store that has no checkout stations or cashiers. Another possible idea is the client's identification (face recognition) that can be extremely helpful to high fashion. After identification, the client's habits, past purchases, and shipping address could appear on a screen for all sales associates to see. A similar approach is suitable for financial services. For example, new customers

can get a bank account within minutes by uploading a photo of their ID and selfie. But, maybe the most critical application is healthcare. Computers won't completely replace the doctors, but they can be a good assistant in the diagnosis like magnetic resonance diagnostic.
All of the mentioned cases are examples of image recognition problem usage. However, using more sophisticated algorithms creates new opportunities in new industries.
Computer vision algorithms have massive potential in the fashion industry. The fashion industry based on visual information and has a natural connection with computer vision. For example, let's consider the following task: changing clothing on people images. In this case, you can be your own designer with the possibility to ease transform your current outfit on the photo into a completely new one. It can boost online clothing shopping significantly. The other possible application is photo images generation. Suppose that several thousands of items arrive in a new batch. Shooting them on cloth hangers standalone is relatively easy and cheap, but taking photos with professional models is time-consuming and expensive. Leveraging available data and reusing images of human models and products would, therefore, be very useful for a fashion business[18]. This task is also applicable for the offline-commerce as well. For example, a client doesn't want to waste time in a shop for "trying on" clothes. In this case, he/she can check how different clothes' is suitable for him/her on the screen.
GAN is a natural class algorithms for this task. According to [12], GAN consists of a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. The game ends when a Nash equilibrium is achieved.
GANs are one of the most exciting algorithms in machine learning and computer vision in particular. GANs used for generating face images and computer game scenes, transferring artwork style, visualizing designs, creating super-resolution images, translating text to images, etc.

1.2 Purpose

The purpose of the master thesis is to investigate and suggest a method to solve an image problem: change outfits on people images. Quality question is essential here because the changing outfits should be visually realistic and have minimum blurring or other harmful effects of image transformation. The existing algorithms and models should be investigated to select the most appropriate model and improve it as much as possible.
A common problem in such type of tasks is result's evaluation [3]. It's known there is no objective loss function that can be used during training the GAN generator models and no way to evaluate the final results of the trained model. Both quantitative (average log-likelihood, coverage metric, inception score, etc.) and qualitative evaluation (neighbors, rapid scene categorization, rating, and reference judgment, etc.) can be used instead. But the main aim is to receive visually relevant results.
The thesis will investigate if GAN model can change clothing on people images. The existing models will be investigated to choose the model to be implemented and improved.
The following questions addressed here:

1. Can GANs be used to changing clothing on people images?
2. Which model should be used for the best possible visual results?
3. How it's possible to improve the model to get better results?

1.3 Approach and Methodology

In this thesis, we are going to investigate methods and models to solve the following task: change outfits on people images. We are supposed to use GAN model for this purpose. Like other deep learning algorithms, GAN models requires a lot of information for training. We think to use existing datasets, such as iPER, Place2 and DeepFashion datasets. The most essential criteria are the visual representation of the changed images, so just qualitative methods will be used for the GAN model evaluation.

1.4 Motivation

GANs are one of the most active-development algorithms during the last years. Many GAN models have been proposed for the last time. As a result, a lot of factors should be taken into account before GAN models choice.
GANs had a lot of hype last years. But this is logical because the really fascinating things are possible with them. Referring to GANs, Facebook's AI research director Yann LeCun called adversarial training "the most exciting idea in the last ten years in ML. So, the first motivation for me is getting in-depth knowledge about GANs. To reach the aim, at least one GAN model needs to be implemented. The choice of model is a crucial decision. A lot of existing models need to be considered and investigated.
As mentioned above, this model could be beneficial for the fashion industry. However, changing clothing on people images isn't a trivial task. The generated part of the image should have high quality without blurring. Another problem is creating long sleeves on the images with T-shirts, for example. As a result, well-known models are not suitable for this task.

1.5 Time plan

Time plan for completing the master thesis is presented in Table 1.1[t].

TABLE 1.1: Time Table for Completing the Thesis.

	Sep. 2019	Oct. 2019	Nov. 2019	Fri. 2019	Jan. 2020
Literature review	x				
Thesis proposal	x				
Development of tools		x			
Data Collection		x			
Prototype model implementation		x	x		
Model improvement			x	x	
Thesis write-up				x	
Submission of thesis					x

Chapter 2

Theory and related works

2.1 Generative adversarial networks

The first paper about GAN was published in 2014 [12]. In this paper, Goodfellow described the basic idea of GAN model, when the generative model is pitted against an adversary; with a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. In that article, the author explored the special case when the generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. This special case was called as adversarial nets.
In order to learn the generator's distribution p_q over data x there is defined a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(z;\theta_g)$, where G is differentiable function represented by a multilayer perceptron with parameters θ_g. There is also defined a second multilayer perceptron $D(x;\theta_d)$ that outputs a single scalar. D(x) represents the probability that x came from the data rather than p_g. And D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. The discriminator and generator are trained on each training iteration and are competing with each other when trying to achieve the objective according to the following equation:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))]. \tag{2.1}$$

Initially generator synthesizes data that far from the real data distribution. This data seems to be easy classifies as fake. But, in order that discriminator is not "experienced", it will a difficult task for the discriminator to distinguish between the real and fake data. During the training process, both generator and discriminator learns: the first one how to generate better data, the second one - how to distinguish real and fake data. So the training process can be seen as a game between the generator and discriminator. The key aspect is the synchronicity of the process because if either one of them wins, it will stop the process.
The described article proved its results on the MNIST dataset and is the real best-seller in the research world. Many models based on the initial GAN's idea have been suggested since 2014. Let's consider the most important models.
Deep convolutional generative adversarial networks (DCGAN) presented in [34]. This article shows how convolutional layers can be used with GANs. According to the article, the suggested architecture can be considered, with certain architectural constraints, as a strong candidate for unsupervised learning. The model was trained on three datasets: three datasets, Large-scale Scene Understanding, Imagenet-1k and Faces dataset. Some improved techniques for training GANs suggested in [35]. This article presented a variety of new architectural features and training procedures that

can be applied to the GAN models. Such techniques as feature matching, minibatch discrimination, historical averaging, one-sided label smoothing, and virtual batch normalization are used for encouraging convergence.
The BigGAN model is the current state-of-the-art model for ImageNet generation [8]. This article combines a lot of modern technics such as self-attention, spectral normalization and conditional GANs with projection discriminators.
All of the described GANs model used for the image generation. However, when we change clothing on people image, we need to transfer part of one image into another. This kind of task requires conditional GANs.

2.2 Conditional GANs and image-to-image translation

Conditional GANs were introduced in [32]. In an unconditioned generative model, there is no control on modes of the data being generated. However, by conditioning the model on additional information, it is possible to direct the data generation process. Such conditioning could be based on class labels, on some part of data for inpainting or even on data from different modality.
Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y. The variable y could be any auxiliary information, such as class labels or data from other modalities. We can perform the conditioning by feeding y into both the discriminator and generator as the additional input layer. Equation (1) remains the same. Conditional GAN is a base idea for image-to-image or text-to-image transfer.
Conditional GANs have a lot of cool modifications and improvements. One of them is PatchGAN [17]. The idea behind PatchGAN is that instead of having the discriminator evaluating the whole image, the model looks at 70x70 regions of the image to determine if they are real or fake. Such an approach helps to produce sharper image outputs from the generator. This model also shows an interesting U-Net style generator architecture as well as using ResNet-style skip connections in the generator model [6].
Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image [4]. Image-to-image translation based on conditional GANs models. The key difference with the usual GANs is that the input to the network is an image instead of random noise. The generated image based on an input image, thus fact makes the model conditional.
Image-to-image translation refers to the unsupervised class problem because it does not require image pairs for training. It means that that output image will be compared with the image in the same domain. No ground truth (like in supervised learning) is available here. So, all of the advantages of unsupervised learning are applicable here.
Well-known example of image-to-image is CycleGAN [41]. There is presented a model to translate an image from a source domain X to a target domain Y without any paired examples. The key factor in this model is the cycle-consistency loss function. As usual, this loss function is a part of the objective function that the generator learns from. The advantage of this function is that it translates input image to the output domain and then translates it back. The back-translated image compares with the original input image using the MAE. Cycle-consistency loss helps to stabilize GAN training.
The suggested model contains two discriminators and two generators. It guarantees

translation in both directions, which are learned simultaneously. There are many cool applications of CycleGAN such as super-resolution, style transfer, horse to zebra transformation, season translations, etc.
CycleGAN as the typical image-to-image translation, has a serious limitation: it transforms images between just two domains. Usually, two generators are needed for each pair of images. This fact limits scalability and robustness in handling more than two domains. StarGAN is an approach to handle this problem [11]. It can perform image-to-image translations for multiple areas using only a single generator.
StarGAN consists of a discriminator D and a generator G. The critical difference is that G takes as an input both the image and target images. The output images are spatially replicated and concatenated with the input image. D acts in a usual way but to keep up with the generator it also has to learn the specific features of each image domain. For this purpose, the discriminator should know the domain class label during training.
Interesting combination of unsupervised learning, 3D-reconstruction, and conditional GAN is presented in [33]. Authors suggested an architecture (HoloGAN) that allows unsupervised learning of 3D representations directly from natural images. After learning the 3D structure and understanding a target pose, HoloGAN can generate new views of the same scene. HoloGAN tries to understand 3D representation of the world and then apply 3D rigid-body transformation for the found representation while other similar GANs learn to map a noise input vector directly to 2D features to generate images. As a result, in HoloGAN, a strong inductive bias about the 3D world is added into the generator network. on the first stage, HoloGAN tries to understand 3D representation, e.g., disentangle pose and identity. Model learns 3D features from a 4D constant tensor (size 4x4x4x512); HoloGAN performs an explicit 3D rigid-body transformation (3D rotation followed by trilinear resampling); projection unit generates 2D images (128x128) from the 3D structure.

2.3 GANs in fashion industry

Despite the considerable progress in GANs models, generating artificial face images is still a challenging task. In particular, one of such challenges is changing specific features like pose, face shape, and hairstyle. A newest NVIDIA papers [22] addresses this problem.
StyleGAN generates the artificial image gradually, starting from a shallow resolution and continuing to a high resolution (1024x1024). By modifying the input of each level separately, it controls the visual features that are expressed in that level, from coarse features (pose, face shape) to fine details (hair color), without affecting other levels [1].
To generate high-resolution images, many approaches from ProGAN [21] used in StyleGAN. But ProGan has limited ability to control specific features of the generated image. To add this control, several new technics are used in StyleGAN. One of them is Mapping Network which goal is is to encode the input vector into an intermediate vector whose different elements control different visual features. The Mapping Network consists of 8 fully connected layers. The second specific part is the AdaIN (Adaptive Instance Normalization) module that transfers the encoded information, created by the Mapping Network, into the generated image. Also, this model doesn't have traditional input which replaced on the constant values. StyleGAN is a state-of-the-art model that produces high-quality and realistic images and also allows superior control of the generated images.

One of the first attempt to implement clothing translation is suggested in [40], where special technic called Pixel-Level Domain Transfer is considered. There are defined two domains; for example, if an image of a dressed person is as a source domain, a piece of the person's clothing is the target domain. Big LookBook dataset is used for training purpose. This dataset contains 84k images in total, where 75k human images are associated with 10k top product images. In [18] presented Conditional Analogy GAN (CAGAN): image-to-image translation network that will exchange one piece of clothing y_i with a new one, y_j, on a given human image x_i (see Fig. 2.1). This task is complicated due to the following:

1. There are never examples of x_i^j, where x_i^j is the modified human image with the swapped fashion item we would like to see.
2. The model needs to find where the old clothing is located and replace it with the new clothing.
3. Additional transformation to correct for illumination, occlusion, 3D rotation, and deformation required for the clothing.

For training D and G authors defined a loss which contains several terms, weighted by constants γ_i, γ_c and introduced the adversarial loss that involves the generator and the discriminator.

