technische universität
dortmund

Deep Feature Interpolation for Image Content Changes

Master Thesis

Rahul Kishan Palyam
April 13, 2018

Supervisors:
Prof. Dr.-Ing. Gernot A. Fink
Fernando Moya Rueda, M.Sc.

Fakultät für Informatik
Technische Universität Dortmund
http://www.cs.uni-dortmund.de

"*Dedicated to the affectionate and unconditional love of my grandparents*"

ACKNOWLEDGEMENT

I would like to express my gratitude to the people who gave their invaluable support during the time of my thesis. Special thanks to my supervisor Fernando Moya, for constantly providing motivation and numerous useful suggestions throughout the course of the entire work. I also wish to express my sincere thanks to Professor Gernot Fink for giving the opportunity to work in the Department of Informatik and guiding me in the right direction. This thesis was a great opportunity for me to explore and acquire knowledge about the latest methods used in the field of Computer Vision and deep learning. The course of Computer Vision was certainly beneficial for me to get interested in this area and pursue it in the future as well. Thanks also to all the colleagues in the department for an encouraging environment to work. It wouldn't be enough to end without expressing thanks to my family and friends who are always beside me with their wonderful support.

CONTENTS

1 INTRODUCTION

The process of making realistic and believable changes in the semantic features of objects in images is known to be challenging and difficult. Image editing may involve transformation of human faces, modifying the color, altering the appearance or texture of an object, or changing the season in outdoor images [UGB+16]. Algorithms have been hand-crafted for such specific tasks until now, which need significant effort and expert knowledge. A method adopted is considered efficient when it makes the required changes while retaining other features and to be as similar as possible to the original image. It should also be robust to the several types of variations in background or illumination conditions. In case of face images, the feature changes can be - adding a smile, eyeglasses or changing facial expressions. Such changes are clearly noticeable when applied to face images that have varying pose, skin tone or other features of a person.

Convolutional Neural Networks (CNNs) are taking the forefront in the field of computer vision. In the context of image editing, CNNs have consistently shown better results when compared to previous standard benchmarks in terms of the quality of changes obtained, scalability for higher resolutions and robustness [ZIE16] [WG16]. Particularly, deep CNNs, which have more number of stacked layers, learn complex features and patterns in images. Also, recent advancements in the area of deep learning have led to designing of models which are suitable for performing a range of image modification tasks, for example - colorization of a gray-scale image [ZIE16], generating images of indoor scenes [WG16] or even modifying illumination conditions in an image [SYH+17]. Two approaches based on learning the reconstruction objectives, namely the Variational Autoencoder (VAE) and on Generative Adversarial Network (GAN), are prominently used for this purpose. These approaches and their variations are being used to produce the changes in images currently. These generative models learn the distribution of natural images in a low dimensional latent space before manipulating the images.

Figure 1.0.1 shows four types of changes on sample face images obtained from a GAN model. Although the generative methods achieve the desired changes of the features in images, they have drawbacks. The VAE framework tends to produce images of blurry, low quality and directly controlling the desired change in terms of the model parameters is difficult in GAN [BLRW16].

Figure 1.0.1: Examples of changes done by a GAN model on face images [GD17]

1.1 MOTIVATION

Considering the modification of a face image, it is a tedious task to generate visually aesthetic changes while preserving the identity of the person in the image. The authors of [UGB+16] propose a simpler method called Deep Feature Interpolation (DFI) for performing facial feature transformations in face images. For the purpose of mapping face images to an Euclidean subspace, where the underlying manifold is linearized, CNNs can be used [BMDR12]. This follows the fact that a CNN, which is trained for large image classification tasks, efficiently captures feature variations in its network layers and is able to separate the image classes linearly [SZ14]. Utilizing this hypothesis, the manipulation of images can be simply done by linear interpolation in the feature space.

(a) Input image (b) Modified image for aging

Figure 1.1.1: Modifications done on a high-resolution image for aging using the DFI method.

The technique can be accounted as simple in terms of several factors. The major factor is that DFI uses pre-trained convnets, which does not involve the process of

training a network from scratch. Another factor is its ability to perform a variety of changes in facial features of a face image because the method is not designed for a specific task. This method requires a set of images to produce image modifications, which can be controlled by a free parameter. Desirable amount of transformation can be obtained by varying another hyperparameter. In addition to these factors, this method is also scalable to high-resolution images as shown in Figure 1.1.1. Moreover, the authors of [UGB+16] claim that this technique shows improved quality of results for various image modification tasks compared to the existing state-of-the-art methods. They also suggest using this method as a baseline comparison for complex generative algorithms on certain image transformation tasks. Therefore, it is highly beneficial to examine the advantages of using DFI.

1.2 OBJECTIVE

The objective of this thesis is, firstly, to provide a quantitative evaluation besides the qualitative results of applying the DFI method on standard face datasets to change facial features, particularly for six desired attributes and computing the face verification measurements for each of the attribute changes by varying the hyperparameter values. Secondly, the goal is also to obtain different feature transformations on the face dataset of Deep Face Recognition [PVZ15] using DFI and analyze the variation in classification accuracy.

1.3 OUTLINE

This thesis is organized as follows: Chapter 2 introduces the concept of Artificial neural networks and the Gradient descent algorithm used for training them. It also describes the necessary details related to Convolutional neural networks. Chapter 3 highlights the similar methods used for facial attribute editing in face images. This chapter gives a description of the two aforementioned generative models based on deep learning methods. Going further, this chapter explains the respective CNNs employed for the DFI as well as the Face recognition tasks. The technique of Deep Feature Interpolation is presented in chapter 4. This chapter gives the detailed explanation of the procedure followed in DFI for mapping a face image into the feature space, carrying out the linear interpolation and the reconstruction process to obatin the modified image. The method followed for the face verification of modified images is also discussed in this chapter. Chapter 5 discusses the experiments performed in this thesis and also gives an analysis of their results. Finally, the relevant conclusions are provided in chapter 6.

2

FUNDAMENTALS

This chapter gives the basic description of an Artificial neural network, which is a general term used for computing systems motivated by the structure of biological neural systems. A brief description of the elementary components of a Convolutional neural network, which are large multi-layered networks used for deep learning methods, is also included in this chapter.

2.1 ARTIFICIAL NEURAL NETWORKS

The theories proposed by McCulloch and Pitts in 1943 [MP43] provides an analysis of biological neural activity formulated through logical algorithms. Based on subsequent researches, new and complex models have been developed, which are used for the applications of artificial intelligence. The information processing in a Biological neural network (BNN) is carried out by an interconnection of a huge number of smaller building blocks or cells called neurons [Roj96]. A typical neuron of the brain consists of four essential parts namely: cell body, dendrites, axon and synapses [LJK16]. These elements assist in the communication and exchange of the information. Synapses are contact points that connect one neuron with another. The synapses determine the amount of signal carried from one neuron to another, which is often called as the synaptic strength [Roj96]. When the strength of the input to the cell body rises above a threshold, it fires an impulse that is transmitted further. The storage, processing and transmission involves a combination of complex chemical and electrical processes. In simple words, each neuron receives the information signal and produces corresponding output response. Modern artificial systems are built by utilizing and adapting the properties of BNNs into their architecture.

Artificial neural networks or ANNs learn to perform tasks such as image or speech recognition by analyzing examples or data related to that specific task. They are composed of several computing nodes or units called McCulloch-Pitts (MCP) neuron, analogous to that of a neuron in BNN. A MCP neuron transforms the inputs to a resulting output by a generalized primitive function [Roj96]. The primitive function is formed by two functional parts. An *integration function* reduces all of the input values to a single intermediate value by summation and next a non-linear *activation function*

evaluates this intermediate value to produce the output[Roj96]. The activation function of the neuron works on the principle of threshold logic. The output is calculated by comparing the total excitation with a threshold. Therefore, basic logical gates like *AND* and *OR* can be implemented with the help of a single McCulloch-Pitts neuron.

2.1.1 *Perceptron*

A MCP unit operates similar to a conventional logic gate. Networks formed by connecting such units can compute logical functions, which can be modeled by specifying the relevant model parameters beforehand [Roj96]. Here the edges of the input channels are unweighted. Further research led to the evolution of the Perceptron algorithm invented by Frank Rosenblatt [Ros62]. Inorder to adapt the network for complex pattern recognition problems, weighting of the inputs were introduced. Accordingly the tuning of model weights, also known as 'learning', is achieved by applying algorithms like Gradient Descent (refer subsection 2.1.4). The weights on the inputs can be viewed as the synaptic strength in a BNN [AH17].

A typical Perceptron is given in Figure 2.1.1. The threshold θ and the weights of the neuron can be represented in a single vector as $W = (w_1, w_2, ..., w_n, -\theta)$ and a corresponding 1 is added to obtain the input vector as $X = x_1, x_2, ..., x_n, 1$. The output can be represented in a matrix notation as:

$$f(g(X,W)) = \begin{cases} 1, & \text{if} \quad W^T X \geqslant 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.1.1}$$

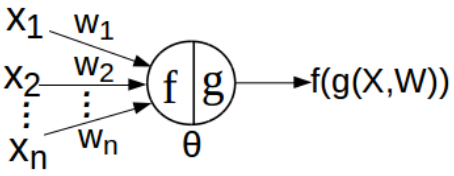


Figure 2.1.1: A visual representation of a Perceptron having threshold θ

Eventually, Minsky and Papert [MP72] made further development to the perceptron. In contrast to Rosenblatt's perceptron, the Minsky-Papert perceptron (or MPP) accepts input values in a subspace of real numbers i.e., $[0, 1] \subset \mathbb{R}$. A MPP is able to solve problems that are linearly separable in input space. To understand perceptron learning, one should consider a MPP as a classifier. The set of positive examples (P) are patterns

for which the output of perceptron is $w'x > 0$. The perceptron produces a negative output ($w'x < 0$) for samples in Negative set (N). The weights are initialized with random values and involves the following procedure for correcting them:

1. Evaluate the output of perceptron for each of the input $x \in P \cup N$
2. If the result is wrong, change the weights
   - If $x \in P$ and $w_t'x < 0$, $w_{(t+1)} = w_t + x$; t++
   - If $x \in N$ and $w_t'x > 0$, $w_{(t+1)} = w_t + x$; t++
3. Repeat the steps from 1. until the output is correct for all the test patterns.

Even though the Perceptron accomplishes an improvement over MCP, it fails when implemented for non-linear separation like exclusive-OR (XOR) or in determination of parity. An arrangement of several MPPs in one input layer feeding forward to an output layer is known as a single-layer perceptron (SLP). A SLP is the simplest form of a neural network. Instead of a threshold, an SLP uses a nonlinear activation. A SLP separates an input space into two subspaces similar to MPP. The following section explains how the architecture of ANN evolved for solving complex issues and increasing the computational power of the system.

2.1.2 *Multi-layer Perceptron*

A multi-layer perceptron (MLP) is a feed forward network having acyclic connections. It consists of an input layer, an output layer and hidden layers. Each layer can contain arbitrary number of nodes. Each node of a layer is connected to all nodes of the preceding and succeeding layers. Hidden layers are the layers of perceptrons between the input and output layer. A SLP separates an input space into two subspaces whereas a MLP is able to identify a convex region in an n-dimensional space. The MLP depicted in Figure 2.1.2 has D input neurons, n_H hidden neurons in a single hidden layer and a c-dimensional output vector.

Instead of manually adjusting the weights of the network, they can be updated automatically by a learning algorithm for solving distinct problems. This is achieved by minimizing the error gradient as discussed in subsection 2.1.4. Different types of activation functions are used in a network depending on their advantages. The various types of activation functions are discussed next.

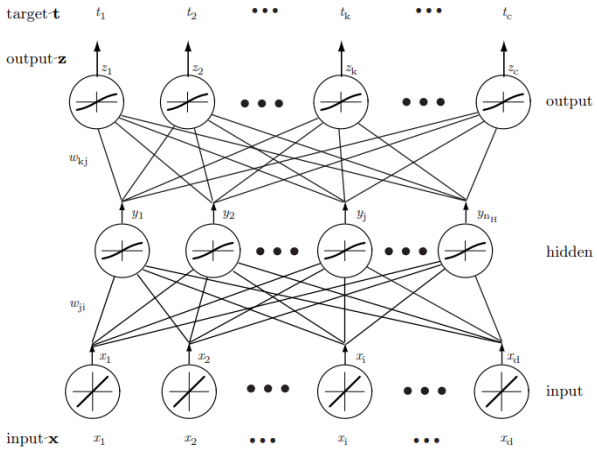


Figure 2.1.2: A MLP network having a single hidden layer. A sigmoid transfer function (see subsection 2.1.3) is used for the neurons in hidden and output layers. [DHS01]

2.1.3 *Activation Functions*

The choice of the network topology and the proper activation function of the neurons are important factors to be considered for effective training of the network. At least one neuron with non-linear activation is enough to make the network non linear [AH17]. Continuity and differentiability of the function is important for back-propagation as explained later in subsection 2.1.4. Besides these two factors, it is desirable if a identity mapping for values near origin exists. An activation or transfer function essentially limits the output to have values within a finite range. Mainly, characteristics of three kinds of non linear activations are discussed here.

- **Sigmoid or Logistic** - A sigmoid is a smooth, continuous non-linear function as plotted in Figure 2.1.3. One can observe that the function maps the output as $f(x) : \mathbb{R} \to [0, 1]$. Equation 2.1.2 calculates the sigmoid function and its derivative. As the output saturates to 1 when $|x|$ increases, the gradient becomes approximately zero and the weight changes are negligible while learning [AH17]. This is known as *vanishing gradients* problem in back-propagation (refer subsection 2.1.4).

$$f(x) = \frac{1}{1+e^{-x}} \quad , \quad f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = f(x)(1-f(x)) \tag{2.1.2}$$

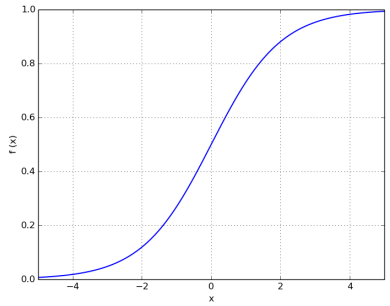


Figure 2.1.3: A logistic or sigmoid function limits f(x) within the range [0,1]

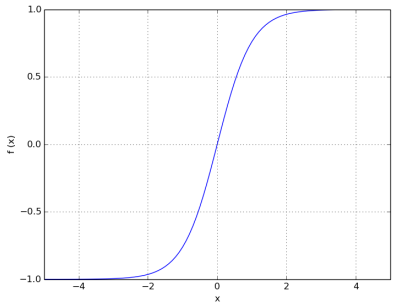


Figure 2.1.4: Tangent hyperbolic function having range [-1,1] for the output f(x)

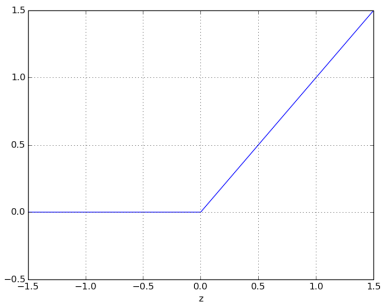


Figure 2.1.5: A ReLU activation function having range $[0, \infty)$ in $\mathbb{R}^+$

- **Hyperbolic tangent** - A hyperbolic (tanh) function is a rescaled version of sigmoid. Its function value and the derivative is given by Equation 2.1.3. The plot Figure 2.1.4 shows that unlike a sigmoid, the output is mapped as $f(x) : \mathbb{R} \to [-1, 1]$. Moreover, it approximates a linear function near origin.

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad , \quad f'(x) = 1 - f(x)^2 \tag{2.1.3}$$

- **ReLU** - A ReLU or Rectified Linear Unit (as shown in Figure 2.1.5) has a function value, which is zero for negative inputs and linearly proportional for positive input. Equation 2.1.4 shows the function and its derivative. From this, one can conclude that the derivative in $\mathbb{R}^+$ always remains 1 and never saturates.

For network architectures such as a CNN (refer section 2.2) that have many hidden layers, this helps in avoiding the *vanishing gradient* problem [AH17]. One drawback of using ReLU activation is that it may lead to a "dead" neuron such that the neuron never activates during training [LJK16]. This might happen when the weight of the neuron becomes negative and for such a value, a ReLU always returns zero [AH17].

$$f(x) = \max(0\ ,\ x) \quad , \quad f'(x) = \begin{cases} 1, & \text{if} \quad x \geqslant 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.1.4}$$

Other than these, there are different types of transfer functions also like Softsign, Leaky ReLU, Softplus, Exponential linear units (ELU) [AH17].

