Facial Animation Controller Using Generative Adversarial Networks

Utrecht University, The Netherlands
Lennert Sietsma*
Supervised by: dr. Zerrin Yumak and **prof. dr. Remco Veltkamp**

Abstract

We introduce a new method to create a facial animation controller. We find a high-level control-space bottom-up from data using the generative part of a Wasserstein Generative Adversarial Network (WGAN). By training a WGAN on face tracking data from the IEMOCAP corpus, we show that a WGAN is able to learn the behavior of the human face. By training the WGAN on different emotions, we show that the WGAN is successful at learning human face movement matching the emotions that it was trained on. We also analyse the behavior of the latent space. We found that the generator provides control over certain aspects of the face and sometimes even relates to emotions. By implementing sliders for the latent space variables we were able to create a facial animation controller using the generative part of the WGAN.

***Keywords*—** Facial Animation, Generative Adversarial Networks

1 Introduction

Computer animation is a popular subject and widely applied in both film and games, but also in more piratical applications such as virtual assistants. The animations of humans is a common practices when it comes to computer animation. The difficult part about animating a human is that we as humans are naturally very good at identifying natural behavior. The animation of the faces of humans is especially difficult as humans transfer emotional information with their expressions. To create a good facial expression, the animator has to be really careful to set the expression with the desired emotion and not create an expression that is unnatural.

There are different methods when it comes to animating a human face. Some methods require the animator to set all the vertices of a 3D-face individually. Methods such as this one are referred to as low-level controllers. These low-level controllers allow for a very high precision when it comes to setting an expressions. The downside of low-level controllers is that they are very time consuming and require an expert hand to make the expression look natural. There are also methods that supply the animator with tools that are able to control certain aspects of the face such as the mouth and eyebrows or sometimes even emotions. These methods are often referred to as high-level controllers. These high-level controllers can be created in two different ways: top-down and bottom-up. When we talk about top-down approaches, we talk about controllers that linked a certain variable to certain aspects of the face. For example, we can have a variable that is linked to the height of the eyebrows. By changing the variable, we effectively raise and lower the vertices that belong to the eyebrow. These top-down approaches can for example use control spaces based on emotional models [2, 32] or appraisal theory [1]. The bottom-up based controllers are created by analyzing the data and creating control variables based on this analysis. To find this high level control space we can use dimensionality reduction techniques such as IsoMap [3] and Local Linear Embedding [4]. In 2010 Stoiber used Principal Component Analysis (PCA) to derive features based on real-life data [31] to create such a bottom-up facial animation controller. Controllers such as this one will also give the animator control over certain aspects of the human face. The advantage of the bottom-up approach is that the limitations of the human face are learned from the data. This means that the expressions created by a bottom-up approach will overall look more natural compared to expression created by a top-down approach.

Since 2010 the field of machine learning has developed new methods that can possibly improve the system that Stoiber created. For this research we will use Generative Adversarial Networks (GANs) to create a facial animation controller bottom-up from real-life data. GANs have already proven their capability to learn human body behavior [27], human speech [28] and human face shapes [34]. Compared to other research around GANs this is, to the best of our knowledge, the first time someone tries to implement a GAN as a facial animation controller. Since GANs are commonly used for the analyses and generation of image data, we will have to make some changes to be able to make the GAN learn from face tracking data and generate facial expressions.

The objective of this research is to find out if GANs can be used as a facial animation controller. To make for a good facial animation controller, we test different hyperparameter settings for the GAN. The goal of this is to find an intuitive mapping between the input of the generator and the generated expressions.

2 Related Work

In this section we first give a theoretical background on learning-based models for human movement analysis [31]. Next, we explain the idea of GANs and discuss their applications in the field of animation. Finally, we explain the idea of WGAN and how it is an improvement on the regular GAN.

*5980968

Contact information: Lennert Sietsma, Utrecht University, Princetonlaan 6, Utrecht, Utrecht, 3584 CB, The Netherlands, lennertsietsma@gmail.com;

To be able to create a facial animation controller bottom-up from data, we need a system that is able to learn a reduced parameter space. Previous research has already shown that we can use learning-based model to learn from human movement. In Bettinger et al. they used a method called state-space trajectories to generate expressive facial animations [8]. They create the state-space using PCA and learn different trajectories through this space to generate facial animations. They were able to generate small segments of facial animations, the results however showed unnatural movement. Later studies showed that realistic human motion acts as a nonlinear system [9]. More recent research used deep learning to analyse human movement [10]. Deep learning is a branch of the machine learning family where a system tries to learn data representations using neural networks (NNs) [11]. There are many types of neural networks and each has their own specialty. One of the most simple variants is called the feedforward neural network or artificial neural network. With these networks the data moves only in one direction: from the input, through the layers towards the output (last) layer. These layers consists out of neurons. A neuron uses a set of inputs, a set of weights and an activation function to output a single value. A feedforward neural network can have any number of layers between the input and output layer. These layers in-between the input and output layers are often referred to as hidden layers. Another type of feedforward neural network is called a multilayer perceptron (MLP). A MLP consists out of at least three layers: input, hidden and output. The network is fully connected, meaning that each neuron of a layer is connected to each neuron of the next layer. MLPs are often used for tasks such as speech recognition [12, 13]. It can also be used on tasks such as point cloud analysis [14, 15], which is very similar to the analysis of 3D points in space. Another commonly used type of NN is called convolutional neural network (CNN). CNNs are often used for computer vision tasks such as image recognition [16, 17], and face detection [18]. Another commonly used type of NN is called recurrent neural networks (RNNs). These networks are mostly used to analyse sequential data which makes them very suitable for tasks such as human action recognition [19, 20, 21], auto-complete keyframe animations [29] and predict 3D human motion [30].

Generative adversarial networks More recently, a new method for training NNs was developed called generative adversarial networks (GANs) [22]. NNs are typically hard to train because they need a lot of training data. With GANs there are two NNs that compete with each other in a minimax game. One of the NNs is generating fake training data (generator) and the other is trying to guess if the data it gets presented is real or fake (discriminator). Because of this architecture we need less training data because we are essentially generating half the training data. Another benefit of this system is that the competitive nature of this implementation encourages the system to learn better. Equation 1 shows this described minimax game between the generator (G) and discriminator (D).

$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

When we look at the definition of the minimax game in equation 1, we see that the discriminator (D) is trying to maximize the equation, while the generator (G) is trying to minimize it. For D to be successful, it needs to be able to tell real from fake images. In the equation this means it should output a one for real samples ($D(x)$) and a zero for the fake samples ($D(G(x))$). This would result in a total loss of zero, the perfect score for the discriminator. On the other hand we have the generator that is trying to generate samples that the discriminator thinks are real. In the equation this means we get the log of a small value, which will result in a larger negative value. This will effectively minimize the equation. The generator produces different samples based on latent space variables (z). These variables are typically sampled from a Gaussian or uniform distribution.

The big advantage of this architecture is that it is not required to have labeled data. Due to the recent success of GANs on image generation [23, 24, 25, 26], the number of applications of GANs is rapidly increasing. GANs have also reached the field of computer animation as it has already shown its capabilities to learn human behavior based on motion capture data [27]. They trained the GAN on motion capture data of humans acting out different tasks. This way the generative part of the GAN generates sequences motion capture data of a human doing a certain task. They showed that GANs are able to generate more human-like motion than regular neural networks. Other research showed that we can train GANs to generate realistic speech driven facial animation [28, 46]. By training the GAN on both speech and video data, the GAN is able to generate a video of someone talking. It even showed that we can generate high quality 3D human head meshes with different identities and expressions [34]. They trained the two different GANs. One was trained on a collection of 3D human head meshes. This GAN learned the identity data of the human heads, training it to generate head meshes with different identities. The other part of the GAN was trained to learn different expressions. They did this by training the GAN on expression data. They acquired this expression data by subtracting the neutral pose from the expression, leaving the deformation vectors belonging to the expression. After the two GANs were trained, they combined them to be able to generate head meshes with different identities and different expressions.

Unfortunately GANs are not perfect and training them can be a very difficult task. One of the difficulties of training a classic GAN [22], is to find a good balance between the generator and discriminator. When, for example, the discriminator is too smart, it is able to tell which samples are real or fake with minimal error. This will cause the loss function to drop to zero. This means we have no gradient to update the weights of the networks (vanishing gradient). This is just one of the problems you can encounter when training a GAN. Since the inventions of GANs, there have been many variations to solve this training difficulty.

Wasserstein generative adversarial network One of these variants uses the Wasserstein distance to improve the classic GAN. The Wasserstein distance is used to measure the distance between two probability distributions. For the Wasserstein generative adversarial network (WGAN) [40], they used this metric to redefine equation 1. In the case of WGAN, the Wasserstein distance would be the distance between the real data distribution and the generated data distribution. Since the original formulation of the Wasserstein distance is highly intractable, the authors of WGAN transformed the formula based on the Kantorovich-Rubinstein duality. The final formula is shown in equation 2.

$$\max_{w\in W} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))] \quad (2)$$

We see that the critic (f_w) is trying to maximize the Wasserstein estimate between the real data ($x \sim \mathbb{P}_r$) and the generated fake data ($g_\theta(z)$). We also see that the critic network is constrained by clamping its weights (w) to a limited range (W). This is due to the Kantorovich-Rubinstein duality transformation of the original Wasserstein distance formula. This transformation makes it that the equation needs to be Lipschitz continuous. The authors of WGAN enforced this constraint by clamping the weights of the critic to a small range. The generator network is not constrained and produces random samples based on latent space variables sampled randomly from a distribution ($z \sim p(z)$). Just as with the regular GAN these variables are typically sampled from a Gaussian or uniform distribution. The objective of the generator is to produce more realistic

samples and so minimizing the Wasserstein estimate.

The big difference between the WGAN and the normal GAN is that the discriminator is not trying to differentiate between real and fake samples anymore. Instead it is approximating a function that is maximizing the output scalar for real data and minimizing for fake data. The difference between these scores is then the estimation of the Wasserstein distance between the real and fake distributions. Because of this change, the authors of WGAN also changed the name of the discriminator to critic, as this better reflects its new role. The great thing about the new objective of the critic is that it can not saturate, but converges to a linear function. While with regular GANs the gradient vanished as the discriminator was trained to much. In fact, it is better to train the critic till optimality so it is able to give a more reliable gradient of the Wasserstein distance. To make sure this happens, the critic is trained multiple times per generator iteration. Being able to fully train the critic also makes it is impossible to encounter mode collapse [40], which is a big plus considering the training stability. Besides the improved training stability, the Wasserstein distance also provides a meaningful loss metric. As the Wasserstein distance decreases, the generated data and real data are more similar. This means that the loss and the quality of the generated samples are related, which makes it easier to evaluate. Another important thing to note is that we can compare the performance of WGANs by looking at the Wasserstein estimate during training. When we do this comparison it is important that the critic networks are the same. This is necessary as the critic produces a constant scaling factor of the Wasserstein estimate that is determined by the networks architecture. When we evaluate the performance of different WGANs based on the Wasserstein estimate, it is essential that the critic networks are identical.

3 Methodology

In this section we will discuss how we will use GANs as a facial animation controller. First we will discuss the data and how we preprocessed it. Secondly we will explain how we implemented the GAN so we can use it as a facial animation controller. Finally we will discuss how we apply the generated expressions to a 3D mesh. The overview of this pipeline is shown in Figure 1.

3.1 Data Preprocessing

For this research we will be using the IEMOCAP corpus [39]. This database consists out of video, audio and motion capture data of both the head and hands of the actor. For this research we will only be using the motion capture data of the face, which is tracked using 55 markers which are mostly placed according the MPEG-4 standard [38]. In total there are 10 actors performing scripted and improvised scenes. For this research we will be using the data of a single actor: male from session 1. We chose this actor because he showed the most extreme facial expression. This might help the system to find the limits of the human face better than we its trained on very subtle expressions.

First of all, we preprocessed the data using the preprocessor created by Oosterom et al. [33]. In the database there are frames where some of the markers are missing. When a marker is missing, the coordinates of these markers are annotated with not a number (NaN). Since the GAN will not know how to handle non-numeric values, linear interpolation is applied to fix these frames and fill in the missing markers. This is done by taking the previously and next known frames (frames without missing markers) and apply linear interpolation between the known markers. Another problem is that sometimes the markers are falsely tracked. This happens when the system thinks a marker is in a certain position while it is not. These falsely tracked markers can be found by looking at the motion of the markers between frames. Since the time of a single frame is relatively short, it is not possible that a marker jumps a large distance. By looking at the movement between frames we can identify these falsely tracked markers and handle them in the same way as the NaN values.

