



The Computational and Performance Aspects of Masked Face Detection and
Recognition

Kachasak Intim

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Applied Mathematics and Computing Science

Prince of Songkla University

2023

Copyright of Prince of Songkla University



The Computational and Performance Aspects of Masked Face Detection and
Recognition

Kachasak Intim

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Applied Mathematics and Computing Science

Prince of Songkla University

2023

Copyright of Prince of Songkla University

Thesis Title The Computational and Performance Aspects of Masked Face
Detection and Recognition

Author Mr. Kachasak Intim

Major Program Applied Mathematics and Computing Science

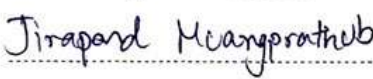
Major Advisor


.....
(Asst. Prof. Dr. Apirat Wanichsombat)

Co-Advisor



.....
(Asst. Prof. Dr. Nathaphon Boonnam)

Examining Committee:


..... Chairperson
(Assoc. Prof. Dr. Jirapond Muangprathub)


..... Committee
(Asst. Prof. Dr. Nualsawat Hiransakolwong)


..... Committee
(Asst. Prof. Dr. Apirat Wanichsombat)


..... Committee
(Asst. Prof. Dr. Nathaphon Boonnam)



..... Committee
(Asst. Prof. Dr. Siriwan Kajornkasirat)

The Graduate School, Prince of Songkla University, has approved this thesis as partial fulfillment of the requirements for the Master of Science Degree in Applied Mathematics and Computing Science.


.....
(Asst. Prof. Dr. Thakerng Wongsirichot)

Acting Dean of Graduate School

This is to certify that the work here submitted is the result of the candidate's own investigations. Due acknowledgment has been made of any assistance received.


.....Signature
(Asst. Prof. Dr. Apirat Wanichsombat)

Major Advisor


.....Signature

(Asst. Prof. Dr. Nathaphon Boonnam)

Co-Advisor


.....Signature

(Mr. Kachasak Intim)

Candidate

ชื่อวิทยานิพนธ์	ประสิทธิภาพเชิงคำนวณและแม่นยำของการตรวจจับและจดจำใบหน้า ที่สวมหน้ากากอนามัย
ผู้เขียน	นาย ศุภศักดิ์ อินทร์ทิม
สาขาวิชา	คณิตศาสตร์ประยุกต์และวิทยาการคำนวณ
ปีการศึกษา	2565

บทคัดย่อ

งานวิจัยนี้เกี่ยวกับการพัฒนาระบบตรวจจับหน้ากาก และการจดจำใบหน้าขณะสวมหน้ากากอนามัย เป็นการศึกษาเพื่อทดสอบประสิทธิภาพของโมเดลต่าง ๆ ได้แก่การตรวจจับหน้ากากใบหน้า และการจดจำใบหน้าสำหรับการใช้งานจริง ในการศึกษา โมเดลตรวจจับใบหน้า ได้แก่ Haar Cascade, SSD, HOG และ MTCNN และสถาปัตยกรรมจดจำใบหน้า ได้แก่ VGG, Inception-ResNet-v2, ResNet50, EfficientNet โดยเราจะใช้การประเมินผลของโมเดลตรวจจับใบหน้าและสถาปัตยกรรมจดจำใบหน้าจากการค้นคว้าจากงานวิจัยต่าง ๆ ที่มีการเปรียบเทียบข้างต้นแล้ว และการทำงานที่เพิ่มขึ้นคือ มีระบบวัตถุภูมิที่พัฒนาบนบอร์ดไมโครคอนโทรลเลอร์ Jetson Xavier NX จากการศึกษาพบว่า โมเดลที่เหมาะสมที่สุดสำหรับการตรวจจับใบหน้าคือโมเดล SSD ซึ่งเร็วและเล็กกว่าโมเดลตรวจจับใบหน้าอื่น ๆ และ เราเลือกใช้โมเดลการจดจำใบหน้าคือ Face Net ซึ่งเป็นโมเดลที่ใช้วิธีการสูญเสียสามเท่าซึ่งมีความแม่นยำดีมาก และมีสถาปัตยกรรม Inception-ResNet-v2 พร้อมการตรวจจับวัตถุภูมิและการทำงานขั้นตอนสุดท้ายของระบบจะเป็นการรายงานผลงานไปยังแอปพลิเคชัน LINE การทดสอบการใช้งานจริงกับบุคลากรโรงเรียน มอ. วิทยานุสรณ์ สุราษฎร์ธานี โดยมีผู้เข้าร่วมทดลอง 76 คน โดยระบบสามารถตรวจจับหน้ากากอนามัยได้ดีมากมีความแม่นยำอยู่ที่ 99% ในขณะเดียวกัน ระบบสามารถจดจำใบหน้าได้ 91% จากผู้เข้าร่วมทดลองทั้งหมด

Thesis Title	The Computational and Performance Aspects of Masked Face Detection and Recognition
Author	Mr. Kachasak Intim
Major Program	Applied Mathematics and Computing Science
Academic Year	2022

ABSTRACT

This research focuses on developing a mask detection and facial recognition system for real-world applications. The study tests different models for face detection, including Haar Cascade, SSD, HOG, and MTCNN, as well as facial recognition architectures, including VGG, Inception-ResNet-v2, ResNet50, and EfficientNet. The chosen face detection model is the SSD model, which is faster and smaller than other models, and the facial recognition model is FaceNet, a triple lossless model with high accuracy that uses the Inception-ResNet-v2 architecture and includes temperature detection. The final system includes reporting results to the LINE application and was tested with 76 participants from PSU. Wittayanusorn Surat Thani School. The system achieved 99% accuracy in detecting masks and recognized 91% of the faces of all subjects.

ACKNOWLEDGMENT

I sincerely thank the thesis advisor Asst. Prof. Dr. Apirat Wanichsombat and Co-Adviser Asst. Prof. Dr. Nathaphon Boonnam for teaching supporting everything with better advice and most importantly. The encouragement that the two teachers have always given this thesis may not done if without the supporting from them for everything in the duration of this research.

I want to thank my family, especially my mom for always supporting me. And encourage my thanks to all my friends who contributed to the development of the system and encouraged me, therefore, I was never discouraged when I met obstacles during the dissertation.

