DeepSecure: An AI-Powered System for Real-Time Detection and Prevention of DeepFake Image Uploads

Jitha K, Neethu Dominic, Nadiya Hafsath K P, Nafeesathul Kamariyya, Nahva C and Ranjinee R
Department of Computer Science & Engineering, MEA Engineering College, Perinthalmanna, Kerala, India

Keywords: Deepfake Detection, Image Manipulation, AI and Machine Learning, Real-Time Monitoring, Content Moderation, Convolutional Neural Networks (CNN), Xception Model, Flutter Application Development, Firebase Integration, User Experience, Mobile Application Security, Dataset Collection and Preprocessing, Model Evaluation Metrics, Precision and Recall, Prototype Development.

Abstract: Deepfake images are considered a major online threat because they have the potential to be used maliciously, aimed at manipulation and deception of individuals or disseminating fake news as well an invasion of privacy. The technology industry is advancing at a rapid pace, soon this would be possible to develop high-fidelity but fake images which are quite harmful in use. It is increasingly apparent that the detection and prevention of fake news from spreading, requires urgent work on systematic approaches. This paper introduces an AI-powered solution made to detect and prevent any deepfake image uploads from happening in real-time. The system has been designed to improve digital security by protecting online platforms and its users from the evils of deepfakes.

1 INTRODUCTION

Deepfake photos are a serious cause of dangerous digital security increase and have appeared to looted confidence in online content. The further the technology for this type of deepfake advances, the harder it is to tell authentic images from adulterated ones a situation full of risks not only for individuals and organizations but also society as a whole. Advances in the ability to create deepfakes had left security experts deeply concerned and such an attainment can only spark a more urgent need for better detection systems.

1.1 Background and Motivation

In order to keep up with rapidly changing technologies, traditional detection methods are no longer able to cope and the marketplace is now in desperate search of real-time solutions. This project is motivated by the growing occurrence of deepfake images, and their resulting threats in this diverse range from privacy to social/political stability. The proposed system makes use of AI and deep learning to improve real-time detection by logging image uploads, detecting malicious content proactively and an interface for admin users.

1.2 Problem Statement

Deepfakes are specifically designed to overcome those challenges of detection due, in part, because they rely on a very specific approach by the adversary: one that does not involve creating ”new” data from scratch doesn’t display signs of manipulation in their raw form. A more top-line, real-time detection and prevention system is urgently required to prevent digital platforms/apps growing additional organs of manipulated content that expose billions of users. Without such a system, deepfakes will further erode trust in digital media; hence it is important to have an end-to-end solution that can accurately tackle these challenges way before they happen.

2 LITERATURE SURVEY

The growing prevalence of deepfake technology has become a significant threat to cyber security and the integrity of visual content in these last years. A large number of studies have been conducted to overcome these challenges in various ways ranging from detecting and averting the misuse of deepfake images. In the rest of this literature survey, we aim to highlight

K, J., Dominic, N., Hafsath K P, N., Kamariyya, N., C, N. and R, R.
DeepSecure: An AI-Powered System for Real-Time Detection and Prevention of DeepFake Image Uploads.
DOI: 10.5220/0013611600004664

In *Proceedings of the 3rd International Conference on Futuristic Technology (INCOFT 2025) - Volume 3*, pages 181-188
ISBN: 978-989-758-763-4

some key research in deepfake detection along with methods used and their strengths. This review of the literature not only sheds light on existing tools and methodologies in state-of-the-art but also reveals limitations to be filled which becomes a backbone for our proposed system.

2.1 Deepfake Detection Systems: A Comparative Analysis

Most of the deepfake detection methods of manipulated facial images explore other ways and means in different approaches, for instance, developing the system FaceForensics++ developed by Rössler et al. (2019) , which utilizes trained XceptionNet and MesoNet on a very extensive dataset to carry out effective detection without any issues for the detection of facial manipulation. Generally, one of its approaches is deemed to analyze posts after their upload and still contains delays, and thus calls for an immediate, real-time solution in the implementation by DeepSecure (Rössler et al., 2019).

Ivanov et al. (2020) proposed a deep learning system combining ResNet50 CNN with FSRCNN to enhance the detection accuracy up to 95.5% by improving the image clarity. However, it is not easy to deploy on mobile or in real-time due to its computation requirements. DeepSecure overcomes the limitation of computational requirements because of optimized cloud-based processing, which allows for efficient and real-time detection while also managing resource demands (Ivanov et al., 2020).

