WROCŁAW UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF ELECTRONICS

FIELD: Computer Science
SPECIALIZATION: Internet Engineering (INE)

MASTER OF SCIENCE THESIS

Research on methods of changing objects in images using Deepfake technology

Badania metod zmiany obiektów na obrazach z wykorzystaniem technologii Deepfake

AUTHOR:
Michał Zendran

SUPERVISOR:
Dr hab. inż. Andrzej Rusiecki, K30W04D03

GRADE:

WROCŁAW 2020

Contents

1 Abstract **3**

2 Introduction **4**
2.1 Motivation 4
2.2 Objective and assumptions 5
2.3 State of the art 5
2.4 Naming conventions and terminology 6

3 Theoretical background **7**
3.1 Artificial neural networks 7
3.2 Convolutional neural networks 8
3.3 Types of machine learning algorithms 10
3.4 Haar feature-based cascade classifier 10

4 Deepfake methods **11**
4.1 Autoencoder 11
4.2 Variational autoencoder 13
4.3 VAE-GAN 14
4.4 CycleGAN 15

5 Datasets **16**
5.1 Dataset description 16
5.2 Data pre-processing 17
5.3 Data augmentation 18

6 Technologies **20**
6.1 Software and Libraries 20
6.2 Hardware 20

7 Networks implementation **21**
7.1 Autoencoder 22
7.2 Variational autoencoder 23
7.3 VAE-GAN 24
7.4 CycleGAN 26

8 Results **28**
8.1 Autoencoder 29
8.1.1 Subject X 30
8.1.2 Subject Y 32
8.2 Variational autoencoder 34

8.2.1 Subject X 35
8.2.2 Subject Y 37
8.3 VAE-GAN 39
8.3.1 Subject X 40
8.3.2 Subject Y 42
8.4 CycleGAN 44
8.4.1 Subject X 45
8.4.2 Subject Y 47

9 Conclusions **49**

Bibliography **50**

List of Figures **54**

Chapter 1

Abstract

To be written at the end of the work.

Chapter 2

Introduction

2.1 Motivation

Machine learning has found many, different applications in the field of image data processing and computer vision. From picture classification to image denoising and resolution enhancement, artificial neural networks has gained the opinion of exceptionally useful tool. But for some time, a new, controversial use-case has been getting more and more attention in both media and research circles. So-called "deepfake" technology has opened doors to many new possibilities of picture generation, but also raised many issues of moral and legal matters.

Deepfake is a technology from the field of machine learning designed to combine and overlay objects in images or videos creating deceptively realistic counterfeits. The name comes from combination of two terms: “deep learning” and “fake”, and has its origins in a Reddit user named “deepfakes”. Initially the term was associated only with face-swapping technology, but with time it was extended to all deep learning implementations of changing objects in images.

Deepfake technology has already found multiple applications such as changing seasons in the landscapes images, transforming horses into zebras or "repainting" images in styles of different artists. But the most controversial and impactful use-case so far is already mentioned face-swapping. In times of overwhelming amount of news it’s getting harder and harder to filter out fake ones from valuable peaces of information. People generally tend not to check sources of information but rather blindly follow hot stories in social medias and television. Such environment combined with capabilities of deepfake gives possibilities of influencing elections by misrepresenting politicians in forged videos to defame or blackmail theme. Another popular use-case of described technology is creating erotic videos by replacing faces of porn actress with faces of well-known celebrities. This application might have less dangerous consequences than influencing world politics but may be hurtful to people that became objects of such act.

Although there are many malicious ways of using deepfake technology it might also be used for good reasons such as helping people to cope with the loss of the loved once or in entertainment filed by de-ageing actors to play younger-selves. Besides, to be able to detect and fight harmful applications of deepfake it might be vital to deeply understand algorithms and techniques behind it. Therefore, conducting research on that part of machine learning field seems to have great meaning in upcoming times.

2.2 Objective and assumptions

This project aims to implement and compare four different methods of changing objects in images with application of artificial neural networks. For sake of this research, human faces were chosen as an object of replacement, as it rises a complex issue of simultaneous color, texture and shape modification.

As there are no numerical methods of measuring the quality of images obtained from deepfake algorithms, the only way of appraising results of methods discussed in this research is visual evaluation. To be able to fairly rate each implemented technique, the same set of images will be used as a learning dataset for all cases. Therefor, effects of all approaches will be visually evaluated and compared with each other, which will result in the final assessment. This rating of methods is the expected outcome of the research.

There are two main factors that will be taken into a consideration during a results evaluation process. First of them is a resemblance of the faked image, to the appearance of the imitated person. The more striking similarity, the better. The other crucial aspect is preservation of original facial expression and pose, as the believable deepfake must capture the source material movements. Resultant of those two factors will be the main feature to be rated. It is assumed that neither of mentioned characteristics should outweigh the other one, but rather, the final effect should be well-balanced composition of both aspects.

2.3 State of the art

Although the deepfake idea is relatively new, a few impressively effective methods has been already developed. The most frequently chosen approaches, in case of generative neural networks are autoencoders [1], variational autoencoders [2] [3], generative adversarial networks [4] [5] and so called Cycle-GAN networks [6] [7] [8]. They are not complete, stand-alone methods for deepfake generation but rather key-parts of deepfake generation technique.

