Acknowledgments

We would like to thank Axis Communications for the opportunity to do this thesis at Axis, and supplying necessary hardware and software. We would also like to thank our supervisors, Jiandan Chen, Martin Ljungqvist och Mikael Nilsson, providing invaluable advise and guidance. Finally, we would like to extend our thanks to the 313 Axis employees who provided their consent to use their employee photos in this thesis.

Abstract

Achieving high performance face recognition often requires large manually labeled training datasets. As such datasets can be difficult to obtain, we investigate whether smaller datasets can be augmented synthetically in order to increase performance.

We use 3D morphable models to create 3D reconstructions of faces from only a single image. The 3D reconstructions are used to render new face images in different poses in order to augment the original dataset. We also investigate whether generative adversarial networks (GANs) can be used to create completely synthetic training datasets for face recognition.

We show that recognition performance can be improved for non-frontal images when augmenting with similarly posed synthetic images. Quality over quantity is found to be one of the most important aspects of the synthesizing procedure, where few high quality synthetic images perform better than many low quality synthetic images. We conclude that if higher quality reconstructions are achieved, the performance could be further improved. For future work, GANs seem promising for the task at hand.

Contents

Glossary

AP *Average Precision*. 28, 42, 45, 52–58, 61

AUC *area under the curve*. 26, 29, 37–40, 50, 52

BEGAN *Boundary Equilibrium Generative Adversarial Network*. 4, 7, 19–21, 23, 35, 36, 40, 66, 72

CelebA *Large Scale CelebFace Attributes*. 23, 35

CMC *Cumulative Match Characteristic*. 28, 49–51, 57, 64

GAN *Generative Adversarial Network*. 4–6, 18, 19, 21, 24, 72

landmark A characteristic point in a face, such as the corner of the eye, the tip of nose, or the corner of the mouth. 8, 9, 21, 22, 29

LFW *Labeled Faces in the Wild*. 16, 23–26, 38–40

MFM *Million Faces Model*. 5, 24, 29, 33, 34, 38, 39, 46, 47, 49–52, 55–58, 65, 71, 72

PnP Perspective-n-Point, A common computer vision problem where one has a set of 3D-points and a corresponding set of 2D-points, and wants to find rotation and translation matrices so that the projection error of the 3D-points on to a 2D-plane is minimized when using a projection matrix C. 29

ROC *Receiver Operator Characteristic*. 26, 29, 37–40, 49–52

SFM *Surrey Face Model*. 5, 21, 24, 29, 30, 34, 38, 39, 41, 43, 46, 47, 49–52, 55, 57, 58, 64–66, 71, 72

SVD *Singular Value Decomposition*. 8

vertex A point of a 3D-model mesh, which contain various attributes such as 3D-coordinates. 5, 7–10, 29, 30

YTF *YouTube Faces Database*. 16

Chapter 1

Introduction

Identifying faces is a daunting task, tackled by many. Modern algorithms, such as Google FaceNet [34], utilizing deep learning architectures are able to achieve great recognition performance. However, in order to properly train these deep neural networks, very large datasets are needed. In the case of [34], 200 million images were used in the training. Unfortunately, due to the immense time and resources needed to construct these types of datasets, this task has become a significant bottleneck in a lot of facial recognition research. This is addressed in the paper [22], where the authors point out that the majority of the top performing models are created by large corporations, that have the resources needed to collect immense datasets. However, these large corporately collected datasets are seldom released for academic use and publicly available datasets are typically much smaller.

In order to mitigate this problem, *synthetic* data may be used. This may include 2D augmentations i.e. in plane rotations, skews, and mirroring. However, using 3D augmentations allows for more variation and rapid expansion of training and testing data, as both pose, camera parameters and facial expression can be used to perform the augmentation.

The core of this report entails augmentation in 3D-space, a very active area of research in the last couple of years. Using a sparse set of 2D facial-landmarks, a *3D-morphable model* of a face is fitted to the landmarks. From this model, an arbitrary number of poses can be applied to each face, thus expanding the dataset drastically. This thesis also investigates how 3D-augmentation might be applied in order to the increase the accuracy of facial recognition, without the tremendous task of manually labeling large datasets.

As *Generative Adversarial Network*s (GANs) have improved quickly since their introduction in [8], it might also be possible to use such networks to create completely synthetic datasets. A recent variant called the *Boundary Equilibrium Generative Adversarial Network* (BEGAN) [4] seems to be the current state of the art when it comes to image generation.

1.1 Main Objective

The goal of this thesis is to investigate how recently suggested data augmentation procedures might be applied in order to improve facial recognition and verification performance. By adjusting pitch, yaw, and facial expression of fitted 3D-models, the number of possible augmentations is very large. However, the augmentation schemes are only helpful if the recognition or verification rates can be improved upon. If the augmen-

tations are applied non-optimally, no improvement, or even worse performance might occur, which makes it imperative to determine what separates a 'good augmentation' from a 'bad augmentation'. Therefore, this thesis investigates a number of recent augmentation techniques and tries to determine how synthesized images should be applied when performing data augmentation to actually make an improvement in the recognition or verification performance.

1.2 Related Work

1.2.1 Effective Face Recognition

In the paper [22] some of the problems of current day facial recognition systems are outlined, as well as new synthesis techniques for faces. There is a possibility to get very high accuracy using current systems, such as Google's FaceNet [34], but at the cost of immense quantities of training data. The problem can be somewhat eased by introduction of augmentation methods, such as mirroring, noise introduction, as well as oversampling, i.e. offsetting the face crop.

Although facial synthesis has been used in prior works, such as [12], this paper proposes to use the technique in order to introduce more variability in both pose and facial expression. In this thesis work, the method in [22] will be referred to as the *Million Faces Model* (MFM).

1.2.2 Morphable Face Models

A Morphable model is statistical approach to fit a 3D-shape to a 2D-image. In [25], the *Basel Face Model* is introduced and has been a basis for further morphable model research, such as [22], [14], and [3]. Although depicting a large portion of the face and allows fitting to both shape- and color spaces, the released model is very restrictive, with little documentation of its features and no framework to fit unknown faces to the model. In [3] a fitting framework is built up using a deep-learning approach, but due to resolution limitations in the color fitting process, high-frequency color-information is lost, making it unsuitable to use the model for the goals of this thesis.

In [17] a novel 3D morphable model is presented, called the *Surrey Face Model* (SFM). In essence, hundreds of face scans are used to create a generic face model and a set of *principal components* describing a face. With this generic face and principal components as a basis, the face-model can be fitted to sets of 2D facial landmarks. Characteristics of a face such as pose, shape and facial expression can be divided into separate parameters, making the fitting process and especially further augmentation simpler. However training such a model is very difficult and as such, the use of these kinds of models are very limited. Unlike earlier but similar models, such as [25], the SFM [17] is released for multiple resolutions, ranging from 3448 - 29587 vertices. A fitting framework is also released to further promote the use of this model.

1.2.3 Generative Adversarial Networks

While deep neural networks have been successfully deployed in the task of classification, a new trend in the machine learning world is to use neural networks to generate new images, using a model called GAN. First mentioned in [8], this network architecture consist of two sub-networks. A generator $G(\mathbf{z})$ that aims, given some input $\mathbf{z}$, to generate a synthetic image very similar to real images from some training dataset. Meanwhile, a discriminator $D(G(\mathbf{z}))$ is being tasked with separating the synthetic images from the

training data. Once the discriminator is unable to tell synthetic from real images, the model has converged. In the work by [6], the generative model is extended and the aim of the paper is to create a generative model where the input represents easily interpretable information, such as which number to generate if applied on the MNIST-digit dataset, or facial characteristics such as "smile", "man" or "beard" in the case of facial generation. In the article [16], the focus has been on the task of synthesizing frontal faces from extreme face poses, such as profile faces, while still maintaining the identity of the original face, using several loss functions to encourage this behavior. The very recent paper [4] describes a novel GAN-architecture, which introduces a new equilibrium concept in order to balance the generator and discriminator functions during training, as well as a novel method to determine convergence of GANs.

Chapter 2

Theory

This chapter first delves into 3D-morphable models, which are one of the methods prominently used throughout the thesis for the task of image synthesis. An overview of neural networks are then given as they are used to perform face verification, recognition, detection and generation. In particular architectural design choices such as inception modules and residual blocks are presented along with different loss functions used during face verification training. The chapter ends with a description of how neural networks are used to perform image generation, with a focus on the BEGAN-architecture.

2.1 Image Synthesis

2.1.1 3D Morphable Model

A 3D Morphable Model is a statistical model of a human face. Using only a few facial landmarks, the model can be fitted against any face. The task is however not trivial, and contains many parameters. This section will first describe the basics of morphable models, then continue with algorithms used to fit the model to an arbitrary face. A 3D-model, $\mathbf{S}$, can be seen as a collection of points in $\mathbb{R}^3$ that spans triangles between them. However, a high dimensional interpretation of the 3D-model is a linear combination of basisvectors in $\mathbb{R}^{3N}$, where N are the number of 3D-points, also called vertices. Formally speaking, the 3D-model can be expressed as

$$\mathbf{S} \in \mathbb{R}^{3N}. \tag{2.1}$$

In the morphable model used in [17], faces of 169 identities were scanned and expressed in the same notations as in 2.1. We denote the 169 scanned faces as M. All faces share similar characteristics, and by utilizing those characteristics, a new basis describing a face can be found. *Principal Component Analysis* can be used to find those facial similarities, described in [24]. First the M scanned faces, in this particular case 169, are expressed in the notation in Eq. 2.1 and brought together in a matrix A. A is a $M \times 3N$ matrix, where each column is a face. The scanned faces can be interpreted as a set of vectors that span a linear subspace to $\mathbb{R}^{3N}$ with M basis vectors. From these scanned faces, a *mean* face, $\mathbf{v_0}$, can be constructed as

$$\mathbf{v_0} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{S_i} \tag{2.2}$$

where $\mathbf{S_i}$ is the i^{th} scanned face and M is the number of scanned faces. Now, $\mathbf{v_0}$ is subtracted from each face in A, thereby moving the origin of the scanned faces to

the centroid of the face-data. This can be seen as reducing the linear subspace by 1 dimension as the most principal component is effectively removed. So now A spans a subspace of $M-1$ basis-vectors.

The problem is now to find the principal vectors of A, and retrieve a basis of *principal components* for a face. This can be solved by applying a *Singular Value Decomposition* (SVD) on A, where the singular vectors $\mathbf{V_i}$ and corresponding standard deviations σ_i form a new basis

$$\mathbf{S}(\boldsymbol{\alpha}) = \mathbf{v_0} + \sum_{\mathbf{i}}^{\mathbf{M-1}} \sigma_\mathbf{i}\alpha_\mathbf{i}\mathbf{V_i} \tag{2.3}$$

where $\mathbf{v_0}$ is the "mean" face and $\mathbf{V}_i$ is the i^{th} principal component, $M-1$ is the number of principal components used. σ_i is the standard deviation corresponding to the i^{th} principal component and α_i are the individual principal component weights.

When the fitting procedure is conducted, the weight coefficients, expressed as $\boldsymbol{\alpha}$ are varied to fit the shape to an input image, as seen in the PhD work [14]. Finally a subset of the 3D-vertices are mapped to a specific landmark point, which in the this model are the 68 *ibug* landmarks, from the work in [31]. The actual fitting works from coarse to fine adjustments, from global transforms to fit the pose, to finer per-vertex augmentations later on.

2.1.2 Camera Fitting

The first part of the fitting framework is to estimate a camera matrix, seen in the works [2] and [14]. This is done by usage of the *Gold standard algorithm*, described in [11]. Firstly, a small preprocessing step occurs as both 3D-model points in $\mathbb{R}^3$ and the image landmark points, in $\mathbb{R}^2$, are normalized and expanded into their respective projective spaces, $\mathbb{P}^3$ and $\mathbb{P}^2$. The model- and image centers are translated by similarity transforms $\mathbf{T}$ and $\mathbf{U}$ so that the Root Mean-Square distance between the points and the center are $\sqrt{3}$ for the model and $\sqrt{2}$ for the image. Then a normalized affine camera matrix $\tilde{\mathbf{P}}$ is calculated, using 4 or more point correspondences. In the fitting framework developed in [17], this is implemented with a Direct Linear Transform, also described in [11]. Lastly, the camera matrix is un-normalized again as $\mathbf{P} = \mathbf{T^{-1}}\tilde{\mathbf{P}}\mathbf{U}$. The 3D-model and corresponding 2D-landmark can be considered to be aligned with each other, and the next part of the fitting procedure is commenced.

2.1.3 Shape Fitting

The 3D-model and the correspondence are now in alignment, and the next step of the fitting algorithm is finding the shape parameters. Finding the optimal shape parameter $\boldsymbol{\alpha}$ from Eq. 2.3 is done by minimizing the shape-cost function

$$E(\boldsymbol{\alpha}) = \sum_{i=1}^{3N} \frac{(y_{m2D,i} - y_i)^2}{(2\sigma_{2D}^2)} + \lambda\|\boldsymbol{\alpha}\|_2^2 \tag{2.4}$$

where y_i is the *i:th* landmark point, and $\boldsymbol{\alpha}$ are the shape parameters. y_{m2D} are the projected 3D-points, using the previously estimated camera $\mathbf{P}$. In other words

$$y_{m2D} = \mathbf{P}\mathbf{S}(\boldsymbol{\alpha}) \tag{2.5}$$

where $\mathbf{P}$ is the estimated camera matrix and $\mathbf{S}(\boldsymbol{\alpha})$ the currently fitted face, generated using Eq. 2.3. The term $\|\boldsymbol{\alpha}\|_2^2$ is a *regularization* term, used to prevent overfitting of

the shape by discouraging large values in $\boldsymbol{\alpha}$, this is further described in [5]. λ controls the influence of the regularizer. Finally, σ_{2D}^2 is an optional variance parameter of the landmark points.

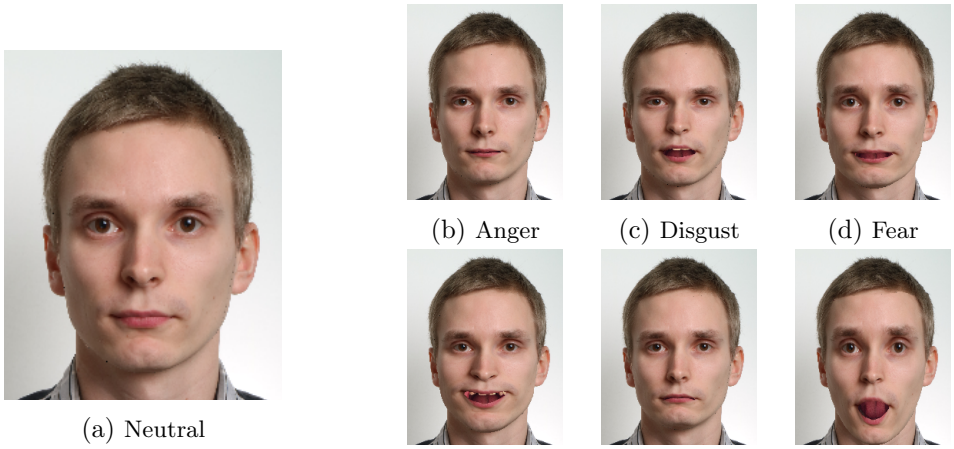
Both the shape and pose fitting procedures are iterable and a shape estimate can be used to refine the camera, which in turn can improve the shape fit by minimizing Eq. 2.4 again. However, since the fitting is based on a sparse set of landmarks in comparison to the dense set of points in the 3D-model, the quality of the fit will degrade the further from the landmarks the specific vertices are.

2.1.4 Facial Expression Fitting

Facial expressions can also be fitted and utilizes a similar methodology as the shape fitting procedure in the previous section. However, now the basis consists of *blendshapes*, per-vertex offsets for facial expressions. In the work [17] these consist of *neutral, anger, disgust, fear, happiness, sadness, surprise*. The *neutral*[1] expression can thought of a sort of expression normalization, as the expression closes the mouth. Renders of the different expressions can be seen in Fig. 2.1. The expression model is defined as follows

$$\mathbf{S}_{expression} = \mathbf{S}_{neutral} + \sum_{i}^{6} \beta_i \mathbf{G_i} \tag{2.6}$$

where $\mathbf{S}_{neutral}$ is the fitted shape, β_i are the weights for a specific blendshape, and $\mathbf{G_i}$ is a blendshape vector corresponding to a specific facial expression. The actual expression fitting occurs in the same way as the shape coefficients in Eq. 2.4. I.e. the vertices that correspond to detected 2D-landmarks are projected down to the image-space and projection error is to be minimized, but now with respect to the expression coefficients $\boldsymbol{\beta}$ instead of the shape coefficients $\boldsymbol{\alpha}$.



(a) Neutral (b) Anger (c) Disgust (d) Fear (e) Happiness (f) Sadness (g) Surprise

Figure 2.1: All blendshapes, as well as the neutral expression.

2.1.5 Texture Fitting

The face pose, shape, and expression have now been determined. Before new faces can be rendered some color information is needed as well. There are different approaches to extracting the color information. In the works by [25], [14], and [2], a color model,

[1] The "neutral" face is simply the estimated face shape without blendshape influence.

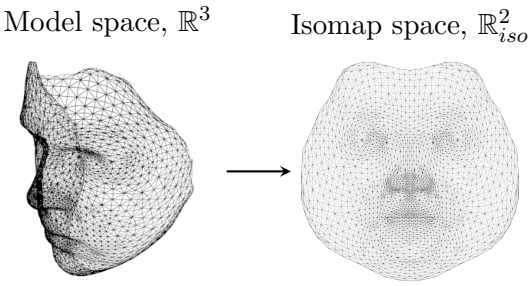


Figure 2.2: The isomap projection.

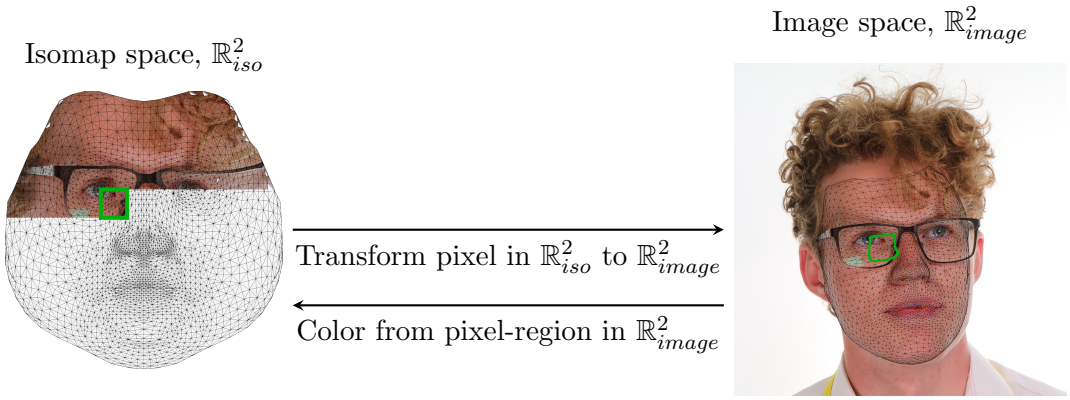


Figure 2.3: Each pixel in the isomap image is traversed and mapped to the projection of the 3D-model in the input image. The region covered by the transformed isomap image pixel is interpolated to determine the pixel color, illustrated by the drawn square in iso-space and its transformed counterpart in image-space. The pixel sizes have been exaggerated in the illustration.

similar to the shape model in Eq. 2.3 is used. However all the visible vertices of the projected mesh can be used for the color fitting estimation. In the PhD-thesis [14] this estimation technique is expanded by using a *Phong reflection model*, which allows for estimation of light- intensity and direction in the input image. An issue with this approach to determine the color-information is that the sampling points are restricted to the vertices, which with lower resolution models means that high frequency information might get lost.

In [17] another approach is suggested, here the *isomap* algorithm, first mentioned in [37] is implemented and used to project the fitted 3D-model to a plane so that the *geodesic* distance, I.e. internal vertex-distances of the model, are preserved. The result of the projection can be seen in Fig. 2.2, and denote the projected space as $\mathbb{R}^2_{iso}$.

Once the isomap has been computed, the task is now to generate an isomap texture of the face. This is done by projecting the 3D-model onto the input image, and then computing a composite mapping from the isomap space, $\mathbb{R}^2_{iso}$ to the image space, $\mathbb{R}^2_{image}$. A texture resolution is now set, which determines how the isomap texture in $\mathbb{R}^2_{iso}$ will be divided into pixels. Now, each pixel in the isomap texture is traversed in $\mathbb{R}^2_{iso}$ and transformed to $\mathbb{R}^2_{image}$. If the transformed pixel is not obscured by other parts of the projected model, the pixel color is determined by sampling the transformed pixel region and applying either bilinear, mean, or nearest neighbor interpolation. The process is repeated until the isomap texture is complete.

2.2 An Overview of Neural Networks

A neural network is a machine learning tool inspired by biological neural networks. In this thesis work, neural networks are used to perform face recognition and face verification. The smallest component of a neural network is the *neuron*. A neuron has an arbitrary number of inputs $a_1, a_2, \ldots, a_n$ and one output a'. Each input has a corresponding *weight* $w_1, w_2, \ldots, w_n$ and each neuron has a corresponding *bias*. A non-linear *activation function* f is often applied to the output so that the output does not depend linearly on the input. The output for a single neuron is calculated according to Eq. 2.7 as

$$a' = f\left(\sum_k w_k a_k + b\right). \tag{2.7}$$

A depiction of a neuron is shown in Fig. 2.4.

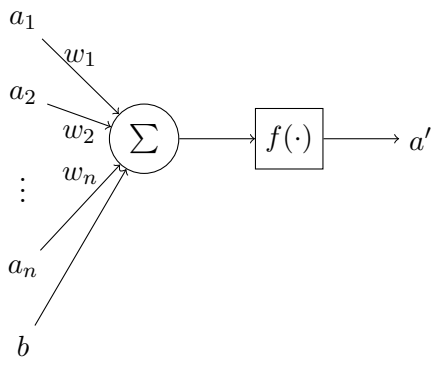


Figure 2.4: An artificial neuron.

A complete neural network can be constructed by grouping together several neurons into *layers* and letting the outputs from neurons in one layer be the inputs to neurons in another layer. An example is shown in Fig. 2.5. In this thesis work, only *feedforward* neural networks are considered. A feedforward network is a special type of neural network where no loops in the network graph are allowed. That is, the output from a layer cannot be connected to the input of a previous layer.

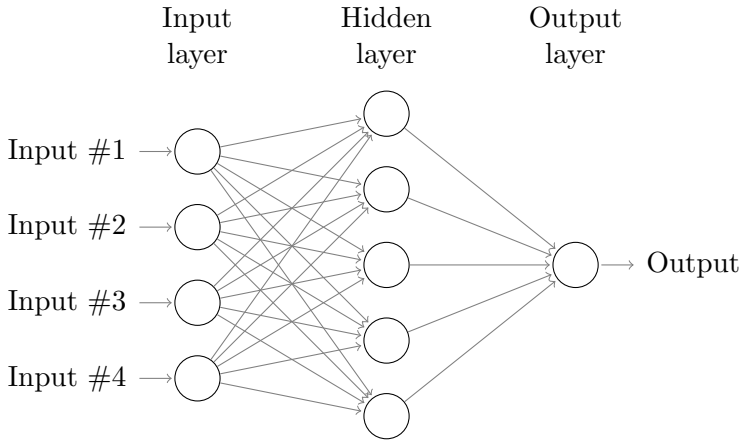


Figure 2.5: A simple neural network with an input layer, a hidden layer and an output layer.

2.2.0.1 Training the Network

For a given input the weights and biases are the parameters that determine the output of the network. In the case of supervised learning a set of labeled training data is used to optimize these parameters so that ideally each input is mapped to the correct output label. This is known as training the network. A trained network can be used to predict the output for some new unseen input data.

In order to train a neural network one must be able to determine the current performance of the network. This is done using a *loss function*. The loss function describes the prediction error the network currently makes with regard to the training data. The goal is then to modify the weights and biases in order to decrease the prediction error.

Training neural networks is achieved using *backpropagation*. Given a loss function $\mathcal{L}$, the partial derivatives $\partial\mathcal{L}/\partial w_i$ and $\partial\mathcal{L}/\partial b_i$ for all weights and biases w_i and b_i are used to optimize the function $\mathcal{L}$ towards a minimum. Ideally this should be the global minimum but in practice a local minimum close to the global minimum is typically good enough.

2.2.0.2 Overfitting

In machine learning in general it is common to divide the available data into a training and testing set. The purpose of the testing set is to not use it while training the network, only once the network is fully trained the testing data is used to evaluate the performance of the network on data it has not encountered before. A properly trained network should be capable of performing well on both the test and the training datasets. A network that performs well on the training set but not on the testing set might be suffering from *overfitting*. Overfitting occurs because the network has started to learn details about the training data that does not generalize to the testing data.

Because the number of parameters in the network can be very large, overfitting easily becomes a major issue. There are many ways to combat overfitting, among them are *dropout*, *weight decay* and *data augmentation*. In this thesis work, data augmentation is the key component that will be investigated.

2.2.1 Convolutional Neural Networks

A convolutional neural network is a special kind of feedforward network. The key difference is the use of convolutional layers instead of only using fully connected layers.

