Abstract

Nowadays pose or shape manipulation is a largely applied field. Especially in CGI. This report is about transfering the body pose manipulation method from Liqian Ma et al[1] to face/head pose manipulation. They use a two staged training process. First they generate a coarse pose manipulated image. Then they transfer details to the generated image by using a GAN-like training setting. Additionally, masking is applied to mark regions of interest.

Transfering this method to facial pose manipulation can be challenging by choosing the pose representation, masking, upsampling method of the generators and implementing it from scratch with the pytorch library.

Contents

1 Introduction

Nowadays image generation is a more and more used task. Tasks can be

- Image completion,
- Data augmentation,
- Image manipulation or
- Transfering properties.

For example generative models can be used to transfer body, head or facial properties from one image to another. One of these properties can be poses. For example the head/facial pose can be transfered from one to another person. For this, powerfull network architectures like DCGANs[2] can be used in different training settings.
One of these methods is the **Pose Guided Person Image Generation** from Liqian Ma et al.[1] They use a double staged training of three networks. While the first stage only produces a coarse manipulated output image, the second stage has the task to transfer details into the generated image by using a GAN like training with a generator and discriminator.
Here we want to transfer this method to head/face pose manipulation. Transfering this can be a challenging task due to pose embeddings, masking and other problems which need to be tackled.
In section 2 the basics for the paper and this report are presented. Afterwards the work from Liqian Ma et al [1] is explained in section 3. Finally the experiments and an evaluation is shown in section 4, where additional improvements beside the paper are explained and chosen. At the end of the section the results are evaluated with two different metrics on the test set.

2 Basics

Before we begin with the main method, we present a basic network architecture and the GAN learning setting. Neural networks, especially convolutional neural networks, and how they are learned can be read at `http://neuralnetworksanddeeplearning.com/`.

2.1 U-Network

The U-network is a general convolutional network architecture created by the Freiburg university Germany.[3] The network has two stages

1. Downsampling and
2. Upsamling.

While the downsampling is done by convolutional and max pooling layers, the upsampling can be arbitrary. For example interpolation or deconvolution (=transpposed convolution). Important is, that we increase the channel size while decreasing the image width and height (Downsampling). In contrast we increase the width and height while decreasing channel size (=Upsampling). Each of these two stages consists of blocks. Each block consists out of three convolutional layers, which behave differently while down or upsampling.

- Downsampling
  Keep height and width almost constant (maybe padding=0 reduces size) and increase the filter/channel size. Downsampling of height and width is done by max-pooling.
- Upsampling
  Reduce the channel size within the first convolutional layer (of the three) and increase the image size between the different upsampling blocks with interpolation or deconvolution.

As we can see in figure 1 the U-Net is named after its shape and has similarities to an autoencoder structure.

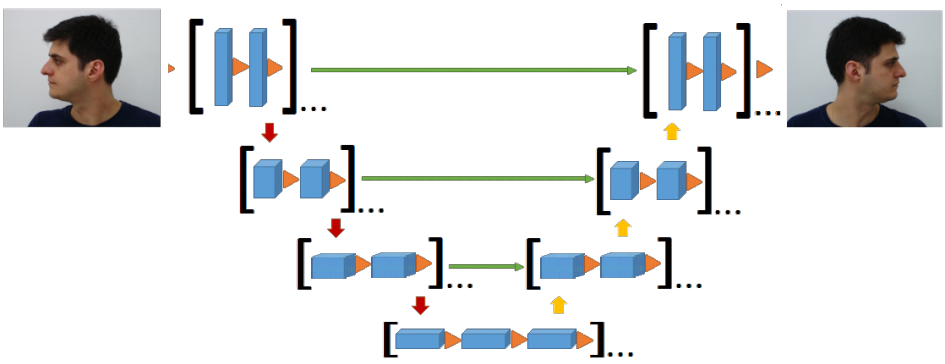

Figure 1: Showing a U-Network and its architecture in U-shape. Source: [3]

In addition to the architectural shape, we have skip connections between down- and upsampling blocks, which are illustrated in figure 1 as green arrows. Generally the ReLU function is applied after each convolution.[3]

2.2 Generative Adverserial Networks

The generative adversarial network(=GAN) setting is a training setting to concurrently learn two networks:

- Generator and
- Discriminator.

