BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

DEEPFAKE DETECTION FRAMEWORK

FRAMEWORK PRO DETEKCI DEEPFAKES

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. JAN BERNARD**
AUTOR PRÁCE

SUPERVISOR **Mgr. KAMIL MALINKA, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

140642

Institut:	Department of Intelligent Systems (UITS)
Student:	**Bernard Jan, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Cybersecurity
Title:	**Deepfake Detection Framework**
Category:	Security
Academic year:	2022/23

Assignment:

1. Learn about deepfakes (voice and video). Explore the current state of deepfakes detection methods (voice and video).
2. Learn about the technologies needed to create web extensions and technologies for creating scalable server applications.
3. Learn about existing deepfake detection solutions (e.g. other commercial web browser plug-ins)
4. Design an extensible framework (server-client or client-only) for deepfakes detection (support for at least 3 detection methods (voice and video)). Design a web extension for deepfakes detection that will use this framework. The solution should support multiple browsers and allow the detection of displayed content and uploaded files.
5. Implement the tool according to the design.
6. Test the functionality and reliability of the resulting implementation. Perform testing on at least two independent publicly available deepfakes datasets.
7. Discuss usability, detection efficiency and possible extensions.

Literature:
Puspita Majumdar, Akshay Agarwal, Mayank Vatsa, and Richa
Singh, “Facial retouching and alteration detection,” in Handbook of Digital Face Manipulation and Detection, pp. 367–
387. Springer, 2022
FIRC Anton a MALINKA Kamil. The dawn of a text-dependent society: deepfakes as a threat to speech verification systems. In: Brno: Association for Computing Machinery, 2022

Requirements for the semestral defence:
Items 1 to 4.

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

Supervisor:	**Malinka Kamil, Mgr., Ph.D.**
Consultant:	Ing. Anton Firc
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

Deepfake creation has improved a lot in recent times and hence is a dreaded menace to society. Deepfake detection methods have also responded with development, but there are still not enough good tools available to the general public. This work focuses on creating a deepfake detection framework that will be easily extended by other detection methods in the future, yet simple and accessible to the general public.

Abstrakt

Tvorba deepfake se za poslední dobu velmi zlepšila a tudíž je obávanou hroznou pro společnost. Detekční metody odhalující deepfake také reagovali rozvojem, ale stále není široké veřejnosti dostupné dostatečné množství dobrých nástrojů. Tato práce se zaměřuje na vytvoření frameworku na detekci deepfake, který bude jednoduše rozšiřitlený dalšími detekčními metodami v budounu a přitom jednoduchý a dostupný široké veřejnosti.

Keywords

deepfake, framework, deepfake detection, containerazation, web plugin

Klíčová slova

deepfake, framework, detekce deepfake, kontejnerizace, webový doplňek

Reference

BERNARD, Jan. *Deepfake Detection Framework*. Brno, 2022. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Mgr. Kamil Malinka, Ph.D.

Rozšířený abstrakt

...

Deepfake Detection Framework

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Mgr. Kamil Malinka Ph.D. The supplementary information was provided by Ing. Anton Firc. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Jan Bernard
May 9, 2023

Acknowledgements

I would like to sincerely thank my supervisor Mgr. Kamil Malinka Ph.D. for all the advice and insightful comments. The same thanks go to consultant Ing. Anton Firc.
I also owe a debt of gratitude to my family, especially my parents, friends for their wonderful support throughout my studies.

Contents

Chapter 1

Introduction

Deefake is the buzzword that has no agreed-upon technical definition. It consists of two words, deep and fake. Deep is referring to deep learning machine learning, which is used for creating fake voices, images, or even videos. It is a fast growing technical field of study and could be a major threat to society. We live in information era and creation of unrecognizable fake media affects us all. Deepfakes could be found on social media, news portals, etc.

This paper shows some security risks and impacts on many different fields of human life. There are already many popular videos containing deepfakes on social networks with millions of views, but also several successful attacks on biometrics systems. People are not able to consistently recognize deepfakes. That is why we need to create detection tools to help anyone with problem that is already there.

There are many different deepfake categories from face swap to face morphing and more. Most of the detection methods focus on a single domain, which means that they are capable of detecting only one deepfake category. There are few tools in the market utilizing those detection methods targeting undemanding/inexperienced users, and some of them are not free to use.

The first two chapters cover general knowledge about deepfakes such as potential risks and security impacts with mention of successful attacks, different types of voice, image, and video deepfakes. You can read about experimental detection methods and available tools in the market.

Second and more important part describes process of development deepfake detection framework and client application, which is the goal of this work. It starts by specifying requirements followed by detailed architecture. Architecture was designed with focus to easily integrates new detection methods into framework.

Last but not least, the implemented framework containing several open source detection methods was deployed into Azure AKS cluster and properly tested. The tests were mostly focused on the testing framework itself, mostly measuring used resources. Integrated detection methods were also tested, and all the results could be found at the end of this paper.

Chapter 2

Security impacts of deepfakes

The creation of fake media and their detection have been a problem since photography was invented. Digital photography or video with tools such as GIMP, Adobe Photoshop or Adobe After Effects allows more people to create fakes than before, still some experience in this area is needed. Media that have been modified or otherwise manipulated are called synthetic media, and they do not depend on whether it is an analogue or digital medium. Deepfakes also fall under this category [18]. Deep learning-powered tools allow even more unexperienced users to easily create trustworthy fakes.

The quality of deepfakes reached a level when a trained person or even an experienced researcher in this field has a problem of spotting them. This fast development allows creating realistically looking assets to art photography or movie production; unfortunately, it can be used for malicious purposes like creating fake porn videos to blackmail people or manipulate public via fake news. There are many use cases where deepfakes can be applied.

This is not a discussion about potential technology; deepfakes are already there and their usage will probably have an increasing trend. Over the last couple months, we could see huge development and popularity explosion around large language models such as ChatGPT [4]. Deepfakes do not have such publicity, but they have the same potential to change whole human society as previously mentioned language models. We are not far from using a pre-trained voice deepfake model powered by some large language model to automatically perform attack itself.

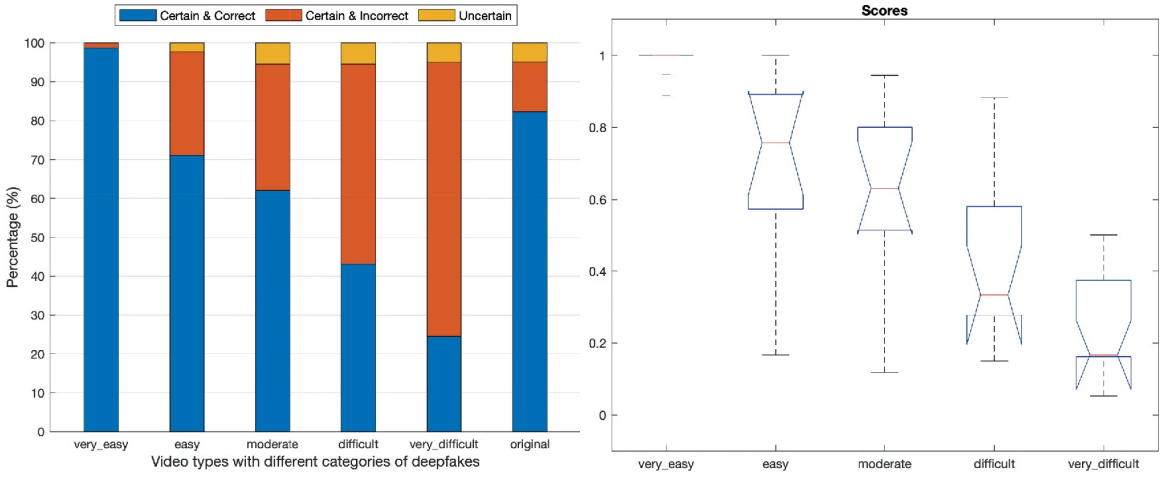
It is putting huge pressure on researchers to develop new forensics tools or any technology which will prevent malicious usage of deepfakes. As mentioned before, creating fakes is not new, and a whole field of study engaged in spotting fakes and developing techniques over 15 years. Despite continuous research efforts in the past, the advent of deep learning changed the rules of the game. [40]

2.1 Human capabilities of deepfake detection

The human ability to recognize fake materials from the originals is in contradiction to their quality. Korshunov and Marcel [21] confirmed this in their research. They created a questionnaire containing several videos, and the subject (interviewee) had to answer after watching the video whether the person in the video was genuine, fake, or uncertain. The videos were manually divided into five categories (very easy, easy, moderate, difficult, and very difficult, original).

The videos were split into several categories manually by the researchers, probably without using any metrics but based on their experience and feelings. Afterwards, ANOVA test shows there is an overlap in several categories, so several videos could be moved to a different category. However, the categories are still significantly different.

The results of test in fig. 2.1 certainly demonstrate that people's recognition ability decreases significantly as the quality of deepfakes increases. Also, the audience of this test knows they are looking for fakes, otherwise we can expect worse results if there will be unsuspected audience (e.g., deepfakes on social media). It is quite alarming that the correct answers in the category of "very easy" reach only 71,1 %. The quality of deepfake increases over time, thus it can be expected that human recognition ability will continue to decrease. [21]



(a) Subjective answers

(b) ANOVA test

Figure 2.1: Subjective answers and median values with error bars from ANOVA test for different deepfake categories. Retrieved from [21].

