BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FAKE FACE DETECTION IN THE DIGITAL IMAGES

DETEKCE PODVRHU OBLIČEJE V DIGITÁLNÍCH SNÍMCÍCH

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. TOMÁŠ LAPŠANSKÝ**
AUTOR PRÁCE

SUPERVISOR **Ing. FILIP ORSÁG, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

148786

Institut:	Department of Intelligent Systems (UITS)
Student:	**Lapšanský Tomáš, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Information Systems and Databases
Title:	**Fake Face Detection in the Digital Images**
Category:	Security
Academic year:	2022/23

Assignment:

1. Learn about image processing methods as well as neural network (NN) techniques and their application to the images.
2. Study methods solving the fake detection in the digital images,. Familiarise yourself with the state of the art algorithms and available data sets and identify the challenges of the (deep) fake face detection.
3. Based on the gained knowledge, propose your own algorithm for the fake detection using the methods of the image processing and NN.
4. Implement a simple script to train and test your algorithm in the Python programming language using the available libraries (e.g. OpenCV, PyTorch or Numpy).
5. Carry out experiments with your implementation. Test detection of the various types of the face fakes (for example on the images generated by the StarGAN neural network) and compare it to the state of the art solutions. Compare your individual NN iterations and identify their weak spots. Attempt iteratively to improve results of your NN design.
6. Summarise and discuss the achieved results and discuss weaknesses and strengths of the proposed NN designs.

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.

Requirements for the semestral defence:

- Items 1 and 2 of the assignment.

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

Supervisor:	**Orság Filip, Ing., Ph.D.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	17.5.2023
Approval date:	3.11.2022

Abstract

In recent years, we can observe the rise and rapid development of neural networks and artificial intelligence in information technology, which include deepfake photos and videos. Generative adversarial neural networks (GANs) are a clear example of this. Nowadays, they can achieve virtually impossible results for the average person to distinguish from reality. Since these networks can therefore be misused for various purposes, it is necessary to be able to distinguish between what is generated and what is real. This thesis explores current state-of-the-art neural network solutions that can serve as suitable models for deepfake detection. We investigate individual architectures that are suitable as a baseline model for detection, address possible improvements to this model, and develop several new architectures. We then investigate these and evaluate their results. In conclusion, we have a discussion of the results and open further questions on this complex issue.

Abstrakt

V posledných rokoch môžeme pozorovať nárast a rýchly rozvoj neurónových sietí a umelej inteligencie v informačných technológiách, medzi ktoré patria aj deepfake fotografie a videá. Generatívne adverzné neurónové siete (GAN) sú toho jasným príkladom. V súčasnosti dokážu dosiahnuť výsledky, ktoré bežný človek nedokáže rozoznať od reality. Keďže tieto siete sa preto dajú zneužiť na rôzne účely, je potrebné vedieť rozlíšiť, čo je vygenerované a čo je skutočné. Táto práca skúma súčasné najmodernejšie riešenia neurónových sietí, ktoré môžu slúžiť ako vhodné modely na detekciu deepfake. Skúmame jednotlivé architektúry, ktoré sú vhodné ako základný model na detekciu, zaoberáme sa možnými vylepšeniami tohto modelu a vyvíjame niekoľko nových architektúr. Tie potom skúmame a vyhodnocujeme ich výsledky. V závere uvádzame diskusiu o výsledkoch a otvárame ďalšie otázky týkajúce sa tejto zložitej problematiky.

Keywords

deepfake, neural networks, deepfake detection, image detection

Klíčová slova

deepfake, neuronove siete, detekcia deepfake, detekcia v obraze

Reference

LAPŠANSKÝ, 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.

Rozšířený abstrakt

V posledných rokoch môžeme v informačných technológiách pozorovať vzostup a rýchly rozvoj neurónových sietí a umelej inteligencie, ktoré zahŕňajú aj deepfake fotografie a videá. Táto technológia dokáže generovať veľmi realistické audiovizuálne záznamy, ktoré sa dajú efektívne uplatniť v zábavnom a filmovom priemysle, ako aj v umeleckej sfére či vo vzdelávaní a vzbudiť väčší záujem širšieho spektra ľudí. Ako to už býva, každá technológia má nielen svetlé, ale aj temné stránky.

Pomocou tejto technológie je možné uskutočniť niekoľko typov útokov v širokom spektre obyvateľstva, či už ide o online šikanu, politické spektrum a ovplyvňovanie davov ľudí šírením poplašných správ, ale aj falošné obvinenia a ovplyvňovanie súdnictva alebo skrývanie zločineckých identít pri overovaní dokladov totožnosti. Niekoľko z týchto vektorov útokov a tiež reálne udalosti s nimi spojené, pri ktorých boli tieto útoky použité, sú opísané v našej práci.

Keďže sme poukázali na dôležitosť prevencie použitia tejto technológie na zlé účely, je potrebné vedieť, ako sa proti týmto útokom účinne brániť. Moderné neurónové siete, ktoré sme skúmali, dokážu v súčasnosti generovať taký realistický obraz alebo video, že je často ťažké odhaliť tieto útoky len pomocou ľudí. Preto sme sa zamerali na konvolučné neurónové siete a zmapovali sme najmodernejšie riešenia týchto sietí, ktoré sme sa snažili prispôsobiť pre nami vykonávanú úlohu. Preskúmali sme niekoľko modelov, z ktorých najlepšie vyšla sieť EfficientNet. Túto sieť sme potom doladili pre úlohu detekcie deepfakes a snažili sme sa dosiahnuť najlepšiu kombináciu hyperparametrov.

Na efektívne vyhodnotenie vizuálneho záznamu sme sa rozhodli vytvoriť kompletnú pipeline na spracovanie deepfake videa (alebo obrazu). V prípade videa je potrebné rozdeliť video na jednotlivé snímky a každú snímku vyhodnotiť samostatne. Potom je možné vykonať napríklad priemer týchto jednotlivých snímok alebo nastaviť prahovú hodnotu, koľko snímok je potrebné klasifikovať ako falošné. V prípade obrázka vyhodnocujeme len samotný obrázok. Potom vykonáme detekciu tváre nad každou snímkou na obrázku pomocou vopred natrénovaného modelu MTCNN. Zistenú tvár potom rozrežeme na rozmery požadované naším modelom a vykonáme centrovanie tváre na stred vstupného obrázka. Ak sa tvár nachádza na okraji obrazu, je potrebné chýbajúci priestor vyplniť konštantnou farbou.

Túto architektúru sme sa potom pokúsili vylepšiť pomocou niekoľkých prístupov. Najprv sme sa pokúsili upraviť model určený na detekciu objektov, ktorý sme sa rozhodli použiť. Tento prístup zvýšil hodnotu AUC (plocha pod krivkou) ROC (Receiver Operating Characteristic) na neznámom datasete o niekoľko percent. Skúmaním rôznych kombinácií zmrazenia predtrénovaných váh sme dospeli k záveru, že najlepšie výsledky sa dosiahnu pri zmrazení prvých 3 blokov siete (sieť obsahuje 7 blokov, po ktorých nasledujú predikčné bloky z detekcnej siete). Toto správanie môže byť spôsobené tým, že sieť môže obsahovať extrakciu príznakov v prvých troch blokoch siete potrebných pre túto úlohu. Túto teóriu v súčasnom stave nemôžeme potvrdiť a bolo by potrebné hlbšie preskúmať jednotlivé bloky EfficientNet.

Experimentovali sme aj s architektúrou, ktorá by mohla vykonávať detekciu deepfake v obraze a zobrazovať ich pomocou masky, aby sme mohli určiť presné miesto úpravy. Týmto prístupom sa nedosiahli požadované výsledky.

Hoci sa nám podarilo vytvoriť architektúru, ktorá dokáže odhaliť deepfake aj nad úplne novými obrázkami, toto riešenie vyvoláva niekoľko ďalších otázok do budúcnosti, ktoré je potrebné ďalej riešiť.

Prvou otázkou je vyhodnotenie konzistencie riešenia a následné porovnanie s inými existujúcimi detektormi. Keďže neexistuje jednotný postup na vyhodnotenie takejto komplexnej úlohy, vyhodnotením sa zaoberá každá práca samostatne. Aj keď sa na výpočet používajú jednotné metriky, ktoré možno použiť na vyhodnotenie modelu, každé riešenie používa na trénovanie iné dáta. Model teda môže byť špeciálne predtrénovaný pre dataset, ktorý sa používa na evaluáciu modelu, a preto sú jeho výsledky lepšie ako výsledky modelov, ktoré tento súbor údajov nikdy nevideli. Naše riešenie použilo na proces trénovania dataset FaceForensics a na komplexnejšie hodnotenie dataset Celeb-DF.

Ďalšou otázkou je, ako efektívne kontrolovať spôsob, akým sa model trénuje. Pri vyhodnocovaní modelu sme narazili na faktor, keď po niekoľkých epochách model dosiahol skvelé výsledky na validačnom datasete s presnosťou viac ako 95 %. Pri vyhodnocovaní modelu po jednotlivých epochách nad novým datasetom výsledky neboli také dobré a časom sa skôr zhoršovali. V niektorých prípadoch nedošlo ku konvergencii k jednotnému výsledku.

Vzhľadom na tieto faktory môžeme tvrdiť, že je potrebné hlbšie preskúmať a zlepšiť proces vyhodnocovania neurónových sietí pre takéto komplexné úlohy a zjednotiť proces vyhodnocovania jednotlivých komplexných úloh, ktoré je potrebné riešiť.

Fake Face Detection in the Digital Images

Declaration

I hereby declare that this Diploma thesis was prepared as an original work by the author under the supervision of Ing. Filip Orsag, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Tomáš Lapšanský
May 15, 2023

Acknowledgements

I would like to thank my supervisor Ing. Filip Orsag, Ph.D., for his advice and frequent consultations on this work. I would also like to thank my good friend Ing. Anton Firc for his advice and consultation, whether on the topic itself or the text of the thesis and its future direction.

I also thank my whole family, especially my parents, for their support during my studies and my friends, who are always close to me.

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

Contents

Chapter 1

Introduction

In recent years, in modern technology, we can observe the rise and rapid development of neural networks and artificial intelligence, including deepfake photos and videos. This technology can generate very realistic audio-visual recordings which can be effectively applied in the entertainment and film industry, as well as in the artistic sphere or education and attract a higher interest from a broader spectrum of people. As it is usual, every technology has not only a light but also a dark side.

With the help of this technology, it is possible to carry out several types of attacks in a wide population spectrum [5], whether it is online harassment, political spectrum and influencing crowds of people by spreading popular and false news, but also false accusations and influencing the judiciary, or hiding criminal identities when verifying identity documents. Several of these attack vectors and also the real events related to them, where these attacks were used, are described in our work.

What makes this technology dangerous is also its availability. We can create deepfakes of audio-visual content or images using several pictures of a person and our mobile phone or computer. Several freely available applications can do this, or many neural network models can help a more experienced user. What makes them dangerous is that nowadays, they are not even demanding computing power.

Since we pointed out the importance of preventing the use of this technology on the wrong target, it is necessary to know how to defend against these attacks effectively. Modern generative adversarial neural networks (GAN) [23] that we have researched can nowadays generate such a realistic image or video that it is often difficult to detect these attacks using human access alone. This fact is also facilitated by the rapid development in this area and the wide public interest in this issue. The wide applicability of GAN models, for example, in the entertainment industry, which is spreading at breakneck speed on social networks, also contributes to this. We, therefore, focused on convolutional neural networks (CNN) [43] and mapped state-of-the-art solutions of these networks that we tried to fit for the task we performed. We explored several models. We then finetuned these networks for the deepfakes detection task and tried to arrive at the best combination of hyperparameters.

For efficient evaluation of the visual record, we decided to create a complete pipeline for deepfake video (or image) processing. In the case of a video, it is necessary to split the video into individual frames and evaluate each frame separately. In the case of a picture, we evaluate only the image itself. Next, we perform face detection over each frame in the image using the pre-trained model. We then cut the detected face to the dimensions required by our model and perform face centring on the centre of the input image. If the face is located

on the edge of the image, it is necessary to fill the missing space with a constant colour. We then tried to improve this architecture with several approaches.

To evaluate the experiments themselves, we defined a metric by which we evaluated them to correspond to the most realistic possible scenario. At the same time, we could compare the different proposed architectures over it efficiently. We trained our chosen baseline architecture and then experimented more extensively with the different extensions we designed. Overall, we have tried two extensions of the baseline architecture, only one of which we got better results than the baseline architecture. With these extensions, we improved the system's overall ability to detect deepfakes in case we compared it with the baseline architecture. The final output of this thesis is a modified architecture of the selected convolutional network that demonstrates increased accuracy in deepfake detection compared to the baseline architecture.

In the discussion, we then discussed the evaluation of our architecture in detail and tried to look at it from a realistic point of view, not just from the point of view of evaluation scores and numbers. Although we have succeeded in creating an architecture that can detect deepfake even over completely new data, this solution raises several other questions for the future that need to be further addressed.

In Chapter 2, we introduce the issue of neural networks and models necessary for creating deepfakes and those used for their detection. A deeper understanding of deepfakes is introduced in Chapter 3, where we explain how deepfakes are formed and present several freely available models with their evaluation. We then address the risks posed by deepfakes and introduce architectures that can be used for their detection and discovery. In Chapter 4, we present a baseline architecture and several other architectures we have proposed for deepfake detection. We also list the datasets we used for training and evaluation and present the detection pipeline. Chapter 5 is then devoted to a summary of our experiments on the architectures and their evaluation, followed by a discussion of these problems and other related issues. The last Chapter 6 summarizes the whole thesis.

Chapter 2

Neural networks

Neural networks are a concept used for both false-content generation and detection. This chapter will describe the basic principles of neural networks and divide them into several subcategories. We will then work with the different types of networks in the following sections.

2.1 Technical background

The architecture of networks of neurons in the human brain inspires neural network algorithms. These algorithms use idealised neuron models. The fundamental principle is that artificial neural networks learn by modifying the connections between their neurons to perform many information-processing tasks [41].

Neural networks are non-linear models for predicting or generating content based on a general input. These networks comprise neurons connected to layers connected sequentially via synapses. These synapses have weights, generally defining concepts learned by the network model. A forward-propagation process executes the network on an n-dimensional input x. Forward propagation refers to calculating intermediate variables for the neural network from the input layer to the output layer with an activation function (ReLU, Sigmoid, etc.) to summarise the neuron's output [36, 42].

Neural networks are defined by Mirsky et al. [42]. Let $l^{(i)}$ denote the i-th layer in the network M, and let $\left\|l^{(i)}\right\|$ denote the number of neurons in $l^{(i)}$. Finally, let the total number of layers in M be denoted as L. The weights which connect $l^{(i)}$ to $l^{(i+1)}$ are denoted as the $\left\|l^{(i)}\right\|$-by- $\left\|l^{(i+1)}\right\|$ matrix $W^{(i)}$ and $\left\|l^{(i+1)}\right\|$ dimensional bias vector $\vec{b}^{(i)}$. Finally, we denote the collection of all parameters θ as the tuple $\theta \equiv (W, b)$, where W and b are the weights of each layer, respectively. Let $a^{(i+1)}$ denote the output (activation) of layer $l^{(i)}$ obtained by computing $f\left(W^{(i)} \cdot \vec{a}^{(i)} + \vec{b}^{(i)}\right)$, where f is often the Sigmoid or ReLU function. To execute a network on an n dimensional input x, a process known as forward-propagation is performed, where x is used to activate $l^{(1)}$ which activates $l^{(2)}$ and so on until the activation of $l^{(L)}$ produces the m-dimensional output y.

To summarise, we consider the network M as a black box and denote its execution as $M(x) = y$. If we want to use supervised learning, we need to define an objective loss function $\mathcal{L}$ and a dataset of paired samples with the form (x_i, y_i). The loss function generates a signal at the output of M that is bask-propagated through M to calculate the error of each weight. Then an optimization algorithm, such as gradient descent or Adam optimizer, is used to update the weights for several epochs.

Some neural networks use techniques such as one-shot or few-shot learning, which enables a pre-trained network to adapt to a new dataset X' similar to X on which it was trained. Two common approaches for this are to perform a few additional training iterations on a few samples from X' and pass the information on $x' \in X'$ to the inner layers of M during the feed-forward process.

2.1.1 Loss Functions

As Mirsky et al. [42] stated, The loss function must be differentiable to update the weights with an optimization algorithm, such as gradient descent. Many loss functions can be applied in different ways depending on the learning. For example, when training M as n-class classifier, the output of M is the probability vector $y \in \mathbb{R}^n$. To train M, we perform forward-propagation to obtain $y' = M(x)$, compute the cross-entropy loss ($\mathcal{L}_{CE}$) by comparing y' to the ground truth label y, and then perform back-propagation and to update the weights with the training signal. The loss $\mathcal{L}_{CE}$ over the entire training set X is calculated as

$$\mathcal{L}_{CE} = -\sum_{i=1}^{|X|}\sum_{c=1}^{n} y_i[c] \log\left(y'_i[c]\right) \tag{2.1}$$

where $y'[c]$ is the predicted probability of x_i belonging to the c-th class.

Other popular loss functions used in deepfake networks include the L1 and L2 norms $\mathcal{L}_1 = |x - x_g|^1$ and $\mathcal{L}_2 = |x - x_g|^2$. However, L1 and L2 require paired images and perform poorly. No further detailed analysis of the theory of loss functions is needed for this work.

2.2 Convolutional neural network

Convolutional neural networks (CNNs) [43] primarily focus on processing input as an image. Thus, by default, CNN neurons are organized into layers with three dimensions representing the image's height, width, and depth representing the activation volume. These dimensions are then reduced using the inner hidden layers, which are then reduced up to $1 \times 1 \times n$ shape, where n represents the number of possible outputs of the classifier.

Figure 2.1 shows a representation of a simple CNN. This network contains several different layers: input layer, convolution layer, pooling layer, fully-connected layer and output layer. We can divide these layers into two categories, feature extraction and classification. The input layer specifies the given dimensions of the input image. The image is then transformed by the convolution layer using the learned weights. The pooling layer then reduces the dimensionality of the image in an attempt to preserve its parametric representation. This information is then classified using fully-connected layers and assigned to a particular output category n according to which classification categories the input image satisfies. The following equation could express this relationship between layers:

$$IN => [CONV => POOL?] * M => [FC] * N => \text{ OUT}, \tag{2.2}$$

IN denotes the input layer, $CONV$ represents the convolution layer, $POOL$ represents the pooling layer, FC represents the fully connected layer, and OUT represents the output layer. M and N are integer numbers, $*$ means repetition and ? means optional. The activations are not mentioned, but the activation always follows the $CONV$ and FC layers [45].

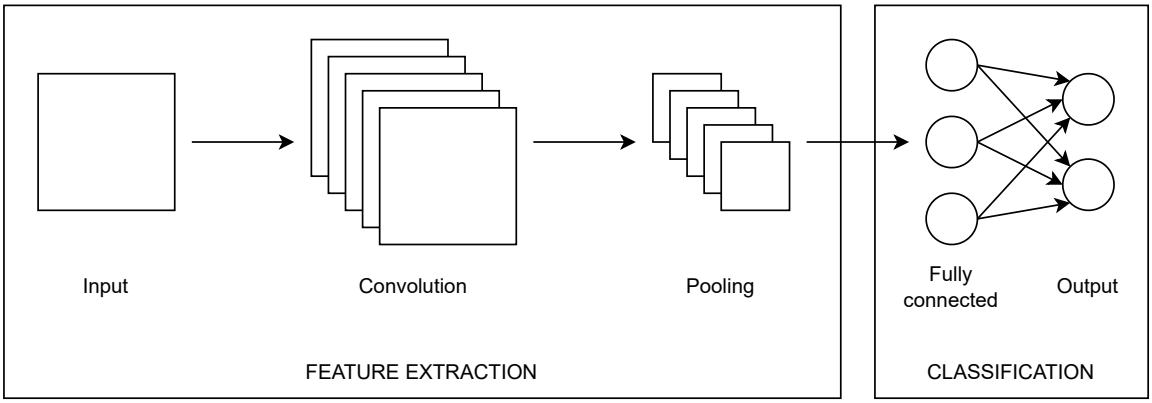

Figure 2.1: Architecture of simple CNN [27].

2.2.1 Convolutional layer

The convolutional layer plays a crucial role in how CNNs operate because the layer's parameters focus on using learnable kernels. These kernels are usually small in spatial dimensions but extend along the entire depth of the input image. When the data reaches the convolution layer, the layer convolves each filter through the spatial dimensionality of the input to create a 2D activation map. While scrolling through the input image, we calculate the scalar product for each value of these kernels. Using this process, the network learns which kernels to use when detecting certain features at specific positions in the input image. This process is called activation by default. Each kernel has a corresponding activation map that folds along the depth dimension to form the entire output from the convolutional layer. Convolutional layers can effectively reduce the complexity of the model by optimizing the output using the hyperparameters depth, stride and zero-padding.

