Keep Smiling

Generative Adversarial Networks for Face Manipulation

Project Report

Johannes Grimm and Emanuel Maus

Heidelberg University
Object Recognition and Image Understanding

Contents

1 Introduction (Johannes Grimm and Emanuel Maus) 1

2 Dataset and Dataloader (Emanuel Maus) 3

3 Generative Adversarial Networks 7
3.1 Simple GAN (Johannes Grimm) 7
3.1.1 Method 7
3.1.2 Results 8
3.2 Complex GAN (Emanuel Maus) 9
3.2.1 Method 9
3.2.2 Results 9
3.3 Split GAN (Johannes Grimm) 10
3.3.1 Method 10
3.3.2 Results 10

4 Conclusion and Outlook (Johannes Grimm and Emanuel Maus) 13

Bibliography 15

Chapter 1

Introduction (Johannes Grimm and Emanuel Maus)

It often happens during taking group photos that one or more people don't smile, which ruins the photo. In such situations you wish for an application that puts a smile on those faces afterwards. Similar functions are also available in Snapchat and co., which shows that this is a current topic and that many companies are dealing with it.
As we know, no two faces are alike. That is the difficulty of our project. The metric of the face is unique and yet certain facial expressions can be recognized. Therefore several Generative Adversarial Networks (GAN) are used to master this task. Furthermore we operate with one GAN in feature space to switch between non-smile and smile state. Because it is necessary to work resource saving, we included a cropper for mouth in the Dataloader. The mouth is used because the main part of smiling in our face is the mouth region. As already mentioned, faces vary greatly, making a lot of training data necessary. Figure 1.1 shows the goal of our GAN (both are original images).

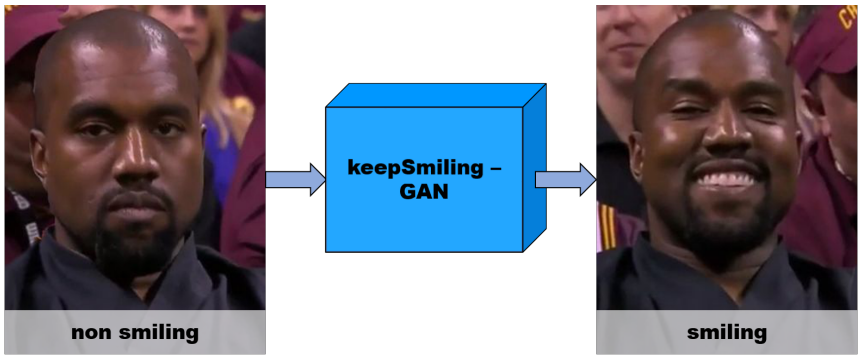

Figure 1.1: Goal of our project.[5]

Chapter 2

Dataset and Dataloader (Emanuel Maus)

The data set "Facial Landmark Detection by Deep Multi-task Learning"[4] consists of about 7500 training images and 1700 testing images (about 1000 smiling and 700 non-smiling). It contains images of faces from frontal view, which smile and do not smile, and was equipped with 5 pre-determined coordinates, which marked the position of certain face regions. This includes the positions of both eyes, the nose and the left corner of the mouth, as well as the right corner of the mouth. These coordinates and the face pictures were the starting point for our dataloader. By means of the training data the different GAN systems were trained and with the help of the test data the feature space of the Split GAN was sampled and its smile generation capability was tested. However, it turned out that the test images in particular were not optimal for our task, as they varied greatly in position, form and type and were also partially wrongly declared. Some faces had been described as smiling, whereas in reality they did not smile. A better data set is described in the summary. Since this project was limited in time, it was necessary to show a certain degree of efficiency so that the GAN systems could quickly learn how to generate the smiling faces. Therefore, one limited oneself only to the most striking characteristic of a smiling face, the smiling mouth. In order to extract the complete mouth from the facial images as far as possible, its natural form was used, an ellipse. The coordinates of both corners of the mouth (in Figure 2.1: right (2), left (3)) and the tip of the nose (in Figure 2.1: (1)) were the starting point for fitting the ellipse. The two corners of the mouth were the focal points of the ellipse and the point on the connecting line of the focal points, which had the smallest distance to the tip of the nose, was the centre of the ellipse (in Figure 2.1: (4)). This ensured that the ellipse went through the point of the tip of the nose. Since the GAN systems work more easily with rectangular images, the enveloping rectangle was determined with the help of these three points, which were taken from the dataset.