2.1.4 *Learning with Gradient Descent*

This section describes an underlying method used for training a layered ANN. As already seen in subsection 2.1.1, the learning algorithm modifies the parameters according to the output of the MPP. In case of a SLP, the objective is to minimize the total number of incorrect classifications. For a weight w and its set of misclassified patterns $F(w) = \{x : w'x < 0\}$, the optimization can be formulated as [DHS01]:

$$\begin{aligned} \text{Objective}: \quad & \min f(w) \\ \text{where} \quad f(w) = & - \sum_{x \in F(w)} w'x \end{aligned} \tag{2.1.5}$$

To obtain the minimal value, the Gradient Descent (GD) method is applied. In the GD approach of optimization, the gradient of the objective function, $\nabla f(w)$, is computed iteratively. The new solution is moved along the negative direction of the gradient, which finally converges to give a minimized objective [DHS01]. The update rule to modify the weights is given as:

$$w_{t+1} = w_t - \eta \nabla f(w_t) \tag{2.1.6}$$

where $\eta > 0$ is a constant called learning rate. The choice of a suitable value for η is important for the convergence process. The gradient for the objective in Equation 2.1.5 is calculated as:

$$\begin{aligned}\frac{\partial f(w)}{\partial w_i} &= -\frac{\partial}{\partial w_i} \sum_{x \in F(w)} w'x = -\frac{\partial}{\partial w_i} \sum_{x \in F(w)} \sum_{j=1}^{n} w_j x_j \\ &= -\sum_{x \in F(w)} \frac{\partial}{\partial w_i} \sum_{j=1}^{n} w_j x_j = -\sum_{x \in F(w)} x\end{aligned} \tag{2.1.7}$$

When this gradient is substituted in Equation 2.1.6, the new weight vector can be calculated as in Equation 2.1.8. One can observe that the updated vector is a simple addition of a multiple of the sum of misclassified samples to the previous weights. In a geometric sense, the gradient of f(w) is proportional to the sum of distances from the misclassified samples to the decision boundary [DHS01].

$$w_{t+1} = w_t + \eta \sum_{x \in F(w)} x \tag{2.1.8}$$

GD with Back-Propagation

In contrast to the optimization as seen before in SLP, the objective in MLP is to minimize an error by gradient method. A smooth activation function like the sigmoid (refer subsection 2.1.3) is chosen for a neuron in MLP. Since such a function is differentiable everywhere and helps in avoiding discontinuities while computing the gradient. For a MLP, a classification error or Total Sum Squared Error (TSSE) is defined as given in Equation 2.1.9. This error is used to efficiently change the weights of all the layers.

$$E(w) = \sum_{p \in D} \frac{1}{2} \sum_{k=1}^{C} \|z_{k_p} - z^*_{k_p}\|^2 \tag{2.1.9}$$

where:

- D $\rightarrow$ set of training data tuple (x,z)
- z $\rightarrow$ output of the net
- z^* $\rightarrow$ target output

Because there is no desired output specified for the hidden neurons, the error cannot be computed directly. For this reason the GD is used in combination of a back-propagation algorithm. In back-propagation, two passes of the network are carried out. First is the *'forward pass'*, where the inputs of the training set are fed into the network one by one. In this step, the output of the hidden neurons as well as the final output z are computed by the network. Next, the gradients are calculated for each of the layers starting from the output layer until the initial layer. This is known as the *'backward pass'*, as the error gradient is propagated from the output layer 'back' to the hidden layer. For a MLP as shown in Figure 2.1.2, the update rules for the weights are given by Equation 2.1.11 and Equation 2.1.10.

$$w_{kj_{t+1}} = w_{kj_t} - \eta \ \frac{\partial E(w_{kj}, w_{ji})}{\partial w_{kj}} \tag{2.1.10}$$

$$w_{ji_{t+1}} = w_{ji_t} - \eta \ \frac{\partial E(w_{kj}, w_{ji})}{\partial w_{ji}} \tag{2.1.11}$$

This rule can be generalized for a network having layers $l = S_1, S_2, ..., S_L$ and a weight matrix having weight $'w'_{mn}$ for an edge from neuron m with output o_m to neuron n with output o_n. One can compute the updated weight by Equation 2.1.12.

$$w_{mn_{t+1}} = w_{mn_t} - \eta \,.\, o_m \,.\, \delta_n \tag{2.1.12}$$

where,

$$\delta_n = \begin{cases} o_n \ . \ (1 - o_n) \ . \ (o_n - z^*_n), & \text{if } n \in S_L \\ o_n \ .(\ 1 - o_n) \ . \ \sum_{k \in S_{l+1}} \delta_k w_{nk}, & \text{if } n \in S_l \text{ and } l < L \end{cases} \tag{2.1.13}$$

The δ_n is called the *backpropagated error* because it is the accumulated error of the backward computation up to the neuron n. Therefore, one can observe that the error gradient is nothing but the output value at node multiplied with the backpropagated error. Therefore, for a sigmoid activation function, the error gradients for Equation 2.1.11 and Equation 2.1.10 can be computed by Equation 2.1.14 & Equation 2.1.15 respectively.

$$\frac{\partial E(w_{kj}, w_{ji})}{\partial w_{kj}} = y_j \,.\, \underbrace{2 \, (z_k - z^*_k) \, z_k \, (1 - z_k)}_{\delta_k} \tag{2.1.14}$$

$$\frac{\partial E(w_{kj}, w_{ji})}{\partial w_{ji}} = x_i \cdot \underbrace{y_j\,(1-y_j)\sum_{k=1}^{C}\delta_k w_{kj}}_{\delta_j} \tag{2.1.15}$$

The back-propagation learning process for MLP can be summarized as follow:

1. Apply the input pattern x and propagate it for forward pass.
2. Calculate the error gradient for output layer $\frac{\partial E}{\partial w_{kj}}$.
3. Proceed to calculate the error for preceding layer successively until the first layer.
4. Update the weights using the gradient according to Equation 2.1.12.

Based on the type of update followed, different training protocols can be implemented. Two of them are mentioned here:

- Online learning - the weights are updated after back propagating each input pattern.
- Batch learning - also termed as learning by epoch, the weights are updated by summing the gradients on a mini-batch of all the training patterns.

2.2 CONVOLUTIONAL NEURAL NETWORKS

As seen in the previous section, the neurons in a hidden layer of ANN are fully connected to all the neurons of a previous layer. This implies that the number of network parameters increases correspondingly to the image resolution and would lead to over-fitting [LJK16]. Inspired by the cognitive vision in animals, which is invariant to the position of a pattern, the structure of ANN was adapted to exploit the spatially local correlation found in images [PG17]. A hierarchical design is inherently useful for recognition architectures because in images, pixels are assembled into edglets, edglets into motifs, motifs into parts and eventually objects are constructed by parts [LKF10]. This led to the development of Convolutional neural networks (CNNs).

A CNN is usually a sequence of many number of hidden layers stacked in multiple stages. By forwarding an input image through the layers of a CNN, the original image is transformed from original pixel values to obtain a final output. The output of a CNN can be utilized as a new representation of an input image or as a class score to

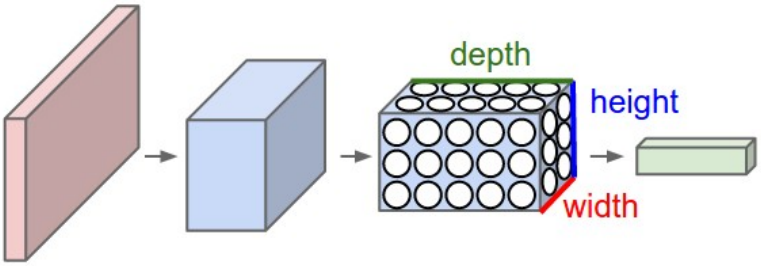


Figure 2.2.1: An example showing the 3D arrangement of neurons in a layer of CNN. [LJK16]

perform object classification [UGB+16]. The architecture of a typical CNN consists of an input layer at the beginning, followed by a combination of layers organized in a particular order and finally composed of fully-connected layers, which are similar to layer of a ANN (refer subsection 2.1.2). The main types of layers used to build a CNN architecture are explained next.

Convolutional Layer

Convolutional layers are important building blocks of a CNN used for feature extraction along the height and width of an input. The input and output of a layer is a set of arrays called *feature maps* or *activation maps* and each feature map represents a specific feature extracted from all the locations of input [LKF10]. In a convolutional layer, neurons are arranged in a 3D structure having height, width and depth as shown in Figure 2.2.1. Similar to a neuron in ANN (Equation 2.1.1), a neuron in this type of layer still computes the dot product of their weights with input volume followed by a non-linear function[1]. But the connectivity of a neuron is restricted to a small subregion of the input called receptive field [LJK16].

Convolution is a mathematical operation performed over two discrete or continuous functions. In the context of a CNN, it is used as a feature detector in a convolutional layer. As an example, the pixel values in a image or a feature map output from another layer of CNN can be considered as a discrete function as shown in Figure 2.2.2. Convolution is performed over 3 x 3 subregion of input using a kernel of the same size as another discrete function. The parameters of kernel (or filter) and input are

1 In a CNN, an activation layer contains activation functions for each of the neurons in a preceding convolutional layer. ReLU function (refer subsection 2.1.3) is used as activation function because of faster learning and to avoid vanishing gradient problem [KSH12]

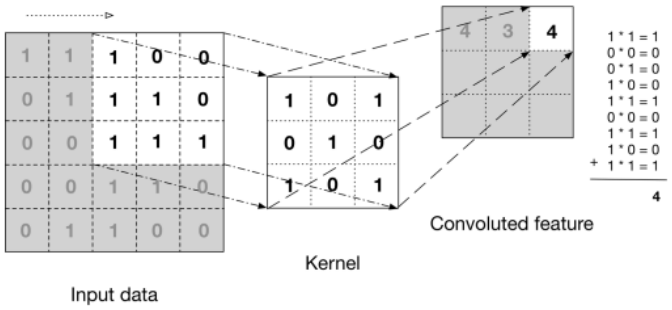


Figure 2.2.2: Convolution operation depicted as matrix multiplication using a 3 x 3 filter (or kernel) on input of same size. [PG17]

multiplied element-wise and their summation is the corresponding entry in output feature map. The size of a filter is a hyper-parameter of convolutional layer.

The sets of weights in a convolutional layer used for convolving an input is referred as a filter, used for producing a feature map. A convolutional layer consists of multiple filters and the number of filters is equal to the depth of that layer. The convolution is performed using a filter only on a small window of input volume, sliding the same filter along the spatial dimensions of the input and computing the feature detections in the output. In other words, when we consider a 2D slice of depth in a layer as 'depth slice', all the neurons in that particular depth slice share the same weights. Usually, a filter slides along the input dimensions in steps of pixels known as stride, which can be used to control the size of the output volume. The local connectivity of neurons and sharing of parameters in a convolutional layer results in a better computational efficiency [LKF10]. Gradient descent method is used to learn the weights in this type of layer and the learned filters activate a neuron when a particular pattern occurs in the neuron's receptive field [PG17].

Pooling Layer

The pooling layer down-samples a feature map from its previous layer. This layer is usually introduced between convolutional layers for reducing the spatial size of a feature map thereby help control over-fitting [LJK16]. The intuitive reasoning behind this layer is to provide translational invariance to a feature since feature detection is more important than the feature's exact location in a feature map. Therefore pooling layer preserves the detected features in a smaller representation and reduces the number of parameters.

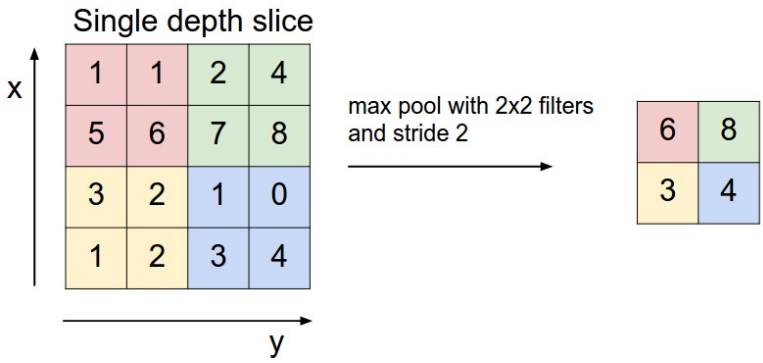


Figure 2.2.3: Max-pooling operation using a 2 x 2 filter. An output entry is the largest input value within the filter bounds and the filter advances with stride = 2 [LJK16].

The most common subsampling functions used are max-pooling and average pooling. The filters in pooling layer operate independently on every depth slice of input. For max-pooling operation, a neuron outputs the maximum value of the input in the filter window as shown in Figure 2.2.3. Likewise, the average over the input values in filter window forms the output of average pooling. Max pooling has demonstrated faster convergence and better performance in comparison to the average pooling and other variants such as l2-norm pooling [SMB10].

Fully Connected Layer

The fully connected layer has the same configuration as a hidden layer in MLP (refer subsection 2.1.2), where the neurons in this layer are connected to all the activations in the previous layer. Fully Connected layer are generally used at the end of CNN for classifying the input image into various classes.

3 RELATED WORK

In this chapter, the related approaches for manipulating features in images are introduced. In particular, this chapter discusses a traditional approach for aging in face images in section 3.2. An introduction to machine learning and generative models is presented in section 3.3, and then, two relevant methods based on deep learning that are popularly used to generate data-driven attribute transformations in images are explained. Further, the architecture of two pre-trained CNNs, employed for the image editing method and the face recognition tasks performed in this thesis, are presented.

3.1 MODIFYING FEATURES IN FACE IMAGES

Image editing can involve transformation of objects, changing of facial features of a person or altering the texture of a material in an image. Earlier approaches for producing semantic changes in the features of an image were specifically designed for a distinct task. For example, the approach introduced in [BV99] explicitly defines certain constraints on facial geometry and texture before modeling 3D faces. Then, a 3D face reconstruction having the changed facial expression is synthesized from a pair of 2D input images. Similarly, the technique presented in [IKS14] involves modeling of illumination conditions for aging of a person in a face image. The method in [IKS14] also requires a process of compiling a large dataset containing 40K images of people at different ages. For automatic removal of eyeglasses from a frontal face image, the method proposed in [WLS+04] uses a learned statistical mapping between pairs of face images with and without eyeglasses. Although the previous methods accomplish convincing results, these approaches are difficult when generalized for "in-the-wild" face images, because of large variations in identity, expression or pose of a person. Due to recent focus on using CNNs in computer vision tasks and with major progress in the field of deep learning, several models have been developed to modify various kinds of features in face images such as pose or emotion of a person [BCW+17]; lighting, appearance or expression [SYH+17]. These models are successful in synthesizing visually realistic changes of a desired feature without affecting other properties in an original image. In the following section, an introduction to machine learning & generative models is presented first, and then the two relevant methods popularly

used for modifying face images are explained. The following section describes a classical approach for modifying age in face images.

3.2 A CLASSICAL APPROACH FOR AGING IN FACE IMAGES

An illumination-aware technique to progress the age of a person in a face image was presented in [IKS14]. This approach produces the age-progressed image considering the pose, expression and illumination conditions. Although illumination and shading are 3D effects, the authors modeled the illumination conditions of an image in 2D using an illumination modeling technique. In this technique, the differences in shape and texture between ages are applied while maintaining the illumination conditions of an input image. The authors created a large dataset consisting of images of people at different ages through internet searches, spanning an age range of 0 to 100. The images were categorized into 14 clusters according to age groups as follows: 0, 1, 2-3, 4-6, 7-9, 10-12, 13-15, 16-24, 25-34, 35-44, 45-56, 57-67, 68-80 and 81-100. This dataset contained a total of 40K images with an average of 1500 images for each age cluster. The process of obtaining an output with age progression for an input face image using this approach is illustrated in Figure 3.2.1. The input image is from a source cluster 's', here it is 2-3, which was progressed to an age of a target cluster 't', here it is 57-67. The first step in this process was warping the input image by the method of [KSSGS11] to obtain a frontal pose image I. The images in the source and the target clusters were also warped using this method.

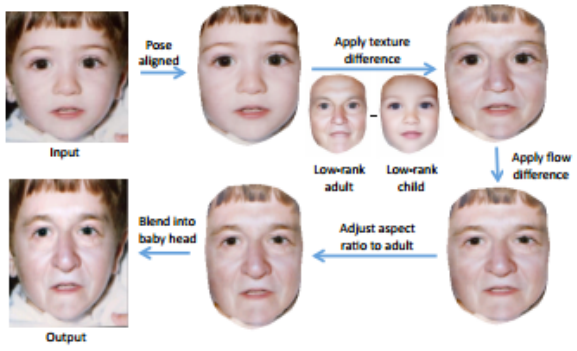


Figure 3.2.1: This figure shows an overview of the steps for illumination-aware aging of a face image. [IKS14]

The next step was to add the texture difference between the source and the target clusters to the input. To this end, the illumination of the images in the source and

the target clusters were matched, or as the authors say, "relighted" to the illumination of image *I*. To start with, the "collection flow" method [KSS12] was used to obtain dense correspondence between the images in each cluster. Collection flow method computes the flow between any pair of n images from a large face dataset. The frontal images of each of the source and the target clusters were given as input to the collection flow method. For the images in each cluster, a matrix M with dimensions $[m \times n]$ was constructed, where n is the number of images in the set and each column has the pixels of an image J_i laid out in a vector. The flow estimation was done by projecting M onto a low-rank subspace using singular value decomposition in which facial expressions and pose are normalized while lighting is preserved. This is called as *expression neutralization* by the authors of [KSS12]. Then, an intermediate optical flow[1] was calculated between the expression normalized image $J_i^{'}$ and an input image J_i in the cluster. Using this intermediate flow, the image J_i was warped to $J_i^{'}$. The process of projecting onto the low-rank subspace, computing the intermediate flow and warping is applied iteratively to obtain flow-aligned images (see Figure 3.2.3). This iterative process is summarized as follows:

1. Initialize k = 4 and intermediate flow F_i to identity. The input images are stacked as columns of the matrix M.
2. A rank-k approximation M_k is computed for M using singular value decomposition (SVD) and setting the singular values k-n to zero. Each column of M_k is a low-rank projection of the image in corresponding column of M, as shown in Figure 3.2.2. The projected images $J_i^{'}$ are extracted from the columns of M_k.
3. The optical flow F_i from $J_i^{'}$ to J_i is computed by the method of [L+09] and J_i is warped to $J_i^{'}$ using this flow.
4. Update k = k + 1 and repeat from step 2 until flow converges.

The flow-aligned images obtained from the collection flow method are averaged. As shown in Figure 3.2.3, the flow-aligned averages are much sharper compared to the average of the original images of a cluster. However, the illumination was observed to be unrealistic in the flow-aligned averages and subtle texture differences in the faces were reduced [KSSGS11]. Hence, the authors of [IKS14] proposed to match the illumination of an input image onto the flow-aligned average image of

1 Optical flow calculates apparent motion of objects between two images. This motion is determined by computing a 2D vector field, where each vector shows the displacement of pixels from the first image to the second image.

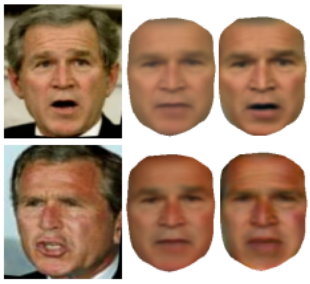


Figure 3.2.2: This figure shows the projected images obtained in the collection flow method. The first column contains the input image. The second column shows the rank-4 projection and the third column shows the rank-12 projection of the input respectively. [KSS12]

every cluster. This was accomplished by computing a rank-4 approximation of a flow aligned average and then matching the illumination. The rank-4 approximation retains the lighting and shading of the input photos, but neutralizes the changes due to facial expressions [KSS12]. Specifically, for a cluster 'j' having flow-aligned average A_j, the rank-4 approximation via SVD is given as:

$$M_j = U_j * D_j * V_j^T \tag{3.2.1}$$

where, M_j is a $[f \times p]$ matrix, 'f' is the number of images in a cluster and 'p' is the number of pixels in an image. Equation 3.2.2 is solved for the co-efficients α and the "relighted" average image that matches the illumination of I is obtained by Equation 3.2.3.

$$\min_{\alpha} \|I - \alpha V_j^T\|^2 \tag{3.2.2}$$

$$A_j^I = \alpha V_j^T \tag{3.2.3}$$

Therefore, the "relighted" average images of the source and target clusters was obtained using the illumination matching process as A_s^I and A_t^I respectively. An optical flow $F_{source-input}$ between A_s^I and I was computed and an illumination matched projection J_s was obtained by warping A_s^I to the input image coordinate frame. Similarly, the flow $F_{target-input}$ between A_t^I and I was calculated and J_t was obtained. The texture difference computed as $J_t - J_s$ was added to the image I.