The number of neurons sets the depth of the output produced by the convolutional layer in the layer corresponding to a particular part of the image. By reducing this hyperparameter, we can minimize the number of neurons in the network, but this implies a reduction in the recognition capability of the model.

We can also set the stride, which determines the depth in the spatial dimension of the entrance through which we place the reception field. If we put the stride low, for example, to a value of 1, we will get a high overlap of the receptive field that would create very high activations. Therefore, it is necessary to set the stride value to a higher number to reduce this overlap factor and thus achieve an output with fewer spatial dimensions.

Zero-padding is a simple process of filling the input boundaries, and it is an efficient method to control the dimensions of the output volumes further.

Using these techniques, we can effectively modify the spatial dimensionality of the output of convolutional layers. The formula can express this modification:

$$\frac{(V - R) + 2Z}{S + 1} \tag{2.3}$$

V means the input volume size ($height \times width \times depth$), R means the receptive field size, Z is the amount of zero-padding set, and S refers to the stride. If the result of this equation is not equal to a whole integer, then the stride is incorrectly set, as the neurons will be unable to fit neatly across the given input. Despite these optimization criteria, the network for real image processing is huge. Therefore, methods have been developed for

truncating the set of parameters in convolutional layers. One of these methods is the so-called parameter-sharing. It works with the assumption that if a region feature is useful for computation in a certain spatial region, it is likely to be useful in another region. Thus, if we assign equal weight and bias to each activation map, we can greatly reduce the parameters produced by the convolutional layer. As a result of this process, we get a state where, in backpropagation, each neuron in the network represents the overall gradient over its depth, so it sets the adjustment of the set of weights rather than each weight separately.

2.2.2 Pooling layer

The pooling layer aims to reduce further the dimensionality of the parameters of the internal representation of the input in the network and thus reduce the computational complexity of the resulting model. The pooling layer uses the so-called „MAX“ function for dimensionality reduction, which applies to each input activation map. This layer is often used as the max-pooling layer with a kernel dimensionality of 2×2 applied with a stride of size 2. These parameters are not the rule but rather the most common approach. This setting reduces the input dimensionality to 25 % with depth preserved.

Due to the nature of the pooling layer, there are mainly two types of max-pooling. One is the one already mentioned, where the values are set to 2×2 parameters with a stride of 2. Thus it is possible to cover the entire spatial dimensionality of the input. But also, overlapping pooling is used, where the stride is set to 2, and the kernel size is set to 3. Increasing the kernel size above 3×3 usually leads to a decrease in the performance of the resulting network.

CNN can also contain so-called general pooling. General pooling layers consist of pooling neurons capable of performing many common operations, including L1/L2-normalization and average pooling. We will not discuss them in detail, as they are not essential for a basic explanation of the process of CNNs.

2.2.3 Fully-connected layer

The neurons in the fully-connected layer are directly connected to the neurons that represent the neural network's output and thus determine the values in the output neurons of the CNN by evaluating the weights. Between these layers, there is no intermediate layer to modify the weights.

2.3 Image synthesis networks

Existing state-of-the-art models are mainly built on deep neural networks, showing the extraordinary capability of synthesising new images. Popular categories of models for face manipulation are generative adversarial networks (GAN) and auto-encoders (AE) [31]. We also describe the architecture of the U-net network, which is used for localization in classification tasks.

2.3.1 Auto-encoders

This approach uses two pairs of encoder-decoder [40]. The encoder is used to extract the latent features from the original image, and the decoder is used to reconstruct the face. It requires two pairs because it is needed that each encoder-decoder pair is first trained on the source and then the target image. Then decoders are swapped, so the original encoder

of the source image and decoder of the target image is used to generate new images from the target image with source image features. The technique is shown in Figure 2.2. This approach is mainly used for face-swapping techniques to transpose a victim's face onto someone else's features while keeping the original facial expression.

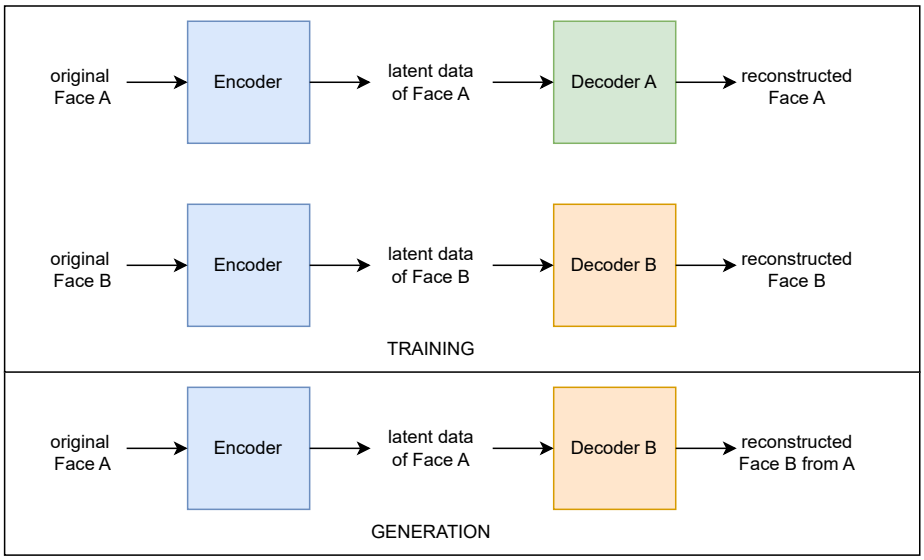

Figure 2.2: Creation of DeepFake using encoder-decoder pairs [40].

2.3.2 Generative adversarial networks

As stated by Goodfellow et al. [23], generative adversarial networks are a framework for estimating models where we simultaneously train two networks. One generator network captures the data distribution, and the discriminator network forecasts the probability that a sample came from the training data. Then the training procedure for the generator network is to maximise the likelihood of the discriminator network making a mistake. The training process is finished when the discriminator network can no longer see the difference between generated and actual samples. After training, the discriminator network is removed, and only the generator network is used. This process is shown in Figure 2.3.

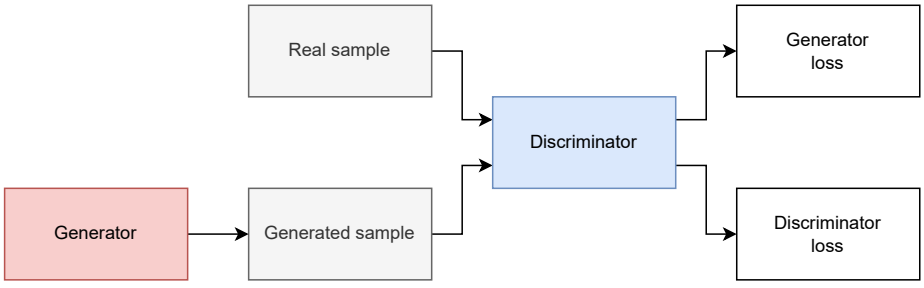

Figure 2.3: Ilustration of generative adversarial networks structure [24].

2.3.3 U-net architecture

For classification tasks, convolutional neural networks are normally used (described in Section 2.2). Still, some tasks require classification and localization of the classified segments in the input image.

Early versions of this kind of network worked on the sliding-window principle, where a local region was created around each pixel, and then its classification was evaluated. This approach is computationally intensive because it is necessary to run the network prediction for each pixel separately. At the same time, having a much larger number of training data relative to the training images is necessary. This approach also adds a significant proportion to the robustness of the network because it is necessary to have many more max-pooling layers.

In the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et at. [47] improved architecture based on a „fully convolutional network“, which provides higher accuracy while requiring less training data. The main idea is to replace pooling layers with upsampling layers. This approach can learn to produce more accurate output images in high resolution since upsampling layers contain a high number of feature channels. The upsampling part is essentially symmetric to the main contracting part, hence the name of the u-net architecture, as it is formed in the letter U. The architecture also does not turn off any fully connected layers and therefore ensures a reduction in the number of trainable network parameters and shortens the training time.

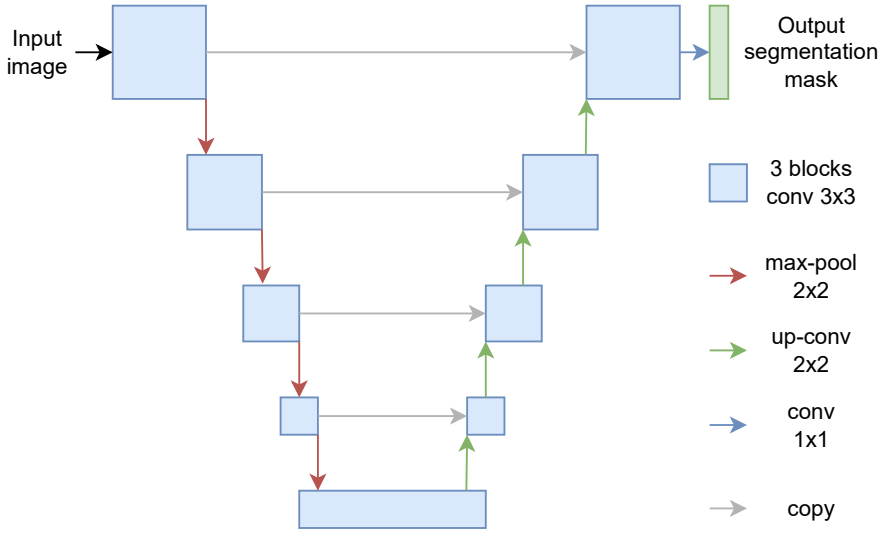

Figure 2.4: Ilustration of U-net architecture [47].

The standard U-net architecture can be seen in Figure 2.4. It is normally divided into left contraction and right expansion parts. The left cast is, by rule, a standard convolutional network and thus consists of a 3×3 repeated application of convolution, each followed by a rectified linear unit (ReLU) and a 2×2 max pooling operation with stride 2 for downsampling. At each step during downsampling, the number of parameters is doubled. Each step in the expansion part again consists of upsampling followed by a 2×2 convolution that halves the number of parameters. In this part, the concatenation with the corresponding part from the contraction part is also important, followed by two 3×3 convolutions and ReLu activation. The final layer then uses 1x1 convolution to map the feature vector to the

desired number of classes. In this architecture, it is necessary to choose the input image size so that each 2x2 max-pooling operation is applied to a layer that has even x and y sizes.

The training can then be done with the same number of input images and their associated desired output masks. However, the authors state that it is preferable to use high-resolution images for training and not to divide them into batches for efficiency on the graphics. At the same time to ensure high training momentum to set fast training. Here we assume that this is more a setting directly for the task for which this network was designed, i.e. Biomedical data. In other cases, we believe deeper experimentation with the network's sensitivity to overtraining is needed.

Chapter 3

Deepfake

DeepFake is a technique to synthesise or modify image/video and audio recordings with deep neural networks based on the creator's will without actual interaction with faked object [42] or person. In recent years, the concept of DeepFakes gained popularity in various ways, but most in manipulating people's faces and voices. Nowadays, to create DeepFake, you only need to open your smartphone and download the application that makes some funny DeepFake for you. Some examples of popular applications for smartphones are Dawn AI [53] or iFace [3]. Dawn AI takes a few pictures of the person and creates a virtual avatar with some thematics like some mythical creature or astronaut or only modifies your hair colour similarly. iFace is an application that works on a face swap technique (which will be explained later). It takes a selfie picture of the person and inserts it into a short video from some popular movie. More approaches to DeepFake creation will be explained further in the text.

3.1 What is a Deepfake

Deepfakes are AI-generated media that reproduce made-up events often authentic to the human eye. The term DeepFake is only a slang term with no agreed-upon technical definition. This word combines 'deep learning' and 'fake', commonly referring to audio or video of a person doing or saying things they never actually did or said generated by an artificial neural network [4, 42].

DeepFakes are created with a deep learning method that relies on a complex computing system called a deep neural network modelled on the biological brain. At first, the network takes training data samples of the targeted person and then uses an algorithm to extract mathematical patterns from this data. Synthetic data of the targeted person are then generated based on these patterns.

Commonly known types of DeepFakes are Face-Swap, Puppet-Master and Attribute-Change. Face-Swap uses multi-scale architecture convolutional neural network (CNN) to paste faces from one image to another. Puppet-Master is a technique that manipulates lip shape. It creates a fake version of the video with the speech of the targeted person using target audio. Attribute-Change generates detailed and continuous facial expression transformations [32]. Technical details about these techniques are explained in further sections.

While commonly discussed types of DeepFakes aim to mimic real people, deep learning is also used to create entirely fictional objects or people. Deep neural networks can generate

synthetic images of non-existing people, things, animals or art-like creations. DeepFakes are a subset of synthetic media. This category includes all AI-generated images, video, audio, and text formats [4].

3.2 State-of-the-art GAN models

Our primary focus in this thesis is generative adversarial neural networks (GANs). These networks have a wide range of representations in deepfakes generation, which we will introduce several in this section. They can produce very realistic images (and not only of people) that are hardly recognizable from reality, and the development of these models is still in progress. They are also the basis for datasets that we will use later.

3.2.1 StarGAN v2

StarGAN [8] is one of the first models capable of the mappings between a large spectrum of features, like skin tone, hair colour, eye colour, etc., with a single generator. The generator transforms an image into a corresponding domain by taking a domain label as an additional input. One of the main problems with StarGAN is deterministic mapping per domain. However, data distribution is far more complex, so it brings problems with predetermined labels.

Choi et al. [9] stated that StarGAN v2 improved this approach and can generate images across multiple domains. Genuine domain labels are replaced with domain-specific codes representing specific domain styles. An improved mapping network transforms random Gaussian noise into style. A new encoder is then used to extract the style from reference images. With this technique, an improved model can synthesise diverse images over domains.

Figure 3.1: Morphed pictures based on the male character with StarGAN v2. Face bottom left is the source shape for feature extraction. The top row shows the shapes from which the pose shapes were selected and their basis for the generated images. The bottom row (except for the image on the left) shows the result of the generation.

We have tried the official TensorFlow implementation of StarGAN v2[1], and produced results are pretty impressive. One of the pretrained models is trained on the CelebA-HQ dataset [33], which consists of 30 000 high-resolution images and can pretty realistically morph features of targeted people. Figure 3.1 and Figure 3.2 show results from image generation.

Figure 3.2: Morphed pictures based on the female character with StarGAN v2. Face bottom left is the source shape for feature extraction. The top row shows the shapes from which the pose shapes were selected and their basis for the generated images. The bottom row (except for the image on the left) shows the result of the generation.

The demonstration on an Animal Faces-HQ dataset (AFHQ) dataset [9], which had been rebuilt using high-quality resize filtering. The pretrained model used animal faces for training, where the network also showed remarkable results, as shown in Figure 3.3.

3.2.2 StyleGAN v2

StyleGAN [34] focuses on designing a generator that tries to control the image synthesis process as much as possible. The framework attempts to adjust the style of image input at every convolutional layer, which it uses to control the power of the images features specifically. It also uses noise injected directly into the network, which leads to unsupervised high-level attribute separation in synthesised images. The generator ingrains the latent input code into a latent space, significantly affecting how the various factors are represented in the network. Nevertheless, it doesn't modify the discriminator or loss function in any specific way.

Newer StyleGAN v2 aims to fix characteristic artefacts and improve the quality of generated images. As Karras et al. [35] stated, version 2 has improved the architecture and training process to eliminate primarily two main issues. At first, blob-like artefacts are displayed in generated images, and second, artefacts are related to progressive growth. The new design also allows for generating higher-resolution images than the previous version.

[1] https://github.com/clovaai/stargan-v2

Figure 3.3: Morphing pictures based on the animals with StarGAN v2. Face bottom left is the source shape for feature extraction. The top row shows the shapes from which the pose shapes were selected and their basis for the generated images. The bottom row (except for the image on the left) shows the result of the generation.

Figure 3.4: StyleGAN v2 generated images that achieve high quality in the eye of the observer.

We used StyleGAN v2 official repository[2] to demonstrate the functionality of this network. They provide a pretrained model trained on the Flickr-Faces-HQ (FFHQ) dataset [35], consisting of 70 000 high-resolution images of people of different ages, ethnicities and backgrounds. Figure 3.4 demonstrates high-quality performances from the same seed image.

At last, as shown in Figure 3.5, not every complete transformation produced by StarGAN v2 is optimal. At some point of truncation, images look glitched or touched with some abstract artistic concept.

[2] https://github.com/NVlabs/stylegan2

Figure 3.5: StyleGAN v2 generated images with unusual outcomes.

3.2.3 GHOST: Generative High-fidelity One Shot Transfer

The GHOST [25] project brings the GAN solution also to the face-swap problem. It mainly focuses on improving known problems in deepfakes, such as eye gaze inconsistency, face edge errors and others. The FaceShifter [37] model is chosen as the baseline for the architecture, which tries to improve the solution for these known errors. The model consists of three main components as shown in Figure 3.6, where X_t and X_s represent input images (or a pair of video images), and Y_st represents the output from the model.

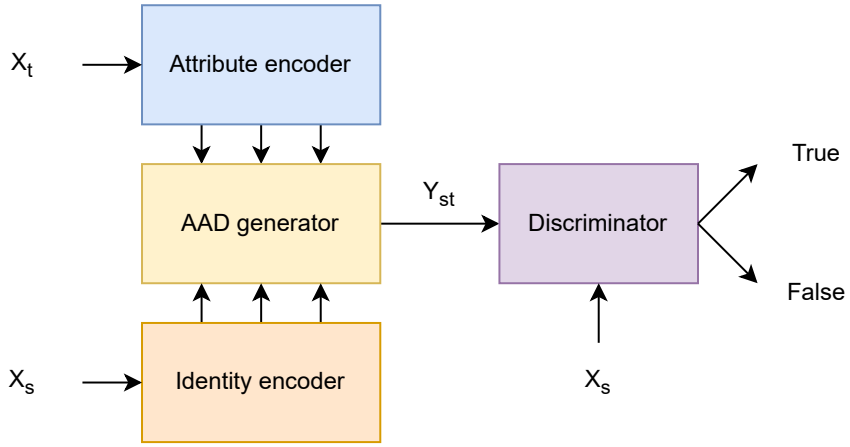

Figure 3.6: Ilustration of GHOST architecture [25].

The identity encoder is implemented using the ArcFace model, which extracts a vector of size $1x512$ from the input image X_s, in which it tries to preserve information about the source of the person's identity.

An attribute encoder is a model built using U-net architecture which extracts features from the target image.

Figure 3.7: Example of using GHOST for image-to-image transition. With source (left), target (middle) and result (right).

Figure 3.8: Example of using GHOST for image-to-image transition in B&W. With source (left), target (middle) and result (right).

The AAD generator is a model that mixes the identity vector evaluated from Xs and the attribute vector evaluated from Xt using AAD ResBlocks during the creation process and generates a new face Y_st with the source identity and target attributes.

Next, the discriminator functions as a standard discriminator in the GAN architectures we described in Section 2.3.2, which tries to decide whether the resulting generated output is real or fake, thus improving the network training process. The solution also comes with an improved loss function in training for better network results.

GHOST has shown remarkable results during testing. It can work with two images and plant a shape into the whole video sequence using one photo, often with very high precision. Of course, there are cases when it doesn't do this very well because it is a generic model that works with one photo as its input. Figure 3.7 and Figure 3.8 show examples produced by this model.

Figure 3.8 shows that GHOST effectively incorporates not only the face into the image but the face into the overall scene, where it works very well with making a face fit in the environment whether by quality, colour, style and other factors that belong to it.

When testing on video, we captured a few frames where the model had a problem, for example, passing the hand in front of the shape. These are specific cases that almost every such model has problems with. Still, in the case of using this kind of deepfake, the material maker can adjust the source and target images/videos according to the model so that these glitches don't happen to him.

3.3 Deepfake threads

The types of deepfake attacks can be of different natures. It can be attacked on specific persons to tarnish their name or to dehumanise them, as attacks on individuals in general. Still, it can also be global attacks that will influence mass opinions. We describe various attack scenarios in this section. We have focused on general attacks using deepfakes without segregating the technology used or the user's expertise because, nowadays, practically everyone has access to this technology in their pocket.

3.3.1 General attack vector

Although many scenarios exist of how a deepfakes attack can be executed, most rely on the same structure. We have described the attack procedure in the same way as Brooks [5] stated.