For Pillow's own crop function, the left, upper corner P_t and right, lower corner P_b are required, which were calculated as follows:

$$\vec{P}_1 \triangleq \textit{coordinate of the nose}$$
$$\vec{P}_2 \triangleq \textit{coordinate of the left mouth corner}$$
$$\vec{P}_3 \triangleq \textit{coordinate of the right mouth corner}$$
$$\vec{P}_4 \triangleq \textit{coordinate of the middle of the ellipse}$$

$$\vec{P}_4 = \left(\frac{||\vec{P}_3 - \vec{P}_2||}{2}\right) \cdot \left(\frac{\vec{P}_3 - \vec{P}_2}{||\vec{P}_3 - \vec{P}_2||}\right) + \vec{P}_2$$
$$\vec{b} = \vec{P}_4 - \vec{P}_1$$
$$\vec{e_a} = \vec{P}_3 - \vec{P}_2$$
$$a = \left(\vec{e_a}^2 + \vec{b}^2\right)^{0.5}$$
$$\vec{a} = \left(\frac{\vec{P}_3 - \vec{P}_4}{||\vec{P}_3 - \vec{P}_4||}\right) \cdot a$$
$$\vec{P}_t = \vec{P}_4 - \vec{b} - \vec{a}$$
$$\vec{P}_b = \vec{P}_4 + \vec{b} + \vec{a}$$

The cut image of the laughing or non-laughing mouth is then rescaled to 64x64 pixels so that all the cut-out mouth images are in the same size. The 64x64 pixel size is an empirically determined size, which results from the average size of the mouths. The complete sequence of this process can be seen in the Figure 2.1. Furthermore, the cropped images are processed using a randomized horizontal flip to ensure greater variability. Before the image with its associated label, which indicates whether the face laughs or not, is transferred to the GAN system via the dataloader, an intensity normalization of the cropped image is performed. A big criticism of this method of cropping is that the size of the region cut out differs greatly between the laughing face and the non-laughing face. The reason for this is that the corners of the mouth lie very close to each other when not laughing and the face is very narrow. This makes the enveloping rectangle so large that it extends over the face. When laughing, however, the corners of the mouth are usually very far away from each other and the face also becomes wider. As a result, this phenomenon does not occur here. This problem could probably be circumvented by using a different type of cut. A diamond, whose corner points are placed in the corners of the mouth and nose points, would be more suitable for this. The fourth

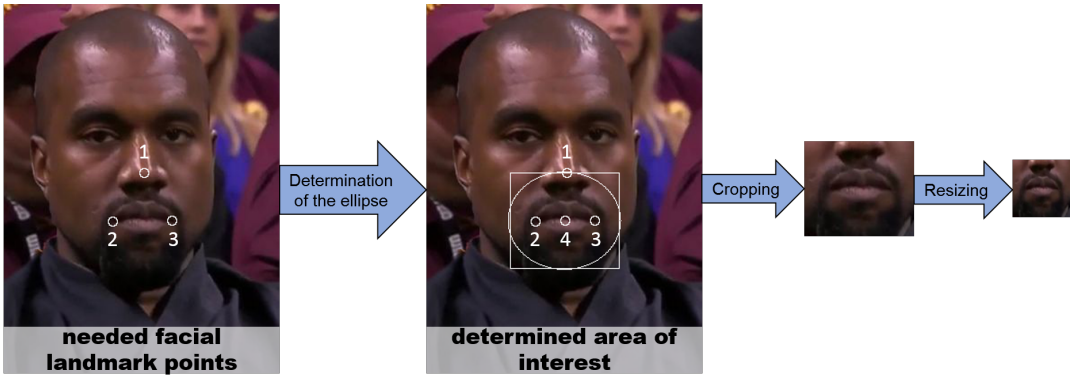

Figure 2.1: Cropping procedure of the Dataloader.[5]

Figure 2.2: Example for non-smiling cropped region.