Another research tested only recognition of audio tracks and they were comparing humans versus computer programs. Attendees had a correct classification between fakes and origins 67 % after the first several rounds. Their accuracy increased while listening and answering to more tracks, but the value stabilizes at 80 %. On average, trained AI performs about 10 % better than human, but this result highly depends on the difference in learning and test dataset. Still, it shows that the computer can outperform humans in spotting deepfakes. [28]

2.2 Potentional risks and impacts

Humans are not good at recognizing deepfakes, but "why should we be worried?". Almost every technology humankind created could be used for good or bad – deepfake is no exception. There are plenty different deepfake categories, and each has its own attack vector or use case. This section is covering potential risks of those categories and their closer description will be covered in chapter 3.

Deepfakes are about gaining someone's trust or influencing him. For the last couple of years, there has been an increasing trend of scamming people, mostly via phone or computer [7]. Targeting only one person/victim, for example, to gain their money or

information. Those attacks are getting better and more credible and using deepfake to impersonate close friend of victim could be next step how to improve it, if it is not already happening.

Creating "fake news" to influence a large audience is the most common use case of deepfakes because we live in an information era. There are many targets of "fake news" such as rigging elections, demoralizing military units, or manipulating the stock market. In this case, politicians, celebrities, and significant personalities will be used in deepfakes to influence the audience. We can only imagine what one person or a high quality deepfake can change with enough media reach. For example, after one tweet from Elon Musk about Tesla's stock, sends shares down more than 10 % almost immediately [37]. [18]

A real example of deepfake is the famous video with Barak Obama insulting Donald Trump, which should spread awareness regarding the fast developing category of new thread[1]. Several years later another video stating Volodimir Zelenskyj talking about surrendering, it was proved that it is a manipulated video, and its purpose was to demoralize Ukraine army and make them capitulate[2].

There are other videos similar to the video of Barack Obama with the same purpose, spreading awareness about deepfake. For example, speech of Czech Republic president Miloš Zeman created by HBO for propagating their new series[3], but also to inform general public about this technology. We can consider those videos and their goal to be mostly success. Most people have heard about deepfakes and they are connecting them with those types of videos. However, many people share deepfakes inadvertently on their social media [2].

Not everyone considered them as a big thread. This is probably because there is no proof that there was a successful deepfake attack or fake news that influence them directly. This is the reason why deepfake could be so dangerous. The video of Volodimir Zelenskyj might have had a good chance to be successful, but it was quite fast debunked because of its terrible quality. People still should be inform about capabilities of deepfakes, but the target of the message should be different. Not saying only be careful we can make politicians say what we want, but show and provide some tools on how to spot the deepfakes.

Another field where deepfakes could be used is to trick biometrics systems to mark attacker as a different person. This technique is used to gain access to secured equipment, to building or even to application (internet banking), etc. Biometrics systems have been proven not ready to deal with deepfakes [11] [9]. The work describing one of attacks is called magic passport and demonstrates how biometric systems could be vulnerable. Most systems will require some changes such as adding a new module to the authentication pipeline, which will be detecting deepfakes [19]. Face or voice biometrics recognition systems are in greatest danger. Falsification of documents is related to this topic and there was a case of smuggling people across borders with an official passport containing morphed photos of two individuals [35].

These cases are only the tip of the iceberg, and in the future, everyone should ask if video on social media with film celebrities is real or even worse, if the evidence in courts is trustworthy or not. The solution to this is to use tools capable of detecting deepfakes. Those tools must be created with caution for unskilled users.

[1] https://www.buzzfeed.com/craigsilverman/obama-jordan-peele-deepfake-video-debunk-buzzfeed

[2] https://www.youtube.com/watch?v=X17yrEV5sl4

[3] https://www.youtube.com/watch?v=FzMnDwpKJrI

2.3 Detection tools requirements

Now we know that deepfakes could be security threats and there is a need for reliable detection tools. There is not many detection tool available for users. Most of them are basically command line scripts which are not for general purpose. More about individual methods could be fined in chapter 4.

The user interface should be simple, yet provide all relevant information. Generalizing anything to one number will be great, but not every time it is possible. Speaking of detection methods it should be able to deliver only one number. On the other hand, it will be nice to show the user where on the image is the manipulated area.

Nowadays people work with many different types of files. Only for audio we can have WAV, OGG, MP3 and many more. This means that the detection method supporting only one file type will not have huge success. There are two types of solution to this problem; first is to support as many file types as possible natively. The second approach will be to utilize existing tools to convert input files to one specific format.

In the end how do we compere different detection methods or even the same methods trained on different datasets. We would have to test each method individually and compare the results manually. There are some problems that need to be sorted out before we will have a reliable detection tool in the market.

Chapter 3

Types of deepfakes

There are plenty of methods on how to create deepfakes, and as its name suggests some of them are based on deep neural networks but not exclusively. This section describes most common types of neural networks used for creating voice, image, or video deepfakes. One of the most popular types for face deepfakes is Generative adversarial network (GAN), and it is used to create completely new faces or face manipulations.

Each method leaves traces in the medium that can then be detected. This is one way to recognize deepfakes so understanding process of creation is an advantage. Detection is described in more detail in chapter 4.

3.0.1 Neural networks

Neural networks are composited from neurons arranged in layers, and each layer is connected sequentially via synapses. Synapses are weighed, and the process of finding the proper value of all weights is called a learning neural network. To obtain results from the input of n-dimensional x process `forward-propagation` is used to propagate x through each layer. [27]

Input to layer is vector a of values calculated by previous layer or in case of first layer x itself. That means result of each layer is also vector calculated by activation function $f(a * W + b)$, where f is activation function (Sigmod, ReLU, etc.), a is input vector, W is matrix of weights between layers i and $i + 1$ and b is dimensional bias. Dimensional bias is a constant offset that helps the network shift the activation function toward the positive or negative side [1]. [27]

Now let's consider the neural network M as a black box and denote its execution as $M(x) = y$. Supervise learning to train M is using paired samples with from (x_i, y_i) and loss function L is defined. Loss function is to generate a signal at the output of M and propagate him back to find error of each weight in synapses. [27]

Optimalization algorithms such as gradient descent are then used to calculate new weights of M for the number of epochs. As a result of this process, the network learns the function $M(x_i) \approx y_i$ and is capable of making prediction on unseen data. More detailed descriptions of this could be found in the work of Y. Mirsky and W. Lee [27]. [27]

Next list shows types of neural networks used for generating deepfakes [27]:

- Generative Adversarial Networks (GAN) – Consist of two neural networks working against each other. One layer is generator and second is discriminator. Generator producing fake features trying to fool discriminator, on the other hand, discriminator is learning to distinguish between real sample and fake one.

- Encoder-Decoder networks (ED) – Contains at least two networks, encoder and decoder. It has narrowed layers towards its center. If encoder En and decoder De are symmetric and they are trained as $De(En(x)) = x$, then the network is called autoencoder. Generating deepfakes using ED trained with function $De(En(x)) = x_g$, where x_g is fake generated features. There is possibility to use multiple different ED chained after each other or using specific variant of ED called variational autoencoder.

- Convolutional Neural Network (CNN) – CNN is learning pattern hierarchies in the data. For deepfakes purposes, it learns filters applied over the input and forming an abstract feature map as the output.

- Recurrent Neural Networks (RNN) – RNN can handle variable length data and during processing it remembers state that can be used in next iteration. RNN are mostly used in audio.

Each architecture has its own subcategories that have small modifications or using some techniques from different architecture. All above mentioned neural networks types are shown in 3.1 and 3.2

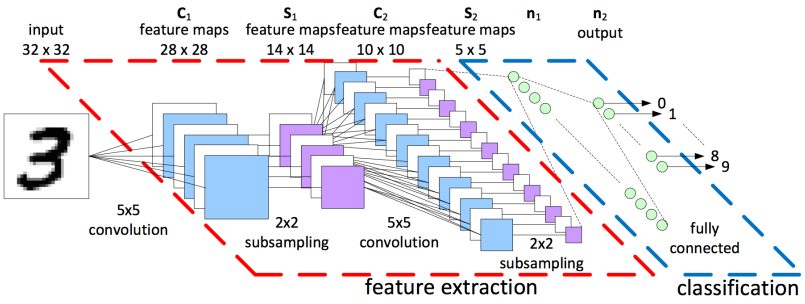


Figure 3.1: Architecture of convolutional neural network. Retrieved from [8].

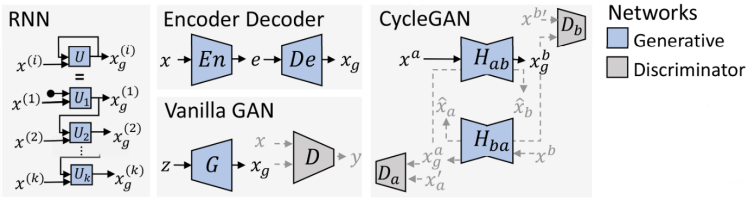


Figure 3.2: Basic neural network architectures (RNN, ED, GAN). Retrieved from [27].

