Generative Adversarial Networks for Improving Face Classification

JONAS NATTEN

SUPERVISOR
Morten Goodwin, PhD

University of Agder, 2017
Faculty of Engineering and Science
Department of ICT

Abstract

Facial recognition can be applied in a wide variety of cases, including entertainment purposes and biometric security. In this thesis we take a look at improving the results of an existing facial recognition approach by utilizing generative adversarial networks to improve the existing dataset.

The training data was taken from the LFW dataset[4] and was preprocessed using OpenCV[2] for face detection. The faces in the dataset was cropped and resized so every image is the same size and can easily be passed to a convolutional neural network. To the best of our knowledge no generative adversarial network approach has been applied to facial recognition by generating training data for classification with convolutional neural networks.

The proposed approach to improving face classification accuracy is not improving the classification algorithm itself but rather improving the dataset by generating more data. In this thesis we attempt to use generative adversarial networks to generate new data. We achieve an impressive accuracy of 99.42% with 3 classes, which is an improvement of 1.74% compared to not generating any new data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Due to its wide variety of real world applications facial recognition has been studied extensively in recent years. Recognizing and validating faces in video can be applied in many environments including biometric security, entertainment like Snapchat[3] has made popular.

Facial recognition is a problem that is originally very difficult to solve with a computer. Any traditional computer program is written with specific instructions that the computer has to do. This makes the task of recognizing a face in an image nearly impossible to do without some sort of machine learning. One of the reasons the task is difficult is the nearly unlimited ways that a face can appear in an image. Different illumination, different angles, also different facial expressions are a few of the variables that makes this task prone to failure in a traditional computer program.

A solution to many tasks that are difficult to do with traditional computer programs is machine learning. And in this case, more specifically deep learning. This thesis will attempt to solve facial recognition using deep artificial neural networks that can be trained to perform tasks in a similar fashion to humans, and often even better than humans.

A well known example of this is the AlphaGo[6] software developed by Google DeepMind, which were designed to play the board game Go. In March 2016, AlphaGo beat Lee Sedol in a best of five match of Go.

Deep learning works by training with examples. One of the main difficulties when working with deep learning is to acquire enough, and good data.

This thesis will go into detail on how the task of applying deep learning to facial recognition can be improved by using generative adversarial network to generate data.

1.2 Motivation

A big problem in the field of machine learning is the availability of data. Different approaches of generating, and growing datasets exists, but none are perfect.

This thesis examines if utilizing generative adversarial networks in the data generation process will improve the results of facial recognition, alone or combined with previously existing data generation techniques.

The main motivation of this dissertation is then to help add a data generation algorithm to the toolbox of machine learning developers and attempting to get one step further in the machine learning field. With a focus to improve the accuracy of face classification with convolutional neural network.

1.3 Thesis definition

This thesis will look in depth at using a few different methods to improve facial recognition accuracy with convolutional neural networks. We will attempt to improve results, without obtaining more real data to train the convolutional neural networks. This thesis will then focus on generating more data that can be used in the training process of the convolutional neural networks.

This report covers the process of building the convolutional network model, attempt at generating more data using generative adversarial networks, as well as analysis of the results. The main focus of the project is to prove or disprove the following hypothesis:

Can generative adversarial networks be used to generate data to improve the accuracy of a convolutional neural network for face classification?

1.4 Contributions

This thesis' contribution to the field would be to further the accuracy of facial recognition using deep learning. More specifically the accuracy is improved by applying generative adversarial networks to the task of generating realistic, but not real data that can be used to more effectively train a regular convolutional neural network alone, or in addition to the original data. This makes applying deep learning to facial recognition tasks where training data is limited a more doable solution.

1.5 Thesis outline

This section will briefly mention the main focus of each chapter.

In in the deep learning chapter (2) we mention and explain the theoretical background of the algorithms used in this thesis which are already known and are not at the research front at the time of writing. In the state of the art chapter (3) we take a look at the state of image recognition, facial recognition and generative adversarial networks today. After which the proposed solution chapter (4) follows, which contains an in depth explanation of the suggested approach. The testing chapter (5) presents the results from testing each approach, including both the existing ones and the thesis' proposed approach.

Chapter 2

Deep Learning

Deep learning[13] is a class of machine learning algorithms[14] that utilizes multiple layers for some sort of feature extraction. Deep Learning can in this case be thought of as a superordinate term for different types of Artificial Neural Networks. In the following sections the theoretical background for the known algorithms used in this thesis will be explained.

2.1 Artificial Neural Networks

Artificial Neural networks is an algorithm which can be used to solve numerous different tasks. Each network consists of multiple layers, where each layer consists of multiple neurons. Each neuron takes multiple inputs and combines that with its internal weights and gives an output.

Artificial neural networks in itself is rather simple algorithm, but once combined with backpropagation and some sort of optimization algorithm, often gradient descent, they can be very powerful and be an effective solution to many problems.

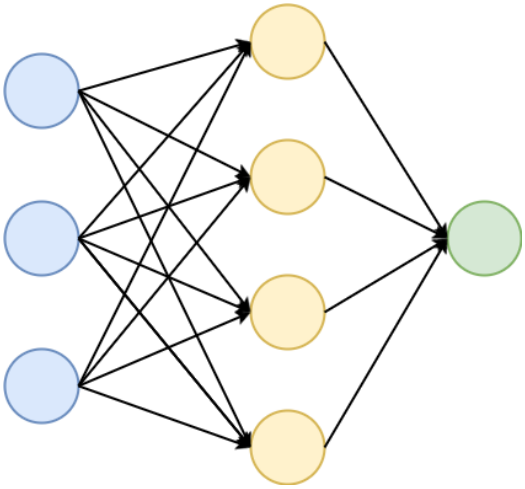

Figure 2.1: A simple multi-layer perceptron neural network where blue circles represents neurons in the input layer, yellow circles represent neurons in a hidden layer, and the green circle represents a neuron in the output layer.

2.1.1 Backpropagation

Originally artificial neural networks were simply what was described earlier, which was slow to train, and simple structures could not produce every possible output (Exclusive-or problem). These issues, and more generally the problem of quickly training multi-layer networks were solved by backpropagation. The backpropagation algorithm is a repeating two-phase cycle. The phases are propagation, and weight update.

When the network receives an input, it is propagated through the network, layer by layer until it finally reaches the output layer. The output the network generates is compared to the desired output using a loss function. Using the loss function an error value is calculated for each of the neurons in the output layer. The error values are then propagated backwards up the network until each neuron in the network has an error value which generally represents its contribution to the generated output.

Backpropagation uses the error values to calculate the gradient of the loss function with respect to the weights in the network. Then in the second phase the gradient is passed to an optimization method, which uses the gradient to update the weights in the network with the goal of minimizing the loss function.

2.2 Convolutional Neural Networks

Convolutional neural networks is a type of artificial neural network which consists of three main layer types: Convolutional Layers, Pooling Layers, and regular fully-connected layers (Same type of layer used in regular fully connected artificial neural network).

This is very similar to the previously mentioned fully connected artificial neural network, but with a few key differences. One of the differences is that the architectures make the explicit assumption that the inputs are images. This allows the network to encode certain properties into the architecture, which in turn makes the network more efficient to implement and also much faster considering the reduced amount of parameters in the network.

2.2.1 Convolutional Layers

Convolutional layers is the main feature of a convolutional neural network and they function in the way that they compute the output of neurons that are connected to local regions in the input. Each computing a dot product between their weights and a region they are connected to in the input.