2.2.1.1 Convolutional Layers

The input to a convolutional layer is a two dimensional grid rather than a one dimensional layer of neurons. A kernel is then convolved with the entire grid to produce the output known as an *activation map*. This can be done using different strides of the kernel which affects the output size of the convolutional layer. The parameters of the kernel are the weights for that layer. Several kernels can be used for each layer and ideally these will learn different weights and therefore produce different activation maps. The output of a convolutional layer is therefore a 3 dimensional tensor $(width, height, depth)$ where the width and height represents the size of the output grid and the depth the number of activation maps. An illustration is shown in Fig. 2.6.

When working with images, the weights for the kernels in the early convolutional layers will often cause the kernel to perform basic image processing tasks, such as edge or blob detection. Further down the network the kernels will start to detect more general features representing the training data.

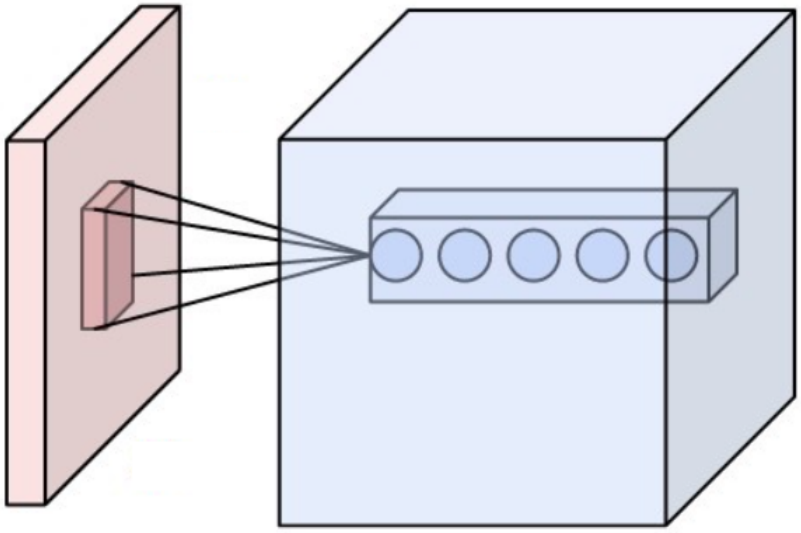


Figure 2.6: A Convolutional layer where a kernel (dark red) is being convolved with an input layer (red). The output layer (blue) has depth 5 in this illustration.

2.2.1.2 Pooling

Pooling layers are used to down-sample the input to the layer. The purpose of this is partially to reduce the number of parameters the model has but also to provide translation invariance. A smaller number of parameters will make the training computationally more efficient and will also reduce the risk of overfitting of the model.

There are several ways to perform the pooling operation. Some of the most commonly used variations are *max pooling*, *average pooling* and *L_2-norm pooling*. The most commonly used pooling operation in the networks used in this thesis work is max pooling. It has been shown to perform better than various other pooling operations by [33]. Max pooling is performed by sliding a grid of size (n, n) over the input with some stride d. The maximum value in the $n \times n$ grid is chosen as its output. In order to down-sample the input by a factor of two a grid size of $(2, 2)$ with a stride of 2 would be used. An illustration is shown in Fig. 2.7.

5	1	**8**	3
2	**6**	4	7
9	0	2	4
5	2	**8**	1

$\longrightarrow$

6	8
9	8

Figure 2.7: A max pooling operation with stride 2 and and grid size $(2, 2)$. The bold numbers in the input to the left represent the maximum value for each grid.

2.2.1.3 Activation Functions

Non-linear activation functions are often applied after the pooling or convolutional layers. There are many choices of activation functions but the one used most commonly in this work is the *rectifier function* which is defined as

$$f(x) = \max(0, x). \tag{2.8}$$

A layer applying the rectifier function is known as a rectified linear unit (ReLU). The benefit of the rectified linear unit is that it is fast to compute while also performing well compared to other popular choices according to [20].

2.2.2 Residual Blocks

A relatively new technique for constructing convolutional networks is the use of *residual blocks* introduced by [13]. The purpose of the residual block is to make it easier to train deep neural networks. A deep network should never have a higher training error than a corresponding shallower network as the additional layers could in the worst case be identity mappings. In practice however, it turns out that this is not always the case as is shown by [13]. Residual blocks are designed to circumvent this problem.

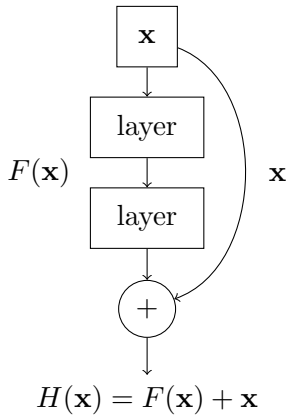


Figure 2.8: A residual block.

A residual block is constructed by adding a shortcut connection from the input to the block to the output of the block as is shown in Fig. 2.8. The mapping that now needs to be optimized is $F(\mathbf{x}) = H(\mathbf{x}) - \mathbf{x}$. According to [13] the residual could be easier to optimize than the original mapping. In the case where the layers should perform identity mappings it could also be easier to optimize the residuals toward zero rather than making the original mapping perform an identity mapping.

2.2.3 Inception Modules

Inception modules were first introduced in [35] and later improved upon in [36]. In its simplest form, the inception module consists of several convolutional and pooling layers in parallel, whose output is concatenated at the end of the module as is shown in figure 2.9. The benefit of these modules is that they are capable of handling features in the input at different scales, compared to a conventional fixed size convolutional layer.

The main problem with this approach is that it becomes very computationally expensive to stack several inception modules on top of each other, in order to create deep networks. This is resolved by dimensionality reduction using 1×1 convolutions as is

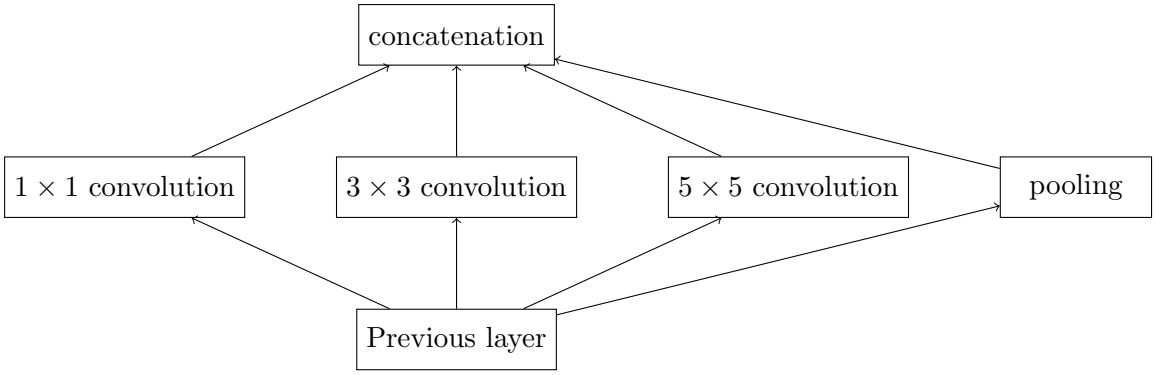


Figure 2.9: A naive implementation of the inception module. No dimensionality reduction is performed.

shown in Fig. 2.10. The 1×1 convolutions shrink the dimension of the filter bank so that fewer 3×3 and 5×5 convolutions need to be performed. This method is based on the idea that a low dimensional feature vector often can contain a lot of information about a higher dimensional vector, such as an image.

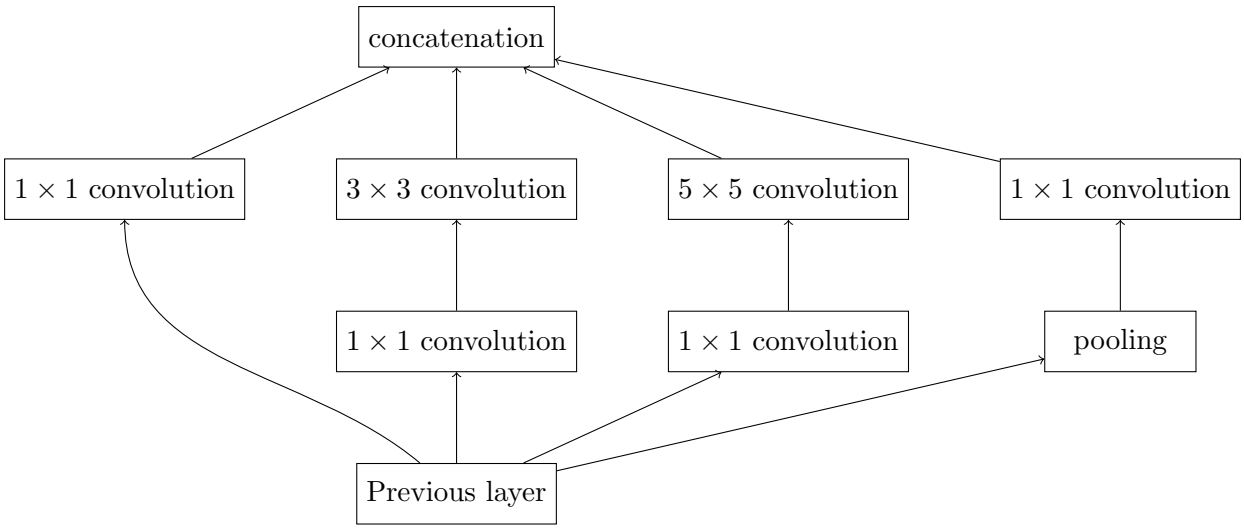


Figure 2.10: An inception module where dimensionality reduction is performed before the expensive 3×3 and 5×5 convolutions.

2.2.4 Loss Functions

To evaluate the performance of the network during training, and to determine how the parameters should be updated, a loss function is needed. The loss function measures the error that the network currently makes.

2.2.4.1 Softmax Cross Entropy Loss

The softmax cross entropy loss is defined as

$$\mathcal{L}_s = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij} \log(p_{ij}) \tag{2.9}$$

where y_{ij} represents the ground truth for sample i and class j and p_{ij} represents the output from the final softmax layer for sample i and class j. The variable n represents the total number of samples and the variable m represents the total number of classes.

2.2.4.2 Center Loss

The center loss function was introduced by [38] who showed that it can reach state of the art performance on the *Labeled Faces in the Wild* (LFW) and the *YouTube Faces Database* (YTF)[39] datasets. The purpose of the center loss function is to group together the deep features that belong to the same class into clusters so that the intra-class variance for the features is small. This is done by training the network as a classifier using the softmax cross entropy loss function and adding the center loss to the total loss. The center loss function is defined as

$$\mathcal{L}_c = \frac{1}{2}\sum_{i=1}^{m} \|\mathbf{x}_i - \mathbf{c}_{y_i}\|_2^2 \tag{2.10}$$

where $\mathbf{x}_i$ denotes the ith feature vector for the y_ith class, $\mathbf{c}_{y_i}$ denotes the y_ith class center. The total loss is defined as

$$\mathcal{L}_t = \mathcal{L}_s + \lambda_c \mathcal{L}_c \tag{2.11}$$

where $\mathcal{L}_s$ represents the softmax loss, the parameter λ_c is a regularization term.

For the center loss to work properly each class center must be updated after every mini-batch. The center for class j during iteration t is updated as

$$\mathbf{c}_j^{t+1} = \mathbf{c}_j^t - \alpha_c \cdot \Delta\mathbf{c}_j^t. \tag{2.12}$$

The centers are updated in the direction $\Delta\mathbf{c}_j$ defined as

$$\Delta\mathbf{c}_j = \frac{\sum_{i=1}^{m} \delta(y_i = j) \cdot (\mathbf{c}_j - \mathbf{x}_i)}{1 + \sum_{i=1}^{m} \delta(y_i = j)} \tag{2.13}$$

where $\delta(condition) = 1$ if the *condition* is satisfied, otherwise $\delta(condition) = 0$. The parameter α_c determines the learning rate of the centers.

Training a network with center loss is similar to training a classifier with only softmax loss. The main difference is that the hyperparameters α_c and λ_c must be chosen. However, experiments conducted by [38] show that α_c and λ_c are not very sensitive to small changes and can be chosen fairly roughly, at least for their architecture and training dataset.

2.2.4.3 Triplet Loss

The idea of triplet loss was introduced by [34] who at the time achieved the record accuracy on the LFW and YTF datasets. The idea behind triplet loss is to minimize the distance between features that belong to the same class and maximizing the distance between features belonging to different classes. This is done by feeding the network three images at a time, an *anchor*, a *positive* and a *negative*. The anchor and the positive should belong to the same class and the negative to a different class. Fig. 2.12 shows an illustration.

Given an image $\mathbf{x}$, the features the network generates are represented by $f(\mathbf{x}) \in \mathbb{R}^d$ where d is the number of features used. The features are constrained by $\|f(\mathbf{x})\|_2 = 1$ so that they lie on the d-dimensional unit sphere. The goal is to make sure that an anchor

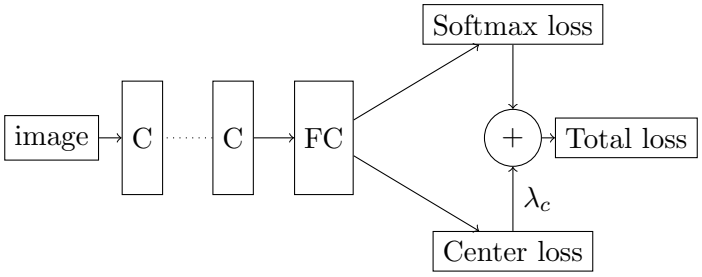
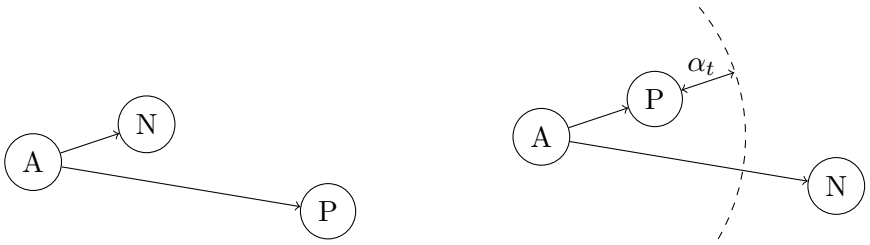


Figure 2.11: A network illustrating the center loss function. The C's stand for convolutional layers and FC for fully connected layers. The final fully connected layer is connected both to the softmax loss layer for linear classification and to the center loss function.



(a) Before training. (b) After training.

Figure 2.12: Illustrating the process of training using triplet loss. Before the network is trained the negative image might be closer to the anchor than the positive image. After training the positive image should be closer to the anchor than the negative image, with some margin α_t.

image $\mathbf{x}_i^a$ for a class is closer to all other images $\mathbf{x}_i^p$ for the same class than it is to any image $\mathbf{x}_i^n$ for any other class. This goal can be expressed as

$$\|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^p)\|_2^2 + \alpha_t < \|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^n)\|_2^2, \forall (f(\mathbf{x}_i^a), f(\mathbf{x}_i^p), f(\mathbf{x}_i^n)) \in \mathcal{T} \tag{2.14}$$

where the set $\mathcal{T}$ represents every possible triplet in the training dataset and α represents the desired margin to have between positive and negative pairs. The loss function to minimize, based on this constraint, is

$$\sum_i \max\left(\|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^p)\|_2^2 - \|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^n)\|_2^2 + \alpha_t, 0\right). \tag{2.15}$$

Training with triplet loss can be quite complicated as selecting triplets in a naive way often results in slow convergence. If a triplet already satisfies the condition in Eq. 2.14 then the loss function will evaluate to zero and it will not contribute to the training. Therefore it is important to choose triplets that violate this condition. Given an anchor image $\mathbf{x}_i^a$ this can be achieved by selecting the positive image as $\text{argmax}_{\mathbf{x}_i^p} \|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^p)\|_2^2$ and the negative image as $\text{argmin}_{\mathbf{x}_i^n} \|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^n)\|_2^2$. For large training sets this will be take a long time to compute and therefore a different strategy is needed.

The authors of [34] propose two different methods of choosing triplets. In this thesis work the "Generate triplets online" method is used. Instead of using the entire training set, a mini-batch of a few thousand images are sampled. The images are sampled in such a way that there are around 40 images for each class. The argmax and argmin

can be computed on this smaller set which should give the network triplets that violate Eq. 2.14 in a reasonable amount of time. Once the sampled triplets are exhausted, new images are randomly sampled.

2.3 Neural Networks for Face Detection

Face detection is required in order to extract crops of faces from a larger image, as is depicted in Fig. 2.13. The cropped face images can then used for tasks such as training face verification networks or extracting deep facial features for face recognition.

Figure 2.13: Face detection used to perform face recognition.

2.3.1 Multi-task Cascaded Convolutional Networks

Face detection using Multi-task Cascaded Convolutional Networks was introduced by [40]. This method consists of three main stages.

In the first stage the input image is resized to several smaller scales, constructing an image pyramid. The image pyramid is then fed as input to a convolutional network called the *Proposal Network* (P-Net). The P-Net finds several candidate bounding boxes in the image. Non-maximum suppression is then used to reject some highly overlapping candidates.

The candidates from the first stage are then fed into the second stage. The second stage consists of another convolutional network called the *Refine Network* (R-Net). This network further rejects some candidate bounding boxes and performs non-maximum suppression.

The third stage is similar to the second stage. The output from the second stage is fed in to another convolutional network called the *Output Network* (O-Net). This network attempts to find the final bounding boxes that contain a face. It also outputs five facial landmarks. One for each corner of the mouth, one for each eye and one for the tip of the nose.

2.4 Neural Networks for Image Generation

2.4.1 Generative Adversarial Networks

A GAN [8] consists of two separate networks, the *generator* G and the *discriminator* D. Given a training dataset, the idea is to let the generator G approximate the latent distribution p_{data}, representing the training data. This can be achieved by letting the generator represent a mapping $G : \mathbb{R}^{N_z} \to \mathbb{R}^{N_x}$ where typically $N_z < N_x$. Here N_z represents the size of the input vector to the generator while N_x represents the total number of pixels in the generated image. In the case of image generation, an input vector $\mathbf{z} \in \mathbb{R}^{N_z}$, sampled from a uniform distribution, would be mapped to an image $G(\mathbf{z}) \in \mathbb{R}^{N_x}$. The discriminator on the other hand is responsible for detecting images

$G(\mathbf{z})$ that were unlikely to have been sampled from p_{data}. The function $D(G(\mathbf{z}))$ represents the probability that $G(\mathbf{z})$ was generated from the distribution p_{data} rather than the distribution approximated by $G(\mathbf{z})$. To achieve these goals the generator and discriminator networks are often trained simultaneously in a two-player minimax game. The generator tries to approximate the latent distribution so accurately that the discriminator cannot tell whether the image $G(\mathbf{z})$ is real or generated. The discriminator on the other hand tries to become increasingly better at detecting the differences between real and generated images. In the end the generator should ideally be able to generate images that are visually indistinguishable from images sampled from p_{data} while the discriminator should be randomly guessing whether the generated images are real or not.

2.4.2 Boundary Equilibrium Generative Adversarial Networks

The *Boundary Equilibrium Generative Adversarial Network* (BEGAN) [4] is a recent improvement on the generative adversarial networks. The key improvements include the equilibrium concept where the idea is to balance the discriminator and generator against each other, so that neither wins too easily. A new way of controlling the trade-off between image quality and image diversity is also proposed. This method uses an auto-encoder based generative adversarial network. The generator network is represented by the decoder part of an auto-encoder network while the discriminator is represented by a complete auto-encoder network.

2.4.2.1 Loss Functions

One important difference between a traditional GAN and a BEGAN is that while the former attempts to match the data distributions the latter tries to match the reconstruction loss distributions for the auto-encoder instead. The loss function $\mathcal{L} : \mathbb{R}^{N_x} \to \mathbb{R}^+$ for the auto-encoder is given as

$$\mathcal{L}(v) = |v - D(v)| \tag{2.16}$$

where $D : \mathbb{R}^{N_x} \to \mathbb{R}^{N_x}$ represents the auto-encoder function and $v \in \mathbb{R}^{N_x}$ represents an image. The losses for the generator $\mathcal{L}_G$ and discriminator $\mathcal{L}_D$ are defined as

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(\mathbf{x}) - \mathcal{L}(G(\mathbf{z})) \\ \mathcal{L}_G = \mathcal{L}(G(\mathbf{z})) \end{cases} \tag{2.17}$$

where $\mathcal{L}(\mathbf{x})$ represents the auto-encoder loss for a real image $\mathbf{x}$ and $\mathcal{L}(G(\mathbf{z}))$ represents the auto-encoder loss for a generated image $G(\mathbf{z})$. The reasoning behind this is that the discriminator should be good at reconstructing real images while it at the same time should attempt to maximize the reconstruction error for the generated images. The generator on the other hand tries to minimize the reconstruction error for the generated images. Since the generator cannot affect the auto-encoder, the only way it can minimize its loss is by generating images that look more realistic.

The theoretical background to the choice of loss functions is based on the 1-Wasserstein distance function between two probability distributions μ_1 and μ_2, defined as

$$W_1(\mu_1, \mu_2) = \inf_{\gamma \in \Gamma(\mu_1, \mu_2)} \mathbb{E}_{(x_1, x_2) \sim \gamma} \left[|x_1 - x_2| \right]. \tag{2.18}$$

Here, $\Gamma(\mu_1, \mu_2)$ represents the set of all couplings of μ_1 and μ_2, where a coupling represents a joint probability distribution γ who's marginal distributions correspond to μ_1

and μ_2. Using Jensen's inequality, the authors of [4] show that $W_1(\mu_1, \mu_2)$ is lower bound by $|m_1 - m_2|$ where m_1 and m_2 are the means of the distributions μ_1 and μ_2. The idea is then to let the distribution μ_1 represent the distribution of the losses $\mathcal{L}(\mathbf{x})$ and let the distribution μ_2 represent the distribution of the losses $\mathcal{L}(G(\mathbf{z}))$. By maximizing the 1-Wasserstein distance between these distributions, the discriminator should be capable of separating real and generated images. Maximizing $W_1(\mu_1, \mu_2)$ can be achieved by letting $m_1 \to 0$ and $m_2 \to \infty$ as $W_1(\mu_1, \mu_2) \geq m_2 - m_1$. This naturally leads to the definition of the discriminator loss $\mathcal{L}_D$ which is minimized by minimizing the loss $\mathcal{L}(\mathbf{x})$ corresponding to m_1 and maximizing the loss $\mathcal{L}(G(\mathbf{z}))$ corresponding to m_2. The generator loss $\mathcal{L}_G$ tries to achieve the opposite by minimizing $\mathcal{L}(G(\mathbf{z}))$ instead.

2.4.2.2 Boundary Equilibrium

The concept of *equilibrium* is defined as

$$\mathbb{E}[\mathcal{L}(\mathbf{x})] = \mathbb{E}[\mathcal{L}(G(\mathbf{z}))] \tag{2.19}$$

as an attempt to balance the discriminator and generator against each other. By introducing a parameter $\gamma_e \in [0, 1]$ the equilibrium constraint can be modified as

$$\gamma_e = \frac{\mathbb{E}[\mathcal{L}(G(\mathbf{z}))]}{\mathbb{E}[\mathcal{L}(\mathbf{x})]} \tag{2.20}$$

which makes it possible to affect the balancing ratio. To attempt to enforce this ratio γ_e during training we note that we would like to achieve $\gamma_e \mathcal{L}(\mathbf{x}) - \mathcal{L}(G(\mathbf{z})) = 0$ which the authors of [4] have solved using proportional control theory. The discriminator loss is modified to become $\mathcal{L}_D = \mathcal{L}(\mathbf{x}) - k_t \cdot \mathcal{L}(G(\mathbf{z}))$ where k_t controls how much emphasis is put on $\mathcal{L}(G(\mathbf{z}))$. The variable k_t is updated every training iteration according to $k_{t+1} = k_t + \lambda_k(\gamma_e \mathcal{L}(\mathbf{x}) - \mathcal{L}(G(\mathbf{z})))$ where λ_k acts as the proportional gain. Putting all the pieces together the final BEGAN objective is

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(\mathbf{x}) - k_t \cdot \mathcal{L}(G(\mathbf{z})) \\ \mathcal{L}_G = \mathcal{L}(G(\mathbf{z})) \\ k_{t+1} = k_t + \lambda_k(\gamma_e \mathcal{L}(\mathbf{x}) - \mathcal{L}(G(\mathbf{z}))) \end{cases} . \tag{2.21}$$

Chapter 3

Methodology

In this section the software tools used for this thesis work are outlined followed by a short introduction to the datasets used for training and evaluation. This is followed by a presentation of methods used for face verification, face recognition and finally the different augmentation schemes. As face recognition, face verification networks and GANs are trained and evaluated, Tab. 3.1 gives an overview of which dataset (section 3.2) is used for what.

Training	Evaluation	Type
Axis employee database	Axis internal dataset	Face recognition
FERET dataset (*Original* subset)	FERET dataset (first half)	Face recognition
FaceScrub	Labeled faces in the wild	Face verification
Datasets generated using BEGAN	Labeled faces in the wild	Face verification
CelebA	N/A	Face generation

Table 3.1: Each row states which dataset is used for training, which dataset is used for evaluation and what type of task is being solved. For face generation there is no evaluation dataset as visual inspection is used to determine if the training was successful or not. For a description of the datasets, see section 3.2.