Deepfake videos, almost impossible to distinguish with the naked eye, are created for entertainment purposes on many YouTube channels as for example “Ctrl Shift Face” [9]. Most of them are created with one of the most advanced ready to use, open-source tool for deepfake generation “DeepFaceLab” which allows genereting exceptionally believable deepfakes without having to acquire extensive knowledge in the filed of machine learning [10]. Another, very effective, open-source tool for creating high-quality deepfakes is “FaceSwap” project [11].

While there are many, great articles and papers that elaborately explain all mentioned approaches of generating deepfakes, no comparison of those methods were found. Additionally, results of solutions developed in this particular research cannot be fairly compared with mentioned state-of-the-art methods, algorithms and tools as they were created with purpose different than such comparison.

2.4 Naming conventions and terminology

Below, all abbreviations and naming conventions used in this research are listed and explained:

- Deepfake – name of the deep learning technology of swapping objects in images or an end result generated witch such technology
- ANN – artificial neural network
- CNN – convolutional neural network
- AE – autoencoder
- VAE – variational autoencoder
- GAN – generative adversarial network
- VAE-GAN – variational autoencoder-generative adversarial network
- CycleGAN – cycle generative adversarial network
- Enc – encoder
- Gen – generator
- Dis – discriminator
- Activation function – transfer function
- Conv – convolutional
- GPU – graphics processing unit

Chapter 3

Theoretical background

3.1 Artificial neural networks

An artificial neural network is a computing system inspired by the neural structure of the biological brain and its way of processing information. Such structure is able to "learn" how to solve certain problems or recognize patterns without being pre-programmed with rules how to do it. Artificial neural networks are main tool used in deep learning, which is part of bigger family of machine learning [12].

Figure 3.1: Deep learning belongingness

Fundamental unit of artificial neural network is so-called "artificial neuron" which is a mathematical function modeled after the structure of biological neuron [13]. Such artificial neuron has a number of inputs $\{x_0, x_1, x_2, ..., x_n\}$ with assigned, separated weights $\{w_0, w_1, w_2, ..., w_n\}$ for each of them to be multiplied by. In the simplest case, all product are summed and passed to a transfer function φ. Additionally a bias value b is added to the sum of products which allows to shift the activation function up or down. Then the output y of the neuron is calculated as follows:

$$y = \varphi\left(\sum_{j=0}^{n} x_j w_j\right) \tag{3.1}$$

General idea of artificial neuron model is illustrated in figure 3.2. Typically, neurons are organized into layers which may perform different operations. The output of neurons from one layer is passed to the input of the next layer or might be a part of output vector of whole network. Modern models of artificial neural networks may consist of tens or even hundreds of different layers, on which depends how well given network will cope with certain tasks. Those layers, during process of network training, can adjust theirs weights to produce output values that correctly solve given problem [13] [14].

Figure 3.2: Artificial neuron model

Artificial neural networks have found many, different applications in various fields, in both, academic and business environments. For some time, they are successfully used to solve problems of system identification and control (processes in industrial sector), image classification (medical diagnosis), financial forecasting, face recognition (security), sequence recognition (speech recognition), pattern recognition (data generation) and many more [15].

3.2 Convolutional neural networks

One of the most commonly applied type of deep neural networks, in case of image processing problems, are the convolutional neural networks. CNNs are constructed with convolutional layers that consist of a certain amount of filters, which allow them to effectively "recognize" patterns such as edges, curves, shapes, textures and colors [16]. One of the most important property of such networks is ability to learn spatial hierarchies of patterns. First layer learns to recognize simple features like edges or curves. Filters of following layers can learn more complex patterns made of features from previous layers, like for example circles, squares, eyes, faces and so on [17]. Another significant property of CNNs is the fact that the patterns they learn are translation invariant, which means that pattern learned in one part of the picture can be recognized anywhere else.

Due to its pattern-recognition abilities, convolutional neural networks have found many, different applications in various fields [18]. Depending on the input data dimensionality, CNNs might be applied to time series prediction (weather forecast) or signal identification (ECG signal identification) in case of f one-dimensional data. Two-dimensional CNNs are very effective with image classification (breast cancer tissues classification), object detection (autonomous vehicles) and face recognition (security). In theory, CNNs might be applied to data with any number of dimensions but it is not a popular approach as such data is hard to understand for humans.

The main idea behind the convolutional layer is a mathematical operation called convolution, which is a combination of two functions that produces third function. In case of CNNs, convolution is performed by sliding given filter (in form of matrix), also known as kernel, over input data. At every point of input data, a matrix multiplication is calculated and sums the result onto the feature map [19]. This process is illustrated in figure 3.3, where exemplary filter, applied on input data produces convolved feature (also called feature map).

Figure 3.3: Convolution operation

3.3 Types of machine learning algorithms

Depending on the problem specification and the form of related training data, different algorithms might be used for the process of artificial neural networks training. The most common types of training methods are: supervised, unsupervised, semi-supervised and reinforcement learning [20] [21].

