BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

DEEPFAKE DETECTION IN VIDEO SAMPLES

DETEKCIA DEEPFAKES VO VIDEU

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **JAN KRUMPHOLC**
AUTOR PRÁCE

SUPERVISOR **Ing. TOMÁŠ LAPŠANSKÝ,**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

155131

Institut:	Department of Intelligent Systems (DITS)
Student:	**Krumpholc Jan**
Programme:	Information Technology
Title:	**Deepfake Detection in Video Samples**
Category:	Artificial Intelligence
Academic year:	2023/24

Assignment:

1. Study image and video processing methods using neural networks, study deepfakes and their detection in video footage.
2. Using the knowledge gained, design an algorithm for deepfake detection using neural networks.
3. Implement the proposed solution in Python using frameworks such as Keras or PyTorch.
4. Experimentally validate the solution's effectiveness over different datasets and compare this solution with existing state-of-the-art solutions.

Literature:

- RATHGEB, Christian, Ruben TOLOSANA, Ruben VERA-RODRIGUEZ a Christoph BUSCH, ed. *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Cham: Springer, 2022, 487 s. Advances in Computer Vision and Pattern Recognition. ISBN 978-3030876630. ISSN 2191-6586.
- FIRC Anton, MALINKA Kamil a HANÁČEK Petr. *Creation and detection of malicious synthetic media - a preliminary survey on deepfakes*. In: Sborník příspevků z 54. konference EurOpen.CZ, 28.5.-1.6.2022. Radešín, 2022, s. 125-145. ISBN 978-80-86583-34-1.
- LAPSANSKY, Tomáš. Fake Face Detection in the Digital Images. Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Filip Orság, Ph.D.

Requirements for the semestral defence:
Item 1. of the assignment.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Lapšanský Tomáš, Ing.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	9.5.2024
Approval date:	6.11.2023

Abstract

In last years, we could see increase of internet frauds and forgeries. Starting with easier detectable cases like phishing and fake ads, through social engeneering and disinformation campaigns, and ending with attacks using artificial inteligence: Synthetics media, and especially deepfakes. These attacks are very effective because it's difficult to validate authenticity of deepfake media for basic user, and they are in rise in last few years with availability and effectivity of creation tools for public. This thesis is focused on video deepfakes: What methods are used for their creation, what are their weak points, and mainly, how to find these weaknesses and decide, whenever media is deepfake or not. We will analyze state-of-the-art methods of detecting deepfakes, what are their strengths and weaknesses, and develop possible new methods of detection. In the end we will compare results with modern solutions and evaluate result.

Abstrakt

V posledních letech si můžeme všimnout nárůstu internetových podvodů a podvrhů. Počínaje snadno odhalitelnými případy, jako je phishing a falešné reklamy, přes sociální inženýrství a dezinformační kampaně a konče útoky pomocí umělé inteligence: Syntetická media, a obzvláště deepfakes. Tyto útoky jsou velmi efektivní, protože je obtížné ověřit pravost média pro běžného uživatele, a také díky nárustu dostupnosti a efektivity těchto nástrojů pro veřejnost v posledních letech. Tato bakalářská práce je zaměřena na video deepfakes: Jaké metody se používají k jejich tvorbě, jaké jsou jejich slabé stránky a hlavně, jak tyto slabé stránky najít a rozhodnout, zda je médium deepfake či nikoli. Budeme analyzovat aktuálně nejmodernější metody detekce deepfakes, jaké jsou jejich silné a slabé stránky, a vyvineme možné nové metody detekce. Nakonec porovnáme výsledky s aktuálními řešeními a vyhodnotíme výsledek.

Keywords

deepfake, neural networks, deepfake detection, detection in video

Klíčová slova

deepfake, neuronové sítě, detekce deepfake, detekce ve videu

Reference

KRUMPHOLC, Jan. *Deepfake Detection in Video Samples.* Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Tomáš Lapšanský,

Rozšířený abstrakt

Vytváření a odhalování deepfakes je nekonečný závod. Když se koncem devadesátých let dvacátého století začaly objevovat první pokusy o deepfake, bylo jasné, že bude potřebný nějaký typ detekce deepfakes.

První pokusy o deepfake byly buď snadno zjistitelné lidským okem kvůli nedokonalosti technologie vytvořit spolehlivý deepfake, nebo jejich vytvoření trvalo dlouho, hlavně kvůli nedostatečnému výpočetnímu výkonu tehdejší doby. Ale jak se objevují nové technologie a výpočetní výkon v průběhu let roste, bylo možné vytvořit věrohodnější deepfakes. Zejména díky průlomům v neuronových sítích v poslední dekádě nejenže dokážeme vytvářet stále pokročilejší deepfakes, ale nástroje pro tvorbu deepfakes jsou stále běžnější. Dnes může každý, kdo má přístup k internetu, bez problémů vytvářet deepfake obrázků, zvuku nebo videa.

Díky tomu je detekce deepfakes potřeba více než kdy jindy. Zejména v oblasti video deepfakes, kde většina lidí předpokládá, že video je těžší zfalšovat než obraz nebo zvuk, a to především z toho důvodu, že nedostatečná synchronizace falešných snímků videa a vznik artefaktů usnadní detekci deepfake. Toto pravidlo bohužel platí jen pro starší deepfake modely, kde bylo prioritou vytvoření spolehlivého deepfake hlasu a prohození jen hlavních částí obličeje a úst, zatímco ostatní části obličeje byly málo synchronizované nebo dokonce nebyly synchronizované vůbec. Současné nejmodernější modely však dokážou vytvářet deepfakes s dokonale synchronizovanými snímky zvuku a videa. Některé z nich dokonce dokážou reprodukovat většinu biologických signálů, jako je mrkání očí, synchronizace rtů, mimika svalů atd. Při této úrovni detailů je pro lidského pozorovatele nemožné rozlišit, zda je video deepfake nebo ne.

Deepfake Detection in Video Samples

Declaration

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Lapšanského. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Krumpholc
May 21, 2024

Acknowledgements

I would like to thank my supervisor Ing. Tomas Lapsansky for his guidance with this thesis. I would also like to thank my family and friends, especially Bc. Jiri Veverka for moral support.

Contents

List of Figures

Chapter 1

Introduction

The creation and detection of deepfakes is a never-ending arms race. As soon as first deepfake attempts began to appear in the late nineties of the twentieth century, it was clear that some type of deepfake detection will be needed.

First attempts of deepfakes were either easily detectable by the human observer because of impossibilities of technology to create a reliable deepfake, or took a long time to create, mainly because of insufficient computational power. But as new technologies emerge and computation power increases over the years, it was possible to create more believable deepfakes. Especially thanks to breakthroughs in neural networks in the last decade, not only are we able to create more and more advanced deepfakes, but creation tools for deepfakes have become more and more common. Today, anyone with access to the Internet can create image, audio, or video deepfake without any problem.

Because of that, deepfake detection is now needed more than before. Especially in the field of video deepfakes, where most people assume that video is harder to fake than image or audio, mainly because of the fact that insufficient synchronization of fake video frames and audio will make detection of deepfake easier. This was true for older deepfake models, where the focus was on creating a reliable deepfake voice and swapping the main parts of the face and mouth. However, current state-of-the-art models can create deepfakes with perfectly synchronized audio and video frames. Some of them can even reproduce most of biological signals, such as eye blinking, lip synchronization, muscle mimic, etc. With this level of detail, it is impossible for a human observer to distinguish whether a video is a deepfake or not.

In this thesis, we firstly look Chapter 2 for technical background. That should help us to chose some of the best detection models and compared them with each other. Trying to find out their strengths and weaknesses, in which situation are they best, and try to look inside models and see how they were created and how the process of detection of deepfake works. After understanding each detection model, we will try to modify these models to increase their detection chances or improve their performance without significantly reducing their detection effectiveness.

Chapter 2

Technical background

There are many methods to create or detect deepfake. This chapter will describe the concept of neural networks and their variations. Some of these concepts are used later in later parts of the thesis.

2.1 Neural network

A neural network (NN) is a model inspired by the structure and function of biological neural networks in the brain. It consists of connected units or nodes called artificial neurons, which loosely model the neurons in a brain. By itself, a single neuron is not very powerful; their advantage is in numbers and layers. These neurons are connected by edges, which model the synapses in a brain. In an artificial neural network, a neuron processing unit can represent different objects, such as features, letters, concepts, or some meaningful abstraction pattern. the type of processing unit in the network is divided into three categories: input unit, output unit, and hidden unit [50]. Each artificial neuron receives signals from connected neurons, then processes them, and sends a signal to other connected neurons. the input unit accepts signals and data from the outside world. the output unit realizes the output of the system processing result. the hidden unit is a unit that is located between the input and output units and cannot be observed outside the system [11].

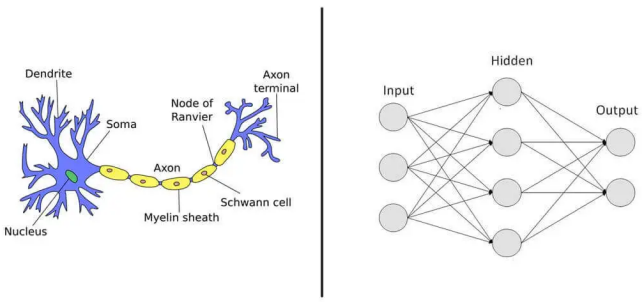

Figure 2.1: Comparison of biological neurons and the artificial neural network [22]

The signal is a real number, and the output of each neuron is computed by using a nonlinear function of the sum of its inputs, called the activation function. the strength of the signal at each connection is determined by a weight that adjusts during the learning process [31].

Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly passing through multiple intermediate layers (hidden layers). If the neural network has more than 2 hidden layers, we call it *Deep neural network*[28].

2.1.1 Loss function

A loss function is a representation that compares the target and predicted output values. It serves as a measure of how well the neural network performs a given task. It quantifies the difference between the actual output of the network and the desired output, which is typically provided during the training phase.

Loss functions are crucial in training neural networks because they guide the optimization process. During training, the goal is to minimize this loss function, which essentially means reducing the disparity between predicted outputs and true targets. This process is usually done using optimization algorithms, which iteratively adjusts the network's parameters to minimize the loss.

2.2 Convolutional neural network

Convolutional neural networks are a specific type of neural network primarily used in the field of pattern recognition within images. the main difference is that neurons are organized into three dimensions in each layer, each dimension representing width, height, and depth [47].

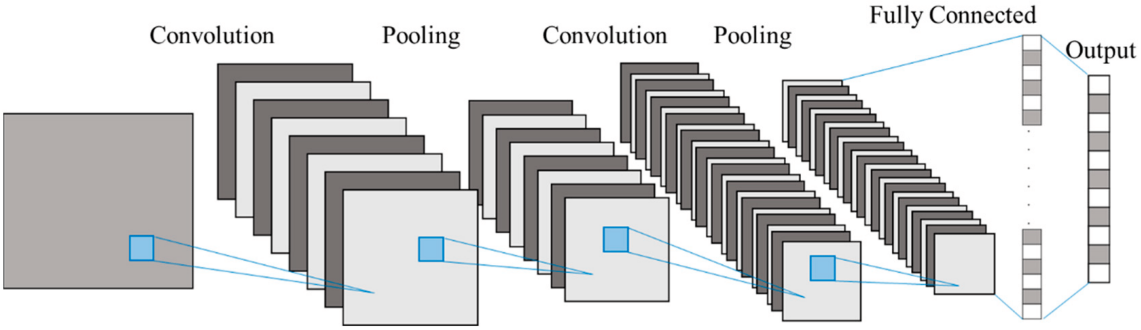

Figure 2.2: Schema of convolutional neural network [30]

CNN consists of an input layer, an output layer, and hidden layers. Each hidden layer consists of several layers that perform operations on the input. These layers are: convolutional layers, pooling layers and fully connected layer [8].

The convolutional layer is the core building block of a CNN, and it is where most computation occurs. It contains a input data, kernel and activation maps. When the data hits a convolutional layer, the layer convolves each kernel across the spatial dimensions of the input to produce a 2D activation map. As the layer moves through the input, the scalar product is calculated for each value in that kernel. From this, the network will learn kernels that 'activate' when they see a specific feature at a given spatial position of the input. These are commonly known as activations [34].

The pooling layers aim to gradually reduce the dimensionality of the representation, thus reducing the number of parameters and the computational complexity of the model. the pooling operation is specified, rather than learned. Two common functions used in the pooling operation are average pooling and maximum pooling. Average pooling calculates the average value for each patch on the feature map, while maximum pooling calculates the maximum value for each patch of the feature map [5]. In two-dimensional feature maps, pooling is typically applied in 2×2 patches of the feature map with a stride of 2. Average pooling involves calculating the average for each patch of the feature map. This means that each 2×2 square of the feature map is sampled down to the average value in the square. In maximum pooling, the results are pooled feature maps that highlight the most present feature in the patch, not the average presence of the feature in the case of average pooling. This has been found to work better in practice than average pooling for computer vision tasks such as image classification.

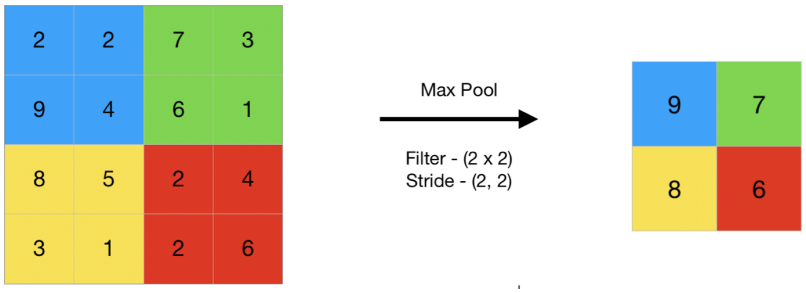

Figure 2.3: Example of max pooling in CNN [20]

The fully-connected layer contains neurons of which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them. This is analogous to the way neurons are arranged in traditional forms of neural networks.

2.3 Generative adversarial network

A generative adversarial network (GAN) is a class of machine learning frameworks to approach generative AI. the concept was initially developed by Ian Goodfellow and his colleagues in June 2014. In a GAN, two neural networks compete with each other in the form of a zero-sum game, where one's gain is the other's loss.

As stated by Goodfellow et al. [16], the adversarial modeling framework is most straightforward to apply when the models are both multi-layer perceptrons.

The basic idea of GANs is to set up a game between two players. One of them is called the generator. the generator creates samples that are intended to come from the same distribution as the training data. the other player is the discriminator. the discriminator examines the samples to determine whether they are real or fake. the discriminator learns using traditional supervised learning techniques, dividing the inputs into two classes: real or fake. the generator is trained to fool the discriminator. To succeed in this game, the generator network must learn to create samples that are drawn from the same distribution as the training data.

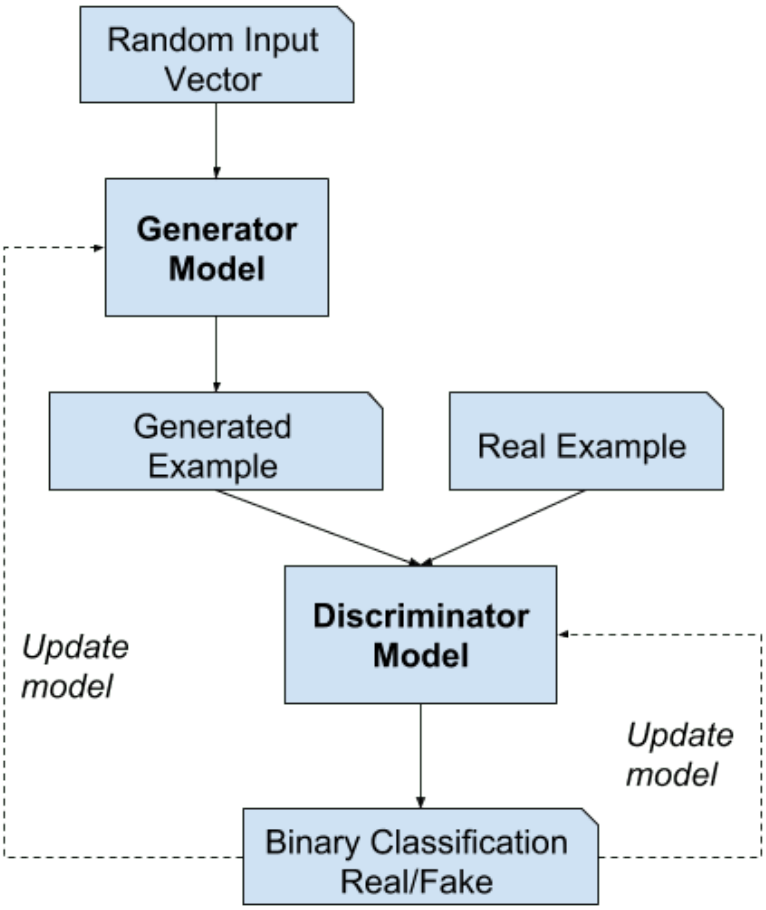

Figure 2.4: Example of GAN architecture [4]

The two players in the game are represented by two functions, each of which is differentiable both with respect to its inputs and with respect to its parameters. the discriminator is a function D that takes x as input and uses $\theta^{(D)}$ as parameters. the generator is defined by a function G that takes z as input and uses $\theta^{(G)}$ as parameters [15].

Both players have cost functions that are defined in terms of the parameters of both players. the discriminator wishes to minimize $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ and must do so while controlling only $\theta^{(D)}$. the generator wishes to minimize $J^{(G)}(\theta^{(D)}, \theta^{(G)})$ and must do so while controlling only $\theta^{(G)}$. Because each player's cost depends on the other player's parameters, but each player cannot control the other player's parameters, this scenario is most straightforward to describe as a game rather than as an optimization problem. the solution to an optimization problem is a (local) minimum. the solution to a game is a Nash equilibrium. If we use the terminology of local differential Nash equilibria as stated by Ratliff et al. [35], in this context, a Nash equilibrium is a tuple $(\theta^{(D)}, \theta^{(G)})$ that is a local minimum of $J^{(D)}$ with respect to $\theta^{(D)}$ and a local minimum of $J^{(G)}$ with respect to $\theta^{(G)}$. Some of the most popular GAN-based deepfake generation methods include FaceSwap [26] and Face2Face [46].