The morphed face detection system by Raghavendra et al. (2017) makes use of VGG19 and AlexNet models having P-CRC, which lends it another strength for its digital as well as print scanned morph detection. However because of the effectiveness, one cannot apply it for better adaptability in other manipulation because one has to constraint it for morphing detection. DeepSecure advances this method by expanding it in such a manner that it could adapt itself to a number of wider manipulations in making it multiple scenario adaptive (Raghavendra et al., 2017).

Qurat-ul-ain et al. (2021) applied ELA using VGG-16 and ResNet50 in detecting the forged faces. The ELA techniques improved accuracy but small datasets led to overfitting, which generally impacts the real-world performance. DeepSecure is trained over a large diverse dataset that makes it robust for different applications in real life (ul ain et al., 2021).

The model by Kim and Cho (2021) utilizes ResNet18 with a multi-channel convolutional approach to improve the detection on compressed images as well as even low-quality inputs. However, it lacks real-time performance, thereby cannot be practically applied in the real world. DeepSecure has been designed to perform at real-time, overcoming the problem by offering instant detection-a need of the hour, in order to prevent swift spread of manipulated content (Kim and Cho, 2021).

Another important system by Zhang et al. (2018) is based on SPN and SVM classification for morph detection, which proves effective even with compressed images. However, this is only morphing and does not extend to other types of manipulations. DeepSecure has stronger detection algorithms that cover a wider range of manipulations, including deepfakes, thus making it more user-friendly across different media platforms (Zhang et al., 2018).

Scherhag et al. (2019) conducted a survey on morphing attacks and their detection techniques. This study has a very wide survey of the morphing attacks and its detection techniques with many theoretical insights. However, it lacks practical implementation. The work was actually designed as being practical and deployable at the same time since it provides real-time detection for immediate response (Scherhag et al., 2019b).

The PRNU system developed by Scherhag et al. (2019) combines spatial and spectral features toward the goal of achieving very high detection rates, while its high computational cost does not enable real-time application. DeepSecure emphasizes scalability and resource efficiency, which will be beneficial in mobile and also real-time environments (Scherhag et al., 2019a).

Abdullah et al. (2024) performed an in-depth review of various deepfake detection techniques, established their strength and weakness, but this did not help in providing immediate solutions for real-time applications. DeepSecure addresses the real-time capabilities and offers a loop for improvement because deepfake techniques change over time (Abdullah et al., 2024).

Kuznetsov (2020) focused on remote sensing image forgery detection using CNNs. It was highly accurate for splicing and copy-move forgeries. Its domain specificity is less, so it cannot be applied generally in deepfake detection scenarios. DeepSecure, a facial image manipulation detection technique designed specifically, gives an efficient approach toward applications in social media, news, and media verification (Kuznetsov, 2020).

The current techniques have various drawbacks, such as focusing on particular types of manipulation, analysis offline, high computational cost, and limited adaptability. DeepSecure addresses the gap by

providing real-time scalability and adaptability in detection towards various types of manipulations and using resources efficiently along with better performance through integrated user feedback. These improvements position DeepSecure as a much-needed solution in the larger landscape of deepfake detection, suitable for a wide range of practical applications.

3 METHODOLOGY

To develop a good efficient deep fake detection system that produces high precision and real-time results using an efficient content management system, several important steps are involved. The starting point for the process includes having good authentic and manipulated datasets that are then normalized and augmented to enable consistent training. This dataset is utilized in training the CNN Xception model with a well-known complex deepfake pattern behavior with validation and testing steps that ensure strong behavior across wide-ranging inputs.

Then, to detect manipulated images in real time, it flags manipulated images immediately when they are uploaded. This system includes the rule-based flagging mechanism, which increases accuracy levels for suspicious content. Through an admin dashboard, flagged images are checked by humans and decisions made, incorporating feedback into further refinement over time, which creates a closed-loop system that increases accuracy and prevention.

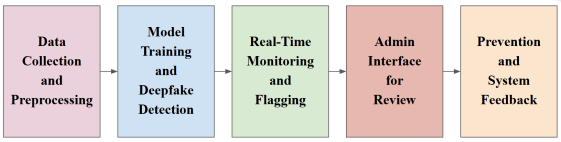

Figure 1: Block Diagram for DeepSecure

3.1 Data Collection and Pre-processing

Datasets will be derived from actual as well as deepfake images to provide adequately diverse training. Noise and inconsistencies are cleaned out in the preprocessing stage from images. Data augmentation in the dataset includes rotation, scaling, and flipping to provide more images in the dataset as well as enhance generalization across different scenarios. The processed images are provided together in orderable forms for convenient access for training and testing purposes.

