INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Biometric Authentication based on Liveness Detection Using Face Landmarks and Deep Learning Model

Ooi Zhi Jie [a], Lim Tong Ming [b,*], Tan Chi Wee [a]

[a] *Faculty of Computing and Information Technology, Tunku Abdul Rahman University of Management and Technology, Malaysia*
[b] *Centre for Business Incubation and Entrepreneurial Ventures, Tunku Abdul Rahman University of Management and Technology, Malaysia*
*Corresponding author: *limtm@tarc.edu.my*

***Abstract*— This paper describes the approach to active liveness detection of the face using facial features and movements. The project aims to create a better method for detecting liveness in real-time on an application programming interface (API) server. The project is built using Python programming with the computer vision libraries OpenCV, dlib and MediaPipe and the deep learning library Tensorflow. There are five modules in active liveness detection progress related to different parts or movements on the face: headshakes, nodding, eye blinks, smiles, and mouths. The functionality of modules runs through face landmarking through dlib and MediaPipe and detection of face features through Tensorflow Convolutional Neural Network (CNN) trained in two different approaches: smile detection and eye-blink detection. The result of implementing face landmarking shows an accurate result through the pre-trained model of MediaPipe and the pre-trained parameter of the dlib 68 landmarking model. And more than 90% classification model accuracy in precision, recall, and f1-score for both trained CNNs in detecting smiles and eyes blinking through the Scikit-Learn classification report. In addition, the prototype API is also implemented using the Python RESTful API library, FastAPI, to test the detection functionality in the prototype Android application. The prototype result is outstanding, as the model excellently requests and retrieves from the API server. The possible research path gives the success of real-time detection on API servers for easy implementation of liveness detection on low-spec client devices.**

***Keywords*— Liveness detection; face landmarking; perspective-n-point problem; deep learning; computer vision; face motion.**

Manuscript received 15 Dec. 2022; revised 11 Jun. 2023; accepted 2 Aug. 2023. Date of publication 30 Nov. 2023.

I. Introduction

Face recognition and liveness detection are rapidly changing fields with lively research communities aimed at discovering new algorithms capable of delivering highly accurate and dependable results. Among the most captivating areas of study are deep learning techniques, which use neural networks to evaluate many facial images and videos. Additionally, traditional geometric modeling and statistical analysis are employed to identify patterns.

Studies in face recognition and liveness are essential because they have the potential to impact various sectors, such as security, law enforcement, marketing, and advertising. For instance, accurate face recognition technology can assist law enforcement in identifying suspects and tracking their movements. It also helps businesses better understand their clients and target their advertising more efficiently.

From Mordor-Intelligence's report, the potential market of facial recognition globally reached $4.4 billion in 2019 and expected to hit $10.9 billion by 2025 [1]. Some countries, example China have implemented this technology anywhere. Face recognition market growth is driven by rising demand for improved security and surveillance systems in various industries, including government, healthcare, and retail [2]. However, ethical concerns related to face recognition and liveness technologies, such as privacy, biases, and discrimination, are crucial. As a result, continued study in this field is essential to ensure that these technologies are developed and used responsibly and ethically.

A. Issue of Spoof Attack in Face Recognition

The major difficulty of face recognition is the false image attacks [3]. A spoof image like printings or 3D printed face mask of an individual's face may be used in this type of attack to gain unauthorized access to secure areas or sensitive data without additional security validation [4]. The related attacks are especially dangerous in high-security environments where access to sensitive information or restricted areas is strictly

controlled. Make face recognition technology in the banking industry an example, and false image attacks could allow unauthorized individuals to access bank accounts and conduct malicious transactions [2], [5]. The articles below also discuss the current weaknesses of the face recognition system and potential issues when confronted with a strong attack using fake images.

TABLE I
PUBLICATIONS THAT DISCUSSED ABOUT FAKE IMAGE ATTACK

Article	Topic Discussed
Face recognition under spoofing attacks: countermeasures and research directions.	Discuss methods of attack, existing methodology and given solution.[6]
A comparative study on face spoofing attacks.	Need to provide more generalized algorithms for the detection of unpredictable spoofing attacks [7]
Face Biometrics Under Spoofing Attacks: Vulnerabilities, Countermeasures, Open Issues, and Research Directions	Discussed spoofing and anti-spoofing in face biometrics and to highlight open issues and future directions [8]
A Study on Spoofing Face Detection System.	A comprehensive overview of the research that has been carried out over the last decades in the emerging field of anti-spoofing [9]
Statistical Analysis of Image Quality Measures for Face Liveness Detection.	A technique for discriminating life from fake images given. Based upon the hypothesis that spoofing schemes leave statistical indication or structure in images [10]

B. Objectives and Proposed Solution

The research aims to investigate methods and algorithms for detecting and validating whether the individual detected in front of the camera is a valid live person or an object with a spoofed image. For facial liveness, this project proposed a Face Recognition and Liveness Detection (FRLD) algorithm that focuses on detecting facial features and facial movement. [11]–[15]. As a result, FRLD can be implemented anywhere, like web applications, mobile applications, kiosk machines, etc. Finally, the face will be validated by a face validation module in order to identify an individual through a face image database.

The study will use a pre-trained model provided in dlib and MediaPipe, which is BlazeFace and dlib 68 landmarks [16]–[18] as fast implementation of face landmarking. The self-trained TensorFlow neural network is trained into two high-accuracy models: one model by two datasets of approximately 4000 for smiling and one model by approximately 4800 for eye-blinks. The algorithms designed and implemented in this research have the potential to greatly improve the security and efficiency of identity verification processes in a wide range of applications.

II. MATERIALS AND METHOD

A. Literature Review: Face Recognition

Face recognition is a technology that is being researched in the field of image processing. Human faces in videos or images are recognized and identified using computers. Normally, people recognize faces by comparing them to ones stored in their memory, and computers do the same with facial recognition systems. When inputting a video or picture into the system, it compares it to the previously stored faces and returns the percentage of similarities.

