https://doi.org/10.2352/EI.2023.35.4.MWSF-381

Detection of Deepfakes Using Background-Matching

Stephanie Blümer; TU Darmstadt; Germany
Martin Steinebach, Raphael Antonius Frick, Niklas Bunzel; Fraunhofer SIT|ATHENE; Darmstadt; Germany

Abstract

In the recent years, the detection of deepfakes has become a substantial topic in image and video forensics. State-of-the-art blind detection methods can detect deepfakes from synthetic data sets with high accuracy. However, they struggle to classify deepfake material that underwent adversarial post-processing or fail to generalize to unseen video data. In this paper, a refined detection pipeline taking advantage of a semi-blind detection scheme is proposed. It combines background-matching with a state-of-the-art CNN-classifier. When classifying videos from the Deepfake Detection Challenge data set the CNN-classifier was previously trained on, the performance did not improve using the new detection scheme. However, the approach was able to achieve superior results on unseen data of the FaceForensics++ data set.

Introduction

Images and videos are frequently shared on the internet, e.g. on social networks or news websites. They can help to provide additional context or can convey a certain trust to an underlying story of an article or a social media post. However, as AI-assisted tools to create manipulated images and videos have become more accessible, they are also more frequently used to disseminate false information on the internet.

Deepfakes are a special kind of media manipulation, where the face inside an image or video can be exchanged with any desired face with the help of a neural network. While formerly developed for entertainment purposes, deepfake-algorithms also bear the risk of being utilized with malicious intent. During the Ukraine-War, deepfakes have been used to produce fake videos showcasing the presidents of both nations surrendering. Thus, images and videos shared on social media or by unverified sources should not be trusted blindly. Further, it shows the need for methods that allow to verify the authenticity of multimedia.

Over the past years, several approaches to detecting deepfakes have been proposed. While these methods are able to achieve a high accuracy on scientific data sets, they often fail to generalize well to unseen data in the wild. In Facebook's Deepfake Detection Challenge from 2020[1] the best performing approach was able to achieve 82.56% accuracy on the public test data set, yet only an accuracy of 65.18% during the evaluation on the private test data set. Often a major challenge is, that thresholds used for the classification of videos from one data set are often not suitable to be used for the classification of other data sets, thus, leading to many false classifications. Another challenge is, that many classifiers are not robust against subtle image post-processing, such as the introduction of random noise or heavy compression. Further, most classifiers work in a blind manner, meaning they solely take advantage of the test sample during the classification.

In this paper, a new deepfake detection pipeline is proposed that tries to find the authentic video, that was used as the source for the newly created deepfake video. It incorporates methods known from image hashing to find videos sharing the same background with the input video and allows the integration of previously developed blind detection methods. By this, the aim is to improve on the detection rate as well as to better generalize towards unseen data.

Background

In this section, we describe the concepts of deepfakes and perceptual (or robust) hashing.

Deepfakes

Deepfakes are a family of *deep learning*-based algorithms that enable to swap a person's face inside a video or image with any desired face. Tools that allow creating deepfakes have been openly available since 2017, with *deepfacelab* and *faceswap* being one of the most popular frameworks. When synthesizing a deepfake, the following steps are often used:

1. **Data Collection:** To create a deepfake, one requires an image or a video that shall serve as source material and which shall be modified as part of the process. Additionally, one needs images and videos of the face that shall be seen in the deepfake.
2. **Faceset Generation:** For each image or video frame inside the respective data sets, faces are extracted and aligned to build a faceset. In the past, the facial region exchanged by the deepfake algorithm lied between the chin and the eyebrows. However, newer solutions are able to replace the whole head.
3. **Model Training:** The facesets are then used to train an AI-model that is often based on an autoencoder network architecture. The autoencoder usually consists of one encoder and two decoders (one for each person), which takes an aligned face image of the person inside the source material as an input and outputs a newly synthesized face from the other person.
4. **Post-Processing:** The synthesized face image is then placed onto the target region. Optionally, several adjustments to the final frame are applied, such as color grading and blurring. In case of deepfake videos, each frame is modified separately and then turned into a video.