3.1 Tools Used

3.1.1 TensorFlow

TensorFlow [1] is a software library designed for machine learning. In this work it is used mainly to define and train convolutional neural networks. It is capable of running on GPUs and uses the NVIDIA CUDA Deep Neural Network library (cuDNN) which delivers GPU accelerated implementations of common neural network routines.

The TensorFlow-Slim library is used as it provides definitions of several common convolutional network architectures. It also provides pre-trained parameters for these networks. The parameters are a result of training the networks on the ImageNet Large Scale Visual Recognition Dataset from 2012.

3.1.2 Eos

The Eos model-fitting library is developed at University of Surrey, presented in the paper [17], and is used to perform the fitting of the SFM to a specified set of ibug facial landmark points, developed in [29], [30] and [31]. Pose, shape, expression and facial

texture can be estimated using this framework. Meshes and textures can be exported as *wavefront objects*, .obj , and images respectively.

3.1.3 Dlib

Dlib is a machine learning library for C++ developed in [19]. Although the library offers a wide range of functionality, its facial landmark detector, based on the paper [18], is the feature mainly used in this thesis. An example of fitted landmarks can be seen in Fig. 3.1.

Figure 3.1: Face with detected ibug landmarks

3.1.4 OpenGL

OpenGL is a graphics library that allows for hardware accelerated computations, mainly concerning graphical applications. The library consists of two parts:

- A graphics language, GLSL (*Graphics Library Shading Language*), that allows the user to write programs, *shaders*, to be run on the computer GPU.
- An API for setting up models, assigning the correct shaders and textures, and rendering the images to buffers on the GPU.

In this thesis, a Python version of the library, *pyopenGL*, as well as GLUT are used to render the fitted 3D-models into new poses. GLUT stands for *openGL Utility Toolkit* and is a small extension to the ordinary OpenGL, handling windowing functions, and user events. This allows the user to actually see what is being rendered with OpenGL.

3.1.5 OpenCV

OpenCV is an open source library for various image processing tasks, such as image alignment, camera calibration, and pose estimation. In this thesis work version 2.4 is used.

3.2 Datasets

3.2.1 FaceScrub

The FaceScrub dataset originally contains 106, 863 face images belonging to 530 different identities, developed in [23]. The dataset is provided in the form of URLs to the images. Unfortunately some of the URLs are no longer valid and therefore the dataset used in

this work contains 77,852 images and 530 identities. Some of the identities in the dataset also belong to the Labeled Faces in the Wild dataset, which is used for evaluation. These are removed so that they do not affect the final results. Therefore the size of the dataset is further reduced to 54,139 images and 384 identities.

3.2.2 Labeled Faces in the Wild

The LFW dataset is a popular dataset for evaluating the performance of face verification algorithms. It originates from the work by [15]. It contains 13,233 face images belonging to 5,749 different identities.

3.2.3 FERET

The FERET dataset consists of 14,051 images of 1,199 identities, collected by the George Mason University under the FERET program, sponsored by the DOD Counter drug Technology Development Program Office, for the work in [26], [27] and [28]. The dataset differs from the previously mentioned sets in that each identity is imaged under large pose variations, ranging from left profile, to right profile.

3.2.4 Large Scale CelebFace Attributes

The *Large Scale CelebFace Attributes* (CelebA) is a dataset consisting of 202,599 images of celebrities over 10,177 identities, collected by the Chinese University of Hong Kong for the work in [21]. This dataset is primarily used in this thesis to train the BEGAN.

3.2.5 Axis Employee Database

A set of 298 high resolution images of Axis employees were obtained at the beginning of the thesis work. Prior to the thesis work, a questionnaire was sent out to all axis employees asking whether or not they would be willing to be participate in a new dataset for use in research. The identities in this dataset are the employees who gave their consent in the questionnaire. There is one portrait image per identity in the database. This database of images is used for 3D reconstruction, as a training set for face recognition and as an aid for manual data collection.

3.2.6 Axis Internal Dataset

A dataset containing Axis employees from selected surveillance cameras at Axis. Using the Axis employee database as an identity gallery, this dataset was collected from surveillance footage in a semi-automatic fashion. During the capture phase, surveillance snapshots were sent to a server hosting a deep neural network, where the network attempted to classify each image to one of identities in the Axis employee database. However, due to issues in the classification, manual labeling was employed as well in order to correctly classify the images. The dataset was also extended by labeling datasets from previous axis projects as well. As there were only a few different cameras from where the footage was obtained, images from all the 298 identities could not be collected. The Axis internal dataset consists of 1,793 images belonging to 31 identities.

3.3 Training Face Verification Networks

Training large face verification networks from scratch that can perform at the current state of the art takes a long time and can require up to millions of correctly labeled train-

ing images. In order to save time we instead fine-tuned existing pre-trained networks where smaller datasets (around tens of thousands of images) can be used to achieve reasonable performance. We are interested in answering the following questions:

- Can center loss and triplet loss, mentioned in section 2.2.4.2 and 2.2.4.3, be used effectively to generate discriminating feature vectors when fine-tuning pre-trained networks, using a relatively small training dataset (around $50,000$ images)?
- Can real images in the training dataset be replaced with synthetic images?
- Can completely synthetic datasets generated by GANs be used to fine-tune the pre-trained networks?

3.3.1 Network Architecture

We use the GoogLeNet [35] (also known as the InceptionV1) architecture when performing the fine-tuning. An overview of the network architecture is shown in Tab. 3.2. The reason behind this choice is that the network achieves a fairly high performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) from 2012 while still being a significantly smaller network than the highest performing networks, such as the Inception-ResNet-V2 or ResNet-152 networks. The pre-trained weights for the GoogLeNet network, trained for the ILSVRC, are provided by TensorFlow-Slim which is another reason for choosing this particular network architecture.

When fine-tuning the network the final fully connected layer is changed to a layer with an output size of 128 in order to create feature vectors in $\mathbb{R}^{128}$. For the last two inception modules, called inception 5a and inception 5b, the pre-trained weights are not loaded and are instead trained from scratch.

3.3.2 Training Procedure

The FaceScrub dataset is used for fine-tuning the network. There are several overlapping identities between the FaceScrub dataset and the LFW dataset. These are removed from the FaceScrub dataset as LFW will be used for evaluation. After this, there are $54,139$ images belonging to 384 different identities left in the dataset. These are used to fine-tune the pre-trained GoogLeNet network both using center loss and triplet loss.

The training dataset is then shrunk down by keeping all images larger than 600×600 pixels and discarding the rest. This leads to a dataset containing $10,309$ images belonging to 382 different identities. This is done so that the SFM can be used to create 3D reconstructions from these images, as it requires a fairly high quality image for the reconstruction to work properly. The network is then fine-tuned on these $10,309$ images. Three different augmented datasets are then constructed so that each contains roughly $50,000$ images. These datasets are constructed using the SFM, the MFM and conventional 2D augmentation techniques respectively.

3.3.2.1 Training Procedure Using Center Loss

When fine-tuning using center loss a batch size of 50 images is used. Stochastic gradient decent is used during backpropagation with a learning rate of 0.01. The parameters specific to center loss were chosen to be $\alpha_c = 0.1$ and $\lambda_c = 10^{-4}$. The networks were fine-tuned for approximately 30 minutes to one hour, depending on the training dataset, and stopped when the validation accuracy on the LFW dataset had converged.

Layer type	Kernel size / stride	Output size
Convolution	7×7 / 2	$112 \times 112 \times 64$
Max pool	3×3 / 2	$56 \times 56 \times 64$
Convolution	3×3 / 1	$56 \times 56 \times 192$
Max pool	3×3 / 2	$28 \times 28 \times 192$
Inception 3a		$28 \times 28 \times 256$
Inception 3b		$28 \times 28 \times 480$
Max pool	3×3 / 2	$14 \times 14 \times 480$
Inception 4a		$14 \times 14 \times 512$
Inception 4b		$14 \times 14 \times 512$
Inception 4c		$14 \times 14 \times 512$
Inception 4d		$14 \times 14 \times 528$
Inception 4e		$14 \times 14 \times 832$
Max pool	3×3 / 2	$7 \times 7 \times 832$
Inception 5a		$7 \times 7 \times 832$
Inception 5b		$7 \times 7 \times 1024$
Average pool	7×7 / 1	$1 \times 1 \times 1024$
Fully connected		$1 \times 1 \times 128$
Softmax		$1 \times 1 \times N_c$

Table 3.2: The layers in the GoogLeNet architecture. All inception modules have an architecture similar to Fig. 2.10. The softmax layer is only used when training using center loss and is not necessary when training using triplet loss. The variable N_c represents the number of classes in the training dataset. The input to the network is an image of size $224 \times 224 \times 3$.

3.3.2.2 Training Procedure Using Triplet Loss

When fine-tuning using triplet loss 30 identities with 30 images per identity are sampled in order to generate triplets that violate the triplet constraint in Eq. 2.14. Stochastic gradient decent is used during backpropagation with a learning rate of 10^{-3}. The margin α_t between the positive and negative pairs was chosen as $\alpha_t = 0.2$. A batch size of 90 was used. The networks were fine-tuned for approximately two hours and stopped when the validation accuracy on the LFW dataset seemed to have converged.

3.3.3 Evaluation

To determine if two images belong to the same identity, the distance between the features vectors from the two images is calculated. If the distance between the feature vectors is below some threshold then the two images are considered to belong to the same identity, otherwise they are considered to belong to different identities. This allows us to calculate the accuracy, false positive rate and true positive rate for each threshold value. False and true positive rates are defined as

$$\text{True positive rate} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \tag{3.1}$$

$$\text{False positive rate} = \frac{\text{False positives}}{\text{False positives} + \text{True negatives}}. \tag{3.2}$$

For different thresholds the accuracy will vary. Therefore the accuracy is reported as the best performing accuracy for all thresholds.

The fine-tuned networks are evaluated on the LFW dataset, where our method falls into the "Unrestricted, labeled outside data" category, as we use labeled data which is not the LFW training dataset. The creators of the LFW dataset provide 10 randomly generated splits of the dataset so that 10-fold cross validation can be used. The final accuracy of the trained network is reported as the mean accuracy for the 10 splits, along with the standard deviation. The true positive rate and the false positive rates are also calculated as the mean of the rates for each split. These rates are plotted as *Receiver Operator Characteristic* (ROC) curves. The *area under the curves* (AUCs) is used as a measure of performance, a higher area under the curve means a higher performance.

3.4 Face Recognition

Face recognition is performed using a pre-trained face verification network as our own fine-tuned networks are not expected to achieve state of the art accuracy. The testing datasets contain labeled images that need to be correctly identified. That is, given an image we need to answer the question, *"What is the identity of the face in the image?"*. The task of face verification, given two images, is on the other hand to answer the question *"Do these two faces belong to the same identity?"*. In order to solve this problem using face verification networks, we have a training database of labeled images that the test images can be compared against. For an input face image, the face verification network returns a feature vector representing the face. Feature vectors that are close to each other are more likely to belong to the same identity than feature vectors that are far apart. Given an image from the testing dataset, we make the guess that the identity of the closest feature vector in the training database is the identity of the test image. In other words, we use nearest neighbour classification. This is illustrated in Fig. 3.2.

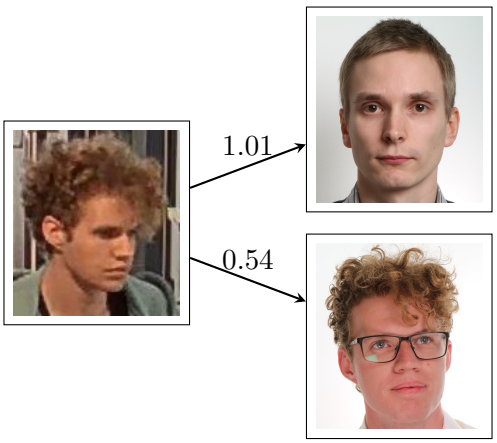


Figure 3.2: The database of images is shown to the right and contains two identities. A testing image is compared to all images in the database. The image with the lowest distance to the testing image is assumed to be the correct identity. In this case the correct choice was made.

Fig. 3.3 shows a principal component analysis of feature vectors from real images, extracted from the face verification network. As can be seen, images captured at different angles produce feature vectors that cluster together into their own groups. As all training images we have are frontal facing images, we might not be able to properly identify images taken at an angle. Ideally we would like to have training samples that have feature vectors close to images taken at an angle as well. However, as we do not have

such images, we aim to create them synthetically using the SFM and the MFM.

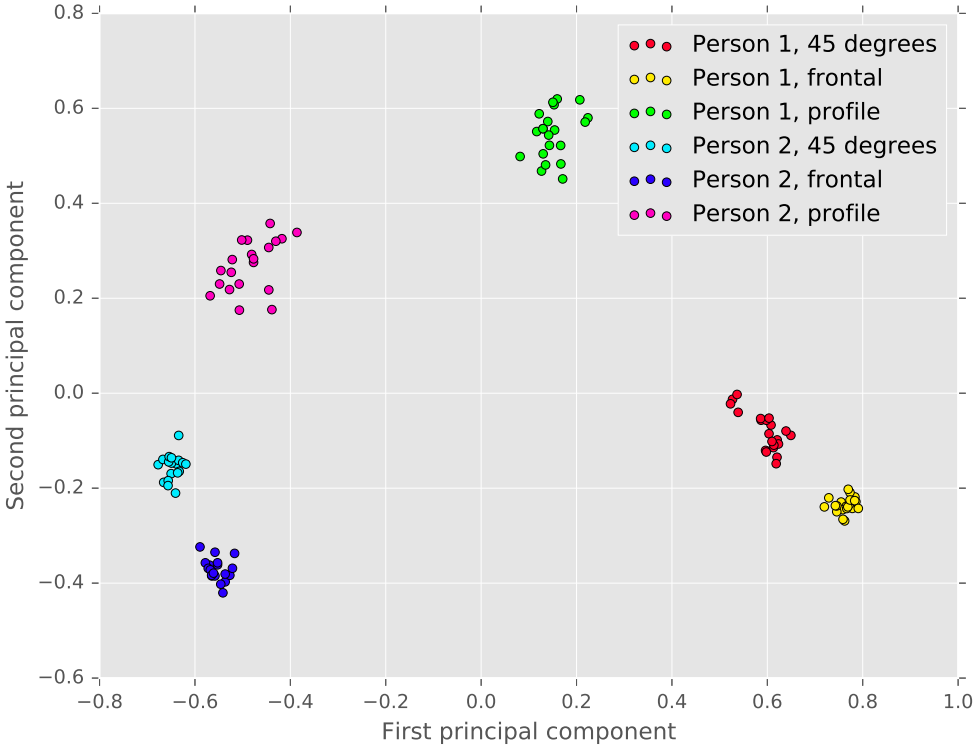


Figure 3.3: Principal component analysis of feature vectors from real images. All feature vectors belong to two identities. For each identity there are frontal images, images taken at approximately a 45° angle and profile images. Images taken at different angles generate different clusters.

3.4.1 Network Architecture

The pre-trained face verification network used for face recognition is an Inception-ResNet-V1 network. This network has been trained on a subset of the MS-Celeb-1M [10] dataset using center loss for input images of size 160×160 pixels [1].

3.4.2 Evaluation

3.4.2.1 Precision, Recall and Average Precision

To evaluate the performance of the different augmentation strategies, we calculate the *precision-recall* curve for the classifier for each augmentation strategy. To be able to calculate precision and recall, we turn the multiclass classification problem into a binary classification problem by introducing a decision threshold. The threshold is used to determine if the closest match found by the face verification network should be considered to correspond to the correct identity or not. This gives us the possibility to find the number of true positives, false positives, true negatives and false negatives, which are calculated according to the following criteria:

[1] The network weights were obtained at `https://github.com/davidsandberg/facenet`

- **True positive:** The distance to the closest match is below the threshold and the correct identity is found.
- **False positive:** The distance to the closest match is below the threshold and an incorrect identity is found.
- **True negative:** The distance to the closest match is above the threshold and an incorrect identity is found.
- **False negative:** The distance to the closest match is above the threshold and the correct identity is found.

The precision and recall are then calculated as

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \tag{3.3}$$

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}. \tag{3.4}$$

To generate precision-recall curves, the precision and recall are calculated at various thresholds. These thresholds are chosen by finding the distance to the closest training image for every testing image, as it is at these distance thresholds that the precision and recall could possibly change. Once all precision and recall values are obtained, the precision can be plotted as a function of the recall.

In this thesis work we use a method called 11-point interpolated precision-recall [7] to present our precision-recall curves. The idea is to remove the jagged structure that a precision-recall curve typically has. This is done by calculating the interpolated precision $p_{interp}(r)$ at 11 fixed recall values $r \in \{0, 0.1, \ldots, 1.0\}$ as

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{3.5}$$

where $p(\tilde{r})$ is the precision at recall $\tilde{r}$. The *Average Precision* (AP) score is then calculated as

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} p_{interp}(r) \tag{3.6}$$

where a higher AP score is better.

3.4.2.2 Cumulative Match Characteristics and Rank-N Accuracy

When performing face recognition on a test image, the database of training images is ranked based on the distance from the test image to the training images. When calculating precision and recall, only the closest image is considered and therefore information regarding the remaining ranked images is lost. When calculating rank-N accuracy, the N closest images are considered. If a training image, who's label matches the test image's label, is included among the N closest training images then this is considered a correct prediction. The *Cumulative Match Characteristic* (CMC) [9] is defined as the accuracy as a function of the rank.

3.4.2.3 Receiver Operator Characteristics

ROC curves are also calculated by calculating the distance for every image in the test dataset to every image in the training dataset to see how well the face verification, which is a part of face recognition, performs. The true positive rates and the false positive rates are calculated as mentioned in section 3.3.3. The AUC is also presented.

3.5 3D-Augmentation

Several types of augmentation were carried out in this thesis work. Firstly, the facial landmarks were extracted in input image, using the Dlib implementation. When the landmarks have been found, or the image has been rejected, the synthesizing of new images ensues. The following synthesis-schemes are evaluated:

- *Pre-Estimated pose and shape augmentation*, using the MFM as suggested in [22].
- *3D-pose augmentation*, using the SFM from the work in [17] and rendered using OpenGL.

Using the MFM approach, the PnP-problem, see [11] is solved using a common algorithm supplied with OpenCV and used to estimate the rotation and translation of a pre-determined pose[2] and shape face-model in $\mathbb{R}^3$ in comparison to the set of input facial landmarks in $\mathbb{R}^2$. Once the pose of the model in relation to the image has been estimated, the model is projected down to $\mathbb{R}^2$ and the image is remapped to the projected coordinates. However, if the image poses are too extreme, I.e. occluding large parts of the face, some parts of the augmentation are skipped, as there simply is not enough data for proper image synthesis. Since the models are supplied with pre-calculated intrinsic camera-matrices and applied poses, this augmentation scheme is very fast, and can be applied to large datasets rapidly.

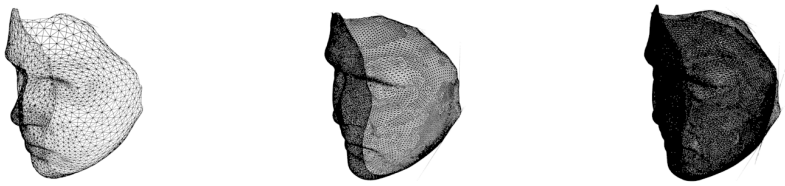
3.5.1 Pose- and Shape Synthesis

The SFM used in this thesis is supplied at multiple 3D-resolutions:

- 3,448 vertices
- 16,759 vertices
- 29,587 vertices

Samples of each resolution can be seen in Fig. 3.4.

[2]The model was provided at specific yaw angles, (0°,±40°,±75°), where the positive orientations are found by mirroring the image.



(a) 3,448 vertices (b) 16,759 vertices (c) 29,587 vertices

Figure 3.4: The wireframe mesh of the SFM rendered at different resolutions

Each model is supplied with corresponding sets of blendshapes and landmark annotations. Of these models, the 3,448 version is used most prominently throughout this thesis. While the larger models might be able to match some parts of the face more accurately, the entire fitting procedure is still only based on the 68 landmark-points, which with the higher resolution models can cause unwanted results and even collapse of the model. This can be aided by stricter regularization parameters, but in turn requires more fit-iterations, resulting in unfeasible computation times.

Using the 3,448 vertex version, the SFM is fitted to the set of landmarks, producing an estimate of the camera pose, 3D-shape and facial expression and facial expression of the depicted face. The process is iteratively applied in order to further refine the shape fit and minimize Eq. 2.4. A complete facial texture is also created using the iso-mapping algorithm. An example can be seen in Fig. 3.5.

(a) Fitted model (b) Corresponding isomap texture

Figure 3.5: Output of the SFM-augmentation procedure

At this point new views can be rendered using OpenGL, which are either pose augmentations by supplying different rotational matrices or expression augmentations by varying the blendshape coefficients. “Expression normalization” can also be applied by simply setting all blendshape parameters to zero. In order to provide more variation in the images during the synthesizing stage, a random background image can be selected each time an augmentation is performed. An example of the procedure can be seen in Fig. 3.6

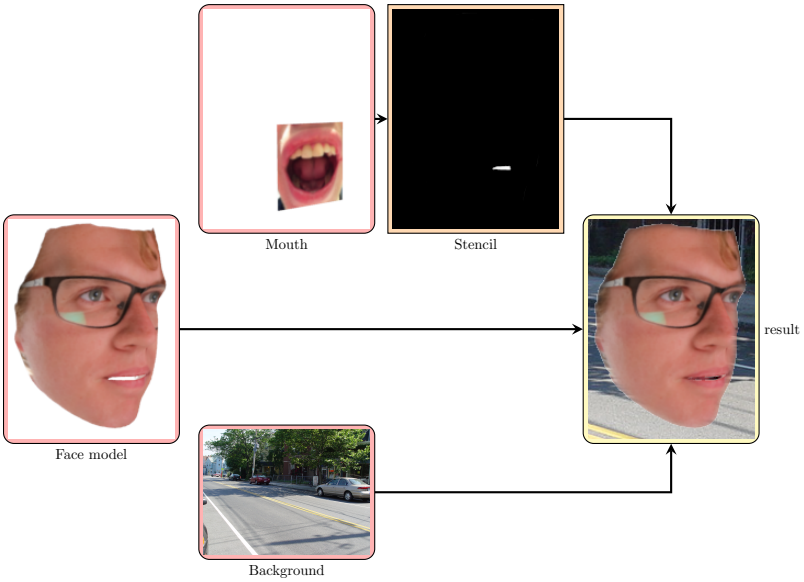


Figure 3.7: Rendering scheme in OpenGL

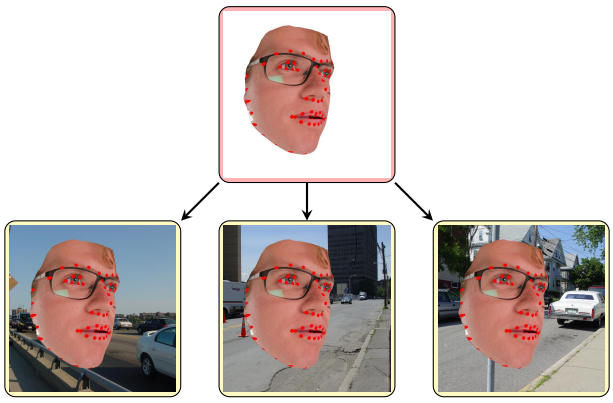


Figure 3.6: Render with random backgrounds applied.

When using large blendshape weights, the background image tended to be shown through the mouth, producing unnatural artifacts that might decrease recognition performance. Especially when rendering back on the input image, some augmentations might get an extra set of mouths. Two counter measures were conceived to counter these artifacts. Firstly, the inside[3] of the face model was colored in a dark red color. Using this technique, the background would get occluded by the inside of the mouth at large yaw angles. Secondly a texture depicting the inside of a mouth is drawn on to a quad behind the face, and a stencil buffer is utilized to mask out parts of the mouth texture either covered by the face or outside the face. The complete rendering scheme is visualized in Fig. 3.7: New poses of the 3D-face can now be synthesized by applying rotational matrices to the model prior to rendering. In this thesis work we vary *yaw* and *pitch*. The rotations are visualized in Fig. 3.8. Note that *roll* is not utilized as it is considered very similar to the common 2D image synthesis technique of in plane rotations of the image.

[3]The "inside" of the model can be thought of as visible parts of the models that are back-facing in relation to the camera, I.e. $\mathbf{n} \cdot \mathbf{c} < \mathbf{0}$ where $\mathbf{n}$ is the triangle normal and $\mathbf{c}$ is the principal axis of the camera.

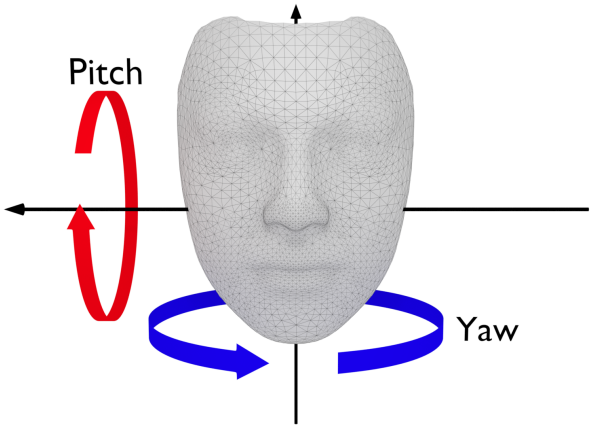


Figure 3.8: Reference image for yaw and pitch rotation.