Bledsoe et al. [19] introduced the first face recognition system in 1964 that could detect face identity based on their features. With a semi-automatic process, operators were asked to input twenty metrics about face features, such as the size of the mouth or eyes. The technologies involved in this subject have progressed more recently than facial recognition technology. Additionally, it is currently frequently utilized in a variety of fields, including security, marketing, and entertainment.

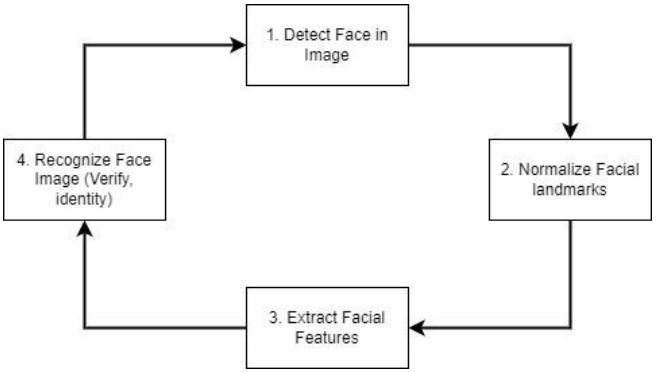

Fig. 1 Steps in facial recognition algorithm [19]

Face recognition software extracts face characteristics from an image or video input. This procedure is to recognize facial landmarks such as the nose, mouth, and eyes. Once these landmarks have been found, they are normalized into numerical values. These numerical data are sent into a machine learning model that has been trained to recognize and categorize different faces. To determine the identification of the individual in the input image or video, the machine learning model analyses the numerical values of the facial traits and compares them to a database of known faces.

B. Literature Review: Liveness Detection

Liveness detection is important in a face recognition system as it plays a crucial role in preventing fraud and ensuring the system's accuracy. Without liveness detection, the system could be vulnerable to manipulation using static images or videos rather than genuine live interactions. This could lead to erroneous identifications and potentially hazardous situations [20].

1) Motion: One of the approaches to achieving liveness detection involves utilizing motion as an indicator. When capturing a person's face in real-time, the 3D shape of their face changes with movement, making it challenging to deceive the system using a static 2D image. However, employing motion as an indicator has its limitations. For instance, it necessitates high-quality video for training, which

may slow down the face recognition process. Additionally, it can be deceived by 3D sculptures [20], [21].

2) Texture: The texture indicator relies on calculating the surface of the "face" input to the face recognition system. Training the model to detect surfaces makes it possible to determine if the face input is a genuine image or a mere photo on paper or a screen. This indicator offers the advantage of convenient implementation since it only requires samples of real and non-real faces. However, a drawback of using texture as an indicator is its inability to detect surface texture when input images or videos are of low quality, and it demands more samples for texture detection training [20], [22].

3) Life Sign: Lastly, the life sign is optimal for implementing a faster face biometric authentication system. The life sign indicator necessitates user interaction with the camera to detect life signs, such as moving the face in specific directions or blinking eyes. Nevertheless, a disadvantage of this indicator is that it will not work when facial features are covered by hands or when wearing a face mask [20].

C. Literature Review: Dataset

Whitehill et al. [23] introduced the GENKI dataset, which comprises 63,000 unfiltered images gathered from personal web pages on the internet. Unlike images collected by laboratory scientists, ordinary individuals captured these worldwide for their personal use. The dataset encompasses three distinct categories: Smile, Non-Smile, and Unclear. For our project, where rapid detection is required, we will focus solely on the GENKI-4K subset. This subset contains 4,000 images, specifically chosen to include only the Smile and Non-Smile categories.

D. Proposed FRLD Algorithm

The FRLD algorithm proposed in this paper is a combination of different Liveness Detection functionalities. It comprises five distinct functions based on MediaPipe Facemesh, dlib 68 landmarks, and TensorFlow Convolutional Neural Networks. Face shaking and nodding functions rely on MediaPipe FaceMesh, which depends on the Blazeface face detection model [18] to determine face direction. Eye-blinks and Smiling functions are processed by separate trained TensorFlow CNN models, producing Boolean outputs based on specific detection needs. The mouth opening function calculates the gap between the upper and lower lips using dlib 68 landmarks [16],[20]. FRLD enables easy implementation of five different liveness detections for real individuals in front of the camera.

The self-trained TensorFlow CNN for liveness detection uses separate training and validation sets. For instance, the smile detection CNN is trained with 20% validation set (981 images) and 80% training set (3921 images). The model's output predicts whether an input image shows a person smiling or not. The project aims to implement a model capable of extracting both face and liveness features from input, enabling face recognition. The resulting process will be thoroughly evaluated to determine its success in verification.

1) Liveness Detection: Head Shake and Head Nod Module: In the Head Shake and Head Nod Module, frames undergo facial landmark detection using MediaPipe FaceMesh [24], which relies on the BlazeFace model [18] to locate key facial landmarks in 2D (x, y) and 3D (x, y, z). After identifying the landmarks, the face direction is calculated using Camera Calibration and Rodrigues' rotation formula [25] to determine the nose angle in 2D. This enables the detection of face shaking or nodding.