The **generator** tries to create realistic images from some input z, trying to estimate a distribution p_{real} with its own distribution p_{fake}. In other words, it tries to minimize the KL-divergence between p_{real} and p_{fake}.
The **discriminator** tries to distinguish between between fake (outputs 0) and real images (outputs 1). For this, the discriminator gets real or fake images as input, but does not know which is real or fake. The network now needs to decide if it's a real-world image or not. In other words, it tries to find a function f that distinguishes between p_{real} and p_{fake}.[4]

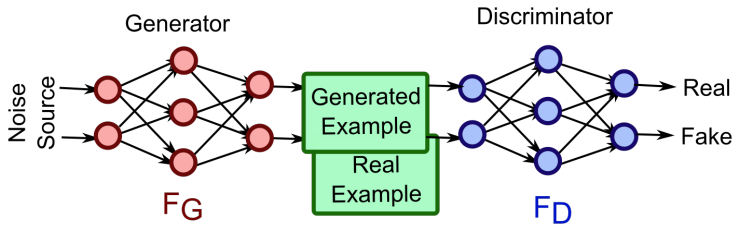

Figure 2: The GAN setting showing the generator and discriminator trying to trick each other. Source: [5]

The loss function can vary from task to task and are explained in section 3.

3 Pose Guided Person Image Generation

3.1 Pose embedding

The pose embedding is fairly simple and used as input for one of the neural networks. The pose is embedded by extracting keypoints and drawing a heatmap for each keypoint. For example a 68 facial keypoint detector on 400x500 pixel images has a 68x400x500 embedding. Each keypoint is encoded as a dot in a matrix with the value of 1 (0 for the background). This embedding can be seen in figure 3 projected into 2D space.

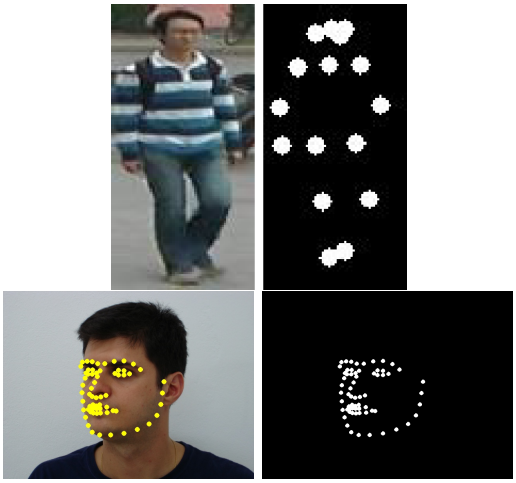

Figure 3: Pose embeddings for bodies transfered to faces.Source: [1][6]

[1]

3.2 Masking

Besides the pose embedding as input, a masking is applied in the later loss functions. The mask is an image/matrix which can be either 0 or 1 in each pixel. The mask depends on keypoints or can have other shapes. For example we can extract keypoints from our image, connect them meaningfully and fill the polygon. Examples for body and facial keypoints can be seen in figure 4

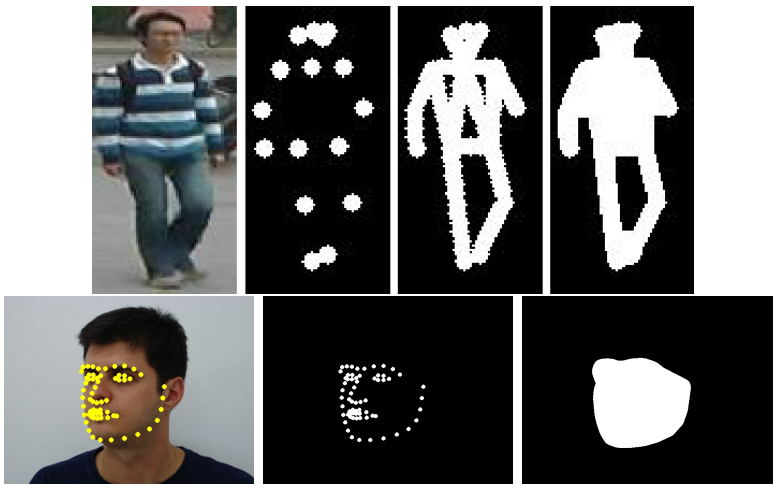

Figure 4: Top row shows the body mask generation process. Bottom row shows the facial analog mask generation process. Source: [1][6]

While the body mask is taken by Liqian Ma et al[1], the facial masking process is derived from their work.[1]

3.3 Deep convolutional GAN

Due to the generator networks are both sligthly adapted U-Networks, the differences to the classic U-Net from subsection 2.1 will be listed here.