The most popular paradigm for machine learning, supervised learning, is applied in case of well-labeled data and is typically used for classification problems. This approach aims to find a function that maps an input to an output based on example input-output pairs. Thanks to the loss function, that measures the accuracy of the network by comparing its outcome with responding labels, model updates its weights to produce more accurate predictions. A significant disadvantage of this approach is the need for large amount of well-labeled data, necessary for training efficient networks. Another very popular type of training, perfect in case of unlabeled data, is unsupervised learning also known as self-organization. The goal of this approach is to model the underlying structure or distribution in the data without human supervision (in form of labeled data). This learning type is best suited for clustering problems where certain relationships in data occur.

A method that combines supervised and unsupervised learning is called semi-supervised learning. The idea is to pre-train models on small amount of labeled data and later use large amount of unlabeled data to increase accuracy and improve generalization skills of trained networks. Such approach allows to harness information from gathered data without spending a lot of time on labeling all of them.

3.4 Haar feature-based cascade classifier

To be able to extract faces from videos, Haar feature-based cascade classifier was used. This detector, provided by OpenCV library [22], is based on Paul Viola's and Michael Jones's method [23]. This technique involves detecting and marking in the image certain characteristic features of human face, in the form of black and white, adherent rectangles, as illustrated in the figure 3.4. Afterwards, thanks to the Viola-Jones method, out of many weak classifiers (features), a chain of filters is created (cascade), in which single filter is responsible for detecting single feature. If the given part of an image, called window, positively passes through all the filters of the cascade, than such window is considered to contain face.

Figure 3.4: Example of Haar-like features detection

Chapter 4

Deepfake methods

4.1 Autoencoder

Autoencoders are the most basic approach to the problem of deepfake generation. In fact, all mentioned methods, except for CycleGAN, are just different variations of this idea. Autoencoder is a type of artificial neural network that learns to reproduce given input in an unsupervised manner. The goal is to train functions $A : \mathbb{R}^n \rightarrow \mathbb{R}^p$ (encoder) and $B : \mathbb{R}^p \rightarrow \mathbb{R}^n$ (decoder) to satisfy condition given in equation 4.1 as described in [1],

$$arg\,min_{A,B} E[\Delta(x, B \circ A(x))] \tag{4.1}$$

where E – expectation over the distribution of x and Δ – reconstruction loss function, which measures the distance between given input and the output of the decoder. General idea of autoencoder model is illustrated in figure 4.1. Typically, architecture of autoencoder consists not only of input and output layers, as this would result in simple coping pixels from the input to the output of the network, but also contains single or multiple hidden layers in between, with the number of neurons lesser than the number of pixels in the input image. Such structure causes bottleneck effect and creates so-called compressed representation at the output of the encoder part, known also as "feature map" or in case of deepfake "latent face". Such compression causes feature map to preserve only information most relevant for later reconstruction and gets rid of unnecessary data.

Figure 4.1: General idea of autoencoder

Generating deepfakes using autoencoders approach consists of three major steps. Described process is illustrated in figure 4.2. Let us assume that X is a set of face images of person x and Y is a set of face images of person y. Firstly, as shown in figure 4.2a, an encoder is trained to produce feature maps for images of faces from both classes. Simultaneously, a corresponding decoder (further called decoder XY) is trained as a measure of

encoder accuracy. Afterwards, two decoders (further called decoder X and decoder Y) are trained separately to reproduce original images from latent faces generated by pre-trained encoder, as presented in figure 4.2b. Finally, decoders are switched to produce images from one class based on feature maps from the other class, which was illustrated in figure 4.2c.

(a) Training autoencoder

(b) Training decoders X and Y

(c) Generating X from Y and Y from X

Figure 4.2: Three steps of generating deepfakes

4.2 Variational autoencoder

Unlike basic autoencoders, which purpose is to reproduce given input by "memorizing" it, variational autoencoders aim to produce latent vectors (feature maps) that follow a unit Gaussian distribution [2]. Such modification allows to generate new images by sampling a latent vector from the unit Gaussian distribution and pass it as an input of the decoder network. The goal in VAE model training is to optimize the loss function given in equation 4.2 [3]. First component of this function is a reconstruction loss $\mathcal{L}_{rec}$, which, by comparing pixels from image x, encoded to a latent vector $z = Encoder(x) \sim q(z|x)$, and reconstructed image $\bar{x} = Decoder(z) \sim p(x|z)$, measures expected marginal log-likelihood of the observations in x. Second component, so called Kullback–Leibler divergence [24], measures the extent to which one probability distribution differs from another. Applying KL divergence to VAE loss function allows to control distribution of the latent vector z.

$$\mathcal{L}_{vae} = \mathcal{L}_{rec} + \mathcal{L}_{kl} \tag{4.2}$$

$$\mathcal{L}_{rec} = -E_{q(z|x)}[\log p(x|z)] \tag{4.3}$$

$$\mathcal{L}_{kl} = D_{kl}(q(z|x)||p(z|x)) \tag{4.4}$$

Presence of KL divergence in VAE loss function generates problem with applying backpropagation to the encoder network, as decoder randomly samples from latent vector z and backpropagation cannot flow through random node as in figure 4.3a. As a solution, a "reparameterization trick" is introduced [25]. This method maps the output of an encoder to separate vectors, mean μ and standard deviation σ, instead of a z-dimensional latent vector. Such representation allows to calculate z in a following way:

$$z = \mu + \sigma\epsilon \tag{4.5}$$

where ϵ is sampled from a multivariate standard normal distribution ($Normal(0, 1)$) and therefore acts as a stochastic component. Such reparameterization allows to run backpropagation process as now latent vector is not purely random, but consists of deterministic and stochastic nodes as illustrated in figure 4.3b.