Figure 2.3: Example for smiling cropped region.

corner point would be approximately in the region of the chin. The problem with the cropper function can be seen in the Figure 2.2 and Figure 2.3. Another problem occurs when the generated laughing mouth is inserted back into the original image of the non-smiling face. Since not only the mouth region changes during laughter, but also the region around the eyes, or in some cases, the light reflection in the cheek regions changes, the original image with the inserted laughter appears very artificial and unnatural. This is clearly visible in Figure 2.4. The original smile of the person was used, which means that this would be the result if the GAN system provided an optimal result. This unnatural way of smiling could be avoided if the GAN systems were trained with the whole face, which of course would mean more computer resources.

Figure 2.4: best possible output with the method used.[5]

Chapter 3

Generative Adversarial Networks

Three different GANs are used to produce smiling mouths.

3.1 Simple GAN (Johannes Grimm)

3.1.1 Method

On the basis of the simple GAN also the concept of GANs are explained. In general, a GAN consists of two neural networks, a generator, which generates new data instances and a discriminator, which evaluates this instances for authenticity. The exact steps are shown in Figure 3.1. The generator gets as an input a d-dimensional noise vector. Afterwards it constructs an image, which is given to the discriminator. The discriminator tries to predict if it is an image of the training set or not. Both networks try to optimize their loss functions (BCE-loss: Equation 3.1) in dependence of each other.

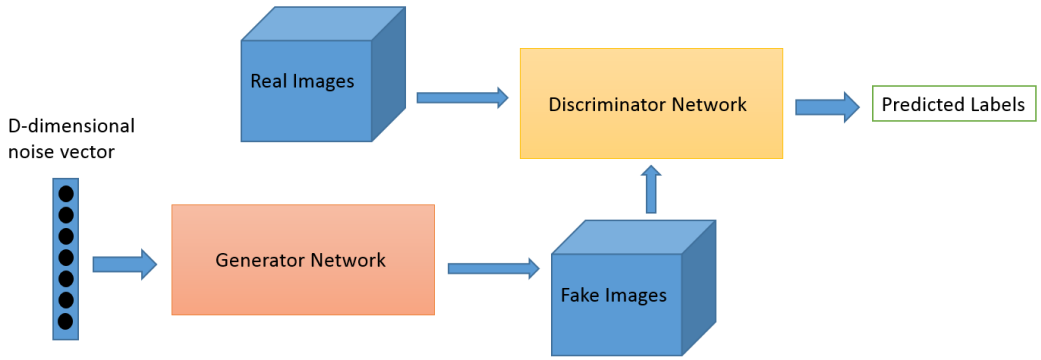

Figure 3.1: Overview of a simple GAN.[3]

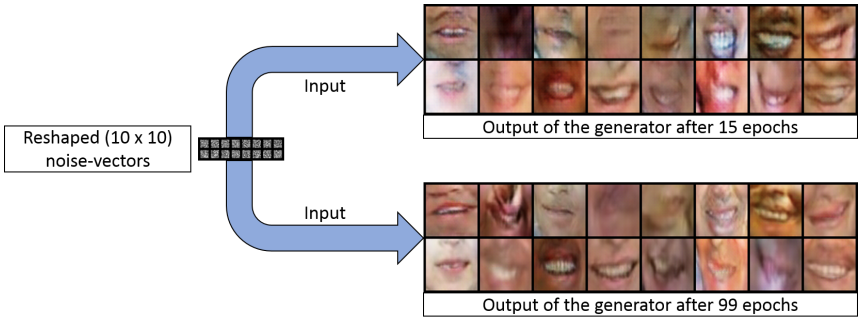

Figure 3.2: Output of the Generator after 15 and 99 epochs.