2.4 Variational autoencoder

A variational autoencoder (VAE) is a generative model with a prior and a noise distribution, respectively. It was introduced by Diederik P. Kingma and Max Welling [24]. VAE combine two types of neural networks, much like GAN. However, they combine two distinct kinds of neural networks that operate differently. In the case of VAE, one network finds better ways to encode raw data in a latent space, while the second, the decoder, finds better ways of transforming these latent representations into new content [27]. These encoder–decoder pairs are used to decompose and recompose two distinct faces. By swapping the decoders, it is possible to transform one face into the other, resulting in a credible output [10]. The key innovation of VAE compared to the classical autoencoder is a new probabilistic model that helps generate new content that is similar but different from the original content. In VAE, the intermediate layer provides a way to represent the data in a probability field that enables the layer to store more varieties and with greater precision. For example, it can represent faces or images of numerical digits with smoother features [27].

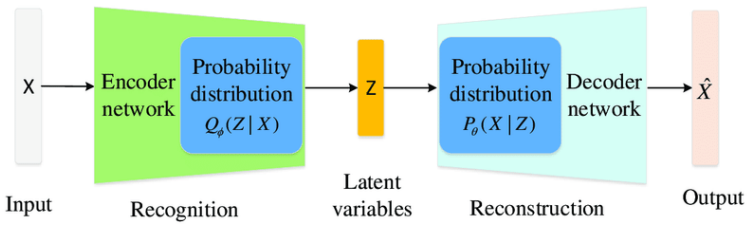

Figure 2.5: Example of VAE architecture [51]

2.5 Diffusion models

Diffusion models, also known as probabilistic diffusion models or score-based generative models, are a class of latent variable generative models. a diffusion model consists of three major components: the forward process, the reverse process, and the sampling procedure. the goal of diffusion models is to learn a diffusion process that generates a probability distribution for a given dataset from which we can then sample new images. They learn the latent structure of a dataset by modeling the way in which data points diffuse through their latent space [40]. Diffusion models can be applied to a variety of tasks, including image noise reduction, in-painting, super-resolution, and image generation. This typically involves training a neural network to sequentially reduce noise images blurred with Gaussian noise. the model is trained to reverse the process of adding noise to an image. After training to convergence, it can be used for image generation by starting with an image composed of random noise for the network to iteratively reduce noise. Diffusion models are typically formulated as Markov chains and trained using variational inference. Examples of generic diffusion modeling frameworks used in computer vision are probabilistic diffusion denoising models, noise-conditioned score networks, and stochastic differential equations [19].

Chapter 3

Deepfake

A deepfake is an audio-visual content, created by artificial intelligence with a neural network, that is authentic to the human observer. It's name is combination of *deep learning*, which is class of machine-learning algorithm, and *fake*[33].

Deepfakes have many forms. Starting with simpler ones, like image deepfake and audio deepfake, and continuing to more complex forms, like video deepfake and real-time deepfakes.

3.1 Danger of video deepfakes

Several internet enthusiasts have posted several live performance films that replace the faces of regular people with the faces of synthetic stars, which causes a topic to be debated. There are raising concerns about video deepfakes because it was widely assumed that videos were reliable and that they could even be used as video evidence in multimedia forensics. In the digital age, deepfake video technology has posed a threat to public confidence. Thanks to the rapid rise in the availability of open-source datasets, significant advancements in the study of topics like GAN and VAE, and significant technological developments in the field of high-speed computing, creating and manipulating fake videos has become a relatively simple task in terms of the cost of creating a manipulated sequence. By superimposing a politician's face over the face of a target actor, a deepfake film might be used to promote false political propaganda. In addition, it could be used to embarrass, shame, or cause a schism among major political institutions, jeopardizing national peace by disrupting the delicate balance of peace between people, governments and nations around the world [18].

3.2 Creation of deepfake

As stated above in Chapter 3 , deepfakes are created by a neural network. They are usually created with specialized types of neural network, especially a convolutional neural network (CNN), variational autoencoders (VAE), or a generative adversarial network (GAN). Most state-of-the-art models for creating deepfakes don't use just basic properties of listed neural types of neural networks, but they incorporate additional layers or even whole additional neural networks to help with authenticity, realism etc.

3.3 State-of-the-art models for creation of video deepfake

There are countless models for creating deepfakes on the Internet. But many of them (especially the ones available as web pages) are flawed. the most common problem is the creation of artifacts. Artifacts are flaws of the deepfake model created due to lack of information or context. In image deepfakes, this can show as distorted or even missing parts of face, wrong color tone. In video deepfakes, artifacts are mostly shown as flickering and jitters. There are many causes why artifacts are created. Common reason is usually insufficiently trained model, bad model architecture, lack of training data, insufficient context from original image etc.

But there are exceptions. Some of the models are so well trained, so well fine-tuned that it is almost impossible for human observer to detect whenever image or video is fake or not. Some of these state-of-the-art models are *FaceSwap*, *Face2Face* and *DeepFaceLab*.

3.3.1 FaceSwap

FaceSwap is a deepfake model that was developed by Iryna Korshunova et al. [26] from TDLab at Ghent University. This model specializes in preserving lighting conditions during face swapping by introducing an additional term to the objective function, which penalizes changes in illumination. Additionally, unlike previous approaches that typically use a single style image, this model employs a multi-image style loss. This allows the model to capture a broader range of describing a style rather than relying on a single reference point, leading to more flexible and nuanced style transfer.

The model computes the style loss by extracting patches from both the input image and multiple style images. Instead of using a single style image for comparison, the model considers a set of style images. These style images describe the identity the model aims to match during face swapping. the selection of style images is customized for each input image based on factors such as pose and expression similarity.

For every patch from the input image, the model finds the best matching patch among all patches extracted from multiple style images. This is achieved by selecting the nearest neighbor patch from each style image using a distance metric, such as the cosine distance, and considering factors like pose and expression similarity.

The model then aggregates the style loss contributions from all selected style images using an aggregation method. the weights may be determined based on factors like the similarity between the input image and each style image.

Another improvement of this model is the solution to the challenge of preserving lighting conditions during face swapping. This is done by introducing an additional term to the objective function, which penalizes changes in illumination.

The model employs a separate Siamese CNN [9] to handle lighting sensitivity. This CNN is trained to discriminate between pairs of images with either equal or different illumination conditions, assuming equal poses. the network learns to distinguish between images based on their lighting characteristics. the siamese CNN is trained using pairs of images from the *Extended Yale Face Database B* [14], which contains grayscale portraits of subjects in various poses and lighting conditions. By providing pairs of images with different lighting conditions but similar poses, the separate network learns to differentiate between images based on their illumination.

The separate CNN extracts feature representations from the images in its last layer. These feature representations capture information about lighting conditions, allowing the network to discriminate between images with different illumination.

The feature representation extracted by the Siamese CNN serves as a basis for comparing illumination conditions between images. the model compares feature representations of the input and generated images to determine if there are significant changes in lighting, and adjusts the generated images accordingly.

In result, by integrating the Siamese CNN and the light loss term into the training process, the model learns to preserve lighting conditions during face swapping.

The model introduces an additional loss term, called *light loss* [26], to the overall objective function. This loss penalizes changes in illumination between the input image and the generated image. By incorporating this loss term into the optimization process, the model encourages the generated images to maintain consistent lighting conditions with the input images.

Figure 3.1: a) the input image. b) face swapped using FaceSwap. c) manual face swap [26]

3.3.2 Face2Face

Face2Face is a deepfake model developed by a team of researchers primarily from Stanford University and the University of Erlangen-Nuremberg. It represents an advancement in the synthesis of facial expressions and movements in target video footage.

At its core, Face2Face employs a technique known as facial reenactment [7], where it transfers the facial expressions of a source actor to a target actor in a given video. Unlike traditional deep-fake methods that rely solely on static images or limited video sequences, Face2Face analyzes and manipulates facial movements in real-time, resulting in deepfake videos in seconds instead of minutes or hours for other deep-fake models [46].

The process of real-time facial reenactment begins with reconstructing the shape identity of the target actor using a global non-rigid model-based bundling approach. This approach resolves geometric ambiguities common to monocular reconstruction by performing a global preprocess on a set of training frames. By capturing the identity of the target actor, the system is prepared to transfer facial expressions accurately.

At runtime, both the expressions of the source and target actor's video are tracked using a dense analysis-by-synthesis approach based on a statistical facial prior. This tracking ensures that the system can accurately capture the facial expressions of both actors in real-time, even without relying on depth data.

To transfer expressions from the source to the target actor in real-time, the system uses a transfer function that applies deformation transfer directly in the used low-dimensional

expression space. This process ensures that the facial expressions captured from the source actor are seamlessly transferred to the target actor's video.

After transferring the expressions, the system re-renders the target's face with the transferred expression coefficients. This step involves compositing the re-rendered face with the target's video background, taking into consideration the estimated environment lighting. the goal is to generate a realistic image of the target actor with the transferred facial expressions.

For mouth synthesis, a new image-based mouth synthesis approach is introduced to generate a realistic mouth interior for the target actor. This involves retrieving and warping the best matching mouth shapes from the sample sequence of the target actor. the system maintains the appearance of the target mouth shape, leading to more realistic results compared to methods that directly copy the source mouth region or use generic proxies.The similarity metric is based on geometric and photometric features [46].