Emotion	Frames
Anger	35159
Excitement	44733
Fear	963
Frustration	51062
Happiness	14624
Neutral	25761
Sadness	58474
Surprise	625

Table 1: Number of frames per emotion after preprocessing

Since we want to create a system that learns natural face movement of a human face, we want to preprocess the data to make the system focus on this. In Cheng et al. [34] they created a system that is able to generate highly detailed head meshes with different identities and expressions. To do this, they trained two different GANs: one for identity and one for expressions. To make the GAN focus on the expressions rather than the identity, they subtracted the neutral pose from data. This way you delete the identity information and are left with the deformations of the face and thus the expression data. We applied the same method with the IEMOCAP data. We manually selected a neutral expression per recording. This is necessary as the markers are reapplied or adjusted between the recordings. To find the best neutral expression, we re-targeted the motion-capture data to a 3D head model (FaceWare Victor). This retargeting system uses a neutral pose and applies the deformations of the tracking data to the model. When the correct neutral expression is applied, the resulting animation should be similar to the video recording of the actor. To see if this was correct, we visually evaluated the results by putting the animation and the original video side by side [35]. After we found the neutral expressions that had the best visual result, we subtracted the selected neutral expression from each of the recordings.

After subtracting the identity from the data, we cluster the data based on the labeled emotion. This is done so we are able to create a well balanced training set with an equal number of framer per emotion. We only used the labeled data and left out surprise and fear as these contained too few samples compared to the other emotions. This left us with the following emotions: frustration, anger, sadness, excitement, happy and neutral. The number of frames per emotion is shown in Table 1.

The GANs are trained using different selections of emotions. The data is normalized after the data selection by subtracting the mean and divide by the standard deviation of the data selection. This is done to help the GAN train faster. Finally the frames are stored as a 1D vector of size 165 (55 3D coordinates) where the coordinates are stored in a 'xyzxyz' order.

3.2 GAN Implementation

First we will discuss which deep learning framework we used to create the GAN. Secondly we will discuss the GAN architecture we used.

Deep Learning Framework

There are many different choices when it comes to picking a deep learning framework. Since this research focuses on analyzing how

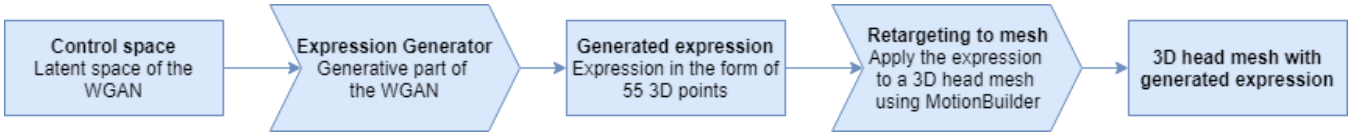


Figure 1: System Pipeline

well GANs perform as an animation controller, we will need to compare different approaches. For this research we chose Keras [37] with a TensorFlow [36] back-end. Keras has a lot of build-in functionality that makes it easier and faster for the user to build a deep learning network. This allows the user to try different architectures without the need of excessive coding. Another reason we chose Keras is that it is able to use different back-ends. To implement the trained GAN in MotionBuilder, we need to be able to import the trained GAN in the integrated scripting environment. This environment runs on python version 2.7, which is not compatible with TensorFlow. By switching the back-end of Keras to CNTK, which is compatible with python 2.7, we are able to import Keras into the integrated scripting environment of MotionBuilder. With Keras imported, it is possible to load and use a trained GAN and use this as an interactive facial animation controller.

GAN Architecture

For this research we started of with a regular GAN as proposed by Goodfellow et al. [22]. Since we wanted to test different architectures and data selections, we needed to re-balance the GAN for each change we made by changing the hyper-parameters. Because the training was highly unstable and finding the right hyper-parameters very time consuming, we decided to go with an improved GAN architecture, namely WGAN [40].

We implemented our WGAN using the recommended hyper-parameters as mentioned in the paper. In the original WGAN paper they tested their method on RGB images of size 64x64. In total they had a total of 12.288 values to evaluate per sample. For this they used a multilayer perceptron (MLP) with four hidden layers with 512 units at each layer. Since our network only needs to evaluate 165 values per sample (55 3D-coordinates), we also used a MLP but with a simpler architecture. For both the generator and the critic we used a MLP with a single hidden layer with 64 units at each layer. The output layers need to be of size one for the critic and size 165 for the generator. The input and hidden layers are using leaky ReLU activation and the output layers are using linear activation. The latent space dimensionality of the generator has to be compact as we propose to use these parameters as controllers for the facial animation controller. For example, a latent space of 100 will require the animator to set a 100 values to acquire a certain facial expression. In Stoiber [31] they found that, when using PCA, the error of their system decreased when they increased the number of dimensions they used to represent their data. They noticed that the error decreased significantly when increasing the number of dimensions from one to six, but would stabilize for dimensions above that. We therefore initially set the latent space dimensionality to six. Since we want to be able to use the generator in the form of a facial animation controller, this is also a good number of control parameters to control the face with. During training we sample the latent space vector z as noise from a Gaussian distribution centered around zero and a standard deviation of one. The architecture for this WGAN is shown in Figure 2. As explained in Section 2, the critic is trying to maximize the difference between the fake and real samples. For this particular implementation it is trying to output high values for real images and low values for fake images. These objectives of the critic are shown in Figure 2 by the blue and red lines. The difference between

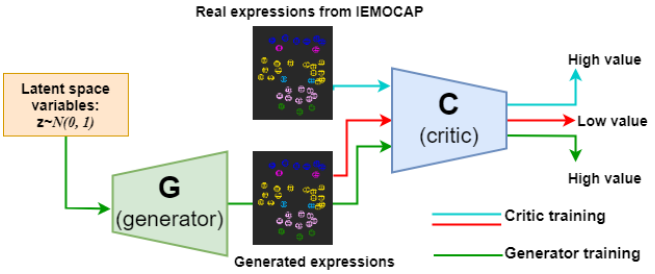


Figure 2: WGAN Architecture

these values is known as the Wasserstein estimate. The generator on the other hand is trying to make the critic output high values for generated images, this is shown in Figure 2 as the green line.

When we look back at Figure 1, the control space is equal to the latent space variables of the WGAN generator. By providing the generator with different latent space variables we can create different expressions. The generator will output an expression in the form of deformations to a neutral pose. These deformations are also still normalized as we normalized our data before training the WGAN. To regain a complete expression, we first undo the normalization by adding the mean deformation and multiplying by the standard deviation. Now we have the expression as a set of deformation vectors per 3D point. To get to the final expression we only have to add a neutral expression to the deformation data. This can be any of the neutral expressions as described in Section 3.1. This will get us our final generated expression in the form of 55 3D-points.

3.3 Animating the 3D Mesh

In order to evaluate the results of this research we visualize the data using a 3D head model. To do this, we use the Autodesk MotionBuilder Actor Face feature. This feature plots an expression on a 3D head mesh based on a set of 3D markers. For this, you provide MotionBuilder with a 3D head model with predefined blendshapes and the tracked markers in a neutral pose. Based on the movement of the tracked markers, MotionBuilder will retarget the points to the model and setting it to the corresponding pose. It is not possible to use all of the 55 3D-points to drive the Actor Face feature in MotionBuilder. Instead we assigned a subset of 3D-points to the head model. This assignment was done by visually evaluation the resulting animation with the ground truth data. We put the resulting animation side by side with the video recording of the actor to see how the well the retargeting was performing [35]. The final configuration of the 3D-points is shown in Figure 3. The numbers inside the blue circles indicate how many markers were used to drive that part of the face. The arrows point to the names of the assigned 3D-points. The names of the points were given by the creators of the IEMOCAP corpus.

The generative part of the trained GAN will output deformations of the 55 3D-points that represent the generated expression. After undoing the normalization and adding a neutral pose, we get the final generated expression. It is important that this neutral pose that we add to the deformations is the same one as is used in MotionBuilder for the neutral pose of the head mesh. This ensures that the generated

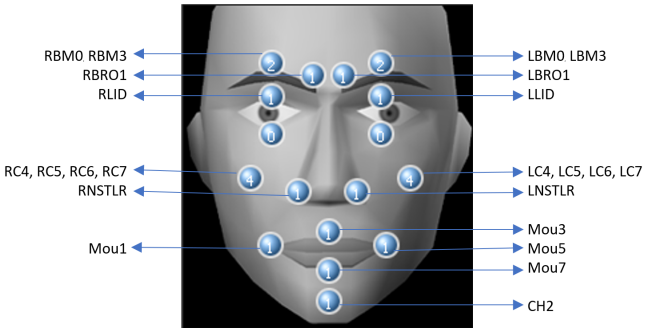


Figure 3: Configuration of IEMOCAP markers on the MotionBuilder Actor Face.

deformation data is retargeted to the model correctly.

4 Evaluation

In this section will we evaluate different aspects of the WGAN. We will start off by doing experiments to improve the WGAN. We do this by changing the hyperparameters to see if this has a positive effect on the results. In the second section we objectively evaluate the WGAN by comparing the resulting expressions to the ground truth data. Finally we will evaluate the results subjectively by visualizing the results using a 3D head model and see if the found control-space has any intuitive meaning.

4.1 Experimental Evaluation

As a first step we train the WGAN on the neutral annotated expressions only. The neutral annotated expressions contains 25761 frames and is split into a training and test set using a 80-20% split. In Figure 4 we can see the Wasserstein estimate during training. The blue line represents the Wasserstein estimate between the training data and generated samples. The orange line represents the Wasserstein estimate between the test data and the generated samples. The Wasserstein estimate stars off high, meaning that there is a large Wasserstein distance between the generated data and the real data. Over time the loss converges towards zero, meaning the Wasserstein distance between the two distribution decreases. The decrease in distance between the two distributions means that the generated samples are getting better as they closer represent the real data. We trained the WGAN using an AMD Ryzen 1600 Six-Core Processor clocked at 3.20 GHz. The training time of this WGAN is about 11 minutes using this processor when we train it for 50.000 generator iterations.

To further improve and evaluate the results of the WGAN we experiment with different latent space sized and critic iterations. The following sections will describe the results of these experiments.

Overfitting

When we want to evaluate the results based on the Wasserstein distance it is important that the WGAN does not overfit. When the WGAN overfits, the correlation between the sample quality and the Wasserstein estimate is no longer there [41]. To see if the networks is overfitting, we plot the Wasserstein estimate on the training and test set. When these two estimates diverge from each other, it means the network is overfitting. When we look at Figure 4, we see that the Wasserstein estimate for the training and test data are converging together and that it shows no signs of overfitting. To illustrate what overfitting would look like we trained the same WGAN using only ten neutral expressions. This way we deliberately overfit the WGAN. The resulting Wasserstein losses are shown in Figure 5. We

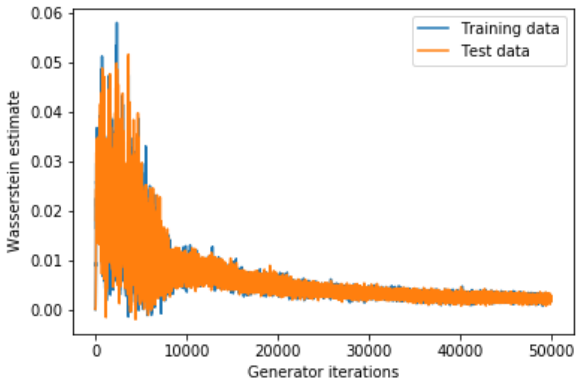


Figure 4: Wasserstein estimate during training of the base network on neutral expressions with the recommended hyperparameters.