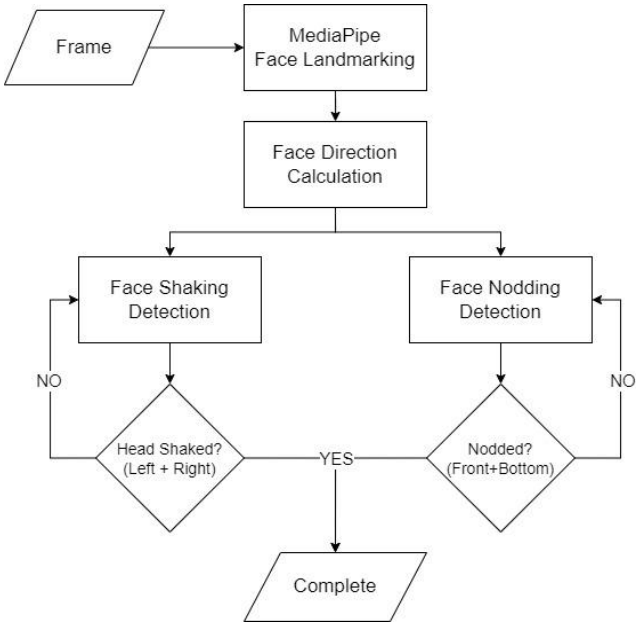

Fig. 2 Flowchart of Head Shake and Head Nod Module

The improved Face Orientation Direction algorithm, based on Camera Calibration and Rodrigues' rotation formula [25], [26] on the MediaPipe FaceMesh, is as follows:

- Perspective-n-Point (PnP) Problem [27] are referred to solve the calculation of face angle by face landmarks received. The formula following is the equations that PnP problem mentioned.

$$m' = A\,[R|t]M' \quad (1)$$

Or,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2)$$

a) (u, v) is coordinates of the projection point in pixel unit.
b) A is a camera matrix, or a matrix of intrinsic parameters.
c) (cx, cy) is a principal point that is usually at the image center.
d) fx, fy are the focal lengths expressed in pixel units.
e) (X, Y, Z) are the coordinates of a 3D point in the world coordinate space.
f) And [R|t] is the rotation and transformation matrix.

- The angle of the head pose is taken from the calculation in PnP problem, the (X, Y, Z) is the exact direction that the head facing. Assume the X and Y as angle and draw a line from the middle point of the 2D graph which is (0,0). The direction of the face can be calculated when,
  a) $y = +ve$ and $|y| = Thereshold$, then head pose should be "Look Up."
  b) $y = -ve$ and $|y| = Thereshold$, then head pose should be "Look Down."
  c) $x = +ve$ and $|x| = Thereshold$, then head pose should be "Look right."
  d) $x = -ve$ and $|x| = Thereshold$, then head pose should be "Look Left"
  e) If four requirements above not fulfilled, consider the *head* pose is "Look Front"
- Refer to the **Fig. 3** as direction reference

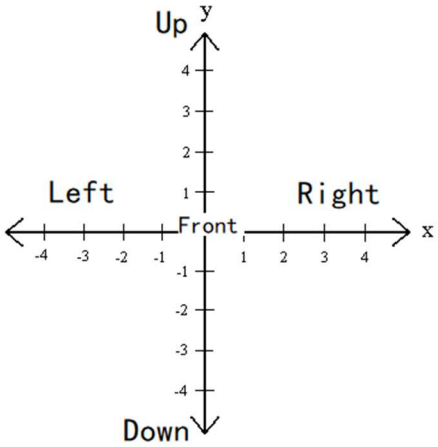

Fig. 3 Example of an image with acceptable resolution

2) Liveness Detection: Eye-Blink Module: In the blink module, frames are initially processed through the open-cv haarcascades algorithm's left-eye and right-eye configurations to obtain cropped images of both eyes. These cropped images are then input into the Eye-Blink TensorFlow Model, which generates binary outputs ("0" for not closed and "1" for closed) for each eye. If both eyes are detected as closed (output: "1" for both), the module registers a blink and concludes its operation [28], [29][30].

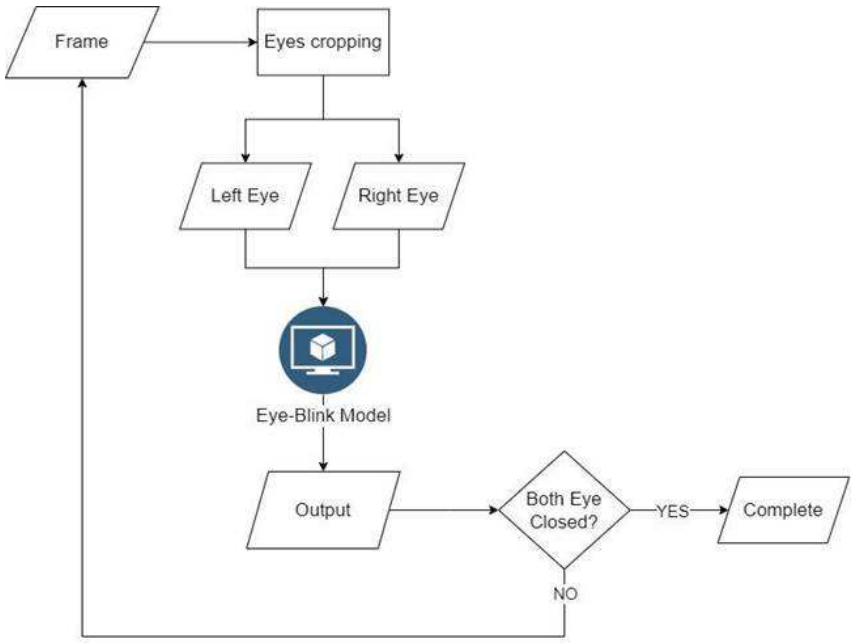

Fig. 4 Flowchart of Eye-Blink Module

The combined approach of open-cv haarcascades algorithm and the Eye-Blink TensorFlow Model follows these steps:

- The original frame with face will crop down the left eye and right eye by using open-cv haarcascades algorithm's left eye configuration and right eye configuration.
- The cropped eye's frame will resize into 24 x 24-pixel image to fulfill the input size of the model.
- The improvement of the input allows any resolution's human face image to be detected by the Eye-Blink

Module because the resolution will be lower down for faster detection speed in Eye-Blink TensorFlow Model

3) Liveness Detection: Smile Module: In the smile module, the frames are first passed through the face-cropping function based on the open-cv haarcascades algorithm's configuration to obtain the cropped images of the face. These images are then input into the Smile TensorFlow Model, which produces an output for face expression in "0" or "1" (non-smile or smile). If the output shows that the face is smiling, the module considers the smile to have occurred and finishes its operation [31], [32]. The combined open-cv haarcascades algorithm with the Smile TensorFlow Model is elaborated in the following set of algorithms.