3.0.2 Voice deepfakes

There are three different modalities for speech synthesis: text-to-speech (TTS), voice cloning (VC), and replay attack (RA) [42]. The last one is based on capturing victim voice and the replay of it. It is quite easy and cheap to perform because it only requires capture and replay device (today we can use for example smartphone) and current methods for voice recognition still have accuracy issues [26]. First two are using AI-synthesis with content regeneration which makes them more indistinguishable for naked ears. [42]

Text to speech (TTS) converts written text to artificial speech, on the other hand voice cloning consumes source voice. Both methods produce synthesis voice saying desired phrases

specified by the input, high-level diagram how those methods works could be seen on fig. 3.3. Voice deepfakes are used independently or with deepfake video (e.g., full puppet). Creating synthesis voice is computationally challenging and one of the goals is making real-time voice cloning. There are several projects that are trying to accomplish this[1] [2].

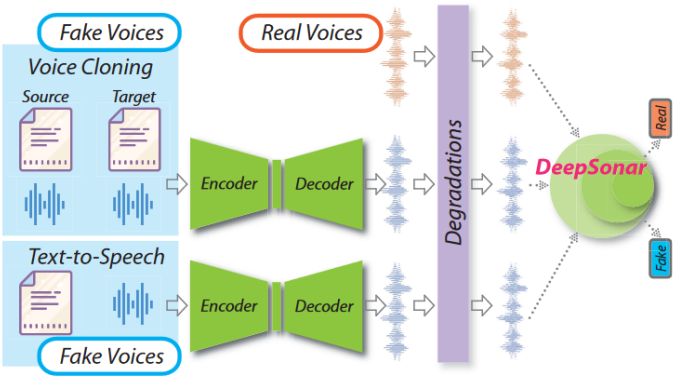


Figure 3.3: Text-to-speech and voice cloning data flow diagram. Retrieved from [42].

3.0.3 Image or video deepfakes

The list of the following deepfakes is based on the work R. Tolosana, et al. [20]:

- Identity swap – Replacing the face of subject with the face of target as shown in fig. 3.4. There are two different approaches, classical computer graphics-based technique and deep learning technique. Generally, the process of swap could be described as face detection, cropping, extraction of intermediate representations, synthesis of new face, and blending the generated face.

Figure 3.4: Examples of real and fake identity swap images. Retrieved from [20].

[1] https://github.com/SolomidHero/real-time-voice-conversion
[2] https://www.resemble.ai/speech-to-speech/

- Full puppet – Method related to identity swap allows creation of so-called puppet. One person (master) is source of facial expression and body movements that are mapped onto target person as shown in fig. 3.5. [18]

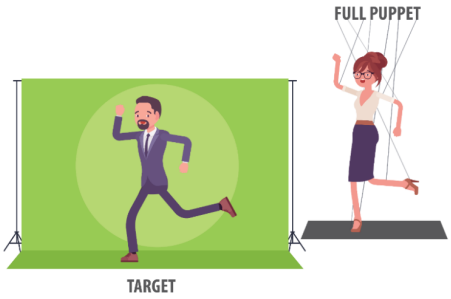


Figure 3.5: Full puppet technique visualisation. Retrieved from [21].

- Morphing – It is a type of manipulation that is used to create artificial biometric face samples. Final face contains resemble biometric information of two or more individuals. It should be possible to be successfully verified by biometrics systems for all individuals who were source for given deepfake. Fig. 3.6 shows an example of a morphed image.

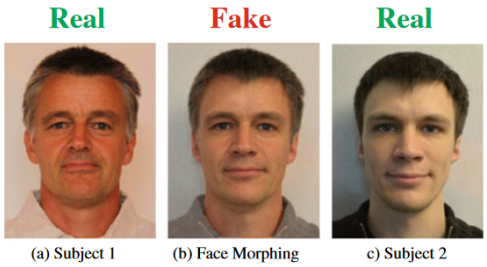


Figure 3.6: Examples of fake morphed identity from Subject 1 and Subject 2. Retrieved from [20].

- Attribute manipulation – Face editing or face retouching technique involves modifying some attributes such as length or color of hair, color of skin, sex, age, adding glasses or other artefacts, and more. Fig. 3.7 shows an example of this technique.

Figure 3.7: Examples of real and fake attribute manipulation category. Retrieved from [20].

- Expression swap – Modifying facial expression of the subject as shown in fig. 3.8. This technique is used as one of part for full puppet.

Figure 3.8: Examples of real and fake expression swap category. Retrieved from [20].

- Audio/text to video – This method related to expression swap synthesising facial expression from audio or text. It is also known as lip-sync deepfakes. Diagram in fig. 3.9 shows how this method works.

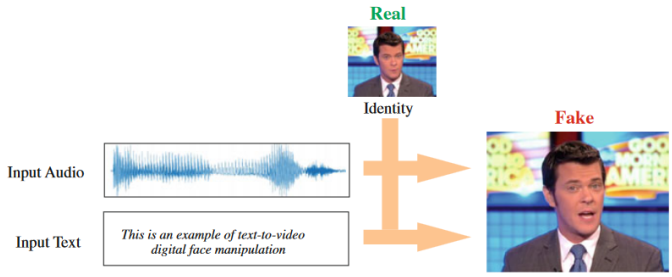


Figure 3.9: Examples of real and fake audio/text to video fake category. Retrieved from [20].

Creating deepfakes nowadays is complex task and many deepfakes is using more techniques so that they could be included into more than one category. Attackers can possibly create identity swap deepfake and after that apply attribute manipulation to this deepfake to tune final results.

Chapter 4

Deepfake detection

As we stated, humans are not good at recognizing deepfakes. Creating deepfakes could leave visible defects (e.g., blurring or misalignment on edges in the image). A. Firc [10] summarizes the list of features to focus on when spotting fakes for humans.

- Facial features – eyes and their movement, eyebrows, glasses, facial expression, hair and facial hair, skin, lips, teeth
- Body features – body position and posture, body movements
- Voice features – unusual tempo, end of words, fricatives, conversation
- General indices – blurring or misalignment on edges, inconsistence noise or audio

This list points to critical parts where defects, which were created during the creation of deepfake, could be spotted. Generation of deepfakes is getting better, plus other masking techniques are being used. The big problem for detection is lossless compression, several chained resizes of an image, application of noise, etc. Basically, all methods that lead to some degree of data loss, but without noticeable change for the naked eye or ear. When generation of deepfakes is not good enough manual post-processing could be used for polishing results (images – Photoshop, GIMP, etc.).

Machine detection could be divided into two categories: standard algorithms looking for physical inconsistency and digital integrity, using the same features as Firc described. Other methods are based on machine learning. The same "masking" techniques listed before (compression, chained resizes, etc.) have the same effect on machine-based detection. Any data loss prevents the usage of reliable methods such as frequency analysis. Still, computers perform better than humans because they can also use different features (e.g., pixel-level features), especially neural networks trained for deepfake detection. [40]

The problem of neural networks is that we do not know what it is learnt to detect. It depends highly on training dataset. Probably the worst case scenario would be only recognizing suspicious images containing traces after "masking" techniques such as double compression, noise patterns, etc.

Another problem of detection algorithms is bad generalization. Most of methods are trained on single domain deepfake (e.g., identity swap), which means they are not able to recognize deepfake from different category. When methods are trained on multi-domain datasets, accuracy is going down. [22]

4.1 Image/Video detection methods

As stated, before most of methods for deepfake detection are targeting only single domain. This section will describe examples of proposed detection methods for image/video deepfakes. There are more conventional approaches and also more "exotic".

P. Majumdar, et al. referring to multiple detection methods for image retouching (makeup, filters) and alternation (fully synthetize faces, morphing) [22]. Most of them use the same pattern which could be described as specific feature extraction followed by support vector machines (SVM) for classification. One of the methods proposed detection of images using face patches as input in the deep Boltzmann machine for feature extraction and SVM for binary classification. Another method uses softmax probabilities as features for the SVM. Other methods, for example, using convolutional neural networks. [22]

L. Spreeiwers, et al. made research on using local binary pattern with SVM for morphing detection [38]. A single LBP histogram contains 59 feature values, which means that for a 3×3 layout, the feature space has 531 dimensions. The SVM classifiers are trained on between 650 and 1,000 samples. They also stated that EER increases to above 20 % while adding Gaussian noise to the deepfakes images.

Non-conventional detection method is heart rate estimation (remote photoplethysmography) by J. Fierret [17]. They are trying to estimate heart rate from video and evaluating frame-by-frame. There are other human physiological processes that could for be used instead heart rate such as blood oxygen or breath rate. The score oscillates during the video and final decision is based on the mean/median/QCD score.

There are many other methods, and each will have its pros and cons, but as we can observe, SVM classification with large range of different feature extractors. Another rising group of detectors is using CNN. There are not many researches using CNN as SVM, but results seems to be promising as we can see in researches [34] [29].

4.2 Voice detection methods

Voice detection complicates different languages; it is similar story to image/video deepfakes. There are face swaps, morphing, etc., and for voice there are different languages and dialects. Voice detection methods also copying trend from image/video detection methods. Support vector machines (SVM) with different feature extractors or CNNs.

Z. Almutairi and H. Elgibreen refer to multiple methods [5]. One of them uses the SVM model with Random Forest to predict synthetic voices based on a feature called Short-Term Long-Term. In this research, they compared SVM with many other classifiers such as Linear Discriminant, Quadratic Discriminant, Linear SVM, weighted K-Nearest Neighbors, and SVM outperforms all of them. Other referred work uses combination of two CNN, 1-D CNN and Siamese CNN. The Siamese CNN contained two identical CNNs that were the same as the 1-D CNN, but concatenated them using a fully connected layer with a softmax output layer. [5]

4.3 Analysis of existing tools for detecting deepfakes

Tools for deepfake detection are slowly getting from command line tools for experts to online tools with user-friendly interface. There are not many tools of this kind and some of them are not free to use. The following lines describe two available tools in a market.