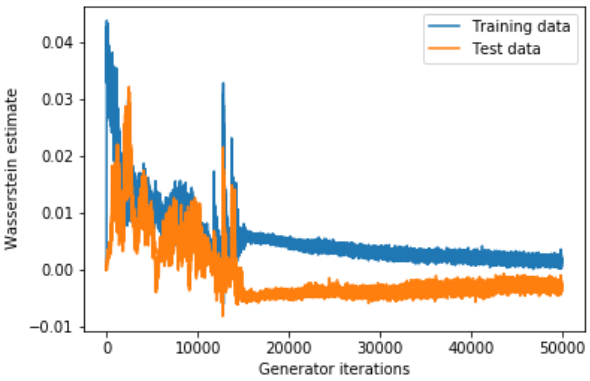


Figure 5: Wasserstein estimate during training. The WGAN is trained using only 10 neutral expressions to show the resulting graph when the WGAN is overfitting. This is shown as the Wasserstein estimate for the training and test data are no longer converging together.

can clearly see that the train and test loss are no longer converging together and thus the network is overfitting and not learning anything meaningful. To make sure none of the changes we make cause the WGAN to overfit, we always look at the Wasserstein estimate for both the training and test data.

Number of Critic Iterations per Generator Iteration

When we look back at the Wasserstein estimate during training of the original system in Figure 4, we see that the distance is not stably converging towards zero. Especially when you look at the bump around 11000 iterations. The most logical explanation for this is that the critic is not trained enough, resulting in a bad estimation of the Wasserstein distance. Because of bad estimate the generator is receiving wrong feedback from the critic. When this happens, it will not be able to adjust its weights correctly. This will cause the generator to generate worse samples, explaining the increase in Wasserstein distance. As explained in Section 2, it does not matter how much we train the critic, just that we train it enough to make for a good estimate. When we train the critic more, the Wasserstein estimate would be more precise and the feedback towards the generator more reliable. To test this we trained the same network with an in-

creased number of critic iterations per generator iteration. We tested this for 10, 20 and 50 critic iterations. The results of the experiments with different critic iterations per generator iteration is shown in Figure 6. We observed a big difference between 10 (Figure 6a) and 20 (Figure 6b) as the bump almost completely disappeared. When we train the critic 50 times per generator iteration (Figure 6c) we also see a slight improvement compared to the 20 critic iterations. We can conclude that the loss converges towards zero in a more stable way as we increase the number of critic iterations per generator iteration. We trained the WGAN using an AMD Ryzen 1600 Six-Core Processor clocked at 3.20 GHz. We trained all WGANs for 50.000 iterations. When we train the critic for 10 critic iterations per generator iteration it takes about 25 minutes. When we increase this to 20 critic iterations the training time increases to about 50 minutes. This increases to about two hours for 50 critic iterations. Since 20 iterations of the critic are enough to provide a reliable Wasserstein estimate, we decided to use this instead of 50 to restrain the training time of the WGAN.

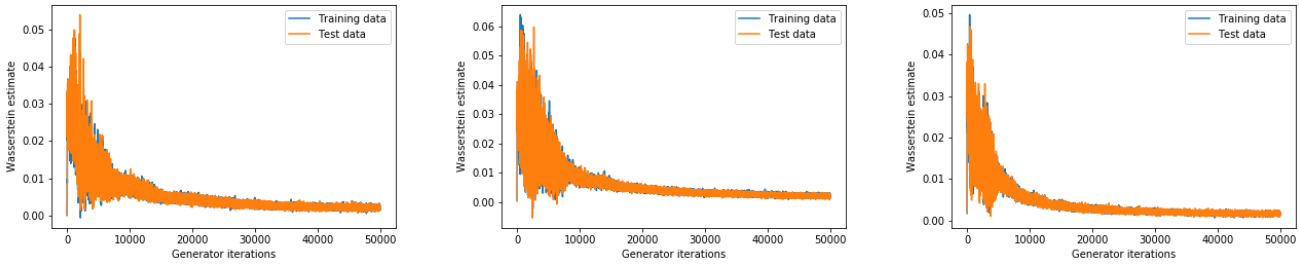
Latent Space Dimensions

We initially set the Latent space dimensions to six as this was found to be the best number of dimensions when using PCA by Stoiber. To see if six dimensions are indeed the right choice, we trained the WGAN using different latent space dimensions. Since it might be possible the ideal latent space dimensionality is dependent on the training data, we analysed the influence of different latent space dimensionalities per emotion as well as combinations of emotions and all emotions combined. When trained on more than one emotion, we make sure there is an equal amount of training data for the selected emotions. To see the effect of the different latent space dimensions we can compare the Wasserstein estimates of the differently trained networks. We can do this because we are not changing the architecture of the critic, only the generator. The generator architecture changes as we change its input dimensionality (latent space dimensionality). This allows us to directly compare the quality of the generated expressions based on the Wasserstein estimate. When we think about the expected behavior of the latent space dimensionality on the Wasserstein estimate, we expect to see that an increase in dimensionality will result in a decrease in error. This is due to the network receiving more input and being able to generate more exact expression and therefor better matching the ground truth data distribution. This effect will weaken as we further increase the number of dimensions. Since our aim is to create a facial controller, we want to minimize the number of control variables while being able to generate a wide variety of expressions. To find this ideal point, we calculate the 98th percentile error over the final 100 Wasserstein estimates of the training process. We use the 98th percentile instead of the mean, just as they did in Stoiber, to reduce the effect of the generator noise on the results. The resulting graphs of this experiment are shown in Figure 7. When we look at the graphs we can see that the error for low dimensionalities is higher. As we increase the latent space dimensionality the error decreases. We also see that the error stabilizes for all trained emotions. The dimensionality at which this stabilization occurs is not the same for all emotions. When we want a WGAN that is able to generate realistic results, we need to pick a latent space dimension large enough so the error has stabilized. We will further evaluate the effect of the latent space dimensionality on the resulting expressions in Section 4.3.

4.2 Objective Evaluation

To see how well this WGAN performs on the data, we start by trained the WGAN with data from a single emotion. We do this for all of the emotions we selected during preprocessing: neutral, happy, excitement, frustration, anger and sadness. To see how well the generated expressions represent the actual data, we generate 1000 random expressions by sampling the latent space from a Gaussian distribution. We then find its nearest neighbor in all data we used from the IEMOCAP database (without the data used for training). We do this by calculating the average Euclidean distance per point between the generated expressions and the expressions in the database. The one with the lowest average Euclidean distance would then be the nearest neighbor. If this neighbor belongs to the same labeled emotion, this means the generator successfully generated an expression that matches with the training data. The results are shown in Figure 8. The emotion below a set of bars indicates the emotion on which the WGAN was trained. The bars represent the percentage of predicted emotional labels. We can see that the WGAN generates samples with the same data as its trained on in most cases. When we look at anger we see that it also scores high for frustration. This can be explained as frustration and anger are neighboring emotion and might show overlap when looking at the expressions [32]. The same goes for excitement and happiness. When we look at the results of the neutral expression we see that it contains a wider variety of expressions than the other emotions. The cause of this most likely lies within the training data. To see if this is the case we visualized the neutral annotated data as a video [43]. The video shows the segments of data that were labeled as neutral. In this video you can see that the actor for example smiles while talking. This will allow the WGAN to learn what a smile is and thus be able to generate smiles. When we calculate the Euclidean distance from this generated smiling expression to the test data, it is likely that there is a closer related smile within the excitement or happy dataset. This explains why the WGAN trained on neutral data also scores high for some other emotions. Since the neutral data is a combination of (subtle) emotions, we will not evaluate on this for the subjective evaluation. Instead we will test on a combinations of the other emotions.

When we look at the last set of bars we see the results for training the WGAN on all of the six selected emotions (including neutral). For this test we selected an equal number of frames from each of the emotions. We did this to ensure that the WGAN learns equally from all emotions. The emotion with the least frames is happy with a total of 14624 frames. Because of this we selected 14624 frames per emotion. Because the data is sequential and recorded as a scene, it could be that for emotions get more expressive as the scene progresses. To avoid discarding valuable data, we pick the frames randomly. After we picked 14624 random frames for each emotion, we split them in training and test data using a 80-20% split. We did this split before we combine the emotions to ensure the training data also contains the same number of frames for all emotions. When we look at the results in Figure 8, we see that the WGAN generates expressions belonging to all emotions. This is to be expected as it was trained on this data. We also see that not all emotions are generated equally. When we look more closely, we see that the WGAN trained on all emotions generates the following emotions from most to least (neutral excluded): sadness, excitement, frustration, anger and happiness. When we look at the highest scores of the WGANs trained on individual emotions, we see that sadness scores the highest. Meaning the sad emotion has the highest density of sad expressions in its dataset. When we look at the other emotions, we see that the density per emotion is ordered from high to low (neutral excluded): sadness, excitement, frustration, anger and happiness. This is the exact same order as the percentages generated expressions for all emotions. When we randomly select data from the individual emotions for the WGAN trained on all emotions, we only select a percentage of the targeted emotion. This means that when we sample from sadness we will get a lot of sad expressions compared to when we sample from happiness. This explains the differences we see in the percentages of generated emotions for the WGAN trained on all expressions.



(a) 10 Critic iterations (b) 20 Critic iterations (c) 50 Critic iterations

Figure 6: Wasserstein estimate with different number of critic iterations per generator iteration. All networks are trained on neutral annotated data only which is split into a training and test set suing a 80-20% split. This figure shows that as we increase the number of critic iterations the training becomes smoother. This can be seen in the graphs as the bump around 11000 iterations starts disappearing the more critic iterations are used.

4.3 Subjective evaluation

Besides objective evaluation, we can also evaluate the result subjectively by visualizing the generated expression. We evaluate each of the previously trained models to see if the learned relation between the latent space variables and the resulting expressions has some intuitive explanation. To visualize the generated expression, we retarget them to 3D head mesh (Victor from FaceWare Tech) using Autodesk MotionBuilder as described in Section 3.3. To see how the different latent space variables influence the generates expressions, we altered a single latent space variable at a time while the others remained at the mean value (zero). Since we trained the system based on a Gaussian distribution with a standard deviation of one, we decided to use a range of [-3, 3] to as input for the generator. We used this range as this would cover 99.7% off the Gaussian distribution and therefor all sampled inputs during training. The following sections will describe our finding.

Anger

In Appendix A we see the results of the WGAN trained on angry annotated data. When we look at the overall expressions we see that there are mostly negative expressions such as frowns. This confirms the results we saw during the objective evaluation as the anger dataset had low scores for both happy and excitement. When we look at the control of each of the individual latent space variables we see that each of the variables controls certain aspects of the face. When we look at the first variable in particular we see that this variable mainly controls the movement of the eyebrows. The second feature moves the corners of the mouth up (column -3) and down (column 3). The third feature opens and closes the face in general by moving both the mouth and eyebrows. The fourth feature moves the jaw, making the model open and close its mouth. The fifth feature seems to be controlling the lips without moving the jaw. This allows the model to show its teeth. Feature six seems to be controlling both the jaw and the lips allowing the model to make more complicated movements with the mouth such as a yawn. When looking at the results we can see that the WGAN correctly learned what angry expressions look like and how different parts of the face move to create these expressions.

Excitement

In Appendix B we see the results of the WGAN trained on excited annotated data. Again we see that the overall expression matches the training data as we see mostly positive expressions. When we look at the individual control of the latent space variables, we again see that they are controlling different aspects of the face. The difference is that now they are focused on setting the face in different positive expressions. The first feature mostly controls the shape of the mouth. The second feature controls the height of the eyebrows. The third feature opens up the face. The fourth feature controls the corners of the mouth as well as the jaw and lips, effectively making a very excited expression. The fifth feature has some subtle control over the shape of the mouth. The last feature controls the jaw and lips which is most likely learned as the actor is talking throughout the recordings.

Frustration

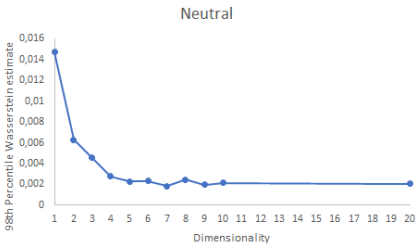
In Appendix C we see the resulting expressions when controlling the individual latent space variables for a WGAN trained on frustrated data. In this case we see mostly negative expressions. It is interesting that, compared to the angry trained WGAN, we also see some sad looking faces such as the one feature six value -3 displays. This matches with what we saw in during the objective evaluation as frustration also scored high for sad expressions. When we look at the individual features we see that the first feature controls the jaw. The second feature controls the corners of the mouth as well as the eyebrows. When the jaw opens the eyebrows raise and vice versa. The third feature also controls the jaw and eyebrows. Interestingly is that when we compare this to feature two it raises the eyebrows as it closes the mouth. Feature four lowers the bottom half of the lips and jaw, resulting in an expression that looks like a sigh. Feature five controls the shape of the mouth as well as the eyelids. Feature six controls the mouth and the jaw as if the model is talking.