(a) Nodes before reparameterization (b) Nodes after reparameterization

Figure 4.3: Reparameterization trick

By applying reparameterization trick to equation 4.4, Kullback–Leibler divergence takes following form [26]:

$$\mathcal{L}_{kl} = -\frac{1}{2}\sum_{i=0}^{z}(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2) \tag{4.6}$$

Generating deepfakes using variational autoencoders approach is based on the same algorithm as autoencoders approach illustrated in figure 4.2. The only difference between those two methods is the way the latent face (latent vector) is obtained.

4.3 VAE-GAN

As the name suggests, variational autoencoder generative adversarial network is a combination of previously mentioned VAEs and GANs. By combining those two approaches it is possible to leverage learned representations to better measure similarities in data space [4]. General idea of this technique is similar to the one described in section 4.2, but instead of VAE's decoder, a GAN generator is used to decode feature maps, as illustrated in figure 4.4.

Figure 4.4: General idea of VAE-GAN

The whole potential of this method comes from the discriminator part. This network, during the training process, learns to distinguish real images (pictures before encoding) from generated ones. On the other side, generator network "tries" to deceive the discriminator by producing better and better fake images, improving the quality of final deepfakes. General idea of GAN training process is ilustrated in figure 4.5. To train VAE-GAN model, three different loss functions, applied to corresponding networks, are required: encoder loss, generator loss and discriminator loss:

$$\mathcal{L}_{Enc} = -E_{q(z|x)}[\log p(x|z)] + D_{kl}(q(z|x)||p(z|x)) \tag{4.7}$$

$$\mathcal{L}_{Gen} = -E_{q(z|x)}[\log p(x|z)] * \gamma - \log(Dis(X)) - \log(1 - Dis(\tilde{X})) \tag{4.8}$$

$$\mathcal{L}_{Dis} = \log(Dis(X)) + \log(1 - Dis(\tilde{X})) \tag{4.9}$$

where $\tilde{X}$ - fake image generated from encoded image X, γ - weight factor to balance the loss function value.

Figure 4.5: General idea of GAN

4.4 CycleGAN

CycleGAN is an approach to the image-to-image translation problems, based on deep learning techniques, which does not require a dataset of paired images. Typically, CycleGAN model's architecture consists of two GAN networks, where one generator ($G : X \rightarrow Y$) is responsible for generating images for one class, and the other ($F : Y \rightarrow X$) for generating images for the second one [27]. The goal of the training process is given in the equation 4.10, as described in [28],

$$G^*, F^* = arg \min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \tag{4.10}$$

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F) \tag{4.11}$$

where X, Y – image domains, D_Y – discriminator for the mapping function G, D_X – discriminator for the mapping function Y, λ – weight factor to balance the loss function value.

General idea of CycleGAN is to have both generators G, F to convert images from one domain to another, in a GAN's manner, described in section 4.3. However, depending only on the GAN's adversarial loss alone do not ensure that images after conversion will maintain important features from its original class. To solve this problem, a cycle consistency loss is applied to the final loss function, as in equation 4.11. It relies on the expectation, that image converted by both generators G, F from one domain to the other, and than back again, should look the same as at the beginning. General idea of this process is illustrated in the figure 4.6.

Figure 4.6: General idea of CycleGAN

CycleGAN networks have proven to be very efficient in many image-to-image translation problems. The most popular applications are:

- changing art styles in images of paintings
- coloring black and white images
- changing the season of the landscape in a photo
- transforming horses into zebras
- transforming apples into oranges

Chapter 5

Datasets

5.1 Dataset description

Creating a high quality, well-balanced dataset is an essential step in the process of deepfake generation. Poorly prepared training set might cause that even great network architectures and state of the art algorithms will produce disappointing results. The most efficient approach to this problem ,in case of face-swapping technology, is obtaining images of targeted people from video recordings, which allows to produce great number of images that cover different facial expressions and head positions. There are several factors that should be taken into a consideration in order to construct proper dataset. First of all, videos of people with at least slight resemblance should be chosen. The most important feature in this matter is skin tone but the more similar appearances, the more deceiving deepfake might be achieved. Another important factor is overall quality of source material. Videos with high resolution and large number of frames per second are best suited for this purpose, but also require more computational power and time to properly train necessary networks. Finally, after a deconstruction of videos into single images, all set should be revised to remove pictures that depict faces which are blurry, deformed or in some ways covered, for example by hand [29].

For sake of this research the "VoxCeleb2" dataset was used. As described in [30], VoxCeleb2 consists of over 1 million utterances of over 6000 celebrities derived from videos from YouTube platform. Source data is diversified in terms of recorded people's genders, ethnicities, ages and accents, but also in terms of videos quality, lighting conditions, stability and lengths. Each recording has a resolution of 224 by 224 pixels and depicts closeup shot of a character's face and shoulders, which is close enough to clearly capture subtle facial expressions.

Figure 5.1: Sample images from VoxCeleb2 dataset

5.2 Data pre-processing

To prepare dataset best suited for purpose of this research, following preparations were made, in accordance with guidelines described in section 5.1. Sample images of pre-processed data are presented in figure 5.2.