3.5.2 Multitexture Rendering

Even if each fitted model has a corresponding texture for the entire face, there are some parts of the face that are occluded, which means parts of the facial information is missing. In order to circumvent this issue a rendering scheme using multiple face-textures was employed. From each input image, an isomap texture as well as the cameras principal axis is saved as seen in Fig. 3.9.

Figure 3.9: Synthesized model textures, and their corresponding camera's principal axis, seen from above.

At render time, all of the textures are loaded with the 3D-model and interpolated using Gaussian interpolators over the model. The interpolation weight of each texture is based upon the dot product between the principal axis of the current camera used, $\mathbf{c}$ and the principal axis of the camera used to extract a specific facial texture, $\mathbf{c_i}$. An example can be seen in Fig. 3.10. Both texture and orientation vectors are loaded into OpenGL using *GLtextureArrays*, which packs the data in a layered structure, which allows for fast and easy access from within GLSL.

(a) Left (b) Front (c) Right

Figure 3.10: Pose synthesis utilizing multi-texture interpolation. The three textures used the same as in Fig. 3.9.

3.5.3 Pose Evaluation

3.5.3.1 FERET Dataset

In order to evaluate possible performance enhancements with pose augmentations, the first half of the FERET dataset is selected as the evaluation dataset. This gives us 7,085 images spread over 725 identities. The data is then sorted into subsets, depending on pose, as seen in Tab. 3.3. A frontal image from each identity is selected for the 3D-augmentation and sorted into a separate subset, *Original*. This subset will also be used as a baseline when testing against the other subsets.

Name	Orientation	Angle(radians)
f	front	0.0
hl	half-left	~ -1.18
hr	half-right	~ 1.18
ql	quarter-left	-0.4
qr	quarter-right	0.4
pl	profile-left	$-\pi/2$
pr	profile-right	$\pi/2$
r	randomly-posed	-
Original	Frontal, for augmentation	0.0

Table 3.3: FERET subsets

The augmentation procedure is then applied, where 10 iterations of the fitting algorithm was applied. Expression-neutralized images are synthesized at yaw angles corresponding to the poses in the dataset. The synthesized images are then verified against each of the testing subsets, where the hypothesis is that the synthesized images at similar yaw-angles as the test subsets will produce higher verification scores than the original frontal subset. In order to compare the two main methods in this thesis, the MFM was employed on the frontal *Original* subset as well. Examples of the test images as well as synthesized images using the two methods can be seen in Fig. 3.11.

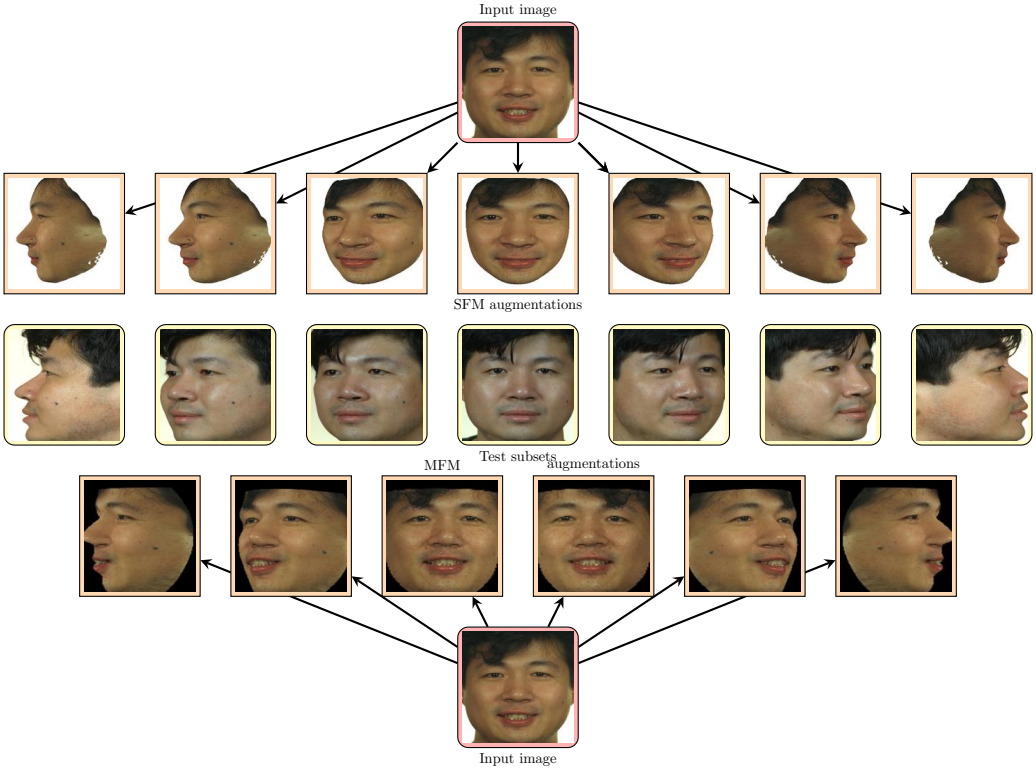


Figure 3.11: Augmentation scheme for the FERET dataset. Using the SFM and MFM, a single image is used to synthesize new poses. Each of the synthesized images are the tested against the images in the test sets. The hypothesis being that synthesized- and test images at similar poses perform better than the initial image.

3.5.3.2 Axis Internal Dataset

Evaluating the face recognition performance of pose augmentations using the Axis internal dataset is done by using the Axis employee database as the training dataset and using the entire Axis internal dataset as the evaluation dataset. The training dataset is augmented using the SFM and the MFM. Conventional 2D augmentation, in the form of in-plane rotations, is also used to compare it to the 3D augmentation techniques.

The 3D reconstructions created using the SFM are rendered at yaw angles in the range $[-0.8, 0.8]$ radians with an increment of 0.1 radians and at pitch angles in the range $[0, -0.4]$ radians with and increment of 0.1 radians. The choice of yaw and pitch angles are based on the yaw and pitch angles appearing in the Axis internal dataset, as we are interested in finding out if synthetic images, with a pose similar to the testing images, can achieve better face recognition performance than real frontal images.

The MFM is used to create synthetic images at $\pm 40°$ which is roughly equal to ± 0.7 radians. The MFM can also generate images at $\pm 75°$ which is roughly equal to ± 1.3 radians. This is however outside of the range we are interested in for the Axis internal dataset and these are therefore discarded.

The synthetic images generated for each pitch and yaw angle pair using the SFM are used as training data when evaluating on the Axis internal dataset, in order to find the optimal pitch and yaw angles. The synthetic images are also used to augment the Axis employee database symmetrically. This means that for each pitch and yaw angle pair, both the positive and negative yaw angle is used at the same time.

3.6 Augmentation Using BEGAN

The BEGAN is used to attempt to create training data both for face verification and for face recognition, which both require a different method of data generation. For face verification it is only important to be able to generate several images belonging to the same identity. For face recognition it is also important to be able to generate several images belonging to the same identity, but the images also have to resemble some existing identity.

3.6.1 Training Procedure

The BEGAN itself was trained on the CelebA dataset for approximately 70 hours on an Nvidia GTX 1070 with 8 GB of GDDR5 RAM. The image size was set to 128×128 pixels and the dimension of the input vector $\mathbf{z}$ was set to 64. The equilibrium parameter was set to $\gamma_e = 0.7$. The training procedure is unsupervised, which means that no manual data labeling is required.

3.6.2 Dataset Generation

3.6.2.1 Face Verification

To be able to use generated images for fine-tuning a face verification network, images that seem to belong to the same identity need to be grouped together. We use a simple approach where the input vector $\mathbf{z}$ is sampled in a uniformly random way to create images that seem to belong to different people. For every input vector that represents a different identity, we randomly generate new input vectors using a multivariate normal distribution with the original input vector acting as the mean of the distribution and using a diagonal covariance matrix $\sigma^2 I$, where I is the identity matrix. For small values of σ^2 the generated input vectors should be different to the mean but fairly close and should hopefully create synthetic images $G(\mathbf{z})$ similar to the mean image but with slight variations.

3.6.2.2 Face Recognition

We also use the BEGAN to generate images that look similar to some existing identity. For some image $\mathbf{x}$, this is achieved by choosing a random sample vector $\mathbf{z} \in [-1, 1]^{N_z}$ where $N_z = 64$ and generating an image $G(\mathbf{z})$. The following optimization problem is then solved

$$\underset{\mathbf{z} \in \mathbb{R}^{64}}{\operatorname{argmin}} |\mathbf{x} - G(\mathbf{z})| \tag{3.7}$$

to find the sample vector that best represents the identity in the image $\mathbf{x}$. In theory the optimization should be subject to $\mathbf{z} \in [-1, 1]^{64}$ but this is ignored for the sake of simplicity, hoping that the optimal vector $\mathbf{z}$ will be a valid vector anyway. Even if the optimal vector is not valid, it can still be used as input to the generator. However, the generated output from a invalid input vector is unlikely to look like a face. Assuming that the face in image $G(\mathbf{z})$ belongs to the identity in image $\mathbf{x}$ it would then be possible to generate more images belonging to that identity, using the same scheme as for creating datasets for face verification.

3.6.3 Evaluation

In order to evaluate if the generated datasets are useful, datasets containing approximately $50,000$ images each were generated for the task of fine-tuning face verification networks. The datasets were generated with a different number of images per identity to see how that affects the performance. For the covariance matrix, $\sigma^2 = 0.08$ was used as this seemed to generate images that had a decent amount of variation within each identity but still not too much to make them look like different identities. The number of images per identity and the number of identities for the generated datasets are shown in Tab. 3.4.

Images per identity	Number of identities	Total number of images
16	$3,125$	$50,000$
32	$1,563$	$50,016$
64	782	$50,048$
128	391	$50,048$

Table 3.4: Four datasets were generated using the BEGAN with a varying amount of images per identity.

Chapter 4

Results and Discussion

The results obtained in this thesis work are presented and discussed in this chapter. We have fine-tuned face verification networks using a variety of datasets with different augmentations, and we have explored the difference between center loss and triplet loss for these networks. We have also used the pre-trained face verification network to perform face recognition on the Axis internal dataset and the FERET dataset. We show results of the synthetic images generated using the different methods as well to perform visual assessment of the quality of the synthetic images.

4.1 Face Verification Using Fine-tuned Networks

4.1.1 Center Loss

Fig. 4.1 shows the ROC curve for different augmentation strategies when fine-tuning using center loss. Tab. 4.1 shows the AUC and the accuracies with their standard deviations for the different augmentation methods.

	AUC	Accuracy
Real images ($54,139$ images)	0.960	0.894 ± 0.019
Real images ($10,309$ images)	0.871	0.791 ± 0.018
SFM ($10,309 \rightarrow 50,000$ images)	0.903	0.827 ± 0.011
2D ($10,309 \rightarrow 50,000$ images)	0.936	0.865 ± 0.015
MFM ($10,309 \rightarrow 50,000$ images)	0.927	0.853 ± 0.016

Table 4.1: The AUC for the curves in Fig 4.1.

The results in Tab. 4.1 clearly show that using more data is useful as the area under the curve and the accuracy for the dataset containing $10,309$ real images is significantly smaller than the area under the curve and the accuracy for the dataset containing $54,139$ real images. The augmented datasets all show an improvement compared to the dataset containing $10,309$ images but none of them can quite reach the same accuracy as the larger dataset with $54,139$ real images.

Among the augmented datasets the conventional 2D augmentation technique works best while the datasets augmented using the 3D augmentation techniques are not quite as successful. The reason for this might be that when performing the 3D reconstruction of faces, several important details such as hair and ears are missing as the 3D model does not cover these areas of the face. Even though the 3D model can be rendered from a variety of different poses it still lacks some fundamental features.

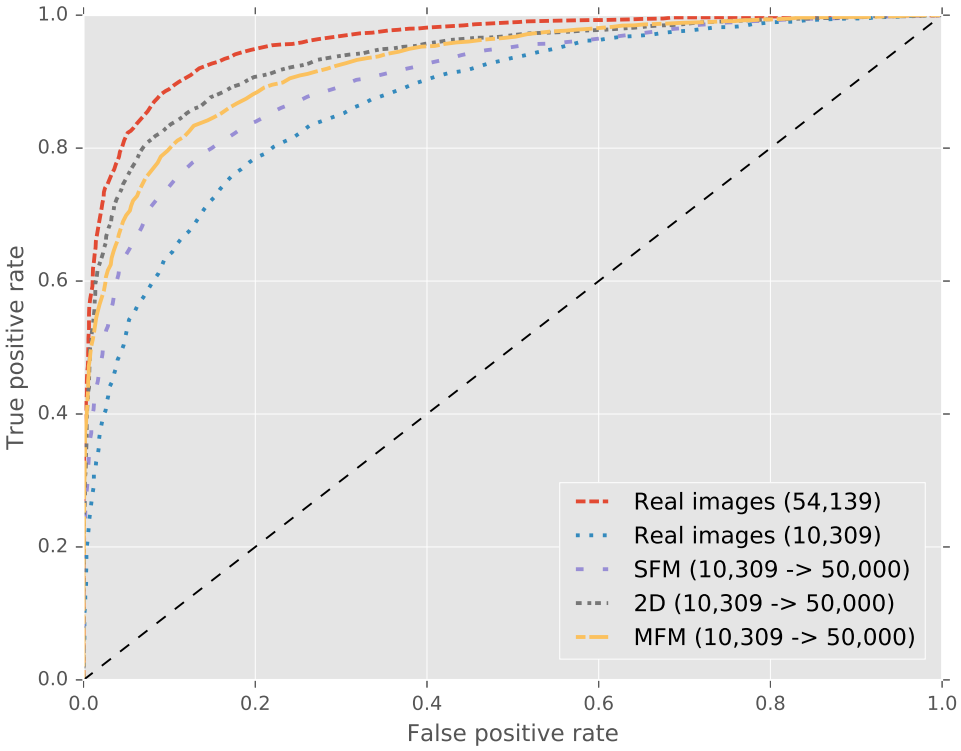


Figure 4.1: The figure shows the ROC curves when evaluating on the LFW dataset, using different training datasets and fine-tuning with center loss. The red curve represents the performance when fine-tuning using $54,139$ real images while the blue curve represents the performance when fine-tuning using $10,309$ real images. The yellow, black and purple curves represent the performance when the $10,309$ real images have been augmented to a dataset of approximately $50,000$ images using the different augmentation techniques.

Another problem with the 3D augmentation techniques is that the reconstruction procedure can fail and create low quality synthetic augmentations. In the worst case the synthetic images might no longer represent the correct identity. This seems to occur more often for the SFM which might explain why it cannot perform quite as well as the MFM. The MFM is capable of predicting when a synthetic image is going to be bad and can therefore reduce the number of bad images. No such fail-safe is implemented for the SFM.

The benefit of the 3D augmentations is that they provide more variation within the identities. If the quality of the synthetic images would be better, they could potentially outperform the 2D augmentations.

4.1.2 Triplet Loss

Fig. 4.2 shows the ROC curves for different augmentation strategies when fine-tuning using triplet loss. Tab. 4.2 shows the AUC and the accuracies with their standard deviations for the different augmentation methods.

Tab. 4.2 and 4.1 show that using triplet loss leads to worse performance than using center loss. Triplet loss seems to require a lot more data and training time in order to

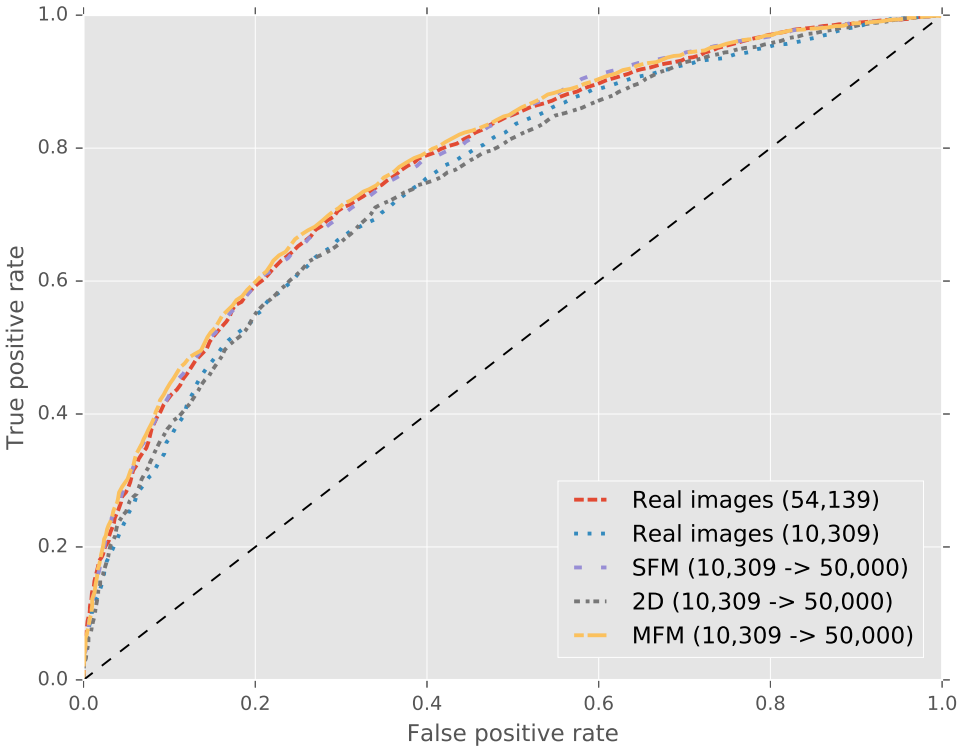


Figure 4.2: The ROC curves when evaluating on the LFW dataset, using different training datasets and fine-tuning with triplet loss. The red curve represents the performance when fine-tuning using $54,139$ real images while the blue curve represents the performance when fine-tuning using $10,309$ real images. The yellow, black and purple curves represent the performance when the $10,309$ real images have been augmented to a dataset of approximately $50,000$ images using the different augmentation techniques.

	AUC	Accuracy
Real images ($54,139$ images)	0.773	0.703 ± 0.010
Real images ($10,309$ images)	0.749	0.679 ± 0.019
SFM ($10,309 \rightarrow 50,000$ images)	0.774	0.696 ± 0.016
2D ($10,309 \rightarrow 50,000$ images)	0.746	0.686 ± 0.019
MFM ($10,309 \rightarrow 50,000$ images)	0.779	0.707 ± 0.011

Table 4.2: The AUC for the curves in Fig 4.2.

achieve high performance compared to center loss. According to [34], 100M-200M face images consisting of 8M different identities were used as training data when training the FaceNet model using triplet loss. The model was also trained for approximately $1,000$ to $2,000$ hours. Comparing this to experiments performed by [38], only 0.7M face images consisting of $17,189$ identities were used and the model was only trained for 14 hours using center loss while still achieving almost as good accuracy. This seems to suggest that center loss might be more suitable in our setting and might explain the lower performance using triplet loss. It is worth noting that triplet loss currently has the highest performance on the LFW dataset, at least as far as we know.

The difference between the different augmentation methods are difficult to analyze

for triplet loss as the performance is quite similar. It is still possible to see that using the large dataset of real images performs better than using the small dataset of real images, as expected.

4.1.3 Generative Adversarial Networks

Fig. 4.3 shows the ROC curve when fine-tuning using center loss and training with the datasets generated using the BEGAN, as mentioned in section 3.6.3. Tab. 4.2 shows the area under the curve AUC and the accuracies with their standard deviations for the different synthetically generated datasets.

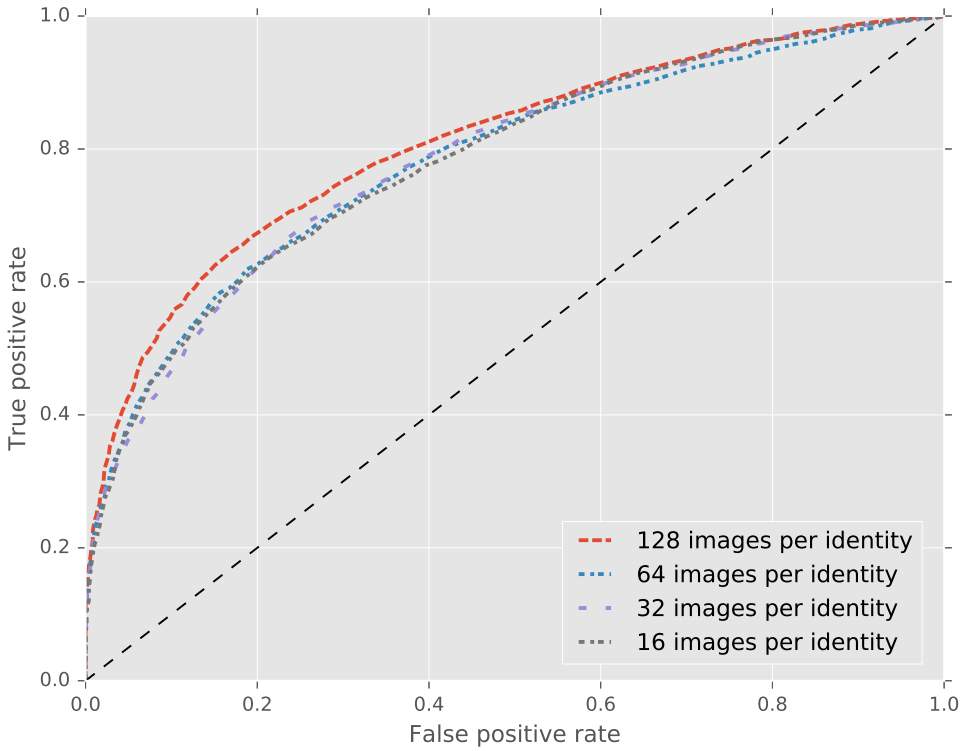


Figure 4.3: The ROC curves when evaluating on the LFW dataset, using BEGAN to synthetically generate training datasets and fine-tuning with center loss. For the four different curves approximately $50,000$ training images were used.

	AUC	Accuracy
128 images per identity	0.807	0.733 ± 0.014
64 images per identity	0.782	0.716 ± 0.015
32 images per identity	0.785	0.713 ± 0.020
16 images per identity	0.783	0.710 ± 0.020

Table 4.3: The AUC for the curves in Fig 4.3.

The performance, when training with completely synthetic datasets, cannot quite reach the same performance as training with only real images. As can be seen from the results in Section 4.4 the BEGAN can generate images that look realistic, however our method of generating the synthetic datasets probably do not introduce enough variation

for each identity to successfully train a face verification network. However, the fact that better than random performance can be achieved using only synthetic data suggests that this method could have potential.

It seems beneficial to use more images per identity and fewer identities than to use few images per identity and many identities, as using 128 images per identity outperforms the other methods. Using 16, 32, or 64 images per identity hardly seems to make a difference as the standard deviation of the accuracies is too large to claim that one is better than the other. The authors of [22] state that it is not only important to have many identities in order to learn differences between the identities. It is also important to have many images per identity to learn the possible variations within an identity. This might explain why using 128 images per identity is the preferable method.

It is also worth considering which dataset construction strategy is most suitable for the BEGAN. A generator could in theory learn to generate any image but in practice this might not occur. There is a limit to how diverse the images generated by the trained BEGAN can be. By creating many different identities there is a risk of having several identities that look very similar. The BEGAN can also fail to create a realistic looking face for certain input vectors. The question is then if it is better to have a few identities where many images look unrealistic or to have many identities where a few images look unrealistic.

4.2 Face Recognition Using the Pre-trained Network

4.2.1 Axis Internal Dataset

The Axis internal dataset acts as an evaluation dataset containing surveillance images. Face recognition is performed using the Axis employee database as the training dataset. To get a baseline performance for the face recognition system the precision-recall curve is calculated for the employee database without any additional augmentations. The results are shown in Fig. 4.4. Different augmentation techniques are then evaluated and presented below.

4.2.1.1 Augmentations Using the SFM

4.2.1.1.1 Finding Ideal Augmentations

Using the SFM we are capable of producing a wide range of different augmentations to the original dataset, as the model can be rotated around freely in 3D space. The reconstructed 3D models of the images in the Axis employee database are rotated and rendered at a variety of different pitch and yaw angles. The precision-recall curve and the average precision are calculated for the synthetic images generated using these pitch and yaw values, in order to find the optimal angles. The average precision for these synthetic images are shown as a heat map in Fig. 4.5. The top-10 yaw and pitch combinations based on the heat map are shown in Tab. 4.4 along with their corresponding average precision scores. When constructing the heat map, the Axis employee database has not been augmented with the synthetic images. Instead, the real images in the database are replaced with the synthetic ones in order to see how well they perform alone.

