EVOLUTION IN ELECTRICAL AND ELECTRONIC ENGINEERING
e-ISSN: 2756-8458

EEEE

Vol. 5 No. 2 (2024) 324-331
https://publisher.uthm.edu.my/periodicals/index.php/eeee

Development of a Mobile Application for Deepfake Detection using the Xception Model

Chong Loo Jia[1], Nan Mad Sahar[1]*

[1] *Faculty of Electrical and Electronic Engineering,*
Universiti Tun Hussein Onn Malaysia, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: nan@uthm.edu.my
DOI: https://doi.org/10.30880/eeee.2024.05.02.039

Article Info

Received: 7 July 2024
Accepted: 29 October 2024
Available online: 30 October 2024

Keywords

Deepfake detection, Xception, mobile app.

Abstract

Deepfake technologies are built with sophisticated AI and machine learning techniques such as Generative Adversarial Networks (GANs) that can generate highly realistic videos. Although deepfake technology is employed in various legitimate sectors like entertainment and education, they also pose a significant risk of being used for deceptive purposes, contributing to the spread of misinformation. To address this issue, a mobile application for detecting deepfake videos was developed. The study evaluated the performance of two deep learning models, Xception and EfficientNet-B7 on the DFDC dataset. The models were trained using Google Colab and were tested for accuracy in detecting deepfake content. The results showed that the Xception model outperformed the EfficientNet-B7 model, achieving an accuracy of 89.28% compared to 75.41% for EfficientNet-B7. Based on these findings, the Xception model was chosen to be implemented in the mobile application developed using MIT App Inventor and Flask API. The mobile app demonstrated 75% accuracy and a detection speed of 3.4891 seconds, representing a step forward in providing a robust and efficient deepfake detection solution for mobile platforms.

1. Introduction

Deepfakes refer to synthetic media where an individual's likeness is replaced in an existing image or video. These manipulations are often created using advanced deep learning techniques, such as autoencoders and generative adversarial networks (GANs), which analyze a person's facial expressions and movements to generate highly realistic depictions of another individual [1]. Examples of some prominent deepfake generation tools include CycleGAN [2], FSGAN [3], and StarGAN [4]. In addition to these tools, there are various methods used to create deepfakes, including face swapping, voice swapping, text-to-speech, and lip-syncing, require robust detection algorithms to differentiate authentic content from manipulated content [5].

While deepfakes have potential applications in fields like entertainment and education, they also pose significant risks when used maliciously to spread misinformation or harm reputations. For instance, the Mayor of London, Sadiq Khan, reported that a fake AI-generated audio clip of him almost caused a serious public disorder, as discussed by Spring et al. [6]. This incident highlights the urgent need for effective deepfake detection methods.

To address this problem, there has been increased research into deepfake detection methods. A recent review by Passos et al. [7] summarized 89 studies proposing various deep learning-based approaches for detecting deepfake content, including convolutional neural networks (CNNs), generative models (GANs), recurrent neural networks (RNNs), and transformer-based architectures. For example, Masood et al. [8] evaluated 10 pre-trained CNN models for deepfake detection, combining them with a Support Vector Machine

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 324-331

(SVM) classifier. Given the strong performance of Convolutional Neural Networks (CNNs) in deepfake detection, this study focuses on identifying deepfake manipulation using CNNs.

The widespread dissemination of deepfakes on the internet and social media has made the development of reliable and accessible detection solutions a pressing priority, especially for mobile platforms where large amounts of digital content are consumed [9]. One notable example is the Korean mobile app KaiCatch, which can identify deepfakes with around 90% accuracy [10]. The development of such mobile-based deepfake detection solutions represents a crucial step in reducing the potential harm caused by the misuse of this technology.

2. Methodology

In this section, the methodology employed in the development of the deepfake detection mobile application is described. The process encompasses training and testing the model using Google Colaboratory, creating a user-friendly mobile application through Flask API and MIT App Inventor, and evaluating the performance of the mobile app. Fig. 1 displays the training and testing process.