1. Two actors (Leonardo DiCaprio and Robert Downey Jr.), further called subject X and subject Y, were chosen as targets of face replacement. This choice was dictated by how well their faces are known and recognizable, which facilitates the final assessment, and by factors mentioned in section 5.1.

2. From the set of all videos of chosen subjects available in "VoxCeleb2" dataset, those which presented subjects in similar age, and had good recording quality, were selected.

3. From each video, every tenth frame was extracted to limit the amount of nearly identical images. Additionally, through Haar feature-based cascade classifiers described in section **??**, the face itself was cut out from each extracted frame to discard unnecessary parts of pictures.

4. Resolutions of all obtained face images varied, therefore data had to be rescaled to a common resolution of 160 by 160 pixels.

5. Final step, was to save obtained images as arrays in a form of a single file, in uncompressed ".npz" format. This allows easy data transferring and simplifies the process of loading data during the training of artificial neural networks.

(a) Sample images of subject X (b) Sample images of subject Y

Figure 5.2: Sample images of pre-processed data

5.3 Data augmentation

While generating results using autoencoders method, a specific relationship was observed. It was noticed, that generated deepfakes look like composition of two faces, one directed to the left and the other to the right. This phenomenon might be seen in figure 5.3a, where first column from the left presents original images, second contains images reconstructed from latent spaces by decoder XY and third presents actual deepfakes obtained by decoding latent faces encoded from subject X, with decoder Y, as explained in section 4.1.

(a) Sight directed to the left (b) Sight directed to the right

Figure 5.3: Models trained without data augmentation

Undoubtedly, deepfakes presented in figure 5.3a are of much worse quality then those generated from the same, but diverted images, presented in figure 5.3b. Such phenomenon occurred, as significant majority of faces extracted from subject Y in created database was directed to the right. That kind of imbalance resulted in models learning, that "proper" reconstruction of subject Y should be directed to the right, regardless of position of original face.

To solve the issue of unbalanced data, so called data augmentation technique was used. This approach assumes that more information might be extracted from collected data, through different augmentation methods [31]. When processing images, following methods might be applied: image inversion, rotation, scaling, zooming, translations or adding Gaussian noise. In case of deepfake generation, significant transformations of original picture would cause, that the generated images not to resemble the originals. Therefore, only one augmentation technique – reflection, was applied. As presented in figure 5.4, networks trained with original data extended by augmented one perform much better when processing images depicting different head positions.

(a) Sight directed to the left

(b) Sight directed to the right

Figure 5.4: Models trained with data augmentation

Chapter 6

Technologies

6.1 Software and Libraries

All scripts and programs necessary for data pre-processing described in section 5.2 were implemented in Python 3.7.6 programming language [32] from Anaconda distribution [33]. This platform provides over 7,500 data science and machine learning packages such as OpenCV (Open Source Computer Vision Library), an open source computer vision and machine learning software library built to provide a common infrastructure for computer vision applications [34], or NumPy, a library that provides a multidimensional arrays and matrices objects, and a large set of high-level mathematical functions to operate on these arrays [35].

Models of artificial neural networks, its training process and validation were implemented on a Google Colaboratory platform that allows to write and execute arbitrary python code through the browser, and is especially well suited to machine learning and data analysis [36]. The most important library used in development process was Tensorflow 2.0, an end-to-end, open source, python-friendly platform designed for complex computations, machine learning and deep learning [37]. Additionally, to simplify networks implementation process, Keras library was used. Keras is an open-source, high-level neural networks API, written in Python and supported in Tensorflow's core library. Created as a higher-level, more user friendly interface to significantly simplify development of deep learning models [38].

6.2 Hardware

Data pre-processing described in section 5.2 were entirely developed on a local computer with Windows 10 operational system, NVIDIA GeForce GTX® 960M GPU, quad core processor Intel® Core™ i7-6700HQ with base frequency 2,60 GHz, 16 GB RAM memory SO-DIMM DDR4 Synchronous with frequency 2133 MHz and 256 GB SSD memory disc SK Hynix SC308.

Google Colaboratory platform described in section 6.1 allows a free access to powerful GPUs such as Nvidia Tesla K80 [39] which is compatible with CUDA technology – a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units. With CUDA, it is possible to significantly speed up computing applications by using the power of GPUs [40].

Chapter 7

Networks implementation

In this chapter, detailed description of each implemented network is presented. All typologies were designed to be as similar as possible, with the only exception for details characteristic for given method. In case of methods incorporating autoencoders approach, size of encoded latent face was set to 200.

Types of layers used in models implementations are: fully connected layer, 2D convolution layer, transposed 2D convolution layer, flatten layer and lambda layer (to apply desired function when constructing customized layer). Activation functions applied to different layers are presented in figure 7.1. Additionally, to improve stability of the training process, batch normalization and instance normalization were applied to selected layers.