Happiness

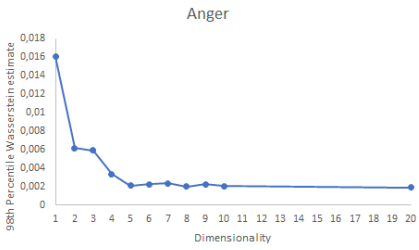
Appendix D shows the results for the WGAN training on happy data. As expected the overall expressions look positive. The first feature shows control over the eyebrows. The second feature shows control over the lips, yaw and eyebrows as if the person is laughing loudly. The third feature shows control over the jaw and lips gradually going from an intense smile to a subtle mouth opening. The fourth feature controls the lips from a subtle smile to an expression looking like disgust. Feature five shows control over the mouth and eyebrows. The mouth movement makes the smile move from the left side of the face to the right and the eyebrows move up as we increase the value of the feature. Feature six shows control over the jaw and lips. The expression moves from a neutral to an mouth opening without showing much teeth as if the person were talking.

Sadness

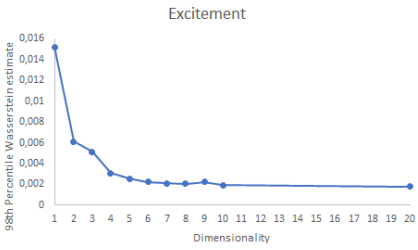
Appendix E shows the results for the WGAN trained on sad data. Again we see that the generated expressions mostly represent the



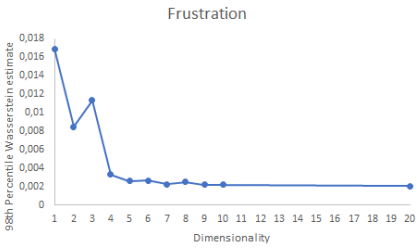
(a) Trained on neutral data



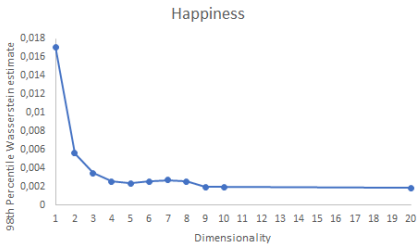
(b) Trained on angry data



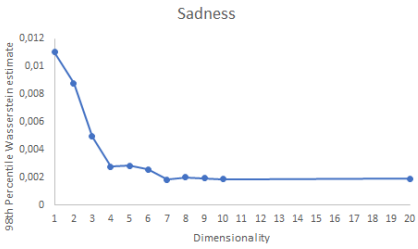
(c) Trained on excited data



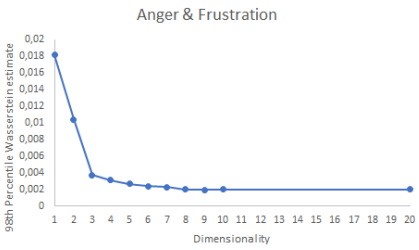
(d) Trained on frustrated data



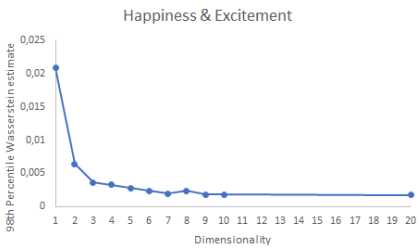
(e) Trained on happy data



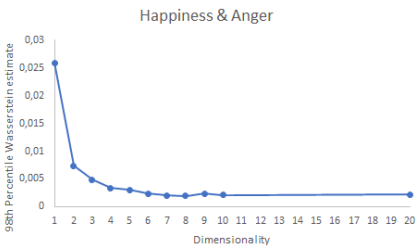
(f) Trained on sad data



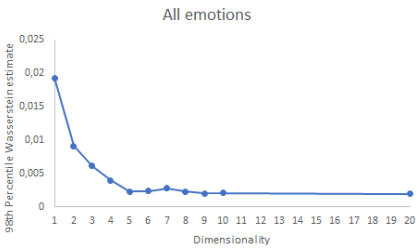
(g) Trained on angry and frustrated data



(h) Trained on happy and excited data



(i) Trained on happy and angry data



(j) Trained on all emotions

Figure 7: 98th Percentile Wasserstein estimate for different latent space dimensionalities with WGANs trained on different emotions. The networks that are trained on more than one emotion use an equal number of training and validation data for the selected emotions. All networks were trained using 20 critic iterations per generator iteration and for a total of 50.000 generator iterations. The graphs show that the increasing the latent space dimensionality benefits the WGANs as the Wasserstein estimates decreases. It also shows that the Wasserstein estimate stabilized when the latent space dimensionality is large enough.

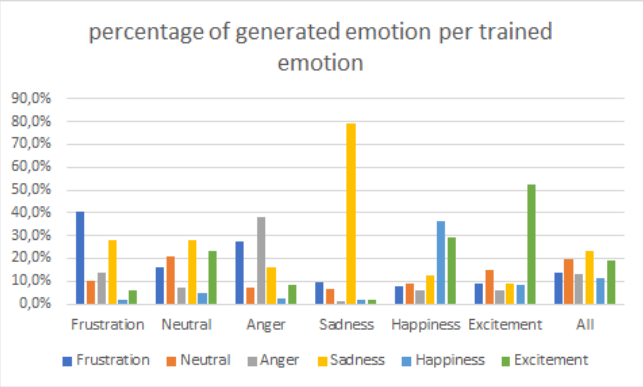


Figure 8: Trained WGANs on data sets containing different emotions. The emotion on which the WGAN was trained is displayed below a set of bars. We show the percentages of the found labels of the nearest neighbor of 1000 randomly generated expressions. The percentage of found emotional labels are shown in the figure as the colored bars. The WGANs were all trained using a latent space dimensionality of six, 20 critic iterations per generator iteration and trained for a total of 50.000 generator iterations.

expressions in the training data. In this case we see many expressions where the corners of the mouth are tucked down, indicating a sad expression. When we look at the control the individual features have, we see that they are again controlling different aspects of the face. The first feature controls the opening and closing of the face. The second feature controls the subtle opening of the mouth. The third feature controls the shape of the lips from negative (sad) to neutral. The fourth feature controls the eyebrows. The fifth feature controls the shape of the lips from neutral to slightly positive. The sixth feature opens and closes the face, but more subtle than feature one.

Anger and happiness

Since we saw that training on individual emotions creates a control-space that maps features to control different aspects of the face, it would be interesting to see how it behaves when we train it on multiple emotions. As a first test we trained the WGAN on two contradicting emotions: anger and happiness. We chose these contradicting emotions as these are easy to visually differentiate. The results of this experiment are shown in Appendix F. We can see a mix of positive (smiling) and negative (frowning) expressions. This is to be expected as we trained on both these expressions. When we look at the influence of the individual features, we see that feature one controls the smiling intensity using the lips and jaw. Feature two controls the eyebrows moving them from a frowning pose to a neutral pose. Feature three opens the lips and jaw. Feature 4 moves the eyebrows and corners of the mouth. Feature 5 controls the shape of the mouth and the eyelids. The sixth feature shows subtle control over the eyebrows makes the expression move from slightly angry to slightly happy. Just as with the singular emotion, we again see that the WGAN is showing control over different aspects of the face. The positive thing is that we can see an equal number of positive and negative expressions.

To see if the WGAN is able to learn that there is a difference between happy and angry, we trained the WGAN with a latent space dimension of two. Ideally the latent space variables would match a single emotion each. For example feature one would be controlling a happiness while feature two would be controlling the anger. We show the results of this experiment in Appendix G. When we look at the individual features, we see that feature one moves the eyebrows while feature two mainly moves the mouth. It is clear that the latent space variables on their own do not represent different emotions.

When we look back at Figure 7i, we see that the error of the angry and happy data starts stabilizing for dimensionalities greater than four. This means that we at least need four dimensions to be able to closely represent the real data distribution. To see how the WGAN would use these four latent space variables, we did another experiment with a latent space dimensionality of four. The results of this experiment can be seen in Appendix H. We can now see both happy, angry and mixed expressions. The first feature controls the eyebrows, going from an angry expression towards and neutral expression with the eyebrows raised. The second feature has the most influence on the lips and jaw. This results in an expression that moves from a neutral expression with the mouth fully closed to an expression where the mouth has an oval shape. The third feature also shows most control over the lips and jaw, going from a round shaped mouth toward a closed jaw smile with the teeth still visible. The fourth feature also controls the lips and the jaw, but more subtle. It goes from a neutral position with the jaw completely shut to an expression with the lips and jaw slightly open, creating a smile. Considering this WGAN is trained on happy and angry data while the actor was talking, the results represent the original data quite well. We see full control over the eyebrows. We also see different ways to control the mouth as if its talking and creating different syllables. We can combine the control of the individual feature to create combination of the expressions we see in Appendix H. The resulting facial animation controller based on this trained WGAN can be seen in this video [44]. As can be seen in video, we can successfully create combination of expression with this controller. However, we have to be careful not to use the maximum range of the variables too much as the network is not trained well for these edge cases. As a result we see some less natural expressions when we put multiple variables to their minimum and maximum values.

All emotions

Finally, for our facial animation controller we want to be able to create all sorts of emotions using a single system with a limited number of variables. We saw that when we picked the latent space dimensionality at which the error started stabilizing, we got the best results for the emotions happy and angry combined. When we look back at Figure 7j, we see that this happens at a dimensionality of five when trained on all emotions: neutral, anger, excitement, frustration, happiness and sadness. The results are shown in Appendix I. We can see that features one and four control different variants of a smile. Features two and three control different negative expressions. Feature five controls the corners of the mouth, which looks like sadness. The results seem to indicate that the WGAN is able to learn the different expressions and map them to different latent space variables. It is however hard to distinguish between happiness and excitement as well as anger, frustration and sadness. In order to evaluate the result better, we train the WGAN on better distinguishable emotions. For this we use the six basic emotions by Ekman [42]: anger, happiness, surprise, disgust, sadness and fear. Since our dataset does not contain samples from disgust, we decided to train the network on the remaining five emotions. Our selection of data only contains 625 frames for fear and 963 frames for surprise, hence we discarded them from training before. Since we want the WGAN to learn equally from all emotions we also need to select 625 frames from all emotions and do. In total this means we will have 3125 frames of which 2500 (80%) will be used for training and 625 (20%) will be used for testing for overfitting. The Wasserstein estimate of both training and test data converged together, meaning there is no sign of overfitting. To see what latent space dimensionality would be best for this particular data selection, we repeated the latent space analysis as was done in Section 4.2. The resulting graph

of this latent space analysis is shown in Figure 9. As can be seen in the figure, the Wasserstein estimate stabilizes for dimensionalities greater than four. We therefor select the dimensionality of four to be the best match for this WGAN trained on this data selection. The results are shown in Appendix J. When we look at the first feature we see that the expressions moves from a neutral to an expression that looks like sadness. The second feature moves from an expression that looks like fear or disgust and moves toward happiness. The third feature moves from anger towards a neutral or surprised expressions. The fourth feature moves from a closed expressions (eyes and mouth completely closed with eyebrows down) to an open expression (mouth and eyes open as well as eyebrows raised). We again see that it is hard to identify an emotion based on a single frame. What we can see is that the WGAN is able the generate a wide variety of expressions using only the individual variables. We can also use all of the features to mix the expressions. We can for example use feature one with value minus three (top-left image of Appendix J), which closes the jaw, combined with feature two to create new expressions. When we look at the second feature we see that for positive values it generates a smile with the jaw open. When we combine this with feature one at value minus three we will get an expression where the jaw is closed, but the lips smile and show the teeth. This is confirmed when we visualize this result in Figure 10. When we are combining emotions, we have to account for the way the WGAN is trained. During training the noise is samples from a Gaussian distribution which has a high density around the mean. The cases at the sides of this distribution are sampled significantly less. For example, the values within range [2, 3] are only sampled in 2.1% of the cases. The odds of the latent space being sampled with all variables within the range [2,3] is even lower. Since these latent space configurations are rarely sampled, the generator has received very little feedback for these cases. As a result of this, the resulting expressions for these cases will look less natural. An example of such an expression is shown in Figure 11. We also implemented this trained WGAN as an facial animation controller. To demonstrate the resulting controller we created a video showing its capabilities [45]. The video shows the effects of the single variables, careful combinations of the variables and finally extreme combination of the variables resulting in less natural expressions. Overall the controller shows good natural expressions belonging to different emotions.