One of the primary focuses of Face2Face is its ability to transfer facial expressions from a source to a target with high realism. While other deepfake models primarily focus on identity swapping or face reenactment, Face2Face excels at replicating nuanced expressions with high accuracy.

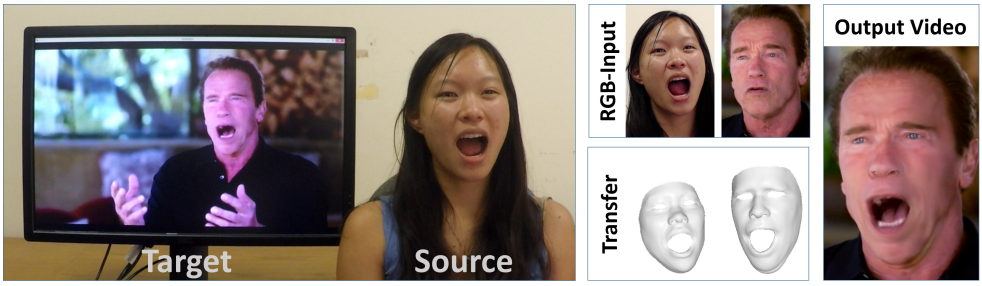

Figure 3.2: Proposed online reenactment setup [46]

3.3.3 DeepFaceLab

The success of DeepFaceLab (DFL) is created by incorporating ideas from previous models into a design that balances speed and ease of use and improvements in computer vision in face recognition, alignment, reconstruction, segmentation, etc.

DFL provides a set of workflow which form the flexible pipeline. DFL can abstract the pipeline into three main phases: extraction, alignment, and merging. These three parts are presented sequentially. In addition, it should be noted that DFL falls into a typical one-to-one face-swapping paradigm, which means that there are only two kinds of data: source and destination.

Face detection is the first step in extraction, which is to find the target face in the given data. DFL regards S3FD as its default face detector. The second step is face alignment. DFL provides two canonical types of facial landmark extraction algorithms to solve this: 2DFAN [6]: Heatmap-based facial landmark algorithm for faces with standard pose and PRNet [13]: With 3D face prior information for faces with large Euler angle (yaw, pitch, roll). After face alignment, a data folder with standard front/side-view faces is obtained. Then TernausNet [21], fine-grained face segmentation network, is used, through which a face could be segmented exactly [29].

Chapter 4

Detection of video deepfake

One of the most difficult problems is detecting video deepfakes. To create a very advanced face-swap movie using deepfake, only one GPU and a vast amount of training data are required. Deep learning approaches, as opposed to standard picture forensic techniques, incorporate feature extraction and feature classification into a network structure and achieve an end-to-end effective automatic feature learning classification methodology.

There are multiple methods of detecting video deepfakes. Some of the detection methods are as follows. Biological signals (such as monitoring eye blinking, lip sync detection), detection of face warping artifacts, facial expression-based detection, etc. [37]. Most detection methods are based on the detection of imperfections in created media.

In video deepfake detection, traditional techniques and approaches, like deep learning, have been the most widely used. In addition, living-body recognition in face recognition and multimedia forensics can provide options. Deep learning is currently thought to be capable of building realistic phony faces, detect and investigate invisible forgery traces, and recognize forgery films. Deep learning approaches, in contrast to standard picture forensic techniques, incorporate feature extraction and feature classification into a network structure and achieve an end-to-end effective automatic feature learning classification methodology [53].

4.1 State-of-the-art models for the detection of deepfake

In this chapter we will study the insides of deepfake detection models. We will look inside various models to see how they were created, what are their improvements over previous models, and try to find out if they can be further improved.

4.1.1 Inception

The Inception model is one of the main milestones in the development of CNN classifiers. Before its creation, most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance. Because of that, they are computationally expensive and they are prone to over-fitting. Another problem was that the important parts in the image can have extreme variations in size. To mitigate these problems, Szegedy et al. [41] decided to change this approach, and instead of using one „universal kernel“, they decided to use multiple kernels with different sizes of filters. the result of this is changes is *Inception module*. It performs convolution on an input with 3 different sizes of filters: 1x1, 3x3, 5x5. Additionally, max-pooling is also performed. the outputs are concatenated and sent

to the next inception module. This change made the network a bit 'wider' rather than 'deeper'.

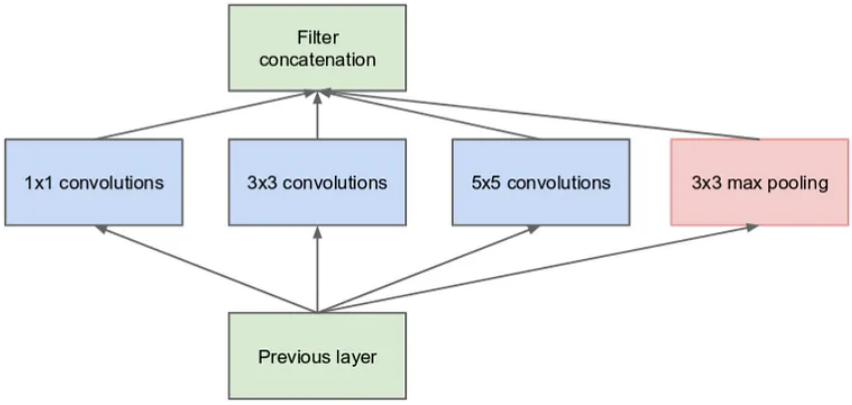

Figure 4.1: 'Naive' Inception module [41]

As stated earlier, deep neural networks are computationally expensive. To make it cheaper, the authors limit the number of input channels by adding an extra 1x1 convolution before the 3x3 and 5x5 convolutions. Although adding an extra operation may seem counterintuitive, 1x1 convolutions are much cheaper than 5x5 convolutions, and the reduced number of input channels also helps. However, the 1x1 convolution is introduced after the max pooling layer, rather than before.

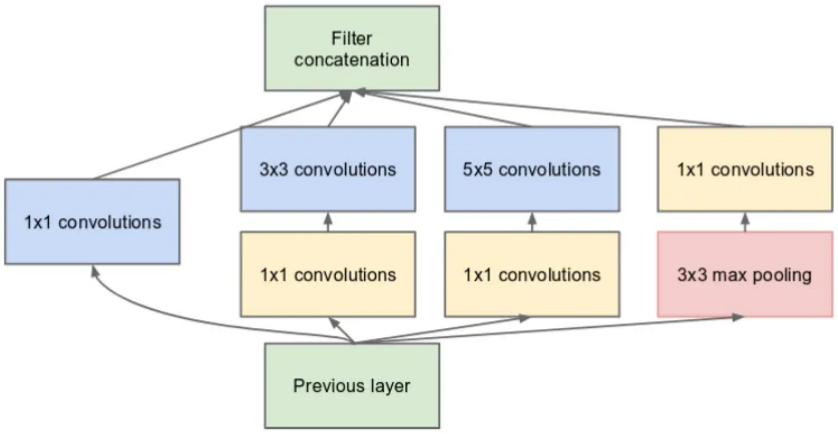

Figure 4.2: Inception module with dimension reduction [41]

Using the dimension-reduced inception module, a neural network architecture was built. Thus Inception v1 was created, or more commonly known as GoogLeNet. the architecture is shown below:

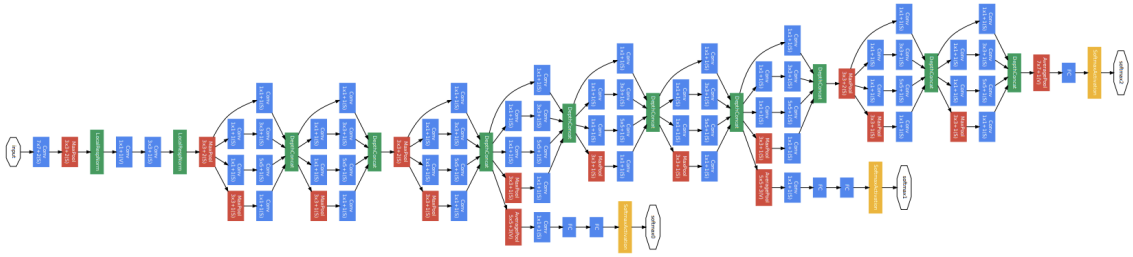

Figure 4.3: Inception v1 architecture [41]

GoogLeNet has 9 such inception modules stacked linearly. It is 22 layers deep (27, including the pooling layers). Because of it's depth, it is subject to the vanishing gradient problem.

The vanishing gradient problem is encountered when training neural networks with gradient-based learning methods and backpropagation. In such methods, during each iteration of training, each of the neural network weights receives an update proportional to the partial derivative of the error function with respect to the current weight [3]. the problem is that as the sequence length increases, the magnitude of the gradient is typically expected to decrease, slowing down the training process. In the worst case, this may completely stop the neural network from further training.