FIGURE 2.1: Changing the upper-body garment of a human: the original x_i wears article y_i; we want to paint him wearing article y_j; the generated image with that property is x_{ji} . Note that CAGAN also generates not-in-place transformations (e.g. the neckline of the pullovers). Resolution is 128x96 pixels. Taken from [18]

Chapter 3

Method

This chapter describes the methods used in the thesis. The analysis of the related works proved that many of the well-known existing GAN models (conditional GAN, CycleGAN, StyleGAN, etc.) generate blurring and "bad looking" images during changing clothing on the images due to the following reasons: 1) it's difficult to capture and save all details related to color, style and texture of clothes; 2) human pose on the different images can be different and it should be taken into account during image generation; 3) some body parts presented on the generated image could be invisible on the source image. This chapter will describes two approaches that designed directly for cloth changing: Virtual Try-on Network and Liquid Warping GAN. General information about the methods, their modifications, loss functions and architecture details are provided in this chapter.

3.1 Changing clothing using Virtual Try-on Network

CAGAN models demonstrate impressive results on the image generation. But CA-GANs are not able to generate graphic details and accommodate geometric changes [41]. As a result they are not suitable for changing clothing on the people images.
To address these limitations, a Virtual Try-on Network is suggested in [13]. The key features of this model are:

1. usage of a clothing-agnostic input people image representation consisting of the following features: pose heatmap, human body representation, face and hair segmentation;
2. generation of the clothing region mask from the source image;
3. warping the target clothing item with help of the generated mask;
4. composition of the warped clothing item and the target person image.

Example of a clothing-agnostic person representation is shown on the Fig.3.1.

Many new models based on the original idea of Virtual Try-on Network have been presented in the recent time. In this thesis we will describe two of them: Characteristic-Preserving Virtual Try-On Network (CP-VTON) and VITON-GAN.

3.1.1 CP-VTON model

CP-VTON model suggested in [38]. Having a source image I_i of a person wearing in clothes c_i and a target clothes c, the goal of the model is to generate new image I_o wearing in the new cloth c_o, which saves the body shape, pose and hair style of the source image I_i and changes c_i on c [38].
Training with the triplets (I_i; c; I_t) where I_t is the ground truth of I_o is logical, but

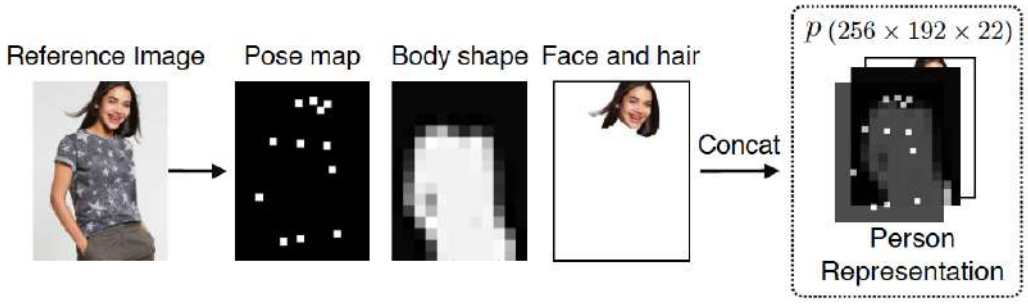


FIGURE 3.1: A clothing-agnostic person representation. Taken from [16]

not practical due to lack of such datasets. Training with triplets (I_t; c; I_t) affects the generalization ability of the model. According to [13], clothing-agnostic person representation p (see Fig.3.1) can be used instead of I_i, in other words, the training triplet (p; c; I_t) is used for training.

The enhancement of Virtual Try-on Network results can be reached by:

- improvement of shape-context matching for aligning clothes and body shape;
- improvement of inferior appearance merging strategy.

Geometric Matching Module (GMM) and Try-On Module (TOM) are suggested for these purposes.

Geometric Matching Module

GMM module pipeline is presented on the Fig.3.2. The main aim of this module is to transform the target clothes c into warped clothes $\hat{c}$ aligned with input clothing-agnostic person representation p. As described in [38], GMM consists of four parts:

1. two networks for extracting high-level features of p and c respectively;
2. a correlation layer to combine two features into a single tensor as input to the regressor network;
3. the regression network for predicting the spatial transformation parameters θ;
4. a Thin-Plate Spline transformation module T for warping an image into the output $\hat{c} = T_\theta(c)$.

Loss function of the module is L_1 distance between the warped result $\hat{c}$ and ground truth c_t:

$$L_{GMM}(\theta) = ||\hat{c} - c_t||_1 = ||T_\theta(c) - c_t||_1, \tag{3.1}$$

where c_t is the clothes worn on the target person in I_t.

Try-On Module

GMM module pipeline is presented on the Fig.3.3. The main aim of this module is to is to fuse $\hat{c}$ with the target person image for generation the final result.
As described in [38], given a concatenated input of person representation p and the warped clothes $\hat{c}$, UNet simultaneously renders a person image I_r and predicts a

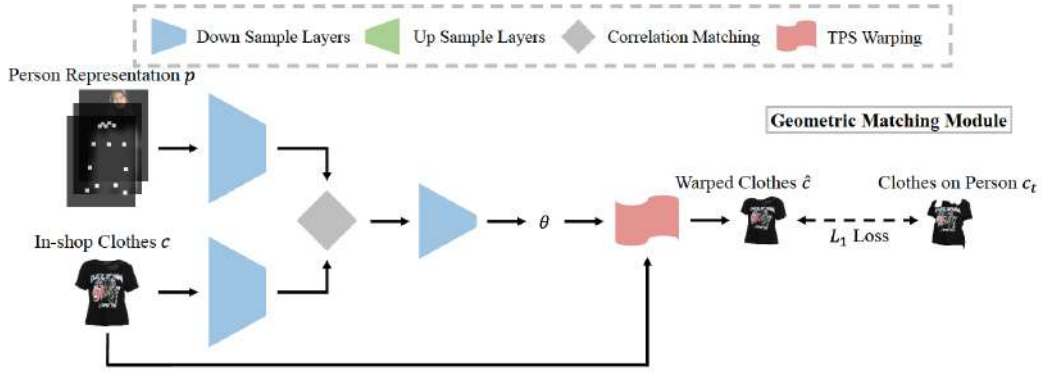


FIGURE 3.2: GMM module pipeline

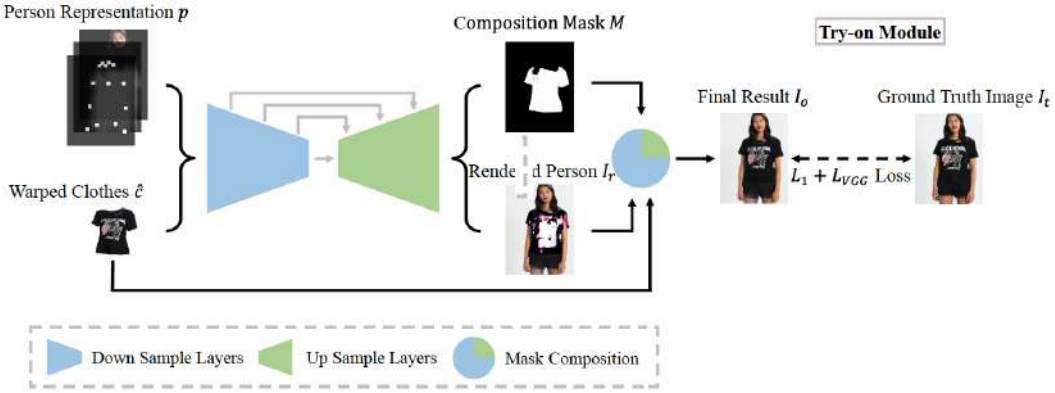


FIGURE 3.3: TOM module pipeline

composition mask M. The rendered person I_r and the warped clothes $\hat{c}$ are then fused together using the composition mask M to synthesize the final result I_o:

$$I_o = M \odot \hat{c} + (1 - M) \odot I_r,$$

where $\odot$ represents element-wise matrix multiplication.
The main aim of the module is to minimize difference between output I_o and ground truth I_t. The overall loss function for TOM is:

$$L_{TOM} = \lambda_{L1}||I_o - I_t||_1 + \lambda_{VGG}L_{VGG}(\hat{I}, I) + \lambda_{mask}||1 - M||_1, \tag{3.2}$$

where the VGG perceptual loss [19] is defined as follows:

$$L_{VGG}(I_o, I_t) = \sum_{i=1}^{5} \lambda_i||\psi(I_o) - \psi_i(I_t)||_1.$$

3.1.2 VITON-GAN model

Some modifications to the CP-VTON model is suggested in [16]. These modifications are:

1. Add adversarial loss to TOM module (see equation 3.2). Discriminator takes the TOM result image and person representation as inputs in order to make decision whether the result is real or fake.

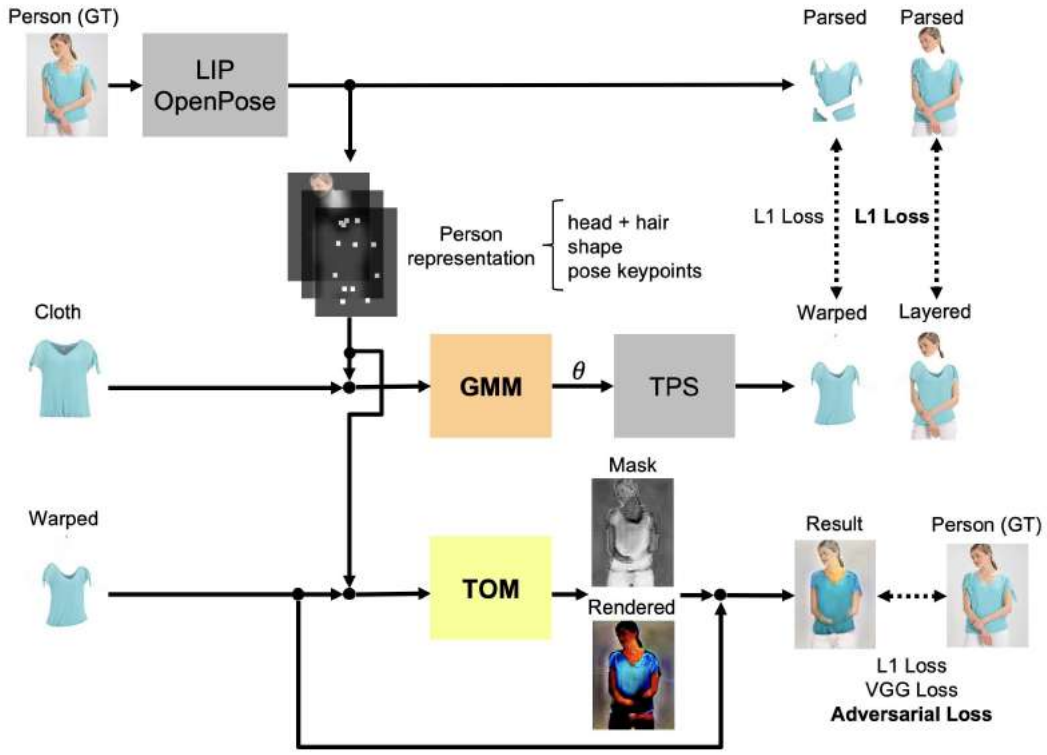


FIGURE 3.4: VITON-GAN pipeline. Taken from [16]

2. Add L1 distance distance between the generated and real images as loss function to GMM module (see equation 3.1).

3. Use additional data augmentation.

The model trained with these modifications called VITON-GAN in this thesis. The pipeline of VITON-GAN is presented on the Fig.3.4.

3.2 Changing clothing using Liquid Warping GAN

This model is presented in the article from ICCV 2019 [28]. The key fetures of these model are: 1) using a parametric statistical human body model which disentangles human body into pose (joint rotations) and shape; 2) using a special Liquid Warping Block (LWB) which addresses the problem of losing important source information.

3.2.1 General description

The model consists of three parts:

- mesh recovery;
- flow composition;
- GAN module.