From the heat map in Fig. 4.5 one can see that images that are rendered slightly from above with a pitch angle of around -0.1 radians, and slightly from the right with a yaw angle of around -0.3 radians, seem to achieve the best performance. Overall, images rendered from the right seem to perform better than images rendered from the

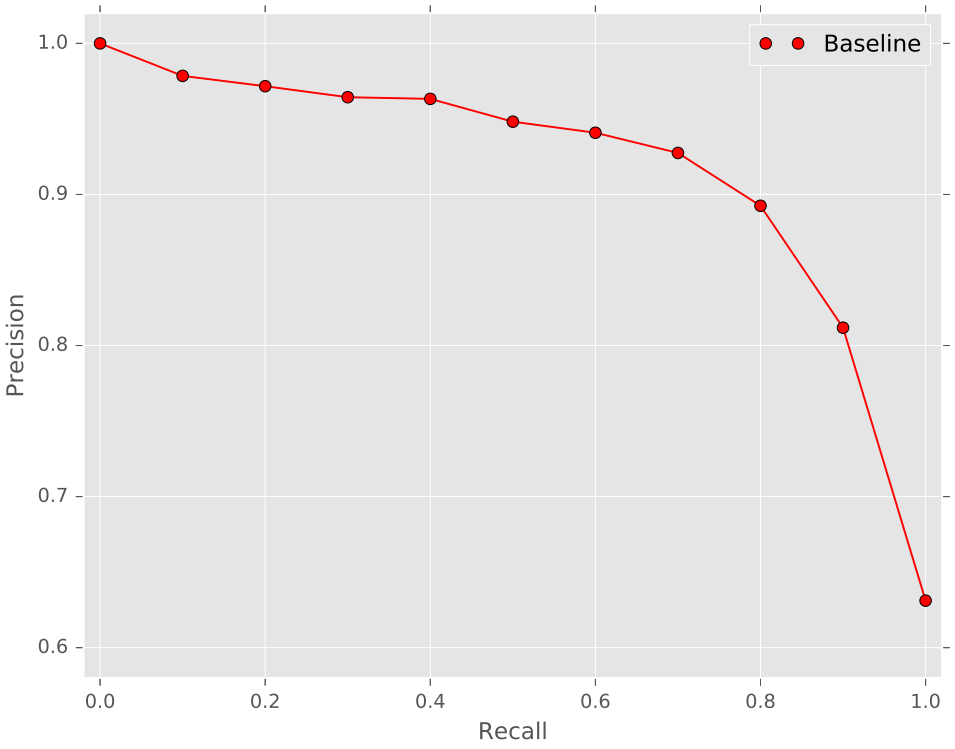


Figure 4.4: The baseline precision-recall using the Axis internal dataset for evaluation and the Axis employee database as the training dataset. The AP is 0.912.

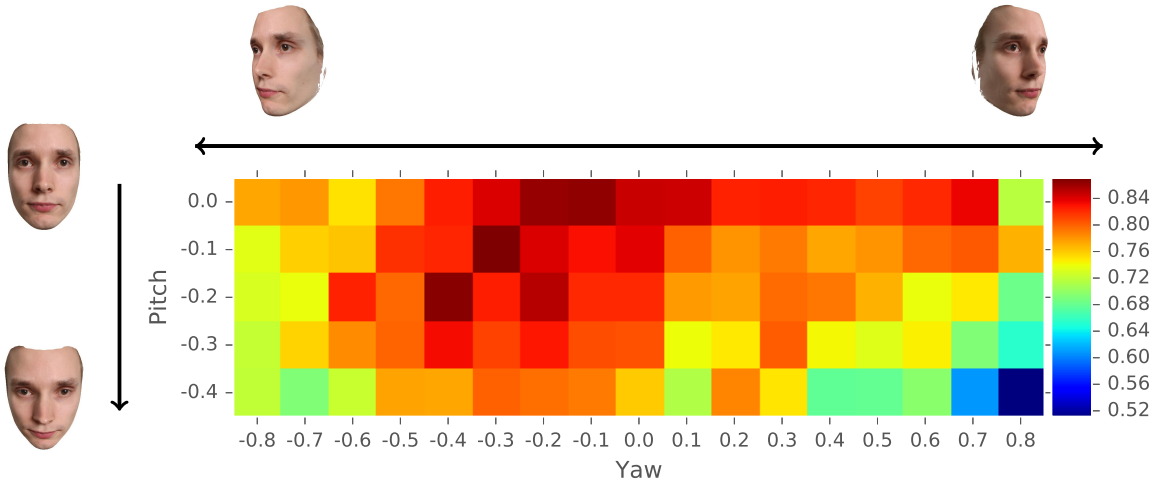


Figure 4.5: A heat map for different pitch and yaw angles. Each square contains the average precision for that pitch and yaw. A negative pitch angle indicates that the image is rendered from above. A negative yaw angle indicates that the image is rendered from the right, and a positive angle that the image is rendered from the left. When both the pitch and yaw are zero the camera is facing straight at the 3D model.

left. In fact, the mean average precision of all images rendered from the right is 0.796

Pitch (radians)	Yaw (radians)	AP
-0.1	-0.3	0.871
-0.2	-0.4	0.867
0	-0.1	0.864
0	-0.2	0.863
-0.2	-0.2	0.853
0	0	0.848
0	0.1	0.847
0	-0.3	0.843
-0.1	-0.2	0.842
-0.1	0	0.839

Table 4.4: The ten best performing pitch and yaw angle combinations, along with their average precision, based on the heat map in Fig. 4.5.

while the mean average precision of all images rendered from the left is 0.755. The Axis internal dataset is gathered from only a few different surveillance cameras, which results in most images being captured from the same direction. Surveillance cameras are also often mounted to capture images from above. These two factors could explain why certain pitch and yaw angles perform better than others. By visually examining the Axis internal dataset, one can see that there indeed seem to be more images taken from the right than from the left. The images that are taken from the left also seem to have a higher yaw angle which also would agree with the heat map.

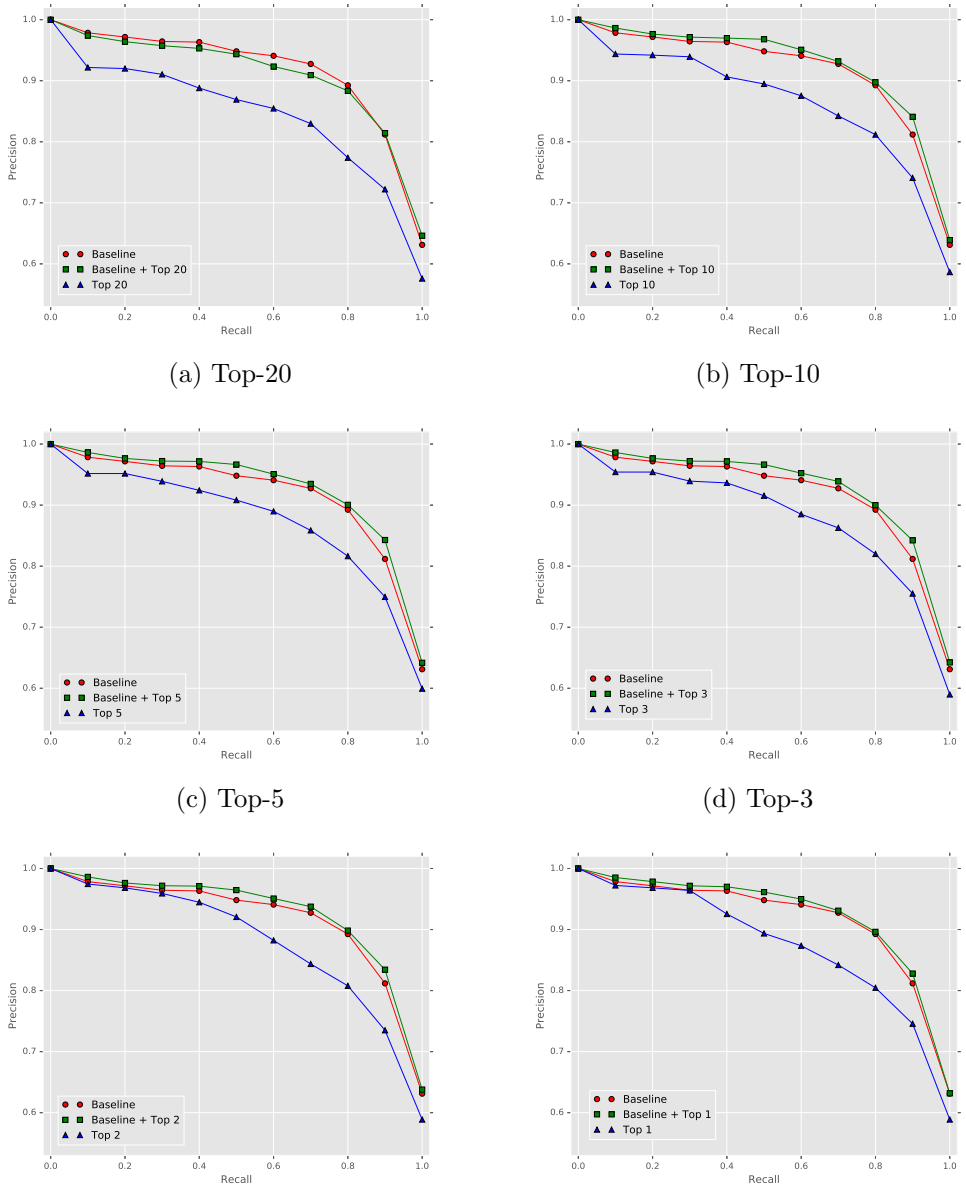
For more extreme yaw and pitch angles the area under the curve starts to deteriorate. This might be caused by the evaluation dataset not containing images taken at such wide angles. It might also be caused by the deteriorating quality of the images rendered at wide yaw and pitch angles.

In order to further explore this result, the original database of Axis employee images is augmented based on the best performing yaw and pitch values from the heat map. The 20 best performing yaw and pitch combinations are used to augment the original database of employee images. The synthetic images are also used without the original real images. The results are shown in Fig. 4.6 where the top 20, 10, 5, 3, 2 and 1 synthetic images are used to augment the employee database. The corresponding average precision scores are shown in Tab. 4.5.

	Only synthetic	Synthetic + real	Only real
Top 20	0.842	0.906	0.912
Top 10	0.862	0.921	0.912
Top 5	0.872	0.922	0.912
Top 3	0.874	**0.923**	0.912
Top 2	0.875	0.921	0.912
Top 1	0.871	0.918	0.912

Table 4.5: The average precision for the top 20, 10, 5, 3, 2 and 1 synthetic images. The largest average precision score is written in bold text.

Fig. 4.6 and Tab. 4.5 show that when augmenting the Axis employee database using synthetic images generated using the SFM, the performance increases. The top 3 augmentations where three synthetic images and one real image is used performs the best. As the database of real images is augmented further the performance starts to



(a) Top-20 (b) Top-10

(c) Top-5 (d) Top-3

(e) Top-2 (f) Top-1

Figure 4.6: The precision-recall curves for the top 20, 10, 5, 3, 2 and 1 synthetic images. Each graph contains three precision recall curves. One for the original images (red), one for the original and the synthetic images (green) and one for only the synthetic images (blue).

decrease. According to the top 20 average precision score, the augmentations seem to be doing more harm than good.

As more and more synthetic images are added to the database of real images, more noise might also added to the augmented database if the synthetic images do not look realistic enough. If the classifier cannot handle high levels of noise, it might lead to augmentations causing more miss-classifications than correct classifications. This will lead to a lower performance compared to not using any augmentations at all.

4.2.1.1.2 Using Generic Augmentation

The best performing synthetic images are chosen based on their performance on the Axis internal dataset. If one does not know the most common yaw and pitch angles of the dataset beforehand it is not possible to find the ideal augmentations. We therefore also choose augmentations that would make sense to use on an unseen dataset. In particular we augment the employee database symmetrically to get side views from both left and right. The average precision for the different symmetrical augmentations is shown as a heat map in Fig. 4.7.

The precision-recall curve for the best symmetric augmentation strategy is shown in Fig. 4.8. The average precision for the four best augmentation strategies are also given in Tab. 4.6.

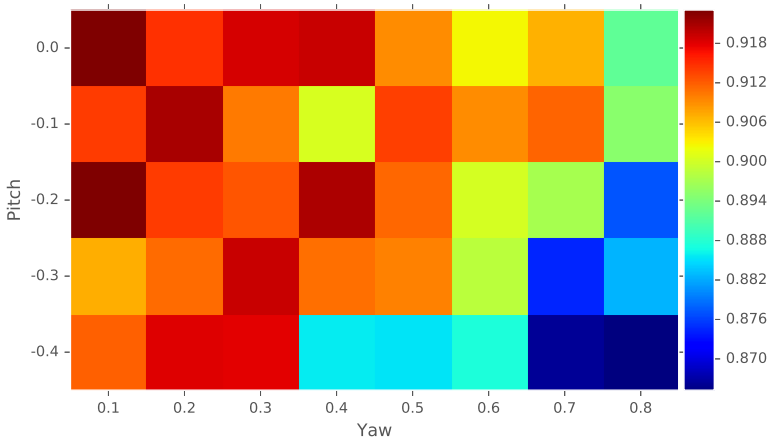


Figure 4.7: A heat map showing the AP for different tilts and symmetrical augmentations. The yaw values on the x-axis indicate that the original images have been augmented using synthetic images rendered at that yaw angle, both from the left and the right side. As an example the augmented database used for the AP in the top left corner contains the original image, a synthetic image with pitch 0 radians and yaw 0.1 radians and another synthetic image with pitch 0 radians and yaw -0.1 radians.

	AP
Pitch: 0 Yaw: -0.1, 0.1	0.923
Pitch: -0.2, Yaw: -0.1, 0.1	0.923
Pitch: -0.1, Yaw: -0.2, 0.2	0.921
Pitch: -0.2, Yaw: -0.4, 0.4	0.921

Table 4.6: The average precision for the four best symmetric augmentations, evaluated on the Axis internal dataset. The highest performing augmentation strategy, pitch 0 radians and yaw -0.1 and 0.1 radians is also shown in Fig. 4.8

The symmetric augmentations give a similar performance compared to choosing the optimal augmentations. Some of the best performing pitch and yaw angles for the symmetric augmentations also seem to appear among the optimal augmentations. As an example, pitch -0.2 radians at yaw -0.4 radians is the second best synthetic image according to Tab. 4.4 but it is also included in the fourth best symmetric augmentation. Even though pitch -0.2 radians at yaw 0.4 radians is not a part of the top perform-

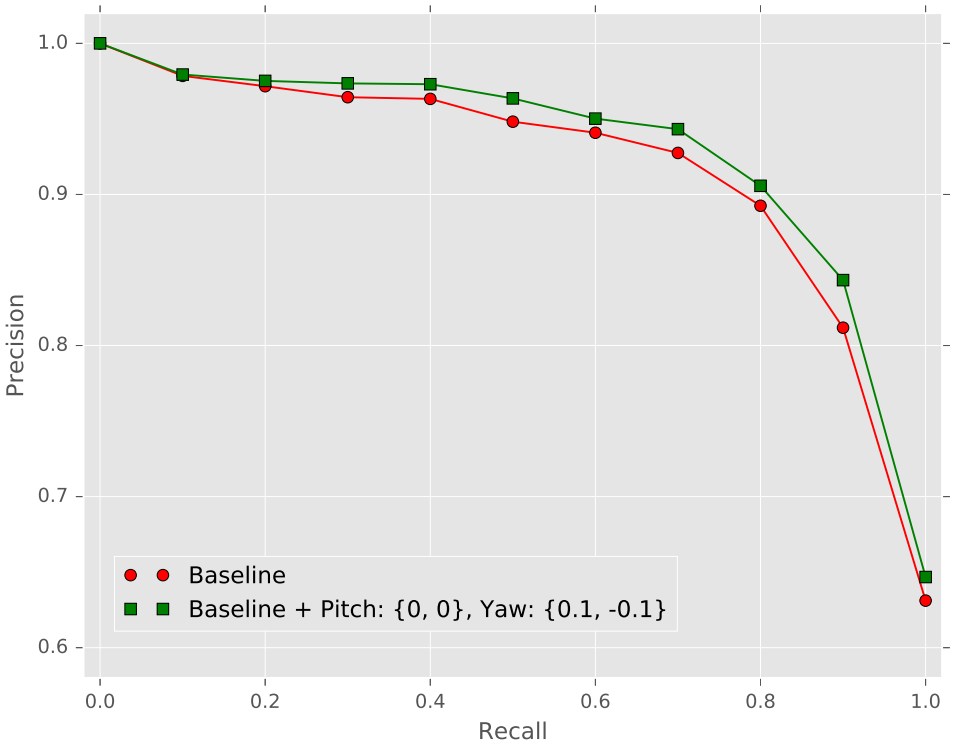


Figure 4.8: The precision-recall curves for the best performing symmetric augmentation strategy, and the baseline. Both are evaluated on the Axis internal dataset.

ing synthetic images it still does not seem to affect the performance of the symmetric augmentation in a negative way.

4.2.1.2 Augmentations Using the MFM

This augmentation scheme is not as flexible as the SFM and can not be rotated around freely in 3D space. Therefore the only synthetic images used are images rotated 40 degrees (approximately 0.7 radians) to the left and to the right at a tilt angle of 0 degrees. The images rendered at 75 degrees are not used since images with such a wide angle do not exist in the Axis internal dataset. Fig. 4.9 shows the precision-recall curves for these augmentations and Tab. 4.7 shows the average precision scores.

	AP
Baseline	0.912
Baseline + synthetic 40	0.912
Baseline + synthetic -40	0.907
Baseline + synthetic $-40, 40$	0.907
Synthetic 40	0.826
Synthetic -40	0.797

Table 4.7: The average precision for the precision-recall curves from Fig. 4.9.

The results using the MFM are not quite as good compared to the SFM. According to Tab. 4.7 it is the database of real images that performs best and any additional

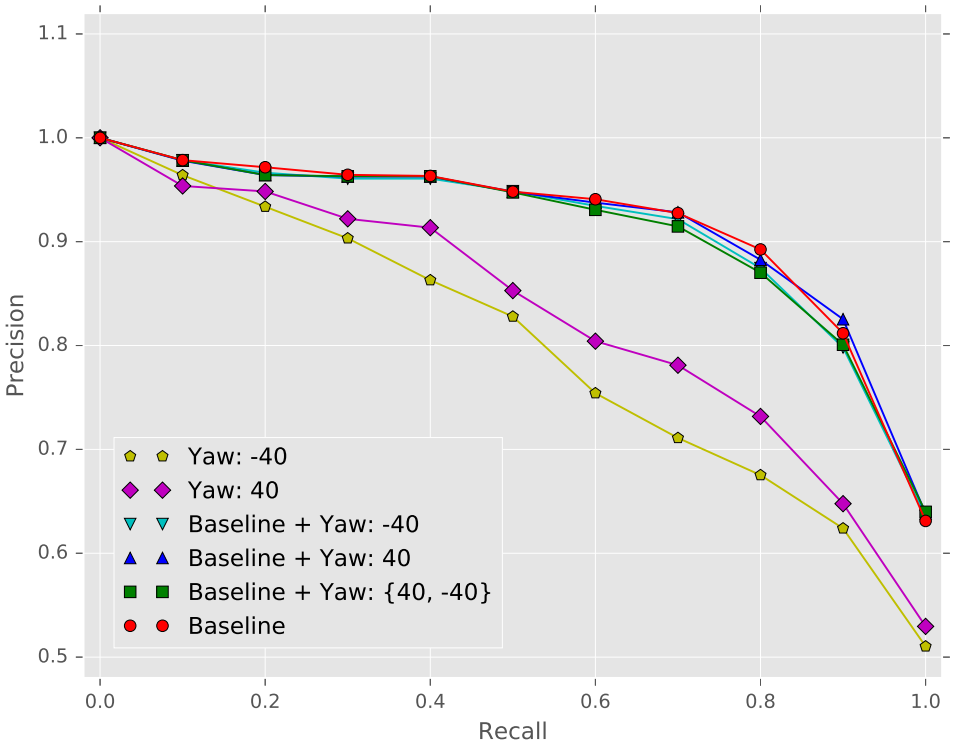


Figure 4.9: The precision-recall curves for the baseline and the different augmentation techniques, evaluated on the Axis internal dataset.

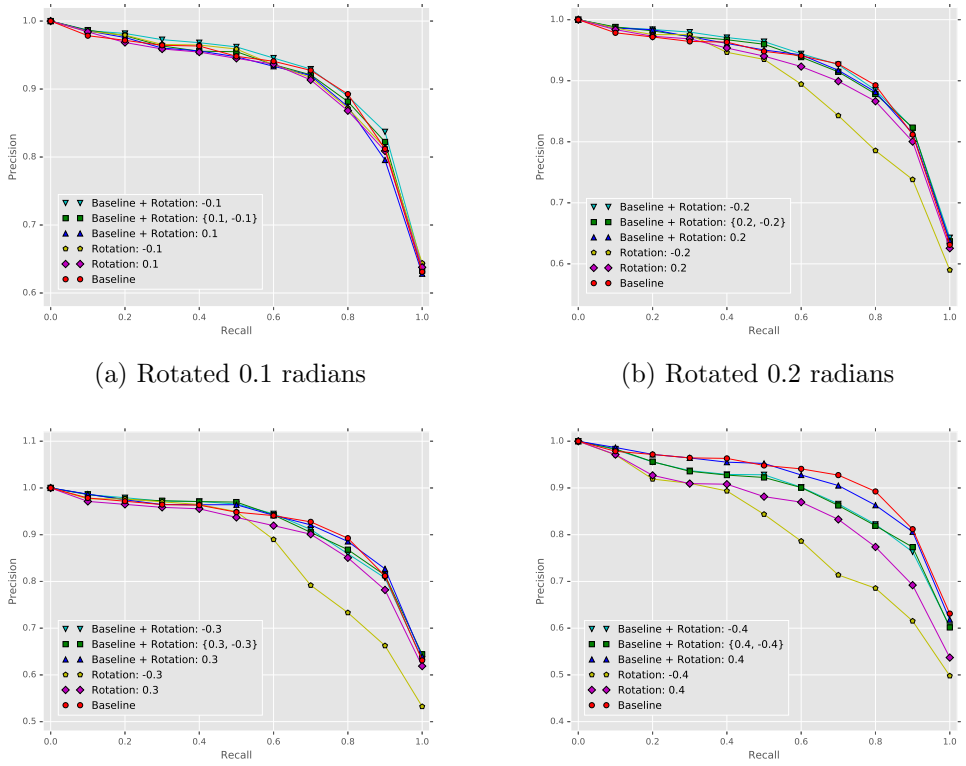
augmentations only decreases the performance. This is perhaps not surprising as the synthetic images can only be rendered at fixed angles that might be too large for this purpose. The best performing augmented dataset is the one using the original images and synthetic images rendered 40 degrees from the left. This seems to contradict the idea that images rendered from the right should perform better on the Axis internal dataset. However, according to the heat map in Fig. 4.5 it seems like slightly wider angles (0.5 to 0.7 radians) also perform better at images rendered from the left for the SFM, which agrees with the results for the MFM.

4.2.1.3 Augmentations Using Conventional 2D Techniques

We create 2D augmentations by performing in-plane rotations of the images in the Axis employee database. Fig. 4.10 shows the result using rotated images as augmentations to the database. Tab. 4.8 shows the average precision for these methods.

For certain rotations the performance increases but not quite to the same level as for the SFM. The performance increase might be related to the fact that some faces in the Axis internal dataset are slightly in-plane rotated.

Comparing the performance of the 2D rotated images alone with the 3D reconstructed images alone, one can see that the 2D rotated images clearly outperform the 3D reconstructed images. Still the 3D reconstructed images together with the original images yields better performance, see Tab. 4.9. This suggests that it is not as easy as simply choosing the best performing augmentation methods and combining them. Instead, using methods that contribute new information might be useful even if they



(a) Rotated 0.1 radians

(b) Rotated 0.2 radians

(c) Rotated 0.3 radians

(d) Rotated 0.4 radians

Figure 4.10: The precision-recall curves for augmentations using 2D rotations. Each graph shows the results for one rotation angle. The baseline is shown along with five additional augmentation strategies. The Axis internal dataset is used for evaluation.

	0.1 radians	0.2 radians	0.3 radians	0.4 radians
Baseline	0.912	0.912	0.912	0.912
Baseline + CW and CCW	0.912	0.915	0.913	0.880
Baseline + CCW	0.919	0.919	0.912	0.881
Baseline + CW	0.907	0.913	0.915	0.905
CCW	0.912	0.879	0.859	0.803
CW	0.907	0.903	0.896	0.846

Table 4.8: The average precision for the baseline and the augmentations performed using 2D rotations, as is shown in Fig. 4.10. CW stands for clockwise rotation while CCW stands for counter clockwise rotation.

are not always perfect on their own.

We also experiment with what happens when we reduce the quality of the original high resolution images. The precision-recall curves are shown in Fig. 4.11. The performance seems to decrease when reducing the image quality. It seems to indicate that high quality is a key component for achieving high performance. This might partially explain why some synthetic images perform worse than realistic ones, as their quality often degrades as a part of the texture fitting process.