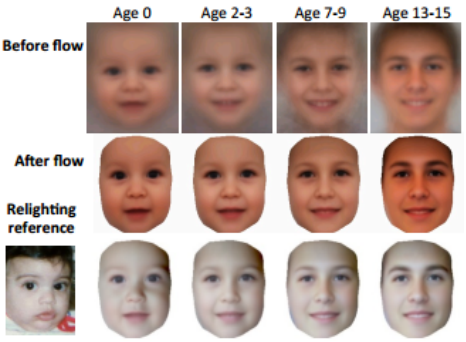


Figure 3.2.3: This figure shows the *relighting* technique used for illumination-aware aging. The first row contains the average pose-aligned images of each cluster. The average images obtained by the collection flow method are shown in the second row. The bottom row shows the *re-lit* averages with the illumination matched to the illumination of a reference input (first image in the row). [IKS14]

The flow of aging from source cluster to target cluster $F_{target-source}$ was mapped to the texture-modified image $I + J_t - J_s$. This was achieved by concatenating two flows as $F_{input-target} \circ F_{target-source}$ and applying this concatenated flow to the texture-modified image. The flow computations were done using the method of [L+09]. The change in aspect ratio was applied to the age progressed image considering the variation of shape of the face over time and finally blended into the input image using the method of [BKD+08] to obtain the modified image shown in Figure 3.2.1.

The authors performed age modification on face images of FGNET dataset [LTC02] and they conducted an experimental comparison by blending age progressed faces onto the ground-truth images. Figure 3.2.4 shows the comparison of the modified results of the age progression method to the ground-truth images. The age modified face image obtained (refer Figure 3.2.4) using the illumination-aware age progression method was blended into the ground-truth image for this comparison.

3.3 APPROACHES BASED ON MACHINE LEARNING

Machine learning is a field of artificial intelligence, where algorithms are developed to learn a specific task without explicitly programming for that task [Sam59]. The goal of a machine learning algorithm is to identify patterns in a training dataset relevant to a task. This is accomplished by establishing classification rules for an analytical model and improving the model using the training data. Machine learning algorithms are mainly classified into two broad categories and are distinguished based upon the

Figure 3.2.4: This figure shows the comparison of the age progressed image with ground-truth image for an input image (left) in age cluster of $0-3$. The second column has the age modified outputs for two target clusters and the last column has the corresponding ground-truth images. [IKS14]

type of data required for training. The two categories are: **Supervised learning** and **Unsupervised learning**. In supervised learning, the training dataset contains data samples associated with their respective label or target value. In contrast, samples from the training dataset are unlabeled in case of unsupervised learning. Therefore, unsupervised learning involves inferring the underlying probability distribution of the training data whereas supervised learning estimates the probability of assigning a label for a given input [GBC16].

Generative models

A Generative model is essentially a probabilistic model used to generate synthetic examples either by explicitly or implicitly defining the distribution over the observed data. A random sample from a fixed distribution (e.g. Gaussian) in a low-dimensional space is termed as a latent variable. Generative modeling involves mapping of a latent variable z to a sample x from a distribution over the input data in high-dimensional space [GBC16]. A generator network g accomplishes the task of this mapping. It is essentially a parameterized computational system that infers the true distribution $p(x)$ of training data. A CNN (refer section 2.2) is typically used as a generator network if input data are images. This is because images have complicated distributions and also the architecture of a CNN is suitable for learning the parameters required for nonlinear mapping of latent variables z [GBC16].

Figure 3.3.1 clearly depicts the goal in generative modeling. A generator network maps samples from a unit gaussian in latent space z to a distribution $\hat{p}(x)$. The

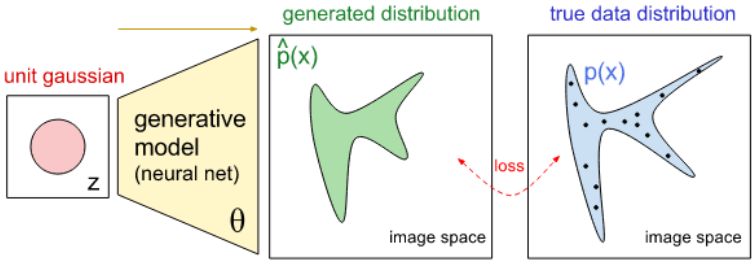


Figure 3.3.1: Schematic overview of generative modeling. The training data samples $\{x_1, x_2, ..., x_n\}$ are represented by dots in the image space of true distribution $p(x)$ and the distribution $\hat{p}(x)$ is described by the model having parameters θ [KAB+16]

generated distribution is similar to the true distribution $p(x)$ described by n training samples $\{x_1, x_2, ..., x_n\}$. In the context of generating images, the difficulty in this modeling is to obtain correct samples from the latent distribution such that the generated outputs resemble the training images, since a random latent variable does not necessarily map to a data point in the true distribution. Therefore to overcome this difficulty, the learning procedures for generative models are designed to achieve two factors. The first factor is to determine a useful way for obtaining the latent variable z. Training a generator network to render realistic images for a given z is the second factor. Two generative approaches based on unsupervised learning mechanism used for producing feature changes in face images are described next.

3.3.1 *Deep Convolutional GAN*

Deep Convolutional GAN or DCGAN, presented in [RMC15], was designed based on the generative adversarial network (GAN), proposed by Ian Goodfellow et al. [GPAM+14]. DCGAN established an useful selection of the model configuration and hyperparameters of a GAN model. GAN is a learning framework for training generative models based on a game theoretic scenario. In the GAN approach, two networks are arranged in an end-to-end manner as shown in Figure 3.3.2 and are trained in parallel. There are two main components in GAN: a generative model and a discriminative model. A generative model efficiently learns the training data distribution and generates plausible outputs that are similar to training samples. A discriminative model acts as its adversary, which performs a binary classification. The task of the discriminator is to distinguish real training data from synthesized data. One can consider the generative network as analogous to an art forger attempting to

create imitations of original paintings and correspondingly the discriminator model is analogous to an investigator distinguishing the fraudulent creations from the original artwork.

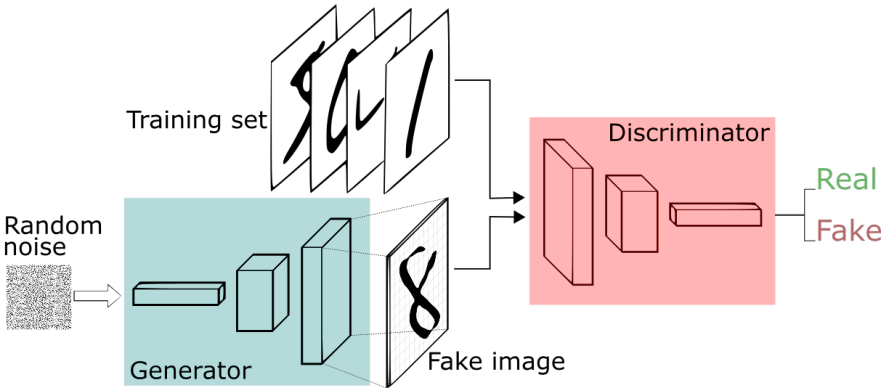


Figure 3.3.2: This figure shows a typical GAN pipeline for training a generative model. The GAN framework is a combination of a generator network producing "fake" images and a discriminator network distinguishes the "real" and "fake" images. [PG17]

3.3.1.1 *Modifications in CNN architecture*

The GAN learning process is difficult when generator and discriminator are neural networks [GBC16]. Images generated by GAN are incomprehensible and it is difficult to scale up GAN using CNNs [RMC15]. During the early stage of training, the discriminator tends to separate real images from generated images with high certainty. Hence, instability occurs when learning the generator parameters [BCW+17]. To solve the limitations of GAN, modifications in the CNN (refer section 2.2) architecture were specified in the DCGAN approach. These changes were as follows:

- Pooling functions were replaced by convolutions resulting in a homogeneous network, because a network consisting of only convolutional layers is sufficient to learn all necessary invariances in training data and exhibit state of the art performance [SDBR14]. By removing pooling layers, a network learns its own spatial upsampling or downsampling according to the authors of [RMC15].

  The generative model in DCGAN has an architecture as shown in Figure 3.3.5. It maps a low-dimensional latent vector z from a uniform distribution onto a high-dimensional image. A standard CNN having a structure as specified in Table 3.3.1 was used as the discriminator. Its convolutional layers were configured inversely

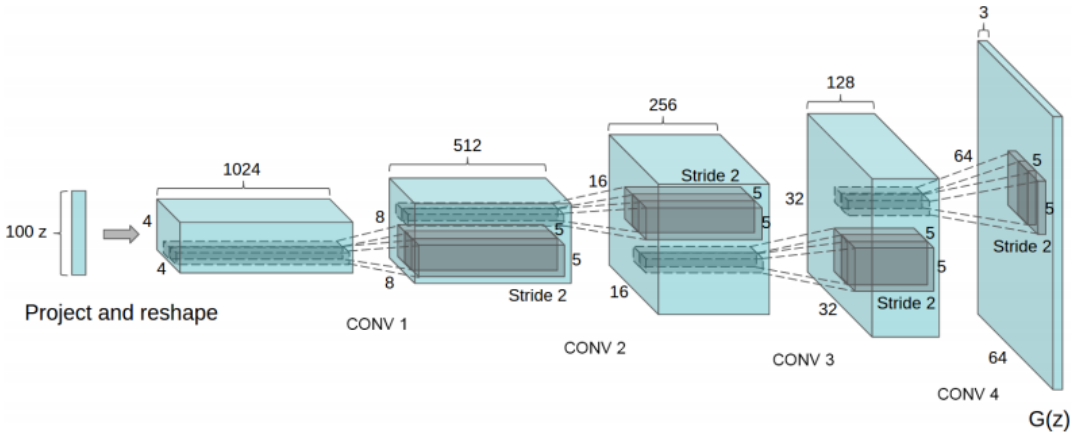


Figure 3.3.3: This figure shows the architecture of the generator network in DCGAN model. The network has a 100 dimensional latent vector as input z which is projected into a 64 x 64 image by using a series of 4 fractionally-strided convolutional layers. [RMC15]

to generator's layers, such that it downsampled the input image for binary classification. The generator is comprised of four layers performing *fractionally-strided* convolutions connected in series to increase the spatial dimension of an input representation. Fractional stride convolution was used to conveniently upsample an input by performing a convolution operation (refer section 2.2). An example of such an operation is shown in Figure 3.3.4.

- The next modification was eliminating the fully connected layers (refer section 2.2) for deeper architectures. Instead, global average pooling method was applied. In global average pooling, each feature map of a layer is averaged and transformed into a vector which is then given to the succeeding activation layer. Avoiding the overfitting problem and robustness to spatial translations of the input are the advantages of using global average pooling [LCY13]. It also increased the model stability [RMC15]. Therefore, output of the last convolutional layer in the discriminator was flattened and provided to the sigmoid output unit.

- Batch normalization was used in both the generator layers (except at the output layer) and the discriminator layers (except at the input layer). In Batch normalization, the input of an activation unit is normalized to have zero mean and unit variance. Batch normalization enables the use of higher learning rates and speeds up training considerably [IS15]. *Mode collapse* is a problem that normally

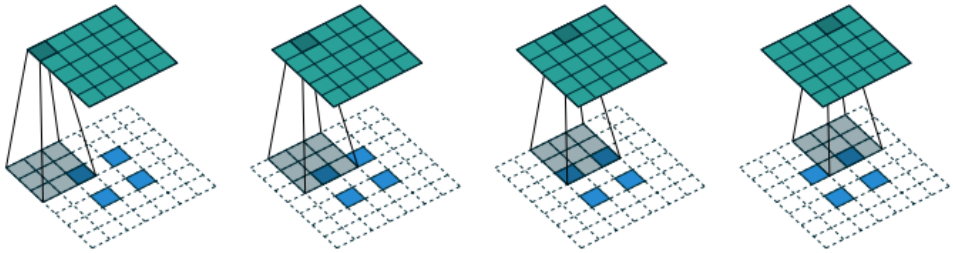


Figure 3.3.4: An example of fractional stride convolution for a 2 x 2 input (in blue). A 2 x 2 border of zeros is padded to the input after inserting a zero between them. Then a 3 x 3 kernel convolves over the input to produce a 5 x 5 output (in green). The generator's layers in DCGAN perform fractional stride convolution. [DV16]

Layers		Description
1.	Input	[64 x 64 x 3]
2.	Convolutional	64 filters of size [5 x 5 x 3], stride 2
3.	Convolutional	128 filters of size [5 x 5 x 64], stride 2
4.	Convolutional	256 filters of size [5 x 5 x 128], stride 2
5.	Convolutional	512 filters of size [5 x 5 x 256], stride 2
6.	Output	[1] Sigmoid unit

Table 3.3.1: Layer configurations of the discriminator in DCGAN

occurs in a GAN model, where the generator maps several different latent inputs to the same output. Using Batch normalization, the mode collapse problem was prevented in DCGAN [RMC15].

- ReLU activation (refer subsection 2.1.3) was applied for all the layers in the generator except for the output layer, which used a tanh function. The model quickly learned the color space of the training distribution because of the bounded range of the tanh activation [RMC15]. In contrast to a GAN that has maxout activation, Leaky ReLU activation given in Equation 3.3.1 was used for the discriminator's

layers in DCGAN. Unlike ReLU, Leaky ReLU returns a small gradient of a constant value for negative inputs.

$$f(x) = \begin{cases} x, & \text{if} \quad x \geqslant 0 \\ \alpha x, & \text{if} \quad x < 0 \end{cases} \tag{3.3.1}$$

3.3.1.2 *DCGAN objective functions*

The DCGAN objective was to train the binary classifier to effectively discriminate between true data & generated data and simultaneously encourage the generator to fit the model distribution p_g such that it eventually converges to the true data distribution p_{data}. The input to the generator G having parameters θ_G is a latent variable z from a uniform distribution p_z. The generator produces an output $G(z)$. The discriminator D having parameters θ_D computes a scalar value given by $y = D(x) \in [0, 1]$. For a given training sample x and a random latent variable z from p_z, the individual objective functions of generator and discriminator are calculated by Equation 3.3.2 and Equation 3.3.3 respectively. From this, one can deduce that each cost function depends on both generator parameters θ_G as well as discriminator parameters θ_D.

$$J^G(D, G) = L\big(D\big(G(z)\big), 1\big) \tag{3.3.2}$$

$$J^D(D, G) = L\big(D(x), 1\big) + L\big(D\big(G(z)\big), 0\big) \tag{3.3.3}$$

where L is the cross-entropy function used to evaluate the dissimilarity between two probability distributions. It is calculated as:

$$L(h^*, h) = -[h \log(h^*) + (1 - h) \log(1 - h^*)] \tag{3.3.4}$$

In Equation 3.3.3, the first term can be interpreted as the log-likelihood of D giving an output of 1 for the training sample x, while the second term indicates the log-likelihood of D assigning 0 for the generator output $G(z)$. Consequently, one can regard the generator's payoff in Equation 3.3.2 as the log-likelihood of D yielding an output 1 for $G(z)$. The generator maximizes its objective function by changing only θ_G, since it has no control over the discriminator's parameters. Correspondingly, the discriminator learns to minimize its objective (see Equation 3.3.3) by controlling only θ_D. In the perspective of game theory, this is similar to a two-player game known

as *minimax game*. The solution for a minimax game is a Nash equilibrium. A Nash equilibrium [2] for DCGAN is obtained by solving Equation 3.3.5. The *value function V(D, G)* in Equation 3.3.5 represents the complete objective function of DCGAN model. Therefore, the training criteria in DCGAN involved maximization of the value function by the discriminator and the generator concurrently minimizing V(D, G).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1-D(G(z))\right)\right] \quad (3.3.5)$$

Substituting Equation 3.3.4 in Equation 3.3.3, one can confirm from Equation 3.3.6 that the discriminator's cost in essence corresponds to a value of $-V(D,G)$ and represents the cross-entropy minimized when training a binary classifier [Goo17].

$$J^D(D,G) = -\log D(x) - \log\left(1-D(G(z))\right) \quad (3.3.6)$$

3.3.1.3 *Training of DCGAN model*

For the purpose of generating high quality face images compared to GAN, the authors of [RMC15] designed a DCGAN model as per the aforementioned guidelines. The training of the DCGAN model to achieve a solution for Equation 3.3.5 is described in this section. The primary idea of DCGAN training was balancing the optimization done between G and D. Training of DCGAN is an iterative process consisting of two elementary steps:

(i) Keeping the parameters of generator fixed, optimize the discriminator.

(ii) Fix the parameters of discriminator and optimize the generator.

A face image dataset containing 3M images of 10K identities was used for the training. The learning algorithm uses the minibatch Stochastic Gradient Descent [3] (SGD) optimization method. The SGD procedure was applied once on the generator after training the discriminator for fixed number of steps indicated by hyperparameter *k*, for DCGAN $k = 1$ was used. The learning process that is repeated for 'n' training iterations is as follows:

1. One obtains a mini-batch of *m* samples $\{z^{(1)}, z^{(2)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.

2 The Nash equilibrium for a GAN is a tuple $(\theta_D^{min}, \theta_G^{min})$, where θ_D^{min} is a local minimum of J_D and θ_G^{min} is a local minimum of J_G [Goo17]

3 Minibatch SGD method is based on Gradient Descent optimization (refer subsection 2.1.4). The key difference compared to standard GD (see Equation 2.1.6) is that in minibatch SGD the training samples are randomly shuffled and then the gradient of objective function is updated for each batch of training samples as: $w_{t+1} = w_t - \eta\nabla f_i(w_t)$, for i = 1,2,...,m training batches

2. One obtains a mini-batch of m data points $\{x^{(1)}, x^{(2)}, ..., z^{(m)}\}$ from true distribution $p_{data}(x)$.

3. The discriminator parameters are updated by computing its stochastic gradient:

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]. \tag{3.3.7}$$

4. Repeat from step 1. for 'k' number of iterations.

5. The generator parameters are updated by computing its stochastic gradient:

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]. \tag{3.3.8}$$

Figure 3.3.5: This figure shows a sample collection of face images generated using a pre-trained DCGAN model available at [Chi15].