4.3.1 Deepware

The Bosnia and Hercegovina recognize danger of deepfakes, while their parent company researched methods to develop an AI-based antivirus engine. Deepware company was founded to develop scanner for deepfake recognition.

Deepware provides REST API with web UI[1], mobile android application. The backend of this project with pre-trained models is accessible on their GitHub as Python command line tool[2].

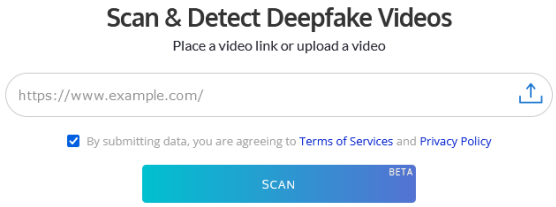


Figure 4.1: Deepware scanner input form

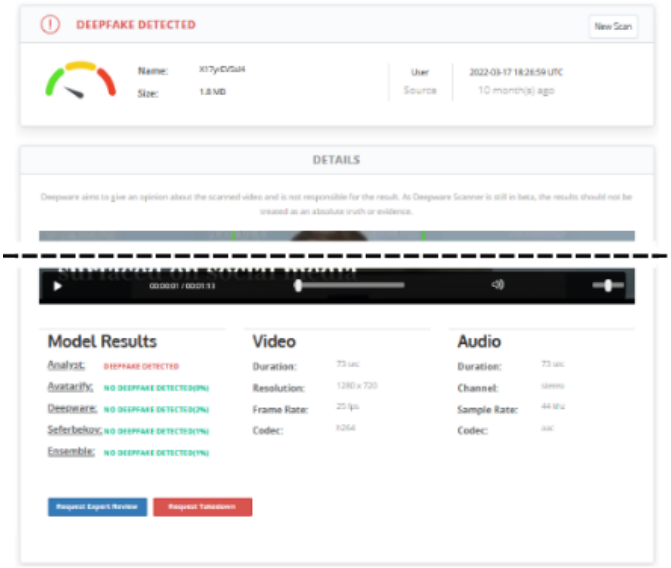


Figure 4.2: Results of Deepware scanner containing probability of deepfakes from multiple detection methods

The web application is stating that the project is in Beta, but unfortunately the last commit to the GitHub repository was made on 7th of June 2022 reported at the time of writing this thesis.

The tool provides a simple user interface to scan only videos. The user can input the link to the video (e.g., YouTube) or upload the video file directly as shown in fig. 4.1 . There are many supported video formats. The only limitation is the length of the video, which does not have to be longer than 10 minutes.

[1] https://scanner.deepware.ai/
[2] https://github.com/deepware/deepfake-scanner

Processing of approximately 1 minute long video takes several seconds (3-10 seconds). Results contains video and audio metadata, results of multiple detection methods/models, and deciding whether it is a deepfake or not with gauge chart of confidence as we can observe in fig 4.2. To use their Rest API, you need to request authentication token. API provides the same functionality as web UI via three methods.

- POST /video/scan
- GET /url/scan
- GET /video/report

The first two methods execute scan on a video file or link and return report ID. Results report could be retrieved by last method and results are returned as JSON. Documentation also code samples for multiple programming languages on how to use API properly. The provided API can be integrated to other processes like mail communication scans or file upload filters.

	Correct (deepfake/original)	Incorrect (deepfake/original)	Scan failed
Celeb DF	10/7	0/0	3
FaceForensics++	4/5	1/0	10

Table 4.1: Deepware manual testing results

Deepware did not provide requested API key, so in table 4.1 you can find results of manual testing. The test contains 40 videos from two different deepfakes datasets (20 from CelebDF and 20 from FaceForensics++). The distribution between original and deepfakes videos was 1:1. The accuracy of detection is decent, but from 40 tested files 13 files did not go through scan, even repeatedly. It is a very small sample, so we leave it to the reader to draw conclusions.

4.3.2 Sensity

Sensity is very similar from the user perspective to Deepware. Based on the post on Sensity blog [36] from 2021 we can explore web UI of their application. The application is not publicly accessible and to obtain access, you need to request it. Sensity provides more tools related to cybersecurity, person identification, and verification.

Sensity allows for detection of images and videos by inserting files or referencing them via a URL link as shown in fig. 4.3. Sensity allows processing of quite small group of file types (png, jpeg, jfif, tiff, mp4, mov). Another limitations are for videos regarding their size (up to 30MB), length (10 minutes), and quality of videos (1440p). [36]

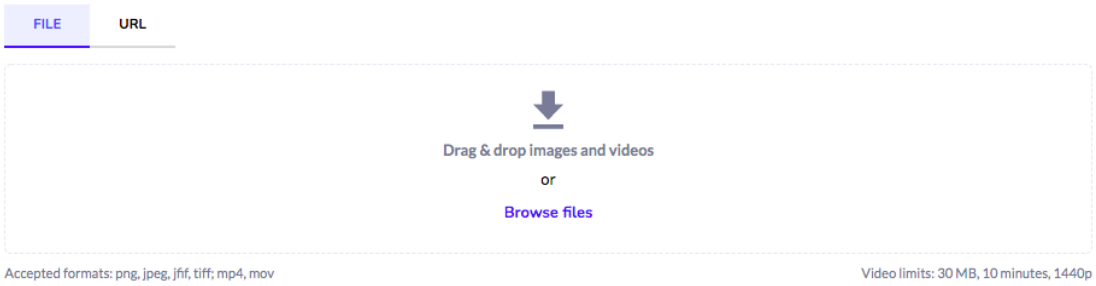


Figure 4.3: Sensity deepfake detection tool input form. Retrieved from [36].

The tool is capable of recognize only face swap and fully synthesized faces by GAN. It is not able to recognize morphed images or other deepfakes. For GAN-genereated faces, it is sometimes able to classify model generator. If deepfake is recognized tool show how confident he is, all shwon in fig. 4.4. Compared to Deepware, it provides image scans; on the other hand, it does not have mobile application.

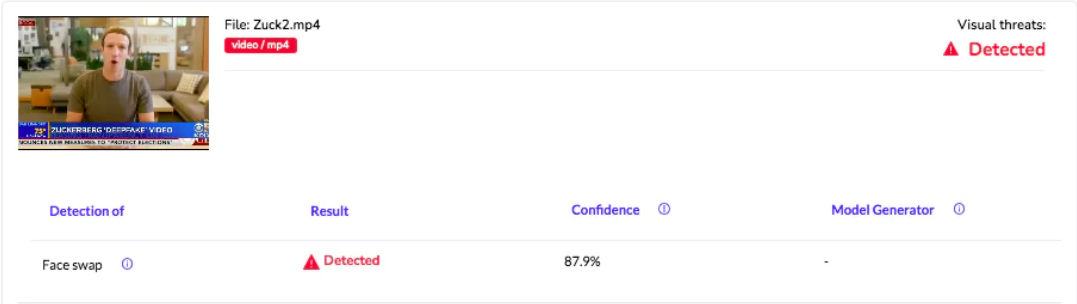


Figure 4.4: Sensity deepfake detection tool results. Retrieved from [36].

4.3.3 Other

Below you can find a list of other open source detection methods. These methods usually require a good knowledge of information technologies and programming to make them work on your computer. There is no guarantee that these methods really work, but two projects were selected for integration into the framework arising in this work.

- Video
  - dfdc_deepfake_challenge (https://github.com/selimsef/dfdc_deepfake_challenge)
  - DeepFake Detection (https://github.com/dessa-oss/DeepFake-Detection)
  - fakeVideoForensics (https://github.com/bbvanexttechnologies/fakeVideoForensics)
  - Deepstar (https://www.zerofox.com/deepstar-open-source-toolkit/)

- Audio
  - AudioDeepFakeDetection (https://github.com/MarkHershey/AudioDeepFakeDetection)
  - specrnet (https://github.com/piotrkawa/specrnet)

Chapter 5

Architecture and technologies analysis

The goal of this work is to develop a deepfake detection framework capable of serving various client applications and an exemplary web plugin that communicates with framework and clearly displays the results to the user. We already analyze the market and test existing tools. Because there is no external contracting party defining their need, we will define requirements by ourselves. Requirements will be divided into two categories, functional and non-functional requirements. It is important to define not only functional logic, but also some limits on processing time, number of users, etc.

5.1 Requirements

As we already said, the framework has to support multiple different types of clients (e.g., mobile, web), so there needs to be a properly defined communication interface between the client applications and the framework itself, which will act as server in this case. Framework allows dynamic changing detection methods, which means it is able to add, remove, or edit detection method. This implicates multiple detection methods at one time for image, video, and voice. Editing detection method means updating the model or constant parameters.

The platform should support as many file types as possible and adding a new method should not affect methods already presented in the framework, within supported file types. Different detection methods have different models depending on their design, so model management will be wrapped completely into the detection method.

The framework should collect statistics and hardware metrics, but the collection of personal information should be omitted. All collected metrics will be used for future improvements of detection methods and optimizing operating costs and performance. We would like to have a short response time with as low operating costs as possible. Those two parameters are in contradiction, so there has to be some balance between them.