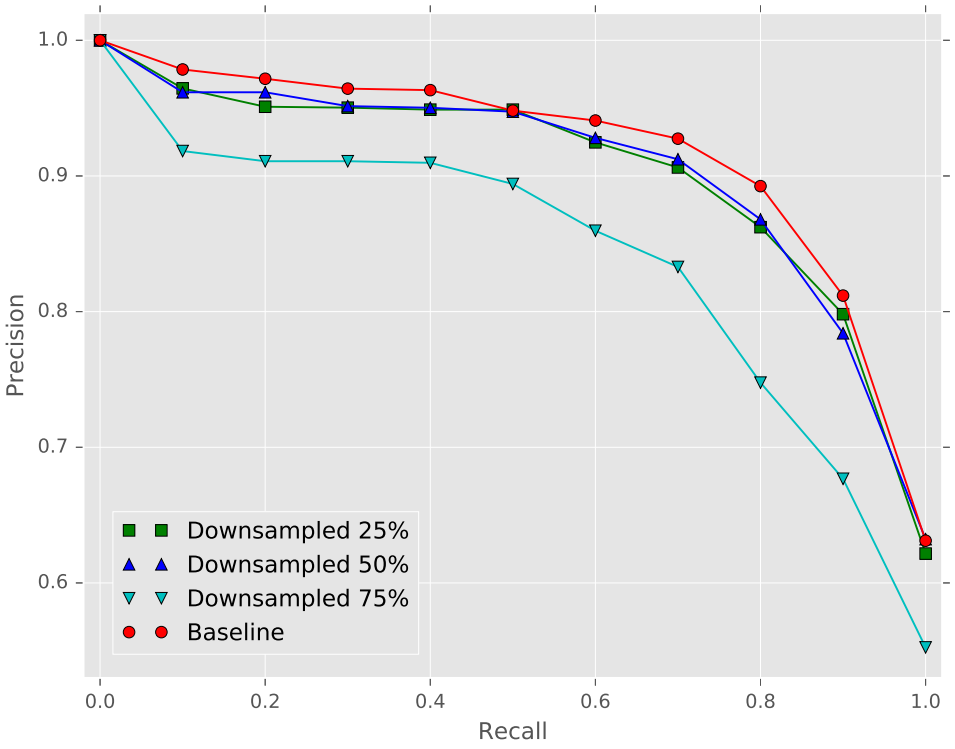


Figure 4.11: The precision-recall curves for the baseline and the downsampled versions of the original Axis employee database. An image that is downsampled 75% is first resized to 75% of its original size and then resized back to its original size, degrading the image quality. The evaluation is performed on the Axis internal dataset.

4.2.1.4 Comparing the Augmentation Methods

We compare the best performing SFM, MFM and 2D augmentation strategies in order to see which one is preferable. Precision-recall, CMC and ROC are used to perform the evaluation.

4.2.1.4.1 Precision-Recall

The precision-recall curves for the different augmentation methods are shown in Fig. 4.12 and their average precision's are shown in Tab. 4.9.

Method	Augmentation Strategy	AP
Baseline	N/A	0.912
SFM	Pitch: 0 rad Yaw: $-0.1, 0.1$ rad	0.923
MFM	Pitch: 0° Yaw: 40°	0.912
2D	CCW, rotated 0.1 rad	0.919

Table 4.9: The average precision for the baseline and the augmentations performed using 2D rotations, as is shown in Fig. 4.12. CCW stands for counter clockwise rotation.

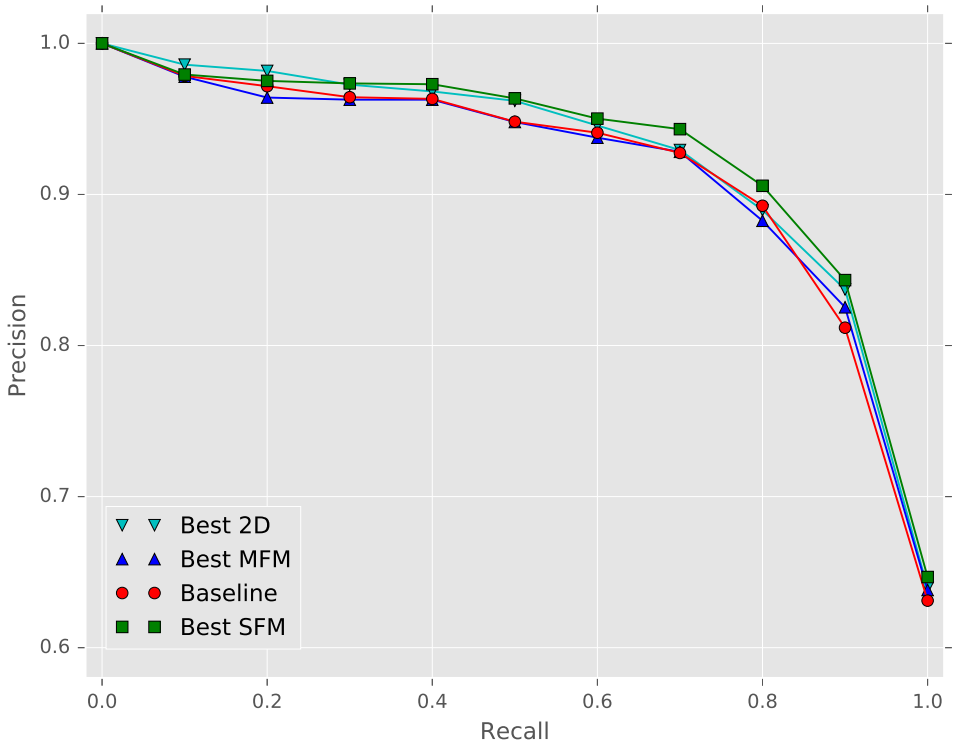


Figure 4.12: The precision-recall curves for the baseline and the best performing augmentation strategies for the different methods. The evaluation is performed on the Axis internal dataset.

4.2.1.4.2 Cumulative Match Characteristic

The CMC are also calculated for the best performing augmentation strategies for each method. The result is shown in Fig. 4.13. The accuracies at rank 1, 10, 50 and 100 are also shown in Tab. 4.10

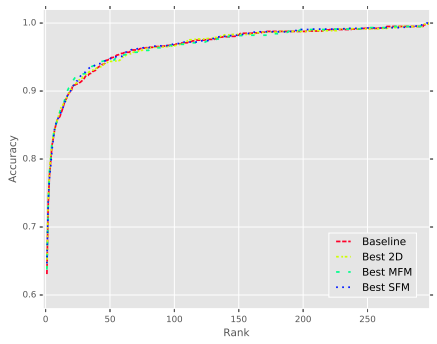
	Rank-1	Rank-10	Rank-50	Rank-100
Baseline	0.631	0.860	0.946	0.968
Best SFM	0.646	0.861	0.948	0.970
Best MFM	0.637	0.861	0.954	0.967
Best 2D	0.639	0.863	0.944	0.967

Table 4.10: The rank 1, 10, 50 and 100 accuracy for the best performing augmentation strategies for the different methods, based on the results from Fig. 4.13.

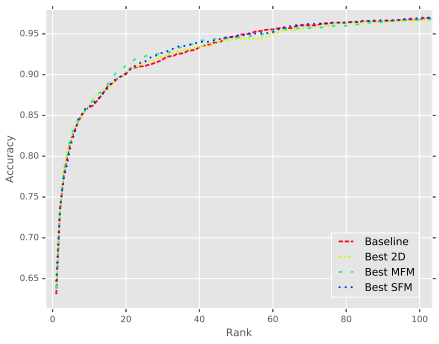
4.2.1.5 Receiver Operator Characteristic

The ROC curves and the AUC are calculated for the best performing augmentation methods, as mentioned in section 3.3.3. The results are presented in Fig. 4.14 and Tab. 4.11.

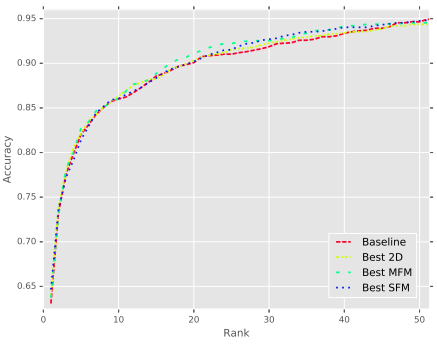
Based on the results from Fig. 4.12 and Tab. 4.9, it seems like the SFM and the 2D augmentations are capable of performing better than the baseline, while the MFM



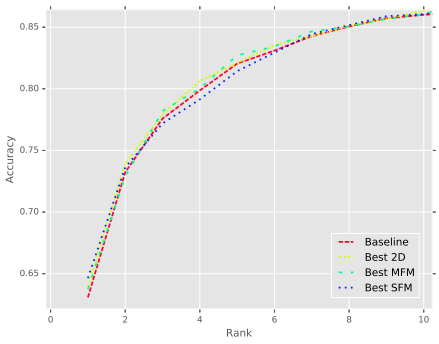
(a) Up to rank 298



(b) Up to rank 100

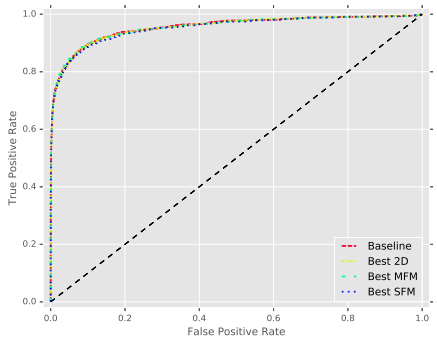


(c) Up to rank 50

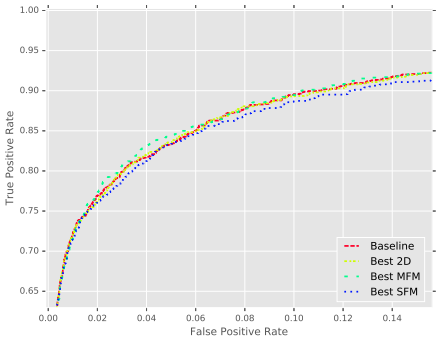


(d) Up to rank 10

Figure 4.13: The CMC curves for the best performing augmentation strategies for each method. Each sub-figure displays the same CMC curves but is gradually zoomed at lower ranks to show more details. The evaluation is performed on the Axis internal dataset.



(a) The whole ROC curve



(b) The top left corner of the ROC curve

Figure 4.14: The ROC curves for the best performing augmentation strategies for each method. The evaluation is performed on the Axis internal dataset.

cannot improve performance. Both the SFM and the 2D augmentations have the benefit of being able to create a wide variety of different augmentations, from which we have chosen the best performing ones. Perhaps the MFM could achieve similar or better performance if it had the same flexibility as the SFM.

The CMC in Fig. 4.13 and Tab. 4.10 show that all methods have quite similar

	AUC
Baseline	0.955
Best SFM	0.953
Best MFM	0.955
Best 2D	0.955

Table 4.11: The AUC for the best performing augmentation strategies for the different methods, based on the results from Fig. 4.14.

performance as the rank is increased. The largest difference is seen at rank-1 accuracy where all three augmentation methods achieve a higher accuracy than the baseline. For low rank values the accuracy initially increases very quickly. This indicates that the face recognizer is often close to finding the correct answer. This could be useful in practice as a face recognition system could potentially return the five best matches and let a human perform the final identification among the top 5 manually.

The ROC curves in Fig. 4.14 and the AUC scores in Tab. 4.11 show very little difference between the methods. The SFM achieves a slightly lower AUC score compared to the other methods, which likely occurs since the true positive rate for the SFM is also slightly lower at false positive rates in the range $[0.02, 0.04]$ and $[0.06, 0.20]$. Ideally one would like to have a true positive rate of 1 with a false positive rate of 0. This is not quite achieved but the true positive rate increases quickly for low false positive rates which is a good sign as one would like to have many true positives while keeping the false positives at a minimum.

Overall the SFM seems to be the most desirable option for data augmentation when evaluating on the Axis internal dataset. While it can only slightly improve the recognition performance compared to the other methods, it seems to have the highest potential as it can be freely rotated in 3D space. The 2D augmentations can also achieve good performance and can be useful, especially if the testing dataset contains in-plane rotated face images. The MFM does not work quite as well on the Axis internal dataset as it is only capable of generating images that rarely appear in the testing set. One additional advantage the SFM has over the MFM and the 2D augmentations when evaluating on the Axis internal dataset is that the training images in the Axis employee database have very high quality and resolution. This is critical for the SFM to perform well but not quite as important for the MFM or the 2D augmentations method.

4.2.2 FERET Dataset

The initial recognition performance, using only one frontal image per identity, on the *f* test subset from FERET dataset can be seen in Fig. 4.15. Here the the *Original* subset of frontal images are used as an identity gallery in order to classifly test images from the frontal testset from Tab. 3.3, *f*. The AP is 0.990 and is calculated using eleven-point average.

4.2.2.1 Augmentations Using the SFM

Testing against synthesized images at yaw angles corresponding to the subsets of Tab. 3.3 gives us the plot in Fig. 4.16a. Models with slight yaw to the right seem to produce slightly higher recognition scores than the real frontal images in the *Original* subset. Changing the testing set to the *ql* produces the results in plot in Fig. 4.16c. Plots corresponding to subsets *ql*, *qr*, *hl*, *hr*, *pl*, *pr* and *r* can be seen in Figs. 4.16c, 4.16d,

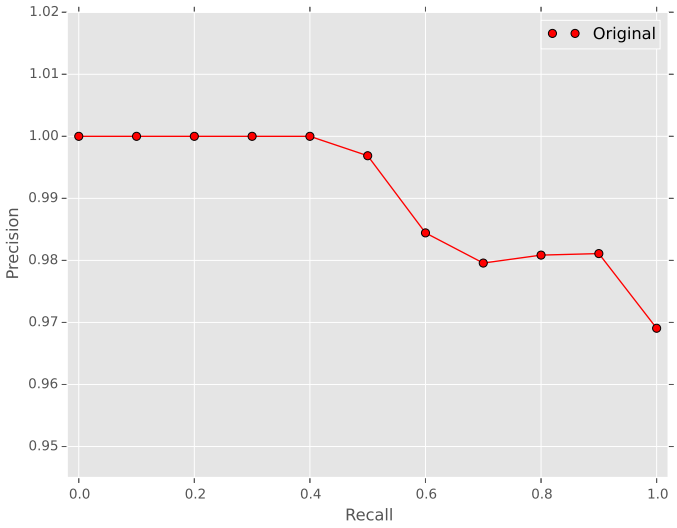


Figure 4.15: Precision recall curve, *f* subset verified against the *Original* subset of the FERET dataset

Pose\Subset	*f*	*r*	*ql*	*qr*	*hl*	*hr*	*pl*	*pr*
Original	*0.9902*	**0.9966**	**0.9982**	*0.9976*	**0.9977**	**0.9944**	**0.9195**	**0.8268**
yaw 0.0	0.9834	0.9893	0.9887	0.9957	0.9879	0.9816	0.8210	0.6869
yaw -0.4	0.9893	0.9897	0.9942	0.9956	*0.9900*	0.9786	*0.8625*	0.7688
yaw 0.4	**0.9941**	*0.9946*	*0.9960*	**0.9980**	0.98845	*0.9881*	0.8302	*0.7688*
yaw -1.2	0.9232	0.9449	0.9450	0.9534	0.9425	0.8679	0.8001	0.7054
yaw 1.2	0.9098	0.9473	0.9473	0.9603	0.8820	0.9193	0.6325	0.7357
yaw -1.5	0.5486	0.5772	0.5772	0.5766	0.5901	0.4975	0.3605	0.3176
yaw 1.5	0.3938	0.4982	0.5637	0.5795	0.4558	0.5157	0.2426	0.3236
From Fig.	4.16a	4.16b	4.16c	4.16d	4.16e	4.16f	4.16g	4.16h

Table 4.12: AP for each subset and orientation augmentation, corresponding to Fig. 4.16. The orientation augmentation that produced the highest AP for each test set is marked in **bold**. The second highest AP is marked in *italics*.

4.16e, 4.16f, 4.16g, 4.16h and 4.16b. The corresponding average precision values can be seen in Tab. 4.12. In Fig. 4.16d, the synthesized images with a yaw similar to that of the *qr* test achieves higher recognition scores than frontal images in the *Original* subset. However, the average precision only increases 5‰-units above the original frontal face.

In nearly all test cases, the synthesized images get lower recognition scores than the frontal facing images in the *Original* subset. Since the synthesized images are based on frontal images, the accuracy of texture fit deteriorates the further from the frontal pose the model is oriented. However, at large poses (subsets *hl*, *hr*, *pl*, and *pr*), the recognition performance worsens. It is also worth noting that the second highest average precision score is oriented in the same direction as the test image in subsets (*hl*-*pr*) in Tab. 4.12. This would suggest that the orientation augmentation, while causing some artifacts that lowers the recognition score, might be able to be used to improve the recognition performance when used in combination with the original image. When one synthesized image per identity is added to the *Original* subset, nearly all testcases improve. The precision-recall curves can be seen in Fig. 4.17 and their corresponding

average precision can be seen in Tab. 4.13. The performance increase are quite small, but present. Looking at the performance on the subsets (*r-hr*, *pr*), the most performance is gained from augmentations that depict similar poses as in the specified test set.

However, it seems that full profile poses are too extreme to provide any improvement, and in many cases lowers the recognition score below the *Original* subset.

Pose\Subset	*f*	*r*	*ql*	*qr*	*hl*	*hr*	*pl*	*pr*
Original	0.9902	0.9966	0.9982	0.9976	0.9977	0.9944	*0.9195*	0.8250
Original + yaw 0.0	**0.9954**	0.9976	0.9986	**0.9988**	*0.9986*	0.9952	**0.9199**	0.8269
Original + yaw -0.4	*0.9914*	*0.9977*	**0.9990**	0.9986	**0.9991**	*0.9954*	0.9189	0.8312
Original + yaw 0.4	0.9914	**0.9978**	*0.9986*	*0.9986*	0.9983	**0.9958**	0.9190	*0.8312*
Original + yaw -1.2	0.9905	0.9966	0.9982	0.9976	0.9979	0.9943	0.9118	0.8250
Original + yaw 1.2	0.9905	0.9968	0.9982	0.9976	0.9979	0.9945	0.9000	**0.8336**
Original + yaw -1.5	0.9903	0.9955	0.9982	0.9976	0.9974	0.9929	0.8332	0.7273
Original + yaw 1.5	0.9902	0.9956	0.9982	0.9976	0.9974	0.9926	0.8436	0.7206
From Fig.	4.17a	4.17b	4.17c	4.17d	4.17e	4.17f	4.17g	4.17h

Table 4.13: Average precision for each subset and orientation augmentation, corresponding to Fig. 4.17. The orientation augmentation that produced the highest AP for each test set is marked in **bold**. The second highest AP score is marked in *italics*.

Adding more images to the identity-galleries of the evaluated poses increases the performance in the range of 1‰ to 1 ppm unit. However from Tab. 4.13 we can see that the best performing augmentations are not necessarily the synthetic images that are posed exactly like the test subsets. Therefore the same recognition test with the identity gallery configuration(the *Original* subset + one synthetic image per identity) was repeated over the entire range ($\pm\pi/2$) in increments of 0.1. The average precision for each test subset is represented as heatmaps found in Fig. 4.18. Due to the distribution of the AP, the color scale is non-linear and unique for each heat map.

Here we see that the best performing synthesized image is posed in the test subset direction, but always with a yaw angle offset towards the frontal pose. However, the angle offset of 0.2-0.5 radians, (11-30 degrees) is persistent in all test subsets. The largest offset occurs when testing against the *pl* and *pr* subsets, which could be caused by quality degradation of the model texture. This limit of 'usefulness' could stem from that reconstruction artifacts from the synthesized image become more prominent at larger pose angles. The smaller angle offsets occurs when testing against subsets *ql*, *hl*, *qr* and *hr*. Portions of these offsets could be attributed to pose estimation errors in the data collection process, such that the posed images in the affected testsets are actually closer to the highest performing synthetic poses. This claim might be plausible considering the pose estimation technique used in the collection of the dataset, mentioned in [27]. Another source of error could be the conversion from degrees, used in the FERET dataset, to radians used in our renderer, where for example the *hl* subset was posed at 22.5°, which amounts to about 0.393 radians, while the rendered pose was at 0.4 radians.

Test subset	Augmentation	AP	*Original* AP
f	*Original*+ yaw(0.0)	0.9954	0.9902
r	*Original*+ yaw(0.2)	0.9978	0.9966
ql	*Original*+ yaw(0.2)	0.9994	0.9982
hl	*Original*+ yaw(-0.7)	0.9991	0.9977
pl	*Original*+ yaw(-0.8)	0.9265	0.9195
qr	*Original*+ yaw(-0.1)	0.9990	0.9982
hr	*Original*+ yaw(1.0)	0.9959	0.9960
pr	*Original*+ yaw(0.9)	0.8437	0.8268

Table 4.14: The best single synthesized image augmentation using the SFM on each test subset. The choice of augmentations are based of the heatmaps in Fig. 4.18

Worth noting is also that local performance maximum appear for the mirrored synthetic image poses, i.e. poses with negative yaw for the positive *qr,hr,pr*, and the other way around. This suggests that even though the added pose is in the "wrong" yaw-direction, the added information to identity gallery can still be useful. The best performing augmentation from each test subset is summarized in Tab. 4.14.

4.2.2.2 Augmentation Using the MFM

Changing to the MFM from the work [22] and using the same subsets as in the prior section, produces the results seen in Fig. 4.19. Average precision can be seen in Tab. 4.15. Here the synthetic images with positive yaw are mirrored images with the corresponding negative yaw. For Figs. 4.19a-4.19h, the performance of the synthetic images are comparable to those in the *Original* subset. The main trend with these augmentations is similar to those of the SFM augmentations, in that orientation augmentations at smaller angles perform better than their more extreme angle counterparts. However, the frontal augmentation, both mirrored and non-mirrored, outperforms the *Original* subset in Figs. 4.19a-4.19d as well as 4.19b. The largest yaw angles, +1.3 and -1.3 radians respectively, produce the lowest AP scores.

Pose\Subset	*f*	*r*	*ql*	*qr*	*hl*	*hr*	*pl*	*pr*
Original	0.9902	0.9967	0.9982	0.9976	**0.9977**	**0.9944**	**0.9195**	**0.8268**
yaw -0.0	*0.9948*	*0.9975*	0.9980	**0.9996**	0.9963	*0.9935*	0.8832	0.7913
yaw +0.0	0.9901	**0.9976**	*0.9985*	*0.9996*	*0.9966*	0.9917	0.8883	0.7945
yaw -0.7	**0.9956**	0.9963	**0.9990**	0.9976	0.9919	0.9839	*0.9058*	*0.8152*
yaw +0.7	0.9948	0.9958	0.9983	0.9980	0.9920	0.9829	0.8997	0.7937
yaw -1.3	0.9599	0.9652	0.9800	0.9701	0.9644	0.9235	0.8127	0.7015
yaw +1.3	0.9356	0.9490	0.9687	0.9523	0.9431	0.9431	0.7574	0.6764
From Fig.	4.19a	4.19b	4.19c	4.19d	4.19e	4.19f	4.19g	4.19h

Table 4.15: Average precision for each subset and orientation augmentation, using the MFM. The orientation augmentation that produced the highest AP for each test set is marked in **bold**. The second highest AP is marked in *italics*.

Adding one augmentation to the identity gallery and then testing against each test subset produces the plots seen in Fig. 4.20. The performance increases in Figs. 4.20b-4.20h, where the greatest performance increase can be seen in the *pr* subset in Tab. 4.16. In this subset the performance increased by nearly 1%-unit, thereby suggesting that the augmentation procedure is helpful.

Pose\Subset	*f*	*r*	*ql*	*qr*	*hl*	*hr*	*pl*	*pr*
Original	0.9902	0.9966	0.9982	0.9976	0.9977	0.994	0.9195	0.8268
Original +yaw -0.0	0.9951	*0.9977*	**0.9996**	*0.9990*	*0.9990*	*0.9958*	0.9235	*0.8331*
Original +yaw +0.0	0.9956	0.9977	*0.9996*	**0.9998**	0.9990	**0.9960**	*0.9237*	0.8299
Original +yaw -0.7	0.9917	0.9977	0.9992	0.9986	0.9987	0.9958	**0.9251**	0.8322
Original +yaw +0.7	**0.9980**	**0.9980**	0.9992	0.9986	**0.9992**	0.9956	0.9228	**0.8332**
Original +yaw -1.3	*0.9980*	0.9969	0.9982	0.9976	0.9982	0.9945	0.9094	0.8172
Original +yaw +1.3	0.9908	0.9970	0.9982	0.9976	0.9977	0.9941	0.8918	0.7965
From Fig.	4.20a	4.20b	4.20c	4.20d	4.20e	4.20f	4.20g	4.20h

Table 4.16: Average precision for each subset and orientation augmentation, using the MFM. The orientation augmentation that produced the highest AP for each test set is marked in **bold**. The second highest AP is marked in *italics*.

Judging by Tab. 4.16, the best augmentations seem to be the mirrored frontal synthesized image (+yaw 0.0) and the positive yaw angle of 0.7 (+yaw +0.7). Using the two top performing synthesized images from each subset form Tab.4.16 together with the *Original* subset increased the performance slightly for some test subsets, but a diminishing returns effect seems to occur, I.e. each added synthetic image affect the result to a lesser and lesser degree. The average precision can be seen in Tab. 4.17. In some of the tests, the possible improvement is beyond the sixth digit of accuracy, meaning that the contribution of the augmentations are minuscule. However, in the extremely posed images, the AP is noteworthy higher. The performance on the *pr* subset can be slightly improved upon up to an AP of 0.8391, by adding the orientation augmentation of -0.7 radians as well.