This model showed an improvement from GAN in model stability and also learns a good representation of images for generative modeling. Figure 3.3.5 shows the generated images by a pre-trained DCGAN model. The ability of a DCGAN model to learn an image representation that distinguishes the semantic attributes of an image is demonstrated in Figure 3.3.6. The latent vectors of images were averaged for each column. Vector arithmetic was applied on these mean vectors and then given to the generator for synthesizing the face images with desired changes. An interpolation to produce varying degree of feature changes was done by adding a uniform noise to the resulting vector of the arithmetic operation. However, one can notice that it is difficult to control the amount of interpolation in the desired attribute change with the model parameters in this method.

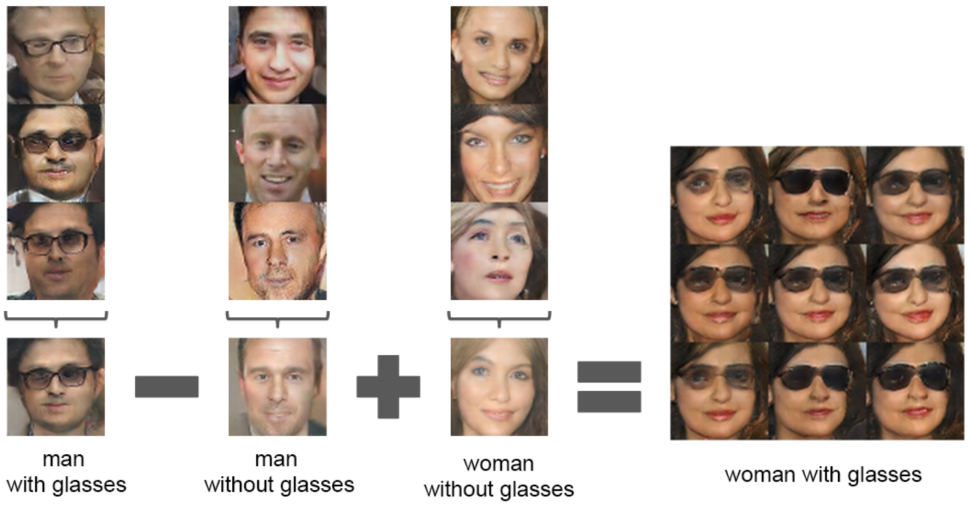


Figure 3.3.6: This figure depicts an arithmetic operation of the latent vectors to produce a feature change in the generated image by the DCGAN model.[RMC15]

3.3.2 *VAE/GAN*

VAE/GAN is a generative method proposed in [LSW15], which combines a generative adversarial network (refer subsection 3.3.1) with a Variational Autoencoder (VAE) to learn a similarity metric. This model was inspired by [GEB15], where the feature representations separating the style from content of an image was obtained using the layers of a CNN. These feature representations were used to transfer a style of one image to another image in [GEB15]. However, in VAE/GAN approach, a GAN model provided the required style and content metrics. One can observe from Figure 3.3.7 that the images generated by VAE are blurry while images produced using a GAN model are sharp but noisy. VAE has stable learning dynamics when compared to GAN [BLRW16]. The VAE/GAN approach effectively incorporated the advantages of both the models to learn better image representations. The concept of VAE is explained in the next section.

Variational Autoencoders

A Variational Autoencoder is a generative model that uses unsupervised learning to reconstruct images. It was introduced by Kingma and Welling in [KW13]. A VAE consists of two networks, an *encoder* (Enc) with parameters ϕ and a *decoder* (Dec) with parameters θ_{Dec}. The encoder network maps an input data x to a vector z in latent

Figure 3.3.7: This figure shows a comparison of generated images by VAE and GAN models. The images generated by a VAE model are in the top row and GAN generated images at bottom.[YCL+16]

space Z. The decoder reconstructs output samples from this latent space as $p_{\theta_{Dec}}(x|z)$. Equation 3.3.9 represents the two networks.

$$z \sim Enc(x) = q_\phi(z|x) \quad , \quad \tilde{x} \sim Dec(z) = p_{\theta_{Dec}}(x|z) \tag{3.3.9}$$

In VAE, the data distribution $p_\theta(x)$ is assumed to be generated from a model with parameters θ. The true posterior of $p_\theta(z|x)$ in latent space is intractable because images have complex probability distributions in a high-dimensional space [Doe16]. Therefore, the encoder having parameters ϕ infers an approximate representation of the true posterior as $q_\phi(z|x)$ by maximizing a variational lower bound on the likelihood of the data. This is in contrast to GAN (refer subsection 3.3.1), where a binary classification provides a criteria for optimizing the generative model. Therefore, VAE method provides an inference model as well as a generative model. The variational lower bound is given by Equation 3.3.10. VAE imposes a prior p(z) on the latent distribution, which is typically a gaussian with zero mean and identity covariance. This is known as *reparameterization trick* [Doe16].

$$\mathcal{L}_{VAE} = -\mathbb{E}_{q_\phi(z|x)}[\log p_{\theta_{Dec}}(x|z)] + D_{KL}(q_\phi(z|x)\|p(z)) = \mathcal{L}_{pixel} + \mathcal{L}_{prior} \tag{3.3.10}$$

In Equation 3.3.10, the first term is the expected log-likelihood of the generated data also known as reconstruction error. The second term is the Kullback-Liebler[4] divergence between the approximate distribution $q_\phi(z|x)$ and the prior $p(z)$ known as prior regularization. One can say that, the encoder restricts the latent space from

4 Kullback-Liebler divergence measures the divergence of a probability distribution Q from an expected distribution P. In the context of machine learning, it is often called the information gain achieved if Q is used instead of P for a particular task.

which the decoder reconstructs the output to a space smaller than the prior p(z) by computing the variational lower bound.

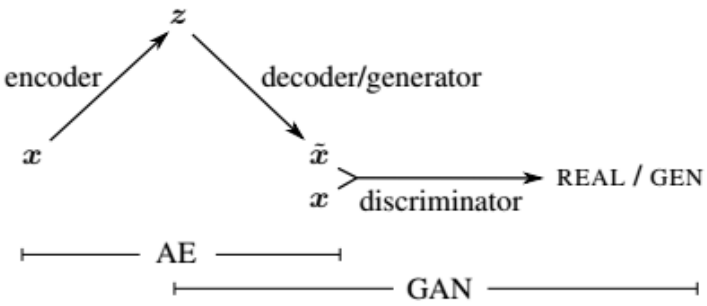


Figure 3.3.8: The combined model of VAE/GAN shows the sharing of parameters between decoder and generator.[LSW15]

The objective of VAE is to optimize Equation 3.3.10. However, the reconstruction error in Equation 3.3.10 is usually evaluated by element-wise metrics like ℓ_2 error and was not a sufficient measurement for images having invariances such as translation. Therefore, the element-wise error was replaced by a feature-wise error measurement in VAE/GAN approach to obtain a better representation of images. To this effect, a GAN discriminator was used in conjunction to VAE as shown in Figure 3.3.8. The discriminator implicitly learns a metric useful for measuring feature-wise error because it classifies real images from generated images [LSW15]. To formulate a criteria for the combined model of VAE and GAN, the hidden representation of the l^{th} layer in the discriminator (represented as $D_l(x)$) was modeled by a gaussian with mean $D_l(x)$ and identity covariance as:

$$p_{\theta_D}(x|z) = \mathcal{N}(D_l(x)|D_l(\tilde{x}), I), \text{ where } \tilde{x} \sim Dec(z) \tag{3.3.11}$$

Now, the element-wise similarity measurement in Equation 3.3.10 was substituted by the reconstruction error in terms of the GAN discriminator given in Equation 3.3.12.

$$\mathcal{L}_{D_l} = -\mathbb{E}_{q_\phi(z|x)}[\log p_{\theta_{Dec}}(D_l(x)|z)] \tag{3.3.12}$$

The value function of GAN (refer Equation 3.3.5) can be represented in the binary cross entropy form as Equation 3.3.13 for a single data point. Since the generator and the decoder perform a similar function of mapping from z to x, the generator was

replaced by the decoder in Equation 3.3.13 and is rewritten as Equation 3.3.14. Therefore, the $\mathcal{L}_{GAN}$ and $\mathcal{L}_{D_l}$ are incorporated in Equation 3.3.10 to obtain the complete optimization criteria of VAE/GAN given by Equation 3.3.15. One can interpret $\mathcal{L}_{D_L}$ as *content* error representing the ability of the model to fool the discriminator, and $\mathcal{L}_{GAN}$ as *style* error representing the model's reconstruction capability.

$$\mathcal{L}_{GAN} = \log(D(x) + \log(1 - D(G(z))) \tag{3.3.13}$$

$$\mathcal{L}_{GAN} = \log(D(x) + \log(1 - D(Dec(z))) \tag{3.3.14}$$

$$\mathcal{L}_{GAN} = \mathcal{L}_{D_l} + \mathcal{L}_{prior} + \mathcal{L}_{GAN} \tag{3.3.15}$$

The following three factors were considered essential for the training procedure:

- The discriminator was not used to minimize $\mathcal{L}_{D_l}$, since this would result in the discriminator classifying every input as generated data. The error signal $\mathcal{L}_{GAN}$ was not backpropagated to encoder, which offered a better result.
- While updating the parameters of the decoder in Equation 3.3.16, a weighting factor γ was multiplied to $\mathcal{L}_{GAN}$. This provided a proper balance between the reconstruction and the discriminating abilities of the model.

$$\theta_{Dec} = -\nabla_{\theta_{Dec}}(\gamma\mathcal{L}_{D_L} + \mathcal{L}_{GAN}) \tag{3.3.16}$$

- In addition to the prior $p(z)$, better results were obtained by using the samples from the approximate posterior $q_\phi(z|x)$ for $\mathcal{L}_{GAN}$ as:

$$\mathcal{L}_{GAN} = \log(D(x) + \log(1 - D(Dec(z))) + \log(1 - D(Dec(Enc(x)))) \tag{3.3.17}$$

The hybrid model of VAE/GAN involved training of both a GAN and a VAE. The training procedure using GD method (refer subsection 2.1.4) was as follows:

- The parameters for encoder (ϕ), decoder (θ_{Dec}) and discriminator (θ_D) are initialized.

- A random mini batch from training set is obtained.
- The samples are encoded into the latent space as $Z = Enc(X)$ and the corresponding outputs are generated by the decoder as $\tilde{X} = Dec(Z)$. The prior regularization term, $\mathcal{L}_{prior}$, and the "content error", $\mathcal{L}_{D_l}$, are calculated as:

$$\begin{aligned} \mathcal{L}_{prior} &= D_{KL}(q_\phi(Z|X)\|p(Z)) \\ \mathcal{L}_{D_l} &= -\mathbb{E}_{q_\phi(Z|X)}[\log p_{\theta_{Dec}}(X|Z)] \end{aligned} \tag{3.3.18}$$

- Samples are obtained from the prior as $Z_p \sim \mathcal{N}(0, I)$ and their corresponding content error is calculated: $X_p = Dec(Z_p)$
- The GAN objective function $\mathcal{L}_{GAN}$ is computed as:

$$\mathcal{L}_{GAN} = \log(D(X) + \log(1 - D(\tilde{X}) + \log(1 - D(X_p)) \tag{3.3.19}$$

- The parameters are updated by their gradients using Equation 3.3.18 and Equation 3.3.12 in Equation 3.3.20 as:

$$\nabla_\phi(\mathcal{L}_{prior} + \mathcal{L}_{D_l}), \quad \nabla_{\theta_{Dec}}(\gamma\mathcal{L}_{D_l} - \mathcal{L}_{GAN}), \quad \nabla_{\theta_D}\mathcal{L}_{GAN} \tag{3.3.20}$$

- Repeat from 2. for n training iterations.

The Figure 3.3.9 shows the results of changing facial features in face images of CelebA dataset(refer subsection 5.1.2). To accomplish this task, an interpolation in latent space was performed. For the VAE/GAN, the original images were rescaled to 64 x 64 pixels. The binary attributes provided by the dataset was used to select the images with a desired attribute & images without that attribute. The latent representation of the images in both of the sets was calculated and the representations of the images in each set was averaged. The difference between the mean vectors is called *visual attribute vector*, which was added to the latent representation of a test image to obtain the modified image. Notably, reconstruction of an input image is not possible by a GAN as it does not have an inference model, which is present in VAE/GAN. Therefore, VAE/GAN showed that an improvement in producing sharper images can be achieved by employing a feature-wise similarity metric instead of the element-wise distance measure of VAE.

Figure 3.3.9: In this figure, first column is the input image to the VAE/GAN model. The next column is the reconstuction, followed by desired changes of bald & bangs produced by the model. [LSW15]

3.4 VGG NEURAL NETWORK

This section discusses the architecture of the CNNs, specifically, the layer configurations of two networks used for the Deep Feature Interpolation (DFI) and face verification tasks in this thesis. The two CNN architectures are given in Table 3.4.1, where the network configuration containing 19 weight-layers is used for DFI and network with 16 weight-layers is employed for face verification.

3.4.1 *VGG for Image classification*

For the purpose of evaluating the performance of CNNs having large number of layers in image recognition tasks, the authors of [SZ14] investigated the network configurations in Table 3.4.1. In these CNNs, convolution filters of size 3 x 3 were used to enable the increase in number of layers. In addition to this, the stride of each convolution filter is 1, which essentially means that the filter is convolved over each pixel of the input. Every hidden layer is followed by a non-linear ReLU activation. In case of large number of layers, reducing the size of filters achieves the simultaneous reduction in the number of parameters, thereby helping to avoid the overfitting problem. The pooling layers perform max-pooling operation with filters of size [2 x 2] and stride 2. The stack of convolutional layers is followed by three fully connected layers and a final softmax layer. Training of the networks was carried out using batch GD with batch size of 256. An initial learning rate $\eta = 10^{-2}$ was decreased 3 times by a factor of 10 during the entire training process. The authors also mention that using

small convolutional filter size was one of the reasons for the networks converging in less epochs as compared to previous architectures despite of the large number of layers.

The input to the CNNs were random crops of training images having a fixed size of [224 x 224]. Two approaches were used for rescaling the training input which can be considered as data augmentation: (i) single-scale, where the smallest side is resized to a fixed value, two sizes - 256 and 384 were used by the authors; (ii) multi-scale, where the smallest side is rescaled by randomly sampling from a range $[256, 512]$. After training the networks, the testing was done by rescaling the input image to a predefined smallest side and then passing the input through a fully-convolutional network, which was obtained by converting the fully connected layers to convolutional layers. This resulted in a score map having varying spatial resolution and hence, they were sum-pooled to obtain a fixed size score. The final score vector was obtained by averaging the score maps of the training input and its horizontal flip.

The networks were trained on the ImageNet challenge [RDS+15] dataset containing 1.3M images of 1000 object classes and demonstrated state-of-the-art performance for image classification. The network having 19 layers reached top-1 and top-5 validation error rates of 25.5% and 8.0%. These results are better than the corresponding error rates of 40.7% and 18.2% of Alexnet [KSH12], which is the best model of the ImageNet-2012 challenge [RDS+15]. For multi-scale testing with scales of [256; 512], each of the 16-layer and the 19-layer networks achieved the same error rates of 24.8% for top-1 and 7.5% for top-5. In conclusion of [SZ14], one can say that a CNN having a large number of layers and appropriate parameters indeed helps in improving the classification accuracy.

3.4.2 *VGG for Face Recognition*

In [PVZ15], a CNN having an architecture similar to that of the 16-layer VGG network in Table 3.4.1 was adapted for the task of face recognition. For this task, the last fully connected layer of the VGG network was modified, either to have a dimension of $N = 2622$ when used as a N-way face classifier (see item (i)) or a dimension $L = 1024$ when obtaining a descriptor for face verification (see item (ii)). A mean-normalized face image of [224 x 224 x 3] was used as the input for the network, which stabilized the optimization according to the authors of [PVZ15]. The training was done on a specially curated dataset containing 2.6M face images for 2622 identities (refer subsection 5.1.3). The dataset was compiled using internet search of celebrity images and filtered to have $1,000$ images per identity using an automatic filter and a final manual annotation.