$$L = \{l_1, .., l_N\}^T, l_i = -w_i[y_i \cdot \log(v_i) + (1 - y) \cdot \log(1 - v_i)] \tag{3.1}$$

3.1.2 Results

For this method we used a 100 dimensional noise vector as an input of the generator. Also just smiling images were taken for training to see how a GAN works in general. Figure 3.2 shows the output of the generator after different epochs. It points out that already after 15 epochs the generator creates some smiling mouths out of the noisy input vector. In most cases you can see the lips and the teeth. If the GAN trains longer the generator produces more detail output images.

3.2 Complex GAN (Emanuel Maus)

3.2.1 Method

This model is more complex than before. The Generator starts from a real image, then goes to the feature space and afterwards generates a resulting image depending on the target classification label (smile, not smile). This adds a further loss function (L1 norm between input image and generated image ($g\_loss_{reconst}$)) to achieve that the generated image does not differ too much from the input image. Another loss function, which is applied here, is the BCE-loss for the class-label between the predicted class label and the input label of the generator. Also the discriminator not only distinguishes between real and fake images (d_loss_{real} and d_loss_{fake}) but also between smiling and not smiling. This adds also the class-label loss (d_loss_{cls}). Furthermore a loss for gradient penalty was used (d_loss_{gp}). It is calculated via:

$$d\_loss_{gp} = gradient\_penalty(output\_discriminator\_of\_x_{hat}, x_{hat}) \tag{3.2}$$

with

$$x_{hat} = \alpha \cdot x_{real} + (1 - \alpha) \cdot x_{fake} \tag{3.3}$$

,where α is a random number vector and

$$gradient\_penalty(y, x) = \left(L2\_norm \left(\frac{dy}{dx} \right) - 1 \right)^2 \tag{3.4}$$

To sum the losses up and show how they are weighted in respect to each other the following equation shows the total loss functions.

$$discriminator\_loss = d\_loss_{real} + d\_loss_{fake} + d\_loss_{cls} + 10 \cdot d\_loss_{gp} \tag{3.5}$$

$$generator\_loss = g\_loss_{fake} + g\_loss_{cls} + 10 \cdot g\_loss_{reconst} \tag{3.6}$$

3.2.2 Results

For this method the whole training image set was used. Figure 3.3 shows the result of this GAN model. You can see that the generated images are still very noisy after 50 epochs. One reason for this is the model complexity. A comparable model[1] trained 24 hours on a 24 gb gpu, so more learning time is necessary. An other points could be the used dataset. It provides outliers as Figure 3.3 shows. Therefore the dataset has to be further investigated and the training set has to be more precisely preselected. Furthermore the different losses have to be harmonized to each other to obtain good results.

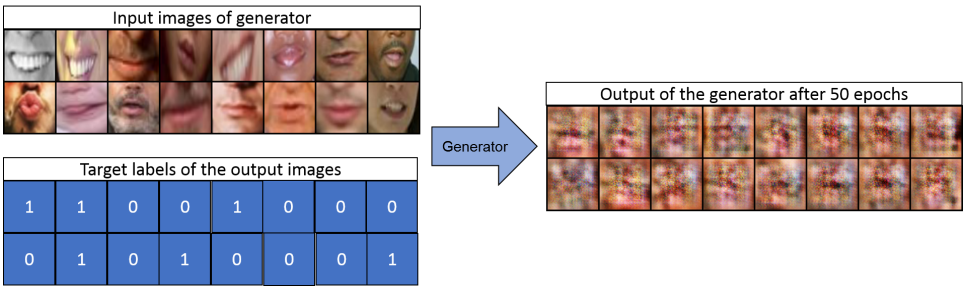

Figure 3.3: Input and output of the Generator after 50 epochs.