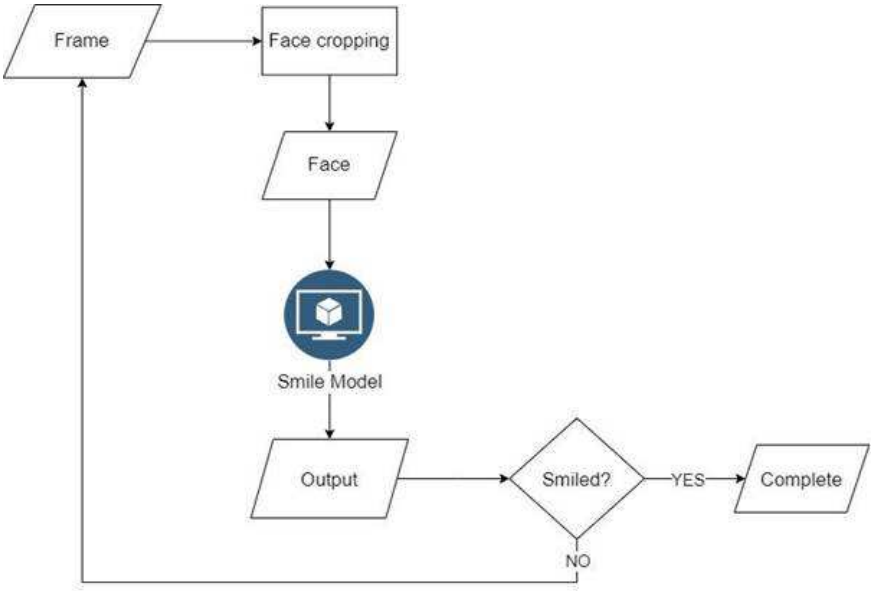

Fig. 5 Flowchart of Smile Module

- The original frame with face will crop down only the face part by using open-cv haarcascades algorithm's front face configuration.
- The cropped face frame will resize into 64 x 64-pixel image to fulfill the input size of the model.
- The improvement of the input allows any resolution's human face image to be detected by the Smile Module because the resolution will be lower down for faster detection speed in Smile TensorFlow Model
- The simplified output in "0" and "1" ease to output the Boolean result for the detection since the detection is to get the result of smiling. The "0" would represent the face expression is non-smile and "1" as smiling face.

4) Liveness Detection: Mouth Opening Module: In the mouth opening module, the frames are first passed through the dlib 68 face landmarking model to detect and locate the face landmarks. Once the landmarks are identified, the distance of lips, known as Mouth Aspect Ratio (MAR), is calculated in integer value based on algorithm from Mauck R. By allocating the threshold value, we allow it to detect if the mouth in the image is opened. The mouth will be considered open when the MAR detected is larger than the allocated threshold value. (MAR > threshold) The procedure of the improved algorithm is shown below:

- We first refer to 68 face landmarks with every indexed landmark to get the mouth landmarks.

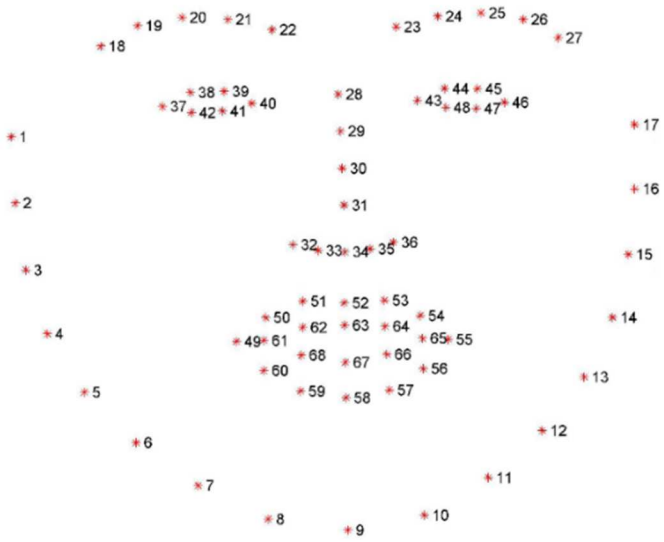

Fig. 6 dlib 68 Face Landmark Reference Index [16]

- After that, the calculation of Mouth Aspect Ratio (MAR) will be following the equation:

Vertical Euclidean distances of the right side of the mouth.

$$A = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}, \quad (1)$$

$$p(x, y) = Landmark\ 51, q(x, y) = Landmark\ 59 \quad (2)$$

Vertical Euclidean distances of left side of mouth.

$$B = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}, \quad (3)$$

$$p(x,y) = Landmark\ 53, q(x,y) = Landmark\ 57 \quad (4)$$

Horizontal Euclidean distances of mouth.

$$C = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \quad (5)$$

$$p(x,y) = Landmark\ 49, q(x,y) = Landmark\ 55 \quad (6)$$

$$Mouth\ Aspect\ Ratio, MAR = (A + B) \div (2.0 \div C) \quad (7)$$

When MAR is larger than threshold value, the mouth will be considered as open.

III. Results and Discussion

A. Implementation of the Proposed Algorithm and Validation

The landmarking function of MediaPipe [24] enables the developer to attach it for several needs in capturing the face and face landmarks. Example by using the FaceMesh function in MediaPipe we will be able to plot the face area that MediaPipe FaceMesh function detected. The face landmarking function in dlib library also provides powerful face landmarking features as the result of mouth detection is overall powerful.