I would like to express my sincere gratitude to the personnel of PSU. Wittayanusorn Surat Thani School uses facial information for system development and support in various matters, thus making this thesis a success.

I would like to thank the Prince of Songkla University this research was funded by the Prince of Songkla University, Thailand. And I would like to thank Assoc. Prof. Dr. Seppo Carrila and the Publishing Clinic for assistance in preparing the manuscript. Finally, I would like to thank the personnel of the Faculty of Science and Industrial Technology who assisted in various matters that made this thesis work flawless and complete.

Kachasak Intim

CONTENT

Chapter 1 Introduction.....	1
1.1 Introduction	1
1.2 Objectives.....	2
1.3 Outcomes	2
1.4 Scope of the Research.....	2
1.5 Materials and tools.....	3
1.6 Research Methodology	3
Chapter 2 Literature Review.....	7
2.1 Computer Vision.....	7
2.2 Convolution Neural Networks (CNN).....	8
2.3 Study on face detection.....	12
2.4 Studies on facial recognition	25
2.5 Hardware and software equipment used in the operation of the system.....	35
2.6 Related Works.....	42
Chapter 3 Research and Methodology	50
3.1 Research and Methodology	50
3.2 Development of mask detection system	52
3.3 Face recognition system while wearing a mask	56
Chapter 4 Result and Discussion.....	62
4.1 Training the face mask detection model.....	62
4.2 Training a face recognition model.....	64
4.3 Practical testing	64
4.4 Sending data to show results to users.....	72
Chapter 5 Conclusion and Future Work.....	74
5.1 Conclusion.....	74
5.2 Future work.....	74

LIST OF TABLES

Table 1 The performance of the architecture used in facial recognition.....	35
Table 2 Hardware Features.	38
Table 3 Software Features.....	42
Table 4 Summary of the models used in related works.	46
Table 5 Summary of the equipments used in related works.	48
Table 6 Model training settings for SSD.....	55
Table 7 Model training settings for FaceNet.....	61
Table 8 Evaluation table of actual use performance from all 76 trial participants.....	65

LIST OF FIGURES

Figure 1. Research Methodology	3
Figure 2. Research Design.....	5
Figure 3. System Development	6
Figure 4. 3x3 filters	8
Figure 5. The picture shows how the filter works while converting image data.....	9
Figure 6. Image Padding.....	10
Figure 7. Pooling is done to reduce redundant data	10
Figure 8. Flattening data.....	11
Figure 9. Operation of the Fully Connected layer.....	11
Figure 10. Convolution Neural Networks (CNN).....	12
Figure 11. SSD architecture.....	13
Figure 12. The architecture of SSD.....	13
Figure 13. Resizing to different scales to create an image pyramid.....	14
Figure 14. Work of Proposal-Network (P-Net) layer	15
Figure 15. Work of Refine-Network (R-Net) layer	15
Figure 16. Work of Output-Network (O-Net) layer	16
Figure 17. Haar-like features	17
Figure 18. Imaging the image integral.....	17
Figure 19. Cascading Classifiers	18
Figure 20. Demonstrate the work of the Haar Cascade.....	19
Figure 21. Reducing the size of the image for the use of HOG.....	20
Figure 22. HOG gradient filter.....	20
Figure 23. effect of gradient	20
Figure 24. Gradient using directional arrows.	21
Figure 25. How to create a histogram of in-cell gradients	22
Figure 26. How to create a histogram of in-cell gradients in case of more than 160 degrees direction	22
Figure 27. Result of HOG	23
Figure 28. Example of finding IoU (intersection over union)	25
Figure 29. Threshold configuration	25
Figure 30. Workflow of FaceNet.....	26
Figure 31. VGG architecture	28
Figure 32. Structure of VGG	29
Figure 33. The architecture of Inception-Resnet-v2.....	30
Figure 34. Operation of Inception-Resnet-v2.....	31

LIST OF FIGURES (CONT.)

Figure 35. Abbreviated implementation of Inception-Resnet-v2.....	31
Figure 36. Resnet Block	32
Figure 37. Bottleneck building block	32
Figure 38. Bottleneck building block for any layer.....	33
Figure 39. Model Scaling in different ways.....	34
Figure 40. NVIDIA Jetson Xavier NX microcontroller board	36
Figure 41. Thermal camera MLX90614ESF Infrared camera	36
Figure 42. Camara Webcam 4k	37
Figure 43. Ultrasonic Sensor HC-SR04	37
Figure 44. Arduino board.....	38
Figure 45. Compare architectures on ImageNet.....	45
Figure 46. System Overview	50
Figure 47. System software and hardware	51
Figure 48. The prototype designs.....	51
Figure 49. Equipment connection for actual use.....	52
Figure 50. Real-World Masked Face Dataset (RMFD) database example	53
Figure 51. Cropping faces in Label image program.....	53
Figure 52. .xml file result of making Label image	54
Figure 53. Training operation of the face mask detection model.....	55
Figure 54. Coddng of Face Detection	55
Figure 55. Using Google from to collect data.....	56
Figure 56. Different angles of the face were used to collect data	57
Figure 57. Dataset Personnel of PSU.Wittayanusorn Surat Thani School	57
Figure 58. Result of Face Alignment work.....	58
Figure 59. Face masks for use in adding image data.....	58
Figure 60. Operation of adding a mask to a non-masked face	58
Figure 61. Result of adding masks.....	59
Figure 62. Training a face recognition model.....	59
Figure 63. Operation of the Euclidean distance function in the triplet loss function..	60
Figure 64. Coddng of Face Recognition.....	60
Figure 65. Block Diagram of Face Detection and Face Recognition System.	61
Figure 66. Showing the results of training a face detection model	62
Figure 67. Confusion matrix of the face detection model.....	63
Figure 68. Face detection model training test results	63
Figure 69. Results of facial recognition training	64

LIST OF FIGURES (CONT.)