A convolutional layer takes the input of a 3D volume and transforms it to another 3D volume through a differentiable function. This means that to utilize convolutional neural networks for classification (i.e., to output a vector that represents which class is correct for the input) we need to flatten the output of the last convolutional layer and connect it to one or multiple fully-connected layers.

2.2.2 Pooling Layers

Pooling layers will perform a down-sampling operation. Down-sampling operations are mainly used to decrease the complexity of the network. This is used to reduce amount of computation needed as the features and complexity decreases. Pooling layers are very useful in convolutional neural networks considering that the convolutional layers will often generate large feature maps which are slow to process. Another task the pooling layers will help with is to control overfitting, which is the case where the networks will learn the exact training data, which effectively means that the network will perform very good on the training data, but not data that the network has not seen before.

A popular pooling operation used in pooling layers is max pooling. Max Pooling will simply output the highest number in the filter, where the filter moves with the stride. Utilizing max pooling will lose some information, but has been proven to be effective in many cases.

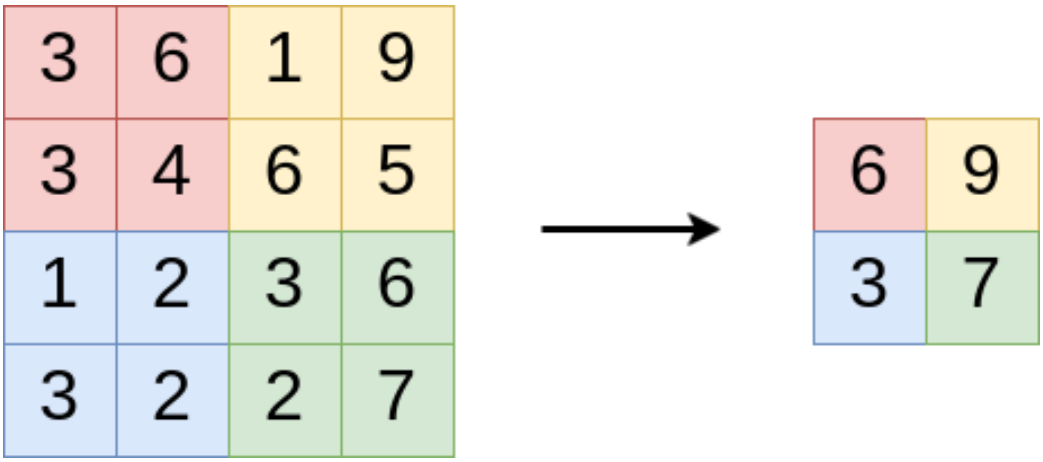

Figure 2.2: Max Pooling Operation with 2x2 filter and 2 stride.

2.2.3 Activation layers

A type layers that is also important, but would not be classified as one of the main features of the convolutional neural network, since it is used across multiple types of artificial neural networks is the activation type layers. These layers apply a function to the output of the previous layer, and does not change the shape of the output volume. A widely used activation layer that has proven to be quite effective is the rectified linear unit (ReLU) layer.

Utilizing the ReLU layer is as simple as element-wise applying:
$f(x) = max(0, x)$ meaning that any value under zero will result in zero.

ReLU layers can be very effective and is found to be greatly accelerate the convergence of stochastic gradient descent compared to other used activation functions (e.g. sigmoid/tanh)[9]. However ReLU units can be found to be quite fragile during training and can "die". By dying we mean that for example a large gradient going through a ReLU neuron could cause the weights to be updated in a way that the neuron will never activate again. If neurons "die", the gradient flowing through the neuron will forever be zero from that point on.

A few measures can be taken against the "dying" ReLU neuron problem. One of these is setting a proper learning rate, as it has been found that a too high learning rate will often result in dead neurons. Another is replacing the ReLU activation layer with a Leaky ReLU activation layer. Leaky ReLU are one of the attempts to fix the "dying" neuron problem with ReLU. Instead of having the regular $f(x) = max(0, x)$ function so that $x < 0$ returns zero, a leaky ReLU will instead have a small negative slope.

The function utilized in Leaky ReLU is $f(x) = 1(x < 0)(\alpha x)+1(x >= 0)(x)$ where α is a small constant. This has been reported to often solve the problem, but the results are not always consistent. The slope in Leaky ReLU can also be a parameter of each neuron, as has been demonstrated with PReLU neurons[8], however the benefit of utilizing this across different tasks is currently unclear.

Another type of activation that achieves similar results to ReLU is the Maxout neuron. Maxout works by generalizing ReLU and Leaky ReLU. This means that the Maxout neuron computes the function $max(w_1^T x + b_1, w_2^T x + b_2)$. This can be thought of as a special case of ReLU that has all the benefits of a ReLU unit, and does not have the "dying" neuron problem. However it does double the amount of parameters for every single neuron leading to a more expensive activation function computationally.

2.2.4 Dropout Layers

Another layer that falls into the category of important, but not necessarily one of the main layers of the network, is the dropout layers[11]. Dropout consists of randomly setting a fraction of the input units to 0 at each update during training time. This might seem counter-productive, but it has proven quite effective in the task of helping the network prevent overfitting.

Chapter 3

State of the art

There are many ways to classify images based on their content, deep learning, more specifically convolutional neural networks is one of the best methods known approaches today.

However any basic convolutional neural network does not necessarily lead to the best results. This chapter will explain some of the methods that are used today to improve the results of convolutional neural networks.

The following sections in this chapter will discuss the state of the art for datasets used in facial recognition testing, image recognition, some approaches to data generation, and generative adversarial networks.

3.1 Datasets

Deep learning, and other machine learning algorithms for recognizing and classifying faces require a relatively large amount of data to effectively do their task.

To effectively compare machine learning algorithms used for facial recognition the need for a common dataset to train and test against is needed. A dataset that is popular and very challenging to accurately classify is the LFW dataset[4]. The LFW dataset has been extensively used for testing machine learning algo-

rithms performance in the task of facial recognition, this results in multiple different baselines to compare against when one is testing a new machine learning algorithm.

A technique that is widely used before applying the face recognition algorithm is to detect facial features and ”align” the features in the image, effectively making the face appear as if it was facing the camera directly. The LFW dataset is available as four different sets whereas one of these is the original set, and the other three is different types of alignment. As the official LFW website[4] puts it:

> *The aligned images include ”funneled images”, LFW-a, which uses an unpublished method of alignment, and ”deep funneled” images. Among these, LFW-a and the deep funneled images produce superior results for most face verification algorithms over the original images and over the funneled images.*

These techniques can greatly improve the results of an algorithm, but is not necessarily the best solution in every environment, especially environments where processing power and time is limited.

3.2 Image Recognition

ResNet or Deep Residual Networks[7], developed by Kaiming He et al. is one of the networks that are considered the latest and greatest in using convolutional neural networks for image recognition. It introduces skip connections and utilizes batch normalization heavily. ResNets are currently state of the art and often the default choice when it comes to using convolutional neural networks in practice.

3.2.1 Facial Recognition

Most facial recognition algorithms is tested on the LFW dataset[4]. The dataset has a website to keep track of results of different algorithms applied to the dataset. Where we can find that the most accurate algorithm where they only used images from the LFW dataset comes from the paper "Class-Specific Kernel Fusion of Multiple Descriptors for Face Verification Using Multiscale Binarised Statistical Image Features"[1] as proposed by S. R. Arashloo and J. Kittler with an accuracy of 95.89%.