1. **Intent** - The attack starts when the attacker who wants to use deepfake decides to execute an attack on the selected target according to the chosen scenario.

2. **Researching** - The attacker needs to research his desired target, which in principle means that he needs to obtain as much material as possible containing image or audio-video recordings of the victim. The type of material may vary depending on the scenario chosen and also on how much close contact the attacker has with the victim. For example, in the case of Cyberbullying, this step is often much easier than in Deepfake kidnapping.

3. **Training model** - In this step, the attacker needs to use the collected data to train a model suitable for the form of his attack. This step is highly dependent on the attacker's knowledge and resources. The type of training and technology used depends on the time spent on this step. In many cases, using commercially available applications where the required resources are insignificant is possible.

4. **Media creation** - After training the model, the attacker creates the desired deepfake according to the chosen scenario. This step can be repeated because the quality of deepfakes can vary depending on the requirements and may not always be sufficient. The last two steps do not always have to be performed by the attacker alone. If the attacker is not experienced or does not have sufficient resources, he can pay for such a service.

5. **Distribution of the deepfake** - Then the deepfake needs to be distributed. The way of distribution can be subtly varied according to the chosen scenario. For example, it can be posted on social networks, sent out via email or by establishing communication using a new fake identity.

6. **Viewer response** - Deepfake has reached the target group of users, and their reaction is awaited.

7. **Victim response** - The victim's reaction is awaited. The victim is a specific subgroup of viewers from which a different reaction to a given deepfake is also expected. Depending on the scenario, it may be a statement that it is fake news. Depending on the scenario, the expected performance may differ.

At every step of a deepfake attack, it is possible to put preventive measures in place to prevent this attack. As a rule, each attack belongs to a different category and, therefore, to a different institution that could prevent this attack. Our work mainly focuses on detecting deepfakes in the „Distribution of the deepfake“ step and thus stopping the attack directly before any damage is done because even if the deepfake is detected later, it can cause profit for the attacker and problems for the victim.

3.3.2 Attack scenarios

To better understand how deepfakes work and their impact on security, we summarised a few scenarios of how deepfakes attacks can work. We have divided these scenarios into three categories: national security and law, commercial use, and society, as Brooks [5] stated. We focused mainly on those where it is possible to use image deepfakes. For deepfakes where it is possible to use audio-video format, it is possible to find several other scenarios or to extend those we describe.

National security and law scenario: Deepfake kidnapping

In this scenario, a criminal gang operating in any tourist location with a high crime rate (such as Mexico) can track down a tourist group. Suppose they can evaluate the tourist group as a suitable victim. In that case, they can use deepfake imagery to create a scene where the gang captures one member of their group, or they can upload footage of a member of their group being injured to make the victim more likely to comply with the gang's demands. Then they can use this material to extort money from other group members. Of course, this scenario only works if there is a situation where the group members are not in constant contact, which is not unusual in tourist destinations. In the end, however, the victims are not injured, and the criminal group does not have to make any extra effort to kidnap them.

National security and law scenario: False evidence in a criminal case

Nowadays, in serious criminal offences, such as murder, the prosecution is already pursuing charges against the accused perpetrator using evidence such as DNA tests, fingerprints or various audio-visual recordings and testimonies. In this scenario, we take the perpetrator as an advanced person in deepfakes or wealthy enough to pay for such a procedure. Such a person may be able to use audio-visual recordings to provide false evidence of, for example, his location within a building (or even another building) at the time of the offence. By doing so, he supplies relevant evidence of his innocence, and at the same time, he can discredit other types of evidence against him, such as witness statements from the crime scene. Thus, he does not only help himself by false evidence but also by reducing the relevance of other evidence against him.

This scenario can also be applied in the opposite case similarly. Thus against a person who is accused and is innocent, we prove in this way the perpetrator or another entity that would have an interest in it introduce „relevant“ evidence that shows that the person should be guilty.

Commerce scenario: Corporate sabotage

This scenario sets an option of spreading misinformation to damage the company's reputation, whether it is the overall reputation of the company, the individual products that the company offers, the company's board of directors and others. Damage to the company's reputation can occur, for example, when some of its top executives or well-known figures are falsely accused, which can negatively affect the company's current and future collaborations or their prices on the stock market.

Commerce scenario: Corporate liability concerns

The following scenario works with a high-quality deepfake video rather than a standalone image. We present this scenario because it is also possible to cut the video into a sequence of images and evaluate whether the individual frames are deepfake.

Here we are not working with a direct impact on the company, but rather for a fraud on the client side, where it is then possible to claim monetary compensation from the company. If a company that manufactures a product (e.g. a mobile phone) has a defective product that, for example, explodes, the attacker can use sufficient knowledge to declare his product equally defective and demand compensation for damages. It is possible to trace original videos of these defective products with other customers to whom the damage was caused. The attacker then uses deepfake to create a similar video with his product, simulating the same type of error that occurred in the past. Based on this, he can hang this video on social networks. As this kind of video is gaining popularity extremely fast, the company can be subsequently pressured to compensate the customer, and it is extremely problematic to verify whether this incident happened. The company would also put itself at great risk if it wanted to prove that the video in question was not genuine, as it could nevertheless cast a bad light on a company that does not want to compensate its injured client.

Commerce scenario: Stock manipulation

In this scenario, the attacker impersonates several high-ranking company employees and, based on their announcements, can manipulate the company's price and make a financial profit. The attacker then creates several profiles on social networks that present themselves as company employees. These may be real employees or fictitious persons who only pretend to be company employees. The attacker then posts an announcement on the fictitious profile of a high-ranking employee (e.g. the CEO of the company), which will affect the company's future share price. He can buy the company's shares if the announcement is positive before the attack starts. If it is rather negative, he can wait for a subsequent fall in the share price, which should probably be corrected in some way after the fake news is revealed. This claim can then be supported on other employee profiles created to increase the relevance of the announcement.

The attack can then be further enhanced by either an audio or audio-video deepfake, where the announcement is made verbally on video. This way, we can increase the relevance of the information even more. The attacker not only benefits himself but also destroys the company's good name and harms other investors in the company. Therefore, it may not be purely an attack where the aim is financial gain but also to harm another person or group.

Society scenario: Cyberbullying

This scenario assumes that the attacker wants to harm the target person exclusively. It is, therefore, not some form of threat for financial enrichment but rather to lower the victim's reputation in a particular group, such as family, work, etc. It may also have the secondary effect of benefiting the victim's rival. A standard method of this attack is to create a picture or video of the victim in a situation that is against the policies of the group in which the person is a member, which will result in, for example, a ban from attending the group's events or a complete exclusion from the group where the victim is located.

Cyberbullying is a form of assault that most often occurs in young people due to the influence of social media. As young people tend to congregate in large numbers in similar groups on platforms such as Instagram, Tiktok and others, it is much easier to spread such fake photos or videos in these groups. Adding to the impact is that if the picture is well taken, even later proof that it is a deepfake can cause permanent damage to a person in the collective precisely because it is a sensation that the whole group is addressing. Still, then it does not matter later on. Another important factor in this issue is also the easy accessibility to the tools for creating deepfakes, which we have discussed in previous chapters. Thus, it is extremely easy to create a deepfake, even for a primary or secondary school student. This deepfake does not have to be extremely high quality, as the general public and experts in this field do not fully judge it.

Society scenario: Pornography

The issue of deepfake pornography has been dealt with more extensively in Section 3.4.3, so this attack scenario, despite its extreme seriousness, will be described only briefly. A common scenario for creating deepfake porn is an ex-boyfriend who cannot get over a break-up with his partner. Since couples usually have a lot of mutual images and videos, it is easy for an attacker to train a neural network from enough material to create such a deepfake. The attacker can then demand that the partner stays in the relationship. If she does not comply, he publishes this deepfake on the Internet. This can be pornographic photographs or videos using the face-swap technique. Since this kind of content is available on various portals, finding a person who, at first glance, resembles the partner so that the deepfake is as believable as possible is not a problem.

Society scenario: Election influence

In this scenario, we are working with influence on the political scene. Let's imagine a situation where candidate A is running for the AP party, and candidate B is running for the BP party. The BP party can create fake news to boost its candidate, using an audio-video deepfake of candidate A to damage his reputation and directly increase its candidate's chances. They can also use forms of deflection, as discussed in the section above (cyberbullying or deepfake porn), to harden candidate A for his potential voters and thus directly attract voters to the side of their candidate. The cases mentioned in Section 3.4.3 work with the same scenario.

Society scenario: Child predator

One of the most serious scenarios with deepfakes is their use against underage children. The attacker, who communicates with a small child, can use deepfakes to create his own

identity as a small child. It is, therefore, easier for him to communicate with the child, as the victim thinks he is dealing with a child of his/her age and not with an adult. Then he can find out various things about the background of the child and his parents, influence him, in the worst case, he can meet him at a dangerous place and possibly kidnap or hurt him. This scenario is one of the worst because the category of victims is often not mature and educated enough to recognize whether they are communicating with a child or an adult. If this attack succeeds, only one last element can prevent this from happening: the timely input of parents, which is often problematic in the online space.

3.4 Incident reports

For a closer approach to the theoretical scenarios in the last section, we will present several reported real incidents divided into subcategories.

3.4.1 Passport morphing attack

Biometric face recognition is widely used in border control applications to recognise individuals based on facial characteristics. The face reference image stored in a passport or other identity documents strongly connects the biometric reference and the document holder. Face recognition systems must be tolerant against intra-subject variation. Morphing attacks can exploit this tolerance bound [17].

This type of attack was first demonstrated by Ferrara et al. [16] and later confirmed by several research works. Two commercially available face recognition software were used with modified parameters according to the European standards used at border controls of European countries. To our surprise, even nowadays, there are countries where the photo of a person for a passport is not taken at a government office, but the person takes the picture at home and only brings it when arranging a passport.

Behaviour

Ferrara et al. [15] stated that if a morphed image similar to the face of two people can be included in a Machine Readable Travel Document (eMRTD), then two persons can share the document. In this scenario, a criminal can exploit the passport of a collaborator without criminal records to overcome security controls. The subject with no criminal records could apply for an eMRTD by presenting the morphed face photo; if the picture is not noticeably different from their face, the officer accepts the image and releases the document.

The conditions for a successful attack are these two.

- The photograph is sufficiently similar to a person applying for a document, and it is possible to deceive a human expert who can see through it.
- A Face Recognition System can successfully recognize a photograph (FRS) in the case of both human subjects.

Several studies have confirmed the high realism of the generated images and their ability to deceive even a trained human worker. This problem becomes all the more severe because attackers often use the small size of the photographs used for passports, typically 3.5cm x 4.5cm, which helps the attacker. After all, the small size of the picture and its printing can effectively hide the flaws caused by the morphing process [46].

3.4.2 Fictional accounts

The Internet and social networks are undoubtedly the strong sides of this time, whether we look at it from the point of view of communication, advertising, influencing people, etc. People use real names or pseudonyms on social networks like Twitter or Instagram. They communicate with their friends, fans or members of multiple groups according to their preferences. Therefore, there is natural space for various fictitious users in these places.

Today, the social network Twitter has switched to a new blue badge model [52], where all a user has to do is pay for a subscription, which costs a few dollars a month. They automatically become „verified“. Previously, it worked differently because a person who wanted this verification badge had to meet a few problematic conditions. Of course, they had to be a public figure or a company. Therefore, social network users could „look up more“ to those who owned this badge because they were sure they knew who they were communicating with and who was posting their „tweets“.

The article published by the CNN news portal in 2020 [44] beautifully presents the abuse of this power. Andrew Walz calls himself a „proven business leader“ and a „passionate student advocate.“ Walz, a Republican from Rhode Island, is running for Congress with the tagline, „Let's make a change in Washington together,“ or so his Twitter account claimed [44]. This profile was given a badge from Twitter because they wanted to make it easier for people to access relevant content from political candidates ahead of the 2020 elections. There would have been nothing wrong with this approach if it had not been discovered that Andrew Walz did not exist. The Twitter profile was created by a 17-year-old student as a free-time activity during the holidays. He later said that his effort was to test Twitter's election integrity efforts. What is also alarming about this story is not only the fact that Twitter managed to verify a non-existent candidate. This could be seen as a short circuit in processing a large amount of data, and one profile slipped through the cracks by mistake. It is also alarming that many candidates have complained that Twitter could not verify their accounts. Still, despite this, the account of a fictitious person has been verified.

Other media outlets have reported similar findings about fake profiles created by deepfakes. For example, an article by James Farrell [14] includes a report from the LinkedIn job network that detected thousands of fake profiles among its users. In this case, people eventually meet a real person in person and potentially get the job anyway, but that does not change the fact that this is disturbing.

In modern society, dating has also moved mainly to the online world. This is evidenced by the dating apps people use nowadays, such as Tinder, Hinge, Bumble, etc. In these apps, we interact daily with thousands of users of our preferred gender; in the more pleasant case, there is some mutual match and the possibility of some further communication. Unfortunately, even in this case, one cannot be sure that one communicates with a real person.

Social networks and dating apps are full of fictitious characters pretending to be real people. They don't always use deepfake technology to generate their images. Even photos stolen from the internet are often enough for this approach. Deepfakes, however, allow deeper communication with the person on the other side. Constructing a new photo according to the current needs is always possible. One of the easiest ways to use it is to use a bot that automatically communicates with the other party and can later get critical information from him, such as payment card details. A more sophisticated method is to use deepfake to obtain a match and then communicate with a real person. In this way and with

very personal communication, attackers can get, for example, compromising images from a person, which they then use to blackmail that person. „There are hundreds, probably even thousands, of Australians who have gone on to pay this money, and they are still getting blackmailed," as the Foster [21] states. Foster [20] also mentioned a practice known as revenge porn, where men exchange such photos of women from their neighbourhood on the internet, which can lead to psychological, not only financial, damage to health. Xu [59] shows a preview of the possible methods of creating these profiles. Still, it is truncated due to the privacy risks created using the Tinder Kaggle Profile Dataset.

3.4.3 Fake accounts and face-swapping

Deepfake technology can create fictitious persons and add a real person's face to a specific scenery. This approach allows the attacker to upload the face of his victim anywhere he wants, whether it is some way of manipulating the person's reputation, convicting him of a crime he did not commit or even „filming" pornographic content.

Pornographic content

In 2018, researchers at Sensity AI found that nearly 90 % of deepfake clips are non-consensual pornographic clips, predicting that this number will double every six months. In an article by Jennifer Savin [49], the author states that after googling the phrase „deepfake porn", the google search engine lists 57 million results, and the interest has increased by 31 % in the last year. But several approaches are available online to create this kind of deepfake easily and relatively quickly. We have mentioned a few simple applications that would be capable of this in Section 3.2, but this is far from all. Several public repositories of face-swap solutions are available online to create such fake photos or videos. For a more experienced user, this approach is no problem because all you need to do is find a solution of sufficient quality. Savin [49] tells the story of a woman whose photos were found online. She admitted that the only thing she found in the photo as possible evidence of fakery was a weak pixelation in the waist area, which could have been removed with Photoshop if she had made a little effort.

Considering that this category of deepfakes is the most used, several more straightforward approaches exist to create such a fake. There are even online forums where people offer their services to make deepfakes. The Vice portal published an article [30] where they set up a deepfake platform and informed about the prices and possibilities in this industry. As they state, they have communicated with several „content creators" who charge an average of around $30 for creating a video, for which 13 seconds of clear footage of the victim was enough. The attacker even told the journalist that the price would be lower later because he had already trained his network on the victim whose video he had provided. After communicating the order, where the video and the link to the pornographic material were handed over, the resulting picture was taken in approximately twenty-four hours. These forums, therefore, give even attackers without experience or the necessary resources the opportunity to attack for a relatively small fee for the damage they can do to their victim.

Sensity AI also published an interesting article [2] about another kind of deepfakes that works in a specific way. Framework DeepNude requests a photo of the victim, who then uses AI to strip naked. Therefore, it is not precisely a face-swap technique because the photo's exposure should be preserved, and there is no need for several pictures or a shorter video. This framework was sold in 2019 to an anonymous buyer after its success, and its different variations are currently available on the internet in the form of applications or

GitHub repositories. This framework also came with automating the process of creating these deepfakes. According to Sensity AI, several bots on the Telegram platform operate with this framework. From the service provider's point of view, it is a fully automated process. Just visit one of these Telegram groups, upload a request, and within moments the attacker will receive the finished photo for a small fee. Sensity AI reports that at the end of July 2020, more than 100 000 photos were taken by such bots and published, and this number continues to grow.

This approach reflects the increasing tendency to use various deepfake attacks and the need to address this situation.

Political influence

Deepfakes of politicians belong, from a global point of view, to one of the greatest threats we face in this technology. The audiovisual images of politicians and their statements impact the public on a vast scale and shape public opinion, whether national or supranational. They are also easy targets for attackers because they can obtain massive datasets of public speeches, photos and various other recordings through the training process of neural networks.

The Medium [28] portal has summarized several publicly available videos that have been created using deepfakes, among them some useful ones, such as an Indian politician who used deepfakes to translate his speech into several languages or the example of Barack Obama, which serves more as an instructional video to let people know what this technology can do. In our opinion, this video looks very convincing. However, some videos have caused a lot of misinformation.

One of the most famous videos is the deepfake of former US President Donald Trump [6], published by the Belgian Socialist Party. In the video, Donald Trump calls on the states to withdraw from the Paris Climate Agreement, just as the United States will. Although the video was imperfect and presented as a fake, it misled many people into thinking it was an actual speech.

Another well-known example of the use of deepfake is the creation of a video by Volodymyr Zelensky calling on the Ukrainian people, during the current war in Ukraine, to lay down their arms and surrender to the Russian Federation [12]. Zelensky subsequently denied the video and declared it false, but this does not change the consequences the footage could and perhaps did bring. There have also been deepfake videos of Vladimir Putin calling on Russian soldiers to surrender in this war. This war is a clear example of a significant threat, even in such military conflicts, where such an act of aggression can cause the morale of the troops to plummet and turn the situation in favour of the aggressor. In such cases, technology is dangerous because fake news gets into the consciousness of even unknown people and shapes their public opinion.

As Medium points out, few known cases of Deepfakes being used for political purposes exist [28]. However, this fact may be due to the development of technology, and the further we go, the more frequent these cases may appear. The war in Ukraine is a clear example of this, where similar incidents have occurred several times during a few months of fighting. In such conflicts, each side's moral support is important for their victory, not the absolute perfection of the technology, because, despite any efforts, the other side will soon declare that the video is fake.

False accusation of criminal activity

As this technology develops, important issues, such as recording evidence in court trials, are also emerging. In the past, oral testimony or a written signed statement was taken as a conviction or valid proof in a trial. In the development of mobile devices such as telephones, photographs, videos, or audio recordings have also come into use. These make it possible to offer evidence even though, for example, the witness who took a photograph might fear for their life and, therefore, not want to appear in court. With them also comes the possibility of one of the parties using deepfake as evidence or one of the parties challenging the evidence concerning this technology.

The modification of photographs as forensic evidence is nothing new. It did not come with deepfake technology. Digital image forensics mainly focuses on the low level of a picture to verify its credibility. However, this does not change the fact that videos and audio recordings are now being added to this, which will only become more and more sophisticated as time goes on.

The issue is also dealt with in several articles [7, 18, 22].

3.4.4 Summary

In this section, we have shown several theoretical scenarios and, at the same time, practical examples from the real world of how we can encounter abuse through deepfakes in almost every segment of our life, whether it is a personal, commercial, or governmental sphere. It is therefore important that we minimize their bad impact on society and work on prevention against these attacks at every available step.

This work, therefore, focuses on the detection of deepfakes at the moment of their publication and in later steps because prevention in the early stages of the creation and acquisition of the necessary data is an extremely complicated task from the point of view of freedom, where without the restriction of current free rights it is virtually impossible to prevent their creation and subsequent dissemination.

3.5 Deepfake detection

Detection of deepfakes is a direction that is the focus of a lot of research because, as shown in the previous chapter, they are a severe security risk in many ways. In this chapter, we will discuss some basic approaches that are used for deepfake detection from a human perspective, we will discuss the basic principle of deep neural networks that try to solve the problem of image classification and deepfakes recognition, and we will introduce state-of-the-art models that are used to solve this problem.

3.5.1 Human detection

Before using neural networks for detecting deepfakes, standard methods focused mainly on the consistency of the image from a low-level view. In addition to the standard search for artefacts that neural networks often leave in their generated images, there are also standard forensic procedures used in the pre-deepfake era to detect possible photo manipulation, and they are still used today.