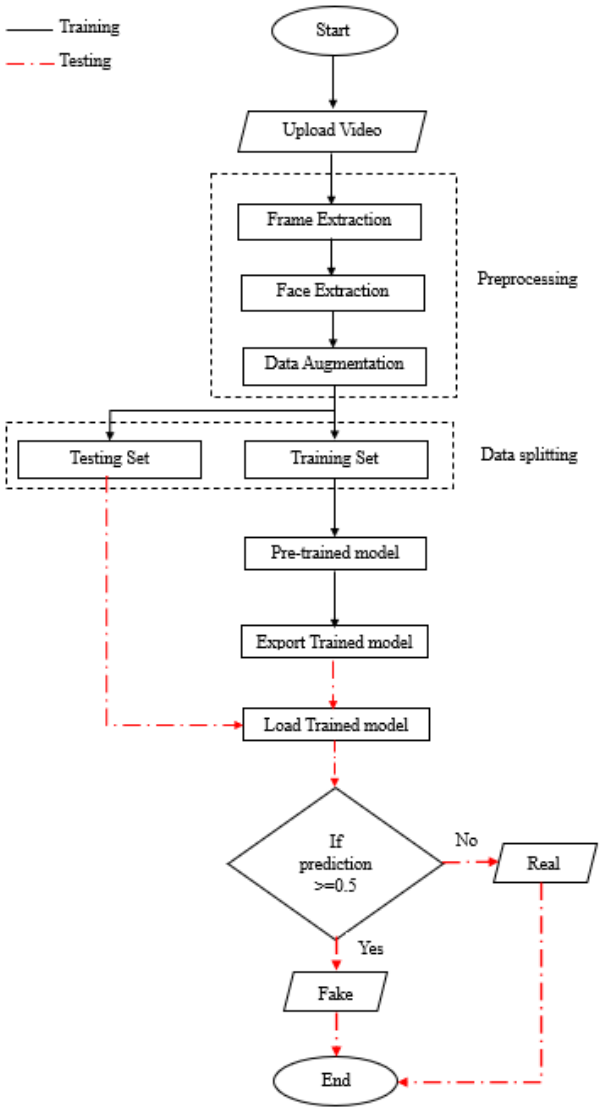


Fig. 1 *Flowchart of the training and testing model*

2.1 Training and Testing model

Referring to Fig. 1, the flowchart shows the training and testing processes of the models. The black arrows indicate the training phase, whereas the red dotted arrows illustrate the testing phase. The models that undergo these processes are Xception and EfficientNet-B7, which are evaluated for their performance. The code repositories for these models are sourced from GitHub repository [11].

2.1.1 Dataset Collection

The dataset utilized to optimize the model for prediction is from Kaggle's DeepFake Detection Challenge (DFDC). A total of 400 videos are used in this project, comprising 200 normal videos and 200 deepfake videos. The DFDC dataset is chosen because it simulates realistic scenarios by incorporating a diverse range of subjects, environments, lighting conditions, and face-swapping techniques. These elements reflect real-world variations and introduce challenges that can greatly influence the model's ability to detect deepfakes. The videos will be uploaded to Google Drive for the training process.

2.1.2 Data preprocessing

Referring to Fig. 1, data preprocessing will be divided into three steps which are frame extraction, face detection and data augmentation. First, the frame extraction process divides the videos into frames, with only a subset of the frames randomly selected to optimize. Next, faces are detected and isolated from the frames using the MTCNN model, then categorized into 'deepfake' and 'normal' folders. Finally, data augmentation techniques are applied to the images, generating new samples to increase dataset diversity. Fig. 2 displays an example of video preprocessing.