The training pipeline is presented on the Fig. 3.5 . Source image (I_s) and target image (I_r) are the input arguments for the model. Based on the input images, the body

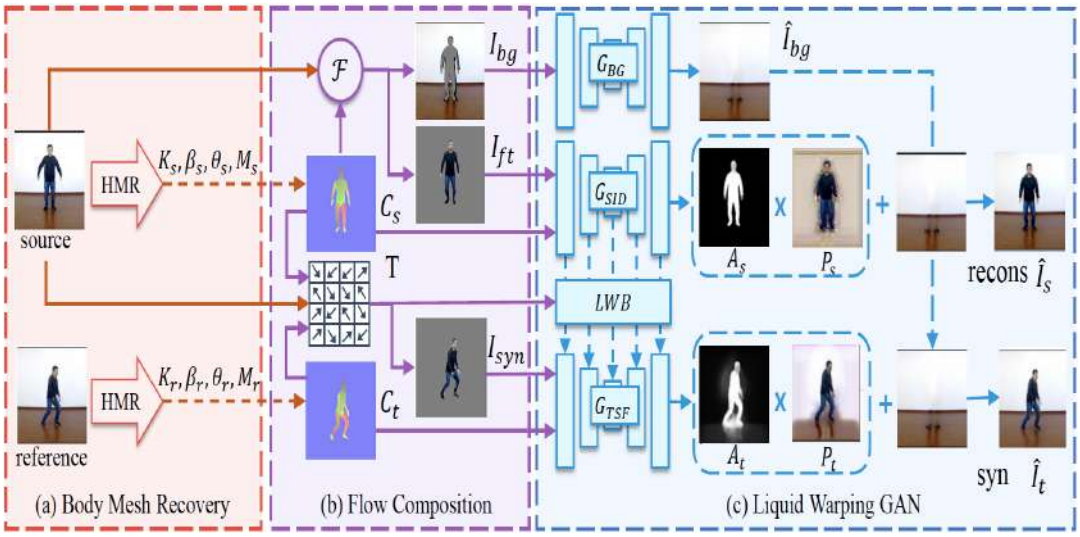


FIGURE 3.5: The training pipeline of Liquid Warping GAN. Taken from [28]

mesh recovery module creates the corresponding 3D mesh and their maps (C_s) and C_t).
Then the flow composition module uses these maps to calculate transformation flow T. The other task of the flow composition module is to divide source image I_s into background I_{bg} and foreground I_{ft}. Finally, using transformation flow T it produces a warped image I_{syn} from the I_s.
The generator consists of three streams. The first one G_{bg} takes I_{bg} as an input image and generate background image $\hat{I}_{bg}$; second one G_{sid} reconstructs source image $\hat{I}_s$ using I_{ft} and C_s; the third one generator G_{TSF} synthesizes the target image $\hat{I}_t$.
The key feature of this model is Liquid Warping Block (LWB) which addresses the problem of losing the source information.

Body Mesh Recovery Module

In this thesis the pre-trained model of human mesh recovery (HMR)[20] is using. Here an image is firstly encoded into a feature with R^{2048} by a ResNet-50[14] and then followed by an iterative 3D regression network that predicts the pose $\theta \in R^{72}$ and shape $\beta \in R^{10}$ of a skinned multi-person linear model (SMPL)[30], as well as the weak-perspective camera $K \in R^{10}$. SMPL is a 3D body model that can be defined as a differentiable function $M(\theta, \beta) \in R^{N_v \times 3}$, where N_v = 6890 - number of vertices, N_f = 13776 - number of faces $\theta \in R^{72}$ - pose, $\beta \in R^{10}$ - shape. The output of the module is the body reconstruction parameters of source and reference image correspondingly: $\{K_s, \theta_s, \beta_s, M_s\}$, $\{K_r, \theta_r, \beta_r, M_r\}$.

Flow Composition Module

A fully differentiable renderer, Neural Mesh Renderer (NMR) [23] is using for render a correspondence map of M_s and M_r under the camera view of K_s. For this purpose vertices of source V_s projected into 2D image space: $v_s = Proj(V_s; K_s)$. Here the source and reference correspondence maps marked as C_s and C_t.
Then the transformation flow $T \in R^{H \times W \times 2}$ (H and W - height and width of the image) is calculated by matching the correspondences between source correspondence

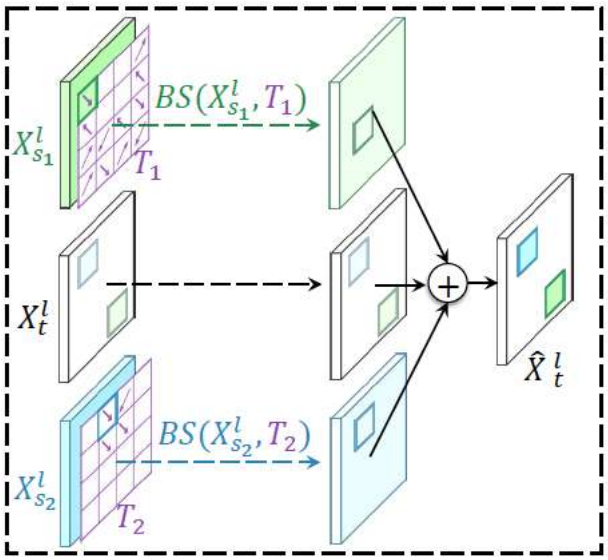


FIGURE 3.6: Liquid Warping Block. Taken from [28]

map C_s with its mesh face coordinates $f_s \in R^{N_f \times 2}$ and reference correspondence map C_t. A front image I_{ft} and a masked background image I_{bg} are derived from masking the input source image I_s based on C_s. Finally, the warped image I_{syn} is generated from applying the transformation flow T on the source images I_s.

Liquid Warping GAN

This stage generates high-fidelity human image under the desired condition. More specifically, it 1) synthesizes the background image; 2) predicts the color of invisible parts based on the visible parts; 3) generates pixels of clothes, hairs and others out of the reconstruction of SMPL.
As already mentioned the generator has three streams. In order to link the source with target streams without losing a source information, LWB is using here. One advantage of the proposed in [28] LWB is that it addresses multiple sources. For example it can preserve the head from source one, wear the upper outer garment from the source two and the lower outer garment from the source three. As a result of this advantage, the different parts of features can be added to G_{TSF} from the independent streams.
Fig. 3.6 presents a structure of LWB. X^l_{s1} and X^l_{s2} are the feature maps extracted by G_{SID} of different sources in l-th layers. X^l_T is the feature map of G_{TSF} at the l-th layer. The output features $\hat{X}^l_t$ aggregate the feature from G_{TSF} and G_{SID} without losing information and can be obtained as follows:

$$\hat{X}^l_t = BS(X^l_{s1}, T_1) + BS(X^l_{s2}, T_2) + X^l_t,$$

where BS - bilinear sampler.
The architecture of liquid warping GAN is presented on the Fig. 3.7.

The generators use ResNet and ResUnet architecture, no parameters share between them. The final image can be obtained as follows:

$$\hat{I}_s = P_s \times A_s + \hat{I}_{bg} \times (1 - A_s),$$

$$\hat{I}_t = P_t \times A_t + \hat{I}_{bg} \times (1 - A_t),$$

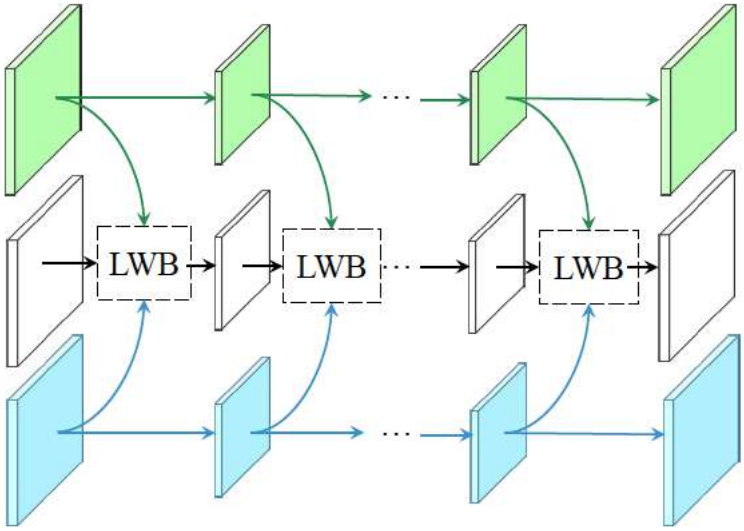


FIGURE 3.7: Liquid Warping GAN. Taken from [28]

where $\hat{I}_{bg}$ - output of G_{BG}.
The discriminator uses Pix2Pix architecture [17].

.

3.2.2 Loss function

The whole loss function consists of: perceptual loss[19], face identity loss, attention regularization loss and adversarial loss.

Perceptual Loss

It regularizes the reconstructed source image $\hat{I}_s$ and generated target image $\hat{I}_t$ to be closer to the original images I_s and I_r. It formulates in the following way:

$$L_p = ||f(\hat{I}_s) - f(I_s)||_1 + ||f(\hat{I}_t) - f(I_r)||_1, \tag{3.3}$$

where f - is a pre-trained VGG-19[37].

Face Identity Loss

It regularizes the cropped face from the generated target image $\hat{I}_t$ to be similar to the ground truth I_r. It forces the generator to preserve the face identity. It formulates in the following way:

$$L_f = ||g(\hat{I}_t) - g(I_r)||_1, \tag{3.4}$$

where g - is a pre-trained SphereFaceNet[27].

Adversarial Loss

It forces the distribution of generated images to the distribution of real images. LSGAN_{110}[31] loss is using in a way like PatchGAN for the generated target image $\hat{I}_t$. The discriminator D regularizes $\hat{I}_t$ to be more realistic-looking. We use conditioned discriminator, and it takes generated images and the correspondence map C_t

as inputs:

$$L_{adv}^{G} = \sum D(\hat{I}_t, C_t)^2. \tag{3.5}$$

Attention Regularization Loss

It prevents the attention map A from saturation. There is no ground truth of attention map A and color map P. In case of saturation of A, generator will not work. In order to avoid this situation, the mask is regularized to be closer to silhouettes S rendered by 3D body mesh. This loss formulates in the following way:

$$L_a = ||A_s - S_s||_2^2 + ||A_t - S_t||_2^2 + TV(A_s) + TV(A_t), \tag{3.6}$$

$$TV(A) = \sum_{i,j} [A(i,j) - A(i-1,j)]^2 + [A(i,j) - A(i,j-1)]^2. \tag{3.7}$$

For generator, the full objective function is shown in the following, and λ_p, λ_f and λ_a are the weights of perceptual, face identity and attention losses [28].

$$L^G = \lambda_p L_p + \lambda_f L_f + \lambda_a L_a + L_{adv}^{G}.$$

For discriminator, the full objective function is [28]:

$$L_D = \sum [D(\hat{I}_t, C_t) + 1]^2 + \sum [D(\hat{I}_r, C_r) - 1]^2.$$

3.2.3 Model architecture

The original architecture of the Liquid Warping GAN is available in the open repository on github [29].
The generator consists of three separate generators: background generator, source generator and transfer generator. More details related to the model's architecture are presented in Appendix A.
Background generator has ResNet architecture which is presented in Table A.5. The architecture of residual block is presented in Table A.3.
The source generator and transfer generator has more complicated (in comparison with ResNet) ResUNet architecture, which is presented in Table A.7 and Table A.9.
The discriminator uses Pix2Pix GAN model, which architecture is presented in the Table A.1.

Chapter 4

Training process

The choice of the two methods to implement, Virtual Try-on Network and Liquid Warping GAN, is based on the comparison study in the theory chapter. Both models show promising results on changing clothes on the people images. The key difference of the proposed methods over CAGAN is that they get pose, body shape and cloth details at the initial step of their pipeline. Using this information the models are able to generate just selected region of the image (cloth) and transform it according to the pose and body shape information. It helps to generate more realistic images. This chapter will describe training process of the selected models with more details.

4.1 CP-VTON and VITON-GAN model training

4.1.1 Dataset

For CP-VTON training there is used dataset proposed in [13]. It consists of 16,253 pairs of person and the corresponding cloth. This dataset was divided on 14221 and 2000 pairs for the training and test purpose, respectively. All of the images have been fitted to a common template so they have the same size 192x256. Example of dataset pair is presented on the Fig.4.1.

CP-VTON uses a human parser and pose estimator for extracting the person information and cloth information independently. This part isn't included in the current pipeline because the already preprocessed dataset from [38] is used instead. In this dataset OpenPose framework [10], [9],[36] [39] is used for extracting pose info into json format as a set of 2d keypoints. The Look Into Person[26] is used for the segmentation labels generation. CP-VTON requires the keypoints from OpenPose and segmentation labels from Look Into Person. This dataset was used in [38] as well. Example of the preprocessed pair is presented on the Fig.4.2. These images will be used used during models training.

As described in 3.1.1, the main aim of GMM is to transform the target clothes into warped clothes. This transformation should correspond to the target's person information. Example of GMM module transformation is presented on the Fig.4.3.