Pose \Subset	*f*	*r*	*ql*	*qr*	*hl*	*hr*	*pl*	*pr*
Original	0.9902	0.9966	0.9982	0.9976	0.9977	0.994	0.9195	0.8268
Original +yaw (+0.0)	0.9956	0.9977	**0.9996**	**0.9998**	0.9990	*0.9960*	*0.9237*	0.8299
Original +yaw (+0.7)	0.9915	*0.9980*	*0.9992*	*0.9986*	*0.9992*	0.9956	0.9228	*0.8332*
Original +yaw (+0.0, +0.7)	**0.9958**	**0.9980**	**0.9996**	**0.9998**	**0.9993**	**0.9961**	**0.9252**	**0.8361**

Table 4.17: Average precision for each subset and the two top performing orientation augmentations, using the MFM. The orientation augmentation that produced the highest AP for each test set is marked in **bold**. The second highest AP is marked in *italics*.

As shown in Tab. 4.13 and 4.16, most of the performance gains are found at more

extreme poses. However, when using synthesized images at extreme poses, (yaw: ± 1.2 up to ± 1.5), the performance generally worsens. The cause of this might be twofold. Firstly, the basis for the synthesized images are frontal images. These were selected for the synthesis basis as the most accurate landmarks can be found from the frontal pose. This also means that the reconstructed textures are of the highest quality in the frontal parts of the face as well, and degrading towards the edges of the 3D-model, where stretching of the fitted image occurs. Secondly, the mesh face model used ends at the ears and just below the jawline of a human. When the model is posed at the most extreme yaw-angles, the 'edge' of the face is visible in the bounding box of the face, thereby causing sharp and unnatural image-distortions.

Changing the background color of the synthesized MFM images to white and doing the same test as in Tab. 4.15, i.e. testing only synthesized images against test subsets, reduced the performance slightly on test subsets *ql* and *qr*. Adding the synthesized images to the *Original* and comparing against the performance results found in Tab. 4.16, the difference in performance was lessened. Although some parts of background are present in the face crop, these do not seem to affect the performance of the MFM that much.

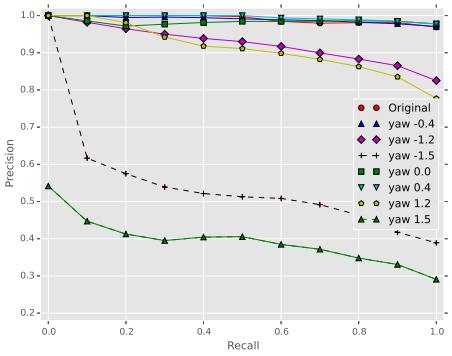
Performing a similar experiment on the SFM, i.e. changing the background color from white to black did not affect the results significantly in the majority of the cases. Using the random background approach from section 3.5.1 the performance was slightly improved for the *hr* subset and the *pr* subset. In the case of the *pr* subset, the AP-score was increased by 2% units when compared to the *Original* subset. In order to more closely observe the different augmentation schemes on the *pr* test subset, CMC-curves were calculated for these testcases. The results can be seen in Fig. 4.21.

From Fig. 4.21 we see that the top performing augmentation switches between the SFM with random background and the SFM with white background. Both datasets are constructed by adding one synthetic image to each identity. This oscillatory behavior of top performing would suggest that random or white background does not necessarily increase or decrease accuracy significantly. However, choosing a background color other than black seems to improve results drastically, most noticeable in Fig. 4.21d. One reason for this behavior could be that the edges of the model are more prominent with the black background, which could cause unnatural responses when feed into the classification network.

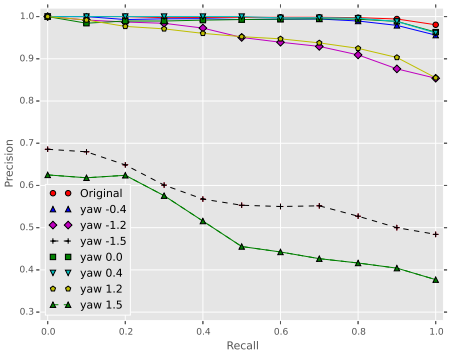
In Tab. 4.18, the best performing SFM augmentations are compared against the best MFM augmentations. There are only very small differences present in the performance, but in the majority of the testcases the MFM seems to slightly outperform the SFM. However, in the *pl* and *pr* the *SFM* outperforms the *MFM* slightly. The reason behind this is that a better pose-fit exists beyond the 75°, (~ 0.7 radians), posed *MFM* model, which we are able to find using the *SFM* model. In conclusion of the tests carried out on the FERET datasets, the *MFM* seem to produce more robust synthesized images that are more likely to be of high quality while the *SFM* provides the means of a more flexible synthesis scheme, allowing for more precise pose augmentation.

Augmentation\subset	*f*	*ql*	*hl*	*pl*
Best MFM (White BG)	(-0.0) **0.9961**	(+0.0) 0.9994	(+0.7) **0.9993**	(+0.7) 0.9240
Best MFM (Black BG)	(+0.0) 0.9956	(+0.0) **0.9996**	(+0.7) 0.9992	(-0.7) 0.9251
Best SFM (White BG)	(+0.0) 0.9954	(+0.2) 0.9994	(-0.7) 0.9991	(-0.8) **0.9265**
Best SFM (Black BG)	(+0.0) 0.9933	(+0.2) 0.9994	(-1.0) 0.9992	(-0.8) 0.9229
Best SFM (Random BG)	(+0.4) 0.9933	(0.2) 0.9992	(-0.8) 0.9992	(-0.9) 0.9245
Original	0.9902	0.9990	0.9977	0.9195
Augmentation\subset	*r*	*qr*	*hr*	*pr*
Best MFM (White BG)	(-0.0) 0.9978	(+0.0) 0.9203	(-0.0) 0.9956	(+0.0) 0.8316
Best MFM (Black BG)	(+0.7) **0.9980**	(+0.0) **0.9998**	(+0.0) 0.9960	(+0.7) 0.8332
Best SFM (White BG)	(+0.2) 0.9978	(-0.1) 0.9990	(+1.0) 0.9959	(+0.9) 0.8437
Best SFM (Black BG)	(+0.5) 0.9979	(+0.2) 0.9990	(+0.1) 0.9959	(-0.8) 0.8310
Best SFM (Random BG)	(+0.5) 0.9979	(+0.2) 0.9992	(+0.7) **0.9960**	(+1.0) **0.8468**
Original	0.9966	0.9976	0.9944	0.8268

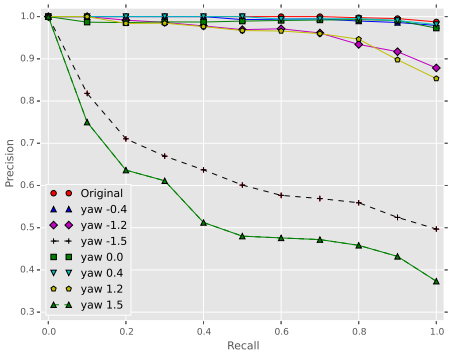
Table 4.18: The best performing augmentations on each subset compared to each other. In each case, the synthesized image has been added to the identity gallery. The yaw for each synthesized image is expressed in radians and in parenthesis next to the AP. The top performing augmentation for the subset is marked in **bold**.



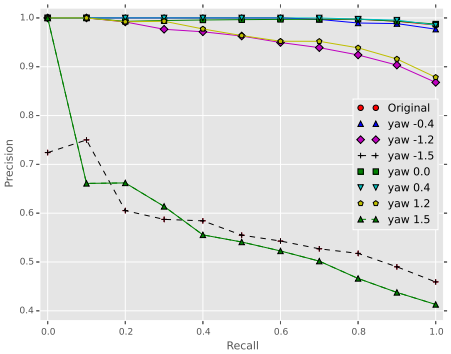
(a) Precision-recall curve, f subset



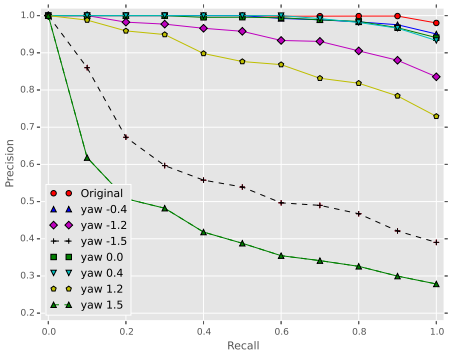
(b) Precision-recall curve, r subset.



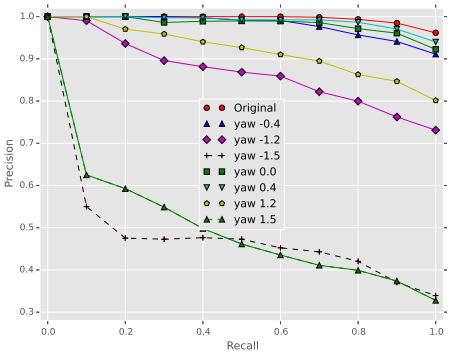
(c) Precision-recall curve, ql subset.



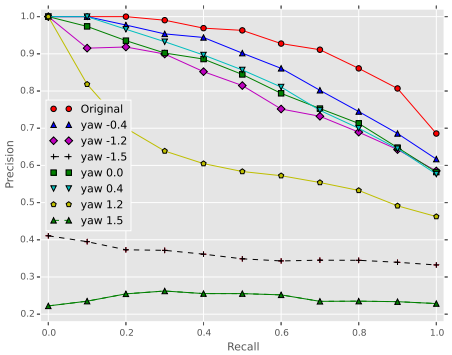
(d) Precision-recall curve, qr subset



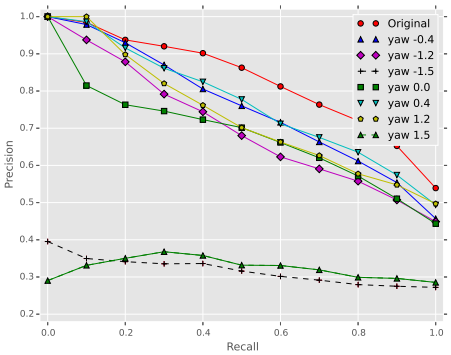
(e) Precision-recall curve, hl subset



(f) Precision-recall curve, hr subset

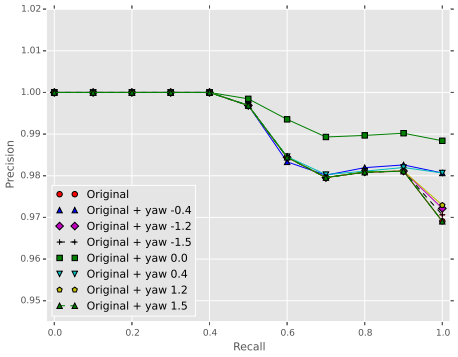


(g) Precision-recall curve, pl subset.

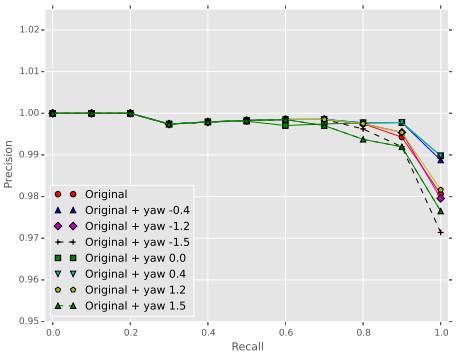


(h) Precision-recall curve, pr subset.

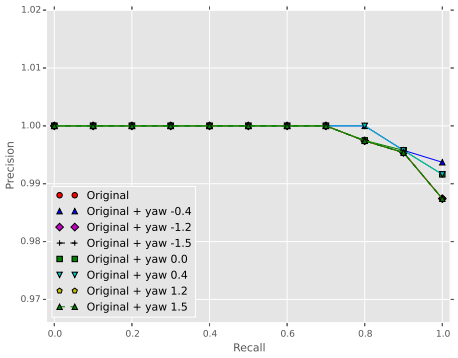
Figure 4.16: Precision-recall curves for each test subset in the FERET dataset, using only synthetic images or only real images for recognition.



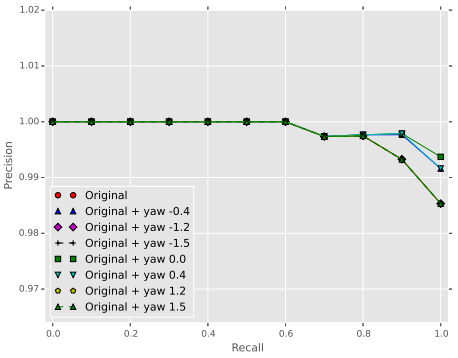
(a) Precision-recall curve, f subset



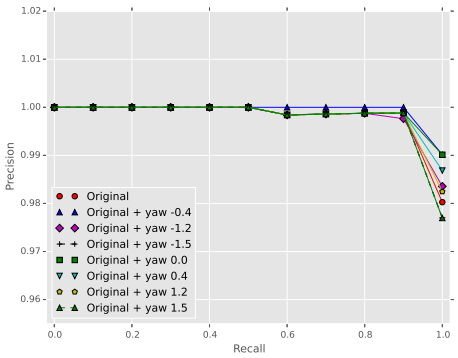
(b) Precision-recall curve, r subset.



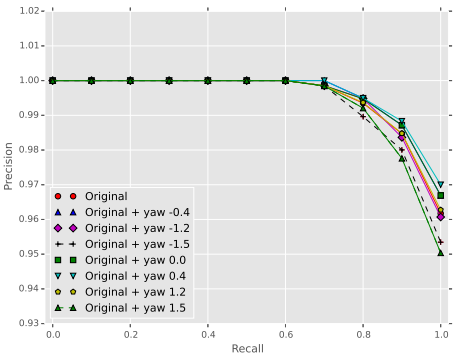
(c) Precision-recall curve, ql subset.



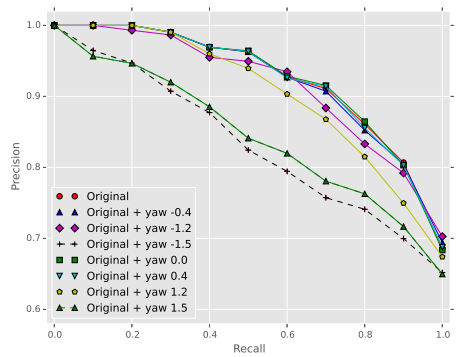
(d) Precision-recall curve, qr subset



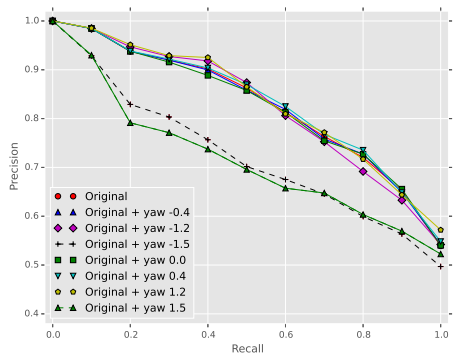
(e) Precision-recall curve, hl subset



(f) Precision-recall curve, hr subset



(g) Precision-recall curve, pl subset.



(h) Precision-recall curve, pr subset.

Figure 4.17: Precision-recall curves for each test subset in the FERET dataset, using the *original* subset and one synthesized image per identity in order to classify each test set.

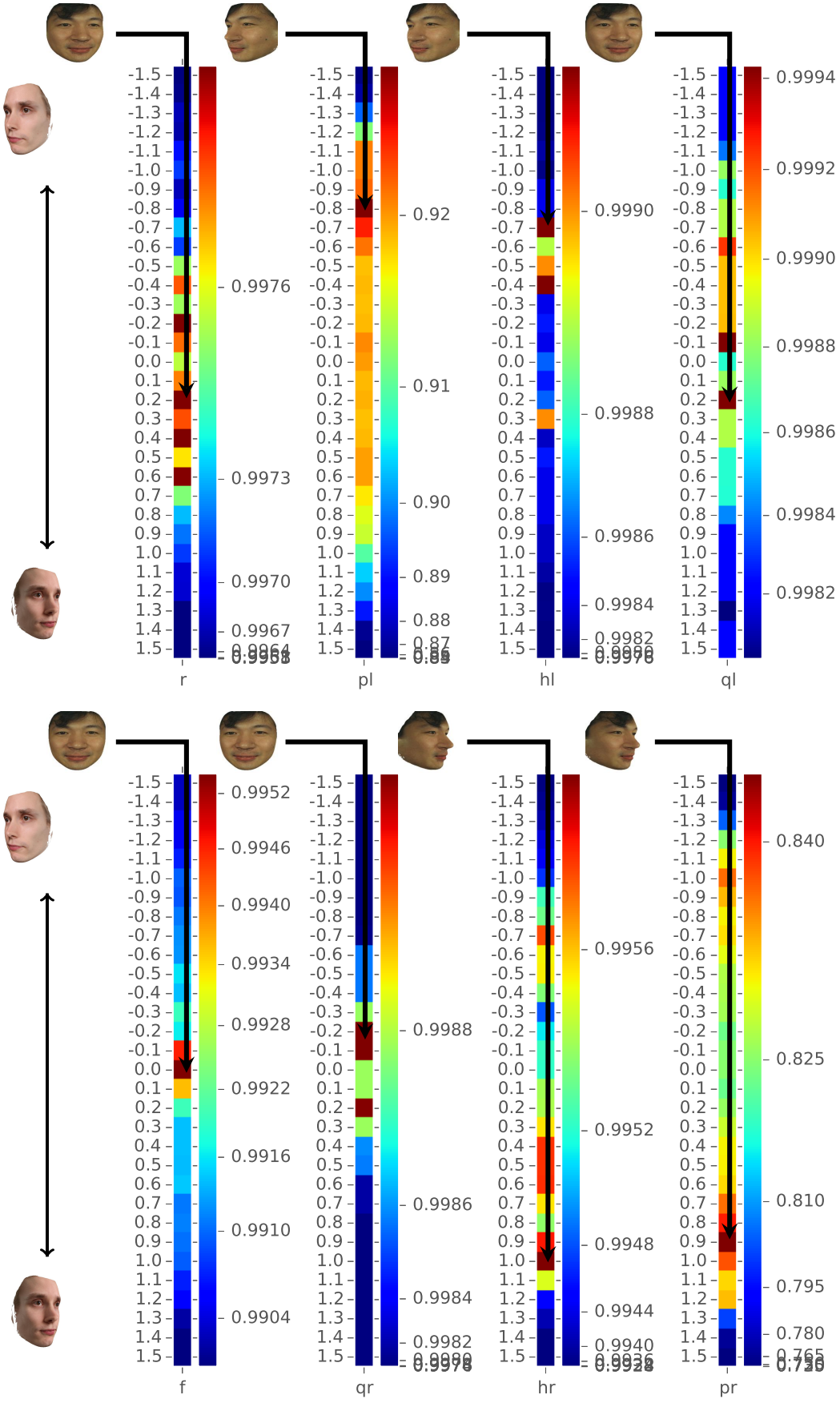
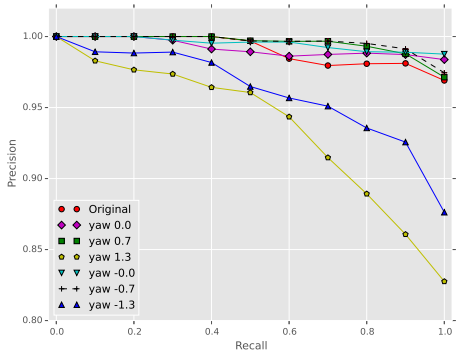
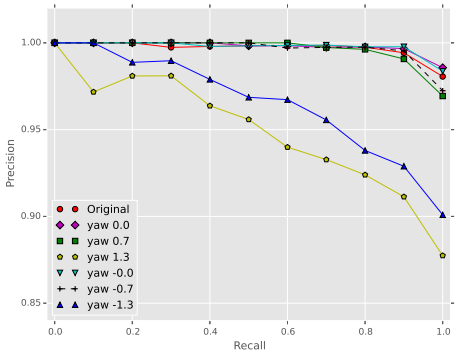


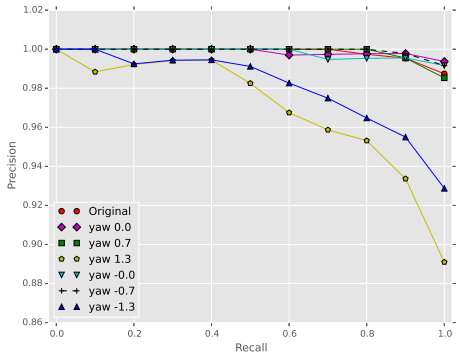
Figure 4.18: Heatmap plots over AP for different poses of the synthesized images for each test subset. Note that the color-mapping is unique for each plot and non-linear.



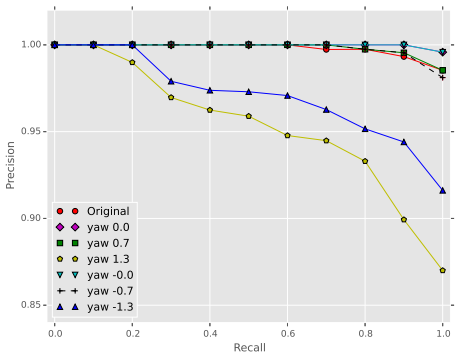
(a) Precision-recall curve, f subset



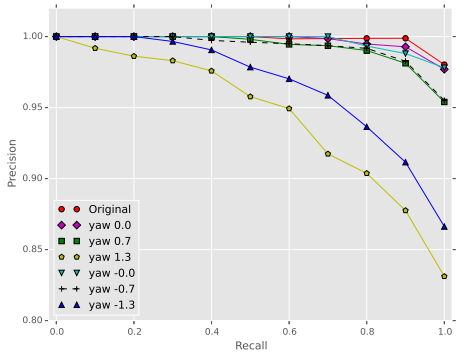
(b) Precision-recall curve, r subset.



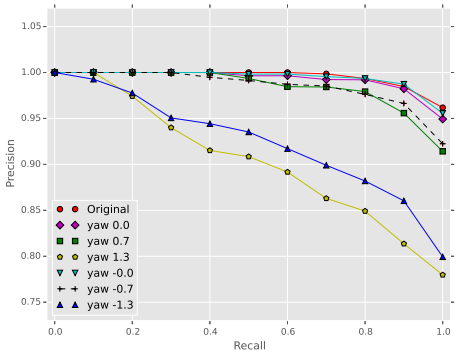
(c) Precision-recall curve, ql subset.



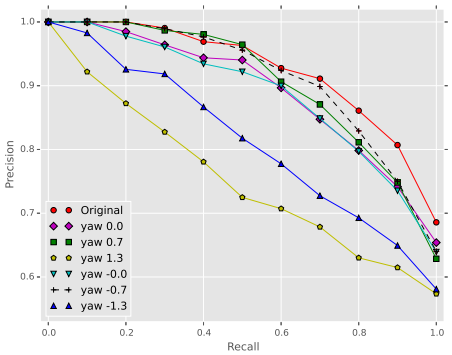
(d) Precision-recall curve, qr subset



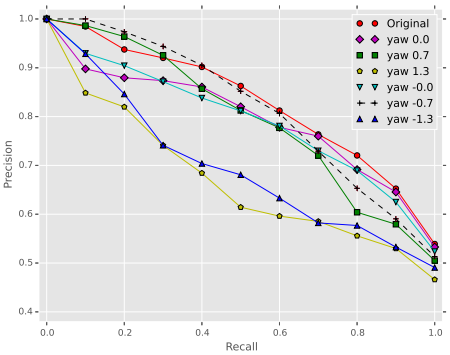
(e) Precision-recall curve, hl subset



(f) Precision-recall curve, hr subset

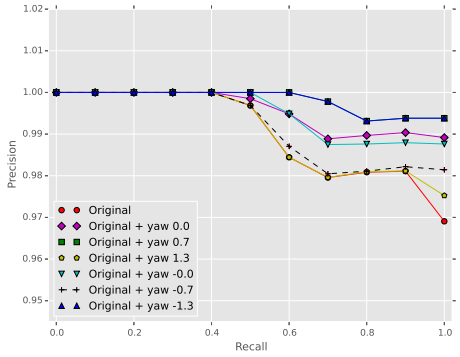


(g) Precision-recall curve, pl subset.

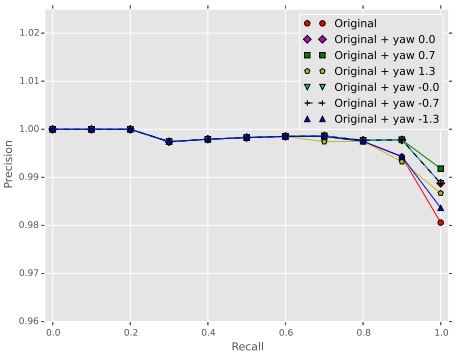


(h) Precision-recall curve, pr subset.

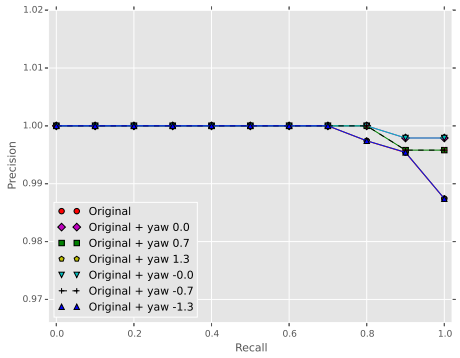
Figure 4.19: Precision-recall curves for each test subset in the FERET dataset, using the only one synthesized image per identity, using the method in [22] , in order to classify each test set.