Perceptual Hashing

Hashes are used in cryptography to verify the integrity of data or messages. The hash-functions are built in such a way, that only the same input will result in the same output or that it is computational expensive to find another input resulting in the same hash. However, this type of hashes cannot be applied when com-

paring deepfaked images or video frames as compression, other post-processing operations and the manipulation itself will lead into the output of entirely different hashes. Therefore, robust hash-functions or perceptual hashing can be used instead to circumvent this disadvantage. Perceptual image hashing algorithms try to capture the semantics of an image as it is perceived by the human auditory system to estimate the similarity between two images rather than comparing their syntactic representations, such as byte-strings. Often multiple image-processing operations are applied on an input image to reduce its content to a low-dimensional data representation, which can later be used for comparison.

Related Work

This section will compare different state of the art deepfake detection methods for videos as well as robust hash methods.

Deep Fake Detection

Instead of evaluating whole videos many approaches base their classification on single frames.

To distinguish real images from copies generated by face-swapping, Nirkin et al. [2] compare the inner region of the face by the outer region. Guo et al. [3] argue, that CNN-based classifiers often do not learn to detect the artifacts or traces introduced by the manipulation but rather the image itself. To prevent this from happening, they take advantage of an additional network, which amplifies the manipulation traces and suppresses the image content.

Durall et al. [4] use the discrete Fourier transform to compute spatial frequencies and feed them into a classifier. Li et al. [5] present a multi-task learning approach, where one branch handles face patches extracted from the frame while the other branch compares the face region with the rest of the frame. Wang et al. [6] focus on improving on the generalizability of detectors. They studied what criteria need to be fulfilled to detect images generated by different CNNs while their model was only trained from data of one CNN.

There are also deepfake detection methods which consider the video as a whole. Some take the audio accompanying the video into account, while others ignore it. Mittal et al. [7] propose a multi-modal classifier. They split the input video into visual and audio tracks to compare how well both fit together.Agarwal et al. [8] also compare the visual and acoustic components of the videos. However, they concentrate on spoken phonemes that do not match the shape of the mouth.

Another approach to detect deepfakes using physiology patterns has been introduced by Li et al. [9] They use eye blinking patterns to determine whether a video is authentic or not.

Masi et al. [10] use deep learning to detect deep fakes. Their architecture consists of two branches, one enhancing frequency, one working with colors. The resulting feature maps get combined again and fed through a network that classifies the video.

Further details on several other approaches can be found in the exhaustive survey on deepfake detection schemes in [11].

Perceptual Video Hashing

In general, perceptual hashes are made to imitate how humans compare two pictures or videos. This means that structures well perceived by humans such as edges and approximate color should have more influence than noise, for example. There are many different algorithms, some of them are introduced here.

The main challenge for robust image hashing is feature extraction. One method is to compare the gray-value of one pixel with the gray-values of the eight surrounding pixels (Local Binary Pattern). Since this is sensitive to noise, many consider adding modifications to this method. Quin et al. use a variation called Webber Local Binary Pattern and add color features [12] to it. Davarzani et al. [13] use Center-Symmetric Local Binary Pattern to improve the robustness against additive noise.

Tang et al. [14] use the same color feature metric as Quin et al. [12] called color vector angle as it is insensitive to chroma adjustments while staying sensitive towards saturation and hue.

Friedrich et al. [15] propose a hash-function that concentrates on the relationship between edges and a secret key. Their method is therefore robust against brightness, color, and contrast adjustments as well as rotation and resizing, while being sensitive about small changes of the secret key.

Since videos include many frames, it takes a lot of computational power to calculate and compare every hash. To avoid this, there are video-based hashing methods. One simple method to create a video hash is to compare successive frames and only keep those with a specific difference to their predecessor. A problem with this approach is that the hashes may have different lengths, so they can not be compared with Euclidean distance. Instead the longest common sub strings are computed [16].