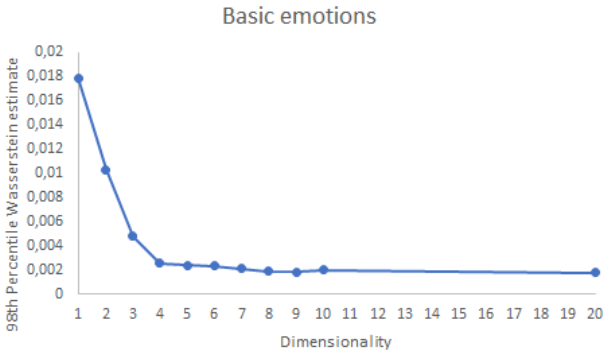


Figure 9: 98th Percentile Wasserstein estimate for different latent space dimensionalities with WGANs trained on five out of the six basic emotions. The networks is trained using 500 samples for each emotion and tested using 125 samples per emotion against overfitting. The WGANs were trained using 20 critic iterations per generator iteration and for a total of 50.000 generator iterations. The graphs show that the increasing latent space dimensionality benefits the WGANs as the Wasserstein estimates decreases. It also shows the decrease in Wasserstein estimate stabilizes for dimensionalities greater than four.

Figure 10: WGAN trained on basic emotions with a latent space dimensionality of 4. The image displays the result of mixing the latent space variables to generate new expression. In this case the latent space input is equal to [-3, 2, 0, 0]. Generating a open mouth smile with the jaw shut.

5 Conclusion

The results of the experimental evaluation show that a WGAN is able to learn from facial expressions in the form of 3D points. By comparing the Wasserstein estimate for different latent space dimensionalities, we find that the accuracy of the generated expressions decreases as we increase the dimensionality. This analysis also shows that this reduction in error does not last forever and stabilizes. Depending on what data the WGAN is trained on, we need different latent space dimensionalities before the error starts to stabilize.

The objective evaluation shows that a WGAN trained on a specific emotion is able to generate expressions which, in most cases, belong to that same emotion. This experiment also shows that the sequences annotated with a certain emotion does not only contain expressions belonging to that emotion. This was confirmed by the visually analyzing the neutral data. It also shows that a WGAN trained on all emotions is able to generate all emotions on which it was trained.

The subjective evaluation shows that the expressions the WGAN generates also visually match the data it was trained on. The WGANs trained on positive emotions are also mostly generating positive looking expressions and the same goes for negative emotions. We also found that the latent space variables of a WGAN trained on a single emotion controls different regions of the face.

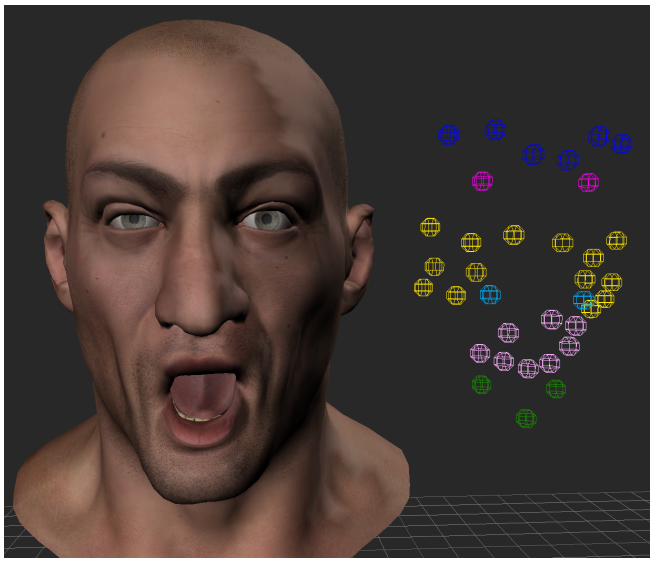


Figure 11: WGAN trained on basic emotions with a latent space dimensionality of 4. The image displays the result of setting the latent space variables to their extreme values. In this case the latent space input is equal to [3, 3, 3, 3]. Generating an less natural expression.

When we train a WGAN on multiple emotions we see that the latent space variables do not match to specific emotions. Even when we set the latent space dimensionality is set to the same dimension as there are emotions, there is no clear correlation. Another finding we did is that when we pick a latent space dimensionality for which the error just stabilized, we got the most meaningful mapping. In these cases we got control over different aspects of the face belonging to certain emotional features.

Overall the evaluation shows that the latent space variables provide control over certain regions of the face and sometimes even over emotions. This indicates that it would be possible for a GAN to learn a meaningful mapping that can be used as a facial animation controller. The problem with this implementation of WGAN is that its learning in an unsupervised manner. Because its unsupervised, the WGAN is learning a mapping between the latent space and the expressions which is randomly assigned by the network itself. When we were to implement a conditional GAN which uses labeled emotional data (supervised learning), we might be able to assign control over a certain emotion to a certain latent space variable. When such a new system were to be created, we would recommend to use a database that is labeled using the six basic emotions [42]. This will help with the subjective evaluation as these emotions are fundamentally different and are easier to distinguish. For example, we saw that emotions such as excitement and happiness as well as anger and frustration are difficult to visually distinguish. Another great property of the basic emotions is that they are shaped by evolution and are therefore similar for all humans [42].

During the evaluation we saw that the expressions labeled with a certain emotion do not always match the underlying data. This is due to the data being labeled as a sequence and not as frames. Another great way to improve this GAN is to train it on sequences of face movement with different emotions. This way the GAN learns how a face moves through time with a certain emotion. This can be done with either tracking data such as we used here or with more detailed face data as provided by the 4DFAB database [47]. This could then be combined with a GAN that learns lip and mouth movement based on speech (audio) [28, 46]. This would allow a system to generate a speaking face with a certain emotion based on speech and an emotional state. Such a system could be very useful in the game and film industry as well as many other applications.

6 Acknowledgements

I would like to thank my supervisor dr. Z. Yumak for her time, feedback and guidance during this thesis. I would also like to thank my second supervisor, prof. dr. R.C. Veltkamp, for taking the time to evaluate my work.

References

[1] Scherer, Klaus R. *Appraisal theory.* Handbook of cognition and emotion (1999): 637-663.

[2] Russell, James A. *A circumplex model of affect.* Journal of personality and social psychology 39.6 (1980): 1161.

[3] Balasubramanian, Mukund, and Eric L. Schwartz. *The isomap algorithm and topological stability.* Science 295.5552 (2002): 7-7.

[4] Roweis, Sam T., and Lawrence K. Saul. *Nonlinear dimensionality reduction by locally linear embedding.* science 290.5500 (2000): 2323-2326.

[5] Weise, Thibaut, et al. *"Realtime performance-based facial animation."* ACM transactions on graphics (TOG). Vol. 30. No. 4. ACM, 2011.

[6] Dutreve, Ludovic, Alexandre Meyer, and Saïda Bouakaz. *Feature points based facial animation retargeting.* Proceedings of the 2008 ACM symposium on Virtual reality software and technology. ACM, 2008.

[7] Curio, Cristóbal, et al. *Semantic 3d motion retargeting for facial animation.* Proceedings of the 3rd symposium on Applied perception in graphics and visualization. ACM, 2006.

[8] Bettinger, Franck, Timothy F. Cootes, and Christopher J. Taylor. *Modelling Facial Behaviours.* BMVC. Vol. 2. 2002.

[9] Li, Yan, Tianshu Wang, and Heung-Yeung Shum. *Motion texture: a two-level statistical model for character motion synthesis.* ACM transactions on graphics (ToG). Vol. 21. No. 3. ACM, 2002.

[10] Min, Jianyuan, Yen-Lin Chen, and Jinxiang Chai. *Interactive generation of human animation with deformable motion models.* ACM Transactions on Graphics (TOG) 29.1 (2009): 9.

[11] Schmidhuber, Jürgen. *Deep learning in neural networks: An overview.* Neural networks 61 (2015): 85-117.

[12] Zhu, Qifeng, et al. *Using MLP features in SRI's conversational speech recognition system.* Ninth European Conference on Speech Communication and Technology. 2005.

[13] Kingsbury, Brian ED, Nelson Morgan, and Steven Greenberg. *Robust speech recognition using the modulation spectrogram.* Speech communication 25.1-3 (1998): 117-132.

[14] Tchapmi, Lyne, et al. *Segcloud: Semantic segmentation of 3d point clouds.* 2017 International Conference on 3D Vision (3DV). IEEE, 2017.

[15] Yang, Yaoqing, et al. *Foldingnet: Point cloud auto-encoder via deep grid deformation.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[16] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. *Imagenet classification with deep convolutional neural networks.* Advances in neural information processing systems. 2012.

[17] He, Kaiming, et al. *Deep residual learning for image recognition.* Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[18] Li, Haoxiang, et al. *A convolutional neural network cascade for face detection.* Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[19] Liu, Jun, et al. *Spatio-temporal LSTM with trust gates for 3D human action recognition.* European Conference on Computer Vision. Springer, Cham, 2016.

[20] Jain, Ashesh, et al. *Structural-RNN: Deep learning on spatio-temporal graphs.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[21] Li, Wenbo, et al. *Adaptive RNN tree for large-scale human action recognition.* Proceedings of the IEEE International Conference on Computer Vision. 2017.

[22] Goodfellow, Ian, et al. *Generative adversarial nets.* Advances in neural information processing systems. 2014.

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

[24] Lu, Yongyi, Yu-Wing Tai, and Chi-Keung Tang. *Conditional cyclegan for attribute guided face image generation.* arXiv preprint arXiv:1705.09966 (2017).

[25] Xu, Qiangeng, Zengchang Qin, and Tao Wan. *Generative cooperative net for image generation and data augmentation.* International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making. Springer, Cham, 2019.

[26] Yang, Jianwei, et al. *Lr-gan: Layered recursive generative adversarial networks for image generation.* arXiv preprint arXiv:1703.01560 (2017).

[27] Merel, Josh, et al. *Learning human behaviors from MOTION CAPTURE by adversarial imitation.* arXiv preprint arXiv:1707.02201 (2017).

[28] Vougioukas, Konstantinos, Stavros Petridis, and Maja Pantic. *End-to-End Speech-Driven Facial Animation with Temporal GANs.* arXiv preprint arXiv:1805.09313 (2018).

[29] Zhang, Xinyi, and Michiel van de Panne. *Data-driven autocompletion for keyframe animation.* Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games. ACM, 2018

[30] Kundu, Jogendra Nath, Maharshi Gor, and R. Venkatesh Babu. *BiHMP-GAN: Bidirectional 3D Human Motion Prediction GAN.* arXiv preprint arXiv:1812.02591 (2018).

[31] Nicolas Stoiber, *Modeling emotionnal facial expressions and their dynamics for realistic interactive facial animation on virtual characters.* Human-Computer Interaction [cs.HC]. Université Rennes 1, 2010.

[32] Plutchik, Robert. *The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice.* American scientist 89.4 (2001): 344-350.

[33] Teus van Oosterom, *Facial Animation Retargeting using Recurrent Neural Networks* MSc thesis, Utrecht University, 2018

[34] Cheng, Shiyang, et al. *MeshGAN: Non-linear 3D Morphable Models of Faces.* arXiv preprint arXiv:1903.10384 (2019).

[35] Sietsma, Lennert. *Comparison of the retargeted IEMOCAP data against the Ground trust video* https://www.youtube.com/playlist?list=PLy9sPTwU_zn1k16TsOIDiKEKpgSz0aSqO

[36] Abadi, Martín, et al. *Tensorflow: a system for large-scale machine learning.* OSDI. Vol. 16. 2016.

[37] Chollet, François. *Keras: Deep learning library for theano and tensorflow.* URL: https://keras.io/ 7.8 (2015).

[38] Ostermann, Jörn. *Animation of synthetic faces in MPEG-4.* Proceedings Computer Animation'98 (Cat. No. 98EX169). IEEE, 1998.

[39] Busso, Carlos, et al. *IEMOCAP: Interactive emotional dyadic motion capture database.* Language resources and evaluation 42.4 (2008): 335.