VGG Architecture	
16 weight layers	**19 weight layers**
Input [224 x 224 x 3]	
2 Convolutional with 64 filters	2 Convolutional with 64 filters
Max Pooling	
2 Convolutional with 128 filters	2 Convolutional with 128 filters
Max Pooling	
3 Convolutional with 256 filters	4 Convolutional with 256 filters
Max Pooling	
3 Convolutional with 512 filters	4 Convolutional with 512 filters
Max Pooling	
3 Convolutional with 512 filters	4 Convolutional with 512 filters
Max Pooling	
Fully connected [4096]	
Fully connected [4096]	
Fully connected [1000]	
Soft-max	

Table 3.4.1: Layer configuration of the VGG neural networks.[SZ14]

The training procedure followed the method seen in the previous section: batch GD optimization using batch size of 64 and an initial η of 10^{-2}. The smallest side of the face images was rescaled to a fixed sized of 256 and a random crop of [224 x 224] from these images was used for training. The images were flipped with a probability of 0.5 for data augmentation. The training of the network was achieved for two objectives:

(i) *Learning a face classifier* - The first objective was to recognize the face images of N distinct individuals in the training dataset. To perform the N-way classification of face images with $N = 2622$ identities, the last fully connected layer of the network having the dimension of 2622 was utilized to obtain a score vector x_t for each training image as:

$$x_t = W\phi(l_t) + b \tag{3.4.1}$$

, where l_t is the training image, $\phi(l_t)$ is the output of the second fully connected layer and the corresponding weights $W \in \mathbb{R}^{NxD}$ with $D = 4,096$. The network

was trained to minimize the empirical *softmax log-loss*, computed by comparing the score vectors to the ground truth class identity $c_t \in \{1, ..., N\}$ as follows.

$$E(\phi) = -\sum_{t} \log\left(e^{\langle e_{c_t}, x_d\rangle} / \sum_{q=1,...,N} e^{\langle e_q, x_d\rangle}\right) \tag{3.4.2}$$

where, $x_d = \phi(l_t)$ and e_c is one-hot vector of class c

(ii) *Learning face embeddings using triplet loss* - To learn a score vector suitable for face verification (refer section 4.2), the CNN trained previously was frozen, and the last fully connected layer was replaced by the layer with dimension $L = 1024$. Therefore, a new face descriptor, given in Equation 3.4.3, was defined by conveniently projecting $\phi(l_t)$ to a low-dimensional space using the weights W' and this descriptor was utilized for minimizing the objective. The objective is the empirical *triplet loss* given by Equation 3.4.4. The face descriptors x_t are better for face-verification tasks than the descriptors $\phi(l_t)$ [PVZ15]. Minimizing the empirical loss ensures that the distance between the face descriptors of one identity was minimum while the distance to face descriptors of other identities was above a triplet margin α [SKP15]. A triplet set T was prepared from the target dataset, in this case the restricted configuration of LFW face dataset (refer subsection 5.1.1), such that it contained image triplets of form (a,p,n), where image a is the anchor, p is its corresponding positive sample and n being a random negative sample of the anchor's identity that violated the triplet margin. Then, the new layer was learned for 10 epochs using stochastic GD with $\eta = 0.25$.

$$x_t = W' \frac{\phi(l_t)}{\|\phi(l_t)\|_2} \quad W' \in \mathbb{R}^{LxD} \tag{3.4.3}$$

$$E(W') = \sum_{(a,p,n)\in T} \max\{0, \alpha - \|x_a - x_n\|_2^2 + \|x_a - x_p\|_2^2\} \tag{3.4.4}$$

For testing the network, the *Equal Error Rate* (refer section 5.2) was used as a measurement for face verification in addition to the classification accuracy. The evaluation was done on "unrestricted settings" of LFW dataset (refer subsection 5.1.1) and Youtube Faces (YTF) [LWM11], which conatins 3,245 videos of 1,595 identities. For a given face image l, the face descriptors $\phi(l_t)$ were obtained after feeding patches

of size [224 x 224] from the four corners and the center after rescaling the image to three sizes of [256, 384, 512] to the CNN, resulting in a total of 30 descriptors. The final descriptor was the average of these 30 face descriptors. Faces were detected using the Deformable Parts Model (DPM) [MRML14]. In case of the YTF, K face descriptors were obtained for each test video by ranking the faces in a video by a facial landmark score and selecting the top K. The K face descriptors were averaged to represent a video. Applying this testing procedure, the network achieved results comparable with the state-of-the-art results. The network achieves an accuracy of 98.95% and 97.3% on the unrestricted setting of LFW and YTF videos respectively. Notably, the network accomplished a higher 100%-EER of 96.70% when 2D alignment was used for the test images compared to 92.83% when the test images were not aligned. Therefore, one can conclude that a deep CNN that is trained suitably without any additions, achieves results comparable to state-of-the-art.

4

METHOD

In the previous chapter, two generative approaches to modify certain desired features in face images were explained. Although these approaches produce the required modifications, training of specifically designed networks and optimizing the networks for a suitable objective is a prior requirement. Besides, the generative approaches synthesize new images with the edited features rather than modifying the original images. In addition to this, these methods do not entirely maintain the identity of the original image [UGB+16]. This chapter discusses, firstly, a novel approach comprising of four generalized steps to manipulate desired attributes on existing face images, secondly, the method of face classification and verification using the pre-trained CNN (refer section 3.4), and lastly, the approach followed in this thesis for combining both these methods for evaluation purposes.

4.1 DEEP FEATURE INTERPOLATION

Deep Face Interpolation (DFI), introduced in [UGB+16], is an image editing method. This method leverages the learned representations of pre-trained CNNs for producing desired changes in face images. In [BMDR12], the authors investigate a hypothesis that the deeper layers of a CNN effectively disentangle the underlying feature variations in images and linearize the manifold into an Euclidean subspace. This hypothesis suggests that a classification network has a better discriminative ability to linearly separate object classes. Inspired by this hypothesis, the DFI approach utilizes appropriate learned feature spaces of a CNN for editing images by linear interpolation. Specifically, the authors consider the 19-layer VGG network trained on ImageNet (refer subsection 3.4.1) because the network is proven effective for classifying objects among large number of categories. The DFI approach consists of four steps to generate the required changes in an image. In the context of modifying a face image, the details of the four steps are given in the following sections.

4.1.1 *Mapping into feature space*

For a given face image 'x' and a required attribute change, a face dataset is split into two sets of images: a source set $S^s = \{x_1^s, ..., x_n^s\}$ having images without the desired attribute and a target set $S^t = \{x_1^t, ..., x_m^t\}$ containing images with the attribute. The first step in DFI is to obtain new representations of the test image, the source set and the target set images using the pre-trained VGG network. The mapping of image x into deep feature space is represented as $x \rightarrow \varphi(x)$. The authors called the vector $\varphi(x)$ as the *deep feature representation* of x. This representation consists of concatenated activations of the CNN's convolutional layers after giving x as the input. The layers selected for concatenation should be appropriate for linear interpolation as well as image reconstruction tasks performed in the further steps. On one hand, the first few convolutional layers preserve essential information of an image in their representation and are beneficial for semantically-meaningful reconstructions [MV15]. On the other hand, the representations associated with the lower layers are linearly separable and they are suitable for interpolation [BMDR12]. Considering these two factors, the authors of DFI chose, particularly, the first convolutional layer of the last three pooling regions in the VGG network (refer subsection 3.4.1) for the purpose of creating the deep feature representation.

Next, the images in source and target sets are mapped to the feature space. To accurately identify a particular attribute in feature space, one selects the source and target images differing only in the required attribute and having other unrelated attributes similar to x. This ensures believable and realistic changes in the transformed image. The number of images in the source and the target sets is limited by a hyperparameter K. The deep feature representations of the images in the source and the target sets are computed similarly to x and these representations are averaged to obtain their mean values as given in Equation 4.1.1.

$$\overline{\varphi}^s = \frac{1}{K} \sum_{x^s \in N_K^s} \varphi(x^s), \quad \overline{\varphi}^t = \frac{1}{K} \sum_{x^t \in N_K^t} \varphi(x^t) \tag{4.1.1}$$

where, N_K^s and N_K^t denotes the K nearest neighbors of x in source and target sets respectively. The linear interpolation in feature space is performed using these averaged feature vectors as described in the next section.

4.1.2 *Linear interpolation using an attribute vector*

The second step in DFI is computing the *attribute vector* 'w' as given in Equation 4.1.2. The attribute vector is the difference of the mean feature representations of the source and target sets ($\overline{\varphi}^s$ and $\overline{\varphi}^t$). It specifies the direction of the required interpolation in the feature space as shown in Figure 4.1.1.

$$w = \overline{\varphi}^t - \overline{\varphi}^s \tag{4.1.2}$$

The vector w mainly contains the feature information regarding a desired attribute and other irrelevant features have negligible values. This is because $\overline{\varphi}^s$ and $\overline{\varphi}^t$ are sparse because of the ReLU activations in the CNN [UGB+16]. Linear interpolation is performed on $\varphi(x)$ using the attribute vector, which is the third step in the DFI approach. This is done by a displacement of $\varphi(x)$ in the direction of w in feature space as given in Equation 4.1.3. According to the authors, $\varphi(y^*)$ corresponds to the feature representation of an output face image y^*, which is the modified image having the desired attribute. The amount of interpolation is controlled by a scaling factor α. Different degree of transformation in the desired change can be visualized in the modified image by varying α. Figure 4.1.1 summarizes the process of interpolation.

$$\varphi(y^*) = \varphi(x) + \alpha w \tag{4.1.3}$$

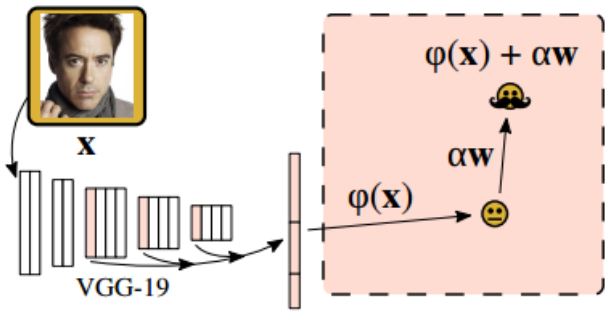


Figure 4.1.1: The DFI technique of linear interpolation in feature space. The deep feature representation of test image x is displaced in the direction of attribute vector w. [UGB+16]

4.1.3 *Reverse mapping*

The final step of DFI achieves the reverse mapping of $\varphi(y^*)$ in Equation 4.1.3 to obtain an image 'y^*' in pixel space. But the feature representation $\varphi(y^*)$ is not uniquely invertible, because CNNs are invariant to certain properties of an image such as viewpoint or illumination and the discriminative network employed in DFI discards such information irrelevant for the classification task [MV16]. Therefore, an approximate solution y producing the feature vector $\varphi(y) \approx \varphi(x) + \alpha w$ is computed. In other words, the goal of reconstruction is finding an image 'y' that has a similar network mapping to 'y^*'. To accomplish this task, the authors of DFI proceeded with an optimization algorithm known as *inversion* introduced in [MV15], as explained in the next section.

4.1.3.1 *Optimization*

The reconstruction of $\varphi(y)$ is achieved by minimizing an objective function. Precisely, the objective function to recover a color image $y \in \mathbb{R}^{H \times W \times C}$ with the desired features is given in Equation 4.1.4. It is formed by two parts: a loss function (or reconstruction error) and a natural image prior used as a regularizer. The loss function is the euclidean distance between the ideal image representation $\varphi(y^*)$ (refer Equation 4.1.3) and the approximated image representation $\varphi(y)$. In the inversion method of [MV15], the value of $\varphi(y)$ in the loss function is initialized with a random noise. One can notice that the objective in Equation 4.1.4 utilizes only the interpolated feature vector $\varphi(y^*)$ and a regularizer. Therefore, the final reconstructed image captures only this information in the inversion method, whereas in DFI, $\varphi(y)$ is initialized with $\varphi(x)$ of the original image.

$$y = \arg\min_{y} \frac{1}{2} \|(\varphi(x) + \alpha w) - (\varphi(y))\|_2^2 + \lambda_{V^\beta} R_{V^\beta}(y) \tag{4.1.4}$$

where:

$$\begin{aligned} R_{V^\beta}(y) &\rightarrow \text{Total Variation regularizer} \\ \lambda_{V^\beta} &\rightarrow \text{Regularization coefficient} \\ \|(\varphi(x) + \alpha w) - (\varphi(y))\|_2^2 &\rightarrow \text{Loss function} \end{aligned}$$

As mentioned earlier, discriminative models such as CNNs disregard information that are important for visualization. Nevertheless, plausible outputs are obtained by restricting the reconstruction to a subset of natural images $y \subset \mathbb{R}^{H \times W \times C}$. Therefore,

an appropriate *image prior* known as *total variation* (TV) regularizer, R_{V^β}, is used for regularizing 'y' in pixel space. Imposing a naturalness prior on the reconstructed image improves the reconstruction quality [MV16]. The TV regularizer given by Equation 4.1.5 encourages images to consist of smooth transitions between neighboring pixels. Figure 4.1.2 depicts the mapping of the target image 'y' to $\varphi(y)$ and the corresponding optimization objective for obtaining 'y'. For optimal tuning of the objective, the loss function and the regularizer are balanced by a regularizer coefficient λ_{V^β}. This normalizes the pixel values to a given range in the reconstructed image [MV15]. A value of $\beta = 2$ and $\lambda_{V^\beta} = 10^{-3}$ was used for the DFI implementation in [UGB+16].

$$R_{V^\beta}(y) = \sum_{i,j} \left((y_{i,j+1} - y_{i,j})^2 + (y_{i+1,j} - y_{i,j})^2 \right)^{\frac{\beta}{2}} \tag{4.1.5}$$

where $y_{i,j}$ is the pixel value at (i, j) in image y

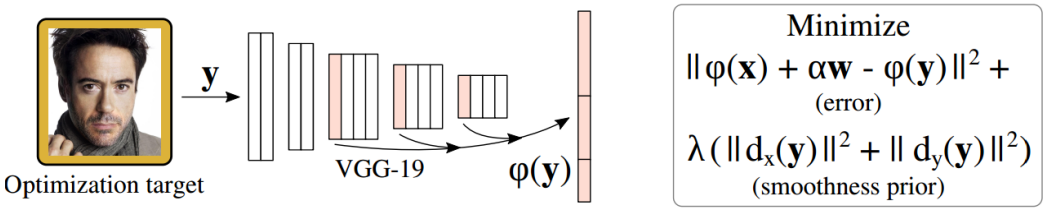


Figure 4.1.2: The process of reconstructing a modified face image in the case of adding facial hair to x. Left: The target image y having the mapping $\varphi(y)$. Right: The objective function of reverse mapping. [UGB+16]

The GD algorithm (refer subsection 2.1.4) is applied to minimize the reconstruction objective. For performing the GD optimization, the gradient of each layer in the deep feature representation is accumulated and composed in an overall derivative using backpropagation method (refer Equation 2.1.4) of the CNN [MV15].

Using the DFI procedure, the authors of [UGB+16] performed several attribute transformations on LFW face dataset (refer subsection 5.1.1). Figure 4.1.3 demonstrates the qualitative results for addition of facial features obtained on two LFW face images. The face alignment method of [ZLLT14] was applied on the input images and were resized to 200 x 200 pixels. The authors rescaled the value of α in Equation 4.1.3

to the value in Equation 4.1.6 for achieving consistent results in different attribute transformations.

$$\alpha = \frac{\gamma}{\frac{1}{d}\widetilde{w}} \tag{4.1.6}$$

where, d is the dimensionality of $\varphi(x)$, $\widetilde{w}$ is the mean squared attribute vector and γ is a constant treated as a hyperparameter. The hyper-parameter values of $K = 100$ and $\gamma = 0.4$ was used for the modifications in Figure 4.1.3.

Figure 4.1.3: The results of changing six different facial attributes on LFW face images using DFI. Input image given to DFI is in the first column. [UGB+16]

As shown in Figure 4.1.4, modifications of facial attributes for high-resolution images of CelebA dataset (refer subsection 5.1.2) was also performed by the authors. For creating source and target sets in DFI, the face attribute classifier associated with the CelebA dataset was used to properly identify the attributes of the images. The face images of the source and target sets were aligned to an input image using the face alignment tool in DLIB [KS14]. One can notice in Figure 4.1.3 that DFI performs photo-realistic changes for a specific attribute while preserving other characteristics of the original image. This is also justified in Figure 4.1.4 since undesired changes are more perceivable at higher resolutions.

Figure 4.1.4: The results of high-resolution face image editing using DFI. Left: Original image. Middle: Adding facial hair. Right: Aging. Modifications done with $\alpha = 4$. [UGB+16]

4.1.4 *DFI Discussion*

As described in the previous section, DFI is a novel approach to change features in face images without involving the efforts of training a specific CNN. The authors demonstrated that an object recognition model such as the VGG network is effective for another entirely distinct task of image content editing. Moreover, they also suggest using this method as a baseline for comparison with the state-of-the-art image transformation methods. Unfortunately, the DFI implementation in [UGB+16] illustrates only the visual quality of the image modification results. Besides, the modifications were generated for fixed hyperparameters K and α in Figure 4.1.3. These hyperparameter values influence the visual aspects of the reconstructed image in DFI. As one can observe in Figure 4.1.5, significant changes in the modified images can be produced by varying the parameter γ (see Equation 4.1.6). When γ is low, the feature modification is indistinguishable from the original image, whereas, the changes are noticeable when γ is steadily increased. Moreover, the appearance of the person in the modified image substantially changes when γ is high. A definitive evaluation of DFI's ability to preserve the identity in the modified face images is therefore beneficial. Additionally, the effect of varying the two hyperparameters values on the identity of the modified images is not completely analyzed. To this end, a necessary description of the face verification approach is given next.

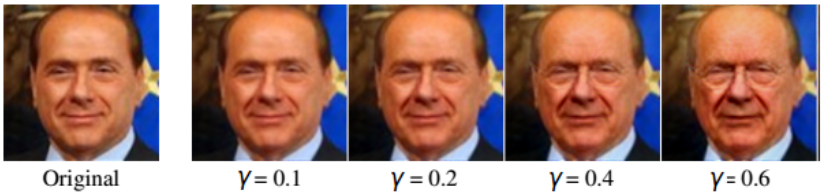


Figure 4.1.5: The DFI results of aging done on a face image by varying γ. [UGB+16]

4.2 DEEP FACE RECOGNITION

Deep Face Recognition (DFR) [PVZ15] is inspired by the performance of the VGG networks in object recognition tasks (refer subsection 3.4.1). The CNN configuration (refer subsection 3.4.2) was thoroughly investigated by the authors of DFR for the tasks of face identification and verification. The network was trained with the VGG face dataset (refer subsection 5.1.3). As mentioned in subsection 3.4.2, training of the network was achieved for two different objectives, namely, for a N-way classification problem and for learning the face embedding using a triplet loss objective.

The objective of learning a face embedding in DFR was accomplished to obtain a better score vector for the task of face verification. The goal of face verification is to confirm whether two face images belong to the same person or not. For this purpose, one utilizes the D-dimensional output of the second fully connected layer, $\phi(l_t)$ (see Equation 3.4.1), as the representation of a face image. Considering two face images l_1 and l_2, one computes the Euclidean distance between their face descriptors $[\phi(l_2), \phi(l_1)]$ as $\|\phi(l_1) - \phi(l_2)\|_2$. If this distance is smaller than a threshold τ, the two face images will portray the same identity. The threshold τ is not uniquely learned by the CNN but calculated separately for a test dataset (refer section 5.2).