- The middle layer **can** be a fully connected layer or convolutional.
- The max-pooling downsampling is replaced by a convolution with stride two.

Additionally some parameters are set.

- The amount of down-/upsampling blocks depends on the input image size.
- Usage of 3x3 filters.
- Number of filter increase/decrease **linearly** with each block.
- No ReLu for fully connected or the final convolutional layer.

The architecture is strongly adapted from DCGANs.[2][1]

3.4 Different Networks

The main method has three different networks:

- Generator 1 (=G1)
  A DCGAN with N downsampling/upsampling blocks. With a fully connected layer as middle layer.
- Generator 2 (=G2)
  A DCGAN with N-2 downsampling/upsampling blocks. With a convolutional layer as middle layer. G2 can have less downsampling layers because it does not use a FC layer as middle layer, which reduces the memory usage drastically.
- Discriminator (=D)
  A fully convolutional network with ReLu activation functions after each convolutional block, with a final fully connected layer.

We can see the overall architecture in figure 5. The explicit training, inputs, outputs and loss functions are explained in section 3.5.[1]

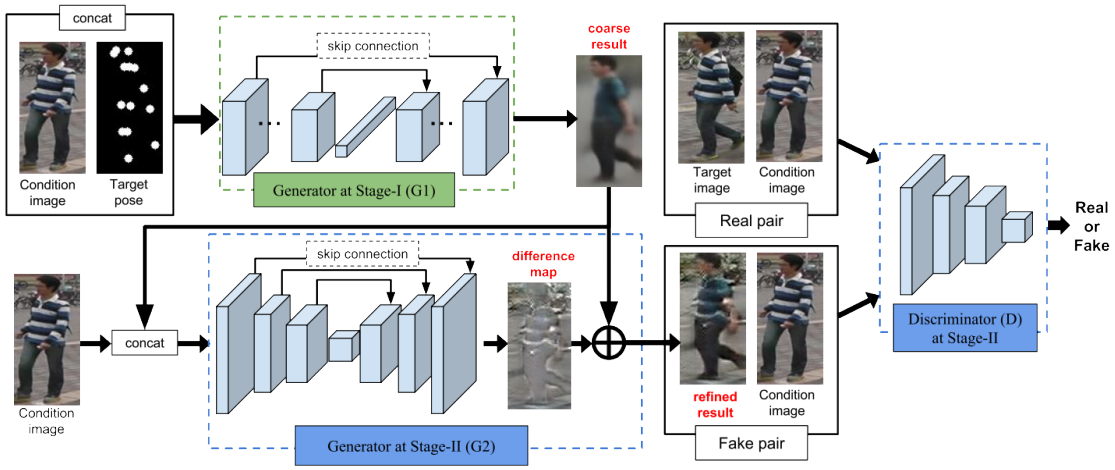

Figure 5: Architectural overview of the networks and its connections.Source: [1]

3.5 Training

The training is performed in two stages:

- **Stage 1**
  In stage 1, a first coarse manipulated image is generated. Mainly the task here is, to change the pose without many details in the generated image. For this, only **generator 1 (=G1)** is trained. G1 gets a source person image as input (RGB) in combination with the target pose embedding. Then the L1 loss is calculated between the target image and generated image from G1. The L1 loss also contains a masking process (mask as described), so it can focus on the facial area by multiplying with a matrix of ones (=where no face mask is) and twos (=where the face mask is). The loss (inclusive masking) can be calculated with L_{G1}.
  - Goal: Try to generate coarse image with new pose
  - Input: Person (RGB source image), Landmarks heatmap(68 channel image of target pose)
  - Output: Coarse pose changed person (RGB image)
  - Loss: $L_{G1} = ||(G1(I_S, P_T) - I_T) \odot (1 + M_T)||_1$
    ,where I_S is the source image, P_T is the target pose, I_T is the target image and M_T is the mask with the dimension of the image.