Different cloud services offer their own tools for collecting metrics, but our system should be operating platform independent. That means framework needs to be consist of application, which collects metrics and provides them to other parts of framework, or person who operates the framework.

From a number of users perspective, the platform will handle up to hundreds of users at one time, and with enough resources the framework should handle even more. Scalability of computationally intensive parts of the framework will help meet these requirements.

The client application has to be simple so that anyone can use it. Users should be able to upload files or put a link to a file. The web plugin allows access to DOM of the webpage so an optional extension for client application will be selection of web element for detection. Application then retrieve metadata from selected element automatically.

Working with provided links or elements on specific pages brings several complications such as authentication. The framework does not have user context, so he is not able to retrieve all data that user has access to. We will not deal with these issues and if the file cannot be access, the processing request will not be created. It will be one of possible future improvements to the framework.

Because it should cover a large group of possible users, results have to be easy to understand and also provides information for experienced users. Last but not least, the client application should enable users to send feedback.

There are several web browsers in market that cover almost all audience, so the developed web plugin should be portable among multiple web browsers. Today it is standard, but it is necessary to mention that the application should be secured. It will affect code of framework, client application, and environment itself (hardware, operating systems, ...). All work is open-source and accessible on the public GitHub repository[1].

Summary of functional requirements:

- Adding, removing, and editing (changing parameters and models) detection methods
- Multiple detection methods at once
- Collecting statistics and feedback
- Detection of image, video, and voice
- Detection of file, URL link, or selected HTML element (optional)
- Understandable presentation of results
- Security

Summary of non-functional requirements:

- Scalability
- Small response time
- Low operating costs
- Portability of the web plugin among multiple web browsers (optional)
- Open source

5.2 Containerazation

The framework requires dynamic management of the detection methods. It means that the framework can contain one or twenty different detection methods. Each method could be developed using different programming languages or technologies. Model management will

[1] https://github.com/PlayerBerny12/VUT-DIP-Code

be encapsulated and handled by each detection method individualy. This leads to some isolation of methods with a well defined communication protocol.

Scalability of framework, isolation and resource management, all this together will be reached via containerization. There are many technologies dealing with containers, such as Docker, Podman, LXC. When we start counting orchestration with automatic scalability, the number of technologies drops down.

Azure and AWS offer their own container orchestrators: Azure Container Instances [23], Amazon Elastic Container Service [6]. As already stated, the framework should be operating platform independent, so looking only for orchestrator supported by all cloud providers, we discover Kubernetes. Google Cloud, AWS and Azure support Kuberentes, and no other alternative supported by all of them exist. [23] [6] [13]

Docker is probably the most widely used technology for containerization. Podman is another containerization technology having similar API to Docker. Podman is newer and has different architecture than Docker which results in a more secure operation and fewer resources required for running [3]. But still, we will stick with duo Kubernetes and Docker because of the huge community and good support of Docker in Kubernetes and also less concerns about possible issues during development.

5.3 Framework

The framework is a collection of detection methods, an interface for the client application (receiving requests, sending report response), and orchestrates/arranges all communication. It will be built into several containers and one of the containers will be providing client application interface. It could be Rest API [32], GraphQL [39] or custom protocol. It is not a good idea to develop a custom communication protocol, and because of simplicity of interface we will use the most used one, Rest API. For this purpose, we can use a huge number of different technologies such as ASP.NET Core, Flusk, Django, Ruby on Rails, etc. Basically all named technologies meet defined requirements (response time, number of users, ect.) because most of them are covered by microservice design and they are able to run in containers. Our choice is ASP.NET Core because it has a huge community, Linux support, many external libraries, open-source development, it is suitable for bigger projects, and it has good speed of program development [24].

Communication among containers will be through message broker. There are several alternatives such as Memphis, RabbitMQ, or Apache Kafka. All of them are open source and possibly suitable for our use case. The final choice is RabbitMQ because of small resource consumption, a quite large community, and this project is developed since 2007. That means we can count on future development and updates. [16]

The framework also needs an application for collecting metrics. Here, we chose Prometheus [31] and Grafana [15]. Both tools are free to use and open source. Prometheus is collecting metrics, but it is not a good visualization tool; for this purpose there will be Grafana. Grafana provides dashboards where all metrics could be presented as graphs, etc. Reasons why we take Prometheus and Grafana is because they have good cooperation and RabbitMQ has metrics compatible with Prometheus [41].

All previously selected technologies also have good support of Kubernetes and in some cases provide custom Kuberetes operators for easier deployment. More details about it are provided in chapter 8.

5.4 Web browser plugin

Web plugins are based on web technologies such as HTML, CSS, JavaScript, TypeScript, etc. Because of portability and good integration, we will choose HTML, CSS, and Typescript. Typescript is a strict syntactical superset of JavaScript and adds optional static typing to the language. It is also compiled into JavaScript [25].

There are several Typescript frameworks that speed up development of application and make portability among different browsers easier. Angular seems to be as good viable option and it is frontend framework, with which we have most experience [12].

Browser extensions are integrated through a standard called "Web app manifest". It is the definition of manifest file used for integration extensions to browser, now in version 3. Standard defines the structure of the manifest that allows the configuration of different types of capabilities. All browsers have partial or full support. Using this manifest should help to integrate the extension with browsers such as Google Chrome, Microsoft Edge, or Mozilla Firefox. [30]

Chapter 6

Framework architecture

The architecture must reflect that each detection method could be developed in different architectures. We can isolate the detection methods from each other and wrap them into independent services. We do not need communication among all detection methods because they do not cooperate or share any data. In this case, the microservice architecture meets all defined requirements.

6.1 Microservice architecture

Microservice architecture is the style of developing applications. Example of design can be observe in fig. 6.1. The microservice allows the application to be separated into smaller logical independent parts. The service then has its own realm of responsibility and they can communicate with each other. Containers are a well-suited microservice architecture example. [14]

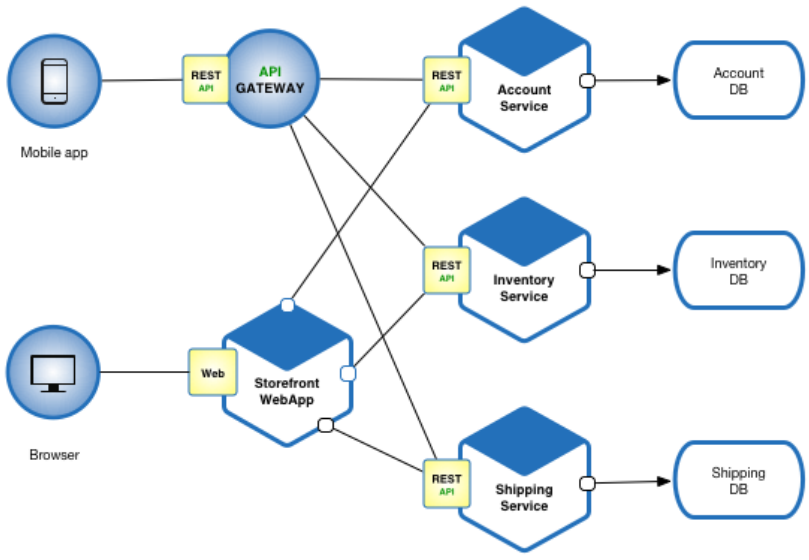


Figure 6.1: Microservice architecture of fictitious e-commerce application

Microservices improve fault isolation that non-functional service should not affect others, in best case scenario. In real cases microservices depend on each other and this might lead to a problem. For example, if one service contains a memory leak, it is not propagated into

other ones. Another benefit was already mentioned, and it eliminates commitment to one technology stack. One service has better maintainability because it should be small (better understandability, faster tools), which leads to more productive developers. [33]

This architecture also has several drawbacks. It increases the complexity of architectural design and additional implementation of cross-service communication. When saving data to database, developers have to deal with distributed system because each service has its own independent database. [33]

6.2 High-level architecture

The user communicates directly with the Rest API endpoint. This part of the design may be labelled as master process because it is handling request, preparing and validating data, selecting which type of processing should be used. After processing is done, it collects all results and distributes them back to the user. Fig. 6.2 contains high-level design of framework with data flow.

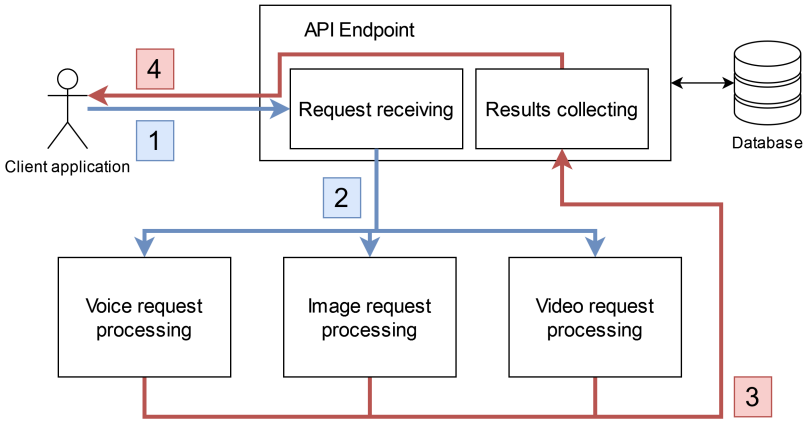


Figure 6.2: High-level design of whole framework

The architecture separates voice, image, and video detection into an independent unit. Each unit contains a processing queue where the request will be assigned by the master process/Rest API. The queue is serviced by one or multiple processing units that contain all related detection methods as shown in fig. 6.3.