Own Previous Work

In the past, we have addressed deep fake detection as well as robust hashing independent from each other. In [17] we show an approach based on the Ghost effect for detecting video deep fakes. In [18] we discuss a detection method based on texture analysis known from image classification and segmentation. Our evaluation of pHash [19] led to our an optimized block hash ForBild [20] [21] with further mechanisms to increase robustness [22] [23] and privacy [24]. We also developed a video hash system based on motion vectors [25].

Proposed Approach

The overall concept is showcased in Figure 2. Unlike with a classic deepfake detection pipeline, where solely the test sample is considered during classification, we assume that a deepfake was created by modifying an already known video source. We also assume that when applying a deepfake detection algorithm to the original and the fake video, the latter will produce a higher likelihood of being classified as a fake. This improves the reliability of a deepfake detection as the decision about which video is fake is now based on a direct comparison of detection scores and not on a generic threshold. The classification pipeline is divided into the following steps: *Face Neutralization*, *Hashing*, *Video Retrieval*, *Deepfake Classification* and *Decision Making*.

Face Neutralization If large portions of the frame are being altered as part of the deepfake manipulation, the hash is likely to differ between authentic and forged frames. Thus, in order to find corresponding frames after the face in one of them has been changed, we have to neutralize the changes. For this, the faces inside the video frames need to be found first. After the facial

region has been determined, the area is neutralized by filling its content with a constant color.

Hashing The neutralized frames are then used for calculating a hash. Here, perceptual hashing algorithms are used, as they are less sensitive to compression artifacts or image post-processing. The computed robust hash is then stored within a database for later usage.

Video Retrieval During classification, hashes based on the frames of the video in question are calculated in order to retrieve any video, that has been processed prior and features the same background. The usage of robust hashing allows searching for exact matching hashes in the database, instead of calculating a hamming distance. Once matching videos have been identified, they are fed to the deepfake classifier. If no video was found during the search that featured the same background, an *UNKOWN* label will be assigned to the video under test and will be stored alongside its hash in the database.

Deepfake Classification For the deepfake classification, any pre-existing solution can be used. It is performed on the input video and the videos with matching backgrounds. In case of detection methods, that work on single frames, the results of multiple frames can be combined to make a more robust decision. If the detector has confident results on most of the frames, it does not take outliers into account. Otherwise it calculates the average prediction over all input frames. The confidence value of the deepfake classification is then used for the decision making.

Decision Making In the end, the input video and all found similar videos will be compared regarding their deepfake detection result. The lowest rated video is labeled as an *AUTHENTIC*. Every other video is labeled as *FAKE*. Additionally, to this decision the names of the videos, as well as their concrete prediction of the deepfake detector are printed out. This makes it possible to allow humans to manually review the original probably altered files. It also allows to check if the videos were mislabeled due to similar confidence scores. After the classification has concluded, the final label for the video under test is stored in the database for next iterations.

Evaluation

In this section, we present the evaluation results of the proposed approach. First, details about the setup used for the experiments are explained. Then the results are showcased and discussed.

Implementation

To perform the face neutralization on the video frames, the regions containing faces need to be estimated. In our implementation we take advantage of a Haar-cascade classifier provided by OpenCV [26]. A Haar-cascade uses Haar-filters to extract Haar-features. Haar-filters essentially check if specific parts are bright or dark in a frame section. To check the whole frame, this section is shifted over the frame. When one iteration is over the image gets resized so that different sizes of faces can be found [27]. The minimum size of faces we search for is 30*30 pixels. The facial areas are then painted gray (Figure 1). This way we receive very similar frames for an original frame and its altered versions. In some cases, other objects might also be detected as faces. However, false positives are negligible in contrast to false negatives, as the misclassifications should apply to both types of frames, authentic and forged ones, and leave enough background to distinguish the current background from completely different backgrounds of other videos.