(a) Sigmoid (b) Tanh

(c) ReLU (d) LeakyReLU

Figure 7.1: Applied activation functions

7.1 Autoencoder

Figure 7.2: Encoder architecture

Figure 7.3: Decoder architecture

7.2 Variational autoencoder

Figure 7.4: Encoder architecture

Figure 7.5: Decoder architecture

7.3 VAE-GAN

Figure 7.6: Encoder architecture

Figure 7.7: Generator architecture

Figure 7.8: Discriminator architecture

7.4 CycleGAN

Figure 7.9: Generator architecture

Figure 7.10: Residual block architecture

Figure 7.11: Discriminator architecture

Chapter 8

Results

This chapter presents results obtained during the research process. Here, outcome of each implemented technique is illustrated, with a detailed description of its generation method and initial analysis. Additionally, to present wider contexts, details of training process of each artificial neural network are discussed in this section.

To ensure comparability of results obtained from each implemented method, all details related to network topologies and training process, except for those that are specific for analyzed approach, should be unified. It is especially important in case of techniques which incorporate autoencoder idea, as dimensionality of latent space, may have great impact on final results, and therefore should be kept the same for each method. Such assumptions ensure that all differences in results arise from exceptional quality of particular technique.

Additionally, to be able to fairly evaluate effectiveness of each implemented method, for all cases, the same set of images was used to generate final results. Images come from the original dataset and were selected to cover different head positions, facial expressions and lightning conditions.

8.1 Autoencoder

The encoder and decoder XY were trained as long, as there was a noticeable improvement in loss function, which in this particular case was 150 epochs, as shown in figure 8.1. If the training process would be prolonged, overfitting would occur and the encoder would lose the generalization ability. Decoder X and decoder Y were trained far longer, past the point where loss function has no noticeable improvement, which in this case was 500 epochs. During the training process, trained models were saved every 50 epochs, and later, to generate final results, the most effective models were selected.

(a) Loss function

(b) Accuracy

Figure 8.1: Training process of AE encoder and decoder XY

Below, results of trained models, generated for each person are presented. First column from the left contains original images to be encoded into latent vectors. Following columns present faces generated by respective decoders, from obtained feature maps. Actual deepfakes are presented in the last column, while remaining columns act as a frame of reference.

As might be observed in the figures below, autoencoders method generates deepfakes with certain similarity to the original images. Although the head position and overall resemblance are correctly depicted in majority of cases, fake images are slightly blurred and do not portray facial expressions accurately. Additionally, as for example in figure 8.7, in some cases, certain artifacts are present in the pictures.

8.1.1 Subject X

Figure 8.2: Sample 1 from subject X after 350 epochs

Figure 8.3: Sample 2 from subject X after 350 epochs

Figure 8.4: Sample 3 from subject X after 350 epochs

Figure 8.5: Sample 4 from subject X after 350 epochs

8.1.2 Subject Y

Figure 8.6: Sample 1 from subject Y after 350 epochs

Figure 8.7: Sample 2 from subject Y after 350 epochs

Figure 8.8: Sample 3 from subject Y after 350 epochs

Figure 8.9: Sample 4 from subject Y after 350 epochs

8.2 Variational autoencoder

Training process for variational autoencoders method were the same as the one described in senction 8.1. The encoder and decoder XY were trained were trained for 150 epochs, as shown in figure 8.10. Decoder X and decoder Y were trained for 800 epochs and during the training process, trained models were saved every 50 epochs. Later, to generate final results, the most effective models were selected.

(a) Loss function (b) Accuracy

Figure 8.10: Training process of VAE encoder and decoder XY

Results presented below are in the same form as in previous section, generated over the same set of images. Deepfakes obtained from variational autoencoders method are of similar quality as those from section 8.1. One major difference, that arises from unique properties of VAE method, is the absence of artifacts in generated images. Except this improvement, blurriness and lack of correct facial expressions remain.

8.2.1 Subject X

Figure 8.11: Sample 1 from subject X after 800 epochs

Figure 8.12: Sample 2 from subject X after 800 epochs

Figure 8.13: Sample 3 from subject X after 800 epochs

Figure 8.14: Sample 4 from subject X after 800 epochs

8.2.2 Subject Y

Figure 8.15: Sample 1 from subject Y after 600 epochs

Figure 8.16: Sample 2 from subject Y after 600 epochs

Figure 8.17: Sample 3 from subject Y after 600 epochs

Figure 8.18: Sample 4 from subject Y after 600 epochs

8.3 VAE-GAN

Training process for VAE-GANs method were similar as the one described in senction 8.1. The encoder and decoder XY were trained were trained for 200 epochs, as shown in figure 8.19. Decoder X and decoder Y were trained for 3000 epochs and during the training process, trained models were saved every 100 epochs. Later, to generate final results, the most effective models were selected.

Figure 8.19: Training process of VAE-GAN encoder and decoder XY

Results presented below are in the same form as in previous section, generated over the same set of images. Deepfakes obtained from VAE-GAN method are significantly different than those from sections 8.1 and 8.2. First difference is that obtained images are much less blurry and have more details characteristic for imitated person. Although the head position and overall resemblance are correctly depicted in majority of cases, face deformations of certain degree are noticeable.