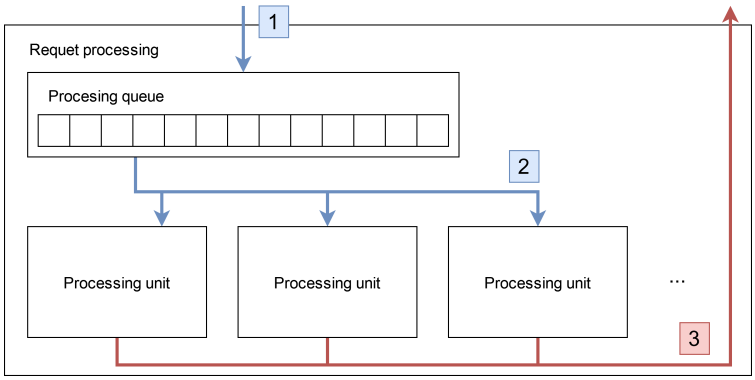


Figure 6.3: Request processing detail

The processing unit works as a parallel pipeline. Closer look at design is in fig. 6.4. Some detection methods require input data in specific format, so the first step is optional data preparation. Some methods are wrapping data preparation by themselves. The next step is the detection method that decides whether the input data contains deepfake or not. Detection methods are different, so are their results, so we need to properly generalize and also normalize them.

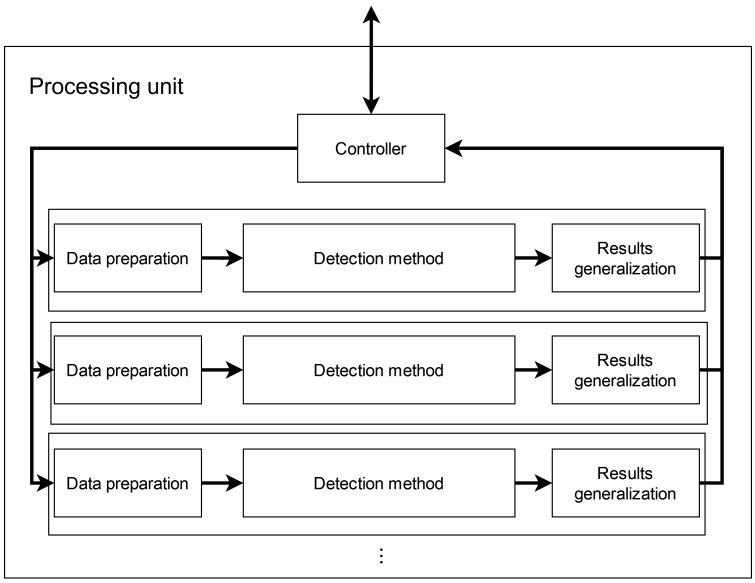


Figure 6.4: Processing unit pipeline

6.3 API Endpoint

As mentioned before, API endpoints communicate directly with user application and with backend processing mechanism. Processing takes same time, so user application creates request and wait asynchronously to be processed by the framework. The API for this purpose contains "detect" and "request" methods. "Detect" methods start processing and"requests" methods manage already running process. There are other support methods for possible health check or providing feedback. All methods are described in following list:

- /ping
  - GET /ping – healtcheck endpoint
- /detect
  - POST /detect/file – creates request with file in HTTP request body
  - POST /detect/link – creates request with link to a file in HTTP request body
- /request
  - GET /request/results/ – returns results of processed request or empty results if request is in process
- /feedback

– POST /feedback/<request_id> - collects user feedback with optional parameter request_id

Each detection method has different processing times. There are two options to return results to user: partial results from the detection methods already completed or returns only when everything is completed. Because there is no need to show partial results, the second option will be used. This requires the use of another queue for respons or some sort of database. The framework needs to calculate the overall score and this requires all results from all methods. Calculating the total score requires considering the univariate methods as well as their domain differences. A single method will usually focus on a single domain (e.g., face swap). We leave the exact definition to the implementation section.

The method `/request/results` could be possibly replaced by websockets which will increase complexity, but on the other hand a user application could better report status of the running process. It is one of the possible extensions to the framework.

6.4 Database

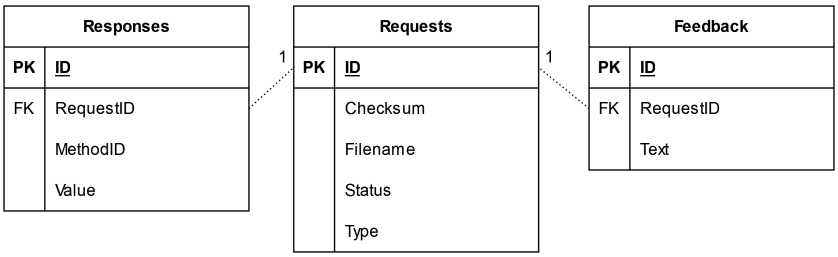


Figure 6.5: Entity-relationship model

6.5 Request processing and processing unit

API endpoint placing all new requests in the request queue. One of the available processing units takes the first request form queue. This request contains all the information such request ID, data for detection, and metadata for this data.

The first step in the processing pipeline is optional data preparation. The detection method could be trained only on a specific resolution of the video/image or on a specific length of the voice sample. This means data processing unit has to be designed for a given detection method. When data are prepared for detection, they continue to the detection unit.

After detection, the same case as optional data preparation is optional data generalization/normalization. Because results need to the be correctly presented to user and the framework needs to calculate over all scores, all results need to the be normalized to same scale and "units". When results contain more specific information, in this case it needs to be generalized. Results could contain field with additional/extra information about results which could also be presented to the user. Results are then pushed to the results queue or database, depending on implementation.

6.6 Monitoring

...

6.7 Containerization and scaling

Detection methods have to be containerized, and to decrease complexity data preparation and results generalization will be encapsulated together with detection method. The processing unit then will be the collection of containers/pods.

The processing unit then can be clone/scale up when needed to process more requests. Scaling up can be done automatically by setting specific rule in container orchestrator. To save costs we can scale down when a peek of request disappeared.

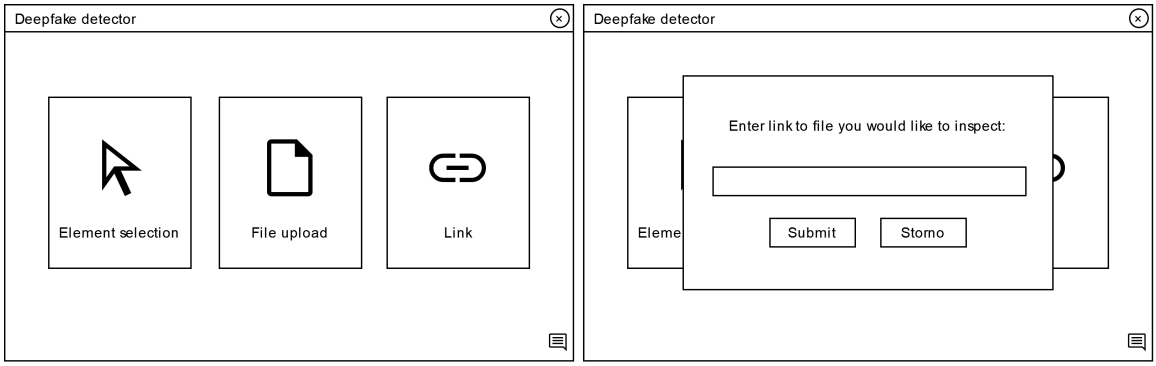
Chapter 7

Client architecture

The use case of the client application is straightforward, so there does not need to be use case or data flow diagrams. The application will try to prepare the file for inspection and send it to the server framework. The next section contains several wireframes of how the client application should look like with description.

7.1 Web browser plugin

It should allow the user to insert a file via file upload, link, or HTML element containing a targeted file. In fig.7.1 we can see type insertion selection, for file upload the user will be prompted by system file upload picker. When user pick link as input, floating windows will pop up, and user then can insert URL to file. Those two selections are the same as other tools in market also provide. Optional improvement will be element selection which switches the application to interactive mode where user can point to HTML element and application will try to retrieve metadata of image or video directly from HTML.

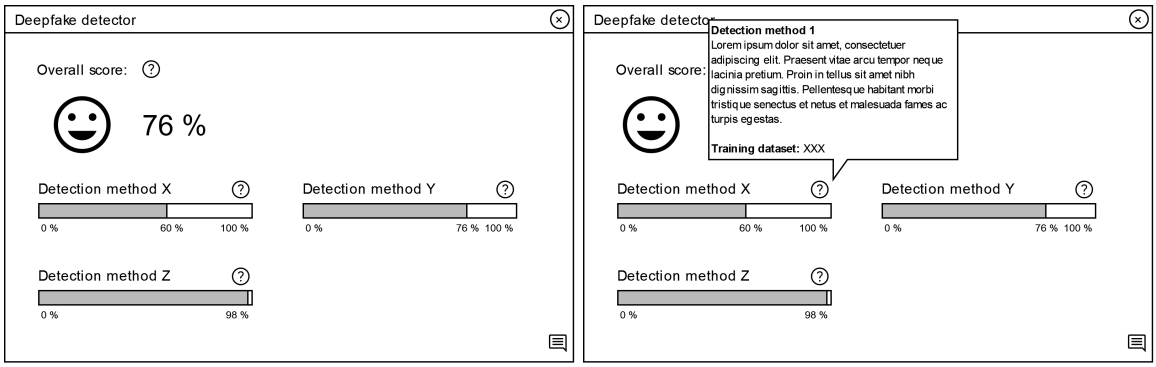


(a) Input type selection (b) Link input in floating window

Figure 7.1: Input type selection screens

Because detection framework will contain several detections method we need to show the result of each method independently. It could be a little confusing for user so there will also be overall score which interprets/generalizes all the results of each method. The overall score will indicate the results as a percentage and as emoticon. The palette will contain four emoticons shown in fig 7.3. For better understanding there are question marks

in the UI that after mouse hover shows tooltip with description. Whole results screen can be view in fig. 7.2.