One of the best methods of detecting fake images is to examine the fingerprint left by the camera [39]. With good photos captured from the same camera, it is possible to analyse the amount left by the camera and compare it with the amount in the scanned image. It is

possible to determine where the image has been modified or its validity using the differences in this sum. However, this method requires knowledge of the camera equipment with which the photograph was taken and, therefore, cannot be applied unless the camera used to take the image is known. Alternatively, detecting this with a small number of images is impossible.

Another approach is called copy-move forgery [11]. This method analyses the patterns in the image, and similarities between them are looked for. When editing images, this method is often used to fill in missing parts necessary for the resulting image. Thus, we can detect repeating patterning between the individual components of the picture. Unlike the previous approach, in this approach, we do not need to know the device from which the image was taken, and at the same time, we can also process an individual image. It is not necessary to have a series of others for analysis.

However, human observation remains the primary method for detecting deepfakes. GANs often create various kinds of artefacts and inconsistencies that can be seen by looking at the image in more detail. When recognising, it is necessary to pay attention to, for example, reflections in human eyes, differences in eye colours, missing earrings, but also the surrounding environment in the image, which is often distorted by neural networks, even though the generated face itself achieves results indistinguishable from reality.

3.5.2 State-of-the-art neural network models

In this section, we describe state-of-the-art models that can be used for image classification and, thus, deepfake recognition. Their basic idea, architecture, and how these models are trained will be described. Most of these models can be trained from scratch, but they contain a wide range of pre-trained models that can be used for the kind of classification we need using finetuning. We have tested these models from both the finetuning and pure weight training perspectives, and they will be used as reference models for comparing the effectiveness of our proposed solution.

VGG

The basis for VGG models was first described by Simonyan et al. [51] in 2015. The cause of this convolutional neural network is the use of small convolutional filters of size 3×3 with stride and pad of 1, along with 2×2 max-pooling layers with stride 2, which are stretched to depth sizes 16 and 19 in the framework of the proposed network. From this approach, the VGG16 and VGG19 models were subsequently developed. The models are focused on not only classification tasks but also localisation tasks. In the following, we will mainly deal with the VGG19 model because it will be used as one of the reference models for comparing the effectiveness of our solution in Chapter 5.

The basic architecture requires an input image size of 224×244 pixels. It is capable of classifying up to 1000 different objects, such as pens, cars, animals and more, using the pre-trained ImageNet dataset [13]. This approach trains the pre-trained model weights to accurately classify features on different images. The architecture of the model is shown in Figure 3.9.

Resnet and Resnetv2

The Kaiming et al. [26] focused on exploring the depth of neural networks. They observed a degradation problem, which means that the accuracy of neural networks saturates, and then

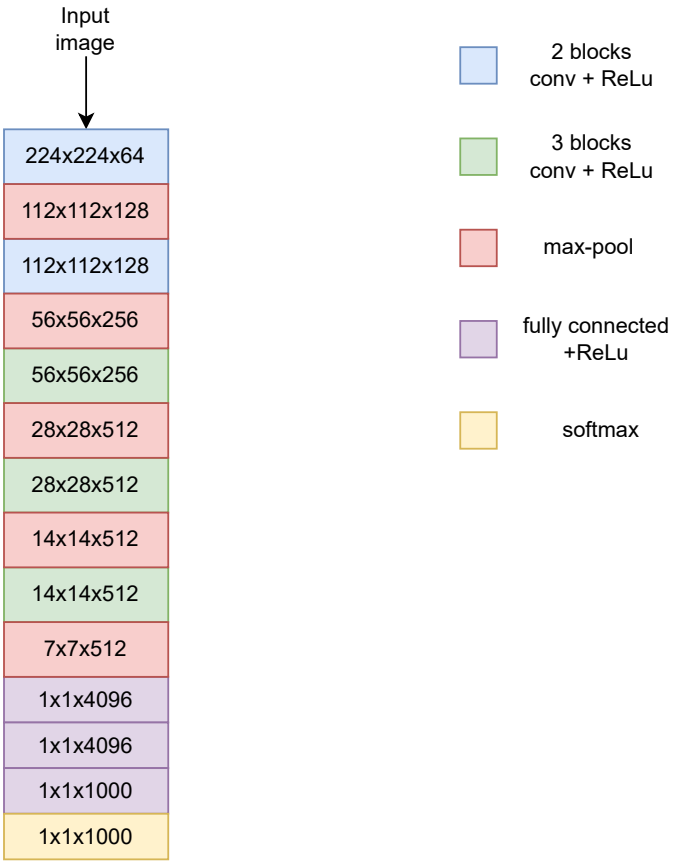

Figure 3.9: Illustration of the network architecture of VGG-19 model: conv means convolution, FC means fully connected [61].

their accuracy degrades rapidly. This degradation does not occur by so-called overfitting, and adding additional layers to the network only causes higher errors in the training process and, subsequently, in testing. The degradation of training accuracy indicates that not all systems are similarly easy to optimize. A possible solution for this problem is the so-called identity mapping, in which the addition of connecting layers to an already existing model occurs. In this process, the error increase is then reduced.

Instead of hoping each few stacked layers directly fit a desired underlying mapping, residual mapping holds these layers explicitly. Formally, denoting the selected underlying mapping as $H(x)$, the stacked nonlinear layers provide another mapping of $F(x) := H(x) - x$. The original mapping is recast into $F(x)+x$. A hypothesis is that optimizing the residual mapping is easier than optimizing the initial, unreferenced mapping. If identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers. Formulation of $F(x) + x$ can be realized by feedforward neural networks with shortcut connections as shown in Figure 3.10.

Shortcut connections are links that omit one or more layers. In this case, shortened links perform an identity mapping, and outputs of identity mapping are added to the outputs of the stacked layers. Identity-shortcut connections do not add any additional parameters or computational complexity. The entire network can still be trained using an optimizer with backpropagation and can be easily implemented using standard libraries without modifying the solvers. The paper's authors declare that despite the great depth

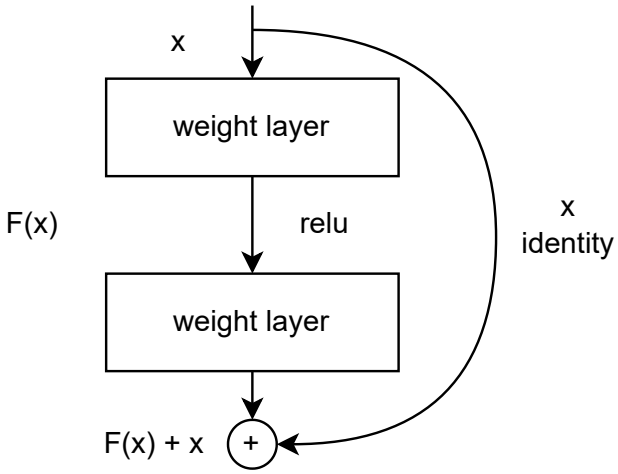

Figure 3.10: Building block of residual learning cited from [26].

of the presented models, their complexity is lower than the VGG models already presented by us.

The ResNet architecture contains 3×3 convolutional layers with a stride of size two that rely primarily on two pillars. First is that for the same output feature map size, the layers have the same number of filters, and second, if the feature map size is halved, the number of filters is doubled to preserve the time complexity per layer.

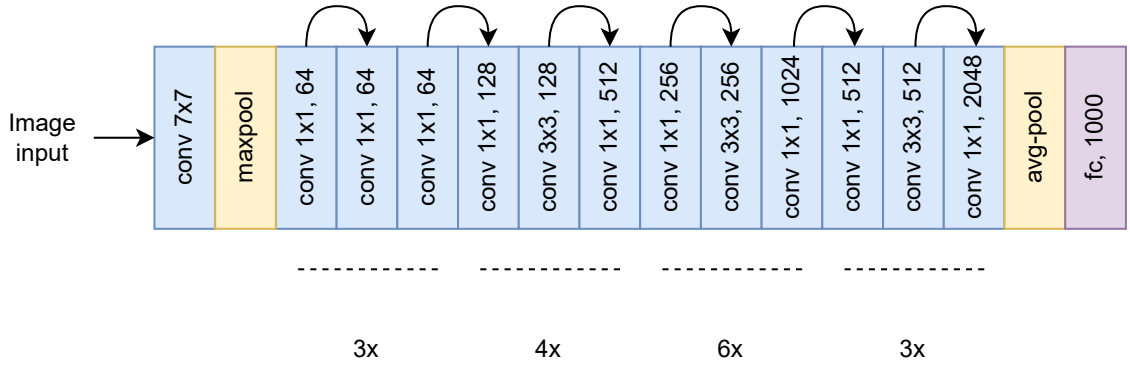

Figure 3.11: Illustration of the network architecture of ResNet50 model: conv means convolution, FC denotes fully connected.

Figure 3.11 shows the architecture of ResNet50. The same skip connections between the individual blocks where the arrows are located are shown in Figure 3.10.

Densenet

The Densenet architecture was introduced later as an extension of the resnet architecture by Huang et al. [29]. The paper's authors argue that even though resnet works on the principle of propagating weights from the previous layer to the current layer, it can still

prevent the flow of information across the network. For this reason, they came up with a new concept called dense connectivity.

This concept is based on the principle that the weights from each layer are propagated to each successive layer and, thus, not only to the next layer. Thus, layer x_n gets the feature map of all its predecessors from x_0 up to x_n, so if we express the layer mapping as a function H, then the computation of this function is

$$H([x_0, x_1, ..., x_{n-1}]) \tag{3.1}$$

where $[x_0, x_1, ..., x_n - 1]$ denotes the concatenation of the feature-maps of layers 0 to $n-1$. The composition function itself is defined as a sequence of three successive operations: batch normalization (BN), rectified linear unit (ReLU) and a 3×3 convolution (Conv). For Formula 3.1 to be functional, the size of the individual feature maps needs to be invariant, which contradicts the sense of down-sampling blocks that reduce dimensionality in neural networks. For this reason, the model is divided into so-called dense blocks, between which transfers are then performed using so-called transition layers, which perform convolution and pooling.

In this kind of architecture, we can also encounter the problem of a significant increase in the number of training parameters since each network block has a massive number of input feature maps. Experimentally it has been proved that due to this propagation of outputs to all blocks, such a large number of blocks is unnecessary to achieve the necessary results. The layout of the densenet architecture can be seen in Figure 3.12.

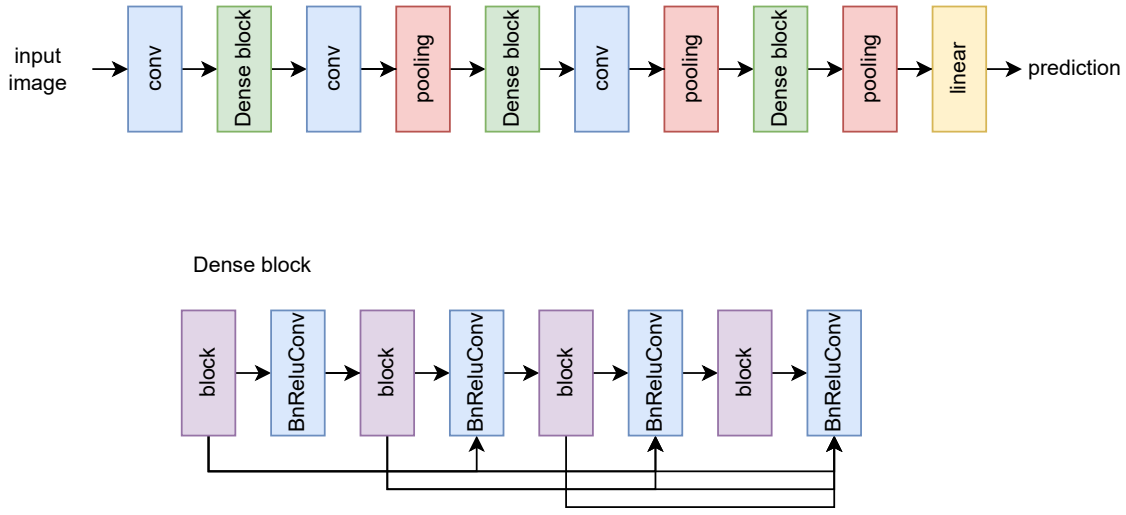

Figure 3.12: Illustration of the network architecture of Densenet121 model: conv means convolution.

Xception

Chollet [10] introduced this network. First, we must introduce the Inception hypothesis that this network uses. The standard convolutional layer tries to learn a filter for mapping in 3D space and mapping cross-channel and spatial correlations simultaneously. The Inception hypothesis tries to simplify and streamline this process that would independently look at cross-channel correlations and spatial correlations. A typical Inception module first looks

at cross-channel correlations via a set of 1×1 convolutions, mapping the input data into 3 or 4 separate spaces more minor than the original input space, then maps all correlations in these smaller 3D spaces via regular 3×3 or 5×5 convolutions. This is shown in Figure 3.13.

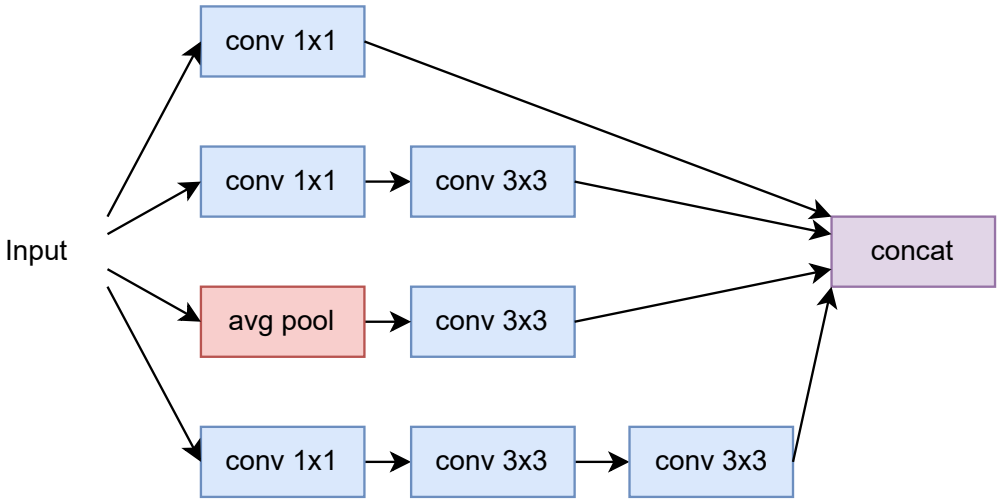

Figure 3.13: Illustration of the Inception block.

Inception blocks can have various forms of modifications. The simplest version of the block is a modification where the channel with average pooling is removed, and the other parts of the network are modified to 1×1 and 3×3 convolution, or there are several different versions of a more complicated block.

The extreme version of the inception module works with the assumption that we first apply a 1x1 convolution to map cross-channel correlations and would then separately map the spatial correlations of every output channel. This block version is almost identical to depthwise separable convolution [50], which is widely used in various neural network implementation frameworks, such as TensorFlow.

There are two significant differences between depthwise separable convolution and the extreme version of the inception module. There is a different order of operations in them. The extreme Inception block performs 1x1 convolution first and then performs channel-wise spatial convolution, while the depthwise separable convolution performs these operations in reverse order. The second difference is the presence or absence of a non-linearity after the first operation.

The Xception architecture works exclusively with depthwise separable convolution layers. This architecture is based on the hypothesis that mapping cross-channel and spatial correlations in the feature maps of convolutional neural networks can be entirely decoupled. This is a stronger statement than the Inception hypothesis, so this architecture is named „Extreme Inception“ or Xception. Xception architecture is a linear stack of depthwise separable convolution layers with residual connections. This makes the architecture easy to define and modify; using a high-level library such as Keras takes only 30 to 40 lines of code.

We do not provide implementation details of this architecture in this paper. The architecture can be considered more complex, and it is unnecessary to describe it in particular for this work. In short, it contains an Entry flow, followed by a Middle flow composed of ReLu and Separable convolution. This block is repeated eight times, followed by an Exit flow which takes care of the inclusion in the predicted class.

EfficientNet

EfficientNet brings significant improvements to the family of convolutional neural networks that focus on adjusting network dimensions, such as its depth or width, when it is necessary to improve the efficiency of the network. To improve the accuracy of neural networks, the robustness of the network is a frequent factor. Tan et al. [55] discuss how to efficiently scale up existing models so that smaller models can be scaled up efficiently, and thus not at the expense of their robustness, which results in a longer training and evaluation process, and also the necessary computational performance.

Since this form of architecture is presented as an efficient extension of other existing models, we will not describe it in detail. However, it represents an approach for making the model training process more efficient, and we will use it in the next part of the thesis, so at least its basic introduction was necessary.

For the next part of the thesis, we will use the implementation of the EfficientNet architecture in Keras based on building blocks that will be discussed in more detail during the experiments.

Chapter 4

Design proposal

In this chapter, we present three architectures that will be used in the experimental part of the thesis. Each of these architectures represents some form of modification of the previous. We will also present the two main datasets we used for training and subsequent evaluation of the models.

4.1 EfficientNet v2

The first basic architecture we built as a baseline for our solution is the implementation of Efficientnet v2 M [56]. This model handles an input of 480 × 480 pixels and is pre-trained on the Imagenet dataset [13] to classify images into 1000 different classes.

The model consists of 7 basic blocks which are connected. Each block is then composed of several smaller subblocks according to the level of immersion. For example, block number 1 contains three subblocks, while the largest block 6 contains eighteen. Figure 4.1 shows a graphical representation of the fourth subblock in block 3, *block3d*. Because of the large size of the network, we have chosen to illustrate only one of these subblocks. The subblocks have subtle differences depending on the subblock nesting, but we think this representation is sufficient for illustration.

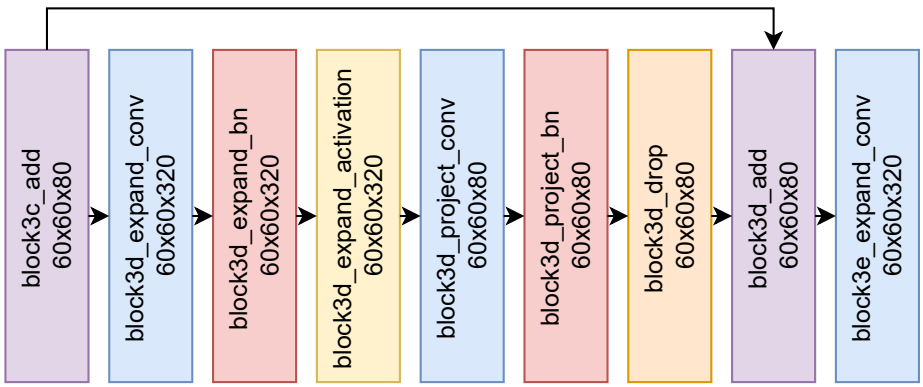

Figure 4.1: Illustration of block3d of EfficientNet v2 M from implementation in keras.

We then modified the network to fit the model for binary classification, which in practice means that we modified the last layers where we added the sigmoid activation function to

set the final value to a range of 0 to 1. 0 in case it is a deepfake, 1 in case the network estimates the input image to be real.

In the case of this architecture, we consider it a baseline in our solution, which we try to finetune as efficiently as possible for our task and then improve with several modifications. These modifications will then be compared with the best results from this model in the same training process.

We tested the EfficientNet v2 L version in the same way in this experiment.

4.2 EfficientDet modification

Tan et al. [57] presented modified efficient net architecture focusing on object detection and subsequent labelling. It uses EfficientNet as its backbone, modifies how it works with outputs, and adds additional connections to the network. It uses so-called BiFPN links to do this. The network then uses feature extraction with this BiFPN network and can predict and box objects in the input image. The design of this architecture is shown in Figure 4.3.

4.2.1 BiFPN

Tan et al. [57] formulate a multi-scale feature fusion problem and then present a solution BiFPN: efficient bidirectional cross-scale connections and weighted feature fusion. Multi-scale feature fusion aims to aggregate features at different resolutions. Formally, given a list of multi-scale features $\vec{P}^{in} = \left(P_{l_1}^{in}, P_{l_2}^{in}, \ldots\right)$, where $P_{l_i}^{in}$ represents the feature at level l_i. The goal is to find a transformation f that can effectively aggregate different features and output a list of new features: $\vec{P}^{\text{out}} = f\left(\vec{P}^{in}\right)$.