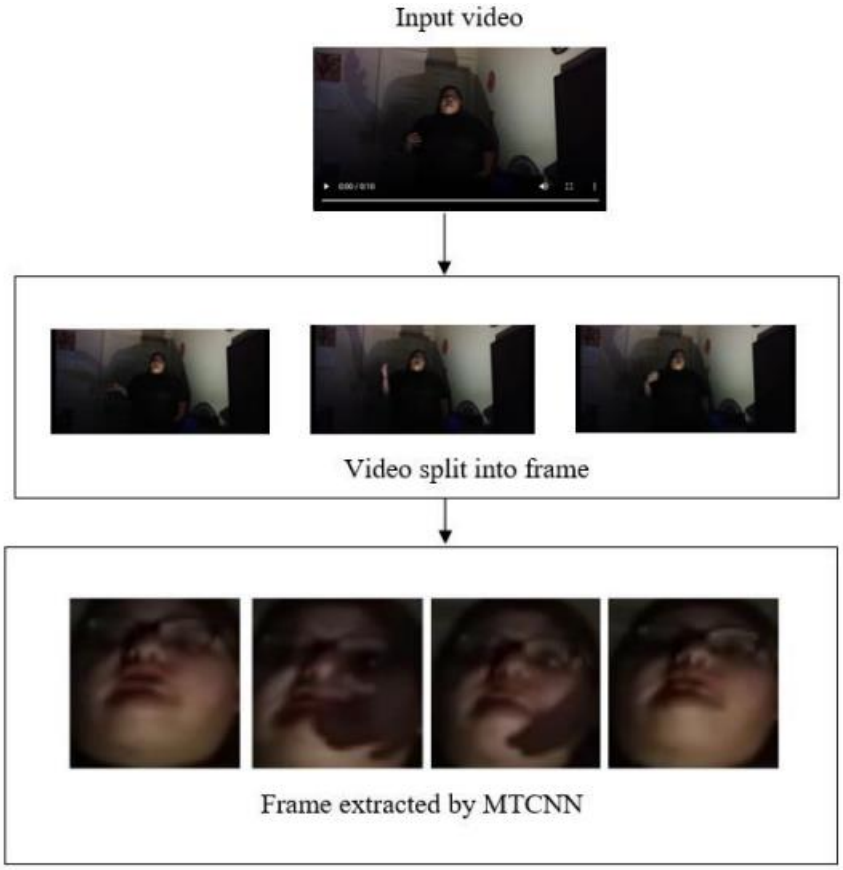


Fig. 2 *Preprocessing of video*

2.1.3 Training model using Google Colaboratory

The deepfake detection model uses a pre-trained model from Keras, which was initially trained on the large-scale ImageNet dataset. By leveraging this pre-trained model, the training process becomes significantly more efficient, requiring fewer epochs to achieve optimal weights compared to training from scratch. The training is conducted on Google Colaboratory, a cloud-based platform that provides access to powerful GPU resources, thereby accelerating the training process and improving overall computing performance.

2.1.4 Video prediction

After training, the model is tested on a separate dataset to ensure an unbiased evaluation of its performance. The trained model analyzes each frame, leveraging the patterns and features learned during training to make predictions. The frame-level predictions are then combined to determine an overall assessment. If the average prediction value across all frames is greater than or equal to 0.5, the video is classified as a deepfake; otherwise, it is considered real.

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 324-331

2.2 Mobile app development

Fig. 3 displays the process of developing the Android-based mobile app. This section provides a detailed description of the app's development using MIT App Inventor and Flask API.

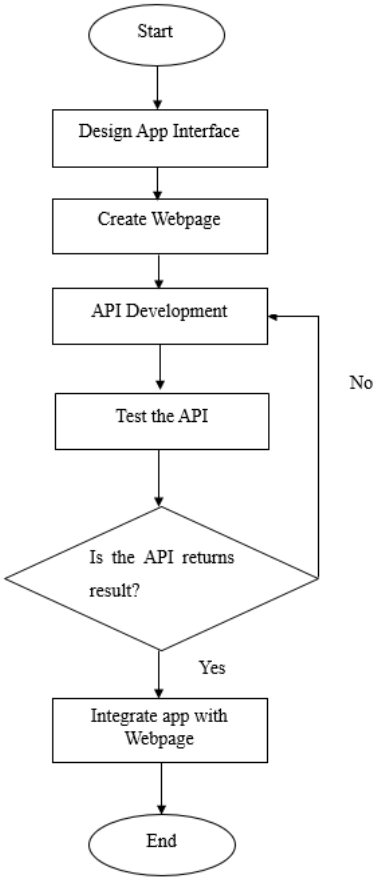


Fig. 3 *Flowchart of app development*

The project utilizes MIT App Inventor to design an application that allows users to upload videos for deepfake detection. The application's homepage features a start button that navigates users to the video upload page. To create a page that allows users to upload a video to an API, an HTML form needs to be set up. This form should have an enctype attribute set to "multipart/form-data" to enable file uploads. The method of the form will be set to "POST" to send data to the server, and the action attribute should point to the PHP script that will handle the file upload. The PHP script serves as an intermediary between the front-end and the Flask-based REST API backend, facilitating the transfer of the uploaded video for further processing and prediction in API.