Based on the processed frames, the robust hashes are computed. Our goal was to find a robust hash, which does not take too long to compute, and is easy to compare. Thus, we decided for a implementation of pHash provided by OpenCV [1]. For computing the hash, the input picture is first resized to 32*32 pixels and converted to a gray scale image. Then, a two dimensional discrete Cosine transform[28] is applied onto the image to retrieve its frequencies.Based on the 32*32 matrix we get back from this computation, we only use the first 8*8 coefficients, thereby we concentrate on the lowest frequencies. The arithmetic mean of the values in this sub-matrix is computed, each frequency value is replaced by 1 or 0. If the value is above the mean its replaced by 1 otherwise by 0. This produces a 64 bit hash. Hashes can be compared by using the hamming distance. The hamming distance counts how many bits are different in two hash strings.

For the deepfake detection itself, we decided on using the deepfake detector provided by Selim Seferbekov[2]. It won the Deepfake Detection Challenge in 2020[3]. The model is based on a frame-by-frame classification approach using EfficientNets [29]. During the experiments, we used the provided model checkpoints that were trained on the Deepfake Detection Challenge data set.

Data Sets

We looked for data sets which fit the following criteria: Firstly, for every modified video the data set has to contain the original video as well. It should be labeled which videos belong together. Secondly, the videos have to be modified by deep learning algorithms that change the face. Thirdly, we looked for one data set that also includes other modifications in the deepfake videos besides the face swap and one data set without such modifications. These modifications can be text overlays or small items added in the background. Data sets that feature with small background modifications are preferable as they are more close to manipulated videos that are shared on the internet. Based on the aforementioned criteria, we opted to use the *FaceForensics++* and *Deepfake Detection Challenge Data Set*.

FaceForensics++

FaceForensics++ is a data set released in 2019 by Rössler et al. [30]. It contains 1000 original videos with over 50,0000 frames. All videos were collected from YouTube and most show news programs. For each video, it was verified that the person looks almost straight into the camera. This was done to ensure that the face swapping methods produce high quality outputs.

The data set contains four different forgery methods: FaceSwap, Face2Face, NeuralTextures, and DeepFakes, a specific

[1] `https://github.com/opencv/opencv_contrib/blob/4.x/modules/img_hash/src/phash.cpp`

[2] `https://github.com/selimsef/dfdc_deepfake_challenge`

[3] `https://www.kaggle.com/competitions/deepfake-detection-challenge/leaderboard`

Figure 1. *Examples of frames with neutralized faces*

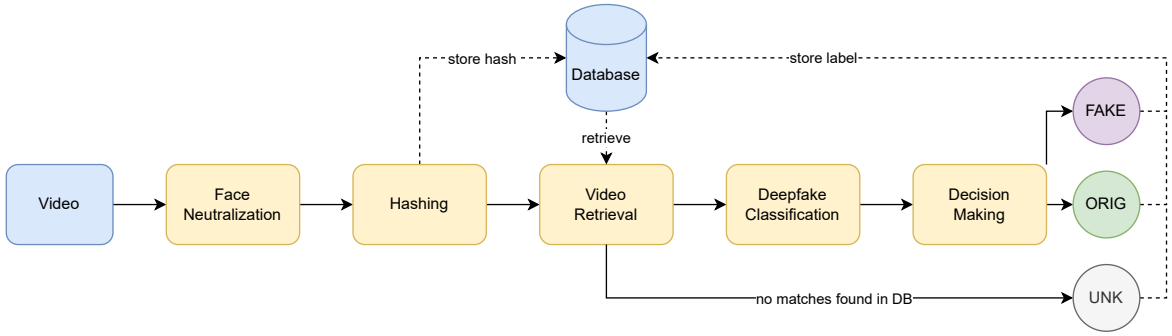

Figure 2. *Workflow of our approach*

deep learning face swapping implementation from GitHub[4]. Out of the four forgery types, only the videos created by the DeepFake method were used.

Deepfake Detection Challenge

The DFDC data set contains over 10,0000 modified videos with over 20,000 originals. It was released in 2020 by Dolhansky et al. [1]. While the videos in Faceforensics++ where collected from YouTube, the DFDC videos show approximately 3500 individuals, who agreed to the use of the videos and were paid to take part in the video scenes. All the fake videos were produced by deep learning face swapping methods.