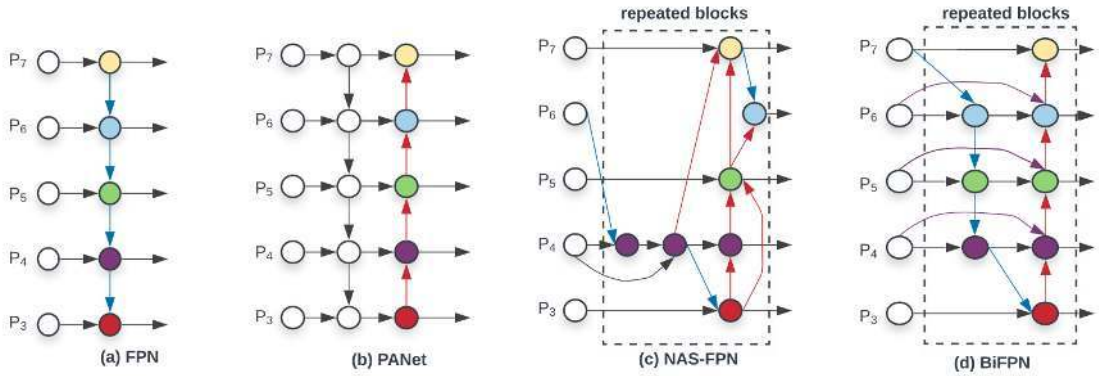

Figure 4.2: Feature network design – (a) FPN introduces a top-down pathway to fuse multi-scale features from level 3 to 7 (P3 - P7); (b) PANet adds bottom-up pathway on top of FPN; (c) NAS-FPN use neural architecture search to find an irregular feature network topology and then repeatedly apply the same block; (d) is our BiFPN with better accuracy and efficiency trade-offs [57].

Figure 4.2(a) shows the conventional top-down FPN. So it takes inputs 3 to 7 from the vector $\vec{P^{in}}$ where P_i^{in} represents the feature level with half the resolution of the input image, i.e. in the case of an input image of size 640×640 pixels, the calculation for P3in is $640/2^3 = 80$, i.e. a resolution of 80×80 pixels. FPN aggregates the feature in a top-down manner

$$
\begin{aligned}
P_7^{\text{out}} &= \text{Conv}\left(P_7^{\text{in}}\right) \\
P_6^{\text{out}} &= \text{Conv}\left(P_6^{\text{in}} + \text{Resize}\left(P_7^{\text{out}}\right)\right) \\
&\dots \\
P_3^{\text{out}} &= \text{Conv}\left(P_3^{\text{in}} + \text{Resize}\left(P_4^{\text{out}}\right)\right)
\end{aligned}
$$

where *Resize* represents the upsampling or downsampling operation for the resolution. Top-down FPN is limited by a one-way transition from up to down. That is why PANet introduced an extra bottom-up aggregation layer, as shown in Figure 4.2(b). NAS-FPN employs neural architecture search to search for better cross-scale feature network topology. Still, it requires thousands of GPU hours during the search, and the found network is irregular and difficult to interpret or modify, as shown in Figure 4.2(c). BiFPN then introduces several architectural improvements. The first fundamental change is removing nodes with only one input. The idea behind this concept is simple since a node has only one input, it should contribute to the overall functioning of the network, leading to its simplification. The second change is to add an extra edge from the original input image to the output node in case they are on the same level. The last change compared to PANet is the addition of multiple bidirectional paths to improve feature extraction as opposed to PANet, which contains only one such path. The architecture is shown in Figure 4.2(d).

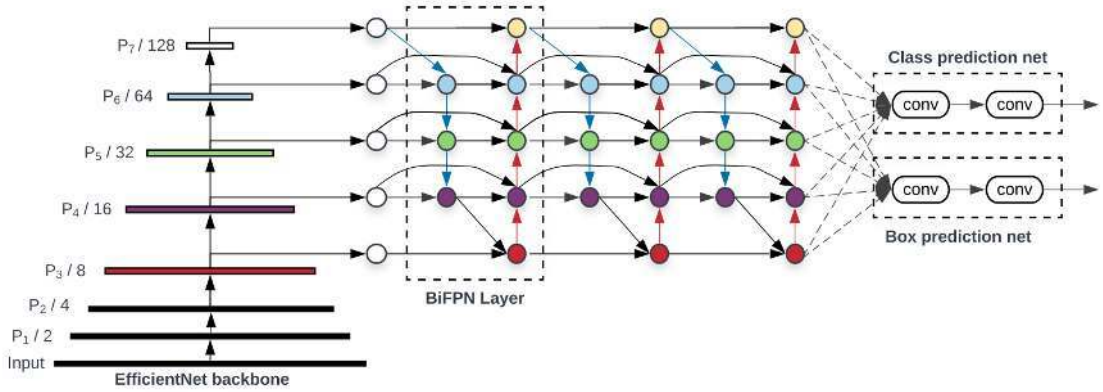

Figure 4.3: EfficientDet architecture diagram with EfficientNet backbone [57].

4.2.2 Proposed architecture

The idea behind our proposed architecture is very simple. Since the extended EfficientDet architecture can perform efficient prediction and boxing simultaneously, it can also serve as a pure prediction mechanism for deepfakes patterns better than the underlying EfficientNet v2 M architecture on which we are based. Therefore, we decided to create a model whose backbone is the aforementioned model with the addition of BiFPN layers to improve the prediction mechanism. Compared to the EfficientDet design, we have removed the part of the network dedicated to boxing since it is unnecessary in our case. Still, it would be interesting to observe such detection in the case of finding artefacts that neural networks leave behind, but we have not investigated this approach. The last block that feeds into the BiFPN layers is block 7, specifically its *block7e_expand_activation* layer, which implies that we have removed the last layers used for prediction from the baseline model and replaced them with BiFPN. Finally, we added a concatenation and dense layer followed by a dropout layer and a sigmoid function for inclusion in the corresponding binary class. We

are also experimenting with a finely extended network design, where we have repeated this procedure twice and added a layer containing batch normalization after the dense layer. So we are experimenting with two different designs, one is basic, and the other is extended.

4.3 EfficientYdet

The basis for this proposal is designed by Tjon et al. [58], which uses the u-net architecture to improve the detection of deepfakes (details are discussed in Section 2.3.3). The basis for this architecture is the connection of upsampling blocks to individual blocks of the convolutional network, thus creating a second segmentation output that creates a localization mask for deepfake. This architecture assumes that incorporating a u-net-like architecture into such a solution can better determine the part of the input image where the manipulated regions are located. While the paper focuses on detecting deepfakes in the video, it takes an approach suitable for our work. It chops the input video into a sequence of images. To speed up the process, 30 images from each video were chosen, assuming the video is 30 FPS and 10 seconds, so 300 frames. Next, shape detection is performed in each image using the MTCNN model [60] and its cropping for the input image to the network. In the case of the training and validation dataset, the same process is performed with the corresponding mask videos (in the case of fake images), and thus not only the classification part of the network is used for evaluation, but also the segmentation of the pixels where a potential deepfake can be found. If the video does not contain a deepfake but a real image, the resulting mask should be black, i.e. without content. The result is a u-net-like architecture called Y-net because the end predictive part of the backbone is preserved, and thus the neural network has two outputs, a classification and a segmentation one, not just one as in u-net.

4.3.1 Proposed architecture

Because of our proposed architecture, we decided to extend our binary architecture for EfficientDet and add a u-net-like segmentation part. The assumption is that BiFPN is also used for introducing segmentation tasks, so such an extension could bring results. The expanding u-net architecture thus uses the outputs from blocks 1, 2, 3 and 5 to introduce upscaling blocks as in the standard u-net we described in Section 2.3.3. It was only necessary to select EfficientNet architectures as backbones whose input resolution is satisfactory, i.e. divisible by powers of 2. This is based on the assumption that the introduction of the upscaling convolution increases the dimensionality upwards again. Hence, we needed to get back to the original input image resolution. We also chose the EfficientNet v2 M and EfficientNet v2 L architectures for the other proposed architectures. We want to observe the immediate improvements (or degradations) when introducing new blocks into the network, not compare completely different models. A slight change from the standard architectures in u-net is that we could also connect the architecture to block7 to get the lowest possible values in the upscaling. This was impossible because we modified the last block and added BiFPN layers, cutting off the rest of the network. Finding a block with a suitable dimensionality to connect to the network was impossible.

It should be noted, however, that the segmentation part of the network has no actual use in automatic evaluation. The segmentation part can help us to understand better the behaviour of a given model and the way it works to support its possible correct training as it adds another input factor, or if we apply it to an actual operation with a human

supervisor, it can in an ideal scenario point out suspicious places that should be more deeply investigated.

4.4 Datasets

Proper selection and evaluation of the dataset are also necessary for this task. Therefore, we selected two datasets, FaceForensics and Celeb-DF, for training and testing. We chose this combination because FaceForensics is a large dataset often used for models related to this task. At the same time, it contains segmentation masks necessary for training in one of our proposed architectures. We have chosen Celeb-DF as a secondary dataset for evaluation to get as close as possible to the practical use of detectors in deepfakes detection. While each dataset contains images that should not be dependent on each other in the case of testing and validation, it is necessary to note that in this area, often only a change of the testing dataset is needed, and the results can change significantly. We used Celeb-DF for a more detailed evaluation of the trained models since it was created using different procedures than the FaceForensics dataset.

4.4.1 FaceForensics

The dataset was introduced in 2019 by Rössler et al. [48]. It comprises 1000 videos (approximately 1.8 million images) with real sources and targets ground truth to enable supervised learning. These videos then use four state-of-the-art methods to create deepfakes. Two (Face2Face and FaceSwap) are graphic-based approaches, and two are learning-based approaches (DeepFakes and Neural Textures). Besides the manipulation output, creators also compute ground truth masks that indicate whether a pixel has been modified, which can be used to train forgery localization methods. We will briefly describe two fake-generating methods we will use in our experiments.

The Deepfake part of the dataset works with applications and available models for deepfake creation. A face from the series in the source video or image collection thus replaces the face in the target sequence. These methods are based on two autoencoders trained to reconstruct the source image into a target face. However, only the part of the image where the shape is located is used for this, and then in postprocessing, the shape is added to the rest of the image, which is not modified.

Neural Textures use a unique rendering approach. They use original video data to learn neural textures of the face in the target video. This is then trained using photorealistic reconstruction loss in combination with adversarial loss. This part of the generated deepfakes mainly focuses on tracking the shapes in the image and, thus, on more accurate shape reconstruction. They also modify especially the facial expressions around the mouth. The regions around the eyes remain unmodified.

The individual components of the dataset were also studied using human observers. In this study, 204 participants, after easy familiarization with the binary classification bag, tried to determine the relevance of randomly selected images, which were distributed in a 1:1 ratio. Subjects also had a limited time to decide, with a rule of 2, 4 or 6 seconds. These results, which can be seen in Figure 4.4, helped us to select the right part of the dataset for our experiments. We were able to use them to identify the strongest and weakest points and use them to select the content of the dataset more broadly, not to define it for a specific group based on the quality of the deepfake. Besides that DeepFakes and Neural Textures are created using deep learning, they also add diversity to our training dataset in

that DeepFakes were the easiest to detect by humans and Neural Textures were the most problematic.

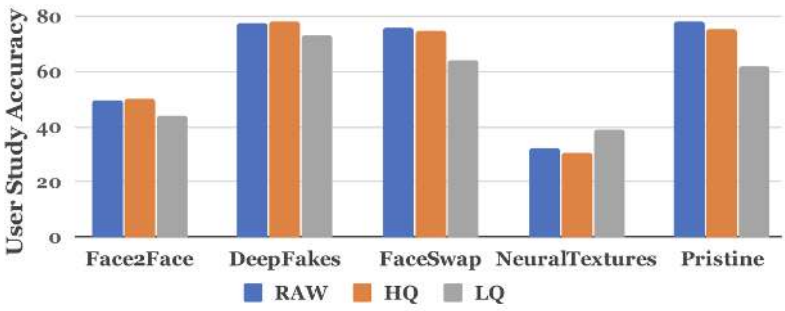

Figure 4.4: The results of a study on all quality levels of the dataset show a correlation between video quality and the ability to detect fakes. With a lower video quality, human performance decreases on average from 68.7 % to 58.7 %. We have quoted this graph from Rössler et al. [48].

The version of the dataset we used contains 363 videos from 28 different actors and 1000 videos downloaded from the youtube platform. It combines common videos from actors into mixes that alternate between source and target videos. This dataset has not been standardized into training, testing and validation. Therefore, by analyzing the individual combinations, we divided the dataset into the following combinations according to the indexing of actors:

- Training dataset: 1, 2, 3, 4, 6, 7, 9, 11, 12, 13, 14, 15, 18, 20, 21, 25, 26, 27
- Validation dataset: 5, 8, 16, 17, 28
- Test dataset: 10, 19, 22, 23, 24

Figure 4.5: Sample images from generated datasets from DeepFake detection, on the left is a bad example, on the right a better one.

In the same way, we split the deepfake images and the associated real images. The neural textures part of the dataset takes video pairs (with source and target video swap) from the youtube platform. We divided this part of the dataset in the ratio 70:15:15 into training, validation and test sets without regard to specific videos. Suppose a video from one side appears, for example, in the test dataset. In that case, the only video in the validation dataset will be its opponent, which should not show problems during the evaluation process.

Figure 4.6: Sample images from generated datasets from Neural Textures, on the left is a bad example, on the right a better one.

In Figure 4.5, we show examples of generated deepfakes against the Deepfake part of the dataset. Figure 4.6 again shows the same example from the Neural Textures dataset. Also, in the attached examples, it is possible to see the differences in the quality of the individual approaches. While with Neural Textures, we had a problem finding an image that could be considered inferior, and we found obvious signs of deepfakes, with the DeepFake detection dataset, the problem was rather the opposite, i.e. to find a sufficiently high-quality image where it is difficult to distinguish whether it is real or fake.

4.4.2 Celeb-DF

Celeb-DF is a dataset focusing on deepfakes detection published in 2019 by Li et al. [38]. The dataset contains 5639 deepfake videos (more than 2 million frames). Source videos are created from publicly available videos on YouTube containing 59 different celebrities of different genders, ethnicities and ages. Deepfake videos are then generated by synthetic methods, mostly publicly available neural network models for generating deepfakes. The auto-encoder is usually formed by two CNNs, the encoder and the decoder. The encoder E converts the input target's face to a vector known as the code. To ensure the encoder captures identity-independent attributes such as facial expressions, there is one single encoder regardless of the subjects' identities.

On the other hand, each identity has a dedicated decoder D_i, which generates a face of the corresponding subject from the code. The encoder-decoder pair is formed alternatively using E and D_i for the input face of each subject and optimising their parameters to minimize the reconstruction errors (l_1 difference between the input and reconstructed faces).

The parameter update is performed with the back-propagation until convergence. The synthesized faces are then warped back to the configuration of the original target's faces and trimmed with a mask from the facial landmarks. The last step involves smoothing the boundaries between the synthesized regions and the original video frames.

The Celeb-DF dataset is comprised of 590 real videos and 5,639 deepfake videos. The average length of all videos is approximately 13 seconds, with a standard frame rate of 30 frame-per-second. 56.8 % of subjects in the real videos are male, and 43.2 % are female. 8.5 % are of age 60 and above, 30.5 % are between 50 - 60, 26.6 % are in 40s, 28.0 % are 30s, and 6.4 % are younger than 30. 5.1 % are Asians, 6.8 % are African Americans and 88.1 % are Caucasians. The real videos exhibit many changes, such as the subject's face size (in pixels), lighting condition, orientation and background.

The dataset creators also focused on improvements such as colour mismatch and temporal flickering using data augmentation. To correct for mismatched skin colour, a colour spectrum matching algorithm was applied at each epoch, and this approach was then applied to the synthesis of the images as well. The temporal flickering of faces was reduced in the DeepFake videos by incorporating temporal correlations among the detected face landmarks. Specifically, the temporal sequence of the face landmarks is filtered using a Kalman smoothing algorithm to reduce imprecise variations of landmarks in each frame.

Figure 4.7: Sample images from the Celeb-DF dataset that looks real.

Overall, Celeb-DF offers videos with good-quality images, where distinguishing between a deepfake and a real image is often problematic. In Figure 4.7 and Figure 4.8, we can see examples of snapshots from this dataset where it is visible that the fakes are in high-quality processing. The drawback is the low quality of the overall video in the dataset. Therefore, in our experiments, we decided to use this dataset as a reference dataset for evaluating the models since they are generated in entirely different ways or videos in the faceforensics dataset and better simulate real-world usage. Thus we can better observe how the network is trained and whether the network does not appear to be overtraining on the data specific to the dataset.

Figure 4.8: Sample images from the Celeb-DF dataset that look unreal.

4.5 Detection pipeline

The vast majority of deepfake datasets focus on their detection in video. Since this work focuses on in-video detection, we have created an entire detection pipeline that starts with video processing and ends with binary output over individual frames in the video. An illustration of the whole detection pipeline can be seen in Figure 4.9.

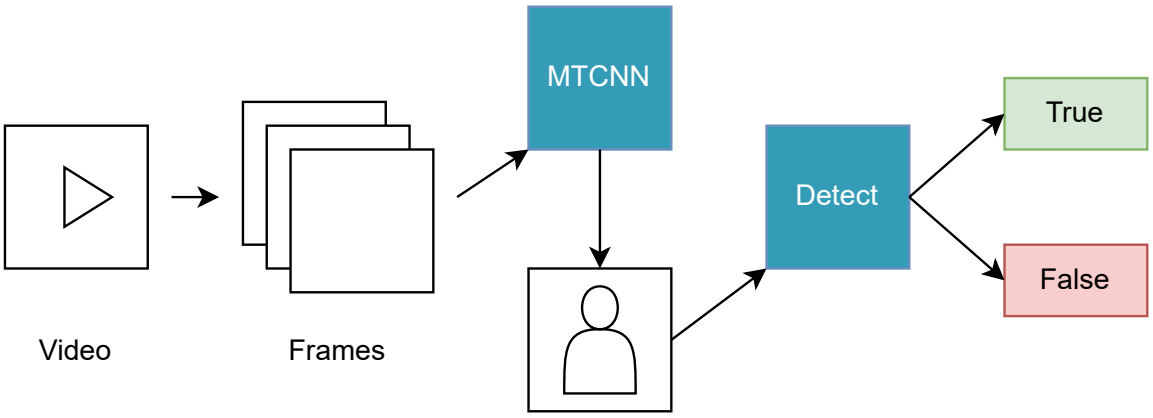

Figure 4.9: The illustration of the detection pipeline starts by splitting the video into individual frames and proceeds by detecting faces in the video using a pre-trained neural network model, which is then pruned into the desired input for the classification model. The latter then perform binary classification.

For the needs of our task, we divided the videos in datasets into frames of 1 FPS. We note that a higher framerate will be needed to use the models for detection over videos. Since the frames in the videos are always more complex scenes, it is necessary to cut out the part of the face in which we want to deal with deepfake detection. We perform face detection over the data using a retrained MTCNN model [60] for these needs.

After performing the face detection, we get the bounding box face from the MTCNN model. Using this bounding box, we can centre the face and perform its excision with a resolution adequate to the model we use for detection. In our case, we chose each model with an input resolution of 480×480 pixels. In case the resolution of the face exceeds this resolution, it is necessary to perform scaling to a lower resolution. To unify the scaling, we permanently reduce the resolution of the overall image itself size until the face's size resolves lower than 480 pixels per side (i.e. scaling x2, x4, x8,...). Note that this solution may not be entirely satisfactory because it can lead to a reduction in the number of artefacts often left by neural networks designed to generate them. In these cases, of course, it is also possible that the shape is located on one of the edges synthesised or some part of it is misplaced, but MTCNN detected the shape. In this case, we centralise the shape to the centre and fill the positions not in the image with a constant (in our case, black) colour.

We use this way of processing video (images) to train and validate models. The pipeline shows the complete video processing and detection in the video's frames and images in a real-world scenario. We would not need to perform this process in the case of, for example, shape biometrics, where input images are standardized using The ISO/IEC 19794-5 Token Face Standard regulates geometry, photometry, and behaviour [54].

Chapter 5

Experiments

In this chapter, we focused on designing our experiments as part of the iterative process of creating the architectures we mentioned in Chapter 4. We look at various models and architectures and our decisions to improve the architectures further iteratively. Later we will also describe the detailed results of individual experiments on the datasets.

5.1 Experiment design

When experimenting with the models, we chose an iterative approach. Each experiment results in an output, based on which we have selected an evaluation condition for the next experiment. We have narrowed down the experiment's focus and tried to analyse it in more detail.