To prevent the middle part of the network from "dying out", the authors introduced two auxiliary classifiers. They applied softmax function to the outputs of two of the inception modules and computed an auxiliary loss over the same labels. the total loss function is a weighted sum of the auxiliary loss and the real loss. However, auxiliary loss is purely used for training purposes and is ignored during inference. the function of total loss used during training is the following:

*total loss = real loss + 0.3*aux loss(1) + 0.3*aux loss(2)*

Inception v2 and Inception v3 were presented in the same paper [43]. the authors proposed a number of improvements that increased accuracy and reduced computational complexity. Inception v2 utilizes the factorization of the 5x5 convolution to two 3x3 convolution operations to improve the computational speed. Although this may seem counter-intuitive, a 5x5 convolution is more than 2.5 times more expensive than a 3x3 convolution. So stacking two 3x3 convolutions leads to a boost in performance. In addition, they factorize the convolutions of the filter size NxN to a combination of 1xN and Nx1 convolutions. For example, a 3x3 convolution is equivalent to first performing a 1x3 convolution and then performing a 3x1 convolution on its output. They found this method to be 33% less computationally expensive than a single 3x3 convolution.

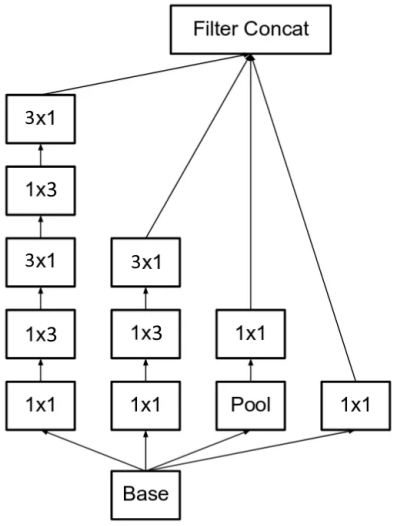

Figure 4.4: Inception v2 module architecture [43]

Inception v3 refers to the last experiment, where Szegedy et al. [43] used the auxiliary component for batch normalization, a type of regularization that aims to reduce the internal covariate shift and, in doing so, aims to accelerate the training of deep neural nets. In Inception v3, the fully connected layer of the auxiliary classifier is also batch normalized, not just the convolutions.

Inception v4 and Inception-ResNet are explained in another paper by Szegedy et al. [42]. Inception v4 focuses on making the modules more uniform. the authors noticed that some of the modules were more complicated than necessary. This enabled a boost to performance by adding more uniform modules. Because of that, the 'stem',which is an initial set of operations performed before introducing the Inception blocks, was modified. Instead of using one linear branch, Inception v4 uses in some parts of stem parallel branches which are joined using filter concatenation.

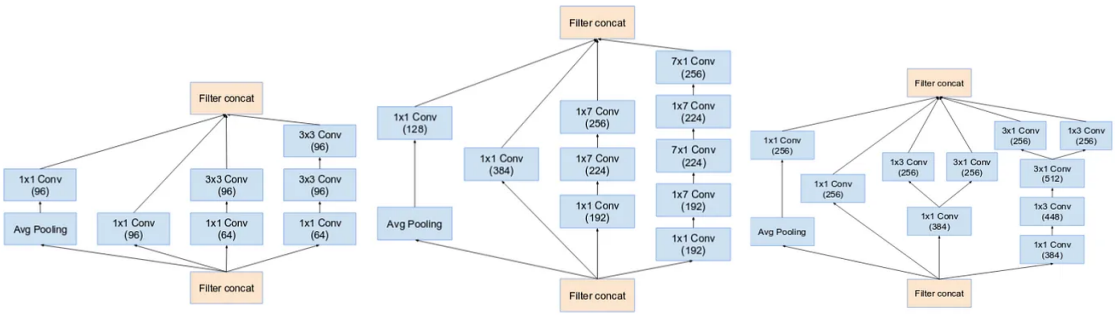

Figure 4.5: Inception v4 modules A, B and C [42]

In each type of Inception module (specified as module A, B, and C), the Inception block is followed by a filter-expansion layer (1×1 convolution without activation), which is used to scale the dimensionality of the filter bank before addition to match the depth of the input. This is needed to compensate for the reduction in dimensionality induced by the Inception

block [42]. Additionally, Inception v4 introduced specialized Reduction blocks, which are placed between each type of Inception module group. They are used to change the width and height of the grid. the earlier versions of Inception had this functionality implemented, but they did not explicitly have reduction blocks.

Inception-ResNet, which is a combination of recent Inception models and discovery of Residual Learning by Kaiming He et al. [17], a hybrid Inception module was proposed. There are two sub-versions of Inception-ResNet, namely v1 and v2. Although they have the same structure for modules A, B, C, and the reduction blocks, they have different stems. This is because Inception-ResNet v1 is based on older Inception v3, while Inception-ResNet v2 is based on newer Inception v4.

The premise was to introduce residual connections that add the output of the convolution operation of the inception module to the input. However, for residual addition to work, the input and output after convolution must have the same dimensions. Hence, 1x1 convolutions were used after the original convolutions, to match the depth sizes. In addition, the pooling operations within the main inception modules were replaced in favor of residual connections. However, these operations can still be found in the Reduction blocks. Interestingly, Reduction block a is the same as in Inception v4.

4.1.2 EfficientNet

EfficientNet is another milestone in the field of deep learning, representing a paradigm shift in the approach to neural network architectures. Developed by Mingxing Tan and Quoc V. Le at Google Research [44], EfficientNet addresses the growing demand for computationally efficient models without compromising performance.

As the scale of deep learning models has increased over the years, so has the computational cost associated with training and deploying these models. This burst in computational demand has posed significant challenges in terms of resource utilization, energy consumption, and deployment on devices with limited processing capabilities. In response to these challenges, EfficientNet emerged as a solution to achieve optimal model efficiency.

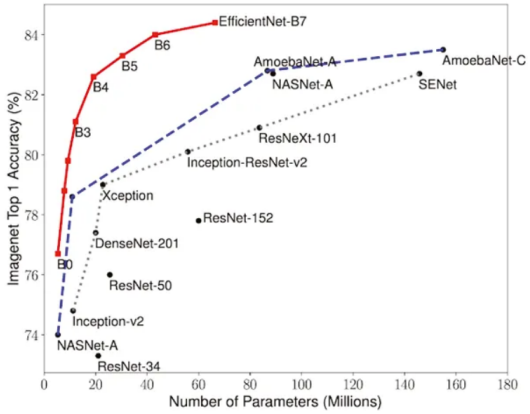

Figure 4.6: Comparision of accurancy and complexity of various detection models [32]

EfficientNet achieves its efficiency through a novel method, called compound scaling, which uniformly scales the depth, width, and resolution of the network [45]. This approach allows for a balanced allocation of resources, ensuring that the model is efficient across

various computational aspects. By optimizing these three dimensions jointly, EfficientNet achieves superior performance compared to traditional scaling methods. But EfficientNet's architecture is not solely the result of compound scaling, but it also involves the use of Neural Architecture Search. This technique is used to design networks that are on par or outperform hand-designed architectures in terms of performance, losses, and overall efficiency.Neural Architecture Search has three main building blocks that can be categorized in terms of search space, search strategy/algorithm, and performance evaluation strategy [12].

The search space defines what type of architecture can be discovered by the NAS algorithm. It is defined by a set of operations that specify the overall structure of the network, the type of units or blocks that define the layers, and the possible connectivity between layers to create architectures. the more elements the search space has, the more complex and versatile it becomes. But naturally, as the search space expands, the costs of finding the best architecture also increase. Types of operations used in defining the search space include sequential layer-wise operations, cell-based representation, hierarchical structure, etc.

The search strategy determines how the algorithm experiments with different neural networks. In general, the algorithm optimizes the child model performance metrics from a sample of the population of network candidates as rewards to create the output of high-performance architecture candidates.

There are various methods that optimize search strategies to deliver better results faster and with consistency. Types of search algorithms include random search, neuro-evolutionary methods [12], Bayesian approaches [49], and reinforcement learning [54].

Some recent evidence suggests that evolutionary techniques perform just as well as reinforcement learning [36]. Moreover, evolutionary methods tend to have a better 'anytime performance' and settle for smaller models. Although earlier NAS techniques were based on discrete search spaces, a continuous formulation of the architecture search space has introduced differentiable search methods, which opened the way for gradient-based optimization [23].

4.1.3 MesoNet

The MesoNet model is one of the first models dedicated to the detection of the Deepfake video falsification technique. Presented by Afchar et al. [1], this model detects forged faces in videos by using a mesoscopic level of analysis, hence its name.

Because microscopic analyses based on image noise cannot be applied in a compressed video context where image noise is strongly degraded and at a higher semantic level, human eyes struggle to distinguish forged images, especially when the image depicts a human face, a mesoscopic approach is used. Predictions are made for each face image extracted from video using the Viola-Jones detector [48] on a frame-by-frame basis aligned using a trained neural network for facial landmark detection.

The two variants of MesoNet purposed in the paper are Meso-4 and MesoInception-4. With Meso-4, there are 4 convolutional layers, each followed by ReLU activation [2]. the first convolutional layer uses kernels of size 3x3, while the following convolutional layers use kernels of size 5x5. Each convolutional layer is followed by a batch normalization layer and a pooling layer. the last convolutional layer is connected to a fully-connected layer with 16 neurons and then the output layer. the architecture is similar to classical CNN, but has a surprisingly good performance.