(a) Results view (b) Tooltip with description of detection method

Figure 7.2: Results view screens

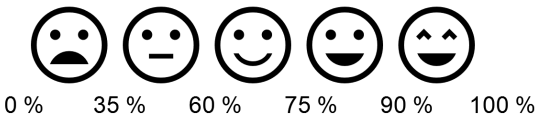


Figure 7.3: Palette of emoticons indicating if inspected file contains deepfake or not

At the bottom of each screen there is an icon that enables the user to the send feedback to application. After clicking on this icon floating window will appear, shown in fig 7.4. When user will be sending feedback on the result page, the internal result ID will be added as an additional parameter to the feedback message.

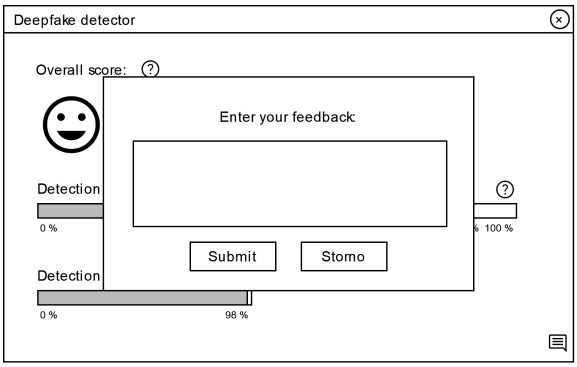


Figure 7.4: Feedback form in floating window

Chapter 8

Framework implementation and deployment

The framework was implemented on the previously defined architecture. The architecture primarily dealt with the main functionality of the framework, but it also covered supporting systems, for example, systems for collecting metrics used in scaling, etc. The overall implemented platform can be seen in diagram in fig. 8.1. The system could be divided into processing - API endpoint, message broker, processing units; and monitoring - metric scraping, observability platform, etc.

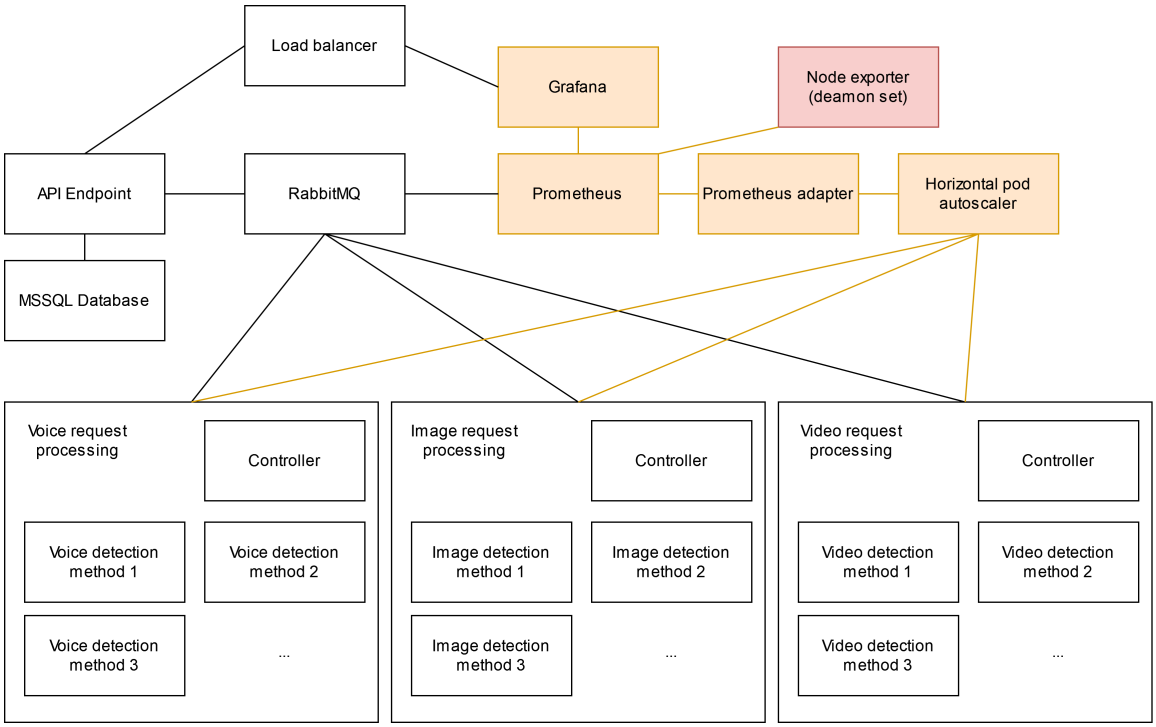


Figure 8.1: Diagram of implemented framework

The individual parts are described in more detail in the following sections. The whole system is designed to work in Kuberentes, the container orchestration tool. The overall implementation involved the development of an endpoint API and a processing unit con-

troller. Subsequently, it was necessary to configure other systems, such as the message broker, the database, or the metrics scraper. Finally, it was necessary to integrate the different selected detection methods; for this purpose a template was created to make the integration as simple as possible.

8.1 API endpoint

The interface for the client applications is a REST API developed in ASP.NET Core 7 with EF Core for relational object mapping over the MSSQL database. The RabbitMQ.Client nuget package is used to communicate with the RabbitMQ message broker. The application uses a software architecture similar to MVC, but without a view implementation. Views are completely replaced by the client application, but it is handled separately. Controllers receive HTTP requests containing individual input parameters that are mapped to objects called viewmodels. When mapping to a viewmodel or directly after it, validations are performed. These include type checking, input length, or when uploading a file, checking for pairs of supported extensions and MIME Types.

If all input data is correct, the query processing can continue. In case of detection, the file is saved to a shared storage where the API endpoint is the only one with write permissions. Other parts of the framework can only read from this storage. The SHA256 checksum of that file is then calculated and compared against the database records. If this checksum is already in the database, the request ID containing this checksum is returned and the file is deleted. When the client queries the results, it gets the answer immediately because it is already in the database and the system is not unnecessarily overloaded with recomputation. However, if the checksum is not found in the database, a new request is created with the status „processing“ and stored in the database. Subsequently, a message is sent via the message broker forcing the detection of this file. The message contains the same information as the database request (id, checksum, filename, status, type).

The OutputService class implements a function that handles the background consumption of messages containing results. When the server starts, it registers an event handler that is called when a message is received in the outuput queue. All the results are stored in the database and the status of the processed request is also changed to „done“. It is also necessary to delete the detected file from the shared storage. The incoming message contains the request ID and a list of responses, where each individual response contains the ID of the detection method and the resulting value. The list may also be empty, indicating an error or that a single method was unable to process the file. Detection methods can only support processing certain file types, so if no method is able to process the file, the list will be empty.

The results are issued to the client upon request via the „requests/results“ endpoint. In case the request processing is not complete, an empty response (HTTP 204) is returned. After (un)successful processing, an object containing the request ID, the overal score, and a list of results consisting of the detection method ID, the detection method name, the description, the name of the training dataset, and the result value is returned to the client. Information about the detection method is read from the configuration file „appsettings.json“ based on detection method ID. The client must perform long pooling until it receives the first non-empty response. The client should timeout in case of a complete system failure.

8.2 Message broker

RabbitMQ, how it works, deployment

8.3 Processing unit

Controller + detection method communincation, deployment

8.4 Monitoring

Prometheus, Grafana - metrics scraping, what is collected and why

8.5 Scalability

Prometheus adapter, horizontal pod autoscaler based on metrics

8.6 Deployment

The deployment of the entire framework is divided into several Kubernetes manifest files, where individual Kubernetes objects are defined. These are persistent entities that represent parts of the system, and each object is of a specific type. The basic types are, for example, pod, service, deployment, etc. If we want to work with these objects, we can only do so through the Kubernetes API. It is possible to define a new API in Kuberentes and other new object types in it. This functionality is used by so-called operators, which are designed to automate repetitive procedures or to facilitate the deployment of a more complex system into another. When deploying this solution, it is important to consider that the RabbitMQ Cluster Operator and Prometheus Operator were used.

...

Chapter 9

Detection methods integration

how to integrarte new method; what is required from container with detection methode; what manifests needs to be change + caution about detection method resources ...

9.1 AudioDeepFakeDetection

train 6 differnet methods (LJ + WaveFake dataset); results of training; automaticlly spliting dataset; own implementation of evaluation of one file; dockerfile optimalization

9.2 fakeVideoForensics

used pretrained model (no description how to train); dockerfile optimalization

Chapter 10

Client application implementation

angular + angular materials

10.1 Functionalities

element selection not implemented, file upload + link is working

10.2 Communication with framework

pooling messages with timeout

10.3 Supported browsers

tested in chrome, firefox (working but not able to upload file because of bug in browser) and edge

Chapter 11

Test experiment and results

...