Deepfake Autoencoder [31] uses one encoder and two decoders, one trained on either the target or source individual. For deepfake creation the footage of the input individual is processed by the encoder trained on the other individual. A second creation method uses a frame-based morphable-mask model[32]. Here facial landmarks are computed from both faces then morphed to fit. Also very important face regions, such as eyes and mouths are copied into the video. This method only works if source and target individual have very similar expressions.

The data set also includes three different methods based on Generative Adversarial Networks (GAN). The Neural Talking Head model [33] is able to learn the facial features of an individual from just a few shots. Face Swap Generative Adversarial Network which is able to do face swapping and reenactment. StyleGAN projects a given face descriptor on the source video.

Some of the videos were post-processed, for example, by sharpening the face. Additionally, distractors were added to 30% of the videos. Distractors can be texts that are overlayed, dots that are moved around from frame to frame, or social media filters that add a flower crown or dog features to the head. Approximately 70% of the videos were processed by color and geometric transformations, frame rate changes, etc. These additional modifications make the video matching very hard, but also represent a realistic scenario.

[4] `https://github.com/deepfakes/faceswap`

	fake	real	∑ actual
fake	985	15	1000
real	15	985	1000
∑ predicted	1000	1000	2000

Results of the proposed approach, FaceForensics++

	fake	real	∑ actual
fake	943	99	1000
real	57	985	1000
∑ predicted	1000	1000	2000

Results of the built-in deepfake detector, FaceForensics++

Results

For the FaceForensics++ data set (Table 1) an accuracy of 98.5% with both false positive and false negative rate at 1.5% was achieved. False positive and false negative rates were the same, as there is a 1:1 ratio between originals and fakes. Thus, for every original labeled incorrectly a fake will be labeled incorrectly as well. The built-in deepfake detector (Table 2) had an accuracy of 92.2% with a false positive rate of 9.9% and a false negative rate of 5.7%.

For the Deep Fake Detection Challenge data set our approach (Table 3 achieved an accuracy of about 88.5%. Suprisingly, this is significantly worse than the 98.8% accuracy achieved by the deepfake detector without the video matching (Table 4), since the deepfake detector model was trained on this data. One problem is, that there are several occasions where the deepfake detector

	fake	real	∑ actual
fake	99965	13625	113590
real	27	5529	5556
∑ predicted	99992	19154	119146

Results of the proposed approach, Deep Fake Detection Challenge

	fake	real	∑ actual
fake	98828	268	99096
real	1164	18886	20050
∑ predicted	99992	19154	119146

Results of the built-in deepfake detector, Deep Fake Detection Challenge

returns a confidence score of 0.5 when its built-in face detection method is unable to find a face in the frames:

- the face was partially out of the bounds of the frame
- the quality of the fake was not good, which led to blurred faces
- the head pose of the individual facing in another direction than straight into the camera
- an overall insufficient lightning or low illumination of the face
- people of color

Discussion

While the experiments showed good results on the unseen data of the FaceForensics++ data set, the approach performed less well on the larger DFDC data set. The FaceForensics++ data set shows that the concept is promising and but also indicates wherein some of the limitations to the proposed approach lie. The issues and potential solutions are discussed in this section.

The matching algorithm is unable to identify the original material On both data sets almost always the original is found as one of the similar videos. However, on the DFDC data set we find many similar videos, many originals showing almost the same scene, as well as many fakes of the same original. To find the source of specific deepfake, for instance of a video of a politician's speech, the search could be run against a database of certified videos. This would only help with fakes made from official videos. With more knowledge about the real world distribution, there would be an indication in which way the video matching algorithm should be enhanced. A hash based on the whole video instead of single frames could be better to distinguish between very similar settings, however it would be more vulnerable to manipulations via cutting.

No face can be found by the face detection algorithm Another problem we identified in the DFDC data set is that the face detector built into the deepfake detector was not always able to recognize the faces. This is a problem as the deepfake detector cannot rate a frame without a face. One reason for this are bad fakes. In this case, bad fake means that it does not show a structure similar to a face. While it would be helpful to filter these bad fakes, they are not as threatening as good fakes as they should be easily detectable for humans.