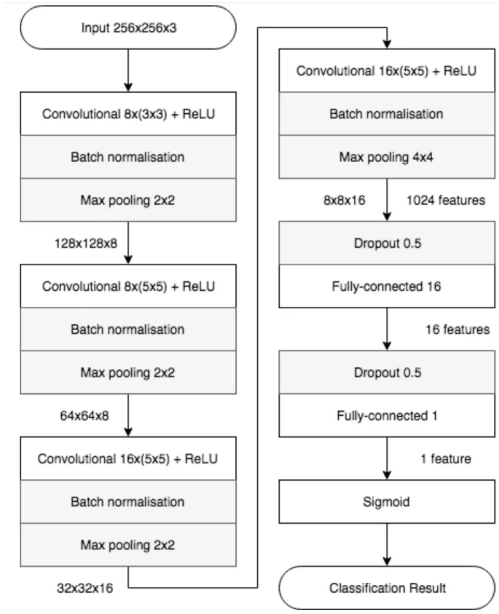

Figure 4.7: Meso-4 architecture [1]

A more advanced variant of MesoNet is MesoInception-4, published in the same paper [1]. In this variant, the convolutional layers are changed to inception modules based on the inception module in GoogleNet, described in Section 4.1.1. Inside each inception module, there are four parallel branches of convolutional layers (two of them dilated), each with a different reception field. However, instead of increasing the filter size, the inception module here increases the reception field by using the dilated convolutional layers with different factors. the feature maps from each branch are then concatenated to form the output of the module.

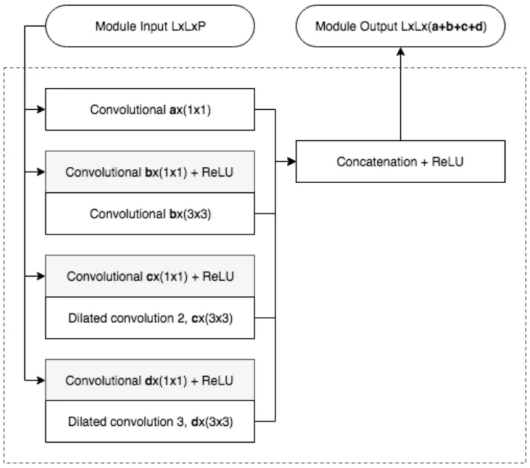

Figure 4.8: MesoInception-4 architecture [1]

Chapter 5

Experiment design proposal

In this chapter, we create design proposals for experiments. We will design models based on the architectures described above. Then we compare these models with their original architecture to see, if these changes increased some of the specs. Model's architectures we used are *Inception-v4*, *Inception-ResNet-v2*, *EfficientNet-v2-B0*, *Meso-4* and *MesoInception-4*.

Each model was trained for 10 epochs with datasets described in Chapter 6. The batch size was selected for all models as 128. However, input dimensions vary between models. Models based on MesoNet architecture, which are Meso-4 and MesoInception-4, have input dimensions 256, 256 and 3 for height, width and depth respectively. For Inception-v4 and Inception-ResNet-v2, which have their stem from Inception architecture, have input dimensions 299, 299 and 3 for height, width and depth respectively. Last input dimension size is based on EfficientNet architecture, with it's width, height and depth of 224, 224 and 3 respectively.

After training a neural network, we compare their parameters. The parameters we'll be comparing are accuracy and loss function during training, accuracy and loss function during validation, testing accuracy, time to train the model, number of trainable parameters and memory requirements[1].

[1]Code used for calculating memory requirements: https://github.com/tensorflow/tensorflow/issues/36327#issuecomment-708571992

Chapter 6

Dataset

Neural network datasets serve as a foundation for training and evaluating machine learning models. There are tens or hundreds of video deepfake datasets varying in length, number of videos, quality, etc. Although common practice is to divide all prepared datasets into training, validation, and testing groups, we decided to use entirely different dataset purely for testing to bring the results closer to real-life situation, when model is challenged in facing data that are different from training and validating set.

6.1 Training dataset

For the training dataset, we used Celeb-DF-v2 presented in the article by Yuezun Li et al. [52]. This huge dataset created in 2019 is made of 590 real videos and 5, 639 deepfake videos, corresponding to more than two million frames.

Figure 6.1: Green box: Real images, Red box: Corresponding DeepFake images [52].

The average length of all videos is approximately 13 seconds with the standard frame rate of 30 frames per second. the real YouTube videos are chosen from publicly available videos, corresponding to interviews of 59 celebrities with a diverse distribution in their genders, ages, and ethnic groups. 56.8% subjects in the real videos are male, and 43.2% are female. 8. 5% are 60 years or older, 30. 5% are between 50 - 60, 26. 6% are 40 years, 28. 0% are 30 years old and 6. 4% are younger than 30. 5. 1% are Asian, 6. 8% are African Americans, and 88. 1% are Caucasian. In addition, real videos exhibit a wide range of changes in aspects such as the subjects' face sizes (in pixels), orientations, lighting

conditions, and backgrounds. the deepfake videos are generated by swapping faces for each pair of the 59 subjects. the final videos are in MPEG4.0 format [52].

6.2 Validating dataset

For the validation of the model, we used an older version of the previously mentioned dataset, Celeb-DF. the Celeb-DF dataset is made up of 158 videos and 795 deepfake videos, corresponding to almost five hundred thousand frames. Because this dataset is part of Celeb-DF-v2 mentioned in Section 6.1, the videos that are shared between both datasets are removed from the training dataset. With this change, we can reduce the likelihood of overfitting by ensuring that the training and validation datasets are different.

6.3 Testing dataset

For our test dataset, we choose a combination of the DF-TIMIT [25] and VID-TIMIT [38] datasets. This combination was done due to the lack of real videos in the DF-TIMIT dataset and vice versa. In result, we get 640 deepfake videos and 559 real videos in a combined dataset.

DF-TIMIT dataset published by Pavel Korshunov and Sebastien Marcel [25] consists of 640 DeepFake videos generated with faceswap-GAN [39]. the videos are divided into two subsets of equal size: DF-TIMIT-LQ and DF-TIMIT-HQ, with synthesized faces of size 64 $\times$ 64 and 128x128 pixels, respectively.

Figure 6.2: a) Original donor b) Original target c) Face swapped [25].

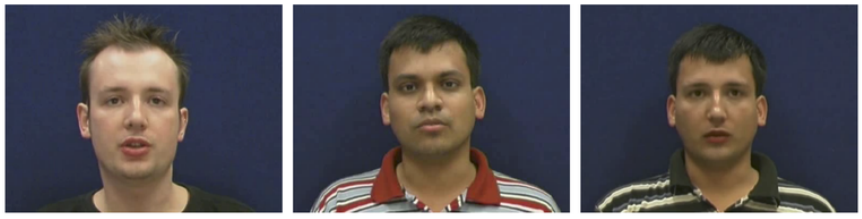

Figure 6.3: a) Original donor b) Original target c) Face swapped [25].

VID-TIMIT dataset published by Conrad Sanderson and Brian Lovell [38] consists of 559 real videos split into frames. In these videos, people read specific sentences. This dataset is usually used for image recognition, but when used in combination with DFTIMIT, it can be part of a valid deepfake dataset.

Chapter 7

Results of experiments

In this chapter, we evaluate the results of the models described in Chapter 5 and compare their results with other models of different architectures.

During the training phase, we can see an expected increase in training accuracy over the epochs and a continuous decrease in loss function. This can be observed for all trained models. The models managed to achieve approximately eighty percent accuracy after ten epochs. However, we can already see huge differences in training times. Surprisingly, the fastest model to train was based on the EfficientNet-v2 architecture with a training time of about an hour, although it has about a hundred times more trainable parameters than the Meso architecture models Meso-4 and MesoInception-4, which take one and a half hours to train. The slowest models to train were Inception-v4 and Inception-ResNet-v2, which was expected considering the number of trainable parameters. The training time for Inception-v4 was ten times longer than the Meso architecture with sixteen hours to train, and the training time for Inception-ResNet-v2 was more than three times longer with more than fifty hours of training.

7.1 Experiment 1: Inception-v4

Inception-v4 was one of the more complex models we created. We expected that increased complexity and more trainable parameters could mean better accuracy at the cost of increased memory requirements and training time. Despite that, Inception-v4 achieved mediocre results. Compared to the Meso or EfficientNet architecture, this model achieved the same or even slightly worse accuracy in training, validating, and testing datasets compared to other models.
Total number of parameters: 41 177 474
Trainable parameters: 41 114 306
Non-trainable parameters: 63 168
Total train time: 16 h
Memory requirements: 27.444 GB

Epoch	Time	Steps	sec/step	Loss	Accuracy	Valid_Loss	Valid_Acc
1	5472	1327	4	0.5685	0.7504	0.4857	0.7485
2	5805	1327	4	0.5452	0.7544	2.8394	0.7583
3	5745	1327	4	0.5466	0.7520	0.8345	0.7256
4	5545	1327	4	0.5095	0.7659	0.4267	0.7517
5	5756	1327	4	0.4924	0.7757	0.7648	0.7719
6	5634	1327	4	0.4849	0.7794	0.7584	0.7768
7	5591	1327	4	0.4807	0.7793	0.7587	0.8026
8	5627	1327	4	0.4786	0.7864	0.7531	0.7897
9	5568	1327	4	0.4745	0.7916	0.7486	0.8049
10	5701	1327	4	0.4703	0.8006	0.7425	0.8035

Table 7.1: Training of Inception-v4 model

7.2 Experiment 2: Inception-ResNet-v2

Inception-ResNet-v2 was an interesting concept of a heavy-weight model with huge memory and train time requirements, but with increased accuracy over light-weight models. However, the opposite is true. Inception-ResNet-v2 managed to barely achieve eighty percent in training accuracy, but it did not manage to reach this milestone in validation accuracy. This is unimpressive result considering its enormous memory and train time requirements.