4.3 EXTENSION

As discussed in subsection 4.1.4, the authors of DFI provide only the qualitative results in [UGB+16]. One of the objective of this thesis is to establish a quantitative assessment of the DFI process, particularly using this method for manipulating certain attributes in face images. Therefore, the face verification approach of DFR is utilized for evaluating the modified images. The method followed to achieve this task is described in this section.

4.3.1 *Identity verification of modified face images*

For the face verification task of DFI, one considers a face-image pair l_1 and l_2. The DFI method is applied on the image l_1 to produce a modified image l_1' having a desired attribute. The face verification approach in DFR utilizes the Euclidean distance as a similarity measurement to compare the high-dimensional face descriptors. Instead, a better performance for face verification task is achieved by using the *Bray-Curtis* dissimilarity [RGF17]. The Bray-Curtis dissimilarity (BC) is a non-euclidean measure quantifying the similarity between two vectors, which is typically used for ecological data [BC57]. The BC measurement between two vectors is defined in Equation 4.3.1 and lies in the range [0, 1]. A BC value of 1 indicates that the vectors are completely dissimilar to each other whereas a value of 0 implies they are similar. In contrast to the multi-cropping procedure in DFR (refer subsection 3.4.2) that requires a total of 30 face descriptors $\phi(l_t)$, one utilizes a configuration consisting only the center crop and its horizontal flip from a face image after resizing it to the three scales of [256, 348, 512] resulting in 6 face descriptors for a test image. This configuration shows better results when compared to the configuration in DFR for LFW face images [FMR17]. Therefore, the 6 face descriptors are computed and averaged for each image l_1' and l_2. The BC dissimilarity is calculated for the mean face descriptors of l_1' and l_2.

$$BC(u, v) = \frac{\sum_{i=1}^{d} |u_i - v_i|}{\sum_{i=1}^{d} |u_i + v_i|} \qquad u, v \in \mathbb{R}^d \tag{4.3.1}$$

To sum up, the aim is to replicate the DFI method and perform a conclusive evaluation on the results of the image modifications, utilizing the face verification approach as described and to analyze the effect of tuning the hyperparameter values on the face verification result. Furthermore, an additional goal is to apply the DFI method on the face dataset of DFR (refer subsection 5.1.3) for obtaining desired attribute changes using different hyperparameter values and subsequently determine the classification accuracy using the VGG network (refer subsection 3.4.2) for each case. The next chapter gives the description of the datasets, the evaluation metrics and the experiments for achieving these objectives.

5 EXPERIMENTS

The previous chapter described the method of DFI for modifying desired characteristics of a face image. Additionally, the face verification approach of DFR was discussed. Furthermore, based on the discussion in subsection 4.1.4, the method for adopting DFR's face verification to evaluate the DFI method was provided. In this chapter, section 5.1 provides a description of the three datasets used for the evaluation of the DFI method. The evaluation metrics for the face verification and classification tasks are described in section 5.2. Finally, the experiments performed in this thesis and the corresponding analysis of their results are presented in section 5.3.

5.1 DATASETS

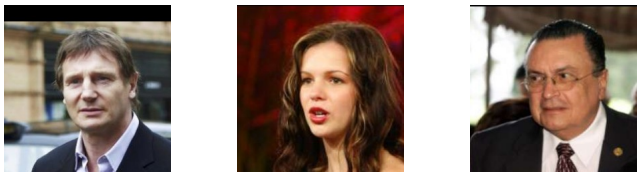
The three datasets used in this thesis are, namely, the LFW face dataset designed prominently for face verification tasks, the CelebA dataset containing high-resolution face images of celebrities having annotations for 40 different attributes, and a large face-dataset of VGG compiled specifically for DFR.

5.1.1 *Labeled faces in the Wild dataset*

Labeled faces in the Wild dataset (LFW) [HRBLM07] is a benchmark dataset used especially for the problem of face verification. This dataset contains 13,233 face images of 5,749 different persons with 73 predicted attribute annotations, see Figure 5.1.1. These images exhibit variability in pose, age, accessories, facial expression, background and illumination, which is suitable for unconstrained face recognition as per the authors. The Viola-Jones face detector was used for face detection and the images were rescaled to 250 x 250 pixels [HRBLM07]. Two unique configurations of the dataset are provided by the authors for training and testing purposes as described next.

5.1.1.1 *Image-Restricted Configuration*

In this configuration, a file containing the information of matched and unmatched pairs of face-images is provided. The file consists of a list of 6000 pairs in total, partitioned into 10 subsets. Each subset has 300 matched and 300 unmatched face-image pairs.



(a) (b) (c)

Figure 5.1.1: Sample face images of LFW dataset. [HRBLM07]

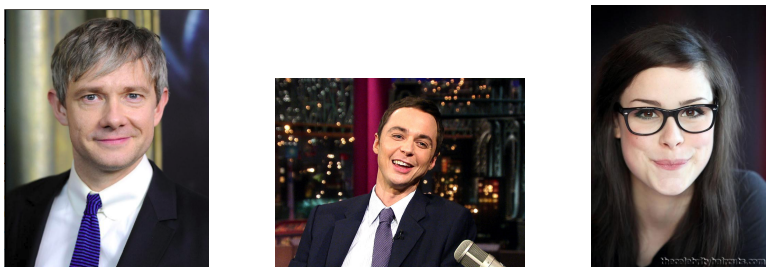
A matched pair implies that the two face-images belong to the same person. Using this configuration, one ensures that the name of a person is not used for inferring equivalency between two face-images, which are not explicitly given in the training set [HRBLM07].

5.1.1.2 *Unrestricted Configuration*

For this configuration, the authors provide a list of identities and a corresponding face-image associated with that identity, rather than image pairs. The list is divided into 10 sets. One can form arbitrary matched or mismatched pairs only using the face-images within each set [HRBLM07].

5.1.2 *CelebFaces Attributes dataset*

CelebFaces Attributes dataset (CelebA) [LLWT14] is a large-scale face attribute dataset. This dataset contains a total of 202,599 celebrity face-images for 10,177 identities (see Figure 5.1.2), where each identity has 20 images on average [LLWT14]. Each image is annotated for 40 facial attributes, which are provided in an attribute file. In the default attribute file, each attribute of an image is in binary format, where 1 indicates the presence of an attribute and -1 implies an absence of it. Moreover, the authors of [LLWT14] provide a deep learning approach for accurately predicting face attribute scores of these images using CNNs. The identity of each image in the CelebA dataset is provided in a list, which can be used to form matched and mismatched pairs. These identities are mutually exclusive from the identities in the LFW dataset.



(a) (b) (c)

Figure 5.1.2: Some high-resolution face images from CelebA dataset. [LLWT14]

5.1.3 *Visual Geometry Group Face dataset*

Visual Geometry Group face dataset (VGG face dataset) [PVZ15] is a large face-image dataset containing 2.6 million images of 2622 unique identities, mainly actors and politicians. The face-images of each identity in this dataset have large variability in pose or background, see Figure 5.1.3. The authors of [PVZ15] followed a five-stage procedure including web searches, automatic filtering of data, and limited manual intervention to efficiently obtain the images labeled with their identities in this dataset. This dataset also has no overlapping with the identities of the LFW dataset. Unlike the LFW and the CelebA datasets, attribute annotations are not provided for the face-images in this dataset. However, the bounding box and pose information for the images are available.

Figure 5.1.3: Sample face images of four identities from VGG face dataset. [PVZ15]

5.2 EVALUATION METRICS

For the purpose of evaluation, two different metrics are calculated depending on the dataset used for obtaining the modified images of DFI. The different metrics used in this thesis are as follows:

(i) *Classification Accuracy* - It is the ratio of number of correctly classified face images to the total number of images in a test setting. The classification accuracy is calculated as:

$$Acc = \frac{t}{n} * 100 \tag{5.2.1}$$

where, 't' is the number of samples correctly classified by the network and 'n' is the total number of samples.

(ii) *Equal Error Rate (EER)* - As mentioned in section 4.2, the objective of face verification is to determine if two face images belong to the same identity. The similarity measurement provides a score for two face descriptors, but it does not sufficiently validate the distances between descriptors of all the identities in a test dataset. The distance measure between face descriptors of the same identity should be smaller than the distance between face descriptors of two different identities. This implies that a threshold τ exists, which separates the face descriptors of the same identity from the descriptors of other identities. Therefore, the similarity measurement is compared to the threshold τ by a verification method. However, when the threshold τ is too high, the verification method wrongly accepts the face descriptors of different persons as having the same identity, known as false acceptance [GER12]. On the other hand, if the threshold is too small, false rejection occurs where face descriptors of the same identity are incorrectly rejected. Therefore, the performance of a verification method depends on the choice of the threshold τ. A reliable metric for evaluating a verification method, proposed by the International Organization for Standardization ISO/IEC $19795-1$, is the Equal Error Rate %ERR [GER12]. This quantifies

the performance of a verification method in terms of the False Acceptance Rate (FAR) and False Rejection Rate (FRR), which are given in Equation 5.2.2.

$$\begin{aligned} FRR &= \frac{\sum_{m \in M} 1}{card(M)}, \quad \text{if} \quad m > \tau \\ FAR &= \frac{\sum_{n \in N} 1}{card(N)}, \quad \text{if} \quad n \leqslant \tau \end{aligned} \tag{5.2.2}$$

where, M is the set of Bray-Curtis distances (see Equation 4.3.1), which works well for high-dimensional vectors [SF15] and is a better similarity metric compared to Euclidean distance for face verification [RGF17], between the descriptors of matched pairs and N is the set of BC distances between the descriptors of mismatched pairs.

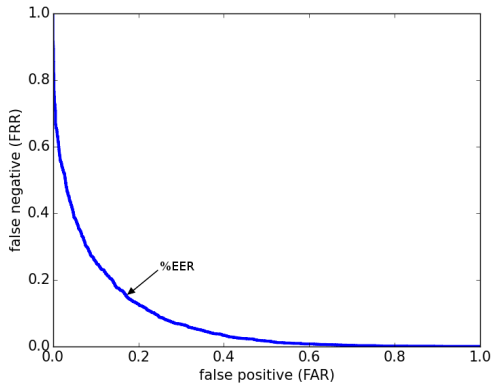


Figure 5.2.1: ROC curve obtained by plotting FAR vs FRR values for each threshold $\tau \in [\tau_{min}, \tau_{max}]$. [GER12]

The %EER is defined as the value where both FAR and FRR are equivalent; that is, the number of false acceptance and the number of false rejections are equal. For computing the %EER, one selects an optimal threshold τ_{opt}, given by Equation 5.2.3, that minimizes the difference between the FAR and the FRR values for all the possible thresholds. Figure 5.2.1 depicts the ROC curve, obtained by plotting FAR versus FRR, used for determining τ_{opt} [GER12].

$$\tau_{opt} = \arg\min_{\tau}(\|FAR(\tau) - FRR(\tau)\|), \forall \tau \in [\tau_{min}, \tau_{max}] \tag{5.2.3}$$

Hence, the %EER is computed as the average of the FAR and the FRR values at τ_{opt}, given by Equation 5.2.4 [GER12]. The advantage of this metric over accuracy is that it is independent of the threshold τ [PVZ15].

$$\%EER = \frac{FAR(\tau_{opt}) + FRR(\tau_{opt})}{2} \tag{5.2.4}$$

5.3 EXPERIMENTAL RESULTS

The experiments performed in this work and their results for the face verification task of DFI (refer subsection 4.3.1) are presented next. The experiments performed for replicating the work of [UGB+16] for modifying desired attributes in face images are given in subsection 5.3.1. Additionally, experiments conducted for analyzing the effect of hyperparameter values on the face verification task are discussed in subsection 5.3.2. Furthermore, the experiments in subsection 5.3.4 investigate the effect of using the DFI method for face-image modifications on the classification accuracy. The evaluation metrics described in the previous section are utilized for these experiments. When the DFI method is applied on the CelebA dataset and the restricted configuration of the LFW dataset, the %EER is computed for the face verification. On the other hand, when it is used for the images in the VGG face dataset, the classification accuracy is calculated using the VGG face classifier (see subsection 3.4.2).

5.3.1 *Replication of the original work*

The face-image modification tasks in [UGB+16] were performed for comparison with the results of VAE/GAN method (refer subsection 3.3.2). Following the approach in [UGB+16], the images in the LFW dataset are utilized for producing desired attribute changes using DFI. The face images are aligned by the method in [ZLLT14] and resized to 200 x 200 pixels. Particularly, the following six facial attributes are selected for the modifications on the LFW dataset: four "positive attributes" - 'Older', 'Smiling', 'Eyeglasses', 'Beard'; and two "negative attributes" - 'Youth', 'No Beard'. These attributes are selected to consider the different changes obtained in the modified image such as adding facial expression, adding of an object and aging. As explained in subsection 4.3.1, the final descriptor of a face image is the average of 6 descriptors $\phi(l_t)$ as this configuration showed better results compared to the configuration of 30 descriptors used in DFR [FMR17]. The descriptors are computed by the CNN for face recognition and verification (see subsection 3.4.2) for the center crop and its

horizontal flip of a face image after resizing it to the three scales of [256, 348, 512]. The Bray-Curtis distance between the face descriptors of the face-image pairs in the restricted configuration of LFW dataset is calculated and the %EER is computed for face verification.

Figure 5.3.1 shows the relation between %EER values calculated for the six attribute changes on the LFW dataset, including the %EER for the original face-images of this configuration (red line). The hyperparameter values of $K = 100$ and $\gamma = 0.4$ are used for the DFI method (refer section 4.1) to replicate the approach of [UGB+16]. The six attribute modifications obtained using this configuration are visualized for an input image in Figure 5.3.2. Among the six attribute modifications, the %EER for the attribute 'Youth' has the least increase of 0.44% from the original %EER of 5.73% while a significant increase is observed for the attribute 'Older' having an error rate of 20.5%.

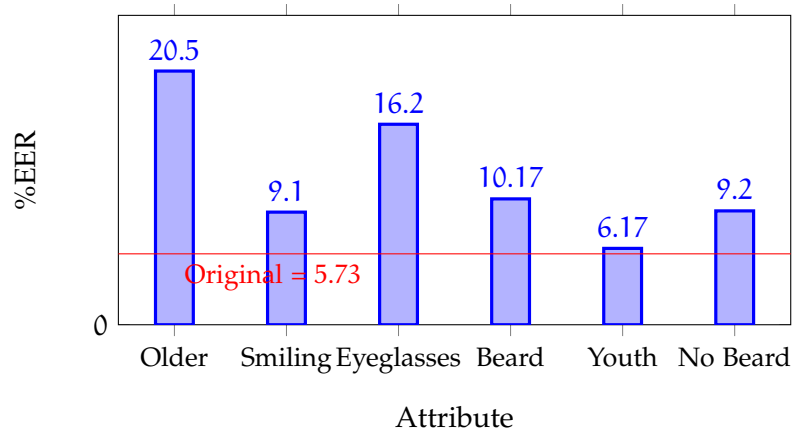


Figure 5.3.1: This plot shows the relation of %EER for six different attributes in the face verification task of DFI using the LFW restricted configuration. The modifications are obtained for hyperparameter values of $K = 100$, $\gamma = 0.4$. [UGB+16]

(a) Input (b) 'Eyeglasses' (c) 'Beard' (d) 'Smile' (e) 'No Beard' (f) 'Older' (g) 'Youth'

Figure 5.3.2: This figure shows all the attribute modifications obtained on an input from the LFW dataset for $\gamma = 0.4$ and $K = 100$.

5.3.2 *Effect of varying the hyperparameters on the face verification of DFI*

The effect of the hyperparameter values of DFI on the face identity verification is analyzed by conducting several experiments on the LFW dataset. The parameter γ controls the amount of interpolation and 'K' indicates the number of images in the source/target sets for DFI. The modifications are obtained for different values of γ and 'K' for each of the six desired attributes. The γ values of [0.2, 0.3, 0.4, 0.5, 0.6] are considered. For each attribute, the value of $K = 100$ is fixed and the modified face images are obtained for each γ value. Then, the %EER is calculated for the face verification task. The results obtained for all of the six attributes are shown in Table 5.3.1. The %EER values of this experiment for the four "positive attributes" are visualized in Figure 5.3.3. One can infer from this graph that the error rate increases correspondingly to the increase in the parameter γ since increasing γ dramatically affects the appearance of the face in an image. This trend is evident for all of the six attribute changes, as seen in Table 5.3.1. The attribute 'Older' exhibits the highest variation of %EER with 9.03% for $\gamma = 0.2$ and 29.87% for $\gamma = 0.6$ while the least variation is for the attribute 'Youth' with 7.7% for $\gamma = 0.2$ and 7.7% for $\gamma = 0.6$ (refer Table 5.3.1).

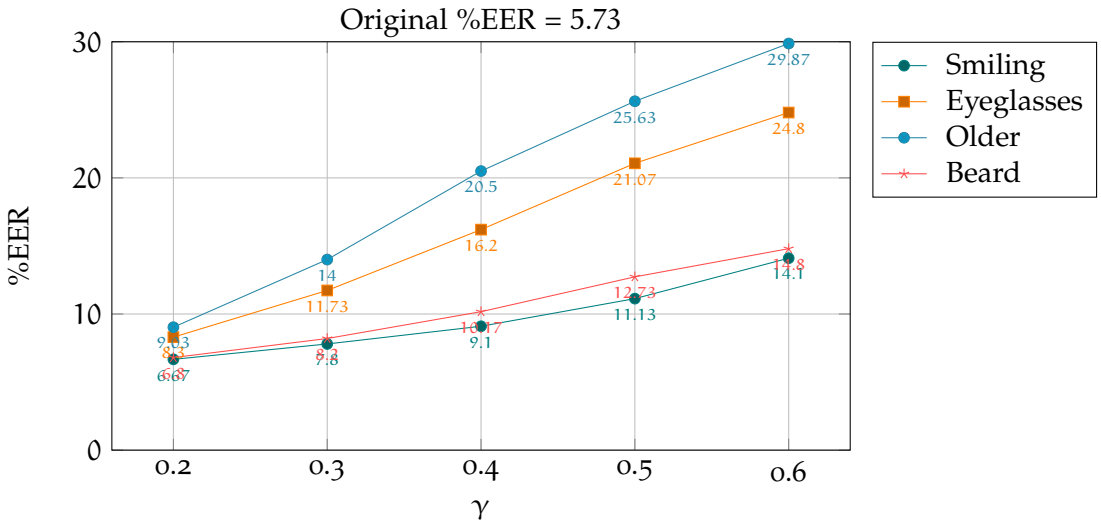


Figure 5.3.3: This plot shows the relation between %EER and γ for four attribute modifications of DFI for fixed parameter $K = 100$.