The backend is developed using a Python-based REST API built with the Flask framework. The API provides one endpoint with a POST method to receive the uploaded video and a GET method to return the prediction result. The API processes the video by extracting frames, detecting faces using MTCNN, resizing the faces, and inputting them into the trained deepfake detection model. After the API completes the prediction, the result will be returned to the user. Finally, the application seamlessly integrates the upload webpage with the MIT App Inventor interface using the WebViewer component, enabling a smooth user experience.

2.3 Evaluation the performance of the mobile app

To assess the effectiveness of the deepfake detection app, both accuracy and speed metrics will be assessed. For accuracy, the app will be tested on 20 videos from the DFDC dataset with known ground truth labels. Each video will be uploaded three times, and the app's predictions will be compared to the true labels using a confusion matrix. This matrix provides a machine learning model's prediction, distinguishing between True Positives (TP), which represent the number of real videos accurately identified; True Negatives (TN), indicating the number of deepfake videos correctly identified; False Positives (FP), referring to the number of deepfake videos misclassified as real; and False Negatives (FN), which represent the number of real video incorrectly identified as deepfake. The equation to calculate the overall accuracy is stated as (1).

(1)

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

To measure the speed, the time taken for video upload, preprocessing, prediction, and result display will be recorded for 10 videos. This process will be repeated five times for each video, and the average detection speed will be computed. The overall execution duration will be calculated as the difference between the timestamps recorded before the upload and after the result display. This will provide insights into the app's performance in CPU environments.

3. Result and Discussion

3.1 Model accuracy and discussion

This analysis concentrates on evaluating the performance of the Xception and EfficientNet-B7 models on the DFDC dataset. Fig. 4 illustrates the graphs of training and testing accuracy of the Xception model. It shows that the Xception model has achieved 89.28% training accuracy with 80.91% testing accuracy across 10 epochs. This underscoring the model's ability to correctly classify videos as either genuine or manipulated. Fig. 5 presents the training and testing accuracy graphs for the EfficientNet-B7 model, showing a training accuracy of 75.41% and a testing accuracy of 66.65% over 10 epochs. The superior performance of the Xception model in both training and testing suggests its greater capability to discern patterns and attributes within the video content, making it the preferred choice for the deepfake detection mobile app.