Figure 70. Comparison of training and testing data..... 68
Figure 71. Real use testing..... 68
Figure 72. The system can detect masks and recognize faces correctly..... 69
Figure 73. System error when detecting incomplete face angles 70
Figure 74. The system cannot recognize faces 71
Figure 75. Test temperature detection 72
Figure 76. Sending data to the LINE application 73

Chapter 1

Introduction

1.1 Introduction

Current technologies are rapidly evolving and are used in many daily lives. One of the most important technologies in our lives is face recognition technology also known as face scanning, this technology is widely used in security systems with face detection and face recognition such as recognizing a person's face at an airport or face detection through entrances and exits of shopping malls or used to find criminals, follow suspects through police surveillance cameras and also used to analyze the faces of employees in the company. While working to assess satisfaction while working or used to record the time of entering and leaving work of employees in the company, etc. We can see that the face scanning system can be applied to various businesses and can be convenient for users as well. It can be concluded that this face-scanning system has come to change and develop our daily lives to the next level.

Currently, there is a spread of the COVID-19 virus that plays a role in changing the well-being of people in society, causing people in society to change the way they live their daily lives to avoid touching or touching things around them less and still need to maintain social distancing. Wearing a face mask while participating in social activities reduces the spread of the COVID-19 virus, and taking preventative measures against COVID-19 alone may not be enough to reduce the spread of the disease. Therefore, it is necessary to develop technology or apply existing technology to adapt to the current situation. The technology that is adapted to the current situation, such as speech recognition technology, face recognition technology, and the use of digital money these technologies can greatly change the daily lives of people in society and also reduce contact while doing various activities.

In this research, we are interested in implementing a model for face detection and facial recognition models applied to human faces while wearing masks and testing whether the model is used to detect faces and which model can be processed with the fastest and best accuracy to be developed into a face scanning system and temperature measurement in various places, along with checking the wearing of a mask and recognizing the faces of different people. It can also result in face comparisons and temperature detection on the user's phone and in order to be

a part of reducing social distance and reducing the spread of the current Covid-19 virus.

Keywords: face detection; neural networks; COVID-19; face recognition

1.2 Objectives

1. To analyze the factors of various methods used to develop the face scanning system to find out which model is the most suitable.
2. To practice measuring with an infrared camera in the face scanning system in the situation of COVID-19.
3. To develop a face scanning system that can send the results of work to the user's phone.

1.3 Outcomes

1. Enable us to get a way to process images of face scanners.
2. Enable us to get a system that can measure temperature.
3. Enable us to get show the results directly to the users.

1.4 Scope of the Research

This research covers the study of the best model to design a face recognition system and match faces through masks and temperature measurements including designing a system that will integrate face detection and face recognition functions and temperature detection. The results obtained from this study. A system that can detect masks and face recognition systems and systems detect body temperature while detecting faces of Personnel of PSU. Wittayanusorn Surat Thani School.

1. The models used for face detection are as follows:
 - 1.1 Haar Cascade
 - 1.2 Single Shot Multibox Detector (SSD)
 - 1.3 Histogram of Oriented Gradients (HOG)
 - 1.4 Multi-task Cascaded Convolutional Networks (MTCNN).
2. The architecture used for face recognition are as follows:
 - 2.1 visual geometry group (VGG)
 - 2.2 Inception Resnet V2
 - 2.3 ResNet50
 - 2.4 EfficientNet.

1.5 Materials and tools

Hardware

- Jetson Xavier NX
- MLX90640 IR Array Thermal Imaging Camera
- Camara Webcam 4k

Software

- Google Colab
- Jetpack Version 4.6
- Python 3.6.9
- Tensorflow-gpu 1.13
- Opencv 3.4.2
- Matplotlib 3.1.1
- NumPy 1.16.6

1.6 Research Methodology

We have a process that studies the system analysis and design process by the System Development Life Cycle (SDLC) by prototyping method as shown in Figure 1.

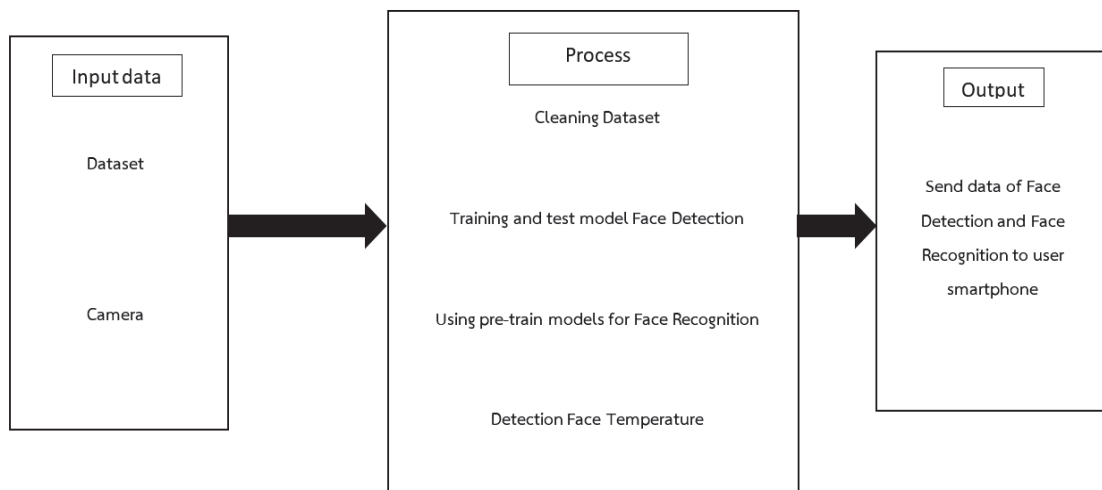


Figure 1. Research Methodology.

For the research process to obtain guidelines for researching and building this prototype system, consists of the following processes.

1.6.1 Literature Review

In this research, we explore the current spread of the COVID-19 virus and the importance of preventing its spread. The people wearing a mask is an effective measure to prevent the initial spread of the virus. However, there is a common problem where many people are not being socially responsible by not wearing a mask. To address this issue, we aim to develop a system that can detect individuals who are not wearing masks and report them to various parts of the organization.

In the first part of our study, we investigate the research on detecting faces in the environment and the factors that can reduce the performance of face detection. Yu et al. (2020) conducted a study and found that facial recognition systems often detect objects that are not faces, such as shapes that resemble a face or faces that are too small or unclear. This is due to the complex model training conditions, which can sometimes lead to false detections. However, deep learning methods often ignore these outliers during the training phase, and the models make unreasonable decisions on the above imperfect images. In conclusion, the incomplete face image is the primary obstacle to training a model for facial recognition.