- **Stage 2**
  In stage 2, the Discriminator (=D) and Generator 2 (=G2) proceed with a GAN-like training, while the weights of G1 are fixed.
  Generator 2 takes the source image and the generated image of G1 (=I_{G1}) as input

and tries to generate a difference map (RGB) which is added on top of I_{G1}. The loss is the binary cross entropy loss between the prediction of the discriminator and 1 (=real image). Additionally to this, we have a similar loss to L_{G1}, to focus onto the proper areas due to masking.

- Goal: Try to refine image of G1
- Input: Person (RGB source image) concatenated with the generated image of G1.
- Output: Difference map (RGB details image)
- Loss: $L_{G2} = L_{BCE}(D(I_S, I_{G2}), 1) + \lambda||(I_{G2} - I_T) \odot (1 + M_T)||_1$
  , where $D(I_S, I_{G2})$ is the discriminator prediction for the fake pair, λ is only a hyperparameter and $||(I_{G2} - I_T) \odot (1 + M_T)||_1$ is anologous to the loss for G1.

The **discriminator** tries to distinguish between real and fake images. The only difference to the normal GAN setting is that the discriminator gets input **pairs**. For a fake pair this is the final generated image (I_{G1} + difference image from G2) and the source image. For a real pair this is just the target and source image. The loss then is the binary cross entropy loss depending if D gets a real or a fake pair as input. Additionally to this, we have a similar loss to L_{G1}, to focus onto the proper areas due to masking.

- Goal: Tries to distinguish between real and fake pairs
- Input: Person (RGB image), Same person with target pose (RGB image)
- Output: 1 or 0
- Loss: $L_D = L_{BCE}(D(I_S, I_T), 1) + L_{BCE}(D(I_S, I_{G2}), 0)$
  ,where $D(I_S, I_{G2}$ is the prediction for a fake and $D(I_S, I_T)$ for a real pair.

[1]

4 Experiments & Evaluation

4.1 Dataset

Originally there were two body pose datasets[1]

- DeepFashion[7]
- Market1501[8]

Due to the body pose estimator did not work that well (too small images), I decided to transfer it directly to facial datasets before testing it on the original. For this, the **FEI Face Database**[6] was used. The dataset contains 2800 images of 200 individuals with 14 images each:

- 10 different angles
- 3 different illuminations
- 1 smiling face

The dataset was split into training(90% = 180 individuals) and test(10% = 20 individuals) dataset. For the pose manipulation only the images with different angles were used.

Figure 6: Some samples from the FEI dataset. Source: [6]

4.2 Additional Contributions

This whole section is about additional contributions (=from me) to the algorithm of Liqian Ma et al.

4.2.1 Masking

There are two possible maskings for the facial application of the pose manipulation.
First, the landmarks are detected by a dlib 68 landmark detector.[9] Then we can transfer the method from the paper to facial landmarks by just filling the polygon, which would look like the following.

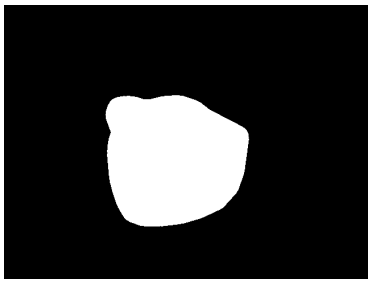

Figure 7: Face mask extracted from the facial landmarks.

Second, we can determine the bounding box with dlib[10] and compute the middle point p. Around this point p, we can draw an ellipse with width w and height h. The height and width are euqal to the height and width of the bounding box scaled by two scalar values. Finally we get the following mask.

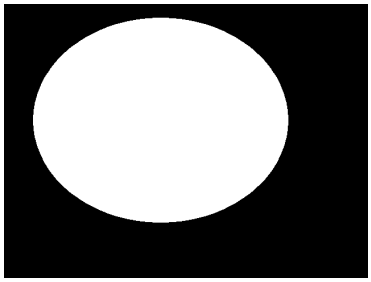

Figure 8: Ellipsoid mask extracted from the facial bounding box.

Both methods have advantages and disadvantages and were compared in combination with the upsampling method.

4.2.2 Upsampling

For upsampling (not closer specified in the paper) generally exit two algorithms:

- Deconvolution (Transposed convolution)
- Interpolation

Both of them were tested in combination of the masking type.

Figure 9: From left top to right bottom (Masking & Upsampling method): Ellipse & Deconvolution, Ellipse & Interpolation, Keypoints & Deconvolution, Keypoints & Interpolation.

