# Deepfake Detection Framework

Bc. Jan Bernard*

**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.

*xberna18@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

## 1. Introduction

The goal of this work is to examine the current state of deepfakes, first, generally, with later focus to detection methods. Based on the acquired knowledge design and develop the deepfake detection framework and client application utilizing it.

**[Motivation]** 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. Tools powered by deep learning allow unexperienced users to easily create trusted fakes. [1]

**[Problem definition]** 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. 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. The results of test in Fig. 1 certainly demonstrate that people's recognition ability decreases significantly as the quality of deepfakes increases [2]. [1]

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. There are many different deepfake types which is not making detection of those fakes easier. You can see only two of those types in Fig. 2.1 and Fig. 2.2 [3]. [1]

**[Existing solutions]** 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.

*Deepwere* - 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 pretrained models is accessible on their GitHub as Python command line tool [2].

*Sensity* Sensity is very similar from the user perspective to Deepware. Based on the post on Sensity blog [4] 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. Unfortunately we did not get an access to it.

**[Results]** In time of finishing this paper there were no tests results yet. Master thesis describing this work in more details will contain results of developed detecion framework with comparison to Deepware tool. Tests should be focused on how framework is working (time response, resource consumption) but also accuracy of detection.

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

## 2. Detection 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.

The client application communicates directly with the Rest API endpoint. API is responsible for 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 client application. Fig. 3.1 contains high-level design of framework with data flow.

The architecture separates voice, image, and video detection into an independent unit. Each unit contains a processing queue. The queue is serviced by one or multiple processing units that contain all related detection methods as shown in Fig. 3.2.

The processing unit works as a parallel pipeline. Closer look at design is in Fig. 3.3. 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. That is reason why we need to properly generalize and also normalize the results in last step.

**[Implementation]** API server operates in ASP.NET Core and it requires to have MSSQL server for saving information about requests and responses. For communication and request queuing message broker was used. RabbitMQ is one of most used one and it supports different technologies for clients, for our purpose we need .NET and Python.

Each processing has one controller which is Python script consuming requests from particular queue and spreading them among all detection methods via REST API. Detection methods requires to have one endpoint called 'detect' which is registered in controller. REST API was used because is easy to understand and also implement. Framework provides template written in Python utilizing FastAPI package for this purpose. This specific endpoint will execute processing pipeline and return results back to controller at the end.

When controller collects all results it send message back to ASP.NET Core application via RabbitMQ. All results are stored in database and provided back to client application upon request.

Whole framework is designed to be run in Kuberentes cluster, that means each component is docker container with prepared Kuberentes manifest for proper deployment.

### 2.1 Client application

It should allow the user to insert a file via file upload, link, or HTML element containing a targeted file. In Fig. 5.1 we can see type insertion selection. Similar functionality provides two previously mentioned tools in market. 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.

Because detection framework will contain several detections method we need to show the result of each method independently. There will also be overall score which interprets/generalizes all the results of each method. Whole results screen can be view in Fig. 5.2.

## 3. Conclusions

Right now framework contains six audio different detection methods. It is able to process given files and return results back. Framework should be able to scale up and down based on incoming traffic. Everything needs to be properly tested. Accuracy of detection highly depends on quality of detection methods which are not in scope of this work.

Client application communicates with framework and properly showing returned results. Even without proper tests results we can declare that framework and client application is working as expected. By this we successfully met our goals of this work.

## 4. Acknowledgment

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.

## References

[1] Luisa Verdoliva. Media forensics and deepfakes: An overview. *IEEE Journal of Selected Topics in Signal Processing*, 14(5):910–932, 2020.

[2] Mathias Ibsen, Christian Rathgeb, Daniel Fischer, Pawel Drozdowski, and Christoph Busch. *An Introduction to Digital Face Manipulation*, pages 3–26. Springer International Publishing, 2022.

[3] Pavel Korshunov and Sébastien Marcel. *The Threat of Deepfakes to Computer and Human Visions*, pages 97–115. Springer International Publishing, 2022.

[4] 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.