5.1.1 Experiment 1: Selecting a fitting convolutional architecture

We focused on selecting a suitable convolutional network architecture for the first experiment. Candidate architectures included convolutional architectures from Section 3.5.2. For training and testing, we used the following versions of the architectures and their models from the Keras implementation: the VGG19, ResNet50, ResNet50v2, Densenet121, Xception, and EfficientNetv2B0. Each of these models has an input resolution of 224×224 pixels. Since the datasets we used later to train our proposed models are more suitable for higher resolutions, we decided to use the 140k Real and Fake Faces dataset[1] available on the Kaggle platform. This dataset consists of all 70k REAL faces from the Flickr dataset collected by Nvidia and 70k fake faces sampled from the 1 Million FAKE faces (generated by StyleGAN). Therefore, the evaluation of the results depends on the review directly of this dataset. We acknowledge the possible problematic data evaluation that may occur by evaluating purely over a test set from a single dataset. Still, we do not thoroughly analyse the results in the first phases of experimentation. Rather, we focus on the overall success rate achieved and the convergence of the models.

We tested the networks with and without Imagenet pre-trained weights for this experiment. We adjusted the network parameters minimally. The only parameter we modified was the learning rate in the optimizer.

We then evaluated the results to decide which architecture seemed most suitable for solving our problem. The assumption was that the best architecture would be EfficientNet,

[1] https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces

because, as we mentioned in Section 3.5.2, each architecture brings mostly the optimization of the previous one, so for such a complex task, we assume this result.

5.1.2 Experiment 2: Finetuning best convolutional model

Based on the evaluation of the previous experiment, we chose the EfficientNet architecture. For the second experiment, we evaluate the pre-trained architecture of EfficientNet. Since our prepared datasets support an input resolution of 480×480 pixels, we selected EfficientNet models that meet these criteria. The v2 M and v2 L architectures meet suitable input. These architectures differ mainly in the robustness of the network. While version 2 M contains approximately 50 million trainable parameters, version L contains twice as many, about 100 million. For these architectures, we experimented with freezing individual blocks (or enabling the coaching of weights in the whole network) from block three to block six. We always froze all the network weights up to the end of the observed block, so for example, in the case of block 5, blocks 1, 2, 3, 4 and 5 were frozen. Because of this architecture, we consider each block as one big unit representing a subnetwork. Thus, when experimenting with preserving the pre-trained weights, we have always either preserved the weights of the whole block or we have re-trained the whole block.

Since, in this case, there is already a risk of overfitting on the dataset, we decided that the guiding factor for evaluating the individual models is the calculation of the ROC (Receiver Operating Characteristic) and AUC (Area Under the ROC Curve)[2] over the FaceForensics test set and also on Celeb-DF dataset. In contrast, the dataset used for training is FaceForensics (as on all our models).

We have trained the models with 10 epochs each time, evaluating the model separately after each epoch to see the model's results. We also evaluated the average between these results after each training epoch. We can thus observe convergence over the validation dataset and, in this case, over the „real data“ that the detector can obtain. Then we can compare these data with the results of the validation dataset, gained loss and accuracy.

The results of this experiment can then be considered as the results of our baseline architecture. These results can then be used to evaluate our improved architectures and their overall comparison with the baseline architecture.

5.1.3 Experiment 3: EfficientDet modification

In this experiment, we explore the architecture of our proposed modified EfficientDet in more detail, including its mentioned more extensive architecture. Next, we compare the achieved results with the results from the previous experiment to see if there is an improvement compared to the baseline architecture. We followed a similar procedure as in the previous experiment. We tried to freeze individual blocks of the backbone architecture of EfficientNet from block 2 to block 6, and we also compared this with the approach without freezing the model weights.

For this architecture, we tried to experiment more with dropout layers. We evaluated each block in the training case with 20, 30, 40 and 50 % dropouts. We then tried the same procedure as in the previous experiment, i.e., evaluation over the Celeb-DF dataset, to evaluate the blocks that achieve the best results. In doing so, we also focused on the overall convergence of the results and the average achieved speedup over the epochs. Then,

[2] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

we also experimented with 0 and 10 % dropouts for the blocks that performed the best results.

We always compare the observed results with a reference architecture. In the case of EfficientDetM, it is the EfficientNet v2 M architecture, and in the case of EfficientDetL, it is the EfficientNet v2 L architecture. We then use the average achieved AUC value to evaluate the improvement (or deterioration) when freezing the weights of the individual layers of the backbone architecture. In this case, we also observe the best achieved AUC values among all the trained models.

5.1.4 Experiment 4: EfficientYdet

Since the faceforensics dataset also contains masks belonging to deepfake images, we decided to test the EfficientYdet architecture, which uses the u-net for its improvement. For this architecture, we focused purely on improving the previous architecture, not on improving the baseline. So, as an experiment, we prepared a dataset with the masks. Since there is always a video that copies its movement from 1 to 1 to a video with a fake shape, we could assign the images to each other and set up a network with two outputs.

As in the paper by Tjon et al. [58], we used binary cross entropy for the prediction part of the network and dice loss for the reconstruction part. Keras allows the use of coupled loss functions, so we assigned a coefficient of 0.5 to both functions to avoid a significant dominance of one part of the network. The last loss should therefore be their average.

As for the freezing of the block weights, because of the iterative approach, we decided to test only the versions of our modified EfficientDet network that showed the best results, so we did not perform this experiment on the freezing of blocks from two to six, but only a subset selected by us based on the results of the previous investigation. We follow the same procedure as the previous experiments in the evaluation case. We calculated the AUC of ROC over the Celeb-DF dataset.

5.1.5 Experiment 5: Compressed images

Since we have always worked with data of maximum quality in all our training processes and experiments, we decided to focus on the impact of compression on the network results over our best model in our last experiment. For this experiment, we performed JPEG compression over the FaceForensics and Celeb-DF datasets at four levels: 100, 80, 60, and 40. Over each of these compression formats, we then performed the same evaluation as in the other experiments, i.e., over the test datasets. We then plotted this result using ROC curves and observed changes in the evaluation processes.

5.2 Experimental results

In this section, we discuss the results of individual experiments and their implications for further iterative development of our proposed architectures.

5.2.1 Experiment 1 results

In this experiment, we mainly observed the achieved accuracy over the validation data during training. The training and validation process over 10 epochs can be seen in Figure 5.1. Due to experimenting with the learning ratio in the network optimizer, we have reached the value of 0.0001. At higher learning rate values, we reached a state where the network could

not train and did not achieve any results. The default value, which is ten times larger, did not produce results in the actual training of the network. The same result was reached when comparing the model's training with and without pre-trained weights. Demonstrably better results were achieved by models with pre-loaded weights from Imagenet compared to those that did not include these scales. This is probably because Imagenet is a large dataset which can be used to achieve good feature extraction over the individual CNNs tested. All networks achieved results after several epochs at the level of 95 % or more with their best result, so it is not relevant to compare individual architectures on this dataset and select the best candidate for our solution. However, it is true that in the confusion matrix, we have observed minor improvements between the different generations of the network but not significant ones. They could be improved by adjusting the hyperparameters or choosing a more suitable dataset for testing. We tried to evaluate the models without their specific improvement for this experiment.

Since we have obtained such results, we can consider this experiment irrelevant. In the process, we needed to review the available models of convolutional networks. In Figure 5.1, we can see various inconsistencies in training. The Resnet50 and Densenet121 architectures achieved a massive drop in validation accuracy during training. Also, none of the networks except the VGG19 model achieved smooth convergence. For model VGG19, however, unlike the others, we can observe a slow convergence to the results. The other models started to reach their peaks much earlier.

These fluctuations in validation accuracy can be caused by, for example, overfitting the network to the training data, which occurred after one of the epochs, and then was changed in the subsequent epochs. The networks have not been subjected to more detailed testing and experimentation with the network hyperparameters. Since we could not choose the best architecture for this experiment, we chose EfficientNet for further experiments. As we showed in Section 3.5, all architectures smoothly build on each other and should solve the problems that appeared in the previous one. EfficientNet is the newest one in this case.

5.2.2 Experiment 2 results

In this experiment, we focused on freezing individual blocks of the EfficientNet architecture. For completeness, in the model we used in Keras version 2.9.0, we always froze the weights on all blocks from the initial block to the blocks: *block3e_project_conv*, *block4g_project_conv*, *block5n_project_conv*, *block6r_project_conv*. The results during training and validation over ten training epochs are shown in Figure 5.2.

During training, we could observe relatively high validation accuracy after a few epochs for most models except that with frozen weights up to block six. In this case, there is very little room left for the trainable parameters of the model since block seven has fewer trainable parameters than the previous blocks. We then performed DET calculations over the dataset using the best-trained models from ten epochs. The resulting DET[3] curves can be seen in Figure 5.3.

From this observation, we could weakly conclude that the best thing to do is to retrain all the network weights and terminate the experiment. Even though the data we use in the test set has never been seen by the network either in training or validation, the data comes from the same dataset and is produced by the same deepfakes techniques. We can therefore assume that these measurements are not indicative. We, therefore, evaluated the ROC and AUC over the test set from the Celeb-DF dataset. We tested every single model

[3] https://scikit-learn.org/stable/auto_examples/model_selection/plot_det.html

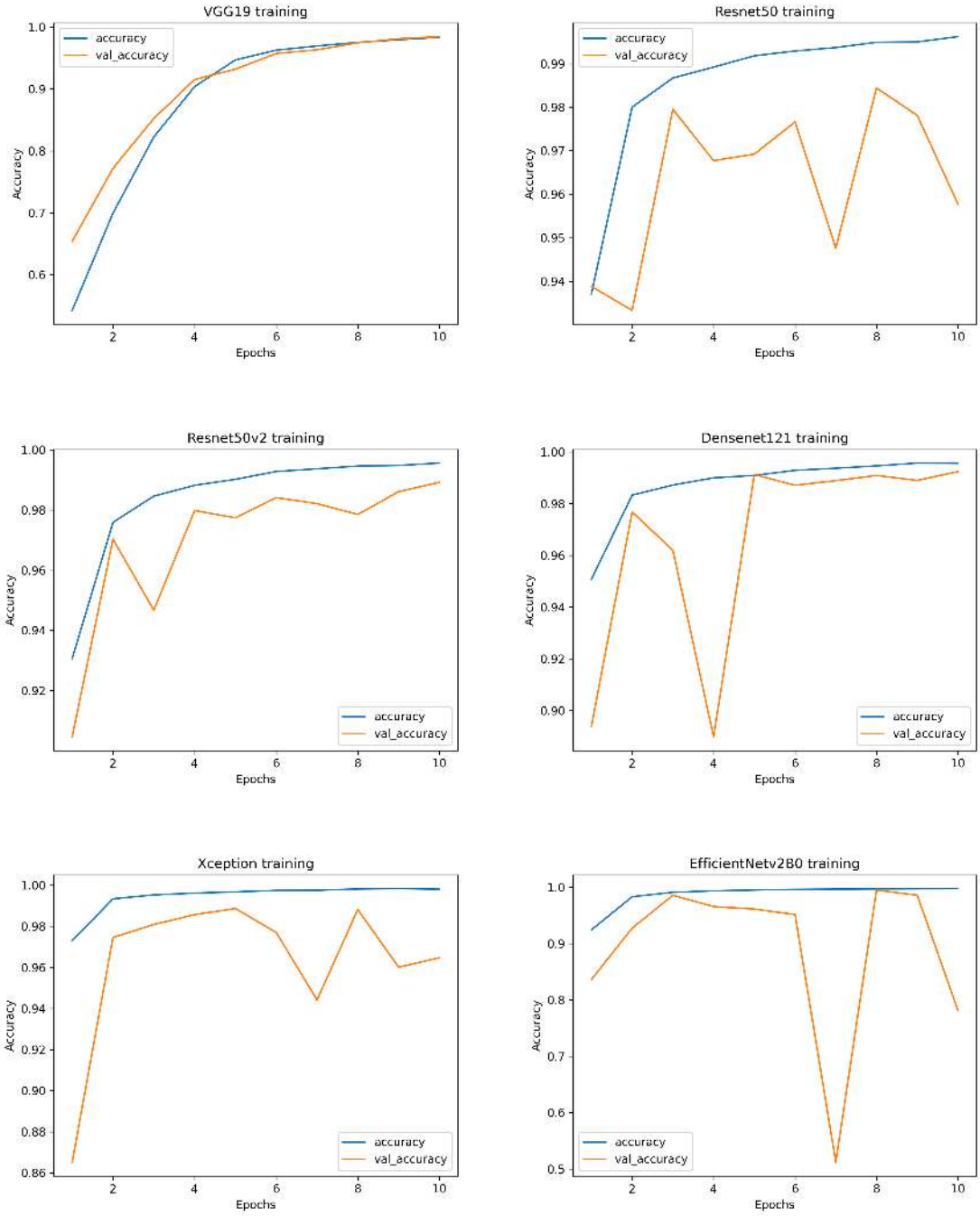

Figure 5.1: Training and validation process of individual convolutional models over a training dataset of 140k Real and Fake Faces.

after every training epoch and tried to observe the behaviour of the network to completely unknown data.

Thus, Table 5.1 shows the AUC results after each training epoch. We have observed the individual AUC values achieved and then the average ones achieved over the individual blocks. As we can see, in this case, the model in which we froze the weights in the first four blocks of the network performed best unambiguously. Epoch eight achieved the best

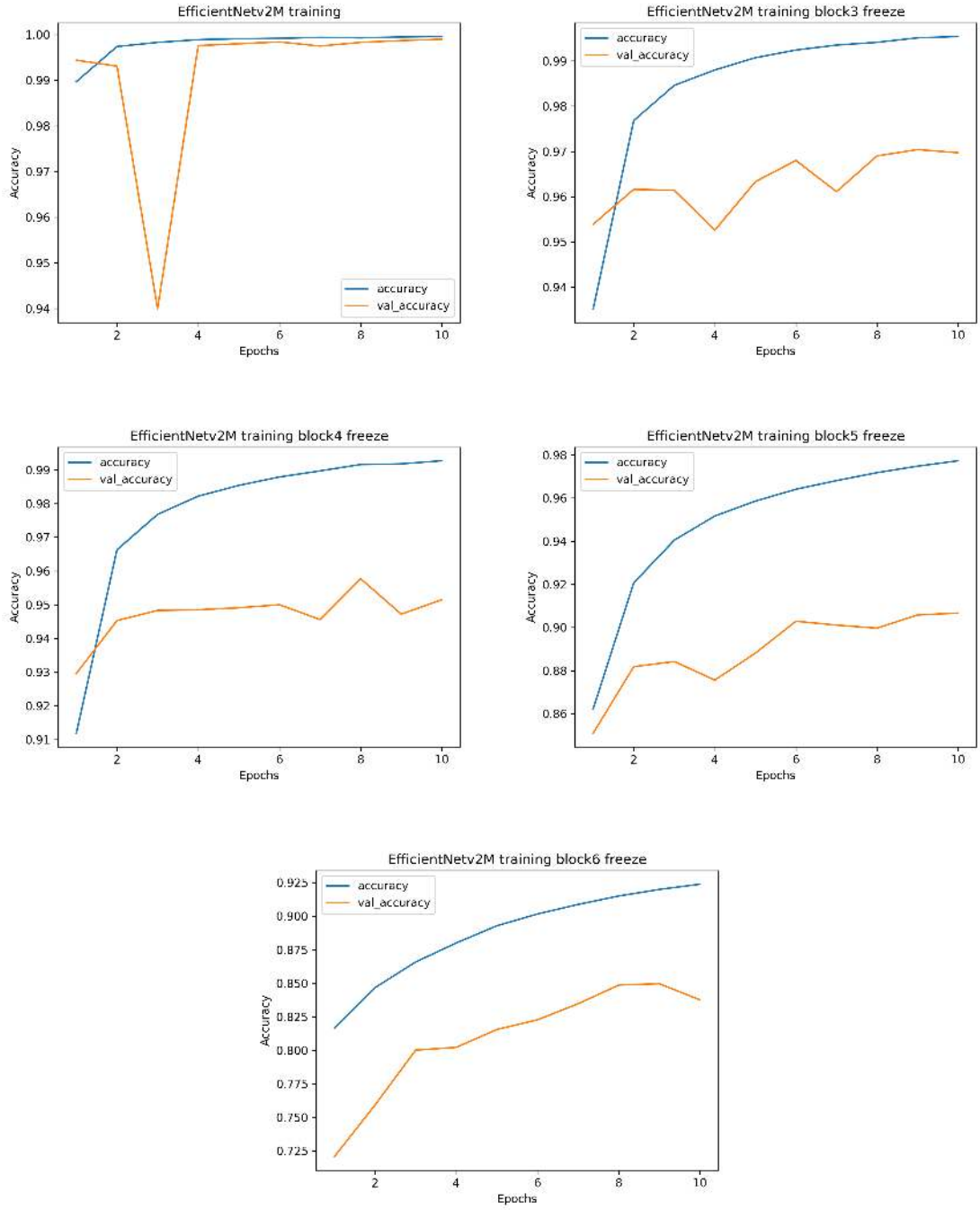

Figure 5.2: Training and validation process of the EfficientNet v2 M model depending on the preloaded Imagenet weights.

accuracy on the validation dataset, 95.77 %, even in the training process. However, this can also be considered a random match since we have no evidence that this evaluation would not be different on another dataset. On the other hand, we would like to point out the decreasing tendency of the AUC value on this dataset in case of freezing a smaller number of blocks. As we can see, the model where we have trained all the model weights performed

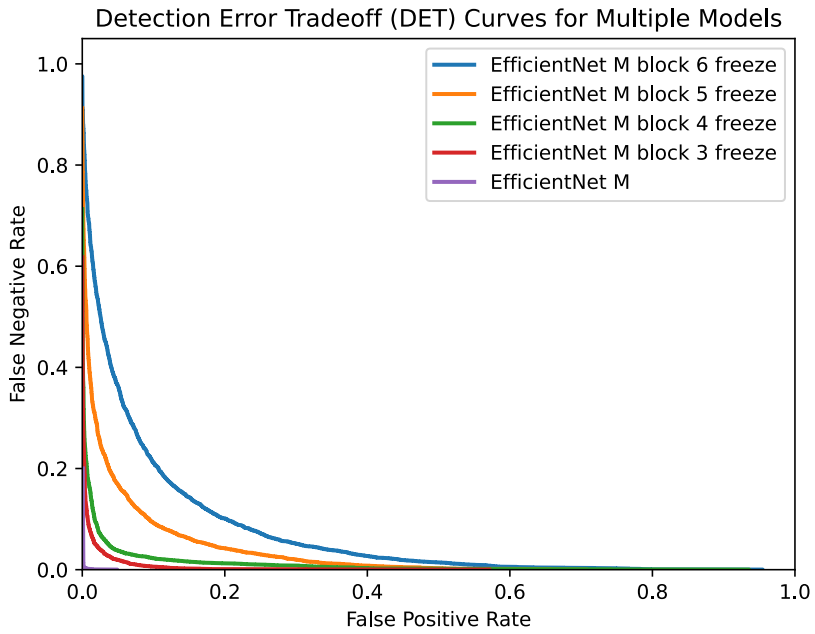

Figure 5.3: DET curves calculated over the best models based on how we froze individual network blocks in the training process.

potentially worse in this case than we would have in the case of random selection, where we have a 50 % chance of a correct evaluation.

Table 5.1: AUC calculated over the EfficientNet v2 M model based on how we froze individual network blocks in the training process using the Celeb-DF dataset. The range of values is from 0 to 1, where 1 symbolizes the best value. The best value achieved is 0.81355.

Epoch	Block 3	Block 4	Block 5	Block 6	Without
1	0.71465	0.80102	0.63415	0.53540	0.36158
2	0.69465	0.73966	0.66337	0.57004	0.46389
3	0.73439	0.78646	0.73628	0.60206	**0.52113**
4	0.76015	0.73170	**0.76677**	0.63670	0.35873
5	0.72410	0.75450	0.73945	0.61132	0.24104
6	0.75004	0.78548	0.73895	0.64612	0.41540
7	0.69710	0.81035	0.72408	**0.64846**	0.50440
8	**0.77165**	**0.81355**	0.76068	0.56969	0.38606
9	0.73647	0.70130	0.71734	0.62131	0.42837
10	0.75414	0.71430	0.73751	0.58812	0.36347
Average	0.73147	**0.76934**	0.72012	0.60457	0.40896

The same evaluation was attempted on the EfficientNet v2 L model since this model has a more robust structure but preserves the dimensionality of the input image and contains seven subdivided blocks. We applied the same training procedure with freezing the block weights, the only difference being that the names of the layers we ended up freezing are subtly different since most of the blocks contain multiple subblocks. The evaluation results

can be seen in Table 5.2. The model without free weights does not have the result from the tenth epoch. This is because the model is robust and could not be trained within 24 hours for ten epochs. By default, we performed the training process using the metacentre project, where we used PBS jobs with a limitation of 24 hours. We did not consider the result from the last epoch as necessary for our further investigation.