4.1.2 GMM module training

In GMM training the following hyper-parameters used: $\lambda_{L1} = \lambda_{VGG} = 1$, Adam [24] optimizer parameters: $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Training process continues for 200k steps, learning rate for the first 100k steps is equal to 0.0001 and linearly decays for the following steps. Changes of the loss function for training GMM model is presented on the Fig.4.4.

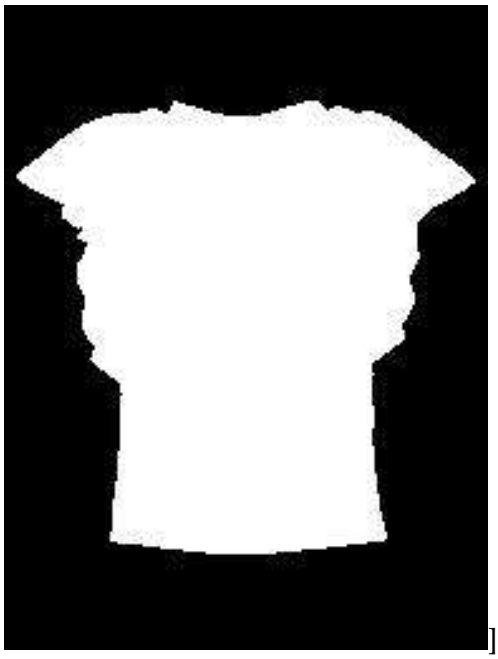
Demonstration of the GMM module training is presented on the Fig.4.5.

(A) Person image

(B) Cloth image

FIGURE 4.1: Example of pair from dataset

(A) Person image segmentation



(B) Cloth image mask

FIGURE 4.2: Example of preprocessed pair from dataset

(A) Cloth image (B) Warped cloth image

FIGURE 4.3: Example of GMM transformation

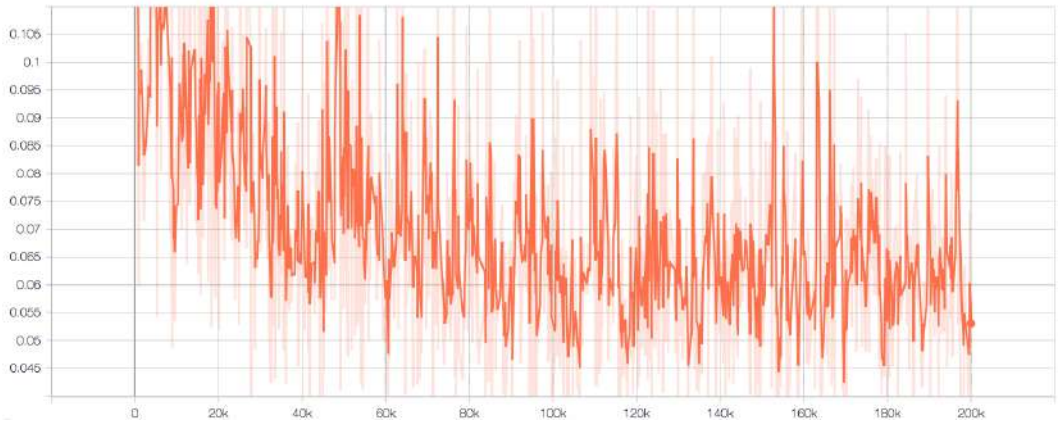
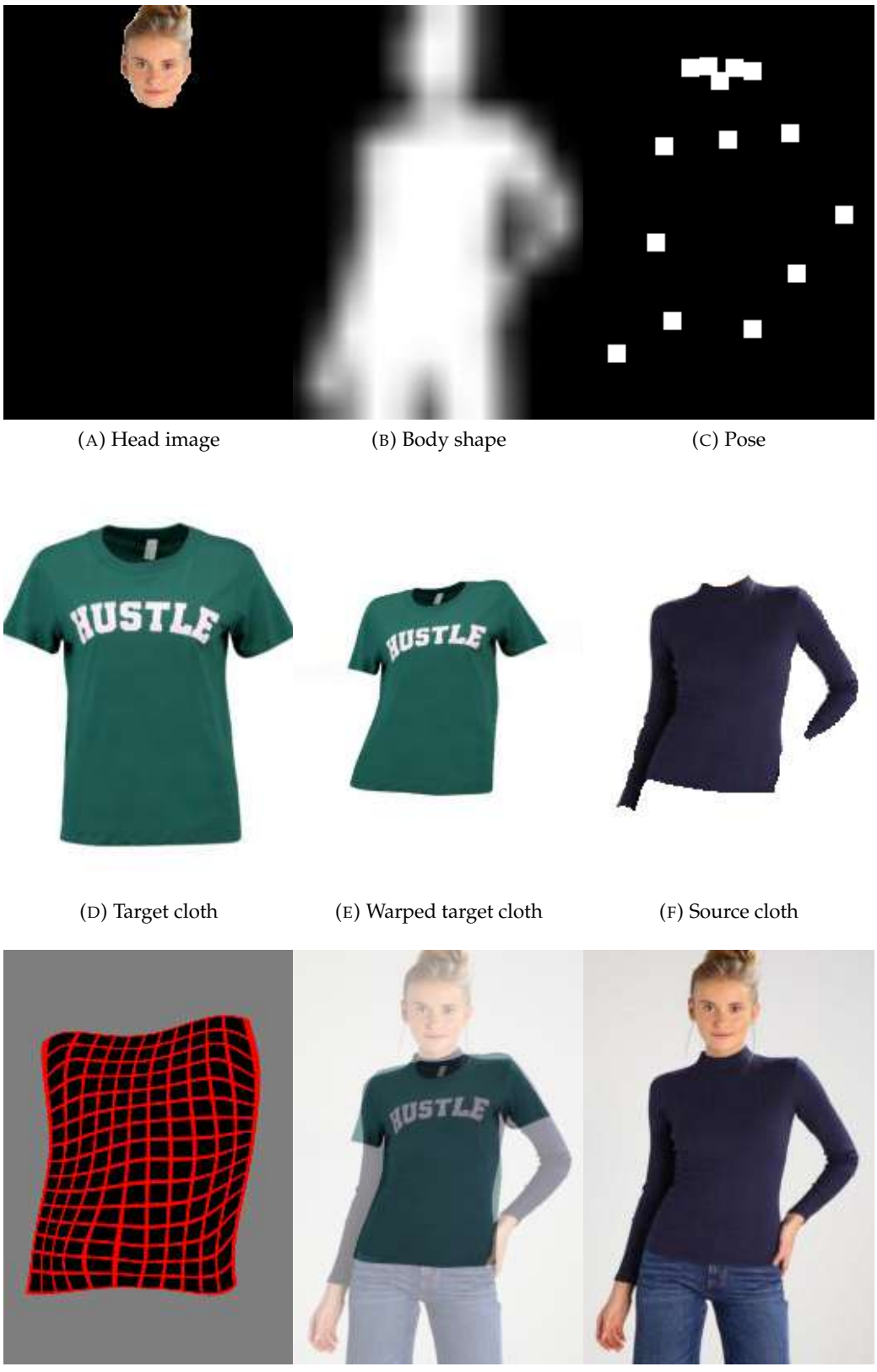


FIGURE 4.4: Loss function for training GMM model. The value of loss function at the end of the training process is 0.0491.

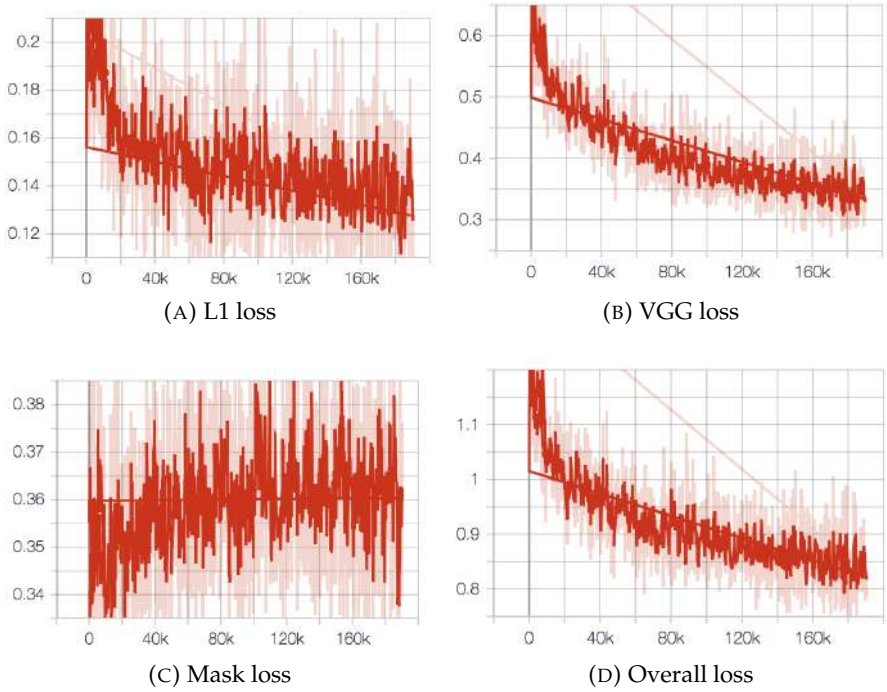


(A) Head image (B) Body shape (C) Pose

(D) Target cloth (E) Warped target cloth (F) Source cloth

(G) Warped grid image (H) Target person (I) Source person

FIGURE 4.5: Demonstration of the GMM module training



(A) L1 loss (B) VGG loss

(C) Mask loss (D) Overall loss

FIGURE 4.6: Loss function for training TOM model

Fig.4.5a, Fig.4.5b, Fig.4.5c, Fig.4.5i and Fig.4.5d are the input images of GMM module. Then module parses source cloth Fig.4.5f, get it's transformation grid Fig.4.5g and change target cloth Fig.4.5d to the warped target cloth Fig.4.5e. Image Fig.4.5h is a result of overlapping Fig.4.5e and Fig.4.5i.

4.1.3 TOM module training

Having the warped image which is aligned with the target's body pose, TOM module is able to generate final try-on result. As described in 3.1.1 Unet generator is used for this purpose. The hyper-parameters are the same as was used for GMM module training. Loss function components of the TOM module is described in 3.1. Changes of the loss function for training TOM model is presented on the Fig.4.6.

Demonstration of the TOM module training is presented on the Fig.4.7.

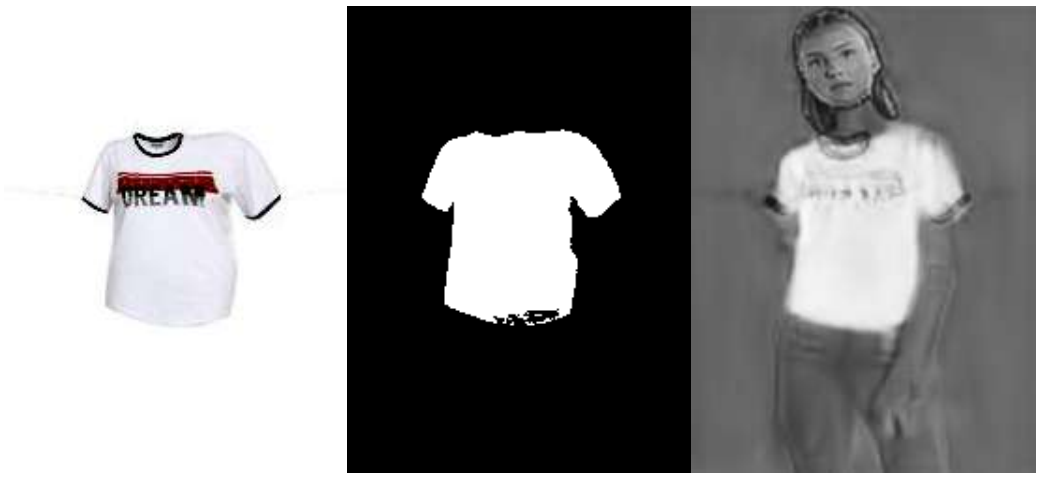
Fig.4.7a, Fig.4.7b, Fig.4.7c, Fig.4.7d and Fig.4.7e are the input images of TOM module. The module generates the image that will split on two images: the Fig.4.7f and Fig.4.7g. These splitted images join with Fig.4.7d using the formula described in 3.1 and produce Fig.4.7h as an output of the generator. The generated image Fig.4.7h and ground-truth image Fig.4.7i are the input of discriminator.