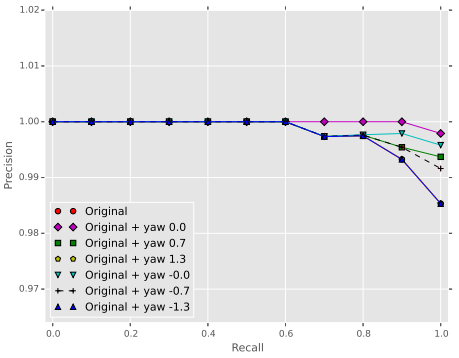
(a) Precision-recall curve, f subset



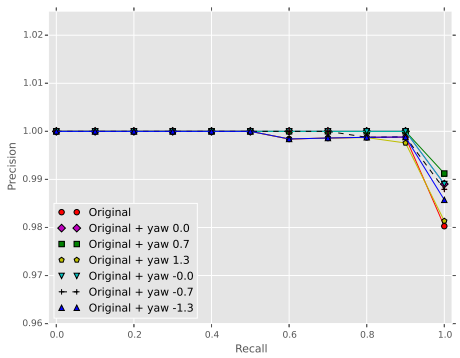
(b) Precision-recall curve, r subset.



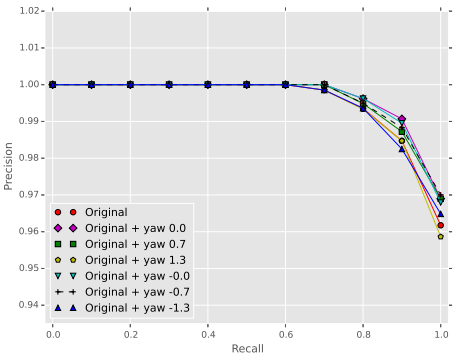
(c) Precision-recall curve, ql subset.



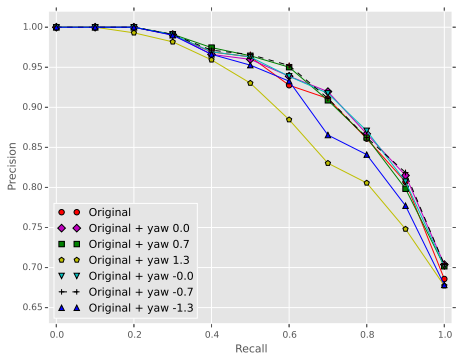
(d) Precision-recall curve, qr subset



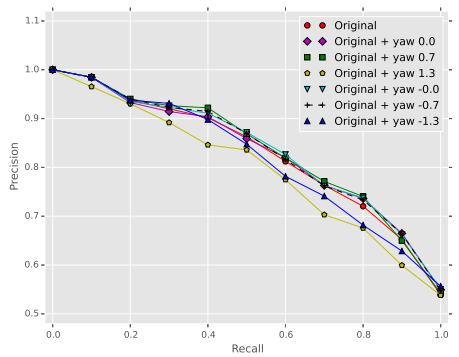
(e) Precision-recall curve, hl subset



(f) Precision-recall curve, hr subset

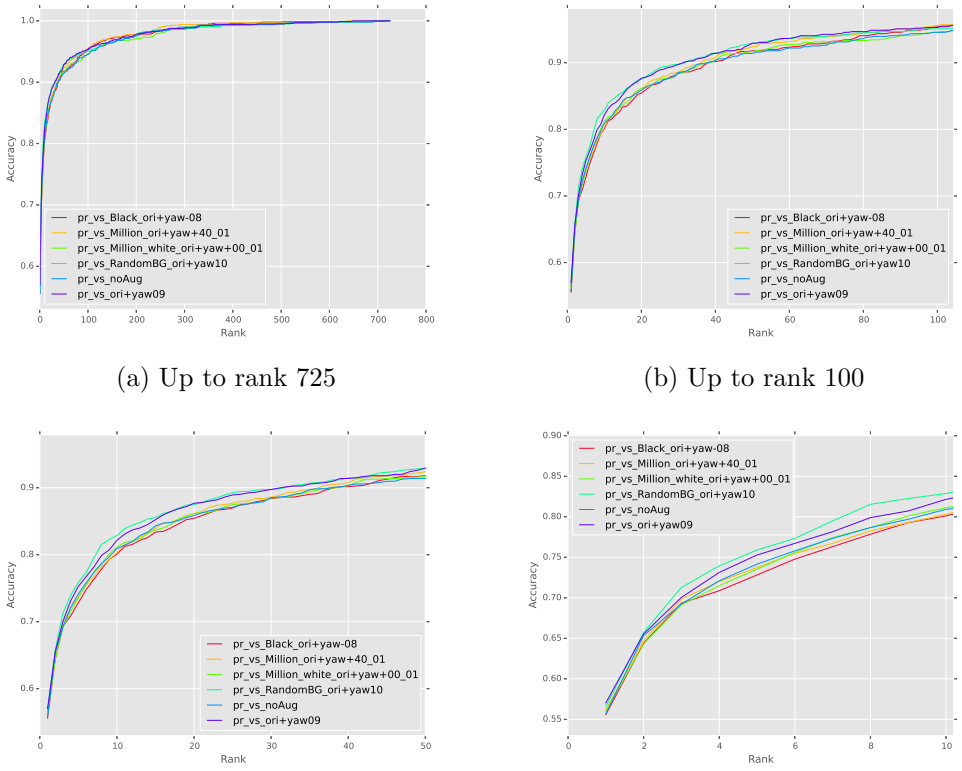


(g) Precision-recall curve, pl subset.



(h) Precision-recall curve, pr subset.

Figure 4.20: Precision-recall curves for each test subset in the FERET dataset, using the *original* subset and one synthesized image per identity, using the method in [22] , in order to classify each test set.



(a) Up to rank 725

(b) Up to rank 100

(c) Up to rank 50

(d) Up to rank 10

Figure 4.21: The CMC curves for the best performing augmentation strategies for the SFM. Each sub-figure displays the same CMC curves but is gradually zoomed at lower ranks to show more details. The evaluation is performed on the *pr* subset of the FERET dataset

4.3 Augmentation

The results of the various augmentation schemes are presented here. Using the MFM described in [22] augmentations at fixed angles could be extracted, the number of angles depending on the face orientation of the initial image. Examples can be seen in Fig. 4.22. Each identity can be rendered at fixed yaw angles.

(a) Input image (b) Frontal (c) 40° (d) 75°

Figure 4.22: Synthesized images using the MFM. The background of the input image is usually distorted, so a solid color is used as background instead.

For identities with a "neutral" facial expression, the augmentations produce images that are similar to the original image, preserving important facial details. However, usage of images with more varied facial expressions produces somewhat erroneous renders. An example can be seen in Fig. 4.23.

(a) Input image (b) Frontal (c) 75° yaw

Figure 4.23: Augmentation with open mouth in original image.

Looking more closely at the mouth in Fig. 4.23c, the teeth of the mouth are actually mapped to the upper lip of the model. This causes a slightly erroneous render, which in some instances could make the recognition harder. If more extreme facial expressions are present, the quality of the render further deteriorates.

The problem of teeth is addressed in the SFM, where the *blendshapes* are employed to attempt to map the input image to the face model more precisely, by finding a matching facial expression to the input image. Since this fitting scheme outputs a full 3D-model with an accompanied texture, more poses can be employed, as well as control of facial expressions, within the confines of the blendshapes. This allows for many possible augmentations by adjusting the rotation matrix for both *yaw*- and *pitch* rotations of the face model. A sample of the possible augmentations can be seen in Fig. 4.24.

The proposed multi texturing technique in section 3.5.2 was found to produce more erroneous results than simply rendering the model with a single texture. Fitted textures captured at similar angles tended to contain slight miss-alignments of key facial features, such as eyes and nostrils, so that many facial features tended to get lost when applying the technique. The same artifacts also occurred when more than ~ 5 textures, captured at evenly distributed pose angles, were used. Differences in lighting conditions as well as various image qualities further introduced distortions that deteriorated the quality of the face. Therefore the multi-texturing technique was not investigated further and all synthesized faces used only one texture per identity, constructed from a single frontal

image. The technique might be utilized if a more robust shape-fitting procedure is employed.

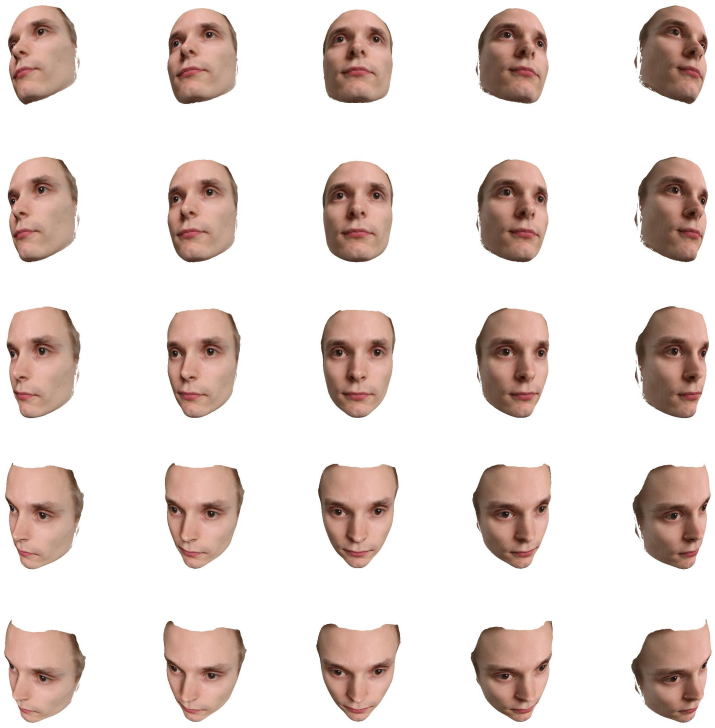


Figure 4.24: A sample of possible synthesized poses using the SFM. Here the "neutral" facial expression is used.

4.4 Synthetic Images Using BEGAN

Using the trained BEGAN network, synthetic images of faces can be generated. Fig. 4.25 shows 25 face images that were generated from random sample vectors.

4.4.1 Generated Images for Face Verification

Fig. 4.26 shows examples of successful and unsuccessful images created for face verification according to section 3.6.2.1. As we are sampling input vectors that are supposed to belong to the same identity from a multivariate normal distribution with a small variance, it is likely that many images will have qualities that make them look similar. As can be seen from Fig. 4.26a most of the images look like they could belong to the same identity. There is also some variation between the images which is desired as long as the images still seem to belong to the same identity.

In Fig. 4.26b the images seem to have low quality. Most of them resemble face images but they are blurry, lacking details and have strange artifacts such as blobs of color appearing randomly over the image. This can occur since the images are based on random generated input vectors. Not all input vectors generate realistic looking images and without manual intervention this is difficult to detect.

For the images in Fig 4.26c there seem to be too much variation within the identity. This is one issue with simply sampling from a multivariate normal distribution. If the

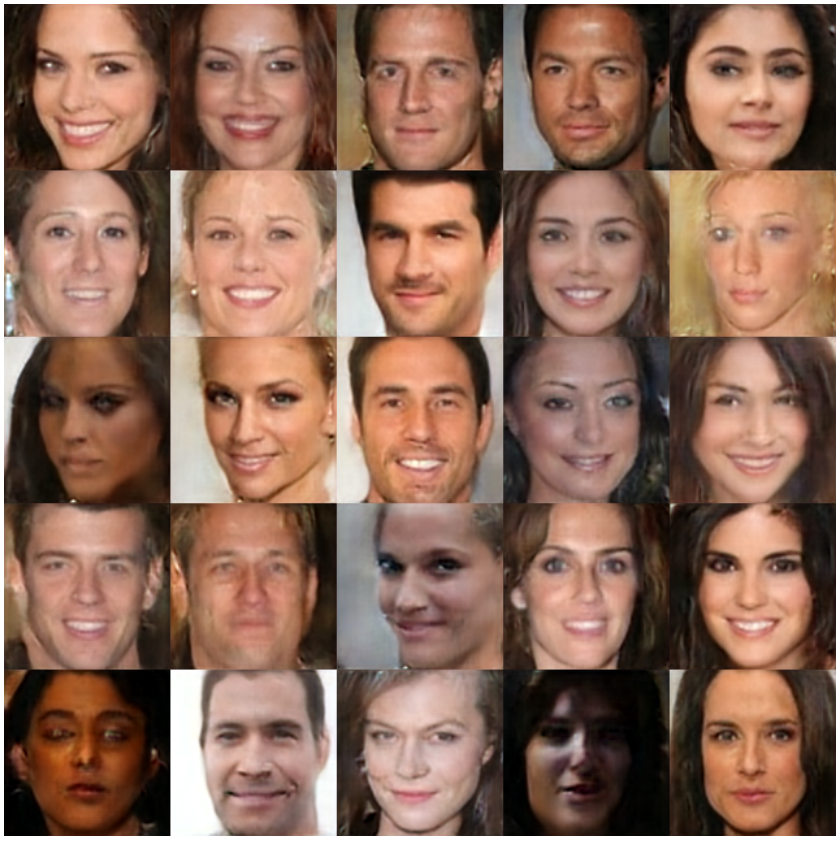


Figure 4.25: Synthetic images generated using the BEGAN network.

variance is too large then images belonging to different identities will appear, if the variance is too small the images will become too similar. Ideally one would like to have large variations within the same identity such as rotations of the head, different facial expressions, different lighting conditions, etc. Using this method this is difficult to achieve.

4.4.2 Generated Images for Face Recognition

The results of the optimization experiment described in section 3.6.2.2 are shown in Fig. 4.27.

As can be seen from Fig. 4.27 the generated images tend to converge towards something resembling each original image. However, the finer details in the original images cannot be generated and therefore the identities in the generated images seem to be different than the identities in the original images. The quality of the generated images is also significantly worse than the original images even though they are both 128×128 pixels large. As the image do not resemble the correct identity these are not used as training data for face recognition.

As the optimization does not constrain the sample vector $\mathbf{z}$ to $[-1, 1]^{64}$, it is possible that the optimal vector found is not a valid vector. Input vectors $\mathbf{z}$ that are not withing the correct range can still be used as input to the network. If this is done the generated images often start to loose many of the qualities that otherwise make them look like face images, such as the eyes, the nose or the mouth.

Even if the optimal input vector is valid, the generator might simply not be able to recreate some of the features of the original images. As an example, non of the optimal

(a) Successful

(b) Low image quality

(c) Too much variation

Figure 4.26: Successful and unsuccessful image generation using BEGAN. Each subfigure shows images belonging to the same identity.

images shown in Fig. 4.27 have glasses while most of the original ones do. Most of the generated images are also frontal, looking into the "camera". The generator can only learn features that exist in the training dataset. If most of the training images are frontal without glasses then that is what the generator will learn. To achieve a more general generator, a more diverse training dataset is needed.

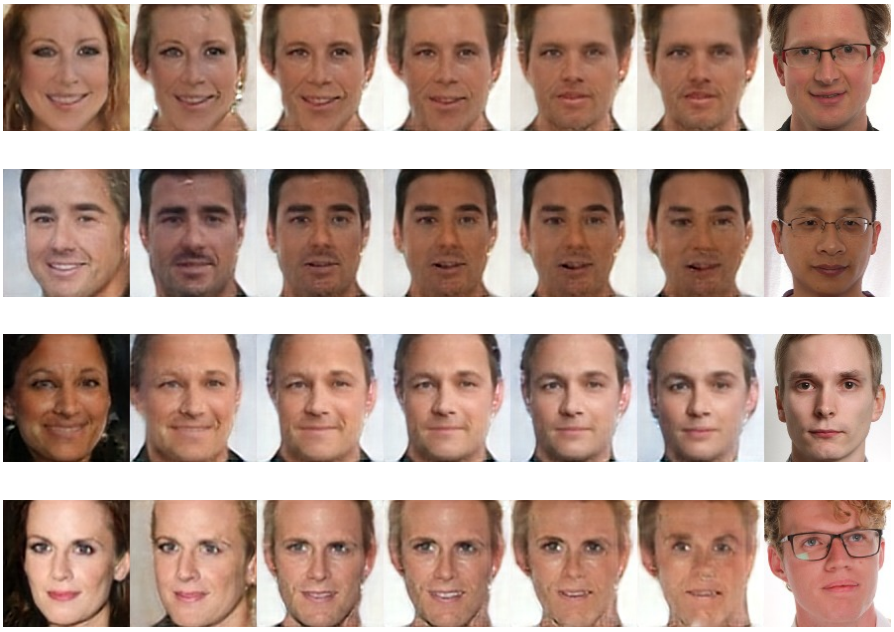


Figure 4.27: The leftmost image shows the generated image $G(\mathbf{z})$ for the initial sample vectors $\mathbf{z}$. The rightmost image shows the original image $\mathbf{x}$. The images the middle show the progress of the optimization.

4.5 Face Recognition Prototype

We run the face recognition system in real time using an Axis P1357-E1 camera. As face detection is a part of performing face recognition, we use the Multi-task Cascaded Convolutional Network, mentioned in section 2.3.1, for face detection. The pre-trained Inception-ResNet-V1 network is used to actually recognize the identities of the detected face images. Both the face detection and the recognition are performed on one Nvidia GTX 1060 with 6 GB of GDDR5 RAM. Tab. 4.19 shows the average time it takes to detect and recognize one face using this setup for different resolutions. Tab. 4.20 shows how the recognition time is affected by using a larger training dataset.

	640×360	800×450	1280×720	1920×1080
Detection	30.08 ms	40.31 ms	81.92 ms	163.76 ms

Table 4.19: Each column represents the time it takes to perform face detection at different camera resolutions.

	298 images	2980 images	29800 images
Recognition	22.81 ms	26.91 ms	29.36 ms

Table 4.20: Each column represents the time it takes to perform face recognition at different training dataset sizes.

Face detection seems to consume more time than face recognition. Unlike face recognition which is always performed on images of size 160×160 pixels, the face detection slows down as the camera resolution increases. This can be seen in Tab. 4.19. The detection time increases as the P-Net, mentioned in section 2.3.1, is forced to handle

images of larger sizes.

As the size of the training dataset increases there are more images that need to be compared in order to perform face recognition. This increases the time needed to find the best match as is illustrated in Tab. 4.20. The Axis employee database consists of 298 training images and reaches up to 894 training images when using the best performing SFM augmentation strategy. This hardly affects the recognition time compared to the detection time. Even when using $29,800$ training images, the time it takes to recognize one face image is smaller than the time it takes to detect the face. This is true even at the smallest resolution of 640×360 pixels.

At resolution 640×360 pixels when using only the 298 training images from the Axis employee database, the total time needed to detect and recognize a face is on average 52.89 ms. This is equivalent to approximately 19 frames per second, which is enough to make the video stream seem smooth.

Chapter 5

Conclusion

In this thesis work we explored different data augmentation techniques. In particular we used 3D morphable models, the SFM and the MFM, in an attempt to enhance face verification and recognition. We also explored how generative adversarial networks could be used to create synthetic datasets for training face verification networks, and if these synthetic images potentially could be used for face recognition as well. We also constructed an evaluation dataset from surveillance images for internal use at Axis.

Using synthetic images in conjunction with the original image, we were able increase the recognition performance, as can be seen in Fig. 4.8. However, the possible performance increase differs greatly between the different synthetic images, where inclusion of low-quality synthetic images actually decrease performance significantly. This can be seen in Tab. 4.5, where performance starts to decrease after using more than 3 synthetic images, as well as in Tab. 4.7.

5.1 The Surrey Face Model

The SFM proved to be useful when performing face recognition. It does require a high resolution image for the 3D reconstruction but can therefore also create high quality synthetic images if the reconstruction is successful. If the pose of the faces in the input images to the face recognition system is known, this method could improve the performance. This is a likely scenario to occur in video surveillance as cameras are often capturing images from a fixed direction. When tested on the FERET dataset at specific pose angles, it was seen that the performance was increased when using augmentations with similarly posed synthetic images as in the test set. A subtle performance increase could also be found if the mirrored synthetic pose image was used instead, i.e. comparing a synthetic image at a negative yaw angle with a test image at a positive yaw.

For face verification the SFM was not quite as useful as the other augmentation methods. As the reconstruction can fail fairly easily for images that are of lower resolution, not sharp enough or where the face is partially occluded the method is not ideal for augmenting large diverse datasets.

5.2 The Million Faces Model

The MFM was useful when training face verification networks. It was both faster and could handle lower resolution images better than the SFM which made it ideal for large data augmentation. It provided synthetic images where the visual quality was decent,

although not quite as high as the best performing synthetic images using SFM. Instead it was stable and often produced usable synthetic images.

For face recognition the MFM was not quite as useful as it can only be rendered at a few different yaw angles. This makes it less ideal for the scenario where a surveillance camera is capturing images at a fixed angle. However, when comparing the synthetic MFM images to similarly posed test images in the FERET set, the MFM achieved slightly higher performance than the corresponding SFM synthetic images. This might be caused by the dense texture mapping applied within the MFM fitting scheme, which causes less distortions near the edges of the face.

5.3 Generative Adversarial Networks

The BEGAN used in this thesis work proved to be very good at creating realistic looking face images. However, for face verification or recognition this is not enough as the training data must be labeled. We attempted to use the BEGAN to generate several synthetic images belonging to the same identity, while still being somewhat different. Our technique was capable of generating images that seemingly belonged to the same identity, but in order to generate more versatile images within an identity a different strategy would be needed. Still, we were capable of fine-tuning a face verification network using completely synthetic data and performing no manual data labeling.

We also tried to generate data for face recognition where we wanted more images of an already known identity. The method used was capable of generating images resembling the known identity but they were not good enough to actually seem to belong to the same person.

The main benefit of the generative adversarial networks was that they generated images of complete faces, including hair and ears, unlike the 3D morphable models that only reconstructed parts of the face. The generated faces were also often more realistic looking than the ones reconstructed using the 3D morphable models. This gives them great potential.

5.4 Future Work

There are a variety of different methods that could be explored in order to improve the current performance. There are recent techniques for improved facial textures for 3D morphable models, such as [32], where a full high quality texture is generated even for occluded regions of the face. Improved landmark detection could also be useful to increase the quality of the reconstruction, and could make the augmentation process more robust. Creating completely new morphable models that covers a larger region of the face would be useful, although difficult and time consuming.

Generative adversarial networks show a lot of promise as they are capable of generating realistic looking faces compared to the slightly artificial look of the morphable models. A new strategy for dataset generation with more variation within identities, compared to our method, would be useful. There are also other interesting works such as [16] that use GANs to generate faces at different yaw angles given a profile input image.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Oswald Aldrian and William AP Smith. Inverse rendering of faces with a 3d morphable model. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1080–1093, 2013.

[3] Anh Tu an Trãn, Tal Hassner, Iacopo Masi, and Gérard Medioni. Regressing robust and discriminative 3D morphable models with a very deep neural network. *arXiv preprint arXiv:1612.04904*, 2016.

[4] D. Berthelot, T. Schumm, and L. Metz. BEGAN: Boundary Equilibrium Generative Adversarial Networks. *ArXiv e-prints*, March 2017.

[5] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. June 2014.

[9] Patrick Grother, Ross J. Micheals, and P. Jonathon Phillips. Face recognition vendor test 2002 performance metrics. In *Proceedings of the 4th International Conference on Audio- and Video-based Biometric Person Authentication*, AVBPA'03, pages 937–945, Berlin, Heidelberg, 2003. Springer-Verlag.

[10] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. *CoRR*, abs/1607.08221, 2016.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[12] T. Hassner, S. Harel, E. Paz, and R. Enbar. Effective face frontalization in unconstrained images. *ArXiv e-prints*, November 2014.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[14] Guosheng Hu. *Face analysis using 3D morphable models*. PhD thesis, University of Surrey, 2015.

[15] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[16] R. Huang, S. Zhang, T. Li, and R. He. Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis. *ArXiv e-prints*, April 2017.

[17] Patrik Huber, Guosheng Hu, Rafael Tena, Pouria Mortazavian, Willem P Koppen, William Christmas, Matthias Rätsch, and Josef Kittler. A multiresolution 3d morphable face model and fitting framework. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 1–8, 2016.

[18] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 2014.

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

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[22] Iacopo Masi, Anh Tu an Trãn, Tal Hassner, Jatuporn Toy Leksut, and Gérard Medioni. Do we really need to collect millions of faces for effective face recognition? In *European Conference on Computer Vision (ECCV)*, October 2016.

[23] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 343–347. IEEE, 2014.

[24] Mortazavian P. *Face Recognition in Low Resolution Using a 3D Morphable Model*. PhD thesis, University of Surrey, 2013.

[25] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301, Sept 2009.

[26] P. J. Phillips, Hyeonjoon Moon, S. A. Rizvi, and P. J. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104, Oct 2000.

[27] P Jonathon Phillips, Sandor Z Der, Patrick J Rauss, and Or Z Der. *FERET (face recognition technology) recognition algorithm development and test results.* Army Research Laboratory Adelphi, MD, 1996.

[28] P.Jonathon Phillips, Harry Wechsler, Jeffery Huang, and Patrick J. Rauss. The feret database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295 – 306, 1998.

[29] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: Database and results. *Image and Vision Computing*, 47:3–18, 2016.

[30] Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, June 2013.

[31] Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. A semi-automatic methodology for facial landmark annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 896–903, 2013.

[32] Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. Photorealistic facial texture inference using deep neural networks. *CoRR*, abs/1612.00523, 2016.

[33] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, pages 92–101, Berlin, Heidelberg, 2010. Springer-Verlag.

[34] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[37] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[38] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. pages 499–515, 2016.

[39] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011.

[40] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.