In another study that deals with similar issues, Wenjing et al. (2023) found that there can be errors in predicting the position of the face if the area is underexposed. This is because the difference in lighting between light and dark areas can be too large and complex to detect at both the subject and pixel levels.

Regarding mask detection, Martinez-Diaz et al. (2020) proposed three facial recognition schemes for the accurate and efficient identification of masked faces. These schemes consider both the masked face and the area around the eyes. When comparing the recognition performance of three small models suitable for smartphones, the results showed that all three models have similar performance, with a difference of only $\pm 1\%$. This indicates that the current technology is sufficient for the system requirements that we are about to develop.

1.6.2 Formulate a hypothesis

Currently, face detection and face recognition have high processing accuracy, but when we encounter problems related to the spread of the COVID-19 virus makes it is necessary for people to wear masks to protect themselves and others from the

COVID-19 virus. Therefore, there will be a problem with the face detection system and facial recognition cannot obtain full face information about a person's face because the face is obscured by the mask, and adding a temperature detection function can increase the efficiency of the system. We can observe that from the data studied facial processing while wearing a mask difficulty recognizing faces and from the study of face detection, it was found that the case that causes repeated errors is the effect of too much or too little light when detecting faces causing the prediction of the position of the face to be inaccurate, and in measuring the temperature, it is very necessary to prevent the spread of the COVID-19 virus.

1.6.3 Research Design

We train the models to learn the face wearing the mask requires the data used in the training: image data containing Wearing a face mask correctly, incorrectly, and not wearing a face mask. After that, the image data of the testers will be collected the test participants are Personnel of PSU. Wittayanusorn Surat Thani School will give participants a test to take pictures of their own face at different angles specified by the researcher, such as a straight face, 90-degree left-right face, and slanted face and faces in different corners wearing masks and not wearing them, etc. There will be a way to collect data via Google so that each participant's picture file can be divided into the same folder as the participant's name. Participate in the experiment correctly and will use the face data obtained from the test participants to recognize the face of the system and add a temperature detection function in the system then send different results to the smartphone of test participants.

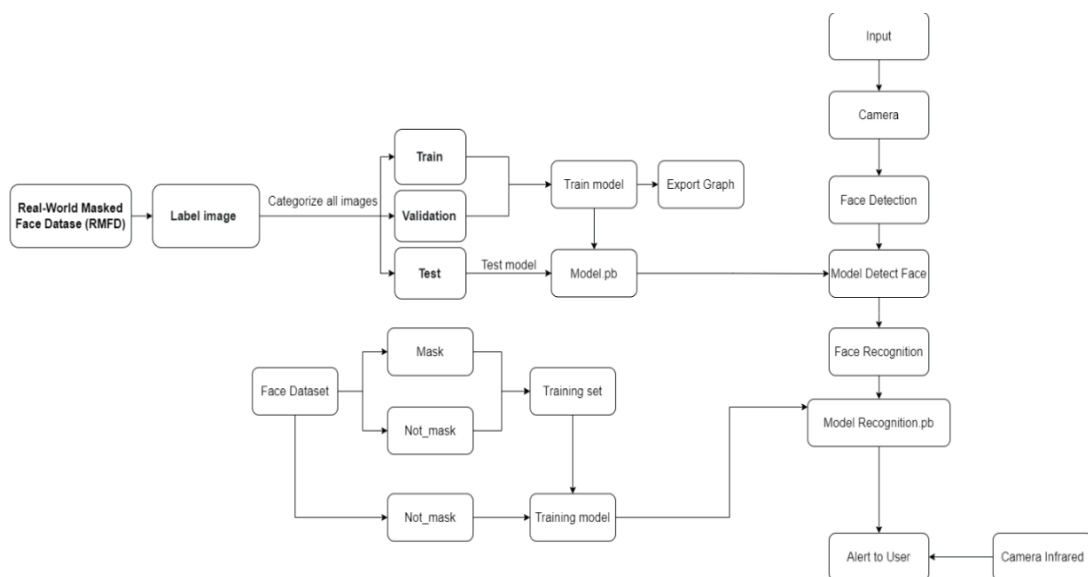


Figure 2. Research Design.

1.6.4 System development

Developed using Jetson Xavier NX as a microcontroller board. Connect to infrared cameras and video cameras and using the Ubuntu 18.04 operating system, after detecting a face from a user, the system sends detection data to the LINE application.

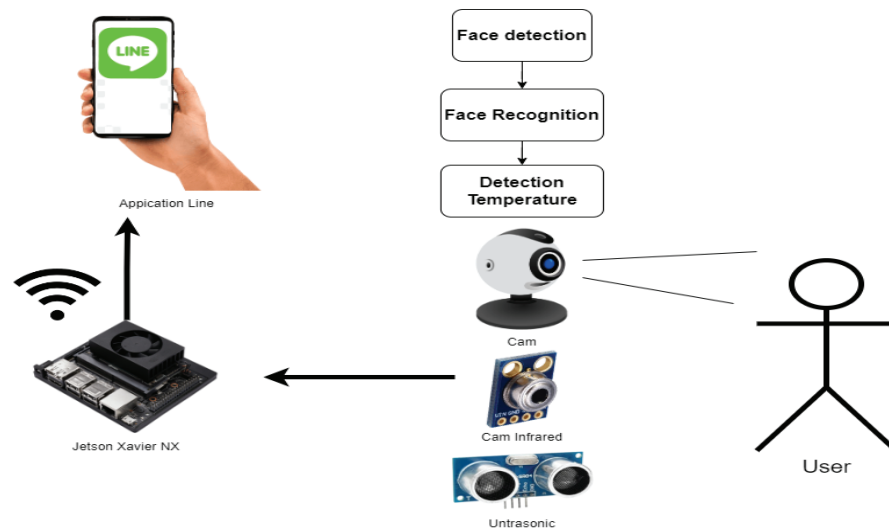


Figure 3. System Development.