Considering the algorithms on the website are tested on the entire dataset, it is a very difficult challenge for deep learning algorithms considering many of the classes only contain a few images and deep learning often requires a lot of training data. However there are categories where they allow data from outside the dataset to be used. In the unrestricted category we can see deep learning approaches be more viable. Examples of deep learning algorithms displayed are the DeepID3[12] algorithm proposed by Y. Sun et al., and the FaceNet algorithm as proposed by F. Schroff et al. The algorithms achieve accuracies of 99.53% and 99.63% respectively. Although these approaches has similarities with the classification algorithm proposed in this thesis, they are not directly comparable, considering the data in this thesis is only taken from the LFW dataset. Another reason why a direct comparison would be wrong is that the tests in this thesis are not performed on all 5,749 classes in the dataset.

3.3 Data generation

A big limitation with many machine learning algorithms is that they require vast amounts of training data before they become effective. Deep learning is no exception. Therefore it has been attempted to create methods that generate data to expand a dataset. Some of these methods are very effective at improving training of artificial neural networks. In this section we go into a few of the more successful ones used today.

3.3.1 Data augmentation

One of the more common, and also more effective ways to obtain more data is data augmentation. There are multiple ways to augment images, but commonly used ones include: flipping images horizontally, flipping images vertically, random crops, zooms, rotations, color perturbation, and translation.

3.3.2 Fancy PCA

There are also more advanced techniques which has been deemed effective in some cases. An example of more advanced approaches is the proposed "Fancy PCA" augmentation algorithm Krizhevsky et al. used when training Alex-Net[9]. Simply put "Fancy PCA" works by altering the intensities of the RGB channels in the training images. In practice, it is firstly performed PCA on the set of RGB pixel values throughout the training data. And then, for each training sample, just add the following quantity to each RGB image pixel: $[p_1, p_2, p_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$ where, p_i is the i-th eigenvector of the 3×3 covariance matrix of RGB pixel values, and λ_i is the eigenvalue of the same matrix. α_i is a random variable drawn from a Gaussian distribution with mean zero and a standard deviation of 0.1.

3.3.3 Summarized

All of these techniques are methods of expanding your dataset through slight manipulation. Something that can be achieved using several different methods, both alone and combined. Utilizing data augmentation can multiply the amount of data several times and will often improve the results greatly.

A relatively recent example of data augmentation improving results is the "Stochastic Pooling for Regularization of Deep Convolutional Neural Networks" paper[15] where the authors achieved state of the art performance relative to other approaches that did not use data augmentation.

3.4 Generative Adversarial Networks

Another way to generate data is to use an approach that actually utilizes artificial neural networks. A very powerful way of doing this is utilizing generative adversarial networks[5] also known as “GAN”.

This consists of utilizing two Artificial Neural Networks that work against each other in a zero-sum game to generate images. The first network is called the discriminator and has the job of classifying generated as real or generated. The second network, called the generator network has the job of generating images from inputted random noise. This model, once trained, will be able to generate vast amounts of images that are similar to the training data.

The images generated by utilizing generative adversarial networks often look authentic to human observers, which leads us to the thought of using it to train convolutional neural networks to classify people in images. There are no known metrics for assessing the performance of the generative part of the generative adversarial network which makes optimizing generative adversarial networks difficult.

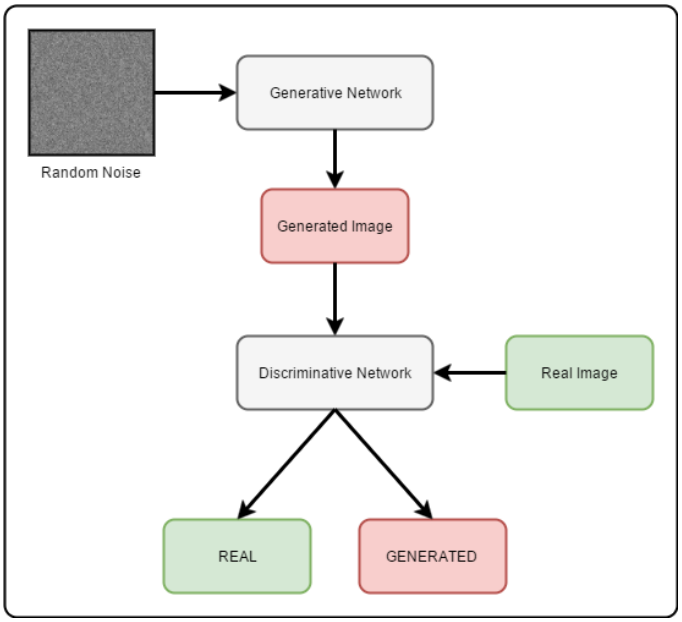

Figure 3.1: Illustrating the Generative Adversarial Model

3.4.1 Softmax GAN

Generative adversarial networks is a recent development in the field of machine learning and further development that make improvements to the approach are often released. A recent and successful development is Softmax GAN[10] by Min Lin. This approach does softmax across samples in a minibatch, and uses cross entopy loss for both the discriminator and generator. In their experiments softmax GAN were able to consistently get good results in cases where regular generative adversarial networks failed.

A regular generative adversarial networks approach is often affected by mode collapse, which is the case where the generator are mapped to a single or few images, and the generator yields low diversity in the outputted images. The paper shows that in their experiments softmax gan are less affected by mode collapse than regular generative adversarial networks.

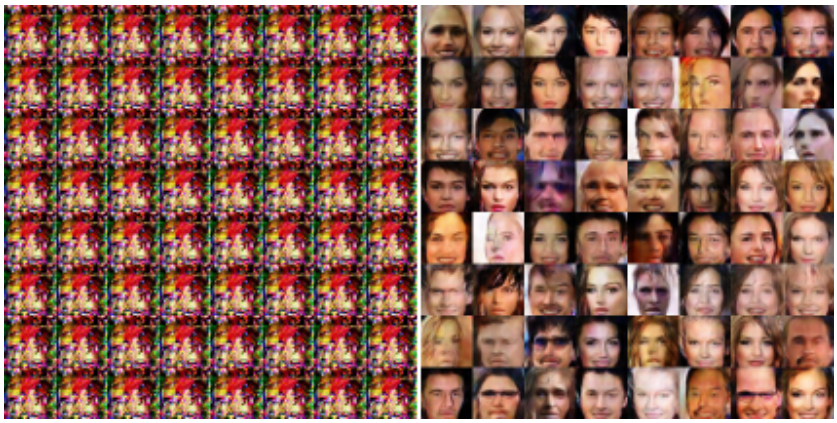

Figure 3.2: "Generator without batch normalization and with a constant number of filters at each layer. Both GAN and Softmax GAN are able to train at the early stages. However, the discriminator loss of GAN suddenly drops to zero in the 7th epoch, and the generator generates random patterns (left). Softmax GAN (right) is not affected except that the generated images are of slightly lower qualities, which could be due to the reduced number of parameters in the generator." Figure and caption from "Softmax GAN" by Min Lin[10].

Chapter 4

Proposed Solution

This chapter will go into detail on how improving face classification accuracy is proposed solved by utilizing generative adversarial networks.

4.1 Proposed solution

The task of improving facial recognition using convolutional neural networks is proposed solved by using data generation. This is to increase the amount of training data in a way that will improve the classification performance of the basic convolutional neural network. The generative method proposed in this thesis is the use of generative adversarial networks[5] to generate images of each person in the dataset. And to then use these images generated by the generative network to train and finally be able to classify images in the original dataset accurately.