11.1 Datasets

which datasets were used for testing (CelebDF, FaceForensics++, LJ + WaveFake, ASV2021) - how was subset created; stats - size, count

11.2 Test case 1 - accuracy

accuracy of detection methods - 4x (250 originals + 250 fakes) - results no so good...

11.3 Test case 2 - small bursts

small burtst (100 files - batches of 5) - resources (cpu, ram), number of replicas, time

11.4 Test case 3 - congestion

small burtst (100 files - all in) - resources (cpu, ram), number of replicas, time

11.5 Tests evaluation

discussion about testing - detection methods not so good but it is replication on their experiment

11.6 Cost analysis

testing system - cpu, ram - average daily cost

Chapter 12

Conclusion

everythings implemented and it is working, :)

Bibliography

[1] *What Is the Necessity of Bias in Neural Networks?* [https://www.turing.com/kb/necessity-of-bias-in-neural-networks]. [cit. 2022-12-29].

[2] Ahmed, S. Who inadvertently shares deepfakes? Analyzing the role of political interest, cognitive ability, and social network size. *Telematics and Informatics*. 2021, vol. 57, p. 101508. DOI: https://doi.org/10.1016/j.tele.2020.101508. ISSN 0736-5853. Available at: https://www.sciencedirect.com/science/article/pii/S0736585320301672.

[3] Alex Gamela, R. F. *Podman vs Docker: What are the differences?* [https://www.imaginarycloud.com/blog/podman-vs-docker/]. [cit. 2023-05-09].

[4] Alexandra Baruffati. *Chat GPT Statistics 2023: Trends And The Future Perspectives* [https://blog.gitnux.com/chat-gpt-statistics/]. [cit. 2023-05-06].

[5] Almutairi, Z. and Elgibreen, H. A Review of Modern Audio Deepfake Detection Methods: Challenges and Future Directions. *Algorithms*. 2022, vol. 15, no. 5. DOI: 10.3390/a15050155. ISSN 1999-4893. Available at: https://www.mdpi.com/1999-4893/15/5/155.

[6] Amazon. *Containers at AWS* [https://aws.amazon.com/containers/]. [cit. 2023-05-09].

[7] Amos, Zachary. *Hybrid Vishing Attacks Skyrocketing: What to Know* [https://itsupplychain.com/hybrid-vishing-attacks-skyrocketing-what-to-know/]. 2022 [cit. 2022-12-28].

[8] Deotte, C. *How to choose CNN Architecture MNIST* [https://www.kaggle.com/code/cdeotte/how-to-choose-cnn-architecture-mnist]. [cit. 2023-01-04].

[9] Ferrara, M., Franco, A. and Maltoni, D. The magic passport. In: *IEEE International Joint Conference on Biometrics*. 2014, p. 1–7. DOI: 10.1109/BTAS.2014.6996240.

[10] Firc, A. *Applicability of Deepfakes in the Field of Cyber Security*. Brno, CZ, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/23761/.

[11] Firc, A. and Malinka, K. The dawn of a text-dependent society: deepfakes as a threat to speech verification systems. In: *SAC '22: Proceedings of the 37th*

ACM/SIGAPP Symposium on Applied Computing. Association for Computing Machinery, 2022, p. 1646–1655. DOI: 10.1145/3477314.3507013. Available at: https://www.fit.vut.cz/research/publication/12595.

[12] Google. *Introduction to the Angular docs* [https://angular.io/docs]. [cit. 2023-05-09].

[13] Google. *What is container orchestration?* [https://cloud.google.com/discover/what-is-container-orchestration]. [cit. 2023-05-09].

[14] Google. *What is Microservices Architecture?* [https://cloud.google.com/learn/what-is-microservices-architecture]. [cit. 2023-01-09].

[15] Grafana Labs. *What is Grafana?* [https://grafana.com/oss/grafana/]. [cit. 2023-05-09].

[16] Hasan, T. *6 Top Message Brokers for Modern Applications* [https://geekflare.com/top-message-brokers/]. [cit. 2023-05-09].

[17] Hernandez Ortega, J., Tolosana, R., Fierrez, J. and Morales, A. DeepFakes detection based on heart rate estimation: single-and multi-frame. In: *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Springer International Publishing, 2022, p. 255–273. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[18] Homeland Security. *Increasing Threat of Deepfake Identites* [https://www.dhs.gov/sites/default/files/publications/increasing_threats_of_deepfake_identities_0.pdf]. 2021 [cit. 2022-12-16].

[19] Ibsen, M., Rathgeb, C., Fischer, D., Drozdowski, P. and Busch, C. Digital Face Manipulation in Biometric Systems. In: *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Springer International Publishing, 2022, p. 27–43. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[20] Ibsen, M., Rathgeb, C., Fischer, D., Drozdowski, P. and Busch, C. An Introduction to Digital Face Manipulation. In: *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Springer International Publishing, 2022, p. 3–26. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[21] Korshunov, P. and Marcel, S. The Threat of Deepfakes to Computer and Human Visions. In: *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Springer International Publishing, 2022, p. 97–115. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[22] Majumdar, P., Agarwal, A., Vatsa, M. and Singh, R. Facial retouching and alteration detection. In: *Handbook of Digital Face Manipulation and Detection: From*

DeepFakes to Morphing Attacks. Springer International Publishing, 2022, p. 367–387. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[23] Microsoft. *Azure Container Instances and container orchestrators* [https://learn.microsoft.com/en-us/azure/container-instances/container-instances-orchestrator-relationship]. [cit. 2023-05-09].

[24] Microsoft. *Overview of ASP.NET Core* [https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0]. [cit. 2023-05-09].

[25] Microsoft. *TypeScript is JavaScript with syntax for types.* [https://www.typescriptlang.org/]. [cit. 2023-05-09].

[26] Mills, A. G., Kaewcharuay, P., Sathirasattayanon, P., Duangpummet, S., Galajit, K. et al. Replay Attack Detection Based on Voice and Non-voice Sections for Speaker Verification. In: IEEE. *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2022, p. 221–226.

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

[28] Müller, N. M., Pizzi, K. and Williams, J. Human perception of audio deepfakes. In: *Proceedings of the 1st International Workshop on Deepfake Detection for Audio Multimedia*. Association for Computing Machinery, 2022, p. 85–91. DOI: 10.1145/3552466.3556531. ISBN 9781450394963. Available at: https://doi.org/10.1145/3552466.3556531.

[29] Nguyen, H. H., Yamagishi, J. and Echizen, I. Capsule-Forensics Networks for Deepfake Detection. In: *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Springer International Publishing, 2022, p. 275–301. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[30] Pete LePage, T. S. *Add a web app manifest* [https://web.dev/add-manifest/]. [cit. 2023-05-09].

[31] Prometheus Authors. *What is Prometheus?* [https://prometheus.io/docs/introduction/overview/]. [cit. 2023-05-09].

[32] Red Hat. *What is a REST API?* [https://www.redhat.com/en/topics/api/what-is-a-rest-api]. [cit. 2023-05-09].

[33] Richardson, C. *Pattern: Microservice Architecture* [https://microservices.io/patterns/microservices.html]. [cit. 2023-01-09].

[34] Roy, R., Joshi, I., Das, A. and Dantcheva, A. 3D CNN Architectures and Attention Mechanisms for Deepfake Detection. In: *Handbook of Digital Face*

Manipulation and Detection: From DeepFakes to Morphing Attacks. Springer International Publishing, 2022, p. 213–234. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[35] Scherhag, U., Rathgeb, C. and Busch, C. Face Morphing Attack Detection Methods. In: *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Springer International Publishing, 2022, p. 331–349. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[36] Sensity Team. *How to Detect a Deepfake Online: Image Forensics and Analysis of Deepfake Videos* [https://sensity.ai/blog/deepfake-detection/how-to-detect-a-deepfake]. [cit. 2023-01-08].

[37] Shead, S. *Elon Musk's tweets are moving markets — and some investors are worried* [https://www.cnbc.com/2021/01/29/elon-musks-tweets-are-moving-markets.html]. 2021 [cit. 2022-12-28].

[38] Spreeuwers, L., Schils, M., Veldhuis, R. and Kelly, U. Practical Evaluation of Face Morphing Attack Detection Methods. In: *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Springer International Publishing, 2022, p. 351–365. DOI: 10.1007/978-3-030-87664-7_5. ISBN 978-3-030-87664-7. Available at: https://doi.org/10.1007/978-3-030-87664-7_5.

[39] The GraphQL Foundation. *Introduction to GraphQL* [https://graphql.org/learn/]. [cit. 2023-05-09].

[40] Verdoliva, L. Media Forensics and DeepFakes: An Overview. *IEEE Journal of Selected Topics in Signal Processing*. 2020, vol. 14, no. 5, p. 910–932. DOI: 10.1109/JSTSP.2020.3002101.

[41] VMware. *Monitoring with Prometheus & Grafana* [https://www.rabbitmq.com/prometheus.html]. [cit. 2023-05-09].

[42] Wang, R., Juefei Xu, F., Huang, Y., Guo, Q., Xie, X. et al. Deepsonar: Towards effective and robust detection of ai-synthesized fake voices. In: *Proceedings of the 28th ACM international conference on multimedia*. 2020, p. 1207–1216.