Table 5.2: AUC calculated over the EfficientNet v2 L model based on how we froze individual network blocks in the training process using the Celeb-DF dataset. The range of values is from 0 to 1, where 1 symbolizes the best value. The best value achieved is 0.81665, we do not consider this value relevant because it was achieved after the first training epoch, so it is likely random.

Epoch	Block 3	Block 4	Block 5	Block 6	Without
1	0.61621	**0.81665**	0.73317	**0.67912**	0.28770
2	0.78505	0.67752	0.74864	0.61619	0.21287
3	0.65378	0.77933	0.73430	0.52846	0.30313
4	**0.79712**	0.74017	0.71622	0.55273	**0.30564**
5	0.77372	0.62209	**0.78956**	0.59024	0.30025
6	0.76230	0.71462	0.78016	0.63984	0.29078
7	0.72379	0.66700	0.78203	0.60616	0.24848
8	0.69848	0.76492	0.77275	0.61046	0.28126
9	0.77292	0.71118	0.76923	0.65926	0.28671
10	0.72999	0.63755	0.77031	0.65283	-
Average	0.73134	0.71310	**0.75964**	0.61353	0.27965

In this case, the network behaves significantly differently than version M. We can again see the fact that the best average results are achieved by freezing the weights up to block four, but at the same time, we can see that the highest achieved AUC value occurred when freezing the weights up to block three. This phenomenon occurred right after the first training epoch. This suggests that it may be a coincidence. Still, it also alerts us to a possible problem with the evaluation metric since we were able to get the best result this early. Then it only sets a decrease, even to lower values than the values with another method of freezing the pre-trained weights.

In general, we consider the best AUC obtained from this experiment to be the AUC value 0.81355 from model M. This value can therefore be considered as a reference for the evaluation of further experiments.

5.2.3 Experiment 3 results

In this part, we did not focus on the results obtained over the validation and test set from the FaceForensics dataset because, in the previous experiments, we concluded that these results were not completely relevant for such a complex task. Observing a different dataset, Celeb-DF, in our case, is necessary.

During this experiment, we focused on a deeper investigation of the network training process. In addition to freezing the pre-trained weights, we also experimented with the dropout parameters of the network and tried to optimize its training process. Figure 5.4 shows the average AUC achieved over the Celeb-DF dataset when freezing individual blocks. This average includes all the different dropouts. From our observation, we could not observe much difference between the individual dropout values.

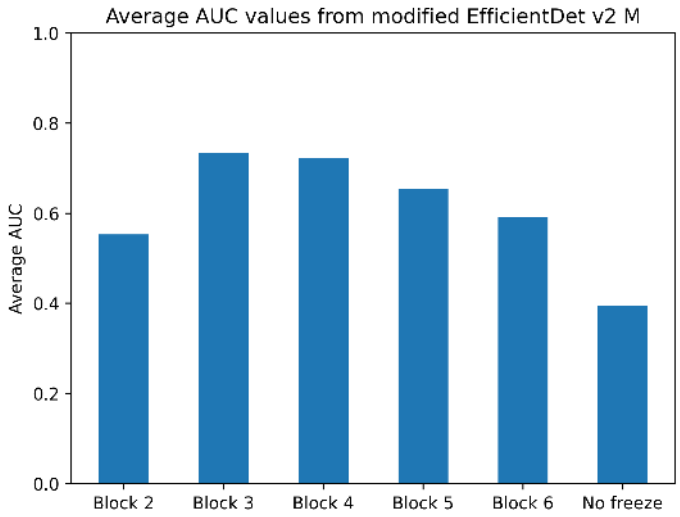

Figure 5.4: Average AUC values achieved with freezing block pre-loaded weights by training the modified EfficientDet network over the Celeb-DF dataset. The average is achieved across all dropouts we have tested.

As we can observe from the average values, we even achieved better results when we froze a smaller number of pre-trained weights, i.e. the first three blocks of the net. This factor may be due to our extension of the network with BiFPN blocks that connect to the network's third, fifth and seventh blocks. Therefore, the blocks connected to the fifth block have more room to adapt to the scales, as they also have the fourth block free for their optimisation and adjustment of the scales. Since we have obtained reasonable results only in these two cases, we only report results from these measurements in Table 5.3 and Table 5.4.

Table 5.3: AUC calculated over the modified EfficientDet v2 M model when we froze individual network blocks up to block three in the training process while using different dropout rates using the Celeb-DF dataset. The range of values is from 0 to 1, where 1 symbolizes the best value. The best value achieved is 0.82147. DR means dropout rate.

Epoch	DR 0.2	DR 0.3	DR 0.4	DR 0.5
1	0.72341	0.72650	0.76388	0.79301
2	0.72644	**0.76568**	0.75130	0.74738
3	**0.77431**	0.73015	0.72495	0.71822
4	0.74781	0.62247	0.74912	0.73482
5	0.68438	0.67978	0.78406	0.72973
6	0.74345	0.74611	0.73127	0.74202
7	0.76712	0.68332	0.77551	0.72068
8	0.64737	0.68225	0.79873	**0.79979**
9	0.68266	0.74775	0.76434	0.69602
10	0.72976	0.73140	**0.82147**	0.67932
Average	0.72267	0.71154	**0.76646**	0.73610

Table 5.4: AUC calculated over the modified EfficientDet v2 M model when we froze individual network blocks up to block four in the training process while using different dropout rates using the Celeb-DF dataset. The range of values is from 0 to 1, where 1 symbolizes the best value. The best value achieved is 0.78536. DR means dropout rate.

Epoch	DR 0.2	DR 0.3	DR 0.4	DR 0.5
1	**0.78536**	0.61947	0.67714	0.71091
2	0.73509	0.67479	0.69264	**0.77534**
3	0.76513	0.72876	0.73825	0.70866
4	0.73067	0.75899	0.76441	0.66238
5	0.66400	**0.76057**	0.63580	0.74545
6	0.76943	0.73608	0.67080	0.70765
7	0.77008	0.71707	0.73093	0.73235
8	0.72458	0.68891	0.74643	0.75162
9	0.71169	0.66381	**0.76159**	0.74276
10	0.75060	0.67810	0.75228	0.73900
Average	**0.74066**	0.70265	0.71703	0.72761

We will therefore focus more closely on block three, as it has objectively achieved better results in this case. As shown in Table 5.3, the best AUC value achieved is 0.4 in the dropout ratio after ten epochs. This value is also slightly higher than the baseline of the EfficientNet architecture. We want to point out the inconsistencies in the results concerning the datasets. While in the case of the baseline architecture, we achieved the best result in the same model, either in the test set over FaceForensics or Celeb-DF, this is no longer the case. The achieved validation accuracy over FaceForensics, in this case, was 96.59 %. This is realistically the third-best result in the training process. The seventh epoch, with a value of 97.01 % over FaceForensics, was the best in this case. Our evaluation metric with Celeb-DF reaches a score of 0.77551, which is not an insignificant reduction compared to the best solution. So, again, we have a question about the correctness of model evaluation for such a complex task, which we will discuss in more detail in the following discussion.

Regarding the extended architecture for the M version, it makes no sense for us to discuss its results in detail. In its evaluation, we achieved lower results than with this presented architecture, so we will not detail them here.

When testing the architecture, we encountered another case where we used EfficientNet v2 L as a backbone. In this case, we will focus on extended network architecture, not basic. In the case of our baseline architecture, although we managed to achieve better results in the case of the M architecture, here we could discuss the opposite.

In this case, only block three has been verified in the context of comprehensive testing. We can discuss a partially expected result because the L architecture is much more robust. Therefore, a large part of the pre-trained weights suitable for feature extraction can be located in earlier parts of the network, so we will not make a deeper comparison of the other blocks. The results we have achieved when filtering the weights into block three can be seen in Table 5.5.

Table 5.5: AUC calculated over the modified EfficientDet v2 L model when we froze individual network blocks up to block three in the training process while using different dropout rates using the Celeb-DF dataset. The range of values is from 0 to 1, where 1 symbolizes the best value. The best value achieved is 0.85232. DR means dropout rate.

Epoch	DR 0.2	DR 0.3	DR 0.4	DR 0.5
1	0.64641	0.63273	0.66113	0.64939
2	0.72119	0.66172	0.72919	0.77837
3	0.68159	**0.74349**	0.66755	0.69232
4	0.74191	0.66343	0.76956	**0.85232**
5	0.68760	0.66416	0.74947	0.68363
6	0.67804	0.62055	0.65813	0.74612
7	0.67701	0.63194	0.69028	0.74604
8	**0.75747**	0.63218	0.69367	0.81394
9	0.71839	0.71272	0.71340	0.67109
10	0.72790	0.72339	**0.77961**	0.68637
Average	0.70375	0.66863	0.71120	**0.73196**

We achieved an AUC value of 0.85232 at a dropout rate of 0.5, which could be considered a significant improvement compared to the baseline network architecture. In this case, this state has already occurred after training for four epochs. But again, we can point out the problem with the FaceForensics validation dataset compared to Celeb-DF. We achieved the best AUC result over Celeb-DF with a FaceForensics validation accuracy of 96.51 %, which is practically the third lowest validation accuracy we achieved here. For example, this factor could have been caused by high overfitting to the training dataset data since the same methods generate both the FaceForensics validation and the test set. Again, there is room for discussion here since training over four epochs is not much.

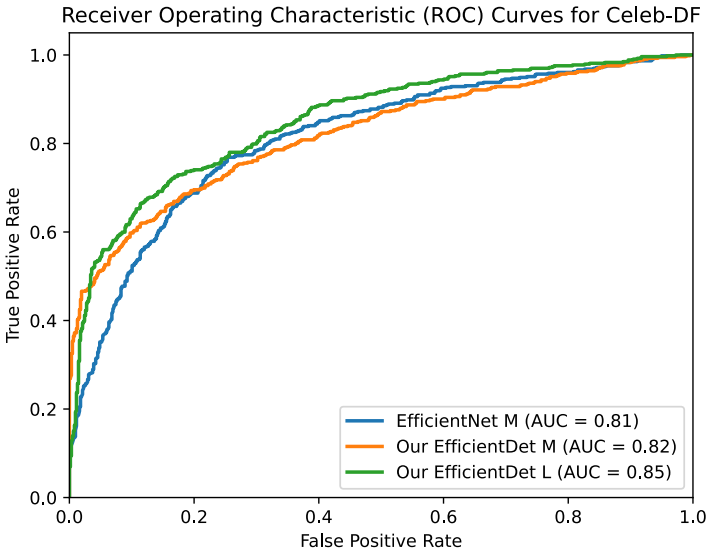

Figure 5.5: ROC and AUC calculated over individual models from the proposed architectures that performed best during the training process over the Celeb-DF dataset.

In Figure 5.5, we compare the best-trained models to the baseline architecture from the first experiment. Based on the chosen ROC and AUC metrics, we can conclude that we were able to improve the baseline architecture for the task of detecting deepfakes in the image with the addition of a minimum of additional parameters to the network since the new architecture has more parameters only an order of magnitude less than the baseline.

5.2.4 Experiment 4 results

We will deal with the fourth experiment only briefly since, from our point of view, it did not produce satisfactory results or bring any significant improvements to the network. As we addressed in Section 5.1.4, we have created a proposed architecture using U-net to improve the accuracy of prediction, which allows us to develop localization of the places that have been modified simultaneously. In the reference paper by Tjon et al. [58], the authors report high accuracy over the FaceForensics dataset. Although we did not copy their procedure in detail, we can confirm this statement because we did not use only EfficientNet as a backbone but our extensive EfficientDet architecture. The training process results can be seen in Figure 5.6.

In our evaluation process over the Celeb-DF dataset, we did not observe any significant improvements in the calculation of AUC and ROC. Within the iterative process, we tested the architecture where we froze the first three network blocks in both cases, i.e. the EfficientDet M and EfficientDet L architectures. In both cases, we experimented with dropout rates from 0 to 0.5, similar to the previous experiments.

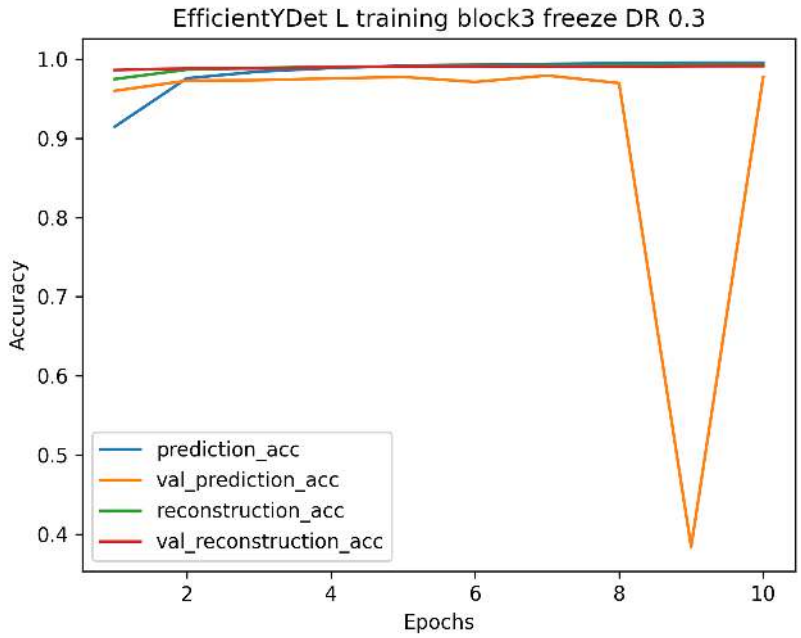

Figure 5.6: Training and validation process of the EfficientYDet L model with preloaded Imagenet weights.

This condition may have arisen for several reasons. We assume that, due to the masks in the network, overfitting over the training dataset data occurred even earlier than in the previous experiments because the propagated adjustments of the weights in the network

were already coming from the reconstruction part, even though we wanted to prevent this scenario through the use of bound loss functions. Another possible reason for this behaviour could be the poor processing of the masks in the network training process. Masks do not provide deeper information about how the input image was modified, but only in which places the modification should be located. Also, u-nets, as mentioned earlier, are used mainly for segmentation tasks in healthcare, which is not quite the same as deepfakes. The original approach is more about detecting anomalies, which can often be solid. In the case of deepfakes, we cannot classify how solid this output is.

5.2.5 Experiment 5 results

In this experiment, we evaluated the overtrained model EfficientDet L which achieved the best AUC score, i.e. 0.85 in previous experiments. We evaluated FaceForensics and Celeb-DF datasets, where we performed JPEG compression. The resulting DET curves can be observed in Figure 5.7 for FareForensics and Figure 5.8 for Celeb-DF.

We can observe a slight increase of DET curves in the evaluations on the individual graphs, but not a big one. The fact that we can watch this factor even with a test set of the training dataset is an expected condition. Even with an unknown dataset, the presence that we can observe may imply that compression can reduce the detector's efficiency. A better question is how much reduction can affect the results. The values we have measured suggest that this impact might not be significant, but a deeper investigation of this result is needed than just an investigation of the AUC score. While we can argue that compression impacts the detection of deepfakes, we cannot determine how large.

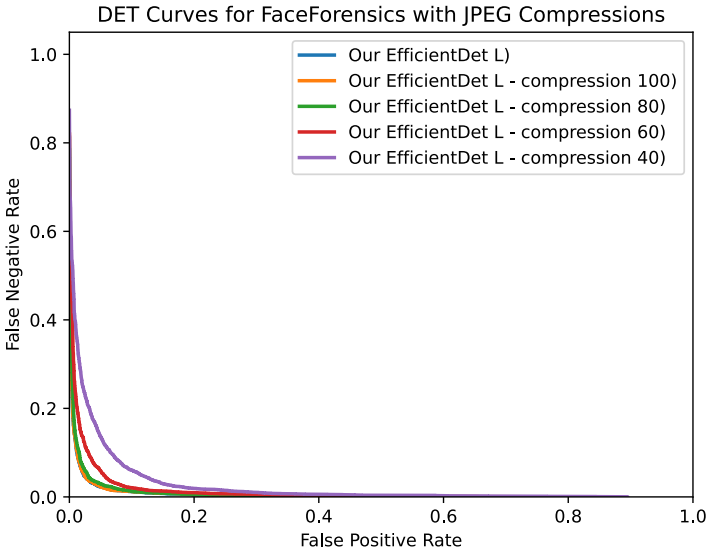

Figure 5.7: DET curves calculated over the EfficientDet L model on the FaceForensics dataset with different compressions. The compression parameter in the histogram shows the compression ratio of the jpeg. Where 100 represents the highest number of retained pixel values, i.e. the minimum compression, and 40 is the lowest, i.e. the highest compression.

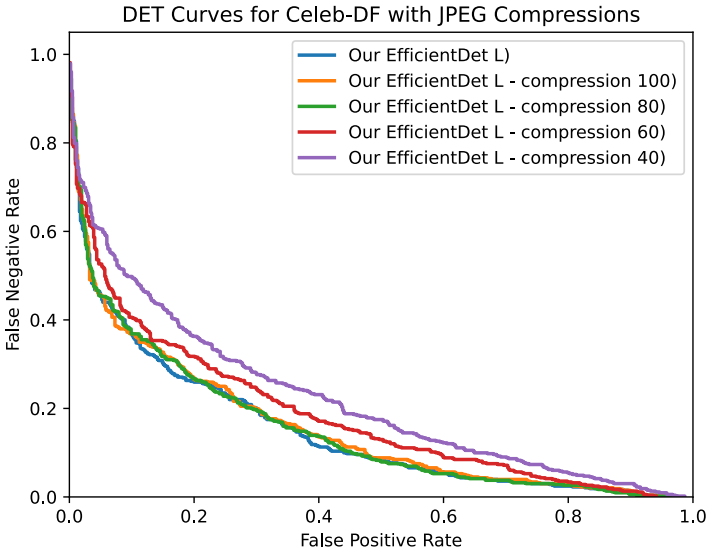

Figure 5.8: DET curves calculated over the EfficientDet L model on the Celeb-DF dataset with different compressions. The compression parameter in the histogram shows the compression ratio of the jpeg. Where 100 represents the highest number of retained pixel values, i.e. the minimum compression, and 40 is the lowest, i.e. the highest compression.

5.3 Discussion

The first finding we should discuss in more detail is the inconsistency between the results of the training validation metric and the general assessment using the Celeb-DF dataset. As we have shown several times, there is no direct connection between the evaluation of the training process and the results of the test set.

Various factors can cause this fact. One of the factors may be the just mentioned overtraining of the model on data from the training dataset. Because the data in different datasets are usually generated using several approaches, the network can learn to detect the models with which the deepfake is developed. We briefly discussed a similar process in Section 3.5.1, where we pointed out that in the case of human detection, we can observe various artefacts or defects in deepfake images. As a rule, this may be more of a problem of the model that generates these artefacts in specific cases or locations. This can lead to a situation where the neural network learns to recognize one or multiple kinds of architecture that create deepfakes. This condition results in the network achieving perfect results for a particular dataset type but not for other architectures.

In this thesis, we tried to solve this problem more complexly by choosing the best model over an unseen dataset, and we used Celeb-DF for this purpose. However, this is only one of the ways we can solve the problem, and we assume that this solution is not optimal either. If we used a different dataset for the evaluation, we might get different results than those we obtained in our case. Thus we would not be able to optimally determine which of the architectures is the best because the results would contradict each other. This area would need to be the subject of further investigation beyond the scope of this thesis.

Another critical problem may be the ROC metric itself, so let's look at its result from our model. In Figure 5.9, we can see the distribution of the early distribution over the

Celeb-DF dataset for individual images from the real image set and from the deepfake image set.

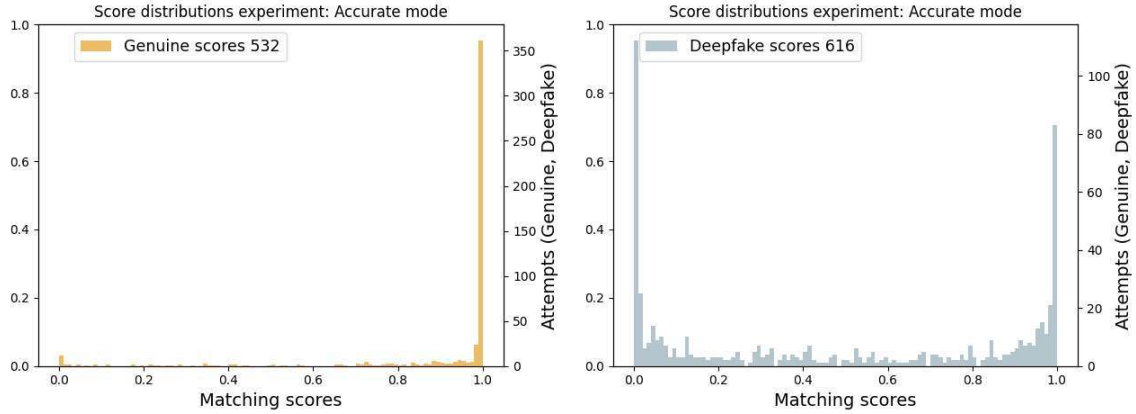

Figure 5.9: Distribution of scores of our best network model over the test set of the Celeb-DF dataset according to the expected results.