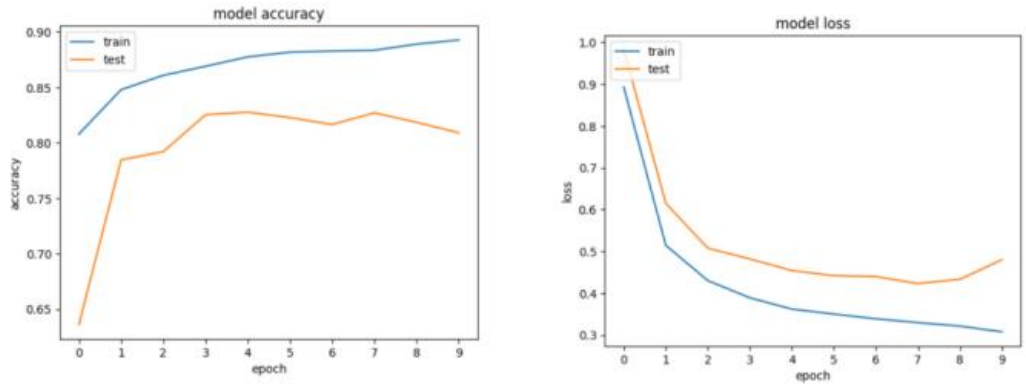


Fig. 4 *Training and testing results of the Xception model*

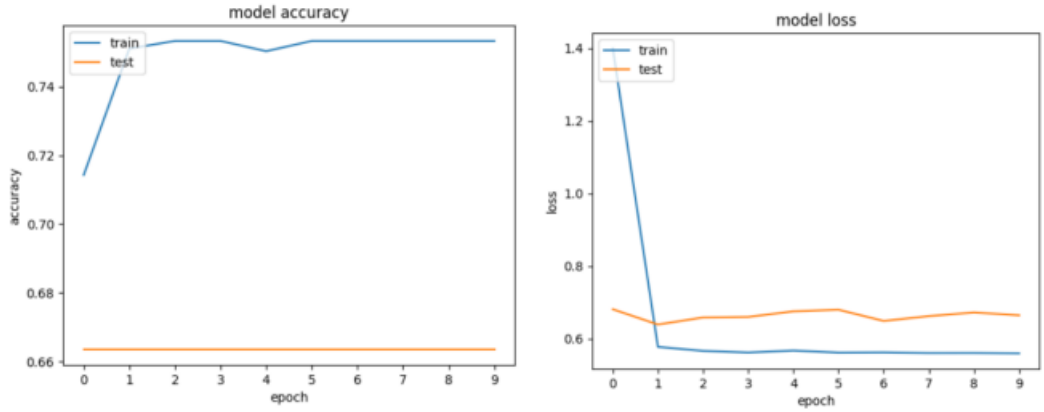


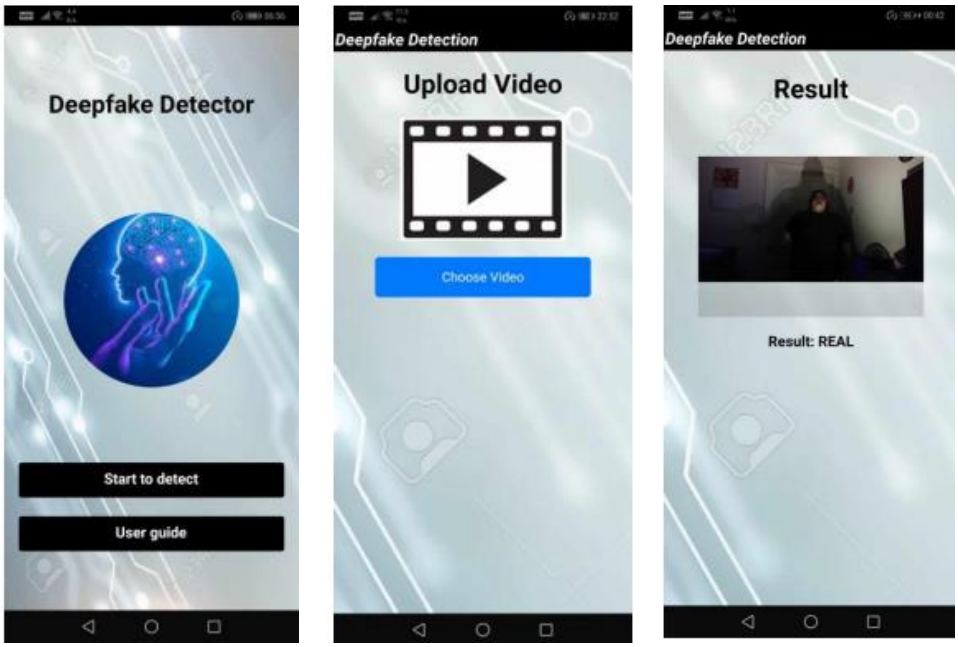
Fig. 5 *Training and testing results of the EfficientNet-B7 model*

3.2 Mobile App

Fig. 6 showcases selective screenshots of the mobile app designed for detecting deepfakes. Fig. 6a features the app's landing screen, while Fig. 6b demonstrates the interface for uploading videos from mobile devices. The app specifies that videos must be at least 10 seconds long and have a resolution between 360p and 720p to ensure accurate processing by the deep learning model. Upon selecting a valid video, a preview is displayed for user confirmation before uploading to the app's API for deepfake detection. Fig. 6c illustrates the prediction and result display by the app, showing the outcome returned by the API indicating whether the video is classified as

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 324-331

real or fake.