1.6.5 System Analysis and data interpretation

The validation dataset serves to validate the model which image data in the audit data set were images that the model had never encountered before during the training phase. We test to see that if the model could detect masked faces, it will be adapted and in the face recognition process. This is necessary to rely on pictures from real users both wearing and not wearing masks to train the face recognition model as a pre-trained model. Therefore, we thought it would be a good idea to increase the accuracy by retraining the model with photographs of trial participants and finally combined with temperature detection.

1.6.6 Thesis Preparation, defense, and Improvement

We summarize the factors used in face detection and face recognition to determine which model provides the best accuracy and processing speed that is most suitable for the application. Then add a temperature measurement system and send the results of the work to the user and prepare, defend, and improve the thesis documentation.

Chapter 2

Literature Review

In this chapter, we describe the research and data we are interested in studying and implementing a number of face detection and recognition systems. Including various technologies, software, and hardware used in the experiment.

2.1 Computer Vision

If comparing computers with humans, Computer Vision is like the “eyes” of the computer, which receive data types such as pictures, videos, or digital media. From the point of view of engineers, the computer is trying to work or understand humans. Vision in the understanding of the computer is a mathematical understanding, known as Convolution Neural Networks (CNN), using arithmetic and statistical processes.

Computer Vision (Stockman et al.,2001) is a sub-field of Artificial Intelligence (AI), specifically if AI allows machines to think for themselves. Computer Vision is what allows machines to see. Which technically enables machines to recognize, understand and respond to visual information the main functions of computer vision are detection and bounding locating and recognizing objects from images such as faces evaluation for dividing the characteristics of an object in an image or comparing objects in different of the image, linking objects different of an image to create a three-dimensional model, those models may be used to prototype AI robots and to find those objects with images requires a large database to train the computer to recognize that object.

Computer vision technology works on the same principle as jigsaw puzzles. In a puzzle or jigsaw puzzle which will have various parts scattered these parts must be put together to form a complete picture. This type of splicing is similar to neural networking for computer vision, in which a computer deconstructs an image and then stitches it together so that the computer can understand the image this work consists of many steps such as data filtering. By working through a multi-layer Deep Neural Network (DNN) to find connections between the fragments of the image in the same style as you would in a jigsaw puzzle.

However, the computer will not get a "solution" of the resulting image like what we see on a box of jigsaw puzzles. But training the system to recognize things requires feeding hundreds or thousands of images until the system can identify the

target object. Instead of teaching the system to look for a cat's whiskers, tail, and ears, the programmer feeds it millions of images of cats for the system to learn. Eventually, the computer learns to look for the features of the feline's appearance on its own.

2.2 Convolution Neural Networks (CNN)

In image processing to compare images, multiple layers of processing are required to make the information more accurate. Image processing is the biometric method of identification. This can now increase the accuracy up to $97.35\% \pm 0.25\%$, while humans have an accuracy of 97.53% , which can be done very close to humans and have a preliminary method called Convolution Neural Networks (CNN) (Albawi, Saad et al., 2017). This is a network or neural prosthesis that simulates human image recognition. In general, images are taken through the process of extracting the properties of the images and comparing them by converting the image data into the form of numbers. In order for the computer to be able to further process the data, which is divided into 2 layers, the first layer is called the feature learning layer, which is extracting the properties of the image to pass on to the next floor is a classification which compares the imported images with those in the database. The first step of feature learning is featuring extraction or convolution, which is to extract the properties of the image by calculating the values in the filter, which is like a device that helps extract the features of the object from the images original, such as lines and contrasting colors will make it more pronounced filters that help extract features used in object recognition. Usually, one filter can extract only one feature of interest. We need to use multiple filters to find a combination of spatial features such as eyes, ears, nose, mouth, etc.

1	-1	-1
-1	1	-1
-1	-1	1

Figure 4. 3x3 filters.

From Figure 4, the position in the middle with the blue frame is the Anchor that is placed on the Pixel of the input image. When the filter is applied to an image in a pixel and to the first pixel of the image, it converts the data into the feature map, and then the filter is transferred over the other pixels in the image one pixel at

a time until every pixel in the image is completed. An example of a Feature map working is shown in Figure 5.

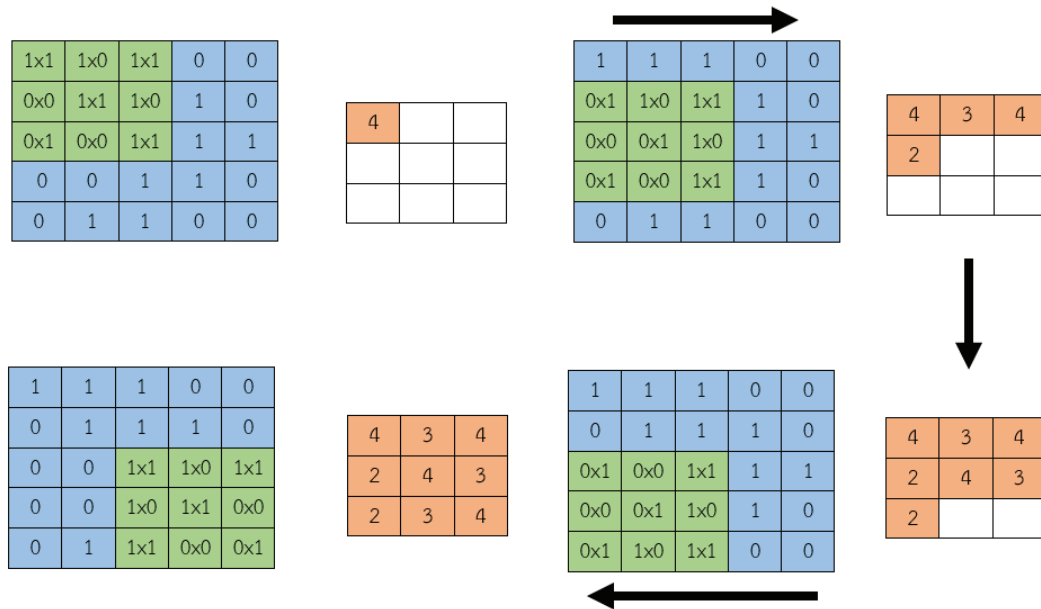


Figure 5. The picture shows how the filter works while converting image data.