8.3.1 Subject X

Figure 8.20: Sample 1 from subject X after 2000 epochs

Figure 8.21: Sample 2 from subject X after 2000 epochs

Figure 8.22: Sample 3 from subject X after 2000 epochs

Figure 8.23: Sample 4 from subject X after 2000 epochs

8.3.2 Subject Y

Figure 8.24: Sample 1 from subject Y after 2000 epochs

Figure 8.25: Sample 2 from subject Y after 2000 epochs

Figure 8.26: Sample 3 from subject Y after 2000 epochs

Figure 8.27: Sample 4 from subject Y after 2000 epochs

8.4 CycleGAN

In case of CycleGAN method, the whole structure was trained for 20000 epochs. During the training process, trained models were saved every 500 epochs, and later, to generate final results, the most effective models were selected. As this approach to generating deepfakes is far different from the other three techniques, the decision of optimal training duration was based on how many epochs the same network needs to solve different, well-tested image-to-image translation problem. Therefore, before training models on face-based dataset, CycleGAN networks were trained to convert horses to zebras. To achieve this, a "horse2zebra" dataset was chosen, which consists of approximately 2600 images of horses and zebras [41]. To ensure comparability of training results, images were scaled to the same sizes as pictures from dataset described in section 5.2. After 20000 epochs, the most effective models were chosen to generate results, which are presented in figure 8.28. First column from the left contains original images to be converted into images from opposite domain. Second column presents actual deepfakes, and the last one, contains images of deepfakes converted back to its original domain.

Figure 8.28: Conversion horse to zebra after 12000 epochs

As shown in the figure above, after 12000 epochs, quite successful horse to zebra conversion was possible. Therefore, it was decided that 20000 epochs will be sufficient to train models for effective face-to-face translation. Results presented below are in the same form as in example 8.28, generated over the same set of images as in previous sections. Deepfakes obtained from CycleGAN method are of significantly worse quality then those presented in previous sections. The only correctly depicted aspect is head position. All images are either considerably distorted or barely changed at all and the overall resemblance is trifling.

8.4.1 Subject X

Figure 8.29: Sample 1 from subject X after 13000 epochs

Figure 8.30: Sample 2 from subject X after 13000 epochs

Figure 8.31: Sample 3 from subject X after 13000 epochs

Figure 8.32: Sample 4 from subject X after 13000 epochs

8.4.2 Subject Y

Figure 8.33: Sample 1 from subject Y after 13000 epochs

Figure 8.34: Sample 2 from subject Y after 13000 epochs

Figure 8.35: Sample 3 from subject Y after 13000 epochs

Figure 8.36: Sample 4 from subject Y after 13000 epochs

Chapter 9

Conclusions

Bibliography

[1] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2020.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013.

[3] X. Hou, L. Shen, K. Sun, and G. Qiu, "Deep feature consistent variational autoencoder," 2016.

[4] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," 2015.

[5] J. Kim, M. Kim, H. Kang, and K. Lee, "U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation," 2019.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017.

[7] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville, "Augmented cyclegan: Learning many-to-many mappings from unpaired data," 2018.

[8] N. Moriakov, J. Adler, and J. Teuwen, "Kernel of cyclegan as a principle homogeneous space," 2020.

[9] "Ctrl Shift Face," (Accessed 10.06.2020). [Online]. Available: https://www.youtube.com/channel/UCKpH0CKltc73e4wh0_pgL3g

[10] I. Perov, D. Gao, N. Chervoniy, K. Liu, S. Marangonda, C. Umé, M. Dpfks, C. S. Facenheim, L. RP, J. Jiang, S. Zhang, P. Wu, B. Zhou, and W. Zhang, "Deepfacelab: A simple, flexible and extensible face swapping framework," 2020.

[11] "FaceSwap," (Accessed 10.06.2020). [Online]. Available: https://github.com/deepfakes/faceswap

[12] V. Shah, "Difference between artificial intelligence,machine learning and deep learning," 2017, (Accessed 10.06.2020). [Online]. Available: https://buzzrobot.com/difference-between-artificial-intelligence-machine-learning-and-deep-learning-ccfd779eca7b

[13] K. Suzuki, *Artificial Neural Networks - Methodological Advances and Biomedical Applications*, 04 2011.

[14] B. Mehlig, "Artificial neural networks," 2019.

[15] O. Abiodun, A. Jantan, O. Omolara, K. Dada, N. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, 11 2018.

[16] X. Huang, "Convolutional neural networks in convolution," 2018.

[17] F. Chollet, *Deep Learning with Python*, 1st ed. USA: Manning Publications Co., 2017.

[18] Z. Li, W. Yang, S. Peng, and F. Liu, "A survey of convolutional neural networks: Analysis, applications, and prospects," 2020.

[19] A. Dertat, "Applied deep learning - part 4: Convolutional neural networks," 2017, (Accessed 10.06.2020). [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[20] D. Fumo, "Types of machine learning algorithms you should know," 2017, (Accessed 10.06.2020). [Online]. Available: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

[21] T. O. Ayodele, "Types of machine learning algorithms," in *New Advances in Machine Learning*, Y. Zhang, Ed. Rijeka: IntechOpen, 2010, ch. 3. [Online]. Available: https://doi.org/10.5772/9385

[22] "Face Detection using Haar Cascades," (Accessed 10.06.2020). [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html

[23] P. Viola and M. Jones, "Robust real-time object detection," vol. 57, 01 2001.

[24] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[25] D. Kamath, "Generating new faces with variational autoencoders," 2020, (Accessed 10.06.2020). [Online]. Available: https://towardsdatascience.com/generating-new-faces-with-variational-autoencoders-d13cfcb5f0a8

[26] S. Odaibo, "Tutorial: Deriving the standard variational autoencoder (vae) loss function," 2019.