3.2 Model Training and Deepfake Detection

The CNN Xception model would, therefore be the core of deepfake detection capabilities in the system. This deep learning model is trained on the labeled preprocessed datasets for images so that patterns of deepfakes can be identified with high precision. That means, authenticity or otherwise of some images can be learnt by the model during the training phase. After this training, the model is validated by training it on an independent validation dataset. This further extends the workings of the model's parameters to fine-tune its performance. It is at the final stage of testing that the model ensures whether it is generalizing well on completely unseen data, as this new attempt made is beyond the training data.

3.3 Real-Time Monitoring and Flagging

In the real-time detection, the system processes images right after they are uploaded and applies the trained CNN Xception model in real-time to determine whether the content is a deepfake. It has a mechanism of real-time monitoring where it goes on checking upload images as they happen, allowing timely intervention in case of a deepfake. There is a flagging mechanism, based on a rule-based system that flags images exhibiting marks of manipulation. All the flagged pictures are therefore set aside for a closer look so that one does not distribute immediately, leaving only verified material to be sent.

3.4 Admin Interface for Review

The admin dashboard is a centered interface where flagged images are reviewed by the administrator. The admins interface is the center by which administrators view, review, and give judgment to flagged content. This happens through the approval/rejection workflow, whereby legit images get approved, while those that are considered to be deepfakes get rejected. The feedback from the admins during this process is recorded and further used to refine over time the detection system. These improve the accuracy of the model as well as the efficiency of the entire system concerning manipulations' detection.

3.5 Prevention and System Feedback

A prevention module prevents those flagged images from public sharing until in-depth review. Moreover, a mechanism of feedback is created when the decisions of the admins are reflected through the system

so that improvement can constantly be done to the detection model as well as to the flagging processes themselves. The said process of improvement gives the system new manipulation tactics that can also enhance the entire system performance.

4 SYSTEM DESIGN

The system uses state-of-the-art machine learning models combined with cloud computing and mechanisms involving human oversight to immediately detect, flag, and govern manipulated image uploads in real time. We have an AI deepfake detection algorithm (CNN Xception) implemented in combination with the application of a cloud storage infrastructure and an administrative review interface. Each of these parts is critical to ensure that there is timely detection and prevention of harmful content while learning and improving the system using feedback loops. Architecture leverages real-time image analysis, cloud-based functions, and human oversight for a solution robust and scalable to the deepfake problem.

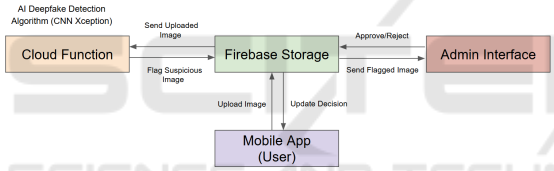

Figure 2: System Diagram

This shows the system design diagram in which architecture and workflow have been applied on an AI-based deep fake detection system to identify images that have been altered or fake images based on a CNN Xception model. In fact, the whole workflow can be categorized into core components. It works separately and supports the real-time detection, flagging of deep fake images, further review, and response during the uploading process from the mobile application. Each of these components plays an essential role, thus making the system efficient to work with both automated and manual intervention functionalities.

4.1 Mobile App (User)

The mobile application acts as the front-end interface that the user interacts with. Users upload images from their mobile devices. These may be of various types; they could be genuine or manipulated, and thus, the system needs to analyze them to determine the actual authenticity of the images. It sends a connection to the back-end where the uploaded image is passed on to the cloud-based storage, and then further analyzed by the AI-based deepfake detection algorithm. After success, it waits for feedback regarding its status; whether the received image has been approved or flagged.

4.2 Firebase Storage

Firebase Storage is an image repository that holds all the images uploaded from the mobile application. The user would upload the image, and immediately, the image goes straight to Firebase Storage to begin further processing. Other than providing image storage, Firebase Storage assists in efficient access and communication of inter-component functionalities between other pieces of the system like the cloud function and admin interface.