3.3 Split GAN (Johannes Grimm)

3.3.1 Method

The Decoder (Generator) from the simple GAN is extended with a separate Encoder to allow mathematical operations in the feature space. The purpose of this procedure is, that we hope that the smiling and non-smiling regime is Gaussian distributed in feature space and therefore can be separated properly. This adds similar to the complex can more loss functions. For the discriminator the gradient loss is added and for the Generator the loss between reconstructed and input image is added. In the following equations the total loss functions are shown.

$$discriminator\_loss = d\_loss_{real} + d\_loss_{fake} + 3 \cdot d\_loss_{gp} \tag{3.7}$$

$$generator\_loss = g\_loss_{fake} + 9 \cdot g\_loss_{reconst} \tag{3.8}$$

In the feature space, the difference of the mean smile and mean non-smile vector represents the shift vector from non-smiling to smiling and vice versa. Besides this modification, the implementation is analogous to the Simple GAN.

3.3.2 Results

Figure 3.4 shows the output of the generator after 199 epochs. With this trained GAN the feature vectors of all test images can now be determined. Afterwards the mean vector for smiling and non-smiling was calculated. The shift vector is therefore the difference between these mean vectors. Figure 3.5 shows the reconstructed images to which the shift vector has been added (non-smile to smile) or subtracted (smile to non-smile) in feature space. The org-column shows the original images and the gen-column the generated ones. On the left side is the translation from smile to non-smile and on the right side vice versa. The left ones show that the generator mainly closes the mouth and puts the corner of the mouths down, but you can also see that the generated mouths look more or less the same. For the last image (Kayne) you can also see that he closes his mouth an have still wide lips but

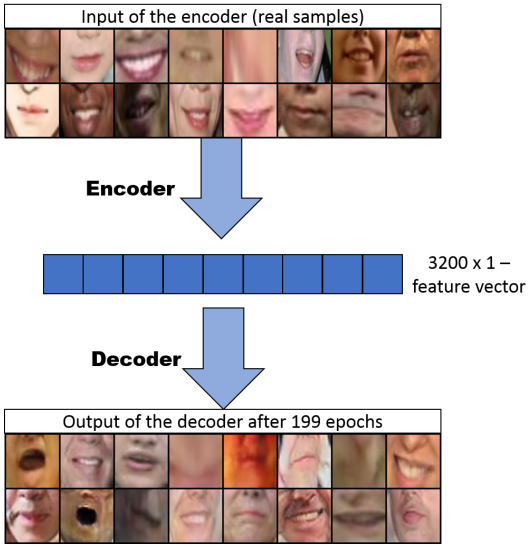

Figure 3.4: input and output of Split GAN.

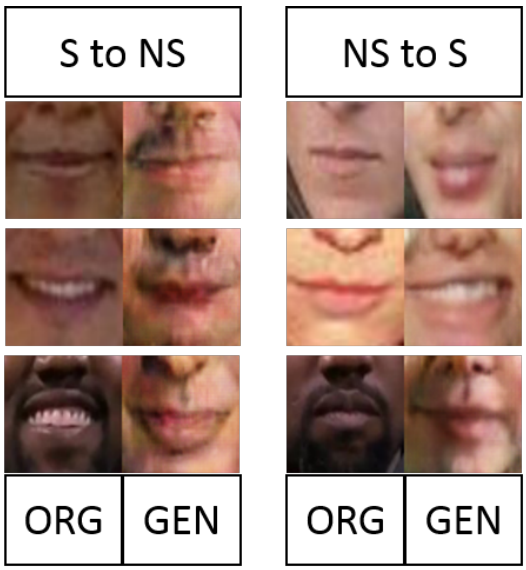

Figure 3.5: Original and reconstructed images of the Split GAN.[5]

label	length(std)	#vectors in other conf. band	# vectors in own conf. band
smile	36.7	48%	54%
non-smile	36.2	50%	59%

Table 3.1: Informations about the feature vectors.