[27] T. Shen, R. Liu, J. Bai, and Z. Li, ""deep fakes" using generative adversarial networks (gan)," 2018.

[28] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017.

[29] "[GUIDE] - Celebrity faceset/dataset creation - How to create celebrity facesets," (Accessed 10.06.2020). [Online]. Available: https://mrdeepfakes.com/forums/thread-guide-celebrity-faceset-dataset-creation-how-to-create-celebrity-facesets

[30] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[31] C. Shorten and T. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, 12 2019.

[32] D. Kuhlman, *A Python Book: Beginning Python, Advanced Python, and Python Exercises*, 12 2013.

[33] "Anaconda," (Accessed 10.06.2020). [Online]. Available: https://www.anaconda.com/about-us

[34] "OpenCV," (Accessed 10.06.2020). [Online]. Available: https://opencv.org/about/

[35] "NumPy," (Accessed 10.06.2020). [Online]. Available: https://numpy.org/

[36] "Google Colaboratory," (Accessed 10.06.2020). [Online]. Available: https://research.google.com/colaboratory/faq.html

[37] "TensorFlow," (Accessed 10.06.2020). [Online]. Available: https://www.tensorflow.org/

[38] "Keras: The Python Deep Learning library," (Accessed 10.06.2020). [Online]. Available: https://keras.io/

[39] "NVIDIA TESLA K80," (Accessed 10.06.2020). [Online]. Available: https://www.nvidia.com/en-gb/data-center/tesla-k80/

[40] "CUDA," (Accessed 10.06.2020). [Online]. Available: https://developer.nvidia.com/cuda-zone

[41] "horse2zebra," (Accessed 10.06.2020). [Online]. Available: https://www.kaggle.com/arnaud58/horse2zebra

List of Figures

3.1 Deep learning belongingness . . . 7
3.2 Artificial neuron model . . . 8
3.3 Convolution operation . . . 9
3.4 Example of Haar-like features detection . . . 10

4.1 General idea of autoencoder . . . 11
4.2 Three steps of generating deepfakes . . . 12
4.3 Reparameterization trick . . . 13
4.4 General idea of VAE-GAN . . . 14
4.5 General idea of GAN . . . 14
4.6 General idea of CycleGAN . . . 15

5.1 Sample images from VoxCeleb2 dataset . . . 16
5.2 Sample images of pre-processed data . . . 17
5.3 Models trained without data augmentation . . . 18
5.4 Models trained with data augmentation . . . 19

7.1 Applied activation functions . . . 21
7.2 Encoder architecture . . . 22
7.3 Decoder architecture . . . 22
7.4 Encoder architecture . . . 23
7.5 Decoder architecture . . . 23
7.6 Encoder architecture . . . 24
7.7 Generator architecture . . . 24
7.8 Discriminator architecture . . . 25
7.9 Generator architecture . . . 26
7.10 Residual block architecture . . . 26
7.11 Discriminator architecture . . . 27

8.1 Training process of AE encoder and decoder XY . . . 29
8.2 Sample 1 from subject X after 350 epochs . . . 30
8.3 Sample 2 from subject X after 350 epochs . . . 30
8.4 Sample 3 from subject X after 350 epochs . . . 31
8.5 Sample 4 from subject X after 350 epochs . . . 31
8.6 Sample 1 from subject Y after 350 epochs . . . 32
8.7 Sample 2 from subject Y after 350 epochs . . . 32
8.8 Sample 3 from subject Y after 350 epochs . . . 33
8.9 Sample 4 from subject Y after 350 epochs . . . 33
8.10 Training process of VAE encoder and decoder XY . . . 34
8.11 Sample 1 from subject X after 800 epochs . . . 35

8.12 Sample 2 from subject X after 800 epochs 35
8.13 Sample 3 from subject X after 800 epochs 36
8.14 Sample 4 from subject X after 800 epochs 36
8.15 Sample 1 from subject Y after 600 epochs 37
8.16 Sample 2 from subject Y after 600 epochs 37
8.17 Sample 3 from subject Y after 600 epochs 38
8.18 Sample 4 from subject Y after 600 epochs 38
8.19 Training process of VAE-GAN encoder and decoder XY 39
8.20 Sample 1 from subject X after 2000 epochs 40
8.21 Sample 2 from subject X after 2000 epochs 40
8.22 Sample 3 from subject X after 2000 epochs 41
8.23 Sample 4 from subject X after 2000 epochs 41
8.24 Sample 1 from subject Y after 2000 epochs 42
8.25 Sample 2 from subject Y after 2000 epochs 42
8.26 Sample 3 from subject Y after 2000 epochs 43
8.27 Sample 4 from subject Y after 2000 epochs 43
8.28 Conversion horse to zebra after 12000 epochs 44
8.29 Sample 1 from subject X after 13000 epochs 45
8.30 Sample 2 from subject X after 13000 epochs 45
8.31 Sample 3 from subject X after 13000 epochs 46
8.32 Sample 4 from subject X after 13000 epochs 46
8.33 Sample 1 from subject Y after 13000 epochs 47
8.34 Sample 2 from subject Y after 13000 epochs 47
8.35 Sample 3 from subject Y after 13000 epochs 48
8.36 Sample 4 from subject Y after 13000 epochs 48