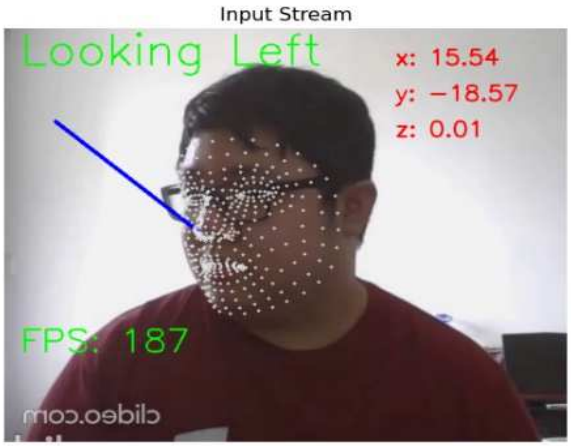

Fig. 7 Result of FaceMesh using MediaPipe when proceed the Face Shaking and Face Nodding Module

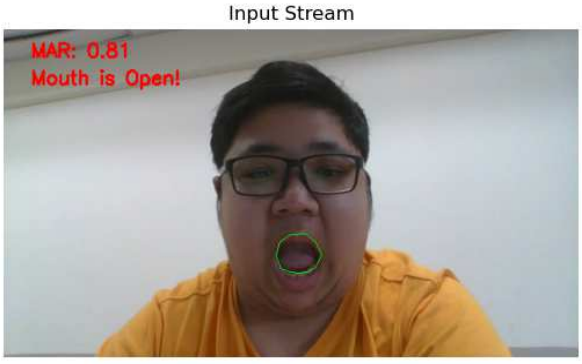

Fig. 8 Result of face landmarking using dlib when proceed the Mouth Opening Module

B. Testing of Self-trained TensorFlow Model (Smiling and Eye-Blink)

1) Smile Detection: Using the GENKI-4K and Smiling or Not Datasets. The training scoring evaluation shows that the model can achieve more than 90% of scorings in terms of precision, recall and f1-score when detecting the smile image and overall accuracy of the model also achieve 91% in detecting the smile and non-smile image.

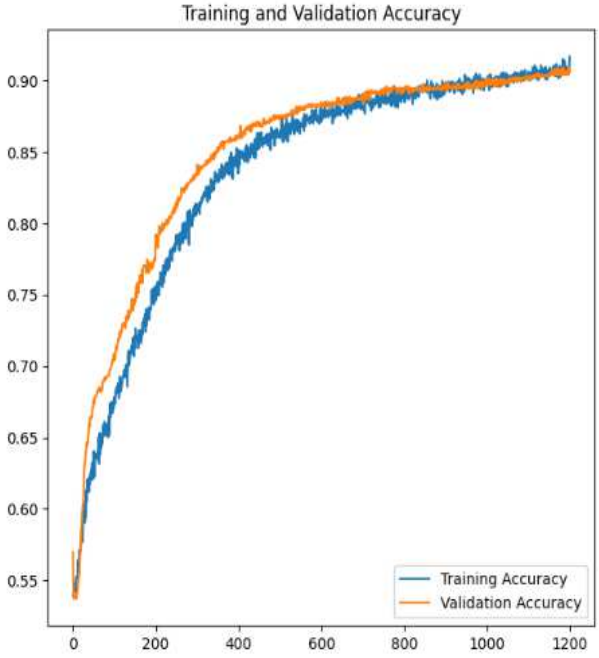

Fig. 9 Training and validation accuracy of Smile Tensorflow Model

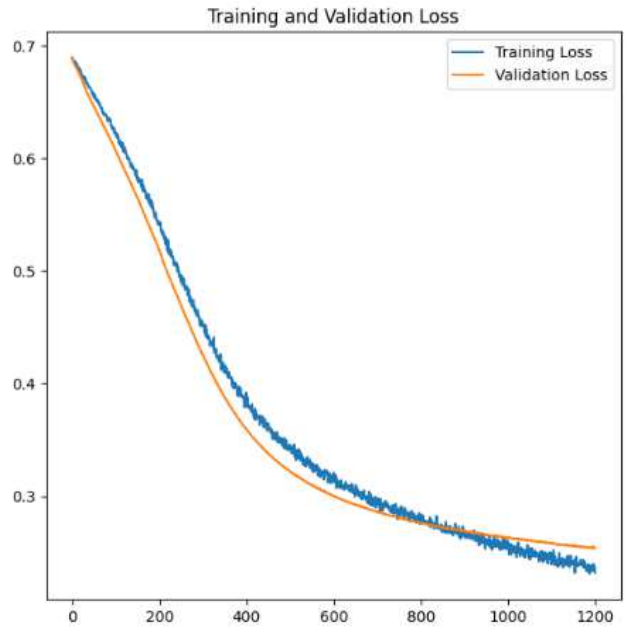

Fig. 10 Training and validation loss of Smile Tensorflow Model

TABLE II
CLASSIFICATION REPORT OF SMILE TENSORFLOW MODEL

	precision	recall	f1-score	support
Non-Smile (Class 0)	0.90	0.91	0.90	454
Smile (Class 1)	0.92	0.91	0.92	527
accuracy			0.91	981
macro avg	0.91	0.91	0.91	981
weighted avg	0.91	0.91	0.91	981

2) Eye-blink Detection: The training scoring evaluation shows that the model can achieve more than 90% of scoring in terms of precision, recall and f1-score when detecting the

closed eye image and overall accuracy of the model also achieve 92% in detecting the open and closed eye.