[40] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. *Wasserstein generative adversarial networks.* International conference on machine learning. 2017.

[41] Gulrajani, Ishaan, et al. *Improved training of wasserstein gans.* Advances in neural information processing systems. 2017.

[42] Ekman, Paul. *Basic emotions.* Handbook of cognition and emotion 98.45-60 (1999): 16.

[43] Sietsma, Lennert. *Neutral annotated data from IEMOCAP (session 1 M)* https://youtu.be/OEjnxKZVASw

[44] Sietsma, Lennert. *WGAN trained on happy and angry as facial animation controller* https://youtu.be/wNl9mfYjHvA

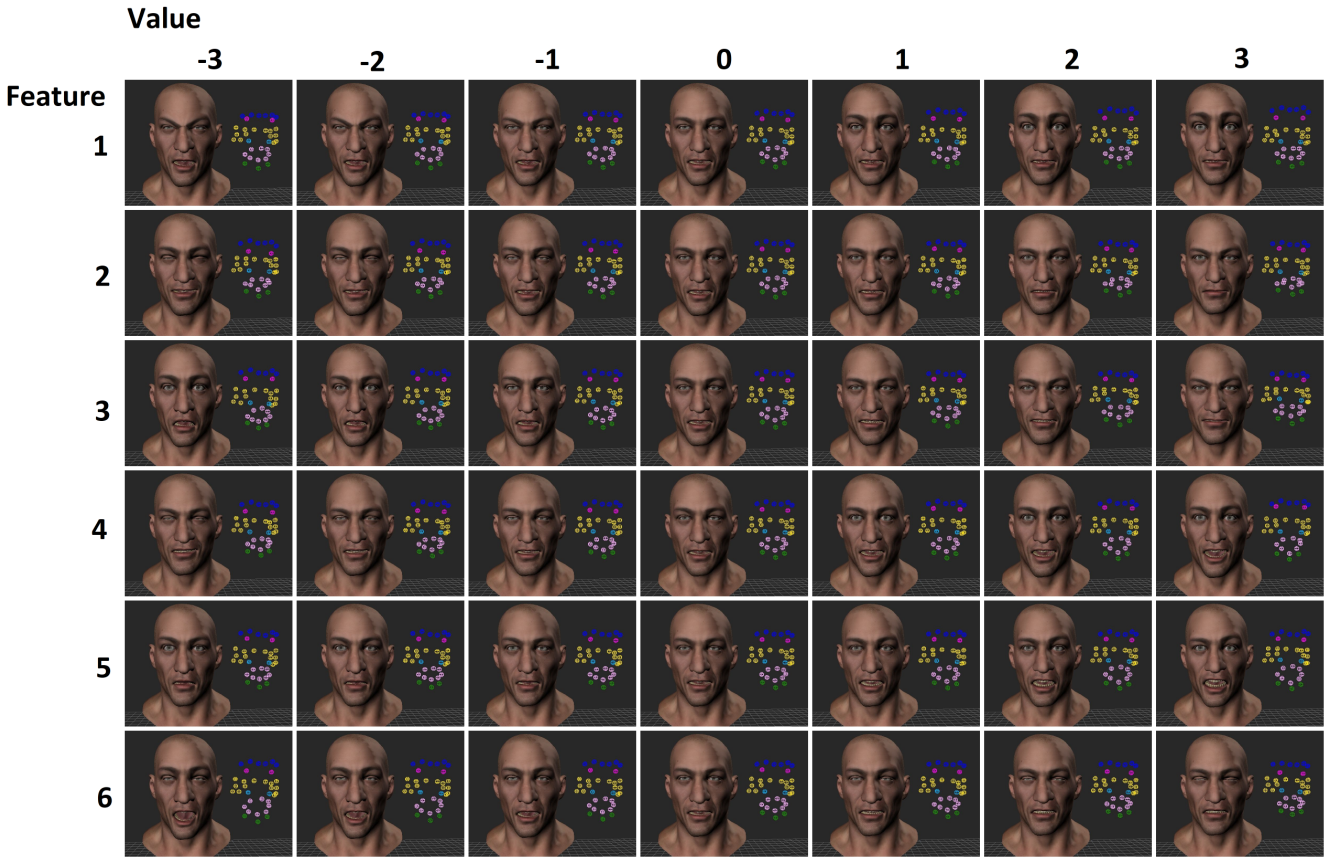
[45] Sietsma, Lennert *WGAN trained on basic emotions as facial animation controller* https://youtu.be/13iwTq16uEA

[46] Vougioukas, Konstantinos, et al. *End-to-End Speech-Driven Realistic Facial Animation with Temporal GANs.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2019.

[47] Cheng, Shiyang, et al. *4dfab: A large scale 4d database for facial expression analysis and biometric applications.* Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

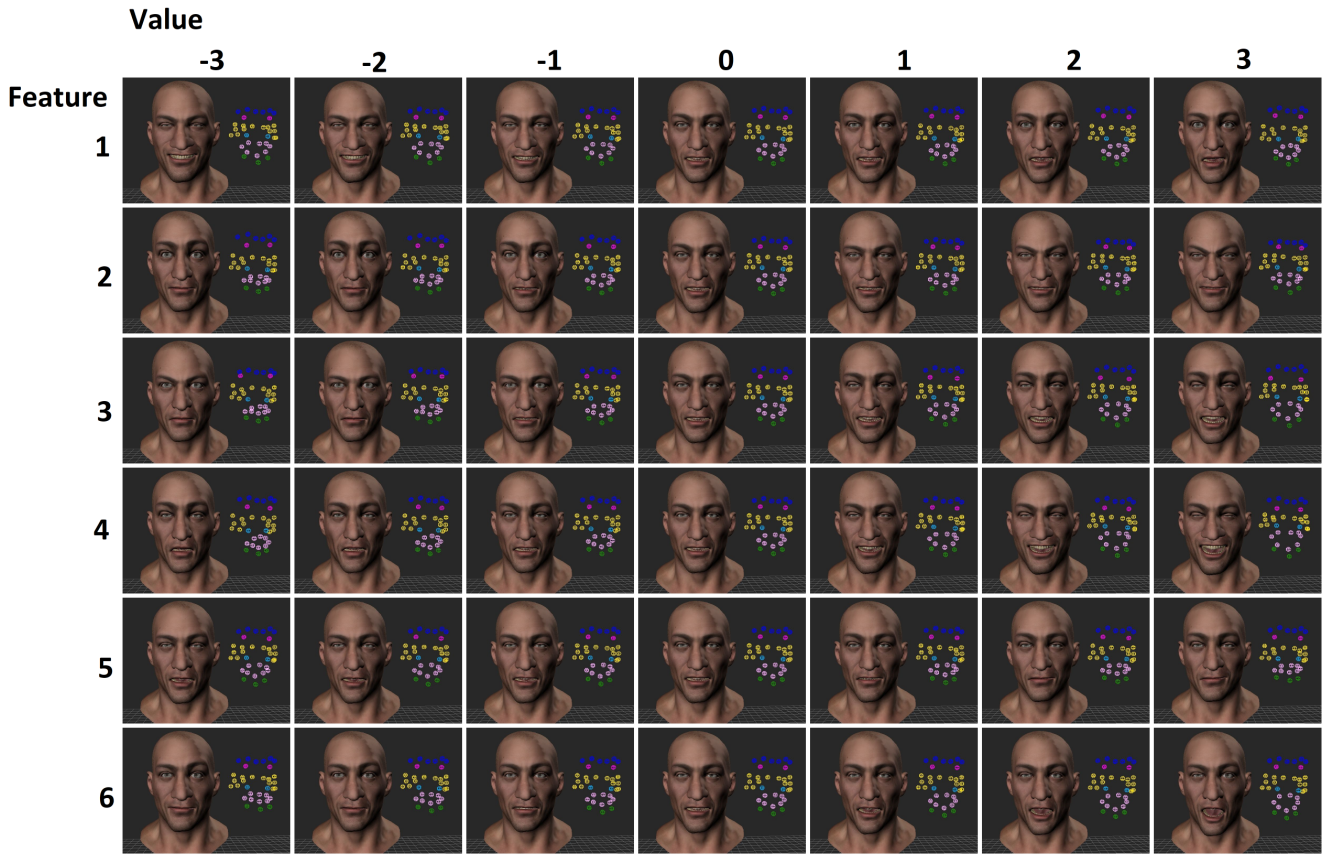
Appendices

A Results of WGAN trained on angry data



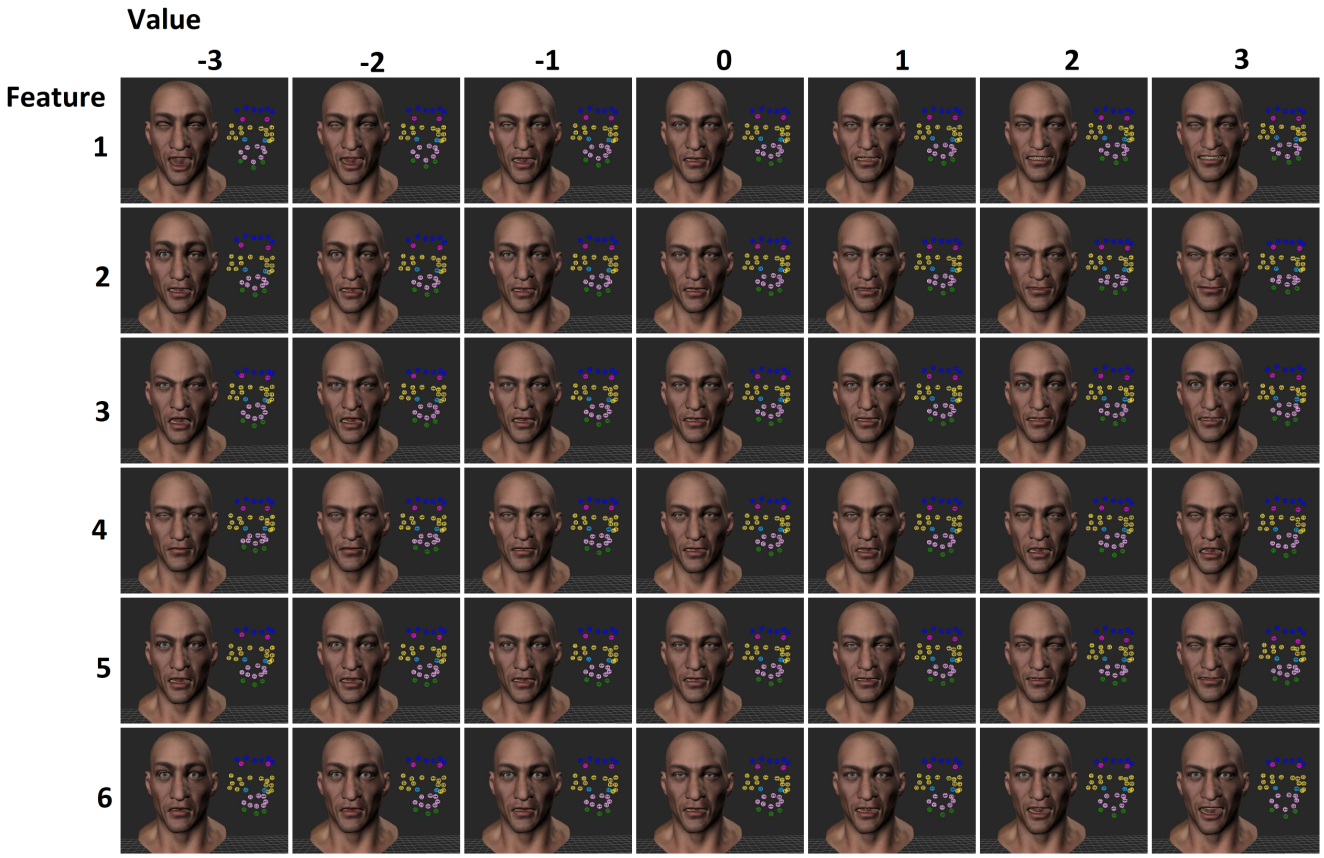
Results of the WGAN training on angry annotated data (35159 frames with a 80-20% train-test split) with 20 critic iterations per generator iteration and a latent space dimensionality of six. The rows in this image represent the influence of each individual latent space variable. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0, 0, 0] and bottom right [0, 0, 0, 0, 0, 3].