Finally we decided to continue with the ellipse masking and interpolation method (=top right), because it showed the best results in many samples.

4.2.3 Source Keypoint Input

G1 often generated images with the unrotated face of the original image.

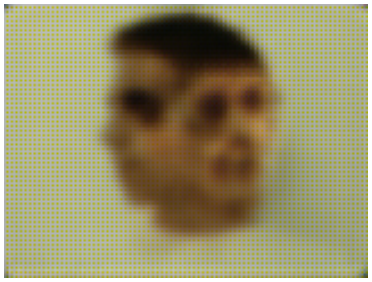

Figure 10: Example of source image pose manipulation failure. G1 does "not know" it should manipulate the source keypoint positions. It first would need to learn a landmark detection.

Due to these artifacts the decision was made to additionally append the source image keypoints to the input of G1.
This increased the quality and the learning rate of G1 drastically, because G1 also would need to learn a facial keypoint detector for the source image. Providing this information removes that task for G1 and provides easily accessible information that it needs to manipulate the source keypoint locations also.

4.2.4 Discriminator Cutoff

Due to G1 or G2 often made the border pixels of the final image worse, n pixels are cut off from the final fake image (input for D).
On the one hand this decision was made, because the discriminator often focused onto the differently colored border pixels due to the padding of the convolution of G2 or G1.
On the other hand G2 sometimes focused too much on correcting the failure at the borders of G1 (due to padding of the convolution), which is why G2 could not focuse onto the main objective, which was to refine the coarse image from G1. An example of this wrong focus is

Figure 11: Example for the wrong focus of G2. We can see a difference image (=output of G2). Maybe you need to zoom in to see the redish borders.

We can see, that the border pixels are almost all red, which means a manipulation of the border mainly.

Due to these two (convergence) problems, the borders were cut off.

4.3 Evaluation Metrics

For the overall evaluation, we chose the Structural similarity and RGB-l1 distance.

4.3.1 Structural Similarity

The structural similarity(=SSIM) is a product of three properties from two greyscale images x and y.

- Luminance (Mean)

$$l(x,y) = \frac{2\mu_x\mu_y}{\mu_x^2 + \mu_y^2}$$

  ,where μ_a is the mean of image a.

- Contrast (Variance)

$$c(x,y) = \frac{2\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2}$$

  , where σ_a is the variance of image a.

- Structure (Combination of both)

$$s(x,y) = \frac{\sigma_{xy}}{\sigma_x\sigma_y}$$

  , where σ_{xy} is the covariance.

If we multiply those three factors, we get the SSIM.

$$\text{SSIM}(x,y) = \frac{4\mu_x\mu_y\sigma_{xy}}{(\mu_x^2 + \mu_y^2)(\sigma_x^2 + \sigma_y^2)}$$

A value of 0 means no SSIM, while 1 means comparing two identical images.[11]

4.3.2 RGB Loss

This loss simply is the L1-distance from the generated to the source image.

$$L_{RGB} = ||I_{G2} - I_T||_1$$

, where I_{G2} is the final generated image and I_T is the original target image.

4.4 Final Results

The final results can be seen in the following and are from 50 epochs of training at each stage of training:

Figure 12: Final result images from left to right: Source image, target pose image, image generated from G1, difference image generated from G2, image from G1 + difference image (=final generated image).Source: [6]

We can clearly see, that G1 produces a good coarse pose manipulated image. It has not that many details, but this is not the objective of G1. G2 then generates the difference image to bring in some details. We can see, that G2 tries to put in some details, but still does not have that much effect. This may be because of the short training time due to time and hardware limitations.

Besides the visual inspection of the generated results, there are also SSIM and RGB Loss. The RGB loss is divided by #channels · #pixels. The following values are for the above images.

SSIM(I_S, I_T)	SSIM(I_T, I_{G1})	SSIM(I_T, I_{G2})	RGB-L1-Distance(I_T, I_{G2})
0.6166	0.7917	0.7904	0.0367
0.6278	0.7733	0.7723	0.0410
0.6503	0.8270	0.8266	0.0322
0.6004	0.7816	0.7810	0.0341
0.6833	0.8074	0.8079	0.0310
0.4707	0.7811	0.7761	0.0375
0.6874	0.8346	0.8325	0.0318

Table 1: I_S is the source image, I_T is the target image, I_{G1} is the image from G1, I_{G2} is the image from G1 + difference image from G2

Additionally the SSIM and RGB-loss were evaluated for the whole test set and averaged.