TABLE III
CLASSIFICATION REPORT OF BLINK TENSORFLOW MODEL

	precision	recall	f1-score	support
Open (Class 0)	0.92	0.93	0.92	543
Closed (Class 1)	0.93	0.92	0.92	525
accuracy			0.92	1068
macro avg	0.92	0.92	0.92	1068
weighted avg	0.92	0.92	0.92	1068

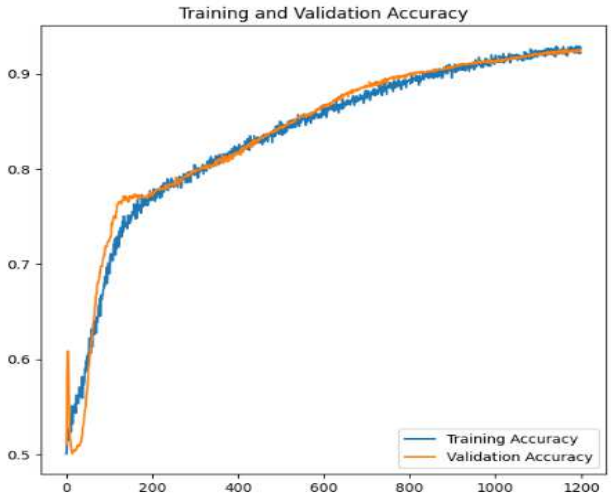

Fig. 11 Training and validation accuracy of Blink Tensorflow Model

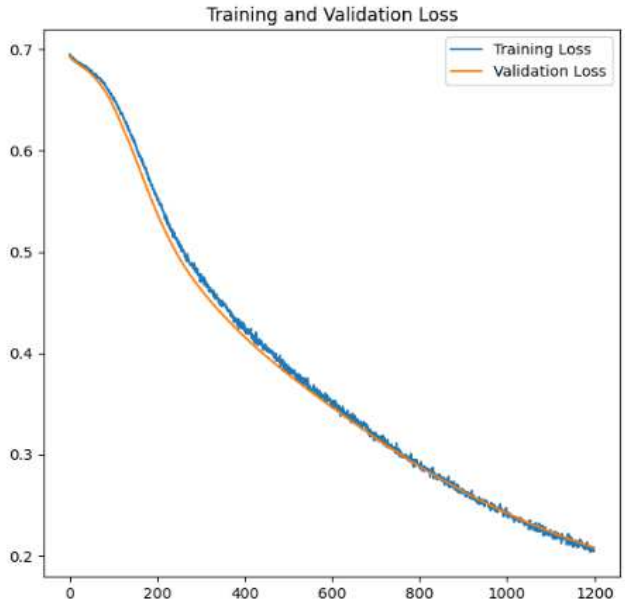

Fig. 12 Training and validation loss of Blink Tensorflow Model

C. Prototype Deployment

To assess the reliability of the FRLD algorithm, deployment becomes essential to identify any potential issues in the proposed detection logic. For the API deployment, FastAPI [33] will be utilized as it allows for swift deployment of a prototype API. FastAPI is a lightweight Python-based web framework designed for creating high-performance APIs using standard Python type hints. Its use of open standards enables the generation of production-ready code and automatic interactive documentation [34].

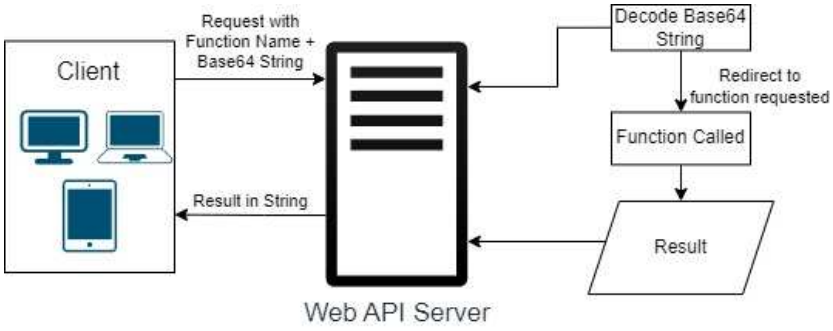

Fig. 13 Flowchart of API Prototype

On the client side, the Android platform will serve as the front-end client for capturing front-camera images. These images will be transmitted to the server using the Volley library, which can send images as string data. Upon receiving the image data, the server will leverage Python's image processing libraries, such as OpenCV or Pillow, to analyze and manipulate the images as required. The server will then respond to the Android client with the image processing findings, which may comprise text extracted from the picture or object recognition data.

RESTful API can simplify the communication between the front-end and back-end. This is because RESTful API standardized the way of data transmission between different systems, ensuring efficient and reliable data exchange.

During the server setup, API testing can be conducted using any programming language capable of converting images into base64 strings and sending them to the web server through API requests. This allows for comprehensive testing of the API's functionality, ensuring that the server can accurately transmit and process images.

As for the Android implementation, the image can be captured through the device's camera and saved as a bitmap image file. Before being delivered to the server, the bitmap picture can be encoded into a base64 string format. This procedure guarantees that the picture is safely transferred, and that the server can quickly handle it upon reception.

Utilizing base64 encoding facilitates the inclusion of binary data, such as images, within the API request. This encoding ensures that the image data is safely transmitted

over networks without the risk of data corruption or loss during transmission.

Overall, combining base64 encoding for image data and implementing RESTful API endpoints creates a robust and reliable communication channel between the front-end Android client and the back-end server. This setup enables seamless testing and deployment of the FRLD algorithm in real-world scenarios, providing valuable insights into the algorithm's performance and reliability in practical applications.