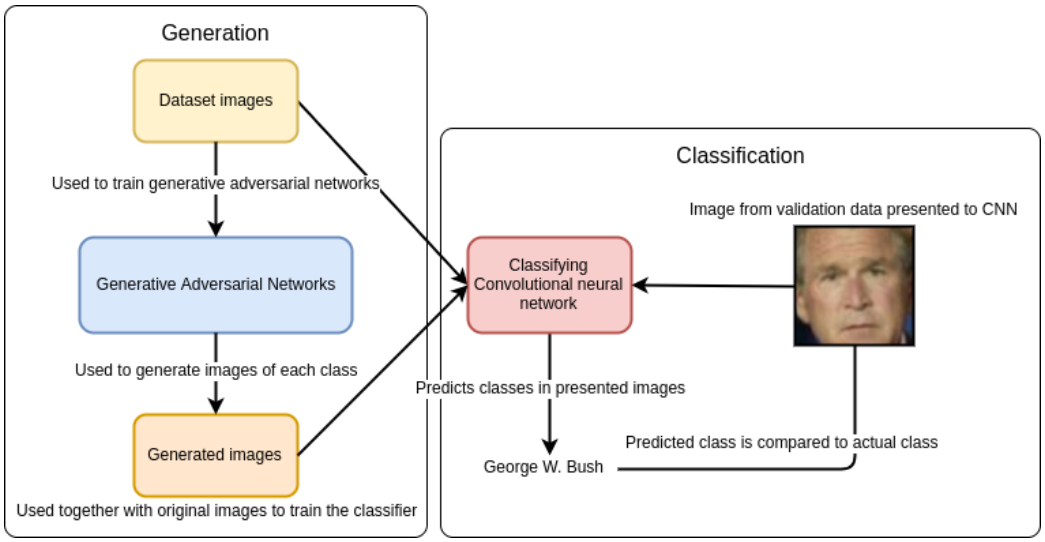

Figure 4.1: Overview of solution, which will be explained more in depth in the following sections.

4.1.1 Data generation

This thesis proposes solving data generation with generative adversarial networks. However in the case of the tested datasets, the data is limited. To improve the training of the generative adversarial networks in cases where the amount of training samples are less than optimal we use data augmentation. In our case we simply rotate the images within a margin of 10 degrees. Although humans sees these augmented images as very similar to the original image, to the convolutional neural network it looks like a very different image. The reasoning behind this is that the pixels representing each part of the images are now in a new location so the convolutional layers are more likely to learn the features instead of the exact combination of pixels.

A problem with the networks learning unnecessary features can occur with this approach if the data is limited to a few samples. An example of this problem would be that the network learned to recognize a background element in the task of facial recognition. The approach suggested in this thesis is not as prone to this error as one might suggest, considering the face detection and cropping happens before

the data augmentation happens. This limits the amount of unnecessary features in the images that the convolutional neural network can learn to recognize, leaving us with mostly important and correct features.

To achieve the best performance when detecting and cropping faces with both speed and accuracy in mind, a Haar Cascades approach is used to detect faces before cropping the images around the detected face. An illustration of this can be seen in Figure 4.2.

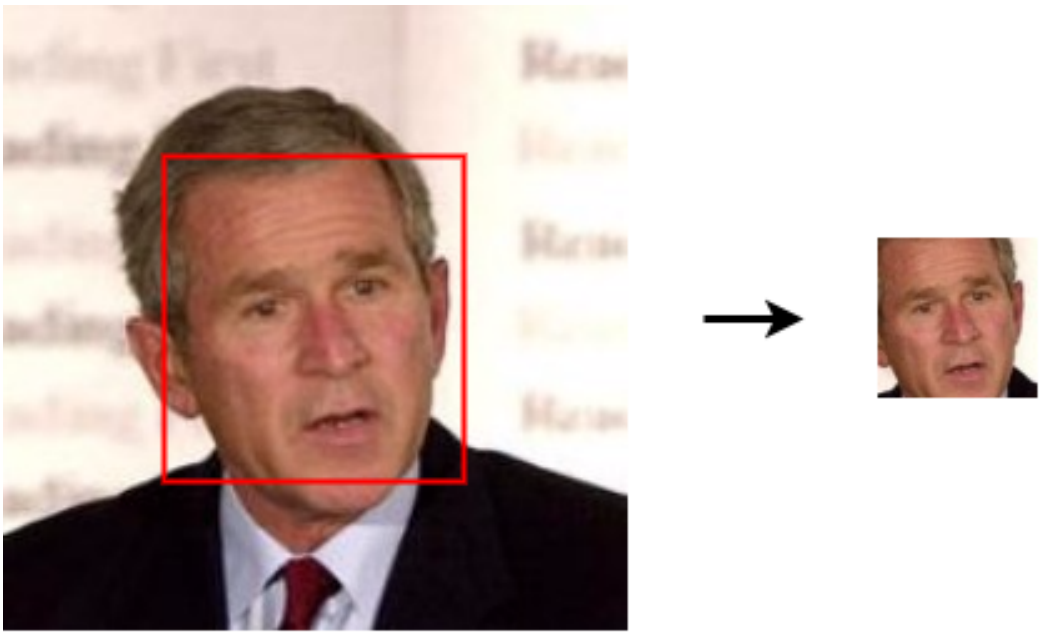

Figure 4.2: An example of an image where the face is detected with haar cascades and then cropped and resized.

4.1.2 Generative Adversarial Networks

Data generation is a big subject and can be done in quite a few different ways. A relatively recent approach to generating images in general, is to utilize generative adversarial networks. Utilizing generative adversarial networks to generate data for our classification is an interesting approach considering that it uses the same algorithm to generate the data as the classifier will use to train and classify faces with. Whether this similarity between the generative part and classification makes a difference in the results is not known.

If the generator outperforms the discriminator, the generated data will theoretically not be separable from real data to the neural networks. If this is in fact the case, the trained recognizing network will in turn be better trained and will likely be better at identifying faces.

In this thesis we rely heavily on manually inspecting the output to optimize the generative network, which might make it hard to discover small differences in performance. However assessing performance of the generative network is still an open research question, and therefore manual inspection is the approach used in this thesis. The first part of the solution is to train the generative adversarial networks to generate faces of each class in the dataset. In our case, that means one generative adversarial network pair for each person in the dataset.

A few different structures for the generative part of the network was tested in the process of creating the generative adversarial network model. When making the network too deep, it became clear that the generated images did not improve when training. In Figure 4.3 we can see an example of generated images after 130,000 epochs of training the generative adversarial networks.

One thing to notice, is when the network was too deep, the generated output after 1,000 epochs was very similar to that of 130,000, and the output of the network only varied slightly rather than creating different images as it did when a better structure was found. A theory for why this happens is because features gets too abstracted, and we simply do not have enough training data to learn the facial features. A solution for this when using deeper networks could have been softmax GANs as mentioned in the state of the art chapter (3.4.1), however the paper was published after the model and images were generated, therefore due to time constraints we were not able to implement softmax gan in this thesis.