The effect of increasing the number of images in source/target sets on the face verification task is also examined by performing relevant experiments. The following

three values are considered for 'K': [50, 100, 200]. The modified images are obtained using DFI for each desired attribute and for each value of 'K'. Subsequently, the %EER is computed for each of the obtained modifications. Figure 5.3.4 visualizes the comparison of the %EER values of the six attributes for all the three values of 'K' and a fixed value of $\gamma = 0.4$ as an example (refer Table 5.3.1 for complete results). Similar to the observations obtained by varying the parameter γ (see Figure 5.3.3), it is noticed that the %EER deteriorates as the value of 'K' increases. This observation is consistent for all of the six attribute modifications. This increase of the %EER is due to the usage of a mean vector calculated as an average of source/target set images. Since information from more number of source/target set images are included into the measured attribute vector, one can conclude that the identity in the images gets corrupted and the distances between the face descriptors varies substantially. Moreover, the attribute vector is more likely to point towards other attributes in feature space and not only towards the desired attribute when 'K' is increased .

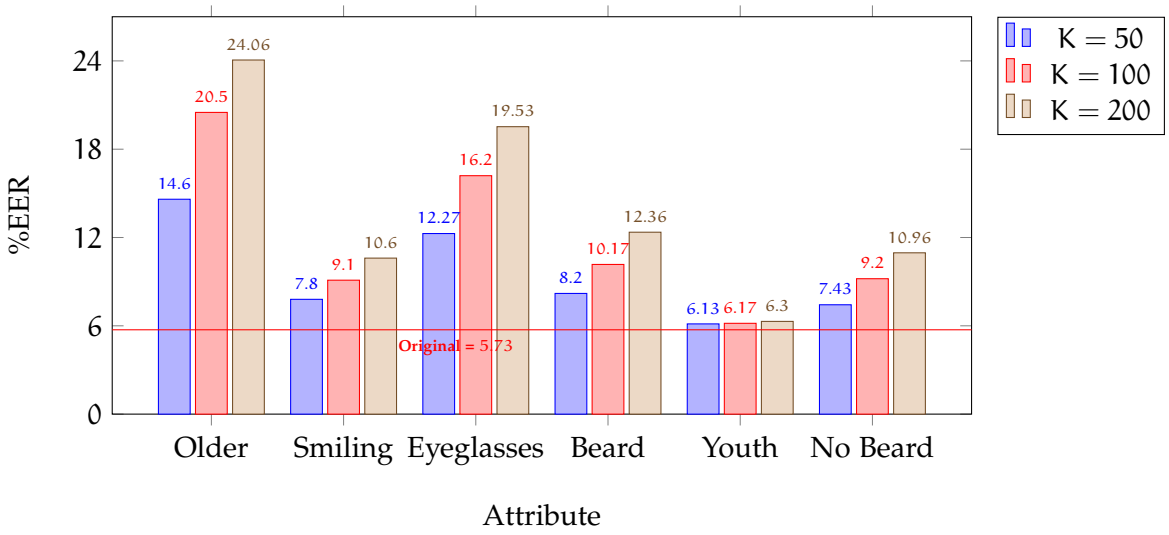
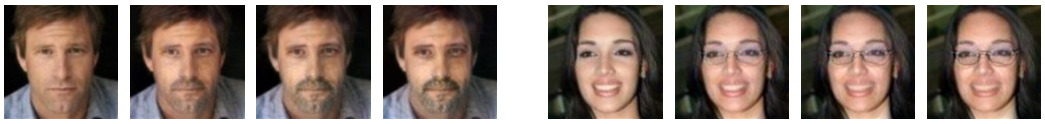


Figure 5.3.4: This graph shows the comparison between the %EER values and the six desired attribute changes for three values of 'K' and fixing $\gamma = 0.4$.

Figure 5.3.6 depicts a plot of %EER vs γ for the desired attribute 'Older', where the modifications are obtained using DFI for each of the three values of 'K'. The lowest %EER is measured for $K = 50$ and the highest for $K = 200$ for all the γ values. It is seen that the %EER for a particular value of 'K' is higher when compared to the %EER for a smaller value of 'K' when γ is fixed. Moreover, when γ is increased, there is a corresponding increase in %EER for all the three values of 'K'. The results of the



(a) Adding 'Beard'. (b) Adding 'Eyeglasses'.

Figure 5.3.5: Two sample attribute modifications obtained on the LFW dataset for $\gamma = 0.4$. For each attribute, first is the input followed by the modified images for $K = 50$, $K = 100$ and $K = 200$ respectively.

face verification task for all the six desired attributes obtained using the DFI method with different values of γ and 'K' are presented in Table 5.3.1. The worst case increase from the original %EER of 5.73% is for the attribute 'Older' for $\gamma = 0.6$ and $K = 200$ having an error rate of 33.36%. Noticeably, the attribute 'Older' has the highest %EER for all the hyperparameter values in comparison with the corresponding values of all the other desired attributes. As shown in Figure 5.3.7, the appearance of the person changes significantly when γ is high, moreover, the face verification and accuracy deteriorate, as seen from Table 5.3.1 and Table 5.3.2. On the contrary, the attribute changes are more pronounced by using a higher value of 'K', as shown in Figure 5.3.5. Therefore, one can consider that the hyperparameter values selected in [UGB+16] are not suitable for producing satisfactory modifications for different desired attributes. Imperfect modifications are produced for face images that do not have a frontal face pose as shown in Figure 5.3.7a, besides, undesired artifacts appear when γ is high. This reaffirms the results in Figure 5.3.6 that %EER steadily increases when both the hyperparameters are assigned larger values.

5.3.3 *Face verification of DFI on CelebA dataset*

To measure the performance of DFI on high-resolution images, an experiment is conducted by applying the method on the face images of CelebA dataset and evaluating the measurements of face verification. For this purpose, the identities provided by the dataset are utilized to create the matched and mismatched pairs of images. Considering the computational effort for high-resolution images, a set of 300 matched and 300 mismatched face-image pairs is prepared. Since aging modifies the appearance of the entire face in an image, the desired attribute 'Older' is selected for this experiment. The modifications are obtained for this attribute using DFI for hyperparameter values of $\alpha = 2.5$ and $K = 100$. The Figure 5.3.8a shows the plot of %EER for the original and the modified images, an example modification result is shown in Figure 5.3.8b. The

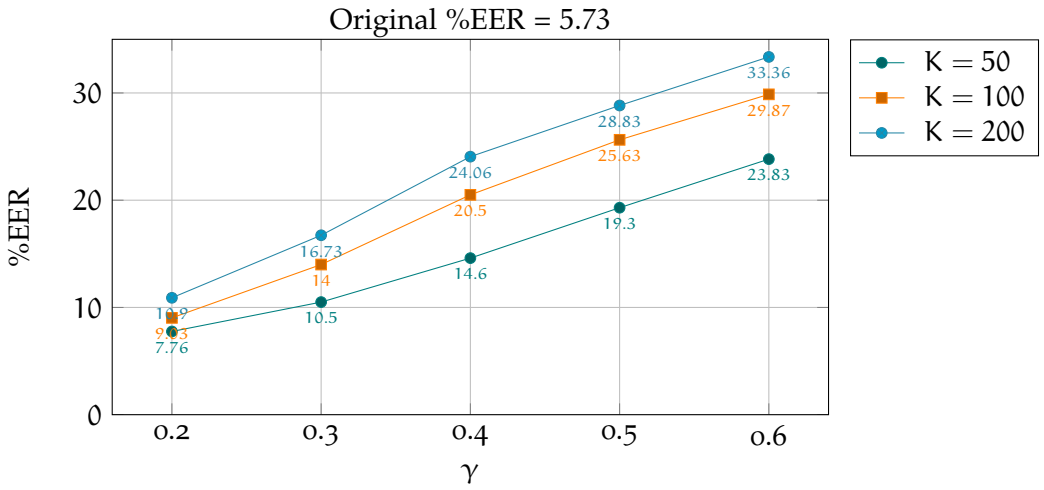
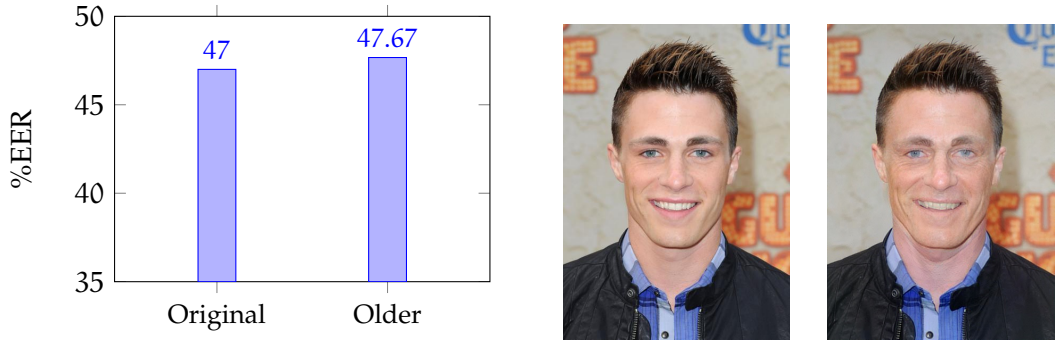


Figure 5.3.6: This graph visualizes the relation between %EER and γ for different values of 'K' for the desired attribute 'Older'.

(a) Attribute 'Older'.

(b) Attribute 'Smiling'.

Figure 5.3.7: This figure shows two sample attribute modifications obtained on the LFW dataset for different γ values and $K = 100$. In each row, first image is the input followed by modified images for $\gamma = [0.2, 0.3, 0.4, 0.6]$ respectively.



(a) This plot shows the %EER of the original and the modified images for attribute 'Older' on the CelebA dataset.

(b) Sample modification obtained for attribute 'Older' (second image) for the input image (left).

Figure 5.3.8: This figure shows the results obtained by applying DFI on CelebA dataset for attribute 'Older' with $\alpha = 2.5$ and $K = 100$.

%EER value of the modified images deteriorates by a value of 0.67% with respect to the original error of 47%. The DFI takes approximately 40 - 50 seconds to produce a modification on one image of the LFW dataset using a single Nvidia GeForce GTX 1080 GPU. This computational time is observed to increase linearly for high-resolution images.

5.3.4 *Effect of DFI on classification accuracy*

The experiments in this section are implemented to investigate the classification accuracy of the VGG network (see subsection 3.4.2) on the modified images of DFI. The attribute transformations are carried out on the face images of VGG face dataset for measuring the accuracy. Specifically, the following three attribute changes are selected: 'Older', 'Youth' and 'Beard', considering the changes in appearance of the entire face by aging and also adding an object, in this case adding beard. Since facial attribute annotations are not available for the images in the VGG dataset, the face images are manually selected for creating the source and the target sets of each attribute, which are required for the DFI method. The modifications are performed using different hyperparameter values of DFI. For this task, the parameter 'K' is varied using four values, specifically, [50, 100, 150, 200], and γ is varied using the following three values: [0.3, 0.4, 0.6]. A test set containing a total of 2937 face-images and at least one face-image per identity is created. The face images are cropped by the bounding

Target Attribute	γ	K		
		K = 50	K = 100	K = 200
Older	0.2	7.76	9.03	10.9
	0.3	10.5	14	16.73
	0.4	14.6	20.5	24.06
	0.5	19.3	25.63	28.83
	0.6	23.83	29.87	33.36
Smiling	0.2	6.27	6.67	6.7
	0.3	6.83	7.8	8.47
	0.4	7.8	9.1	10.6
	0.5	9.6	11.13	13.67
	0.6	11.3	14.1	17.03
Eyeglasses	0.2	7.5	8.3	9.83
	0.3	9.6	11.73	14.93
	0.4	12.27	16.2	19.53
	0.5	16.4	21.07	24.47
	0.6	20.03	24.8	28.6
Beard	0.2	5.9	6.8	7.1
	0.3	6.77	8.2	9.57
	0.4	8.2	10.17	12.36
	0.5	9.37	12.73	—
	0.6	11.43	14.8	18.03
Youth	0.2	6.03	5.7	5.7
	0.3	5.7	5.76	5.8
	0.4	6.13	6.17	6.3
	0.6	7.1	7.7	7.53
No Beard	0.2	6.1	6.63	7.03
	0.3	6.4	7.67	9.3
	0.4	7.43	9.2	10.96
	0.6	10.1	13.53	15.93

Table 5.3.1: This table shows the %EER values obtained for the six desired attributes. The modifications are produced for the restricted configuration of the LFW dataset using the DFI method with different hyperparameter values.

box provided with the dataset and rescaled to 200 x 200 pixels, similar to the image pre-processing followed for the LFW images. After obtaining the modified face images from DFI for the test set, the class score of a modified image is compared to the ground-truth label of the original image to evaluate the accuracy.

Figure 5.3.9 shows the plot for analyzing the effect of the parameter γ on the classification accuracy. An accuracy of 96.69% is measured for the original images in the test set (red line). The three attribute changes are obtained using the DFI method with different values of γ and a specific value of $K = 100$. It is observed that the accuracy decreases with the simultaneous increase in γ for all the three attributes. When γ is increased from 0.3 to 0.6, the attribute 'Older' displays the highest variation in accuracy with approximately 6% whereas the attribute 'Beard' shows the least variation of 1.49%.

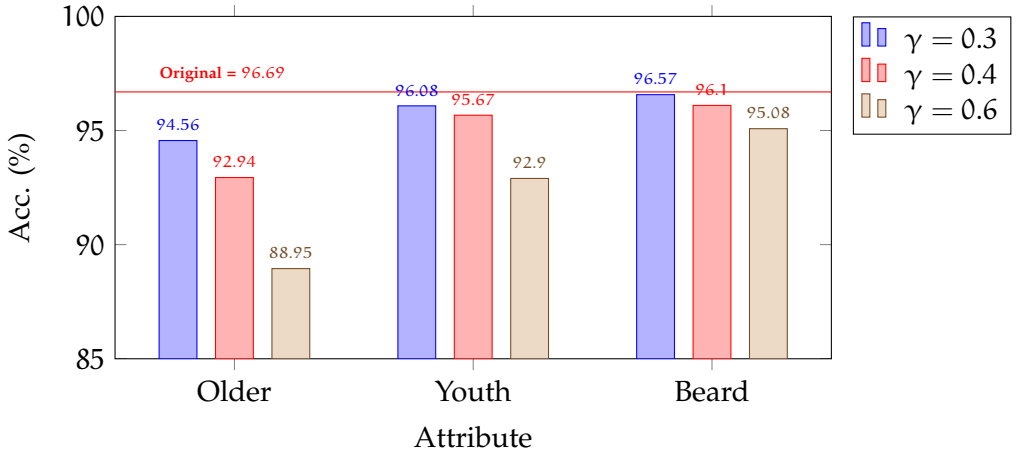


Figure 5.3.9: The plot of classification accuracy vs the desired attributes. The modifications on VGG face dataset are obtained with different values of γ and a fixed value of $K = 100$.

For analyzing the effect of the parameter 'K' on the accuracy, the attribute modifications are obtained for different values of 'K'. The corresponding accuracies for the attribute 'Older' for all the three γ values are plotted in Figure 5.3.10. The modifications obtained for a sample image is visualized in Figure 5.3.11. For a single γ value, the best accuracy is obtained for $K = 50$, however, the accuracy drops when 'K' is increased to a higher value. A significant decrease of more than 11% in accuracy is observed for $\gamma = 0.6$ when the value of 'K' increases from 50 to 200.

The plot of accuracy vs the three attributes for different values of 'K' and fixed value of $\gamma = 0.4$ is shown in Figure 5.3.12. The accuracy typically decreases when

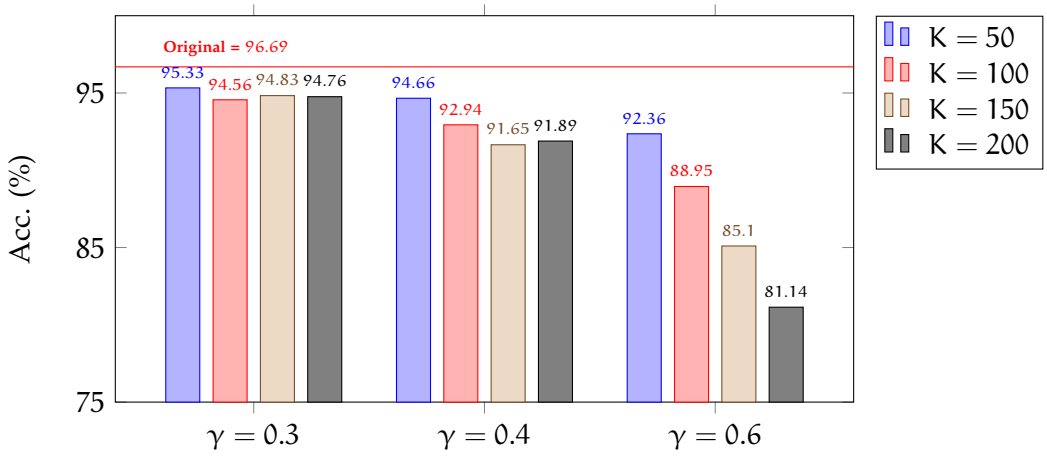


Figure 5.3.10: The plot of classification accuracy vs γ for the attribute 'Older' for different values of 'K'.

Figure 5.3.11: The first image is the input from the VGG face dataset followed by modifications obtained for 'Older' with K = 100 and $\gamma = [0.3, 0.4, 0.6]$.