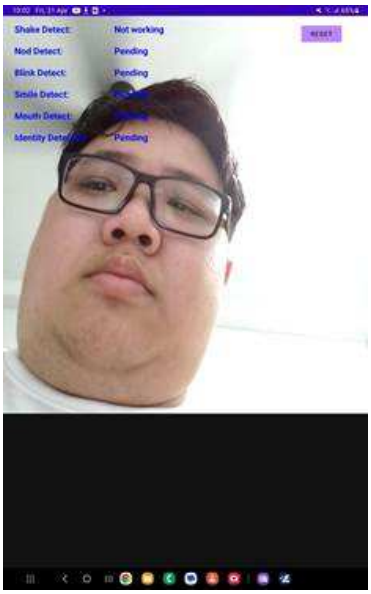

Fig. 14 Interface of Android prototype

Fig. 15 Information panel of Android prototype

IV. Conclusion

In this study, an API prototype is built and testing in an Android mobile device to test for reliability of the API to mobile device. The success of detection the liveness by front camera image on mobile phone means the mindset of process the image on server or serverless service are possible. And suits the concept of edge computing [35]. Our team expects more researchers to focus on this topic and propose a better solution in detecting face liveness.

The study also helps to reduce false positives in face recognition systems. By ensuring that the face being scanned is a live person, the system can be more accurate in identifying individuals. This is especially critical in applications like access control and identity verification, where false positives can lead to unauthorized access or security breaches. The system can be more dependable and accurate with liveness detection, lowering the danger of false positives.

From hospitals and government buildings to sports stadiums and music halls, this adaptability may be useful in a variety of sectors and environments. With liveness detection, the system can be more accurate and secure, providing enhanced protection and peace of mind.

A. Achievements

Remarkably, MediaPipe and dlib are able to compute the face direction and mouth opening without extra effort, such as machine learning or pre-processing of the input image. Unfortunately, the smile detection model still has some difficulties detecting smiles, which may be related to the limited sample size used in training. However, overall, the self-trained model achieves high scores in detection.

During the deployment stage, it is fortunate that the image from the Android application prototype still works after being sent through the network to the server by using base64 encoding to convert the image to a string. The Web API server is functioning well, with some minor errors such as connection issues when using different internet networks (e.g., accommodation versus university). The university's firewall is believed to be blocking potential spamming of requests to prevent DDOS attacks.

B. Limitations and Future Improvements

Because of the time limitation, the project could only proceed with implementing a slightly trained model combined with a pre-trained model for the system. One possible improvement for the project would be to add more datasets or test different variants of Deep Learning models, apart from Convolutional Neural Network. Transfer learning for the face features model could also be tested to determine its performance compared to that of this project. Additionally, fine-tuning the model using a larger dataset or more sophisticated techniques, such as adversarial training, could further improve the system's robustness against attacks.

The deployment of the client prototype is still too simple and buggy. The way of implementing the client through webpage would be more reliable in this case as the using of API leads to necessary of internet connectivity like webpage. Moreover, the website deployment could enable more flexible and efficient access to the system, with the ability to handle multiple requests and provide real-time feedback to users. The website could also offer more customization options and integrate with other tools or services, such as biometric authentication or identity management platforms.

Acknowledgements

We thank Tunku Abdul Rahman University of Management and Technology (TAR UMT) for providing financial support and technical support when completing this study.

References

[1] Mordor-Intelligence, "Facial Recognition Market - Growth, Trends, Covid-19 Impact, and Forecasts (2023 - 2028)," *Mordorintelligence.com*, [Online]. Available: https://www.mordorintelligence.com/industry-reports/facial-recognition-market.

[2] D. Sharma and A. Selwal, "A survey on face presentation attack detection mechanisms: hitherto and future perspectives," *Multimed Syst*, vol. 29, no. 3, pp. 1527–1577, 2023, doi: 10.1007/s00530-023-01070-5.

[3] J. K. Khan and D. Upadhyay, "Security issues in face recognition," *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, Sep. 2014, doi: 10.1109/confluence.2014.6949341.

[4] R. Singh, A. Agarwal, M. Singh, S. Nagpal, and M. Vatsa, "On the Robustness of Face Recognition Algorithms Against Attacks and Bias," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 09, pp. 13583–13589, Apr. 2020, doi: 10.1609/aaai.v34i09.7085.

[5] M. C. Gentile, D. Danks, and M. Harrell, "Case Study: Does Facial Recognition Tech Enhance Security?," *Harvard Business Review*. Nov. 2022. [Online]. Available: https://hbr.org/2022/11/does-facial-recognition-tech-enhance-security

[6] L. Li, P. L. Correia, and A. Hadid, "Face recognition under spoofing attacks: countermeasures and research directions," *IET Biom*, vol. 7, no. 1, pp. 3–14, Jan. 2018, doi: 10.1049/iet-bmt.2017.0089.

[7] S. Kumar, S. Singh, and J. Kumar, "A comparative study on face spoofing attacks," *IEEE Xplore*. pp. 1104–1108, May 2017. doi: 10.1109/CCAA.2017.8229961.

[8] A. Hadid, "Face Biometrics Under Spoofing Attacks: Vulnerabilities, Countermeasures, Open Issues, and Research Directions," *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2014, doi: 10.1109/cvprw.2014.22.

[9] P. Kavitha and K. Vijaya, "A Study on Spoofing Face Detection System," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:198996446

[10] E. A. Raheem and Ahmad, "Statistical Analysis of Image Quality Measures for Face Liveness Detection," *Lecture notes in electrical engineering*, vol. 547, Jan. 2019, doi: 10.1007/978-981-13-6447-1_69.

[11] Y. Li, Y. Li, Q. Yan, H. Kong, and R. H. Deng, "Seeing Your Face Is Not Enough," *Computer and Communications Security*, Oct. 2015, doi: 10.1145/2810103.2813612.

[12] D. Garud and S. S. Agrwal, "Face liveness detection," *IEEE Xplore*. pp. 789–792, Sep. 2016. doi: 10.1109/ICACDOT.2016.7877695.

[13] J. Yang, Z. Lei, S. Liao, and S. Z. Li, "Face Liveness Detection with Component Dependent Descriptor," *IEEE Xplore*. pp. 1–6, Jun. 2013. doi: 10.1109/ICB.2013.6612955.