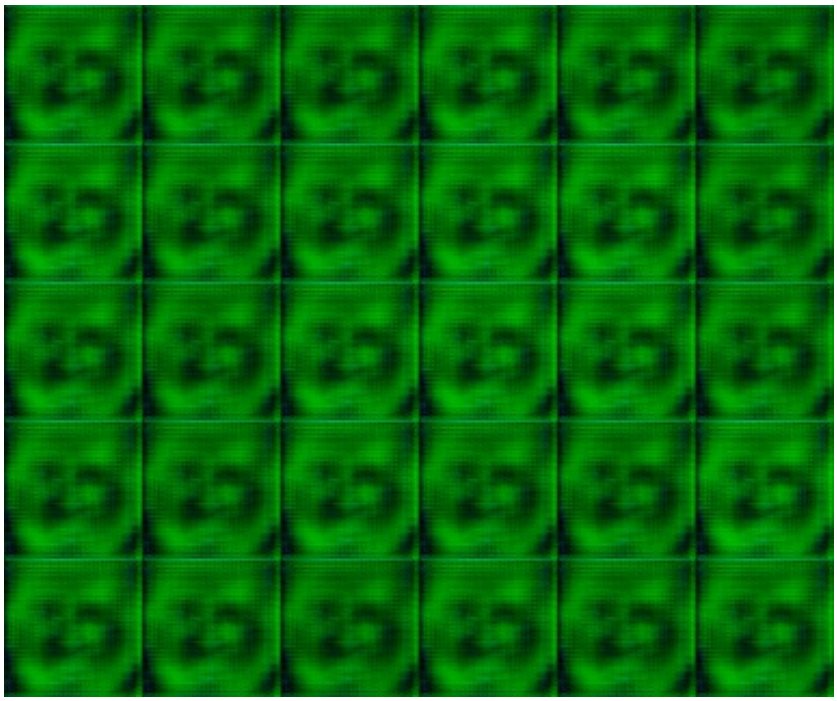

Figure 4.3: Images generated of “George W. Bush” by a generative network after 130,000 epochs of training with a too deep generative network.

In Table 4.1 and 4.2 we can see the structures that was decided on for the generative network and the discriminator network used in the generative adversarial network model respectively.

0	INPUT: 100 datapoints of uniform noise from 0-1
1	Regular Fully Connected Layer with 12,288 neurons (Same as amount of features in desired image output 64x64x3)
2	Batch normalization
3	LeakyReLU Activation Layer with α of 0.2
4	Convolutional Layer with 64 3×3 filters
5	Batch normalization
6	LeakyReLU Activation Layer with α of 0.2
7	Convolutional Layer with 128 3×3 filters
8	Batch normalization
9	LeakyReLU Activation Layer with α of 0.2
10	Deconvolution Layer with 3 3×3 filters to get desired output shape

Table 4.1: Generative Neural Network Structure.

0	INPUT: 64×64 RGB images
1	Convolutional Layer with 64 5×5 filters and a stride of 2
2	LeakyReLU Activation Layer with α of 0.2
3	Dropout Layer
4	Convolutional Layer with 128 5×5 filters and a stride of 2
5	LeakyReLU Activation Layer with α of 0.2
6	Dropout Layer
7	Regular Fully Connected Layer with 256 neurons
8	LeakyReLU Activation Layer with α of 0.2
9	Dropout Layer
10	OUTPUT: Regular Fully Connected Layer with 2 neurons (one for generated image, and one for real image)

Table 4.2: Discriminator Neural Network Structure.

Whether the structures mentioned in Table 4.1 and 4.2 are the perfect structures for generating images, are hardly likely, but they are able to generate images that are recognizable by human observers.

After training for 50,000 epochs, the faces were recognizable by human observers. Even though the images are recognizable, they are by no means perfect, and can easily be differentiated from real photographs by humans. In Figure 4.4 it can be seen that images keeps improving with training, even after 50,000 epochs. If the generative adversarial networks for each class were to be trained for more than the decided 50,000 epochs, the classifying network would most likely perform better. However it was decided that in this thesis 50,000 epochs of training were a good compromise between good looking images and short enough training time.

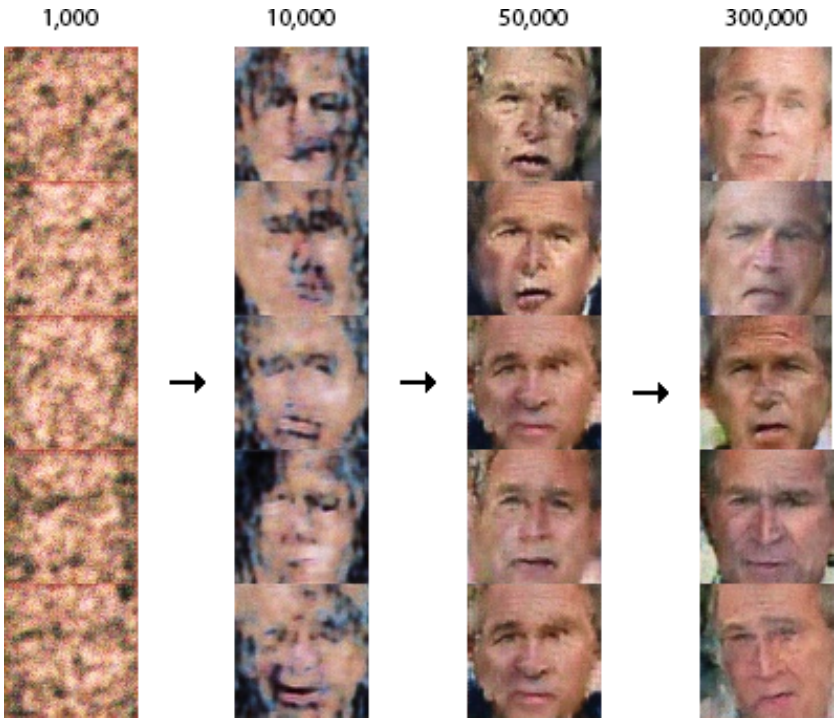

Figure 4.4: Images generated of “George W. Bush” by a generative network after x epochs of training.

4.1.3 Classification

The process of classifying faces will happen as previously mentioned with a separate convolutional neural network. This network will be structured similarly to other convolutional neural networks successfully used to classify faces. Tests are conducted with multiple different sets of data. That is unmodified data, augmented data, and data generated with generative adversarial networks. Looking at different results and analyzing the effectiveness of each approach. This will provide a good baseline of how generative adversarial networks compare to other approaches.

The structure of the convolutional neural network used for classification is simple, but also effective. It is structured similarly to other convolutional neural networks used for image classification. A compressed description of the structure is that it consists of 9 convolutional layers which are each followed by a ReLU activation function. Every filter is 3x3, and the first three layers consists of 32 filters, the next three 64 filters, and the final three 128 filters.

The structure of the convolutional neural network used for classification can be found in detail in Appendix A.

4.2 Claim to Originality

To the best of our knowledge no generative adversarial network approach has been applied to facial recognition and classification. Even though there are papers which suggest using generative adversarial networks for generating images of faces, similarly to what has been done in this thesis, none of these has been applied to the task of training a separate convolutional neural network for classification with the generated data. An example of a released paper that proposed generating faces with GAN images are the previously mentioned softmax gan [10] by M. Lin.

Chapter 5

Testing

This chapter will present experiments conducted with different types of data augmentation, including the approach suggested in previous chapters of this thesis.

5.1 Convolutional Neural Networks

Test were conducted in three main categories. The first category is without any data generation at all (5.1.1). This is done to get a general idea of what a convolutional neural network can do with as little help as possible, and to get a baseline to compare results against when applying different types of data generation. The second category test is general data augmentation (5.1.3). Where a successful and widely used data augmentation approach were tested. The third and final category is applying the solution proposed in this thesis, where we include images generated by separate generative adversarial networks in the dataset(5.1.5). Finally we summarize the results (5.1.8). The tests were performed with the 200 classes which contained most images. Other tests were also performed with only 3 classes to get an idea of how good the performance was when we know the images generated by the generative adversarial network was good. Tests were performed on limited amount of classes because of time constraints, but also because 4,069 of the 5,749 classes in the LFW dataset only contains one image.

5.1.1 No data generation approach

The convolutional neural network approach is a very powerful algorithm by itself and yielded relatively good results, even with little optimization of the parameters of the different layers in the network.

The convolutional neural network that was used in the classification part of the tests is structured like seen in Appendix A. The structure is a relatively simple convolutional neural network, which proved to be quite effective despite the network's simplicity.