Use-Case Dependency The DFDC data set and the FaceForensics++ data set represent two different applications of deepfakes. On one hand, most of the videos in the DFDC data set are not shot by professionals. They show informal and private settings in contrast to the videos of the FaceForensics++ data set, which were mostly news shows collected from YouTube. They are professionally shot and lit and the faces present in those videos directly face the camera. As both data sets seem to cover different use-cases, the differences in the video content may also explain the dissimilar results achieved on those data sets. Yet, the detection of both types of deepfaked content may be relevant. Thus, a deepfake detector with really good results in one use case would also be desirable.

Good False Negative Rate On both data sets the proposed approach improves the false negative rate compared to the built-in deep fake detector. Almost every video rated as negative is an original.

Conclusion

This paper revolved around the research question, whether a deepfake detection pipeline can be enhanced by introducing background-matching. Although the conducted experiments during evaluation have revealed its potential, the approach needs to be further improved to bring practical benefits.

We combined an existing deepfake detector with a basic video matching algorithm that uses frames where faces have been replaced by gray rectangles. The overall aim was to compare the confidence scores provided by the detector between the video under test and other videos sharing the same background. The video with the lowest likelihood is classified as the original while the others are classified as forged.

This approach resulted in a higher accuracy compared to using the plain deepfake detector without video matching when classifying previously unseen data from the FaceForensics++ data set. Experiments on the Deepfake Detection Challenge data set showed worse results, but we identified a few reasons for this behavior and discussed possible strategies to tackle them.

There are many potential improvements of this approach. Both the video matching method or the deep fake detector could be exchanged. Instead of a frame-based hashing algorithm a deep learning hashing approach could be used. We see this rather as a first step towards an alternative to "blind" (using the term in the way of watermarking and steganalysis) deepfake detection.

Acknowledgment

This research work has been funded by BMBF and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

References

[1] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The DeepFake Detection Challenge Dataset.

[2] Yuval Nirkin, Lior Wolf, Yosi Keller, and Tal Hassner. DeepFake Detection Based on the Discrepancy Between the Face and its Context.

[3] Zhiqing Guo, Gaobo Yang, Jiyou Chen, and Xingming Sun. Fake face detection via adaptive manipulation traces extraction network.

[4] Ricard Durall, Margret Keuper, Franz-Josef Pfreundt, and Janis Keuper. Unmasking DeepFakes with simple Features.

[5] Xurong Li, Kun Yu, Shouling Ji, Yan Wang, Chunming Wu, and Hui Xue. Fighting Against Deepfake: Patch&Pair Convolutional Neural Networks (PPCNN). In *Companion Proceedings of the Web Conference 2020*, pages 88–89. ACM.

[6] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. CNN-Generated Images Are Surprisingly Easy to Spot... for Now. page 10.

[7] Trisha Mittal, Uttaran Bhattacharya, Rohan Chandra, Aniket Bera, and Dinesh Manocha. Emotions Don't Lie: An Audio-Visual Deepfake Detection Method Using Affective Cues.

[8] Shruti Agarwal, Hany Farid, Ohad Fried, and Maneesh Agrawala. Detecting Deep-Fake Videos from Phoneme-Viseme Mismatches. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2814–2822. IEEE.

[9] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking. page 8.

[10] Iacopo Masi, Aditya Killekar, Royston Marian Mascarenhas, Shenoy Pratik Gurudatt, and Wael AbdAlmageed. Two-branch Recurrent Network for Isolating Deepfakes in Videos.

[11] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection, 2020.

[12] Chuan Qin, Yecen Hu, Heng Yao, Xintao Duan, and Liping Gao. Perceptual Image Hashing Based on Weber Local Binary Pattern and Color Angle Representation. 7:45460–45471.

[13] Reza Davarzani, Saeed Mozaffari, and Khashayar Yaghmaie. Perceptual image hashing using center-symmetric local binary patterns. 75(8):4639–4667.

[14] Zhenjun Tang, Yumin Dai, Xianquan Zhang, Liyan Huang, and Fan Yang. Robust image hashing via colour vector angles and discrete wavelet transform. 8(3):142–149.