4.2 Liquid Warping GAN

4.2.1 Dataset

For training the suggested model, iPER dataset is used[28]. There are 30 subjects of different conditions of shape, height and gender. Each subject wears different clothes and performs an A-pose video and a video with random actions. Some subjects might wear multiple clothes, and there are 103 clothes in total. The whole dataset consists of 206 video sequences with 241,564 frames. The training/testing ratio is

(A) Head image (B) Body shape (C) Pose



(D) Warped cloth (E) Masked warped target cloth (F) Composite output

(G) Rendered output (H) Output (I) Ground truth

FIGURE 4.7: Demonstration of the TOM module training

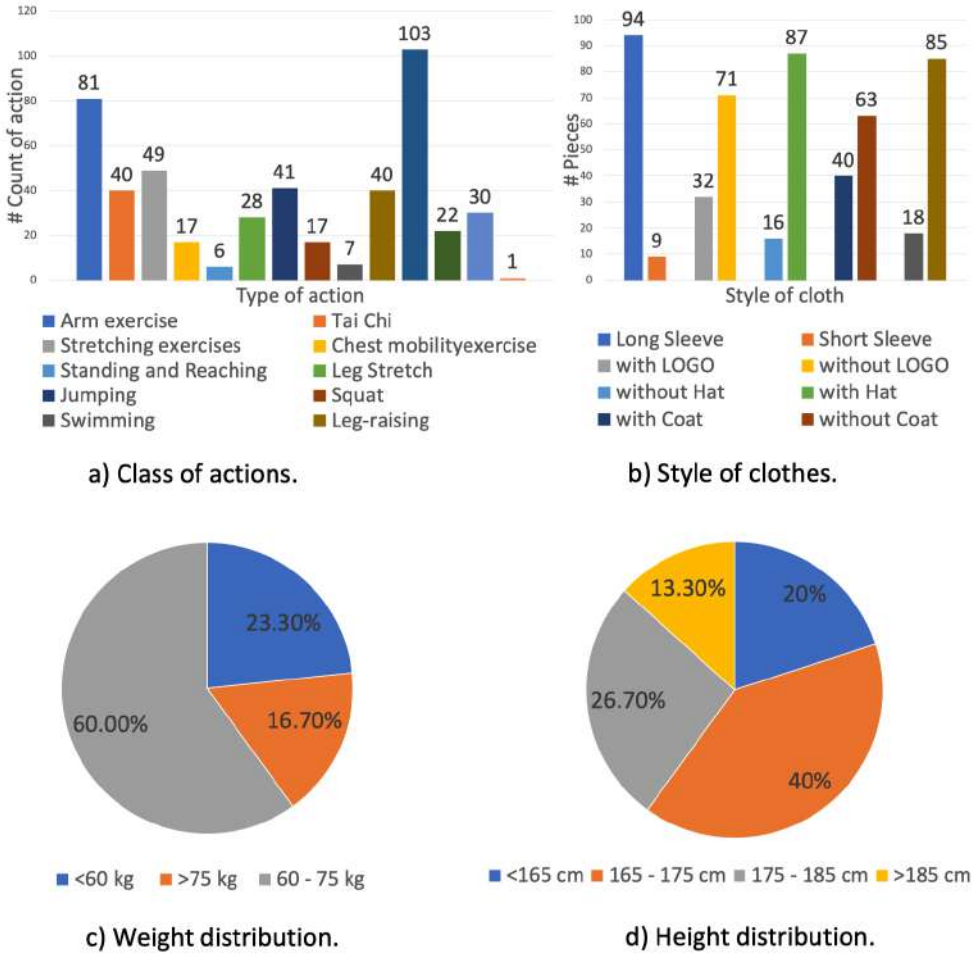


FIGURE 4.8: Details of iPER dataset: (a) shows the class of actions and their number of occurrences; (b) shows the styles of clothes; (c) and (d) are the distributions of weight and height of all 30 actors. Taken from [5]

8:2. More details of iPER dataset is presented on the Fig.4.8. .
Additionally training option is to use Place2 dataset for improving the ability of background generalization. This dataset can be downloaded from [7].

4.2.2 Training Liquid Warping GAN

The training process organized in the following way. Two random images selected from each video. They are normalized to range [-1..1] and resized to the size 256x256. As it was already noted, human mesh recovery is using pre-trained HMR model. SMPL is using for rendering purpose.
The Liquid Warping GAN training process continues for 30 epochs. The following hyper-parameters are using: $\lambda_p = 10.0, \lambda_f = 5.0, \lambda_a = 1.0$. Adam is used for parameter optimization of both generator and discriminator. The initial value of learning rate is chosen as 0.0002 for both generator and discriminator and linearly decays each epoch. The mini-batch size depends on the available GPU memory and was selected as 10 for this experiments.

Discriminator loss function is presented on the Fig.4.9.

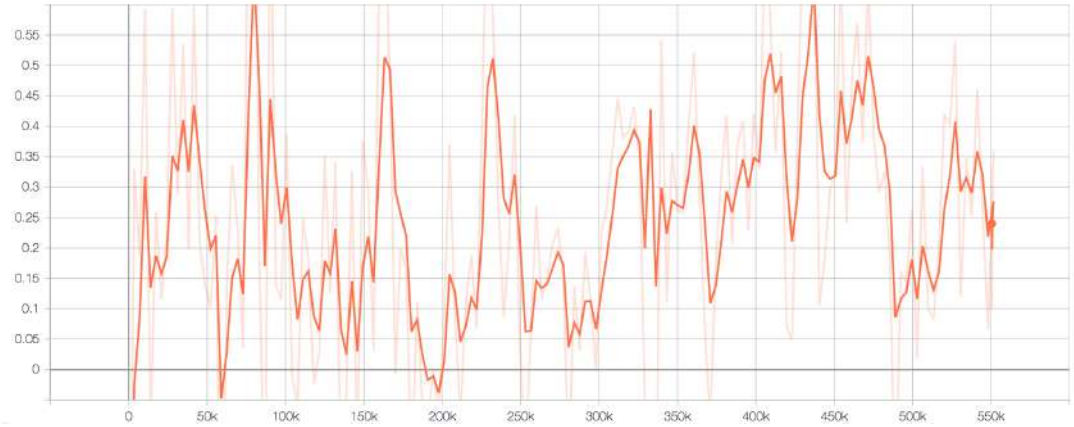


FIGURE 4.9: Liquid Warping GAN: discriminator loss function. Loss value at the end of training is 0.3.

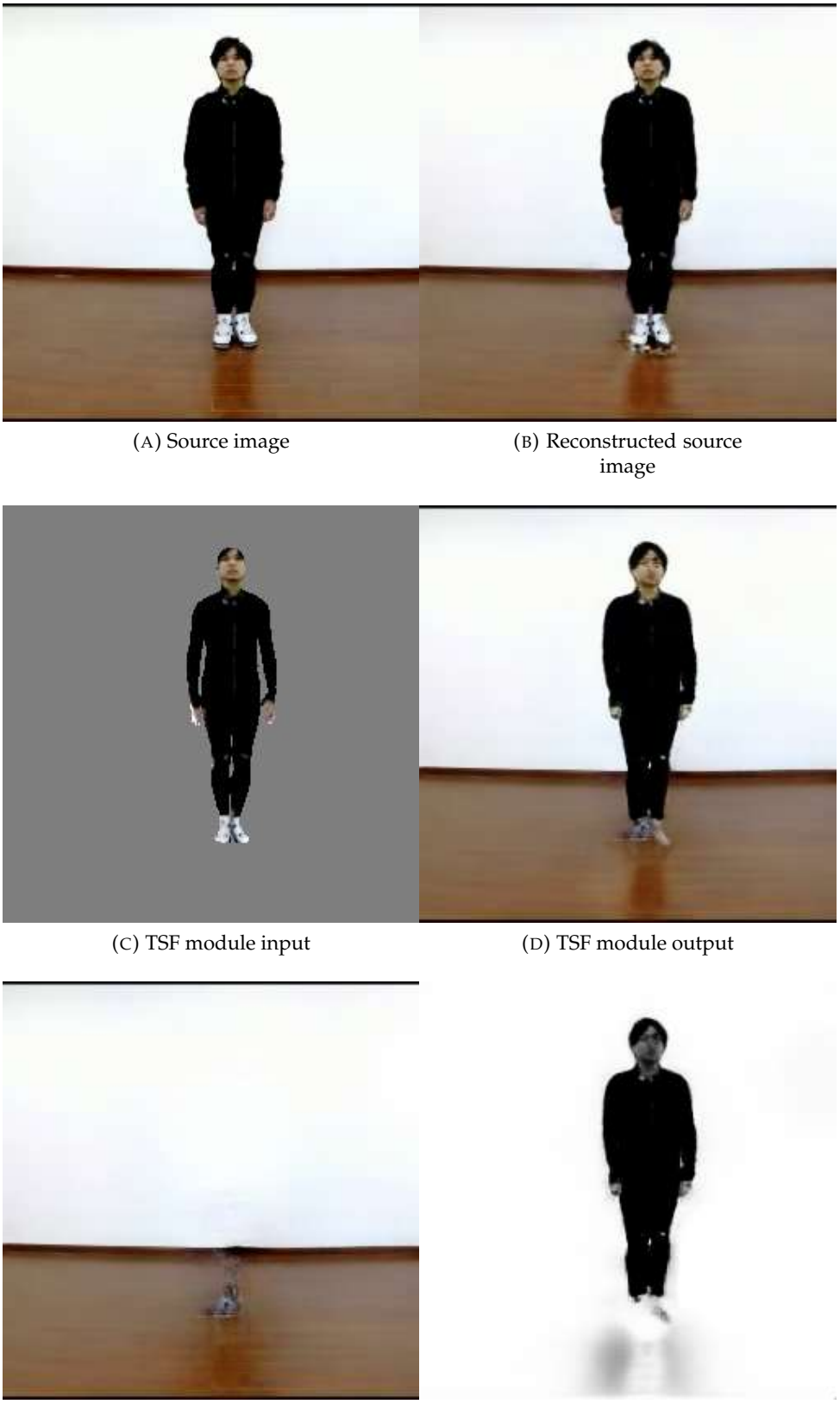
As described in 3.2.2, whole loss function consists of 4 elements. The corresponding plots are presented in in Appendix A. Perceptual loss function (see equation 3.3) is presented on the Fig.A.1.

Face identity loss (see equation 3.4) is presented on the Fig.A.2.

Adversarial identity loss (see equation 3.5) is presented on the Fig.A.3.

Attention regularization loss identity loss (see equation 3.6 and 3.7) is presented on the Fig.A.4.

Demonstration of the GMM module training is presented on the 4.10. In terms of definitions from section 3.2: 4.10a corresponds to the source image I_s, 4.10b - reconstructed source image $\hat{I}_s$, 4.10c - TSF input I_{syn}, 4.10d - TSF output $\hat{I}_t$, 4.10e - restored background $\hat{I}_{bg}$, 4.10f - attention map A_s.



(A) Source image

(B) Reconstructed source image

(C) TSF module input

(D) TSF module output

(E) Restored background

(F) Attention map A visualization

FIGURE 4.10: Demonstration of Liquid Warping GAN training

Chapter 5

Results

Results of Viton-GAN and Liquid Warping GAn are presented in this chapter. Several quantative indicators for evaluating models that generate images such as the Inception Score [35] and FID [15] are not useful for judging the quality of dressing [25]. So, just qualitative evaluation is used in this chapter.