Testing the classifying network without any data generation at all was a relatively simple procedure that gave surprisingly good results considering its simplicity. All the processing that was done before feeding the data to the convolutional neural network was detecting and cropping the faces from the LFW dataset. This was done with a simple haar cascades approach as described briefly in the solution chapter of the thesis (4.1.1).

5.1.2 No data generation approach results

The results of the no data generation approach was relatively good. After only 100 epochs of training the network achieved a validation accuracy of $>60\%$. Which is a relatively high number considering the large number of different classes the network has to recognize. To get a understanding of how good the network actually is, we can think of if the training process failed completely and the network guessed at random this would achieve an accuracy of about $\frac{1}{200}$ or 0.5%.

One thing that is definitely a factor that affects the results negatively is overfitting. Overfitting means that the network learns the exact data instead of learning the features we want it to learn. In our case this means that the network starts to learn every pixel in the original training data instead of learning what each person in the dataset looks like.

An indicator of overfitting can be seen when comparing the results of predicting on the training set and testing set. With the training set the accuracy is much higher than what the network achieves with unseen data as can be seen in Figure 5.1.

We can see in our results that the classifying network achieves a very high accuracy of >98% on the training data. However with the testing data the network achieves the lower accuracy of 60%. As previously mentioned this could indicate overfitting. There are many approaches to limiting overfitting, and some has even been used in the original network. One of them is the dropout layers used in the convolutional neural network. Even though dropout layers are designed to prevent overfitting, it is still difficult to completely prevent it. Especially with such a small dataset compared to the amount of classes it contains.

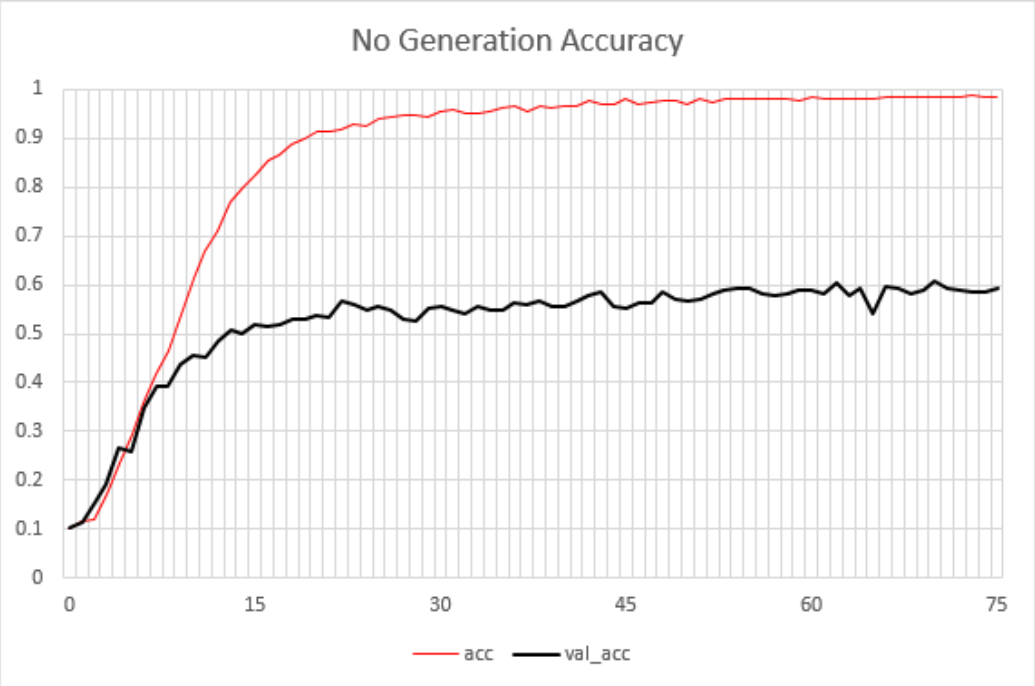

Figure 5.1: The accuracy when classifying validation data after the first 75 epochs of training CNN without any data generation on 200 most populated classes.

5.1.3 Data augmentation approach

Comparing the different types of data augmentation to the approaches without any generation is not as straight forward as one might think. When not dealing with generation of data, there is only one dataset and that is what is used. When optimizing data generation there is a large probability that it is better to utilize a larger sample for training, or change some parameters (e.g. batch size, number of epochs). This leads us to asking the question is it correct to compare results with these parameters differently set?

The tests in this thesis has been conducted without changing the network itself for the different approaches, and with as little change as possible to the different parameters. This may not yield the highest accuracy possible in every test, but will give us a more fair comparison between the different approaches.

In this approach the generated data was created by taking the original data, and rotating the images within a margin of 20 degrees, flipping the images horizontally, slight color jitter. For each epoch there were generated a sample of 50,000 images. This is a lot more than the original dataset of just a few thousand images in total. Even though the images are generated using the originals, the bigger sample size per epoch is probably one of the biggest factors playing into the performance gain.

Testing the classification process with a simple convolutional neural network, a generated sample size of 50,000 images per batch, we can see clear improvements in accuracy when utilizing data augmentation. A big part of the improvements we can see in the results presented in this section of the thesis, can be accredited data augmentation because of its effect on overfitting. When generating images that are slightly different from one another the network effectively sees different data so instead of learning the exact images, as it does when overfitting, the network learns the features in the pictures which impacts performance drastically in a positive way when testing with unseen images.

5.1.4 Data augmentation approach results

The results of the regular data augmentation approach was overwhelmingly good. The improvement over doing no data generation is very prominent in that it is so much better with a relatively small effort. When the only augmentation done is the rotation within a margin of 10 degrees, and generating a sample size of 50,000 for each epoch, the results go from the original 50% to an impressive >84% on unseen validation data. The network seems to reach some of its best performance after only a few epochs. It can be seen in Figure 5.2 that is a graph of the 75 first epochs of training that the validation results plateau after only about 8 epochs. This indicates that the actual training process does not seem to benefit from long training periods over many epochs.

It can be seen in Figure 5.2 that overfitting has been dramatically reduced from the no generation approach, which is indicated by both the much higher validation accuracy, but also the lower training accuracy.

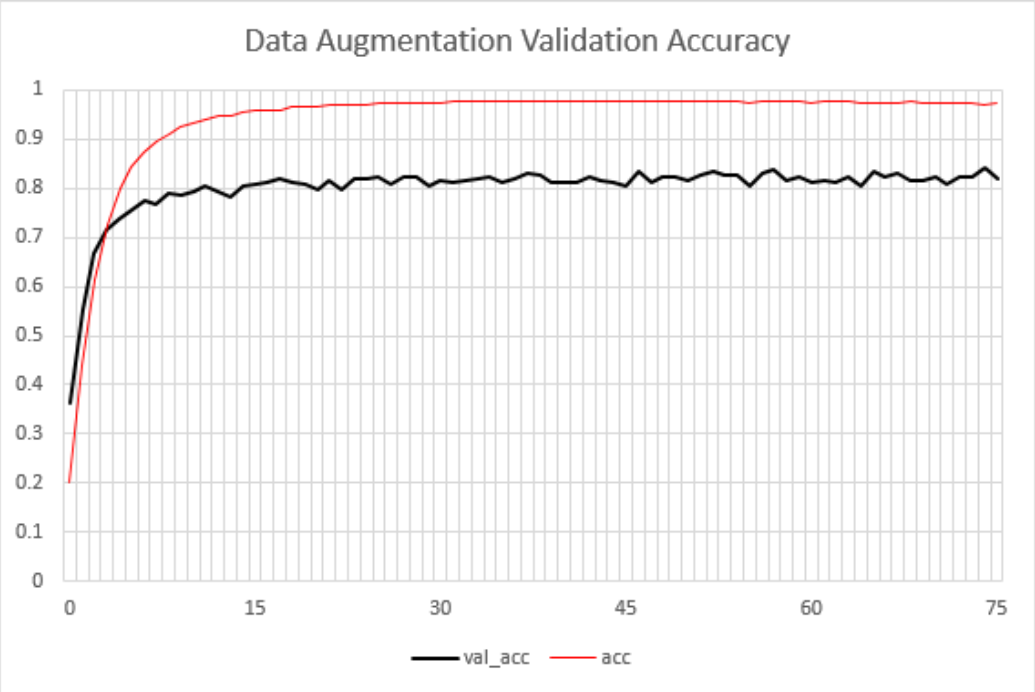

Figure 5.2: The accuracy when classifying validation data after the first 75 epochs of training CNN with data augmentation.