B Results of WGAN trained on excited data



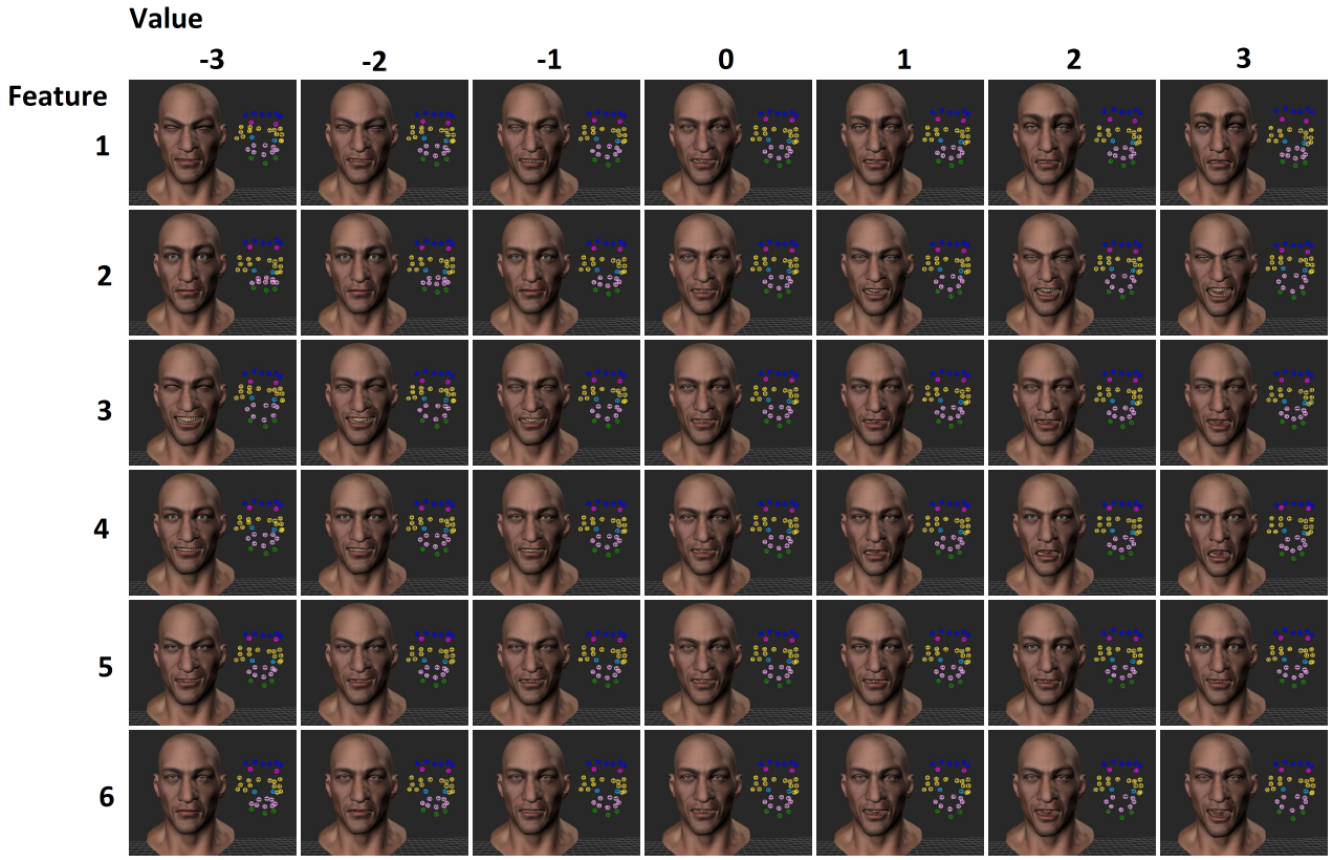
Results of the WGAN training on excited annotated data (44733 frames with a 80-20% train-test split) with 20 critic iterations per generator iteration and a latent space dimensionality of six. The rows in this image represent the influence of each individual latent space variable. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0, 0, 0] and bottom right [0, 0, 0, 0, 0, 3].

C Results of WGAN trained on frustration data



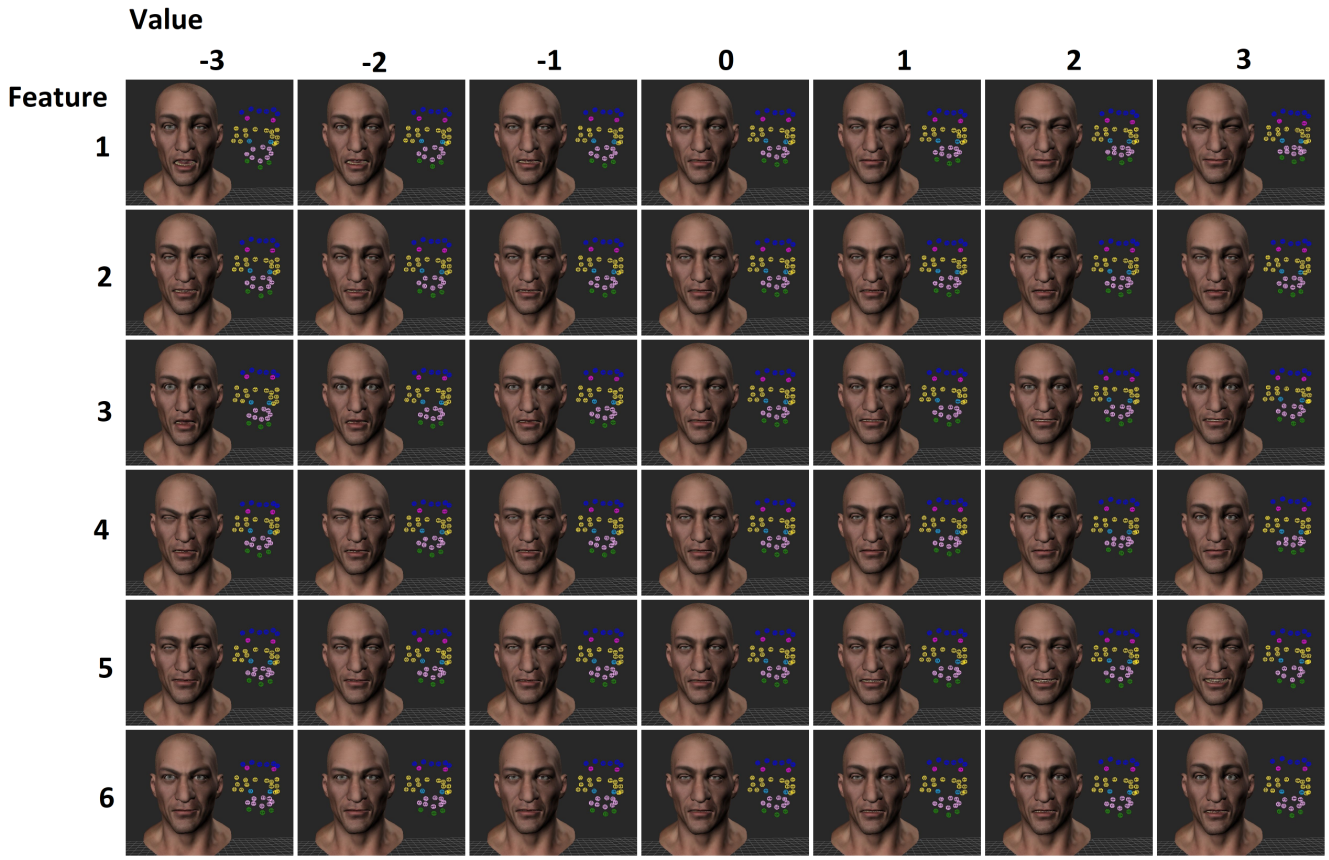
Results of the WGAN training on frustrated annotated data (51062 frames with a 80-20% train-test split) with 20 critic iterations per generator iteration and a latent space dimensionality of six. The rows in this image represent the influence of each individual latent space variable. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0, 0, 0] and bottom right [0, 0, 0, 0, 0, 3].

D Results of WGAN trained on happy data



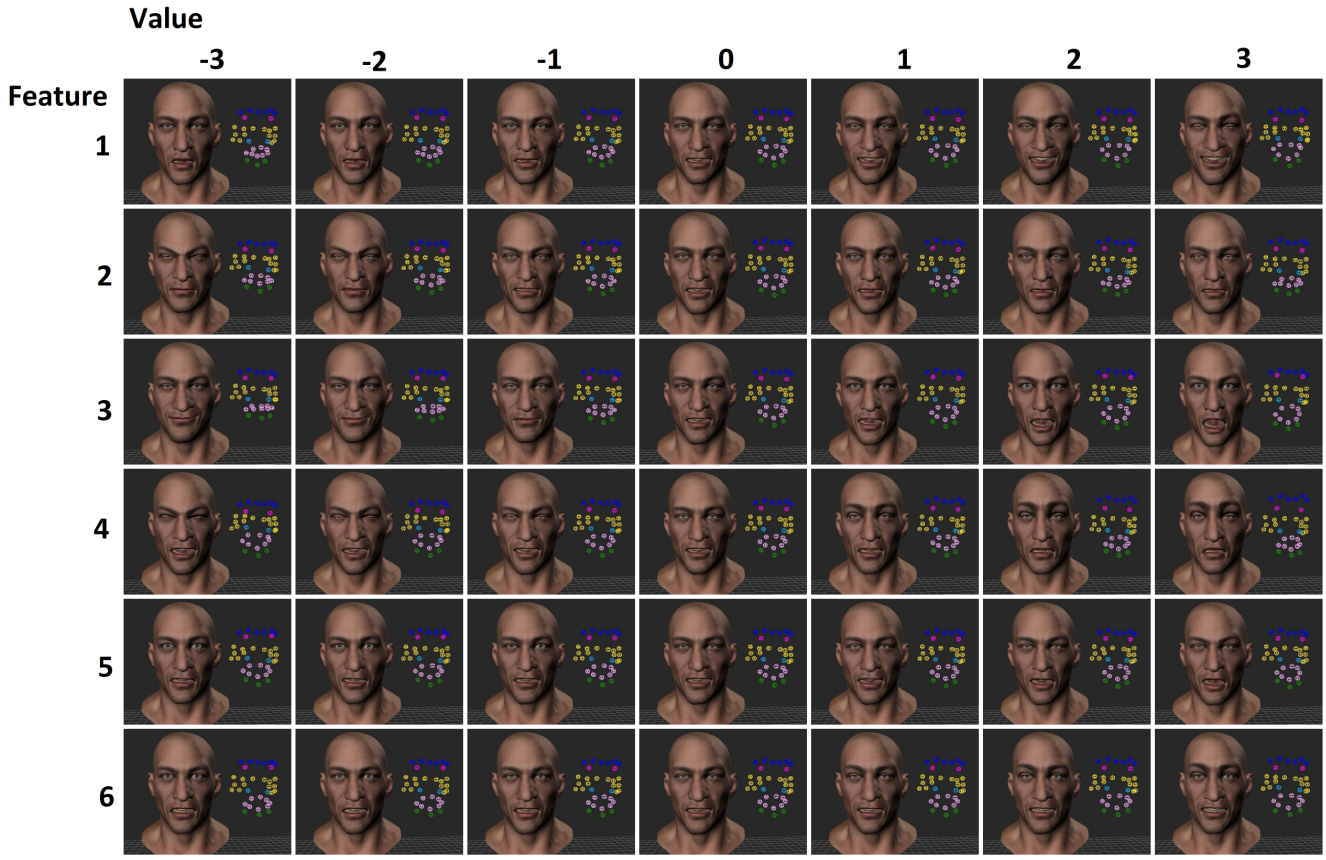
Results of the WGAN training on happy annotated data (14624 frames with a 80-20% train-test split) with 20 critic iterations per generator iteration and a latent space dimensionality of six. The rows in this image represent the influence of each individual latent space variable. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0, 0, 0] and bottom right [0, 0, 0, 0, 0, 3].