[15] J. Fridrich and M. Goljan. Robust hash functions for digital watermarking. In *Proceedings International Conference on Information Technology: Coding and Computing (Cat. No.PR00540)*, pages 178–183. IEEE Comput. Soc.

[16] Hany Farid. An Overview of Perceptual Hashing. 1(1).

[17] Raphael Antonius Frick, Sascha Zmudzinski, and Martin Steinebach. Detecting "deepfakes" in h. 264 video data using compression ghost artifacts. *Electronic Imaging*, 2020(4):116–1, 2020.

[18] Raphael Antonius Frick, Sascha Zmudzinski, and Martin Steinebach. Detecting deepfakes with haralick's texture properties. *Electronic Imaging*, 33:1–7, 2021.

[19] Christoph Zauner, Martin Steinebach, and Eckehard Hermann. Rihamark: perceptual image hash benchmarking. In *Media watermarking, security, and forensics III*, volume 7880, pages 343–357. SPIE, 2011.

[20] Martin Steinebach, Huajian Liu, and York Yannikos. Forbild: Efficient robust image hashing. In *Media Watermarking, Security, and Forensics 2012*, volume 8303, pages 195–202. SPIE, 2012.

[21] Martin Steinebach. Robust hashing for efficient forensic analysis of image sets. In *Digital Forensics and Cyber Crime. Third International ICST Conference, ICDF2C 2011, Dublin, Ireland, October 26-28, 2011, Revised Selected Papers*. Springer, 2012.

[22] Martin Steinebach, Huajian Liu, and York Yannikos. Efficient cropping-resistant robust image hashing. In *2014 Ninth International Conference on Availability, Reliability and Security*, pages 579–585. IEEE, 2014.

[23] Martin Steinebach, Tiberius Berwanger, and Huajian Liu. Towards image hashing robust against cropping and rotation. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–7, 2022.

[24] Uwe Breidenbach, Martin Steinebach, and Huajian Liu. Privacy-enhanced robust image hashing with bloom filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–10, 2020.

[25] Huajian Liu, Sebastian Fach, and Martin Steinebach. Motion vector based robust video hash. *Electronic Imaging*, 2020(4):218–1, 2020.

[26] Vadim Pisarevsky. Kipr/opencv.

[27] Paul Viola and Michael J Jones. Robust Real-Time Face Detection. page 18.

[28] Syed Ali Khayam. The Discrete Cosine Transform (DCT): Theory and Application.

[29] Artem A. Pokroy and Alexey D. Egorov. EfficientNets for DeepFake Detection: Comparison of Pretrained Models. In *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, pages 598–600. IEEE.

[30] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Niessner. FaceForensics++: Learning to Detect Manipulated Facial Images. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–11. IEEE.

[31] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou, and Weiming Zhang. DeepFaceLab: Integrated, flexible and extensible face-swapping framework.

[32] Dong Huang and Fernando De La Torre. Facial Action Transfer with Personalized Bilinear Regression. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, volume 7573 of *Lecture Notes in Computer Science*, pages 144–158. Springer Berlin Heidelberg.

[33] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-Shot Adversarial Learning of Realistic Neural Talking Head Models. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9458–9467. IEEE.

Author Biography

Martin Steinebach is the manager of the Media Security and IT-Forensics division at Fraunhofer SIT. In 2003, he received his PhD at the Technical University of Darmstadt for this work on digital audio watermarking. In 2016, he became honorary professor at the TU Darmstadt.

Stefanie Blümer is an undergraduate computer science student at the Technical University of Darmstadt.

Raphael Antonius Frick is a researcher at the Media Security and IT Forensics division at Fraunhofer SIT. His current research is dedicated to detecting artificially generated multimedia with special focus on detecting deepfakes.

Niklas Bunzel received his M.Sc. degree in computer science and IT security from Technical University Darmstadt in 2020, respectively. He is a PhD student at the TU-Darmstadt and a research scientist at Fraunhofer Institute for Secure Information Technology (SIT). His major research interests include artificial intelligence, IT security and steganography.