As we can observe, the model achieves high success and confidence in the case of real images. Of course, there are real images that the model classifies as deepfake or is unsure about, but this percentage is relatively small compared to the rest of the set. As far as deepfake images are concerned, we can see two main spikes. One is in the certainty that it is a deepfake image, and the other is precisely in the non-detection of the deepfake and its classification as a real face. The rest of the set is relatively evenly distributed. This state is not optimal either because the detector either detects with high confidence that it is a deepfake or with high confidence that it is a real image. Thus there is no normal distribution over the elements of the dataset.

This situation leads to hardly determining threshold value to distinguish a deepfake from a real image. While it is true that we could set this threshold relatively high, for example, somewhere around 90 %, and we would be able to detect many deepfakes, this state is not the optimal solution we are looking for. It would be much better for us if there were a Gaussian distribution of the individual sets with as little overlap as possible. Thus we could determine this threshold much more efficiently according to what we would expect from the detector. For example, in the case of a system reporting suspicious activity, we would be able to set the threshold to detect all deepfakes and even a subset of real images. This would achieve a state where we would get a cast of fake reports, but we would be almost guaranteed to get the deepfake images under control as well. Of course, this is only one scenario, which is impossible to generalize. It would depend on the application, but this is not exactly the state we have reached with our detector.

Thus, the problem also directly affects the evaluation using the AUC value. Since we achieve very high precision for real images also included in the AUC evaluation, this value may be somewhat distorted. However, the opposite could also occur in this case, where we would achieve a high success rate when evaluating deepfakes. The distribution that we set for deepfakes would be set for real images. Neither of these extremes is entirely satisfactory. Since we have mentioned the evaluation problem in the training process, it is not easy to prevent this condition.

We want to elaborate on the evaluation of the model itself and its comparison with other works. In the context of our exploration, we have tried to extend the EfficientNet architecture and improve its early performance for recognizing deepfakes in the image. According to our comparisons, we managed to improve the detection capability of the architecture by several AUC points, which can no longer be considered a negligible number. However, there is no standardized way to compare these architectures. Since we have defined a metric that we use to evaluate the model and have verified our results with it, we can declare the experimental part of our work a success based on this metric. And yet, for reasons we have already stated earlier in the discussion and pointed out during the evaluation, we cannot say with absolute certainty that the architecture we present here is the right one and is indeed generically better. For such an evaluation, comparing overall architectures and choosing the same evaluation procedure would be necessary. A good example is the Deepfake detection challenge by Meta [1], in which the resulting comparisons over unknown data were not published until the challenge was completed. However, this is again a generic question that we can discuss and would need further investigation, which is beyond the scope of this thesis.

The main thing we would like to point out is the actual deployment in real operation. As mentioned by Firc et al. [19], synthetic face datasets are limited. Many of these datasets could be considered obsolete at a time when there is active work on GAN models for generating deepfakes. For example, the FaceForensics dataset we used for training our model was published in 2019. During this time, many models have been developed that can generate deepfakes with higher quality and with a much lower number of detectable fragments. For this reason, we cannot estimate how our model would behave on these deepfakes and, therefore, how it could perform in case of deployment in real traffic. Testing was only performed on publicly available datasets.

At last, we would like to point out in the discussion the detection of deepfakes in the image itself. The datasets over which we have performed training and evaluation are videos. These series of videos always give us several frames which are superimposed on each other. Face-swapping videos might be easier to detect than single images, as they contain temporal information [19]. It is helpful to reconsider this fact in the future, and thus if we want to focus on deepfakes as a general problem, whether it is possible, and to what extent it is possible, to detect them in an image or whether it is better to focus mainly on single types of attacks, such as Face morphing which we discussed in Section 3.4.1.

Our observations, therefore, raise several research questions that will need to be addressed in the future:

- How to properly evaluate deepfakes detection models for their best generalisation?
- Is it appropriate to use already outdated datasets in this field, or would it be necessary to perform new ones to address this issue?
- How effective is it to solve deepfakes in the image as an in-general problem? Is it better to focus exclusively on individual types of attacks that are relevant?

Chapter 6

Conclusion

Deepfakes in any audio-visual form are becoming a part of our everyday lives over time, often without us even realising it. Despite their many possible uses for the proper purposes, such as education, cinematography and others, they also create many threats that we must face in the present and the future. We have pointed out several possible attack vectors in Section 3.3.

In the paper, we have discussed several architectures and functional models for creating deepfakes images, where distinguishing them from real content is often difficult. Indeed, this technology is still imperfect. Therefore, we can find and recognise flaws even among these advanced architectures. This task is often complicated and not so easy for the average person.

Since we have pointed out the importance of recognising what a deepfake is and what is a real contention, we have reviewed several architectures of convolutional neural networks that can be potentially suitable for solving this problem. Then we tried to select and improve the best architectures by an iterative procedure. For appropriate testing of our proposed architectures, we have proposed a way of evaluating models using two different datasets to be as close as possible to real-world scenarios, i.e. that we cannot evaluate conclusions purely from one set, which is often generated by the same procedure. We have therefore processed several datasets that we have used in our models' training and validation process, and in Section 4.4, we have also discussed their strengths and weaknesses.

Since most of these datasets are composed of videos, and often the input where the deepfake needs to be detected is not only the shape of a person, we have also designed a complete detection pipeline that serves for preprocessing the video or image before it can be evaluated using our presented model.

In the experimental part of the work, we have attempted to discuss the training and evaluation process of the univariate models in a broader spectrum. We also tried experimenting with different parameter values and freezing the pre-trained model weights for all models to keep feature extraction as optimal as possible. For our first proposed architecture, a modified EfficientDet model, we achieved a significant improvement in detecting deepfakes compared to the baseline EfficientNet architecture, the results of which we then discuss in more detail. The second architecture we investigated, where we placed the extension using the U-net network, was unsuccessful in this respect. For this one, again, we assume fast retraining due to a higher rate of dataset-specific inputs, which we used to evaluate the network training.

At the end of the experimental part of the work, we tried to analyse in detail the results of our improved architecture. We looked at its output values and tried to explain them from

the point of view of improving the early graphs and their use in real cases. This summary then gave us a broader view of the model we are presenting, and we were able to look at its strengths and weaknesses. We have also discussed the entire evaluation and training process in more detail and, at the same time, pointed out several problems associated with detecting deepfakes in the image in general. This gave us the space to open several questions for discussion on the solution to this problem in general, not only on our solution itself.

Bibliography

[1] *Deepfake Detection Challenge results: An open initiative to advance AI.* Available at: https://ai.facebook.com/blog/deepfake-detection-challenge-results-an-open-initiative-to-advance-ai/.

[2] Ajder, H., Patrini, G. and Cavalli, F. Automating image abuse - Deepfake bots on Telegram. Sensity AI. october 2020. Available at: https://www.medianama.com/wp-content/uploads/Sensity-AutomatingImageAbuse.pdf.

[3] Babel Corporation. *Iface: AI Face Swap app.* 2022. Available at: https://apps.apple.com/sk/app/iface-ai-face-swap-app/id1471067972.

[4] Bateman, J. *Deepfakes and Synthetic Media in the Financial System.* Carnegie Endowment for International Peace, 2020. Available at: https://www.jstor.org/stable/resrep25783.

[5] Brooks, T. and etc. *Increasing Threat of Deepfake Identities.* Homeland Security. Available at: https://www.dhs.gov/sites/default/files/publications/increasing_threats_of_deepfake_identities_0.pdf.

[6] Burchard, H. von der. Belgian socialist party circulates 'deep fake' Donald Trump video. Politico. Available at: https://www.politico.eu/article/spa-donald-trump-belgium-paris-climate-agreement-belgian-socialist-party-circulates-deep-fake-trump-video/.

[7] Celebi, N., Liu, Q. and Karatoprak, M. *A Survey of Deep Fake Detection for Trial Courts.* arXiv, 2022. DOI: 10.48550/ARXIV.2205.15792. Available at: https://arxiv.org/abs/2205.15792.

[8] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S. et al. *StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation.* 2017. Available at: https://arxiv.org/abs/1711.09020.

[9] Choi, Y., Uh, Y., Yoo, J. and Ha, J.-W. *StarGAN v2: Diverse Image Synthesis for Multiple Domains.* 2019. Available at: https://arxiv.org/abs/1912.01865.

[10] Chollet, F. *Xception: Deep Learning with Depthwise Separable Convolutions.* arXiv, 2016. DOI: 10.48550/ARXIV.1610.02357. Available at: https://arxiv.org/abs/1610.02357.

[11] Christlein, V., Riess, C., Jordan, J., Riess, C. and Angelopoulou, E. An Evaluation of Popular Copy-Move Forgery Detection Approaches. *IEEE Transactions on Information Forensics and Security.* 2012, vol. 7, no. 6, p. 1841–1854. DOI: 10.1109/TIFS.2012.2218597.

[12] Cote, J. Deepfake and fake news pose a growing threat to democracy, experts warn. Northeastern news. Available at: https://news.northeastern.edu/2022/04/01/deepfakes-fake-news-threat-democracy/.

[13] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. et al. *ImageNet: A large-scale hierarchical image database*. 2009. DOI: 10.1109/CVPR.2009.5206848.

[14] Farrell, J. Researchers say deepfake accounts are making the rounds on LinkedIn. *SiliconANGLE*. Available at: https://siliconangle.com/2022/03/28/researchers-say-deepfake-accounts-making-rounds-linkedin/.

[15] Ferrara, M. and Franco, A. Morph Creation and Vulnerability of Face Recognition Systems to Morphing. In: Rathgeb, C., Tolosana, R., Vera Rodriguez, R. and Busch, C., ed. *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Cham: Springer International Publishing, 2022, p. 117–137. DOI: 10.1007/978-3-030-87664-7_6. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_6.

[16] Ferrara, M., Franco, A. and Maltoni, D. *The magic passport*. 2014. DOI: 10.1109/BTAS.2014.6996240. Available at: https://www.researchgate.net/publication/283473746_The_magic_passport.

[17] Ferrara, M., Franco, A., Maltoni, D. and Busch, C. *Morphing Attack Potential*. 2022. Available at: https://arxiv.org/abs/2204.13374.

[18] Ferraro, M. F. and Gurney, B. J. The Other Side Says Your Evidence Is A Deepfake. Now What? Wilmerhale. Available at: https://www.wilmerhale.com/en/insights/publications/20221221-the-other-side-says-your-evidence-is-a-deepfake-now-what.

[19] Firc, A., Malinka, K. and Hanáček, P. Deepfakes as a threat to a speaker and facial recognition: An overview of tools and attack vectors. *Heliyon*. 2023, vol. 9, no. 4, p. e15090. DOI: https://doi.org/10.1016/j.heliyon.2023.e15090. ISSN 2405-8440. Available at: https://www.sciencedirect.com/science/article/pii/S2405844023022971.

[20] Foster, A. Names and locations of Aussie women being listed on sick revenge porn websites. *News.com.au*. Available at: https://www.news.com.au/technology/online/security/names-and-locations-of-aussie-women-being-listed-on-sick-revenge-porn-websites/news-story/b431e8c907c4ccf9dcba531c994da0e3.

[21] Foster, A. The terrifying Tinder scam catching out countless Australians. *News.com.au*. Available at: https://www.news.com.au/lifestyle/relationships/dating/the-terrifying-tinder-scam-that-countless-australians-are-being-caught-out-by/news-story/81db374b44be3f554a78063cb172d41c.

[22] Foster, C. Deepfake Evidence - What it is and how to spot it. Brodies. Available at: https://www.lexology.com/library/detail.aspx?g=be75a3a5-595b-4dc8-ac4e-7e6f0523257f.

[23] Goodfellow, I. J., Pouget Abadie, J., Mirza, M., Xu, B., Warde Farley, D. et al. *Generative Adversarial Networks*. 2014. Available at: https://arxiv.org/abs/1406.2661.

[24] Google. *Overview of GAN Structure*. 2022. Available at: https://developers.google.com/machine-learning/gan/gan_structure.

[25] Groshev, A., Maltseva, A., Chesakov, D., Kuznetsov, A. and Dimitrov, D. GHOST—A New Face Swap Approach for Image and Video Domains. *IEEE Access*. 2022, vol. 10, p. 83452–83462. DOI: 10.1109/ACCESS.2022.3196668.

[26] He, K., Zhang, X., Ren, S. and Sun, J. *Deep Residual Learning for Image Recognition*. arXiv, 2015. DOI: 10.48550/ARXIV.1512.03385. Available at: https://arxiv.org/abs/1512.03385.

[27] Hiep, P. and Joo, R. A Deep Learning Approach for Classification of Cloud Image Patches on Small Datasets. *Journal of Information and Communication Convergence Engineering*. The Korea Institute of Information and Communication Engineering. september 2018, vol. 16, no. 3, p. 173–178.

[28] Hofesmann, E. Have Deepfakes influenced the 2020 Election? Medium. Available at: https://medium.com/voxel51/have-deepfakes-influenced-the-2020-election-c0fc890aca0f.

[29] Huang, G., Liu, Z., Maaten, L. van der and Weinberger, K. Q. *Densely Connected Convolutional Networks*. arXiv, 2016. DOI: 10.48550/ARXIV.1608.06993. Available at: https://arxiv.org/abs/1608.06993.

[30] Jacoby, E. I Paid $30 to Create a Deepfake Porn of Myself. *Vice*. Available at: https://www.vice.com/en/article/vb55p8/i-paid-dollar30-to-create-a-deepfake-porn-of-myself.

[31] Jiang, L., Wu, W., Qian, C. and Loy, C. C. *DeepFakes Detection: the DeeperForensics Dataset and Challenge. In: Rathgeb, C., Tolosana, R., Vera-Rodriguez, R., Busch, C. (eds) Handbook of Digital Face Manipulation and Detection*. Springer International Publishing, 2022.

[32] Kang, J., Ji, S.-K., Lee, S., Jang, D. and Hou, J.-U. *Detection Enhancement for Various Deepfake Types Based on Residual Noise and Manipulation Traces*. 2022. 69031-69040 p. Available at: https://ieeexplore.ieee.org/document/9802100.

[33] Karras, T., Aila, T., Laine, S. and Lehtinen, J. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2018. Available at: https://paperswithcode.com/paper/progressive-growing-of-gans-for-improved.

[34] Karras, T., Laine, S. and Aila, T. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2018. Available at: https://arxiv.org/abs/1812.04948.

[35] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J. et al. *Analyzing and Improving the Image Quality of StyleGAN*. 2019. Available at: https://arxiv.org/abs/1912.04958.

[36] lab, S. studio. *Forward Propagation, Backward Propagation, and Computational Graphs*. 2022. Available at: https://d2l.ai/chapter_multilayer-perceptrons/backprop.html.

[37] Li, L., Bao, J., Yang, H., Chen, D. and Wen, F. *FaceShifter: Towards High Fidelity And Occlusion Aware Face Swapping*. 2020.

[38] Li, Y., Yang, X., Sun, P., Qi, H. and Lyu, S. *Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics*. 2020.

[39] Lim, E., Toh, K., Suganthan, P., Jiang, X. and Yau, W.-Y. Fingerprint image quality analysis. november 2004, p. 1241 – 1244 Vol.2. DOI: 10.1109/ICIP.2004.1419530.

[40] Masood, M., Nawaz, M., Malik, K., Javed, A. and Irtaza, A. *Deepfakes generation and detection: state-of-the-art, open challenges, countermeasures, and way forward*. February 2021. Available at: https://www.researchgate.net/publication/349703826_Deepfakes_generation_and_detection_state-of-the-art_open_challenges_countermeasures_and_way_forward.

[41] Mehlig, B. *Machine Learning with Neural Networks*. Cambridge University Press, oct 2021. Available at: https://arxiv.org/abs/1901.05639.

[42] Mirsky, Y. and Lee, W. The Creation and Detection of Deepfakes. *ACM Computing Surveys*. Association for Computing Machinery (ACM). january 2021, vol. 54, no. 1, p. 1–41. DOI: 10.1145/3425780. Available at: https://doi.org/10.1145/3425780.

[43] O'Shea, K. and Nash, R. An Introduction to Convolutional Neural Networks. *CoRR*. 2015, abs/1511.08458. Available at: http://arxiv.org/abs/1511.08458.

[44] O'Sullivan, D. A high school student created a fake 2020 candidate. Twitter verified it. *CNN*. Available at: https://edition.cnn.com/2020/02/28/tech/fake-twitter-candidate-2020/index.html.

[45] Phung and Rhee. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences*. october 2019, vol. 9, p. 4500. DOI: 10.3390/app9214500.

[46] Robertson, D., Mungall, A., Watson, D., Wade, K., Nightingale, S. et al. Detecting morphed passport photos: a training and individual differences approach. *Cognitive Research: Principles and Implications*. june 2018, vol. 3. DOI: 10.1186/s41235-018-0113-8.

[47] Ronneberger, O., Fischer, P. and Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv, 2015. DOI: 10.48550/ARXIV.1505.04597. Available at: https://arxiv.org/abs/1505.04597.

[48] Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J. et al. *FaceForensics++: Learning to Detect Manipulated Facial Images*. 2019.

[49] Savin, J. Deepfake porn is on the rise – and everyday women are the target. *Cosmopolitan*. Available at: https://www.cosmopolitan.com/uk/reports/a41534567/what-are-deepfakes/.

[50] SIfre, L. and Mallat, S. *Rigid-Motion Scattering for Texture Classification*. 2014.

[51] Simonyan, K. and Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv, 2014. DOI: 10.48550/ARXIV.1409.1556. Available at: https://arxiv.org/abs/1409.1556.

[52] Southern, M. G. *Twitter's New Verification System Has Blue & Gold Checkmarks* [https://www.searchenginejournal.com/twitters-new-verification-system-has-blue-gold-checkmarks/473934/]. 2022.

[53] Splice Video Editor. *Dawn - AI Avatars*. 2022. Available at: https://apps.apple.com/sk/app/dawn-ai-avatars/id1643890882.

[54] Standards, N. I. of and Technology. *Face Recognition Vendor Test (FRVT) Quality Concept*. 2019. Accessed: 2023-04-25. Available at: https://www.nist.gov/system/files/documents/2019/04/23/frvt_quality_concept_1.0.pdf.

[55] Tan, M. and Le, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR*. 2019, abs/1905.11946. Available at: http://arxiv.org/abs/1905.11946.

[56] Tan, M. and Le, Q. V. *EfficientNetV2: Smaller Models and Faster Training*. 2021.

[57] Tan, M., Pang, R. and Le, Q. V. *EfficientDet: Scalable and Efficient Object Detection*. 2020.

[58] Tjon, E., Moh, M. and Moh, T.-S. *Eff-YNet: A Dual Task Network for DeepFake Detection and Segmentation*. 2021. DOI: 10.1109/IMCOM51814.2021.9377373.

[59] Xu, A. Y. Generating Believable Tinder Profiles using AI: Adversarial & Recurrent Neural Networks in Multimodal Content Generation. *Medium.com*. Available at: https://medium.com/gradientcrescent/generating-swipeable-tinder-profiles-using-ai-adversarial-recurrent-neural-networks-in-dd68bd98c2f3.

[60] Zhang, K., Zhang, Z., Li, Z. and Qiao, Y. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*. Institute of Electrical and Electronics Engineers (IEEE). oct 2016, vol. 23, no. 10, p. 1499–1503. DOI: 10.1109/lsp.2016.2603342. Available at: https://doi.org/10.1109%2Flsp.2016.2603342.

[61] Zheng, Y., Yang, C. and Merkulov, A. *Breast cancer screening using convolutional neural network and follow-up digital mammography*. May 2018. DOI: 10.1117/12.2304564.

Appendix A

Contents of the included storage media

Root folder contains executable `main.py` used for experiments in Section 5 with `README.md`, `requirements.txt` and following directories:

- `custommodels`: Y-net models architectures
- `generators`: dataset processing files for executable *main.py*
- `models`: CNN models architectures
- `preprocessing`: scripts for preprocessing of datasets
- `processing`: checkpoints saving handler
- `weights`: best models weigths