(a) **(b)** **(c)**

Fig. 6 *Screenshots of the Mobile App for deepfake detection. (a) Landing Screen, (b) Video Selection Screen, (c) Result Screen*

3.3 Evaluation the performance of the app

Fig. 7 presents a pie chart evaluating the accuracy of the mobile app. It tested 20 videos from the Kaggle DFDC dataset with known ground truth labels. The results show that 8 real videos (40%) were correctly classified as real, while 2 real videos (10%) were misclassified as deepfakes. Additionally, 7 deepfake videos (35%) were accurately identified, and 3 deepfake videos (15%) were incorrectly labeled as real. Calculating the binary classification accuracy, the model achieved an average accuracy of 75%. Although this performance is acceptable, the drop from the higher training accuracy suggests potential issues in handling the diversity and variability of real-world data. Factors such as differences in skin tone, ethnicity, accents, gender, and age could impact the model's accuracy.

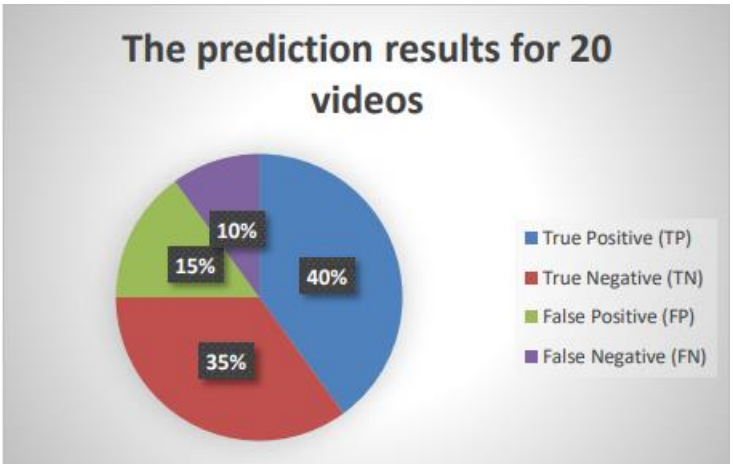


Fig. 7 *The mobile app's prediction results for 20 videos*

Fig. 8 displays a bar chart illustrating the execution speed of the app. For this analysis, 10 videos were selected, each processed 5 times. The slowest average prediction speed observed was 4.2693 seconds, while the fastest was 2.3202 seconds. Across all 50 runs, the average processing time calculated was 3.4891 seconds. Considering the computational complexity of the Xception model and the CPU-based testing environment, an

average prediction time of 3.4891 seconds can be deemed efficient for the mobile app's deepfake detection capabilities.

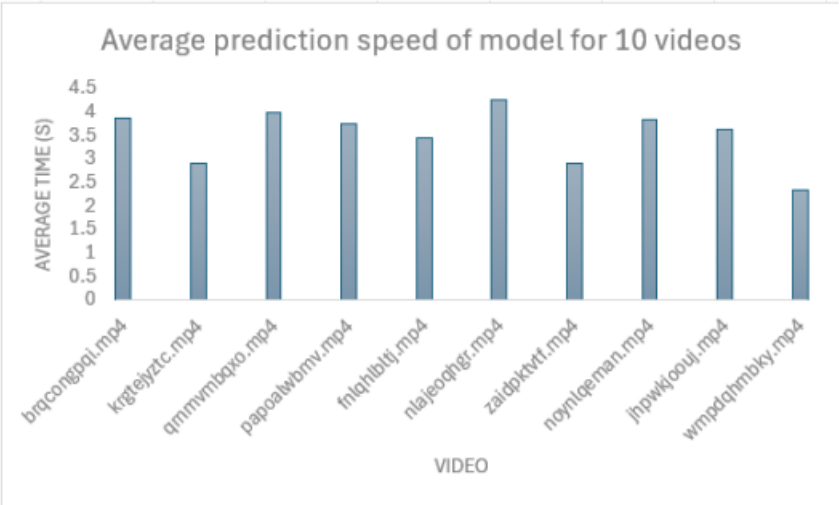


Fig. 8 *Average prediction speed of mobile app for 10 videos*

4. Conclusion

In conclusion, this project has successfully developed an Android application capable of real-time detection of deepfake videos, fulfilling all the set objectives. Through a thorough evaluation of Xception and EfficientNet-B7 models, it was determined that the Xception model demonstrates superior robustness in identifying deepfake content. This Xception model was then seamlessly integrated into the application, which was built using MIT App Inventor and Flask API, enabling users to verify the authenticity of videos in real time.