Usually, a kernel filter extracts one feature of interest. We need to use multiple filters to find several spatial features combined and converts image data into two-dimensional or $N*N$ forms, such as the edges or curves of a face. In addition to tracing the filter to extract the properties of the image convolution is also improved, the stride is improved, and the stride's filter padding determines how much we move the filter by step. for example, Figure 5, the stride is set to 1. Increasing stride will cause the property of images to have less overlapping space and more stride configurations allow us to get a smaller feature map. This is another way to increase the accuracy of the CNN architecture and padding. From Figure 6, we can see that the pixel at the edge of the blue color is never centered filter when it's over because we can't extend the filter beyond the edge of the image, the resulting feature map is smaller than the Input Image. Therefore, in order to make the feature map equal to the Input Image and center the pixels at the edge of the image filter. When taped we will need to do some padding by padding the border by adding 0 (Zero Padding) around the original image because in some problems the import image along the edge of the image may be important to some decisions. We need to keep the features along the edges of the image as well.

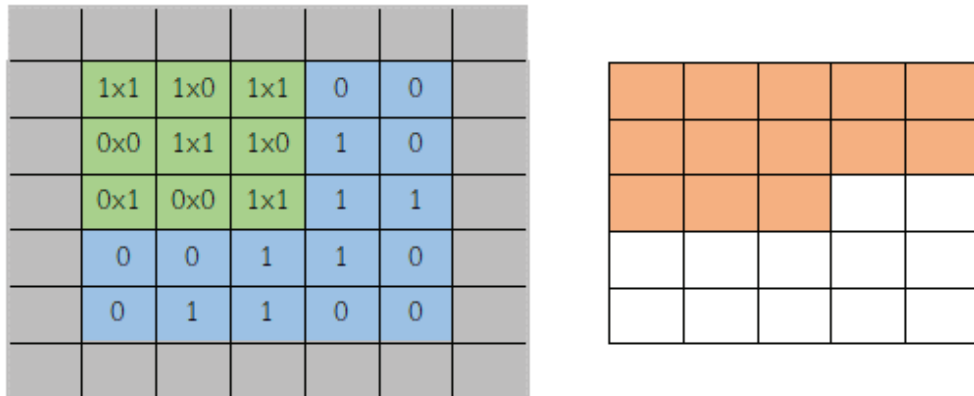


Figure 6. Image Padding.

Combining those properties in order to shrink them before computation in the next layer is a minimization called Pooling to reduce the redundancy of data or to reduce the size of the data but still retain the prototype of the original data. It selects a representation of the image by maximizing the value or the average from the pixels in the Feature Map according to the specified size, such as 2x2 size which will reduce the image size by half as shown in Figure 7. This pooling is very useful in reducing the size of the model as the use of multiple convolutions increases the complexity of the data, pooling helps in this respect.

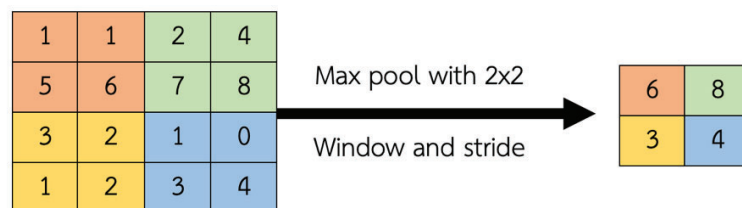


Figure 7. Pooling is done to reduce redundant data.

When the function of the feature learning layer is completed, it will be the next function of the layer classification. Initially, we take the properties obtained from pooling, and after the last convolution, we get a feature map, then convert the data from the $N \times N$ matrix to $1 \times N$ called flattening and after flattening shown in Figure 8.

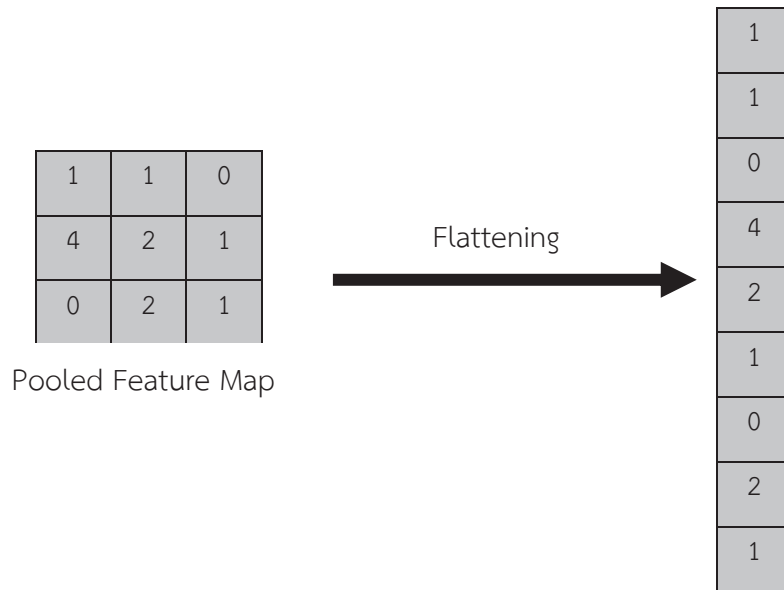


Figure 8. Flattening data.

To make it easier to compare the data with the database to predict the object the Fully Connected layer is a layer that uses the neural Khan framework to bring the data. This is divided into 3 work phases: the input layer, Fully-connected layer, and the output layer. When flattening is completed, we will get the data in the Input layer, then the next step is to bring the data to find the relationship in the connected layer of each node of the connection layer like the ears, eyes, nose, and mouth of dogs and cats. The model will try to connect the relationships from these data to predict that the input image is closest to what is shown and shown in the Output layer when it is most accurate from Figure 9.