5.1 Virtual Try-on Network results

As described in 3.1 there are suggested two models based on Virtual Try-on Network:

- CP-VTON
- VITON-GAN

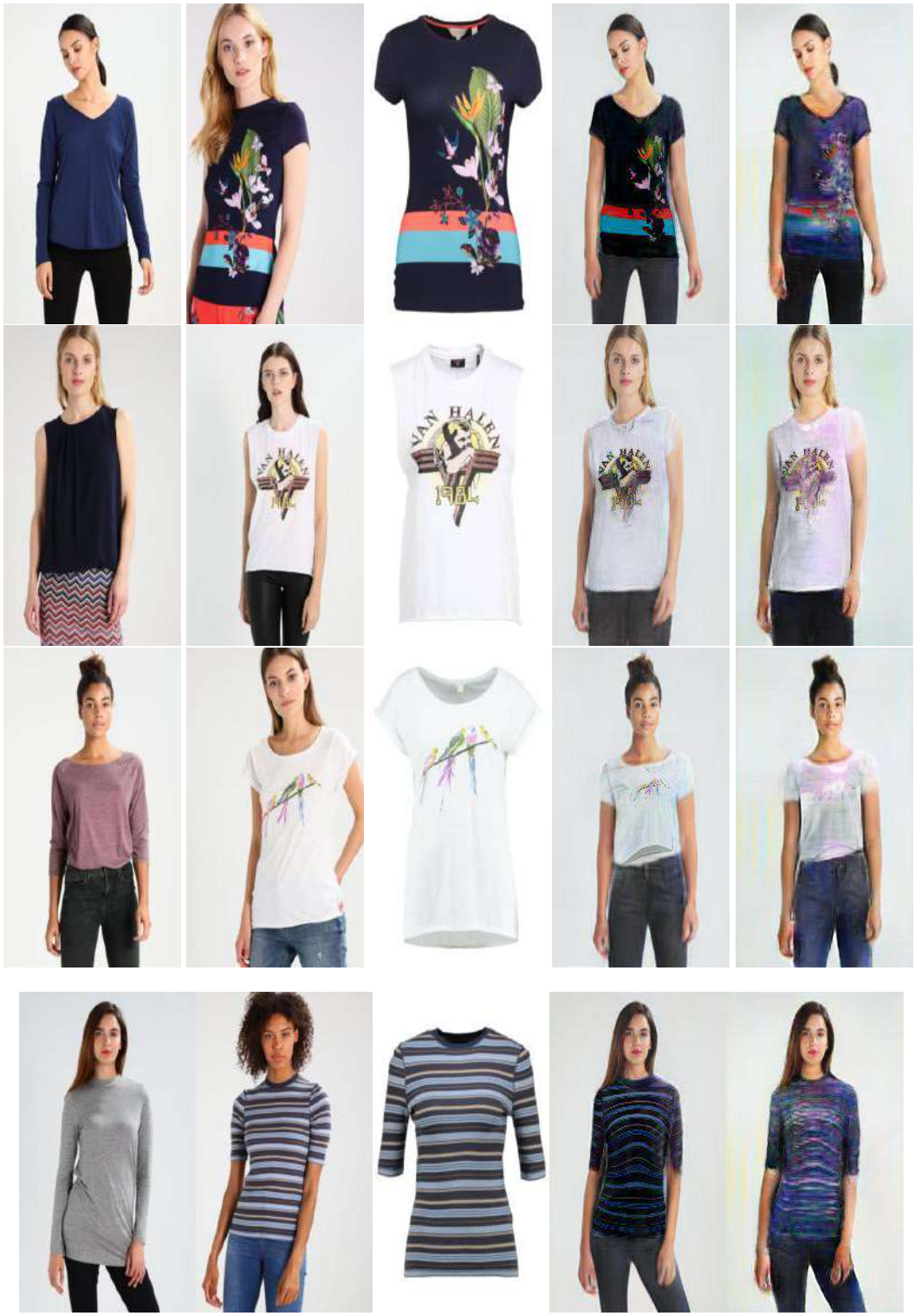
For comparing results, images from dataset described in 4.1.1 are used. Qualitative results of Virtual Try-on Network models is presented on the Fig.5.1. The following columns are presented on this image:

- A - source image
- B - target image
- C - target cloth
- D - CP-VTON results
- E - VITON-GAN results

In general, the results presented in 5.1 look good. Visually, the results in column D better than in column E, so we can make a conclusion that CP-VTON model provides better results. The result image in the higher resolution are available in Appendix B. However in some cases Virtual Try-on Network models generate bad results. Some examples of the failed cases are presented on the Fig.5.2.
From 5.2 we can make a conclusion that model can not handle case when the hands have unusual position, for example, along the body. Such examples are presenteds on the first three examples from Fig.5.2. On the fourth example the source image is back to camera. We can consider these situations as "corner cases". Additional training images are required for handling such "corner cases".

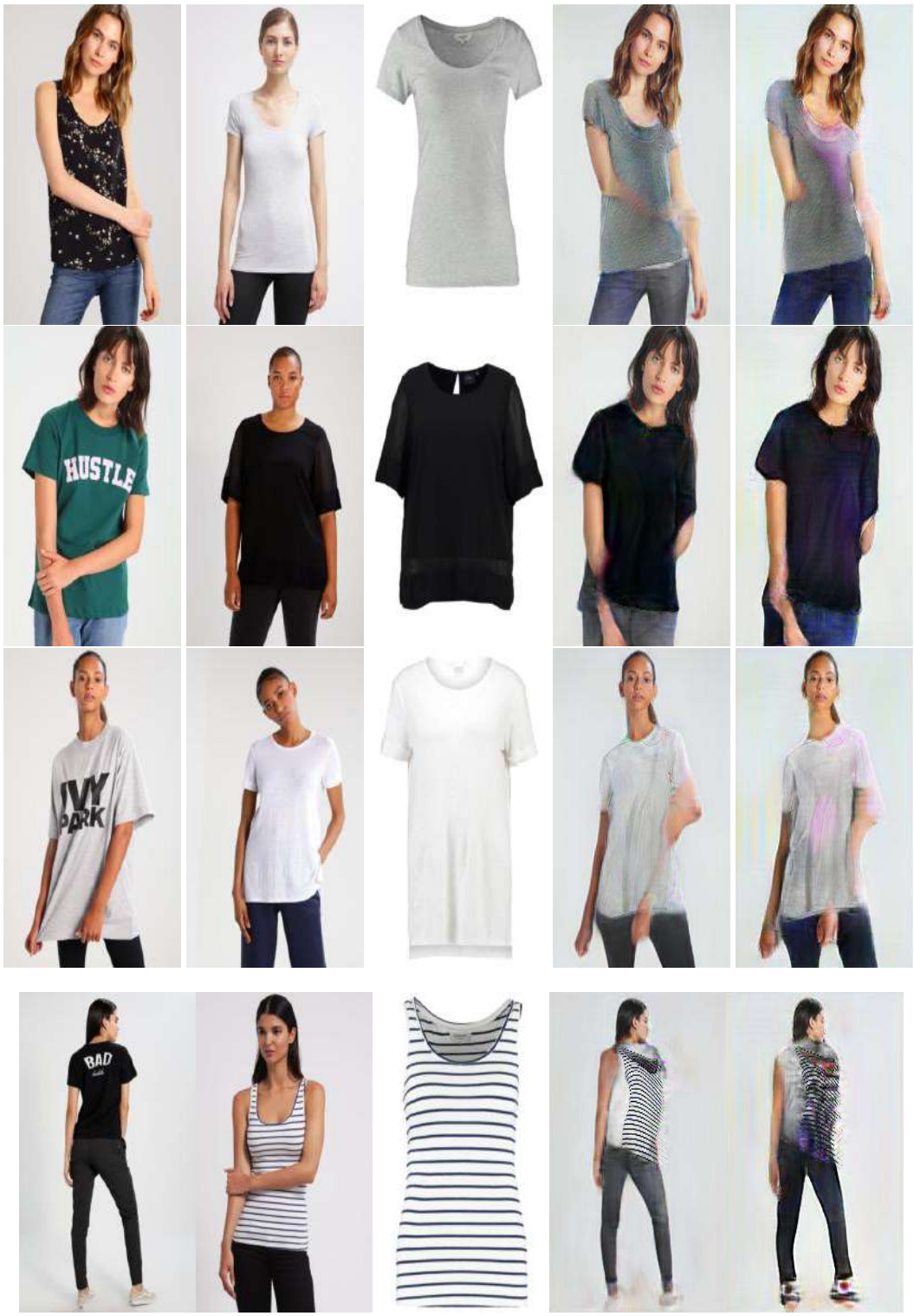
5.2 Liquid Warping GAN results

The results of three modifications of Liquid Warping GAN training process are considered here:



(A) Source image (B) Target image (C) Target cloth (D) CP-VTON (E) VITON-GAN

FIGURE 5.1: Qualitative results of Virtual Try-on Network



(A) Source image (B) Target image (C) Target cloth (D) CP-VTON (E) VITON-GAN

FIGURE 5.2: Failed cases of Virtual Try-on Network

- training with iPER dataset;
- training with iPER+Place2 dataset;
- iPER+Place+DeepFashion dataset.

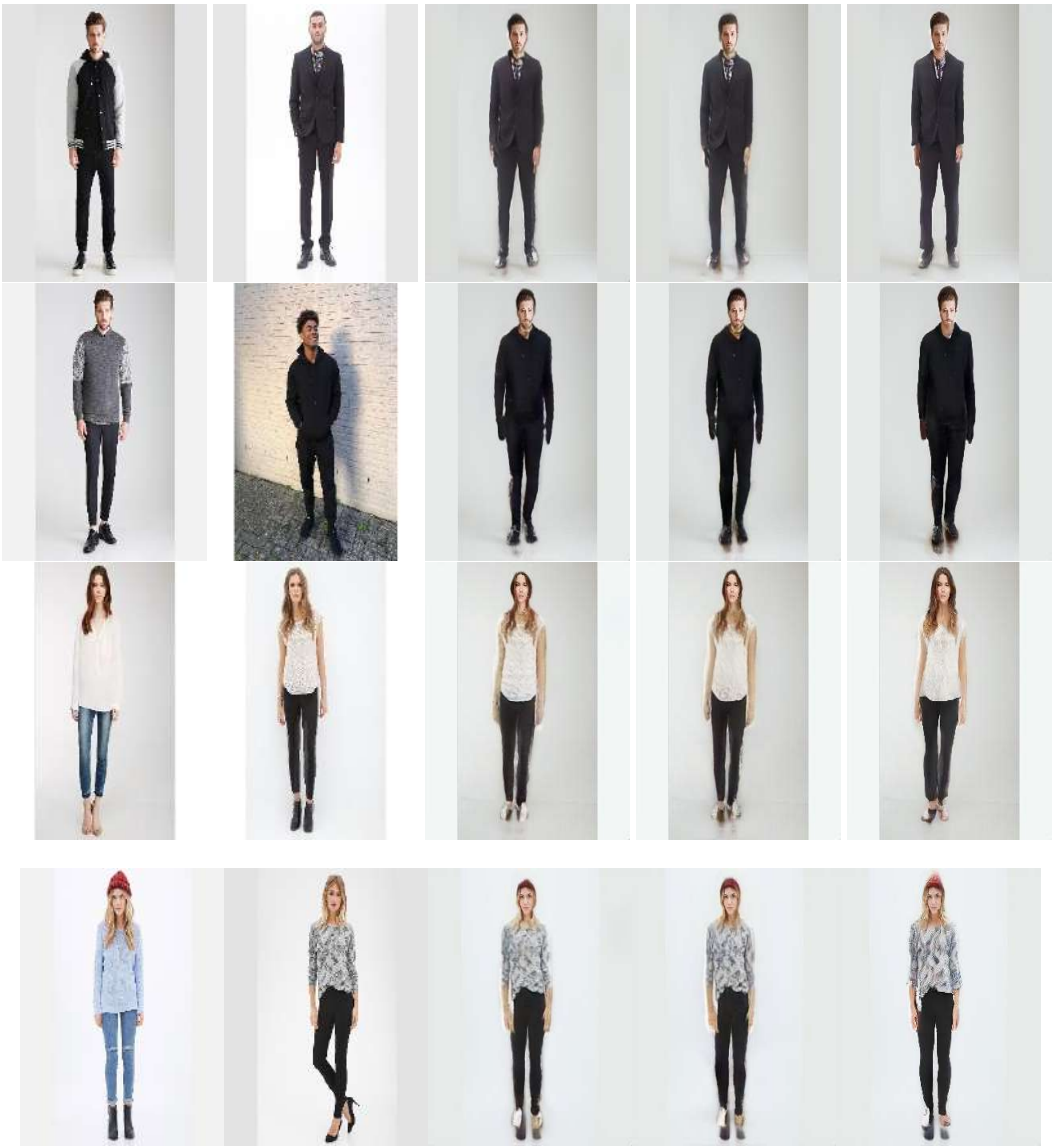
The training procedure with iPER+Place+DeepFashion dataset isn't done in this thesis, the coefficients of the pre-trained model are used instead. More details about training with iPER dataset and iPER+Place2 dataset are provided in 4.2.2.