Total number of parameters: 75 259 042
Trainable parameters: 75 231 522
Non-trainable parameters: 27 520
Total train time: 52 h
Memory requirements: 51.855 GB

Epoch	Time	Steps	sec/step	Loss	Accuracy	Valid_Loss	Valid_Acc
1	19400	1327	14	1.0338	0.5647	1.0638	0.6178
2	19387	1327	14	1.1589	0.6651	0.9634	0.6238
3	19274	1327	14	1.1755	0.7184	0.931	0.6439
4	18963	1327	14	0.9064	0.7349	0.9257	0.6622
5	19002	1327	14	0.9518	0.7457	0.8398	0.6978
6	18430	1327	14	1.0604	0.7568	0.7828	0.7139
7	19110	1327	14	1.0136	0.7706	0.836	0.7350
8	19353	1327	14	0.825	0.7704	0.8518	0.7504
9	18752	1327	14	0.98	0.7864	0.6767	0.7751
10	18914	1327	14	0.6779	0.8034	0.7368	0.7867

Table 7.2: Training of Inception-ResNet-v2 model

7.3 Experiment 3: EfficientNet-v2-B0

EfficientNet-v2-B0 was an unexpected surprise during the comparison of the models. Even though its memory requirements were number of times higher than for Meso architecture, training time for this model was fastest. It also managed to gain the same level of accuracy

as other models, which were trained for longer time. Only possible negative could be higher memory requirements compared to Meso models.
Total number of parameters: 5 921 874
Trainable parameters: 5 861 266
Non-trainable parameters: 60 608
Total train time: 1 h
Memory requirements: 14.398 GB

Epoch	Time	Steps	sec/step	Loss	Accuracy	Valid_Loss	Valid_Acc
1	351	1327	0.264	0.8678	0.7290	1.1584	0.7361
2	341	1327	0.273	0.6541	0.7441	0.9763	0.7492
3	335	1327	0.274	0.6109	0.7439	0.8761	0.7490
4	342	1327	0.276	0.5980	0.7471	0.8957	0.7486
5	348	1327	0.278	0.5793	0.7457	0.8537	0.7397
6	331	1327	0.270	0.5813	0.7442	0.8367	0.7418
7	330	1327	0.269	0.5534	0.7532	0.8298	0.7534
8	337	1327	0.272	0.5086	0.7805	0.8345	0.7698
9	341	1327	0.271	0.4635	0.8028	0.8202	0.7761
10	328	1327	0.273	0.4158	0.8248	0.7997	0.7934

Table 7.3: Training of EfficientNet-v2-B0 model

7.4 Experiment 4: Meso-4

Meso-4 is a great example of a lightweight model. Compared to Inception and EfficientNet architecture, Meso can achieve very similar results in therms of accuracy as EfficientNet with approximately same training time, but with much lower number of parameters and memory requirements.
Total number of parameters: 44 522
Trainable parameters: 44 426
Non-trainable parameters: 55
Total train time: 1 h and 20 min
Memory requirements: 0.884 GB

Epoch	Time	Steps	sec/step	Loss	Accuracy	Valid_Loss	Valid_Acc
1	583	1327	0.437	0.6868	0.7296	0.7348	0.6843
2	527	1327	0.397	0.5267	0.7581	0.7164	0.7168
3	496	1327	0.374	0.5139	0.7609	0.7091	0.7249
4	472	1327	0.356	0.5074	0.7627	0.7001	0.7461
5	454	1327	0.342	0.4878	0.7739	0.6924	0.7435
6	437	1327	0.329	0.4643	0.7879	0.6885	0.7610
7	431	1327	0.325	0.4437	0.7983	0.6723	0.7753
8	423	1327	0.319	0.4252	0.8067	0.6767	0.7812
9	414	1327	0.312	0.4066	0.8162	0.6532	0.7924
10	409	1327	0.308	0.4000	0.8201	0.6264	0.7942

Table 7.4: Training of Meso-4 model

7.5 Experiment 5: MesoInception-4

MesoInception-4 is supposed to be a join of the best of both the Meso and Inception architectures. However, the results of this model are nearly identical to the results of Meso-4, only with higher memory requirements. Total number of parameters: 44 711
Trainable parameters: 44 605
Non-trainable parameters: 106
Total train time: 1 h and 25 min
Memory requirements: 2.234 GB

Epoch	Time	Steps	sec/step	Loss	Accuracy	Valid_Loss	Valid_Acc
1	552	1327	0.413	0.9008	0.7049	0.8947	0.6848
2	513	1327	0.387	0.5321	0.7557	0.7167	0.7467
3	486	1327	0.366	0.5089	0.7693	0.6837	0.7673
4	485	1327	0.365	0.4869	0.7808	0.6634	0.7591
5	483	1327	0.364	0.4643	0.7911	0.6471	0.7658
6	481	1327	0.362	0.4462	0.8011	0.6386	0.7709
7	479	1327	0.361	0.4306	0.8066	0.6196	0.7762
8	474	1327	0.357	0.4180	0.8120	0.6072	0.7833
9	475	1327	0.358	0.4053	0.8188	0.5805	0.7898
10	473	1327	0.356	0.3846	0.8277	0.5437	0.7958

Table 7.5: Training of MesoInception-4 model

7.6 Accuracy against new testing dataset

However, during the testing of models with an entirely new type of dataset, the models did not maintain the previous level of precision achieved during the training and validation phase. We can see heavy shifts of accuracy and loss function between training and testing datasets. The results can be seen in Table 7.6.

Model name	Testing_accurancy	Testing_loss
EfficientNet_v2_B0	0.43378	1.34676
Meso_4	0.33933	1.18363
Meso_Inception_4	0.39824	0.95373
Inception_v4	0.38676	0.98176
Inception_ResNet_v2	0.36046	1.05375

Table 7.6: Results of testing of models on TIMID dataset

Chapter 8

Conclusions

The requirement of models for detecting video deepfakes is becoming more and more vital, as the spread and availability of deepfakes is increasing every day.

In this thesis, we have learned what deepfakes are, how they can be dangerous to our daily lives, and how they are created. We looked into many creation methods to understand the process of creating video deepfake and what are modern state-of-the-art tools to create them. Then we looked at the other side: the detection of video deepfakes. We found out how we can detect deepfakes, and what current models and solutions are. Then we tried to create our own deepfake detection models and tried to compare them to see, what are strong and weak points of each architecture. Although the results of the training and testing datasets were vastly different, we can see on this how specific are deepfake detection models and how big difference can change of dataset to entirely different type make with model accurancy.

Bibliography

[1] Afchar, D., Nozick, V., Yamagishi, J. and Echizen, I. Mesonet: a compact facial video forgery detection network. In: IEEE. *2018 IEEE international workshop on information forensics and security (WIFS)*. 2018, p. 1–7.

[2] Agarap, A. F. Deep learning using rectified linear units (relu). *ArXiv preprint arXiv:1803.08375*. 2018.

[3] Basodi, S., Ji, C., Zhang, H. and Pan, Y. Gradient Amplification: An efficient way to train deep neural networks. *CoRR*. 2020, abs/2006.10560. Available at: https://arxiv.org/abs/2006.10560.

[4] Brownlee, J. A Gentle Introduction to Generative Adversarial Networks. *Machine Learning Mastery*. 2019. Available at: https://machinelearningmastery.com/.

[5] Brownlee, J. *Machine Learning Mastery With Python: Understand Your Data, Create Accurate Models and Work Projects End-To-End*. Independently published, 2021.

[6] Bulat, A. and Tzimiropoulos, G. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In: *Proceedings of the IEEE international conference on computer vision*. 2017, p. 1021–1030.

[7] Cao, C., Weng, Y., Lin, S. and Zhou, K. 3D shape regression for real-time facial animation. *ACM Trans. Graph.* New York, NY, USA: Association for Computing Machinery. jul 2013, vol. 32, no. 4. DOI: 10.1145/2461912.2462012. ISSN 0730-0301. Available at: https://doi.org/10.1145/2461912.2462012.

[8] Carlsson, T. and Strömberg, O. *Accuracy and Robustness of State of the Art Deepfake Detection Models*. 2023.

[9] Chopra, S., Hadsell, R. and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, vol. 1, p. 539–546 vol. 1. DOI: 10.1109/CVPR.2005.202.

[10] Coccomini, D. A., Caldelli, R., Falchi, F. and Gennaro, C. On the Generalization of Deep Learning Models in Video Deepfake Detection. *Journal of Imaging*. 2023, vol. 9, no. 5. DOI: 10.3390/jimaging9050089. ISSN 2313-433X. Available at: https://www.mdpi.com/2313-433X/9/5/89.