E Results of WGAN trained on sad data



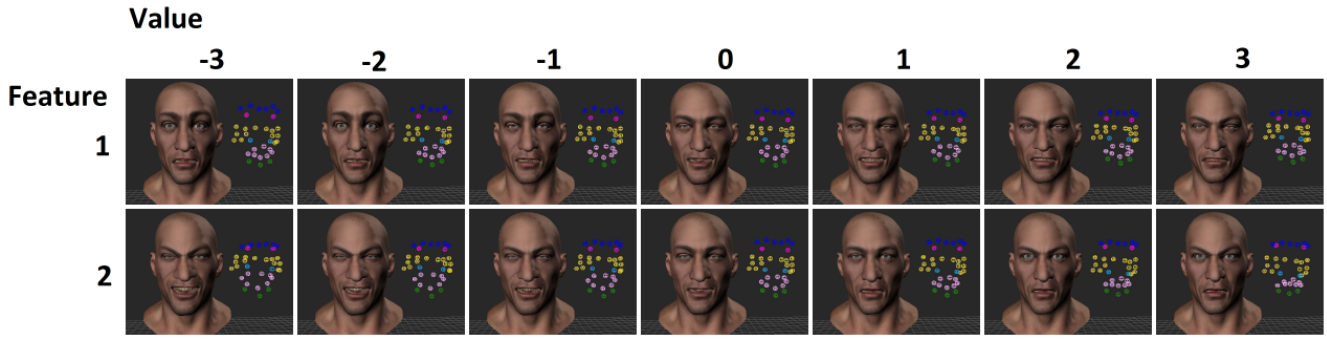
Results of the WGAN training on sad annotated data (58474 frames with a 80-20% train-test split) with 20 critic iterations per generator iteration and a latent space dimensionality of six. The rows in this image represent the influence of each individual latent space variable. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0, 0, 0] and bottom right [0, 0, 0, 0, 0, 3].

F Results of WGAN trained on angry and happy data



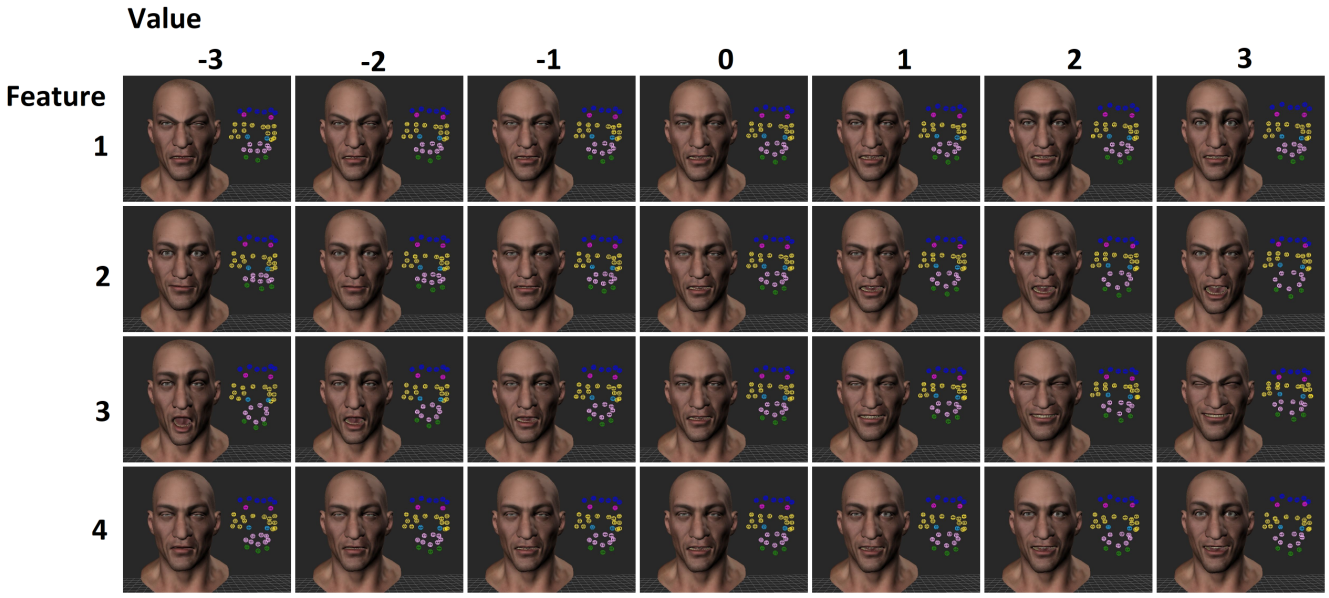
Results of the WGAN training on happy and angry data with the same number of frames per emotion (14624 per emotion both split and combined using a 80-20% train-test split). The WGAN was trained using 20 critic iterations per generator iteration and a latent space dimensionality of six. The rows in this image represent the influence of each individual latent space variable. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0, 0, 0] and bottom right [0, 0, 0, 0, 0, 3].

G Results of WGAN trained on angry and happy data with a latent space dimensionality of two



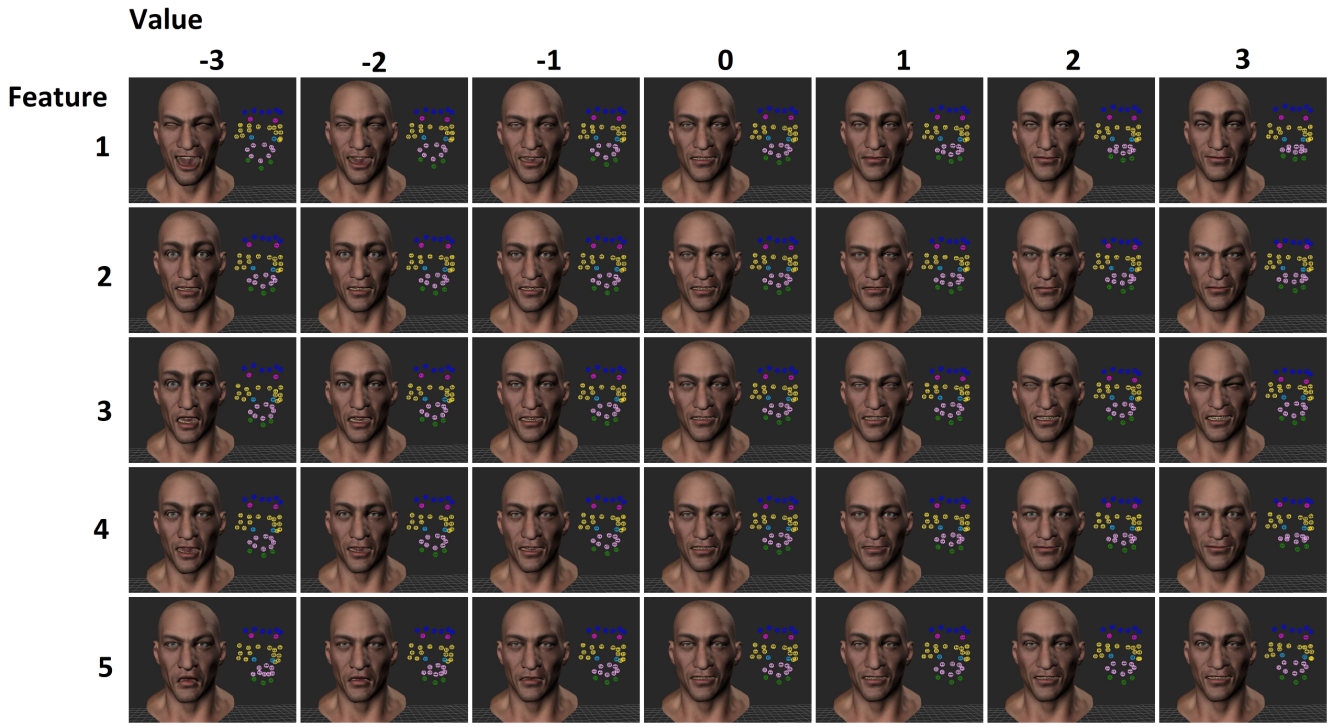
Results of the WGAN training on happy and angry data with the same number of frames per emotion (14624 per emotion both split and combined using a 80-20% train-test split). The WGAN was trained using 20 critic iterations per generator iteration and a latent space dimensionality of two. The rows in this image represent the influence of each individual latent space variables. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0] and bottom right [0, 3]. By setting the latent space to two we have the same number of latent space dimensions as there are emotions in the training data. This experiment was conducted to see how the control of the latent space variable would change. In the ideal case the latent space variables would match to an emotion, but this was not observed.

H Results of WGAN trained on angry and happy data with a latent space dimensionality of four



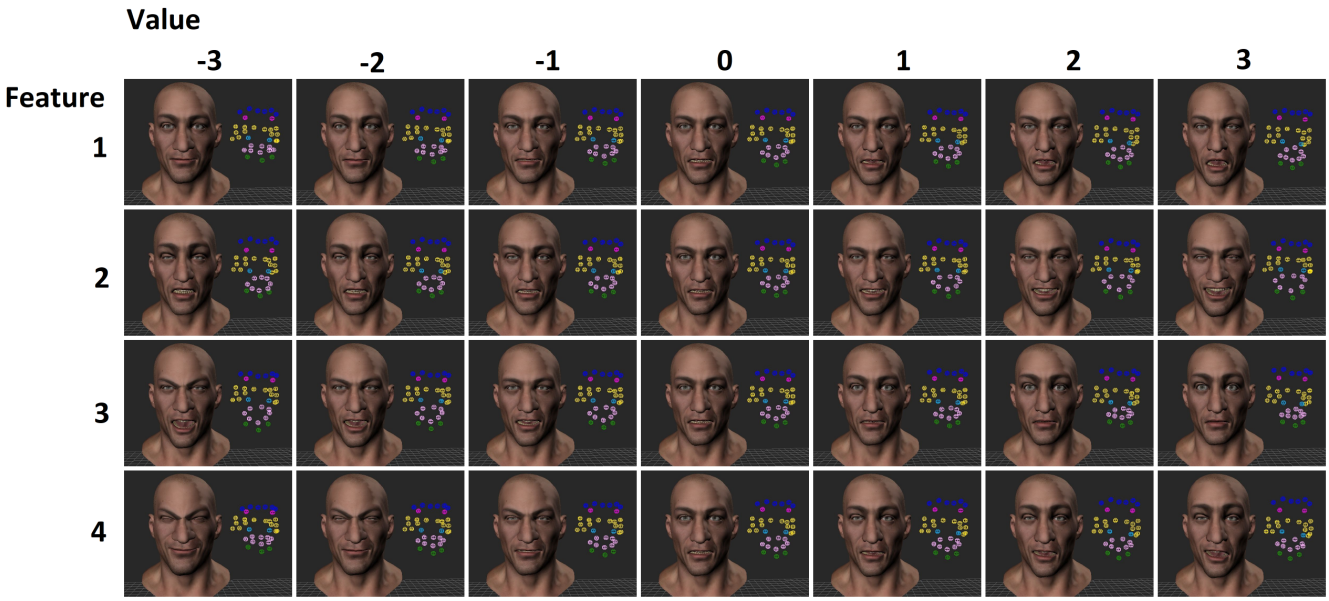
Results of the WGAN training on happy and angry data with the same number of frames per emotion (14624 per emotion both split and combined using a 80-20% train-test split). The WGAN was trained using 20 critic iterations per generator iteration and a latent space dimensionality of four. The rows in this image represent the influence of each individual latent space variables. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0] and bottom right [0, 0, 0, 3]. During the experimental evaluation we tested multiple latent space dimensionalities. It was found that when the WGAN was trained on happy and angry data, the error of the generated expressions would stabilize for latent space dimensionalities greater than four. To see how to the latent space variables would behave, we subjectively evaluated the results. We observed that this latent space dimensionality offered the best control over the expressions.

I Results of WGAN trained on all emotions



Results of the WGAN training on all emotions with the same number of frames per emotion (14624 per emotion all split and combined using a 80-20% train-test split). The WGAN was trained using 20 critic iterations per generator iteration and a latent space dimensionality of five. We chose this latent space dimensionality as this was found to be the minimum to stabilize the error of the generated expressions. The rows in this image represent the influence of each individual latent space variables. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0, 0] and bottom right [0, 0, 0, 0, 3].

J Results of WGAN trained on basic emotions



Results of the WGAN training on five basic emotions with the same number of frames per emotion (625 per emotion all split and combined using a 80-20% train-test split). The WGAN was trained using 20 critic iterations per generator iteration and a latent space dimensionality of four. The rows in this image represent the influence of each individual latent space variables. The latent space variables that are not changed are at their mean value zero. For example, the top left image will have a latent space input of [-3, 0, 0, 0] and bottom right [0, 0, 0, 3].