he changed his color. The right ones show that the main goal of the generator is to open the mouth and show teeth for a smiling face . The second image pair shows this really good. For the Kayne image you can see that it leaves the mouth closed but puts the mouth corner higher. Like before he changed again his color. This is due to the reason that our dataset provides much more light-skinned people. So to keep the color the dataset has to provide much more skin colors. A problem with this translation method is that the main informations about the original face get lost. This means for further investigation we should apply another loss function that minimizes the loss between the input image and the translated image like method 2 and give the generator also informations about the desired label. Therefore we should also train our model with such a translation vector.
To look how far the smile and non-smile region are separated in feature space, we take a closer look at the standard deviations of the mean smile-/ non-smile vectors. Table 3.1 shows standard deviation of the mean vectors and the count of smile vectors in the 1 sigma interval of the mean non-smile vector and vice versa. You can see that standard deviation is around 36.5 for both mean vectors. Compared to the shift vector, which is around 8.4, it is much larger. This is also supported by the last two columns of the table. For smile 48% of the smile vectors are in the confident band of the non-smile vectors and around 50% of the non-smile vectors are in the smile region. On the other hand 54% of smile vectors are in the 1-σ interval around their own center and for the non-smile vectors there are 59%. This proves that the two regions overlap and are not discretely separated as hypothesized. To improve this the feature space should be further reduced and the dataset should provide images where you have smile and a non-smile faces of the same person.

Chapter 4

Conclusion and Outlook (Johannes Grimm and Emanuel Maus)

In the dataloader we took the dataset and cropped it to reduce the image size. Due to the smaller image size the computation time was minimized. Therefore we could train several GANs and optimize the parameters or the GAN system. One disadvantage of the used crop method is that when we crop smiling faces by using ellipses the area of interest is more broader than the cropped area of a non-smiling face. Another point is that while smiling not only the mouth region is changing compared to the non-smiling face, but also i.e the eye region. To improve the dataloader we could use a better cropping method (diamond) or taking the whole faces into account. This would result in more computing time or the usage of more computing power.
The simple GAN produced very fast good results out of noisy input vectors. After a few epochs the generator learned, that there is a mouth opening in the middle of the image. Several epochs later the structure of the nose and its location is learned. The more epochs the generator ran, the more facial details it learned.
The Complex GAN was developed with the intention of producing desired output images. To control the output of the generator a target label vector was given. This vector determines, whether the input image should switch to a smiling or non-smiling state. After some training time we could not recognize any progress in the generated output images. This could be repaired by adjusting the parameters (different weights of the loss functions). Also more learning time could be necessary because of complexity of the model.
The split GAN is constructed similar to the complex GAN, but without controlling it with the target class labels. By sampling the feature space with the test images the mean location of smiling- and non-smiling-feature vectors were determined. With the help of them the mean translation vector was calculated and applied to the test images, so that the smiling faces changed to non-smiling faces and vice

versa. Overall this transformation was successful, but in general the face characteristic changed a lot. For example the skin got always the same color after shifting. After investigating the feature space we found out that the distance between the center of the two regimes is small in comparison to the standard deviation. This shows that the assumption that they are Gaussian distributed and separable is not correct. To guarantee the separability of smiling images and non-smiling images in the feature space further information has to be provided to the generator. Therefore the connection between feature space vectors of smiling or non-smiling faces and the states of the output images of the decoder (whether the output image is smiling or not) have to be investigated and taken into account. Another point is that our feature space is pretty large (3200 dimensions). To examine the feature space better and to classify the smile and non-smile regime it could be an advantage to reduce the feature space by using more convolution layers or via principle component analysis. Also a dataset[2] which provides smiling and non-smiling images of one person could help to train the GAN better.

Bibliography

[1] Choi et al. "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation". In: ().

[2] Gevers et al. *UvA-NEMO Smile Database*. `http://www.uva-nemo.org/index.html`.

[3] Xu et al. "An empirical study on evaluation metrics of generative adversarial networks". In: ().

[4] Zhang et al. "Facial Landmark Detection by Deep Multi-task Learning, in Proceedings of European Conference on Computer Vision". In: ().

[5] *Edward Roberts. Kayne West Image*. `https://www.mirror.co.uk`.