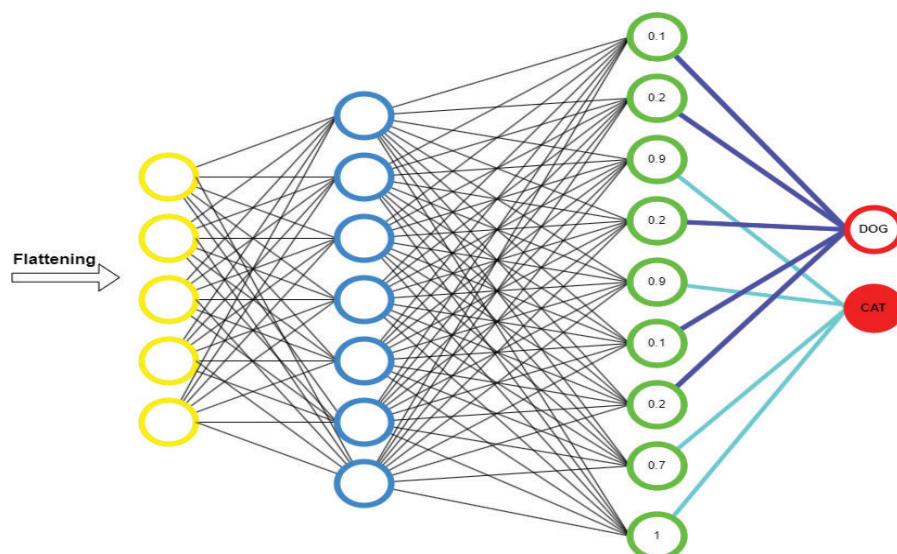


Figure 9. Operation of the Fully Connected layer.

The last layer of work is the Softmax layer, which in this layer performs the function of giving the output a probability to be calculated in the process classification as shown in Figure 10.

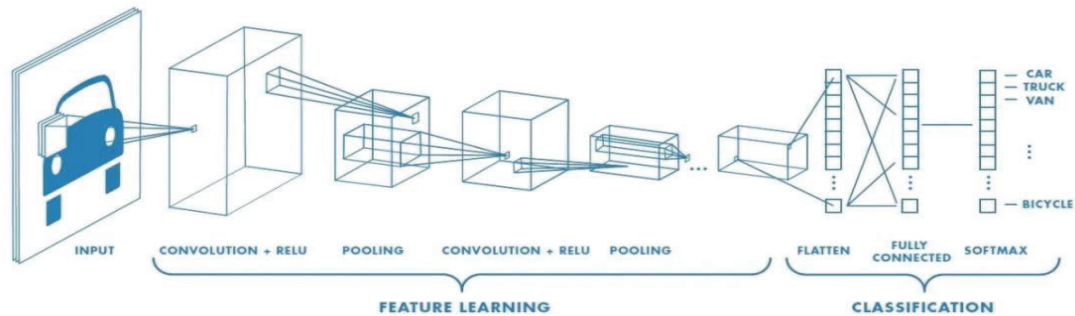


Figure 10. Convolution Neural Networks (CNN).

(<https://shorturl.asia/MO6xu>)

2.3 Study on face detection

The earliest developments of face scanning systems were face detection and the popular computer image capture methods were Haar Cascade (Soo, Sander, 2014), Single Shot Multibox Detector (SSD) (Liu et al., 2016), Histogram of Oriented Gradients (HOG) (Dalal et al., 2005) and Multi-task Cascaded Convolutional Networks (MTCNN) (Zhang et al., 2016) models. Each of the above methods has different face detection speeds and face detection accuracy. Processing at 1.54 frames per second, but MTCNN has higher accuracy than SSD. However, it must be selected according to the task used, for example, if the accuracy of face detection is not required and the face detection speed is required. If using an SSD, you want to use high identification accuracy and do not need a lot of processing speed, use MTCNN.

2.3.1 Single Shot Multibox Detector (SSD)

SSD is an algorithm used to detect objects in real time. Developed from R-CNN, the name implies:

- Single Shot: taking one shot and detecting objects and includes object classification All processing can be performed in a single operation.
- Multi-Box: is a namely technique for bounding box regression developed by Szegedy et al.
- Detector: The network detects objects and classifies detected objects.

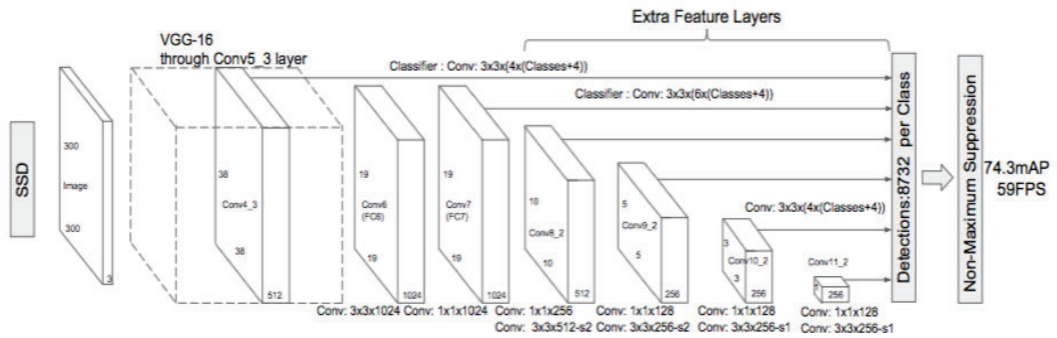


Figure 11. SSD architecture.

The architecture of the SSD uses the VGG-16 architecture, but does not use a fully connected layer, the reason VGG-16 is used is the basic structure of the image processing. and popular for problems where learning transfer improves outcomes instead of the fully connected layers of the traditional VGG onwards. set of layers. The convolutional has been added to the conv6 from Figure 11. It allows feature separation at multiple levels and gradually reduces the size of the input in each subsequent layer until it is called a feature layer. In total, the SSD makes 8,732 predictions using the six layers and all the convolution results go to the final object detection and the detection of the SSD.

MultiBox of SSD

The SSD bounding box regression technique is a fast, class-independent method for generating box bounding coordinates using initialization-based loop network. This technique uses the 1x1 convolution, which shrinks as the number of dimensions decreases, while the "width" and "height" remain the same.