SSIM(I_S, I_T)	SSIM(I_T, I_{G1})	SSIM(I_T, I_{G2})	RGB-L1-Distance(I_T, I_{G2})
0.7012	0.8207	0.8192	0.0326

Table 2: Averages for the testing dataset.

We can see, that there is a general pose change by SSIM(I_S, I_T)<1.0 between source and target in the original images. Due to SSIM(I_T, I_{G1}) & SSIM(I_T, I_{G2}) > SSIM(I_S, I_T) it shows, that the algorithm is manipulating the pose.
The results decrease with applying G2, but only a very small amount.As mentioned, this may be because of the short training time due to time and hardware limitations. Overall, this verifies the results from the visual inspection and table 1. The RGB-L1 distance is in the interval [0,1] , so there would be a 6 % error per pixel, if we assume the average is 0.5 per pixel per channel. This seems to be pretty low.
Additionally, an indicator for G2 and D improving is the up and down for the training loss for G2+D, which can be seen in figure 13.

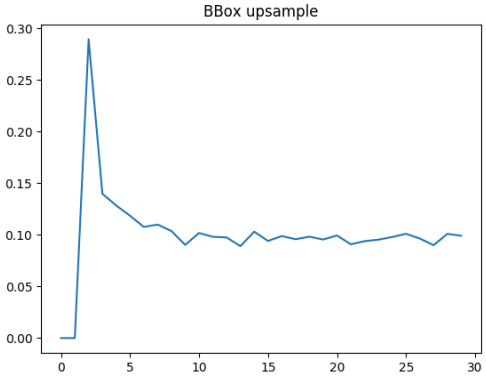

Figure 13: Training loss of G2+D of 50 epochs.

5 Conclusion

In conclusion the transfer of the algorithm to facial points worked well. Even if the dataset is very small, G1 performed very well. The masking and upsampling method were compared due to the transfer from body to facial poses. It showed, that upsampling with interpolation and the ellipsoid masking method worked best.
G2 did not improve the results that much. Still, it is shown that it produces the wanted output in the GAN setting by taking a look at the difference images. The visual inspection results were confirmed by the individual SSIM and RGB-loss and with the average SSIM and RGB-loss.
The theory that the results may improve with more training epochs has been substantiated by taking a look at loss trend of G2+D in figure 13.
While the algorithm works, there are still improvements to make like testing an alternative input for target poses, increasing the speed of learning for G2, increasing the dataset size or even transfering this process to arbitrary poses/structural inputs.

6 Problems

- Pose estimator for body pose – Too small images in market 1501 dataset.
- Size of input images – Too many weights for FC layer.
- Generator 2 diverged wrongly – There now is a small pretraining of discriminator to tackle that.
- G2 does not improve the image that much. Maybe there needs to be a longer training time. GAN settings can take long to converge.

7 Further Improvements

- Increase the training size.
- Transfer to other poses.
- Try to give contour image (from edge detector) instead of target pose heatmap as input $\implies$ Arbitrary input manipulations.

References

[1] L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. V. Gool, "Pose guided person image generation," *CoRR*, vol. abs/1705.09368, 2017.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." `https://arxiv.org/abs/1511.06434`, 2016. [Online; accessed 04-August-2019].

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[4] Ian J. Goodfellow and Jean Pouget-Abadie and Mehdi Mirza and Bing Xu and David Warde-Farley and Sherjil Ozair and Aaron Courville and Yoshua Bengio, "Generative adversarial networks," 2014. [Online; accessed 4-August-2019].

[5] Unknown. `https://www.araya.org/2018wp/wp-content/uploads/2016/10/gan.png`. [Online; accessed 04-August-2019].

[6] B. Artificial Intelligence Laboratory of FEI in São Bernardo do Campo, São Paulo, " FEI Face Database ." `https://fei.edu.br/~cet/facedatabase.html`, 2006. [Online; accessed 04-August-2019].

[7] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[9] Adrian Rosebrock, "The landmarks are detected by a dlib 68 landmark detector.," 2017. [Online; accessed 4-August-2019].

[10] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[11] Wikipedia contributors, "Structural similarity," 2019. [Online; accessed 4-August-2019].