[11] Dongare, A., Kharde, R., Kachare, A. D. et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*. Citeseer. 2012, vol. 2, no. 1, p. 189–194.

[12] Elsken, T., Metzen, J. H. and Hutter, F. Neural architecture search: A survey. *Journal of Machine Learning Research*. 2019, vol. 20, no. 55, p. 1–21.

[13] Feng, Y., Wu, F., Shao, X., Wang, Y. and Zhou, X. Joint 3d face reconstruction and dense alignment with position map regression network. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, p. 534–551.

[14] Georghiades, A., Belhumeur, P. and Kriegman, D. From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2001, vol. 23, no. 6, p. 643–660. DOI: 10.1109/34.927464.

[15] Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks. *ArXiv preprint arXiv:1701.00160*. 2016.

[16] Goodfellow, I., Pouget Abadie, J., Mirza, M., Xu, B., Warde Farley, D. et al. Generative adversarial nets. *Advances in neural information processing systems*. 2014, vol. 27.

[17] He, K., Zhang, X., Ren, S. and Sun, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770–778.

[18] Heidari, A., Jafari Navimipour, N., Dag, H. and Unal, M. Deepfake detection using deep learning methods: A systematic and comprehensive review. *WIREs Data Mining and Knowledge Discovery*. 2024, vol. 14, no. 2, p. e1520. DOI: https://doi.org/10.1002/widm.1520. Available at: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1520.

[19] Ho, J., Jain, A. and Abbeel, P. Denoising Diffusion Probabilistic Models. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. and Lin, H., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020, vol. 33, p. 6840–6851. Available at: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[20] Qayyum, R. *Introduction To Pooling Layers In CNN*. 2022.

[21] Iglovikov, V. and Shvets, A. Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. *ArXiv preprint arXiv:1801.05746*. 2018.

[22] Joseph, J. *A Gentle Introduction to Neural Networks*. 2019. Available at: https://clevertap.com/blog/neural-networks/.

[23] Júnior, J. H. C., Della Lucia, F. L., Sutili, T., Melloy, D. A. A. and Figueiredo, R. C. Gradient-based Optimization for Unrepeatered Optical Systems. In: *2019 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*. 2019, p. 1–3. DOI: 10.1109/IMOC43827.2019.9317624.

[24] Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *ArXiv preprint arXiv:1312.6114*. 2013.

[25] Korshunov, P. and Marcel, S. *Deepfakes: a new threat to face recognition? assessment and detection*. 2018.

[26] Korshunova, I., Shi, W., Dambre, J. and Theis, L. Fast face-swap using convolutional neural networks. In: *Proceedings of the IEEE international conference on computer vision*. 2017, p. 3677–3685.

[27] Lawton, G. *Variational autoencoder*. 2023. Available at: https://www.techtarget.com/searchenterpriseai/definition/variational-autoencoder-VAE.

[28] LeCun, Y., Bengio, Y. and Hinton, G. Deep Learning. *Nature*. may 2015, vol. 521, p. 436–44. DOI: 10.1038/nature14539.

[29] Liu, K., Perov, I., Gao, D., Chervoniy, N., Zhou, W. et al. Deepfacelab: Integrated, flexible and extensible face-swapping framework. *Pattern Recognition*. 2023, vol. 141, p. 109628. DOI: https://doi.org/10.1016/j.patcog.2023.109628. ISSN 0031-3203. Available at: https://www.sciencedirect.com/science/article/pii/S0031320323003291.

[30] Lo, C.-C., Lee, C.-H. and Huang, W.-C. Prognosis of Bearing and Gear Wears Using Convolutional Neural Network with Hybrid Loss Function. *Sensors*. 2020, vol. 20, no. 12. DOI: 10.3390/s20123539. ISSN 1424-8220. Available at: https://www.mdpi.com/1424-8220/20/12/3539.

[31] M., C. *Pattern Recognition and Machine Learning*. 1st ed. New York, NY: Springer, august 2006. Information Science and Statistics.

[32] Mingxing Tan, Q. V. L. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling. *Google Research*. 2019. Available at: https://research.google/blog/efficientnet-improving-accuracy-and-efficiency-through-automl-and-model-scaling/.

[33] Mirsky, Y. and Lee, W. The Creation and Detection of Deepfakes: A Survey. *ACM Comput. Surv.* 1st ed. New York, NY, USA: Association for Computing Machinery. jan 2021, vol. 54, no. 1. DOI: 10.1145/3425780. ISSN 0360-0300. Available at: https://doi.org/10.1145/3425780.

[34] O'shea, K. and Nash, R. An introduction to convolutional neural networks. *ArXiv preprint arXiv:1511.08458*. 2015.

[35] Ratliff, L. J., Burden, S. A. and Sastry, S. S. Characterization and computation of local Nash equilibria in continuous games. In: *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2013, p. 917–924. DOI: 10.1109/Allerton.2013.6736623.

[36] Real, E., Aggarwal, A., Huang, Y. and Le, Q. V. Regularized evolution for image classifier architecture search. In: *Proceedings of the aaai conference on artificial intelligence*. 2019, vol. 33, p. 4780–4789.

[37] Salman, S. and Shamsi, J. A. *Comparison of Deepfakes Detection Techniques*. 1st ed. 2023. DOI: 10.1109/ICAI58407.2023.10136659.

[38] Sanderson, C. and Lovell, B. *Multi-Region Probabilistic Histograms for Robust and Scalable Identity Inference*. June 2009. ISBN 978-3-642-01792-6.

[39] SHAOANLU. *Faceswap-GAN*. GitHub, 2024. Available at: https://github.com/shaoanlu/faceswap-GAN.

[40] SONG, Y., SOHL DICKSTEIN, J., KINGMA, D. P., KUMAR, A., ERMON, S. et al. Score-based generative modeling through stochastic differential equations. *ArXiv preprint arXiv:2011.13456*. 2020.

[41] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S. et al. Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun 2015, p. 1–9. DOI: 10.1109/CVPR.2015.7298594. ISSN 1063-6919. Available at: https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594.

[42] SZEGEDY, C., IOFFE, S., VANHOUCKE, V. and ALEMI, A. Inception-v4, inception-resnet and the impact of residual connections on learning. In: *Proceedings of the AAAI conference on artificial intelligence*. 2017.

[43] SZEGEDY, C., VANHOUCKE, V., IOFFE, S., SHLENS, J. and WOJNA, Z. Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 2818–2826.

[44] TAN, M. and LE, Q. Efficientnetv2: Smaller models and faster training. In: PMLR. *International conference on machine learning*. 2021, p. 10096–10106.

[45] TAN, M. and LE, Q. V. Mixconv: Mixed depthwise convolutional kernels. *ArXiv preprint arXiv:1907.09595*. 2019.

[46] THIES, J., ZOLLHOFER, M., STAMMINGER, M., THEOBALT, C. and NIESSNER, M. Face2face: Real-time face capture and reenactment of rgb videos. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 2387–2395.

[47] VENKATESAN, R. and LI, B. *Convolutional neural networks in visual computing*. 1st ed. London, England: Routledge, october 2017. Data-Enabled Engineering. ISBN 978-1-351-65032-8.

[48] VIOLA, P. and JONES, M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. 2001, vol. 1, p. I–I. DOI: 10.1109/CVPR.2001.990517.

[49] WHITE, C., NEISWANGER, W. and SAVANI, Y. Bananas: Bayesian optimization with neural architectures for neural architecture search. In: *Proceedings of the AAAI conference on artificial intelligence*. 2021, vol. 35, p. 10293–10301.

[50] WU, Y.-c. and FENG, J.-w. Development and application of artificial neural network. *Wireless Personal Communications*. Springer. 2018, vol. 102, p. 1645–1656.

[51] YANG, Y., ZHENG, K. and WU, C. Improving the Classification Effectiveness of Intrusion Detection by Using Improved Conditional Variational AutoEncoder and Deep Neural Network. *Sensors*. june 2019, vol. 19, p. 2528. DOI: 10.3390/s19112528.

[52] Yuezun Li, H. Q. and Lyu, S. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. In: *IEEE Conference on Computer Vision and Patten Recognition (CVPR)*. 2020.

[53] Zhang, Z., Luo, C. and Zhao, Z. Application of probabilistic method in maximum tsunami height prediction considering stochastic seabed topography. *Natural Hazards*. Springer. 2020, vol. 104, no. 3, p. 2511–2530.

[54] Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *ArXiv preprint arXiv:1611.01578*. 2016.

Appendix A

Contents of the included storage media

- `/` ..... Storage media
  - `dataset/` ..... Root dir of all datasets
    - `Celeb-DF` ..... Root dir of Celeb-DF dataset
    - `Celeb-DF-v2` ..... Root dir of Celeb-DF-v2 dataset
    - `DFTIMIT` ..... Root dir of DFTIMIT dataset
    - `VidTIMIT` ..... Root dir of VidTIMIT dataset
  - `latex/` ..... Source code of this thesis in LaTeX format
  - `results` ..... Results of experiments
    - `data` ..... Data of models in .csv format
    - `graphs` ..... Graphs of models parameters
  - `src/` ..... Source code root dir
    - `dataset_processing` ..... Dir with source codes for processing datasets
    - `models` ..... Root dir for source code of models