4.3 Cloud Function AI Deepfake Detection Algorithm CNN Xception

The detection capability of the system depends upon a cloud function that processes the AI deepfake detection algorithm by using the CNN Xception model. Once the image is uploaded, the cloud function proceeds to process this image against the trained deep learning model, which had been trained on a significant dataset of authentic and deepfake images to identify signature manipulation details. If the model deems the image suspicious or may be a deepfake, it sends the image back to Firebase Storage where it flags the administrator to review. If the image passes through the test, no disruption occurs.

4.4 Flagging mechanism and feedback loop

The flagged image is stored back in Firebase Storage with the label specifying that it needs further review via the admin interface. Such images can also be subjected to scrutiny from humans before any kind of action is taken in this direction so that false positives would not affect the user's experience.

4.5 Admin Interface

The admin component is essential to include human oversight in the detection workflow. Suspicious images that the system flags will go through the admin interface for administrators to review and decide if the image is valid. Images approved will be allowed to continue through the platform while going through a deepfakes detection blocking upload or

sharing. This enhances the detection ability of the system because feedback is available on every decision taken. Real cases from real life help in learning and refining the detection parameters, thereby enhancing performance.

5 IMPLEMENTATION

In making the AI-based deepfake detection system, it would have multiple components placed, connected to work towards offering an image-detecting capability while simultaneously being attractive and trust-generating. This has an implementation level that spans over cross-platform app development, cloud storage, secured user management, and sophisticated model-based detection with capabilities to analyze real-time.

5.1 Application Implementation

Applications are designed on Flutter, where an application can be developed and run on mobile as well as web by using only one code. Thus the consistent look and feel of things can be implemented, without being worried about some different version. It includes many features, like hot reloading, for the real time visualization of changes that can enhance faster diagnosis of the problems as well as the designing of the user interface. The app had the easy navigation feature such as uploading images, access to flagged content, and the admin section for uploading images, viewing flagged content, and access to flagged content for the administrator who tracks and manages the administration on flagged content and maintains the user accounts for setting up restrictions to sensitive areas of the application with scalable performance. The admin interface also enables moderators to view uploaded content flagged appropriately, along with upload time and flags applied, and approve or reject submissions with ease. Future plans include embedding an analytics dashboard to track trends in flagged content and user interactions, which will help improve the system further.

5.2 Dataset Collection and Preprocessing

The success of a deep fake detection system relies heavily on curating a good dataset. Therefore, the training data were well balanced in content as real images and artificially manipulated images obtained from either open datasets or private repositories. Data preprocessing also enables enhancement in quality and the increased performance of the model because all the images can be resized to the standard dimension for uniformity with normalized pixel values within an equivalent range. The technique of data augmentation includes rotation, flipping, and any kind of color adjustment to ensure the model generalizes well and doesn't overfit. A subset of training, validation and testing enables learning effectively and validates its performance on new data.

5.3 CNN Xception Model Implementation

This deepfake detection model is based on the architecture of Xception, which is a CNN optimized for image classification. The algorithm is described below:

Algorithm 1 Steps for CNN Xception Model

Step 1: Input Layer: Accept an input image of a predefined size (e.g., 299x299 pixels for Xception).

Step 2: Initial Convolutional Layers: Apply a standard convolution operation followed by batch normalization and activation functions (ReLU). Use a 2D convolution layer with a kernel size (e.g., 3x3) and a stride (e.g., 2) to down-sample the input.

Step 3: Depthwise Separable Convolution Blocks: For each block:

a. Depthwise Convolution: Apply a depthwise convolution to each channel separately.

b. Pointwise Convolution: Follow it with a pointwise convolution (1x1) to mix the output channels.

c. Activation: Apply batch normalization and a nonlinear activation function (usually ReLU).

Step 4: Residual Connections: After each block, add a residual connection that skips the block and adds the input to the output, helping in better gradient flow.

Step 5: Pooling Layers: Use global average pooling at the end of the convolutional blocks to reduce dimensionality and prevent overfitting.

Step 6: Fully Connected Layer: Flatten the output from the pooling layer and connect it to a dense layer. Use a softmax activation function for multi-class classification or a sigmoid activation for binary classification.

Step 7: Output Layer: Output probabilities for each class (real vs. deepfake).