Qualitative results of Liquid Warping GAN is presented on the Fig.5.3. The following columns are presented on this image:

- A - source image
- B - target image
- C - results of model trained with iPER dataset
- D - results of model trained with iPER+Place2 dataset
- E - results of model trained with iPER+Place2+DeepFashion dataset

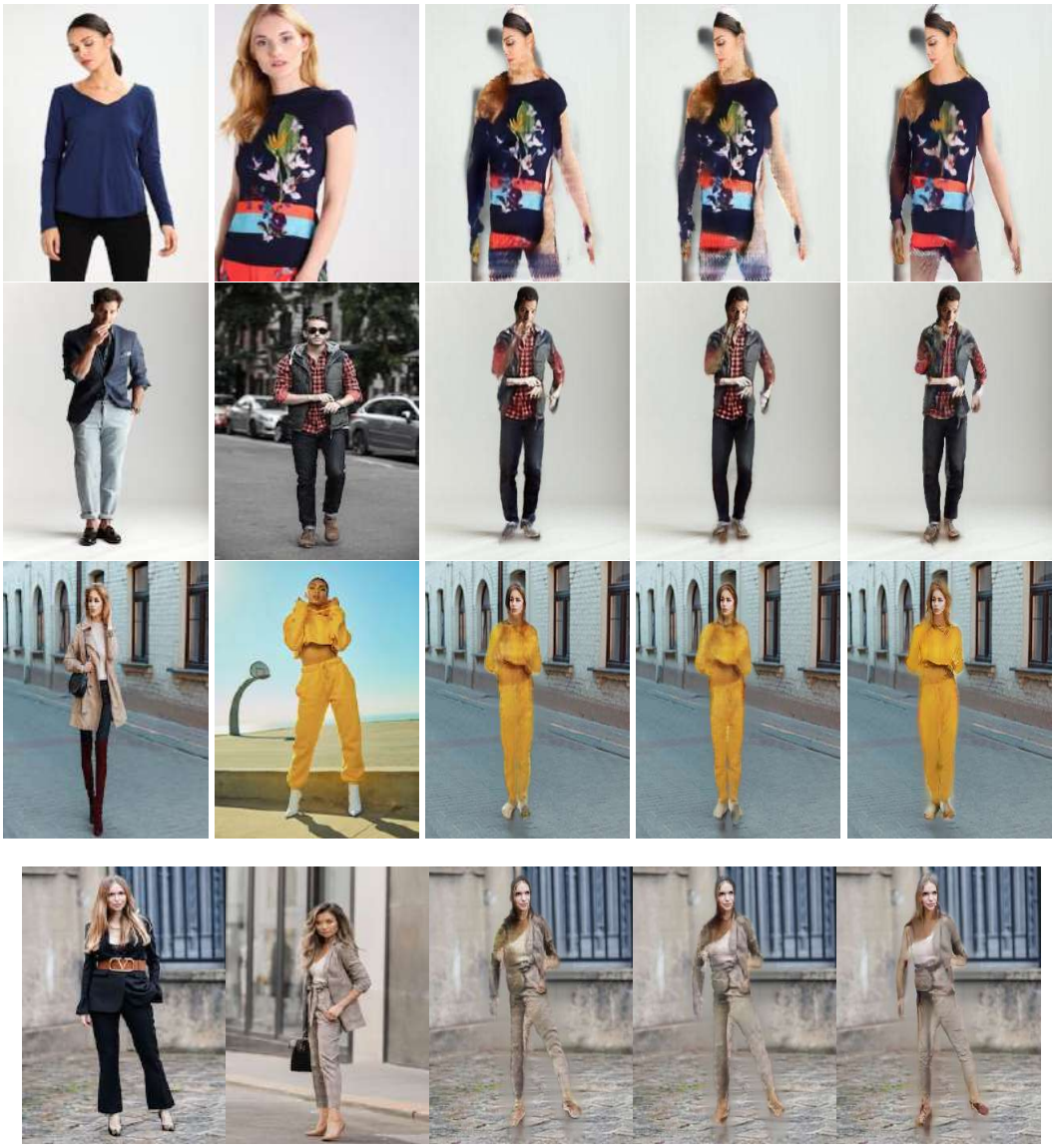
There are used custom images from internet. In general results are acceptable, however some blurring effects also take a place. All of the models provide competitive results, maybe with slightly better results in the last column. The results in higher resolutions are available in Appendix B.

Also some failed cases of Liquid Warping GAN have investigated. Examples of the failed cases are presented on the Fig.5.4.
First failed example is taken from the dataset used for Virtual Try-on Network. The reason of this fail is that Liquid Warping GAN had been trained for the full height images, but there is input with half-height image.
In the second example, the hands are along the body and Liquid Warping GAN also handles such cases poorly.
In the third and fourth examples, the cloth transfer happens but the resulting images have non-natural view.



(A) Source image (B) Target image (C) iPER (D) iPER+Place2 (E) iPER+Place2+ DeepFashion

FIGURE 5.3: Qualitative results of Liquid Warping GAN



(A) Source image (B) Target image (C) iPER (D) iPER+Place2 (E) iPER+Place2+ DeepFashion

FIGURE 5.4: Failed cases of Liquid Warping GAN

Chapter 6

Conclusion

The purpose of the thesis was to investigate the possibilities of changing clothing on the people images using GAN models. Among the investigation was the goal to choose the existing models, reproduce them, compare the results and generate as visually realistic images as possible.
Changing clothing on the people images have been done using Virtual Try-on Network and Liquid Warping GAN models. The theory, implementation and training of the selected methods have been provided. Results have been compared by visual inspections. Also, modifications to proposed models have been made in order to improve the model results. The following section describes what was done in this thesis with more details.

6.1 What was done

- learned the general principles of the GAN models;
- researched the existing GAN models that are applicable in fashion industry;
- selected two approaches for the clothing transfer: Virtual Try-on Network and Liquid Warping GAN;
- described the selecterd methods: loss functions, implementation and training details, model's architecture;
- from scratch trained two models related to Virtual Try-on Network approach: CP-VTON and VITON-GAN;
- from scratch trained two models related to Liquid Warping GAN: using iPER dataset and iPER+Place2 datasets;
- compared the results obtained from the trained models.

The investigations have resulted in answers to the following questions:

1. *Can GANs be used to changing clothing on people images?*
   In general, yes. The trained models demonstrate the acceptable results on the test data.
   However, for generalization purpose, the additional improvements required. As it was demonstrated on the Fig.5.2, the CP-VTON and VITON-GAN don't handle cases when hand is along the body, when a person is back to camera or when a full height people image is provided. Also, at the current moment, CP-VTON and VITON-GAN require specially prepared input data, because their pipeline isn't able to get clothing-agnostic person representation from source

image.
As it was shown on the Fig.5.4, Liquid Warping GAN also has some limitations. It's suitable just for full height people image, it doesn't handle cases when hands have unusual positions, it has problem in hair style transfer. Additional training data required for handling these problems. In comparison with CP-VTON and VITON-GAN, Liquid Warping GAN has HMR module for getting body shape and pose information.

2. *Which model should be used for the best possible visual results?* According to the obtained visual results - CP-VTON model.

3. *How it's possible to improve the model to get better results?* See Section 6.3.

6.2 Implications

Understanding of the opportunities with GAN models, in the changing clothing on the people images, provides a platform for development a framework that can be valuable in the fashion and marketing industry, social networks, people photos and entertainment area. From the investigation made in this thesis, the selected models can be used as a base for the further development.

6.3 Future work

According to the experiment results, I would suggest to use CP-VTON model as a base for the future development. The following further steps for improvement are required:

1. transform source image to the clothing-agnostic person representation automatically in the pipeline;

2. check if the selected model handles images with full-height person images;

3. investigate how to handle cases when hands are along the body or in other unusual position;

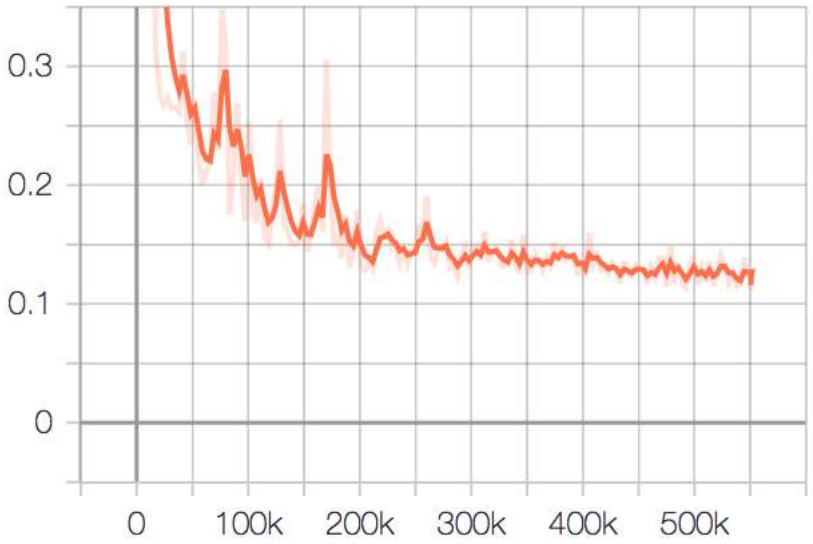
4. increase the resolution of the generated images.

Appendix A

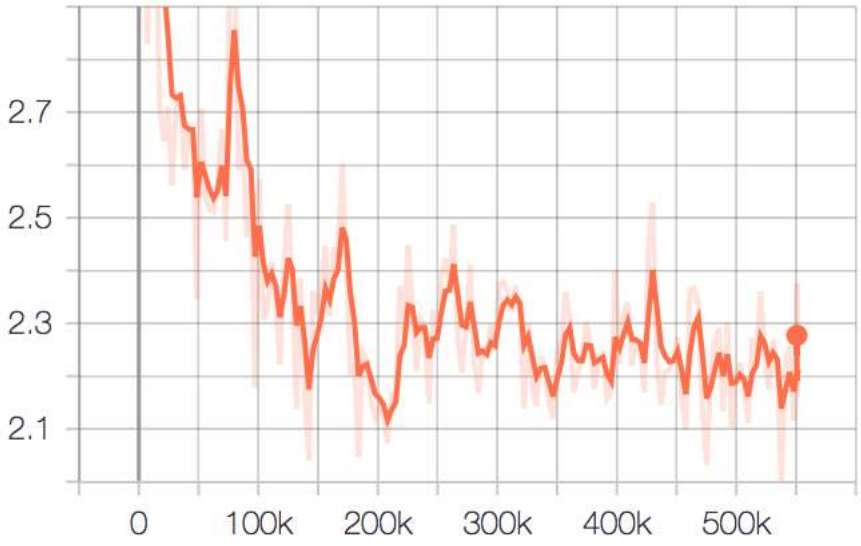
Liquid Warping GAN architecture and loss functions