The performance assessment of the Xception model-based application revealed an accuracy rate of 75% in correctly identifying deepfake videos, with an average processing speed of 3.4891 seconds per video in a CPU-based environment. These findings are significant for enhancing the reliability of digital media, offering a valuable tool for users in various sectors, including media, law enforcement, and social media, to combat misinformation and preserve trust in digital content.

Future work will prioritize further enhancing the model's accuracy and prediction speed. Firstly, the training dataset will be expanded to include a more diverse set of samples, empowering the model to generalize better and improve its detection capabilities. Secondly, in-depth fine-tuning of the model parameters will be conducted to optimize its performance for the given task. Lastly, the implementation of a GPU-based environment will be explored to significantly boost the application's processing efficiency and reduce the overall latency in video authentication.

Acknowledgement

The authors would like to express their sincere gratitude to the Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, for the unwavering support provided throughout the completion of this work.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Chong Loo Jia, Nan Mad Sahar;* ***data collection:*** *Chong Loo Jia;* ***analysis and interpretation of results:*** *Chong Loo Jia;* ***draft manuscript preparation:*** *Chong Loo Jia, Nan Mad Sahar. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen, D. T. Nguyen, T. Huynh-The, S. Nahavandi, T. T. Nguyen, Q.-V. Pham, and C. M. Nguyen, "Deep learning for deepfakes creation and detection: A survey," 2019, arXiv:1909.11573.
[2] J.-Y. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks", Proc. IEEE Int. Conf. Comput. Vis. (ICCV), pp. 1-13, Oct. 2017.
[3] Y. Nirkin, Y. Keller and T. Hassner, "FSGAN: Subject agnostic face swapping and reenactment", Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), pp. 7183-7192, Oct. 2019.
[4] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation", Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 8789-8797, Jun. 2018.
[5] Y. Patel et al., "Deepfake Generation and Detection: Case Study and Challenges," in IEEE Access, vol. 11, pp. 143296-143323, 2023, doi: 10.1109/ACCESS.2023.3342107.
[6] Spring, M., “Sadiq Khan says fake AI audio of him nearly led to serious disorder.” 13-Feb-2024. [Online]. Available: https://www.bbc.com/news/uk-68146053 (Accessed: 16 June 2024).
[7] Passos, L.A., Jodas, D., Costa, K.A., Souza Júnior, L.A., Rodrigues, D., Del Ser, J., Camacho, D. and Papa, J.P., 2022. A review of deep learning-based approaches for deepfake content detection. https://doi.org/10.1111/exsy.13570
[8] M. Masood, M. Nawaz, A. Javed, T. Nazir, A. Mehmood, R. Mahum, Classification of deepfake videos using pre-trained convolutional neural networks, in: 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2), IEEE, 2021, pp. 1–6, https://doi.org/10.1109/ICoDT252288.2021.9441519
[9] Mohzary, M., Almalki, K. J., Choi, B. Y., & Song, S. (2023). MobiDeep: Mobile DeepFake Detection through Machine Learning-based Corneal-Specular Backscattering. Proceedings - IEEE Consumer Communications and Networking Conference, CCNC, 2023-January, 1104–1109. https://doi.org/10.1109/CCNC51644.2023.10059841
[10] Korea Tech Today, “KAIST Develops Media Deepfake AI Detecting Mobile App 'KaiCatch',” Mar. 2021.[Online]. Available:https://www.koreatechtoday.com/kaist-develops-media-deepfake-ai-detecting-mobile-app-kaicatch/
[11] Shah, D. "VIT-xception-for-deepfake-detection", GitHub. Available at: https://github.com/dhirajssh/vit-xception-for-deepfake-detection (Accessed: 15 March 2024).