5.1.5 GAN data augmentation approach

The generative adversarial network generation approach is one that looks promising and could probably be utilized to achieve performance better than those of no data generation. However in this thesis we can see that generating the data from generative adversarial network actually performs worse than both the original network with no data generation in most cases, and especially worse than the data augmentation approach.

An issue that can be seen is the massive drop in accuracy when predicting the testing data compared to the training data. Whilst this is usually an indication of overfitting, it is not necessarily true in this case. Since the images in the training set in this case are actually generated images from the generative adversarial networks, there might be reason to believe that the generated images simply does not represent the actual faces in the original data well enough.

A reason for why the generated images might not be good enough for the convolutional neural network to learn facial features could be that there is very limited diversity in the generated images for the classes that contain few original samples. Obviously the diversity is limited by the amount of images in the original dataset, however there are indications that the generative adversarial networks for the classes with fewer samples are affected by mode collapse. Which is what occurs when generative adversarial networks are "locked" to a single or few images and achieves low diversity in generated output.

5.1.6 GAN data augmentation approach results

The highest achieved validation accuracy training only using the images generated by the generative adversarial networks is 40%. The images generated does then at least improve the network from random guessing, however it does not provide better results than either of previously known approaches.

As mentioned we see that the highest achieved accuracy on the training data is much higher at >99.7% than that of the testing data at 40%. While this is usually an indicator of overfitting, there is reason to believe that in this case it is mostly due to the generated images not being good enough for achieving a higher accuracy.

The images does look like the faces in the validation data, however there are some obvious artifacts that might make a convolutional neural network perform worse.

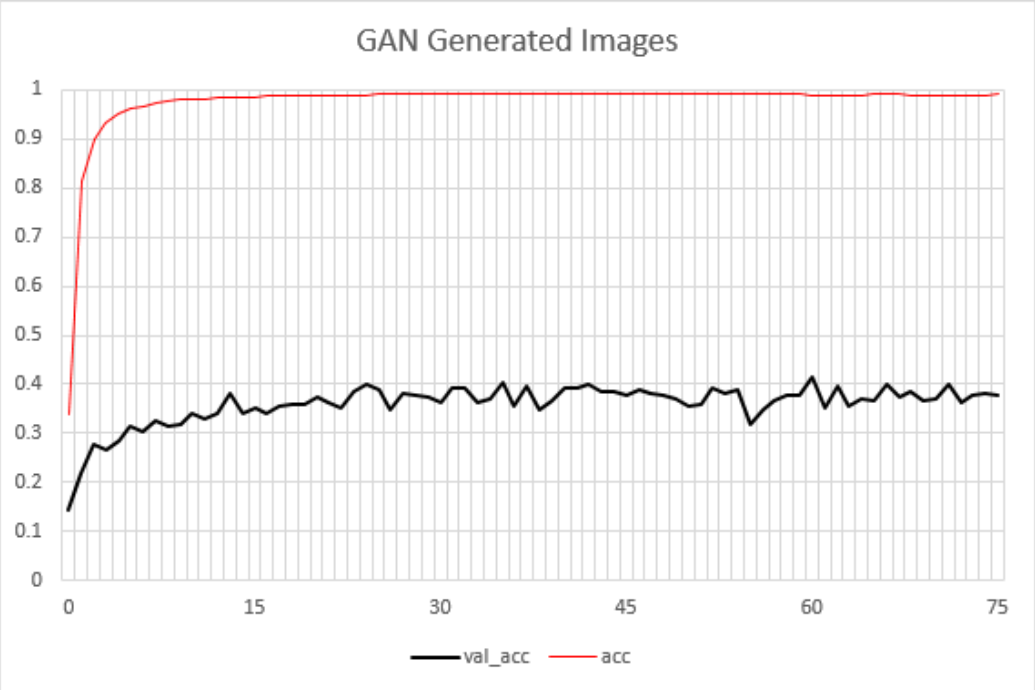

Figure 5.3: The accuracy when classifying validation data after the first 75 epochs of training CNN with GAN generated images.

5.1.7 GAN data augmentation with few classes

Considering the limited time and also the big difference in amount of images in each class, a test were also conducted where fewer classes were included in the test. This was done to see if the better generated images would improve classification in some ways. These classes were the ones with most original training data, which is why these generated images looked better than for the classes with fewer original samples.

In one test with only 3 classes, we attempted after first training the classifying network for 100 epochs to train the same network with data generated with the generative adversarial networks for another 100 epochs. This lead to the network classifying the validation data with a 99.42% accuracy compared to achieving 97.68% accuracy in 200 epochs of training without utilizing any data generation. Although this test does not actually prove that the technique is effective; it does indicate that in some cases the GAN approach might be viable.

The training with the original dataset shows signs of overfitting and when looking at the results presented in this section there is reason to believe that the GAN approach is a viable approach to improving accuracy in cases where the amount of data is sufficient to train the generative adversarial networks, but using only the original data results in overfitting.

It can be seen in Figure 5.4 that although accuracy is overall higher after training for a few epochs with the generated images, it appears that the stability of the accuracy over epochs is somewhat worse. It is likely that the instability is very prominent in the graph because of the low amount of samples in the validation data, so a slight swing in performance can be easily seen in the graph. Whether this stability will outweigh the performance gain in a real world scenario is difficult to predict, but given that some validation data is available to perform a few tests, it is likely that it would not matter.

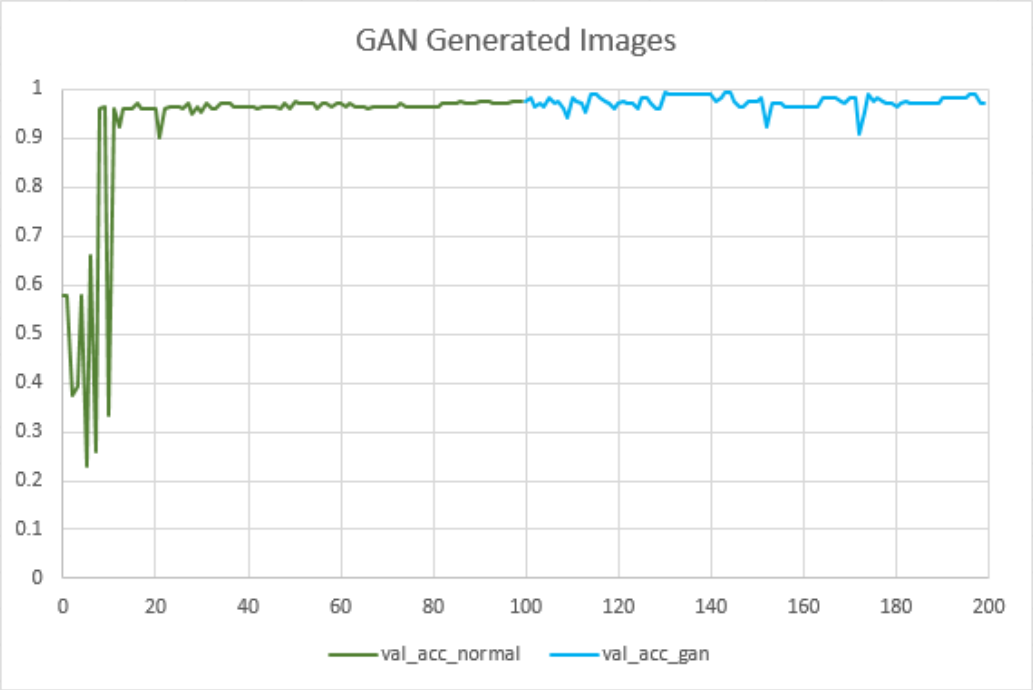

Figure 5.4: Accuracy when classifying validation data when first training the CNN with normal images for 100 epochs, then training it with GAN generated images for 100 more epochs.