A.1 Architecture

A.2 Loss functions



(A) Recovery loss



(B) Generated image loss

FIGURE A.1: Perceptual loss

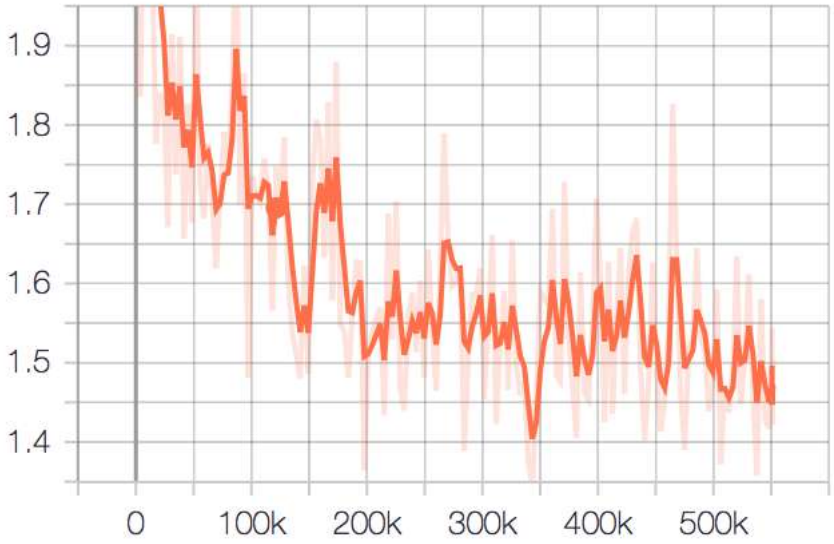


FIGURE A.2: Face identity loss

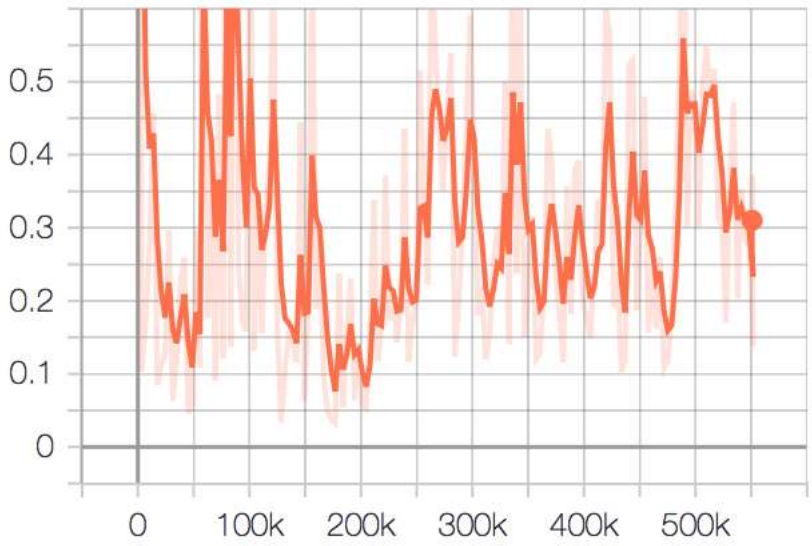
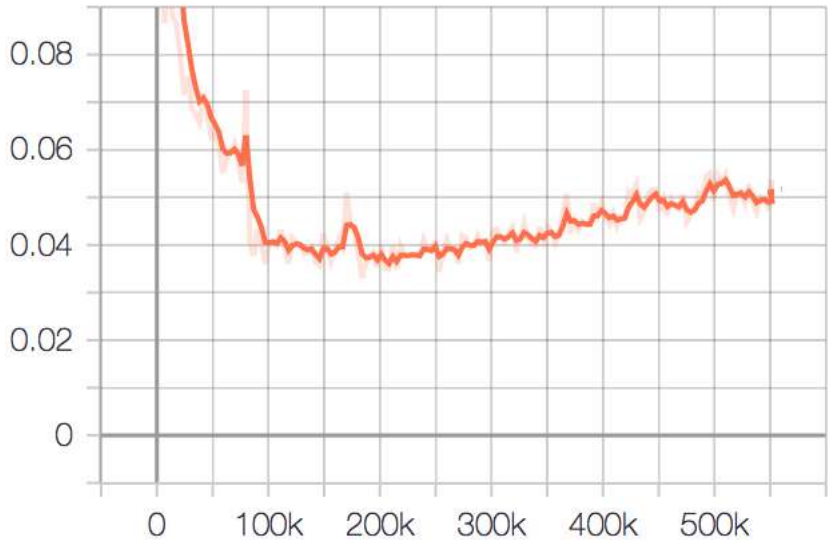
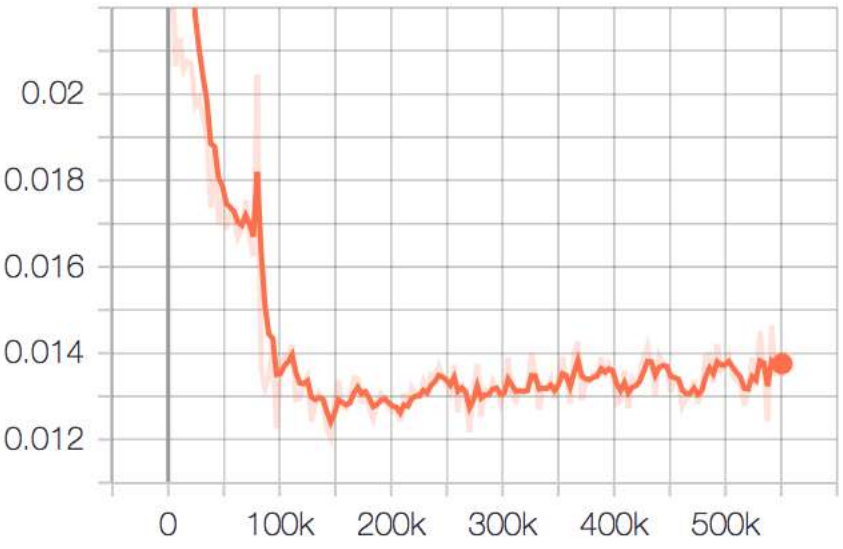


FIGURE A.3: Adversarial loss



(A) Attention loss (equation 3.6)



(B) Smoothed attention loss (equation 3.7)

FIGURE A.4: Attention regularization loss

TABLE A.1: Pix2Pix architecture (discriminator)

Layer number	Description
(0)	Conv2d(6, 64, kernel size=(4, 4), stride=(2, 2), padding=(1, 1))
(1)	LeakyReLU(negative slope=0.2, inplace=True)
(2)	Conv2d(64, 128, kernel size=(4, 4), stride=(2, 2), padding=(1, 1))
(3)	InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track running stats=False)
(4)	LeakyReLU(negative slope=0.2, inplace=True)
(5)	Conv2d(128, 256, kernel size=(4, 4), stride=(2, 2), padding=(1, 1))
(6)	InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=False, track running stats=False)
(7)	LeakyReLU(negative slope=0.2, inplace=True)
(8)	Conv2d(256, 512, kernel size=(4, 4), stride=(2, 2), padding=(1, 1))
(9)	InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=False, track running stats=False)
(10)	LeakyReLU(negative slope=0.2, inplace=True)
(11)	Conv2d(512, 512, kernel size=(4, 4), stride=(1, 1), padding=(1, 1))
(12)	InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=False, track running stats=False)
(13)	LeakyReLU(negative slope=0.2, inplace=True)
(14)	Conv2d(512, 1, kernel size=(4, 4), stride=(1, 1), padding=(1, 1))

TABLE A.3: Residual block architecture

Layer number	Description
(0)	Conv2d(512, 512, kernel size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(1)	InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(2)	ReLU(inplace=True)
(3)	Conv2d(512, 512, kernel size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(4)	InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=False)

TABLE A.5: ResNet architecture (background generator)

Layer number	Description
(0)	Conv2d(4, 64, kernel size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
(1)	InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(2)	ReLU(inplace=True)
(3)	Conv2d(64, 128, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(4)	InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(5)	ReLU(inplace=True)
(6)	Conv2d(128, 256, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(7)	InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(8)	ReLU(inplace=True)
(9)	Conv2d(256, 512, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(10)	InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(11)	ReLU(inplace=True)
(12)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(13)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(14)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(15)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(16)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(17)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(18)	ConvTranspose2d(512, 256, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(19)	InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(20)	ReLU(inplace=True)
(21)	ConvTranspose2d(256, 128, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(22)	InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(23)	ReLU(inplace=True)
(24)	ConvTranspose2d(128, 64, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(25)	InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(26)	ReLU(inplace=True)
(27)	Conv2d(64, 3, kernel size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
(28)	Tanh()

TABLE A.7: ResUNet architecture: encoders and decoders

Layer	Encoders
(0)	Conv2d(6, 64, kernel size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
(1)	InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(2)	ReLU(inplace=True)
(3)	Conv2d(64, 128, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(4)	InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(5)	ReLU(inplace=True)
(6)	Conv2d(256, 512, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(7)	InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(8)	ReLU(inplace=True)
(9)	Conv2d(256, 512, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(10)	InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(11)	ReLU(inplace=True)
	Resnets
(0)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(1)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(2)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(3)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(4)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(5)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
	Decoders
(0)	ConvTranspose2d(512, 256, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(1)	InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(2)	ReLU(inplace=True)
(3)	ConvTranspose2d(256, 128, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(4)	InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(5)	ReLU(inplace=True)
(6)	ConvTranspose2d(128, 64, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(7)	InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(8)	ReLU(inplace=True)

TABLE A.9: ResUNet architecture: skippers and regularizations

Layer	Skippers
(0)	Conv2d(512, 256, kernel size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(1)	InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(2)	ReLU(inplace=True)
(3)	Conv2d(256, 128, kernel size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(4)	InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(5)	ReLU(inplace=True)
(6)	Conv2d(128, 64, kernel size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(7)	InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(8)	ReLU(inplace=True)
	Img reg
(0)	Conv2d(64, 3, kernel size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
(1)	Tanh()
	Attention reg
(0)	Conv2d(64, 1, kernel size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
(1)	Sigmoid()
(9)	Conv2d(256, 512, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(10)	InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(11)	ReLU(inplace=True)
(12)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(13)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(14)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(15)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(16)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(17)	ResidualBlock(512, kernel size =(3,3), stride=(1, 1), padding=(1, 1))
(18)	ConvTranspose2d(512, 256, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(19)	InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(20)	ReLU(inplace=True)
(21)	ConvTranspose2d(256, 128, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(22)	InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(23)	ReLU(inplace=True)
(24)	ConvTranspose2d(128, 64, kernel size=(3, 3), stride=(2, 2), padding=(1, 1), output padding=(1, 1), bias=False)
(25)	InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=False)
(26)	ReLU(inplace=True)
(27)	Conv2d(64, 3, kernel size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
(28)	Tanh()

Appendix B

Results

B.1 Virtual Try-on Network results

There are presented results from Fig. 5.1 and Fig. 5.2 in higher resolution. First column corresponds to CP-VTON results, second - to VITON-GAN results.

B.2 Liquid Warping GAN results

There are presented results from Fig. **??** and Fig. **??** in higher resolution. First row corresponds to the results of model trained on iPER dataset, second - on iPER+Place2 datasets, third - on iPER+Place2+DeepFashion datasets.


VAN HALEN

Bibliography

[1] Explained: A style-based generator architecture for gans - generating and tuning realistic artificial faces. https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431.

[2] How computer vision can change the automotive industry. https://medium.com/neuromation-blog/how-computer-vision-can-change-the-automotive-industry-b8ba0f1c08d1. Accessed: 2018-08-08.

[3] How to evaluate generative adversarial networks. https://machinelearningmastery.com/how-to-evaluate-generative-adversarial-networks.

[4] Image-to-image translation. https://towardsdatascience.com/image-to-image-translation-69c10c18f6ff.

[5] Impersonator dataset details. https://svip-lab.github.io/dataset/iPER_dataset.html.

[6] Must-read papers about gans. https://towardsdatascience.com/must-read-papers-on-gans-b665bbae3317.

[7] Place 365 dataset. http://places2.csail.mit.edu/download.html.

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[9] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[10] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[11] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[13] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S Davis. Viton: An image-based virtual try-on network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7543–7552, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[16] Shion Honda. Viton-gan: Virtual try-on image generator trained with adversarial loss. *arXiv preprint arXiv:1911.07926*, 2019.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[18] Nikolay Jetchev and Urs Bergmann. The conditional analogy gan: Swapping fashion articles on people images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2287–2292, 2017.

[19] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[20] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018.

[21] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[23] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Shizuma Kubo, Yusuke Iwasawa, and Yutaka Matsuo. Generative adversarial network-based virtual try-on with clothing region. 2018.

[26] Xiaodan Liang, Ke Gong, Xiaohui Shen, and Liang Lin. Look into person: Joint body parsing & pose estimation network and a new benchmark. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):871–885, 2018.

[27] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[28] Wen Liu, Zhixin Piao, Jie Min, Wenhan Luo, Lin Ma, and Shenghua Gao. Liquid warping gan: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5904–5913, 2019.

[29] Wen Liu, Wenhan Luo Lin Ma Zhixin Piao, Min Jie, and Shenghua Gao. Liquid warping gan: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[30] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015.

[31] Xudong Mao, Qing Li, Haoran Xie, Raymond Yiu Keung Lau, Zhen Wang, and Stephen Paul Smolley. On the effectiveness of least squares generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[32] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[33] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. *arXiv preprint arXiv:1904.01326*, 2019.

[34] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[35] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[36] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[38] 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.

[39] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[40] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016.

[41] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.