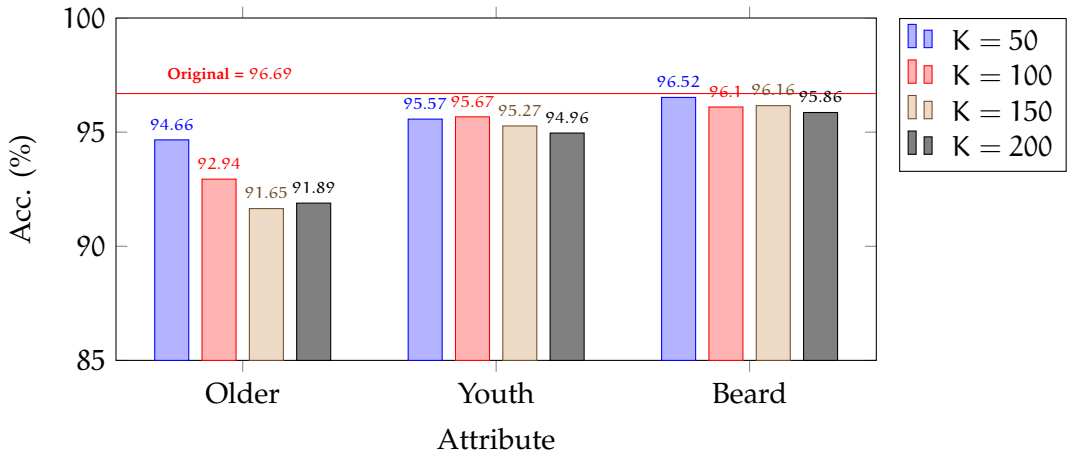


Figure 5.3.12: The graph of classification accuracy vs the attributes for modifications with different 'K' values and $\gamma = 0.4$

Figure 5.3.13: The first image is the input followed by modifications obtained for 'Beard' with $\gamma = 0.6$ and $K = [50, 100, 150, 200]$ respectively.

'K' increases, except a slight increase for the attribute 'Older' ($K = 150$ and $K = 200$) and the attribute 'Youth' ($K = 50$ and $K = 100$). Pointedly, the accuracy for all the attributes decreases within the range of 5% from the original value in Figure 5.3.12. Figure 5.3.13 highlights the influence of the images in source/target sets on the quality of the modification. The images in the LFW dataset are aligned using the facial landmarks provided with the dataset. Since the face images of VGG dataset lack facial landmark annotations, they are cropped using only the bounding box of the face available in the dataset. Therefore, as shown in Figure 5.3.13, improper modifications are obtained by the DFI method, particularly if an attribute such as adding a beard is centered in a specific location. The accuracy measurements for different values of the hyperparameters for all the three attributes are tabulated in Table 5.3.2. Overall, the least accuracy is obtained for the changes in attribute 'Older' for all the values of γ and 'K' when compared to the corresponding accuracy for the other two attributes. There is an absolute deterioration of 15.55% from the original accuracy of 96.69% in case of the attribute 'Older' for $\gamma = 0.6$ and $K = 200$. This is the largest decrease in accuracy when compared to the accuracy measured for all the three attributes and all the hyperparameter values.

Target Attribute	γ	**K**			
		$K = 50$	$K = 100$	$K = 150$	$K = 200$
Older	0.3	95.33	94.56	94.83	94.76
	0.4	94.66	92.94	91.65	91.89
	0.6	92.36	88.95	85.1	81.14
Youth	0.3	96.45	96.08	95.91	95.57
	0.4	95.57	95.67	95.27	94.96
	0.6	94.72	92.9	91.58	90.77
Beard	0.3	96.46	96.57	96.46	96.52
	0.4	96.52	96.1	96.16	95.86
	0.6	95.44	95.08	94.84	94.96

Table 5.3.2: This table shows the classification accuracy (in %) for all the three desired modifications on the VGG face dataset using DFI with different hyperparameter values (Original accuracy 96.69%).

6

CONCLUSION

The task of modifying a face image to obtain believable and realistic changes in facial attributes without significantly affecting the identity and appearance of the person in the image is difficult. The Deep Feature Interpolation method, introduced in [UGB+16], provides a simple mechanism for obtaining several attribute transformations in face images. This approach utilizes a Convolutional neural network (CNN) that is pretrained for object recognition to manipulate an image. In addition to this, the DFI method utilizes only the images in the source and the target sets to produce the modifications on a test image. The target set conatins images having a desired attribute and the source set has images without that particular attribute. The source and target sets are used to compute the attribute vector that isolates the desired attribute in feature space. A hyperparameter controls the number of images in the source/target sets while the amount of attribute modification is controlled by another hyperparameter. These parameters can be varied freely to obtain specific amount of transformation as one desires. The goal of this thesis was to apply DFI for editing facial attributes in images and evaluating the ability of DFI to preserve the identity in the modified images. The transformations were obtained on face images of the LFW and the CelebA datasets for several desired attributes. The evaluation was achieved using the face verification procedure of DFR [PVZ15]. The observations of [FMR17] were considered for calculating a better similarity metric for the face verification. Also, a comprehensive assessment was performed to analyze the effect of the hyperparameters on the attribute modifications. To this purpose, the modifications were obtained by tuning the parameters and the verification metrics were subsequently computed. The results show that a steady increase in error occurs when the parameters are assigned higher values. One can conclude that, when the parameters have large values the attribute vector fails to suitably capture the variance between the images of the source and the target set.

The performance of DFI was further investigated by measuring the classification accuracy, including the evaluation for different hyperparameter values. For measuring the accuracy, the DFI method was applied method on the VGG face dataset for producing the modifications. The results of this experiment display a trend similar to the face verification results, that is, the accuracy decreases for simultaneous increase in hyperparameter values. A remarkable advantage of the DFI method is the scalability to

high-resolution images. Therefore, the face images of the high-resolution dataset were modified using DFI. The face verification performed on these images showed only a slight increase from the original error rate. These observations lead to a conclusion that the DFI method with reasonable hyperparameter values does not extremely degrade the identity in a modified face image. The influence of the hyperparameters and the face image alignment on the visual aspects of the modification was discussed and certain inherent limitations of using DFI were indicated. As future improvement, better modification results can be obtained by adopting a suitable face-alignment method using facial annotations. As pointed out in [UGB+16] and also observed during the testing, the DFI method requires a significant time for producing the modifications for high-resolution images. Hence, there is scope for reduction in computational time taken by DFI for editing high-resolution images, which is beneficial for real time applications. Additionally, the DFI method with appropriate parameters can be used to obtain modified images, which can be utilized as data augmentation technique for fine-tuning of the fully connected and last convolution layers of a face image classifier.

BIBLIOGRAPHY

[AH17] Aghdam, H.H. ; Heravi, E.J.: *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. Springer International Publishing, 2017. – ISBN 9783319575490

[BC57] Bray, J R. ; Curtis, John T.: An ordination of the upland forest communities of southern Wisconsin. In: *Ecological monographs* 27 (1957), Nr. 4, S. 325–349

[BCW+17] Bao, Jianmin ; Chen, Dong ; Wen, Fang ; Li, Houqiang ; Hua, Gang: CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training. In: *CoRR* abs/1703.10155 (2017)

[BKD+08] Bitouk, Dmitri ; Kumar, Neeraj ; Dhillon, Samreen ; Belhumeur, Peter ; Nayar, Shree K.: Face Swapping: Automatically Replacing Faces in Photographs. (2008), S. 39:1–39:8. – ISSN 0730–0301

[BLRW16] Brock, Andrew ; Lim, Theodore ; Ritchie, James M. ; Weston, Nick: Neural Photo Editing with Introspective Adversarial Networks. In: *CoRR* abs/1609.07093 (2016)

[BMDR12] Bengio, Yoshua ; Mesnil, Grégoire ; Dauphin, Yann ; Rifai, Salah: Better Mixing via Deep Representations. In: *CoRR* abs/1207.4404 (2012)

[BV99] Blanz, Volker ; Vetter, Thomas: A Morphable Model for the Synthesis of 3D Faces. In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1999 (SIGGRAPH '99). – ISBN 0–201–48560–5, 187–194

[Chi15] Chinthala, Soumith: *A torch implementation of DCGAN*. https://github.com/soumith/dcgan.torch. Version: 2015

[DHS01] Duda, R.O. ; Hart, P.E. ; Stork, D.G.: *Pattern classification*. Wiley, 2001 (Pattern Classification and Scene Analysis: Pattern Classification). https://books.google.de/books?id=YoxQAAAAMAAJ. – ISBN 9780471056690

[Doe16] Doersch, Carl: Tutorial on variational autoencoders. In: *arXiv preprint arXiv:1606.05908* (2016)

[DV16] Dumoulin, Vincent ; Visin, Francesco: *A guide to convolution arithmetic for deep learning*. http://arxiv.org/abs/1603.07285. Version: 2016. – cite arxiv:1603.07285

[FMR17] Fernando Moya Rueda, Wilmar: *Deep Face Recognition*, Technische Universität Dortmund, Master's thesis, 2017. http://patrec.cs.tu-dortmund.de/pubs/theses/ma_moya.pdf

[GBC16] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[GD17] Gorijala, Mahesh ; Dukkipati, Ambedkar: Image Generation and Editing with Variational Info Generative AdversarialNetworks. In: *CoRR* abs/1701.04568 (2017)

[GEB15] Gatys, Leon A. ; Ecker, Alexander S. ; Bethge, Matthias: A Neural Algorithm of Artistic Style. In: *CoRR* abs/1508.06576 (2015)

[GER12] Giot, Romain ; El-Abed, Mohamad ; Rosenberger, Christophe: Fast computation of the performance evaluation of biometric systems: application to multibiometric. In: *CoRR* abs/1202.5985 (2012)

[Goo17] Goodfellow, Ian J.: NIPS 2016 Tutorial: Generative Adversarial Networks. In: *CoRR* abs/1701.00160 (2017)

[GPAM+14] Goodfellow, Ian ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: Generative Adversarial Nets. Version: 2014. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf. In: Ghahramani, Z. (Hrsg.) ; Welling, M. (Hrsg.) ; Cortes, C. (Hrsg.) ; Lawrence, N. D. (Hrsg.) ; Weinberger, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, 2672–2680

[HRBLM07] Huang, Gary B. ; Ramesh, Manu ; Berg, Tamara ; Learned-Miller, Erik: Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments / University of Massachusetts, Amherst. 2007 (07-49). – Forschungsbericht

[IKS14] I. Kemelmacher-Shlizerman, S. M. S. S. Suwajanakorn S. S. Suwajanakorn: Illumination-Aware Age Progression. (2014)

[IS15] Ioffe, Sergey ; Szegedy, Christian: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, 2015 (ICML'15), S. 448–456

[KAB+16] KARPATHY, ANDREJ ; ABBEEL, PIETER ; BROCKMAN, GREG ; GOODFELLOW, IAN ; KINGMA, DURK ; al., JONATHAN H.: Generative Models. (2016). https://blog.openai.com/generative-models/

[KS14] Kazemi, V. ; Sullivan, J.: One millisecond face alignment with an ensemble of regression trees. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, S. 1867–1874

[KSH12] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. (2012), S. 1097–1105

[KSS12] Kemelmacher-Shlizerman, Ira ; Seitz, Steven M.: Collection flow. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* IEEE, 2012, S. 1792–1799

[KSSGS11] Kemelmacher-Shlizerman, Ira ; Shechtman, Eli ; Garg, Rahul ; Seitz, Steven M.: Exploring Photobios. (2011), 61:1–61:10. http://doi.acm.org/10.1145/2010324.1964956

[KW13] Kingma, Diederik P. ; Welling, Max: Auto-encoding variational bayes. In: *arXiv preprint arXiv:1312.6114* (2013)

[L+09] Liu, Ce u. a.: *Beyond pixels: exploring new representations and applications for motion analysis*, Massachusetts Institute of Technology, Diss., 2009

[LCY13] Lin, Min ; Chen, Qiang ; Yan, Shuicheng: Network In Network. In: *CoRR* abs/1312.4400 (2013)

[LJK16] Ly, Fei-Fei ; Johnson, Justin ; Karpathy, Andrej: Convolutional Neural Networks for Visual Recognition. (2016). http://cs231n.github.io/neural-networks-1/

[LKF10] LeCun, Y. ; Kavukcuoglu, K. ; Farabet, C.: Convolutional networks and applications in vision. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010. – ISSN 0271–4302, S. 253–256

[LLWT14] Liu, Ziwei ; Luo, Ping ; Wang, Xiaogang ; Tang, Xiaoou: Deep Learning Face Attributes in the Wild. In: *CoRR* abs/1411.7766 (2014)

[LSW15] Larsen, Anders Boesen L. ; Sønderby, Søren K. ; Winther, Ole: Autoencoding beyond pixels using a learned similarity metric. In: *CoRR* abs/1512.09300 (2015)

[LTC02] Lanitis, A. ; Taylor, C. J. ; Cootes, T. F.: Toward automatic simulation of aging effects on face images. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), Nr. 4, S. 442–455

[LWM11] Lior Wolf, Tal H. ; Maoz, Itay: Face Recognition in Unconstrained Videos with Matched Background Similarity. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2011)

[MP43] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Dec, Nr. 4, 115–133. http://dx.doi.org/10.1007/BF02478259. – DOI 10.1007/BF02478259. – ISSN 1522–9602

[MP72] Minsky, M.L. ; Papert, S.: *Perceptrons: An Introduction to Computational Geometry*. Mit Press, 1972 https://books.google.de/books?id=Ow1OAQAAIAAJ. – ISBN 9780262130431

[MRML14] M., Mathias ; R., Benenson ; M., Pedersoli ; L., Van G.: Face Detection without Bells and Whistles. In: *Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8692. Springer, Cham* (2014)

[MV15] Mahendran, Aravindh ; Vedaldi, Andrea: Understanding Deep Image Representations by Inverting Them. In: *CoRR* abs/1412.0035 (2015)

[MV16] Mahendran, Aravindh ; Vedaldi, Andrea: Visualizing Deep Convolutional Neural Networks Using Natural Pre-Images. In: *CoRR* abs/1512.02017 (2016)

[PG17] Patterson, J. ; Gibson, A.: *Deep Learning: A Practitioner's Approach*. O'Reilly Media, 2017 https://books.google.de/books?id=qrcuDwAAQBAJ. – ISBN 9781491914236

[PVZ15] Parkhi, Omkar M. ; Vedaldi, Andrea ; Zisserman, Andrew: Deep Face Recognition. In: *BMVC*, BMVA Press, 2015, S. 41.1–41.12

[RDS+15] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li: ImageNet Large Scale Visual Recognition Challenge. In: *International Journal of Computer Vision (IJCV)* 115 (2015), Nr. 3, S. 211–252. http://dx.doi.org/10.1007/s11263-015-0816-y. – DOI 10.1007/s11263–015–0816–y

[RGF17] Rueda, Fernando M. ; Grzeszick, Rene ; Fink, Gernot A.: Neuron Pruning for Compressing Deep Networks using Maxout Architectures. In: *CoRR* abs/1707.06838 (2017)

[RMC15] Radford, Alec ; Metz, Luke ; Chintala, Soumith: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: *CoRR* abs/1511.06434 (2015)

[Roj96] Rojas, Raúl: *Neural Networks: A Systematic Introduction*. New York, NY, USA : Springer-Verlag New York, Inc., 1996. – ISBN 3–540–60505–3

[Ros62] Rosenblatt, F.: *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Spartan Books, 1962 (Report (Cornell Aeronautical Laboratory))

[Sam59] Samuel, A. L.: Some Studies in Machine Learning Using the Game of Checkers. In: *IBM Journal of Research and Development* 3 (1959), July, Nr. 3, S. 210–229. http://dx.doi.org/10.1147/rd.33.0210. – DOI 10.1147/rd.33.0210. – ISSN 0018–8646

[SDBR14] Springenberg, Jost T. ; Dosovitskiy, Alexey ; Brox, Thomas ; Riedmiller, Martin A.: Striving for Simplicity: The All Convolutional Net. In: *CoRR* abs/1412.6806 (2014)

[SF15] Sudholt, Sebastian ; Fink, Gernot A.: A modified isomap approach to manifold learning in word spotting. In: *German Conference on Pattern Recognition* Springer, 2015, S. 529–539

[SKP15] Schroff, Florian ; Kalenichenko, Dmitry ; Philbin, James: FaceNet: A Unified Embedding for Face Recognition and Clustering. In: *CoRR* abs/1503.03832 (2015)

[SMB10] Scherer, Dominik ; Müller, Andreas ; Behnke, Sven: Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In: *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*. Berlin, Heidelberg : Springer-Verlag, 2010 (ICANN'10). – ISBN 3–642–15824–2, 978–3–642–15824–7, S. 92–101

[SYH+17] Shu, Zhixin ; Yumer, Ersin ; Hadap, Sunil ; Sunkavalli, Kalyan ; Shechtman, Eli ; Samaras, Dimitris: Neural Face Editing with Intrinsic Image Disentangling. In: *CoRR* abs/1704.04131 (2017)

[SZ14] Simonyan, Karen ; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *CoRR* abs/1409.1556 (2014)

[UGB+16] Upchurch, Paul ; Gardner, Jacob R. ; Bala, Kavita ; Pless, Robert ; Snavely, Noah ; Weinberger, Kilian Q.: Deep Feature Interpolation for Image Content Changes. In: *CoRR* abs/1611.05507 (2016)

[WG16] Wang, Xiaolong ; Gupta, Abhinav: Generative Image Modeling using Style and Structure Adversarial Networks. In: *CoRR* abs/1603.05631 (2016)

[WLS+04] Wu, Chenyu ; Liu, Ce ; Shum, Heung-Yueng ; Xy, Ying-Qing ; Zhang, Zhengyou: Automatic eyeglasses removal from face images. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (2004), March, Nr. 3, S. 322–336. http://dx.doi.org/10.1109/TPAMI.2004.1262319. – DOI 10.1109/TPAMI.2004.1262319. – ISSN 0162–8828

[YCL+16] Yeh, Raymond A. ; Chen, Chen ; Lim, Teck-Yian ; Hasegawa-Johnson, Mark ; Do, Minh N.: Semantic Image Inpainting with Perceptual and Contextual Losses. In: *CoRR* abs/1607.07539 (2016)

[ZIE16] Zhang, Richard ; Isola, Phillip ; Efros, Alexei A.: Colorful Image Colorization. In: *CoRR* abs/1603.08511 (2016)

[ZLLT14] Zhu, Shizhan ; Li, Cheng ; Loy, Chen C. ; Tang, Xiaoou: Transferring Landmark Annotations for Cross-Dataset Face Alignment. In: *CoRR* abs/1409.0602 (2014)