5.1.8 Summary of results

It can be seen that there are some indication that the GAN approach to generating images are able to improve classification results. To summarize the results we present a table for easy comparison of the testing.

We can see in Table 5.1 that when training with the images generate from the generative adversarial networks after first training 100 epochs with data from the original dataset, we achieve almost as good results as traditional data augmentation approaches, and we outperform no data generation. This is a strong indication that with some optimization the generative adversarial network approach would be viable to implement in cases where there are sufficient amounts of data to train the generative adversarial networks, but training with the original network results in overfitting.

Approach	**Num Classes**	**Epochs**	**Test Accuracy**
No data generation	200	200	66.35%
Regular data augmentation	200	200	83.85%
GAN Images	200	200	40.00%
No data generation	3	200	97.68%
Regular data augmentation	3	200	100.00%
GAN Images	3	200	84.39%
GAN Images after 100 epochs*	3	200	99.42%

Table 5.1: Summary of testing results. *This is the result after first training 100 epochs with no data generation and afterwards training the same classifier network for 100 epochs with images generated with GAN.

Chapter 6

Conclusion and further work

In this chapter we summarize the findings, and give some suggestions for future work.

6.1 Conclusion

In this thesis we have utilized generative adversarial networks to attempt to improve accuracy of a convolutional neural network classifier for faces. This was successful in the sense that images are generated successfully, and the faces in the generated images look like the classes in the original dataset to human observers. In one case the generated images were able to improve results by 1.74% which is a drastic improvement. This is indication that the approach has the potential to be viable after further research and optimization.

Although the approach was able to improve results in some cases, it can be seen that the generative adversarial network approach for generating training data to improve accuracy of facial recognition using convolutional neural networks has yet to consistently improve the results of classification; especially in cases where the amount of training data is not sufficient for training the generative adversarial networks adequately.

6.2 Contributions

We learn that using generative adversarial networks to generate data used for training convolutional neural networks is working, and we also see indications that it might even be a preferable approach in some scenarios. These scenarios include those where training with the original dataset results in overfitting, but the amount of data still is sufficient for training the generative adversarial networks.

6.3 Further Work

For the future there are a few options that are very viable approaches to improving the results produced by the algorithm proposed in this thesis. In this section we will mention a few.

6.3.1 Image generation

The main issue with the approach as it is implemented in this thesis, is that generated images are not sufficiently good to help improve the accuracy for all classes. Especially those with few amount of images in the original dataset. The most obvious approach to improve the generated images would be to optimize the network parameters. Both the layers in the network, and the parameters of each layer is not perfect. It is likely that both of these elements have a lot of improvement potential. This would most likely lead to better generated images, and therefore also better performance of the classification convolutional neural network. The classification networks layers and parameters are also relatively untested, and likely not perfect. If the only goal of the thesis was to improve performance of facial recognition and not attempt to improve the data itself, optimizing the classifying network itself would be a much higher priority.

6.3.2 Image diversity

Another improvement that would be a natural step to further the results, would be to implement the techniques proposed in the "Softmax GAN" paper by M. Lin[10] in an attempt to minimize mode collapse, and generate more diverse images for the classes where the limited data is an issue. An example of this can be seen in the state of the art chapter (3.4.1).

Bibliography

[1] S. R. Arashloo and J. Kittler, "Class-specific kernel fusion of multiple descriptors for face verification using multiscale binarised statistical image features," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2100–2109, Dec 2014.

[2] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Evan Spiegel, Bobby Murphy, Reggie Brown, "Snapchat," [Online; accessed 06-February-2017]. [Online]. Available: https://www.snapchat.com/

[4] Gary B. Huang, Manu Ramesh, Tamara Berg, Erik Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," October 2007.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *ArXiv e-prints*, June 2014. [Online]. Available: http://adsabs.harvard.edu/abs/2014arXiv1406.2661G

[6] Google DeepMind, "AlphaGo," [Online; accessed 06-February-2017]. [Online]. Available: https://deepmind.com/research/alphago/

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[8] S. R. Kaiming He, Xiangyu Zhang and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015. [Online]. Available: http://arxiv.org/abs/1502.01852

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[10] M. Lin, "Softmax GAN," *ArXiv e-prints*, Apr. 2017. [Online]. Available: http://adsabs.harvard.edu/abs/2017arXiv170406191L

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[12] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *CoRR*, vol. abs/1502.00873, 2015. [Online]. Available: http://arxiv.org/abs/1502.00873

[13] Wikipedia, "Deep learning - Wikipedia, the free encyclopedia," 2017, [Online; accessed 06-February-2017]. [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning

[14] ——, "Machine learning - Wikipedia, the free encyclopedia," 2017, [Online; accessed 06-February-2017]. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

[15] M. D. Zeiler and R. Fergus, "Stochastic Pooling for Regularization of Deep Convolutional Neural Networks," *ArXiv e-prints*, Jan. 2013. [Online]. Available: http://adsabs.harvard.edu/abs/2013arXiv1301.3557Z

Appendices

Appendix A

Convolutional Neural Network Structure used for classification

0	INPUT Image is 64×64 RGB images
1	Convolutional Layer with 32 3×3 filters
2	ReLU Activation Layer
3	Convolutional Layer with 32 3×3 filters
4	ReLU Activation Layer
5	Convolutional Layer with 32 3×3 filters
6	ReLU Activation Layer
7	Dropout Layer
8	2×2 MaxPooling Layer
9	Convolutional Layer with 64 3×3 filters
10	ReLU Activation Layer
11	Convolutional Layer with 64 3×3 filters
12	ReLU Activation Layer
13	Convolutional Layer with 64 3×3 filters
14	ReLU Activation Layer
15	Dropout Layer
16	2×2 MaxPooling Layer
17	Convolutional Layer with 128 3×3 filters
18	ReLU Activation Layer
19	Convolutional Layer with 128 3×3 filters
20	ReLU Activation Layer
21	Convolutional Layer with 128 3×3 filters
22	ReLU Activation Layer
23	Dropout Layer
24	2×2 MaxPooling Layer
25	Regular Fully Connected Layer with 256 neurons
26	ReLU Activation Layer
27	Dropout Layer
28	Regular Fully Connected Layer with 128 neurons
29	ReLU Activation Layer
30	Dropout Layer
31	OUTPUT: Regular Fully Connected Layer with 200 neurons

Table A.1: Classification Convolutional Neural Network Structure

Appendix B

URL to Git repository

Source code can be found in following repository:

`https://bitbucket.org/deeplm/dlm`