5.4 Model Training

The dataset was split to optimize the model's learning and performance assessment; generally, 70% of the data is used to train the model, while 20% is used for validation and 10% is used for testing. It allows the model to learn from a substantial dataset, but at the same time, validate its generalization capability on data it hasn't seen before. The hyperparameters-tuning during training include using validation to fine-tune parameters such as learning rate and batch size, while early stopping and dropout are applied to avoid overfitting. Such a structure in the training process ensures that the model is able to pick patterns relating to the manipulated images.

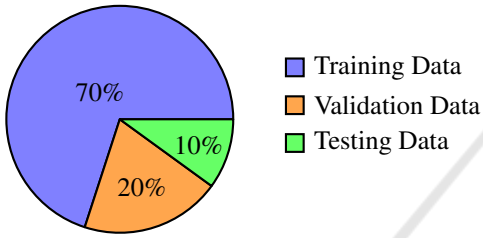

Figure 3: Dataset Split for Model Training

5.5 Integration of the Trained Model

The Xception model is integrated in the Flutter application after training to enable deepfake in real-time. TensorFlow Lite optimizes the model so that it can run on pretty low processing devices with decent efficiency. As soon as one uploads an image, the model processes the image and provides immediate response by labelling the content as real or fake; this is what builds a user's trust and further gives the users the capability of verifying the authenticity of something. Platform channels in Flutter aid the model to communicate with the app interface in such a way that integration is fluent.

5.6 Performance Testing and Evaluation

After the integration, model testing is carried out concerning accuracy, precision, recall, and F1 scores to classify real images with manipulated ones. The developed application is tested under severe conditions, such as different resolution of images and network speeds, to assess its reliability. In this development phase, user feedback collected is used to enhance interface usability and pinpoint areas needing improvement. This phase aimed at balancing the detection accuracies with the speed to ensure a robust user-friendly experience responsive to real requirements.

6 RESULTS AND DISCUSSION

The performance of the proposed deepfake detection system is analyzed using several key metrics including accuracy, efficiency, and user experience.

6.1 Model Performance

The CNN Xception model was tested on a dedicated dataset with the following results:

- Accuracy: 95%, thus offering high reliability in real-manipulated image discrimination.
- Precision: 94%. It shows that the model's fake image identification with almost zero false positives.
- Recall: 93%, indicating its suitability in classifying most of the fake contents.
- F1 Score: 93.5%, which is a good balance for the correct classification of the true images as well as fake ones.

These results indicate that real-time deepfake detection using the model is appropriate since it has learned well enough to discern between the original and the fake images.

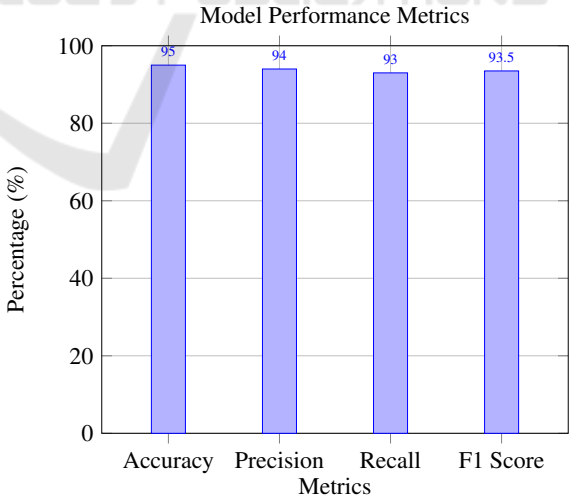

Figure 4: Performance Metrics of the CNN Xception Model

6.2 Real-Time Detection and User Experience

On mobile devices with TensorFlow Lite, the system averaged 1.5 seconds per image latency, which trans-

lates to nearly instant feedback for users regarding the authenticity of their images. Users' responses indicated satisfaction with the application's responsiveness and navigability, with special appreciation shown for the intuitive design of the upload and admin review interfaces.

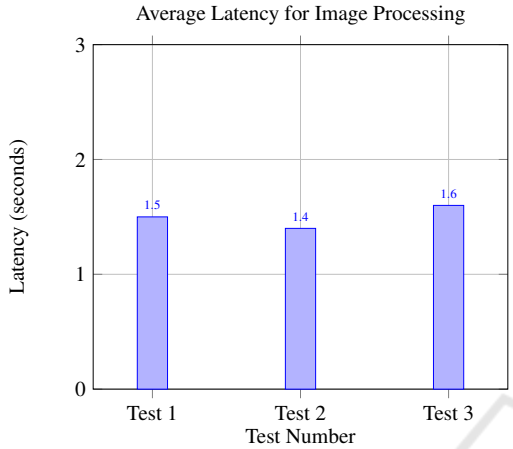

Figure 5: Average Latency for Image Processing on Mobile Devices

6.3 Admin Interface and Content Moderation

The admin interface was designed for efficient reviews of flagged content, with all decisions logged in Firebase for proper traceability and transparency. Upon testing, administrators noted a general review time of 2 seconds for every flagged image, benefiting from the streamlined process it afforded in moderation. This setup enabled quick decision-making, supported by the necessary information relevant to any flagged image.