[14] A. Ali, F. Deravi, and S. Hoque, "Liveness Detection Using Gaze Collinearity," 2012 Third International Conference on Emerging Security Technologies, Sep. 2012, doi: 10.1109/est.2012.12.

[15] L. Wang, X. Ding, and C. Fang, "Face live detection method based on physiological motion analysis," *Tsinghua Sci Technol*, vol. 14, no. 6, pp. 685–690, Dec. 2009, doi: 10.1016/s1007-0214(09)70135-x.

[16] M. Jabberi, A. Wali, B. B. Chaudhuri, and A. M. Alimi, "68 landmarks are efficient for 3D face alignment: what about more?," *Multimed Tools Appl*, Apr. 2023, doi: 10.1007/s11042-023-14770-x.

[17] S. Hangaragi, T. Singh, and N. N, "Face Detection and Recognition Using Face Mesh and Deep Neural Network," *Procedia Comput Sci*, vol. 218, pp. 741–749, 2023, doi: 10.1016/j.procs.2023.01.054.

[18] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub-millisecond Neural Face Detection onMobile GPUs," *arXiv:1907.05047 [cs]*, Jul. 2019, [Online]. Available: https://arxiv.org/abs/1907.05047

[19] P. Shaha, U. Sharma, and K. Pawar, "Face Recognition Technology," *International Journal of Research in Engineering, Science and Management*, vol. 1, no. 9, pp. 149–151, Sep. 2018, [Online]. Available: https://www.ijresm.com/Vol_1_2018/Vol1_Iss9_September18/IJRESM_19_32.pdf

[20] S. Chakraborty and D. Das, "An Overview of Face Liveness Detection," *International Journal on Information Theory*, vol. 3, no. 2, pp. 11–25, Apr. 2014, doi: 10.5121/ijit.2014.3202.

[21] S. Li, X. Dong, Y. Shi, B. Lu, L. Sun, and W. Li, "Multi-angle head pose classification with masks based on color texture analysis and stack generalization," *Concurr Comput*, vol. 35, no. 18, 2023, doi: 10.1002/cpe.6331.

[22] C. Gao, X. Li, F. Zhou, and S. Mu, "Face liveness detection based on the improved CnN with context and texture information," *Chinese Journal of Electronics*, vol. 28, no. 6, pp. 1092–1098, 2019, doi: 10.1049/cje.2019.07.012.

[23] J. Whitehill, G. Littlewort, I. Fasel, M. Bartlett, and J. Movellan, "Toward Practical Smile Detection," *IEEE Trans Pattern Anal Mach Intell*, vol. 31, no. 11, pp. 2106–2111, Nov. 2009, doi: 10.1109/tpami.2009.42.

[24] I. Grishchenko and V. Bazarevsky, "MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device," *Google AI Blog*. Dec. 2020. [Online]. Available: https://ai.googleblog.com/2020/12/mediapipe-holistic-simultaneous-face.html

[25] L. Fraiture, "A History of the Description of the Three-Dimensional Finite Rotation," *Journal of The Astronautical Sciences*, vol. 57, no. 1–2, pp. 207–232, Jan. 2009, doi: 10.1007/bf03321502.

[26] X. Zhao, S. Sulaiman, L. Chen, M. Dong, Y. Duo, and H. Song, "Continuity Rotation Representation for Head Pose Estimation without Keypoints," in *ACM International Conference Proceeding Series*, 2023, pp. 358–363. doi: 10.1145/3594315.3594341.

[27] F. Rocca, M. Mancas, and B. Gosselin, "Head Pose Estimation by Perspective-n-Point Solution Based on 2D Markerless Face Tracking," *Springer eBooks*, vol. 136, pp. 67–76, Jul. 2014, doi: 10.1007/978-3-319-08189-2_8.

[28] E. Garea-Llano and A. Morales-Gonzalez, "Framework for biometric iris recognition in video, by deep learning and quality assessment of the iris-pupil region," *J Ambient Intell Humaniz Comput*, vol. 14, no. 6, pp. 6517–6529, 2023, doi: 10.1007/s12652-021-03525-x.

[29] V. Panwar and Pooja, "A Review on Iris Recognition System using Machine and Deep Learning," in *3rd IEEE 2022 International Conference on Computing, Communication, and Intelligent Systems, ICCCIS 2022*, 2022, pp. 857–866. doi: 10.1109/ICCCIS56430.2022.10037643.

[30] N. K. Singh, S. Mishra, and A. Bhardwaj, "Eye Blinking Detection Test," in *Proceedings - 2021 3rd International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2021*, 2021, pp. 1734–1736. doi: 10.1109/ICAC3N53548.2021.9725633.

[31] R. Rao and V. N. Hedge, "Recognition and Classification of Smiles using Computer Vision," in *2022 1st International Conference on Artificial Intelligence Trends and Pattern Recognition, ICAITPR 2022*, 2022. doi: 10.1109/ICAITPR51569.2022.9844198.

[32] L. Ruan, Y. Han, J. Sun, Q. Chen, and J. Li, "Facial expression recognition in facial occlusion scenarios: A path selection multi-network," *Displays*, vol. 74, 2022, doi: 10.1016/j.displa.2022.102245.

[33] P. Bansal and A. Ouda, "Study on Integration of FastAPI and Machine Learning for Continuous Authentication of Behavioral Biometrics," in *2022 International Symposium on Networks, Computers and Communications (ISNCC)*, 2022, pp. 1–6. doi: 10.1109/ISNCC55209.2022.9851790.

[34] P. Bansal and A. Ouda, "Study on Integration of FastAPI and Machine Learning for Continuous Authentication of Behavioral Biometrics," in *2022 International Symposium on Networks, Computers and Communications, ISNCC 2022*, 2022. doi: 10.1109/ISNCC55209.2022.9851790.

[35] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An Overview on Edge Computing Research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020, doi: 10.1109/access.2020.2991734.