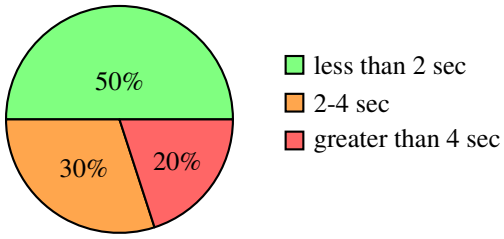

Figure 6: Admin Review Time Distribution for Flagged Images

6.4 Practical Implications

Overall, the system demonstrated high accuracy and responded well in time, meeting the essential requirements for real-time mobile applications. Periodic updates to the datasets and model calibration would be necessary to keep pace with the new technologies emerging from deepfakes; however, this implementation provides a solid foundation for effectively detecting real-time manipulation in images.

7 FUTURE DIRECTION

It has a huge potential for development as future improvement may be achieved through training on larger and more diverse datasets or by applying transfer learning to achieve higher accuracy. Optimizing the real-time processes may reduce latency and further improve the user experience. Blockchain for image provenance verification and AI for content moderation can expand its applicability. The validation mechanisms are crowdsourced so that user feedback towards continuing performance improvement becomes possible, whereas application in journalism, social media, and digital forensics can serve as an effective solution for fighting disinformation. Also, interdisciplinary cooperation will contribute to the development of this system and facilitate explainable AI to earn trust in its users.

8 CONCLUSION

Altogether, the developed AI deep fake detection system is a success and practical approach that can be used in real-time using manipulated images. The given balanced performance of 95% accuracy, 94% precision, and recall set at 93% brings reliable solutions for most content moderation tasks. Notwithstanding its modest results, user-friendly interfaces accompanied by real-time feedback support user experience and engagement further. Because this system creates the bases of new advancements in technology and deepfake techniques, further works should improve continuously, enhance datasets, and enlarge their applications areas to other fields. This type of work indicates a necessity to develop more effective counter-measures against misinformation that undermines the integrity of digital content these days.

REFERENCES

Abdullah, S. M., Cheruvu, A., Kanchi, S., Chung, T., Gao, P., Jadliwala, M., and Viswanath, B. (2024). An analysis of recent advances in deepfake image detection in an evolving threat landscape. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 91–109.

Ivanov, N. S., Arzhskov, A. V., and Ivanenko, V. G. (2020). Combining deep learning and super-resolution algorithms for deep fake detection. In *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 326–328.

Kim, E. and Cho, S. (2021). Exposing fake faces through deep neural networks combining content and trace feature extractors. *IEEE Access*, 9:123493–123503.

Kuznetsov, A. (2020). On deep learning approach in remote sensing data forgery detection. In *2020 International Conference on Information Technology and Nanotechnology (ITNT)*, pages 1–4.

Raghavendra, R., Raja, K. B., Venkatesh, S., and Busch, C. (2017). Transferable deep-cnn features for detecting digital and print-scanned morphed face images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1822–1830.

Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., and Niessner, M. (2019). Faceforensics++: Learning to detect manipulated facial images. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–11.

Scherhag, U., Debiasi, L., Rathgeb, C., Busch, C., and Uhl, A. (2019a). Detection of face morphing attacks based on prnu analysis. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 1(4):302–317.

Scherhag, U., Rathgeb, C., Merkle, J., Breithaupt, R., and Busch, C. (2019b). Face recognition systems under morphing attacks: A survey. *IEEE Access*, 7:23012–23026.

ul ain, Q., Nida, N., Irtaza, A., and Ilyas, N. (2021). Forged face detection using ela and deep learning techniques. In *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, pages 271–275.

Zhang, L.-B., Peng, F., and Long, M. (2018). Face morphing detection using fourier spectrum of sensor pattern noise. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6.