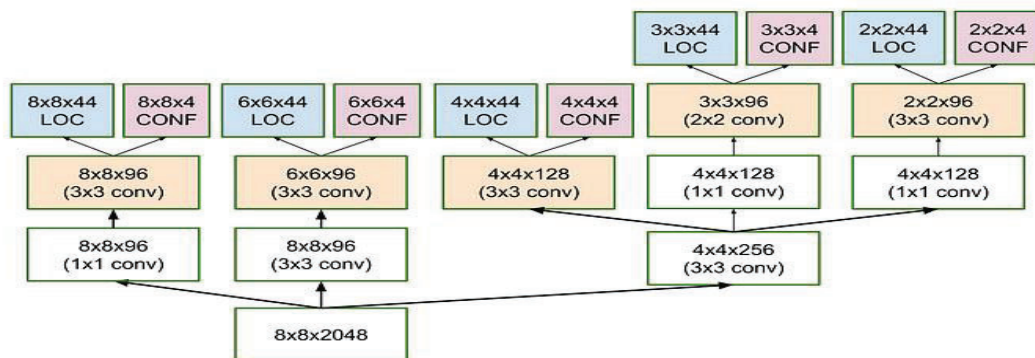


Figure 12. The architecture of SSD.

In SSDs, the MultiBox's loss function has been implemented and includes two critical areas (Eddie Forson, 2017)

- Confidence Loss: This measures the network's confidence in detecting an object in a bounding box for which the loss function is calculated.
- Location Loss: Measures the distance between the network's predicted boundary box and the actual boundary box in the training dataset using the L2-Norm to calculate the loss function.

The formula for the loss function, which measures the prediction distance, is:

$$\text{Multibox}_{loss} = \text{Confidence}_{loss} + \text{Alpha} * \text{Location}_{loss} \quad (1)$$

2.3.2 Multi-task Cascaded Convolutional Networks (MTCNN)

MTCNN or Multi-task Cascaded Convolutional Networks (MTCNN) is a framework developed to work for face detection and face positioning. This three-step process is capable of recognizing the face and location of landmarks such as the eyes, nose, and mouth. MTCNN uses multitasking learning in the early stages, and CNN was used to quickly generate selector windows. In the second step finally, in the third step, a more complex CNN is used to further refine the results and display the location of landmarks on the face. MTCNN's initial work was to take images and resize them to different scales to create an image pyramid. Which is the network input shown in Figure 13.

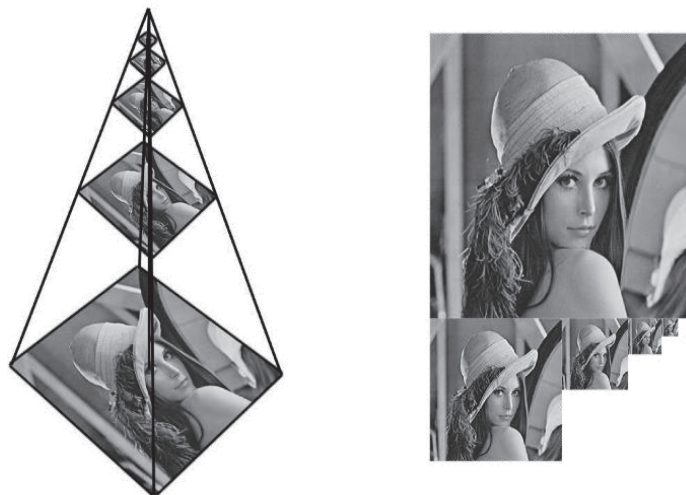


Figure 13. Resizing to different scales to create an image pyramid.

(<https://huytranvan2010.github.io/Architecture-MTCNN/>)

After going through the reconstruction process, the next step is to bring the images obtained through the CNN process. The first layer, the Proposal-Network (P-

Net) layer is the first model, which has the least number of filters and layer depth, known as overall, it is the least complex out of all 3 models, whose function is exactly as the name suggests, that is, it is responsible for finding the bounding box of the page in the image roughly and send the results to the next level to resize the image 12 times. This has to be done because the P-Net itself has very few CNN filters, which are known to CNNs that cannot capture the same image of different sizes. Therefore, use the method to reduce the image size and run it several times instead, and when it finds the Bounding Box of the man page in any image size, we then calculate back to the position in the actual size image. The output of the P-Net consists of many bounding boxes, we select only the bounding boxes as well. The probability of having more faces than the threshold and Use Non-Maximum Suppression (NMS) gives us only unique bounding boxes shown in Figure 14.

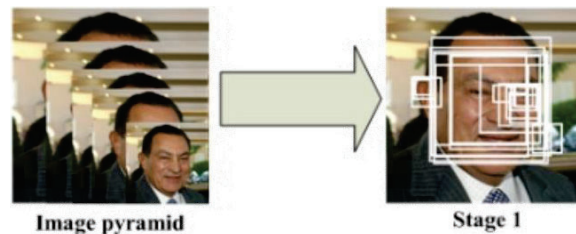


Figure 14. Work of Proposal-Network (P-Net) layer.

(<https://vincentblog.xyz/posts/an-overview-of-face-detection>)

The next layer, the Refine-Network (R-Net), is a more complex CNN model than the P-Net in that it takes the images in the bounding box derived from the P-Net as input one by one. The R-Net will loop one bounding box at a time, then it will find the Bounding Box of the pages in those images again so we can get a more accurate bounding box. Then it is selected only the bounding boxes that have the probability of having pages exceeding the specified threshold and NMS to eliminate duplicate bounding boxes as in P-Net shown in Figure 15.

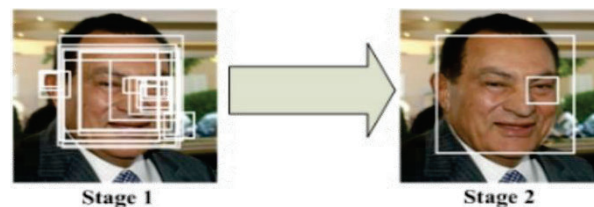


Figure 15. Work of Refine-Network (R-Net) layer.

(<https://vincentblog.xyz/posts/an-overview-of-face-detection>)

The last layer is the output network (O-Net) is the most complex neural network, the O-Net uses the output of the R-Net to find the bounding box and face recurrence probabilities. The process is similar to that in R-Net, but more special than in O-Net, it will find the Face Alignment of each face found as well. Face Alignment consists of 5 positions: two eyes, a nose, and a left edge. the right side of the lip is shown in Figure 16.

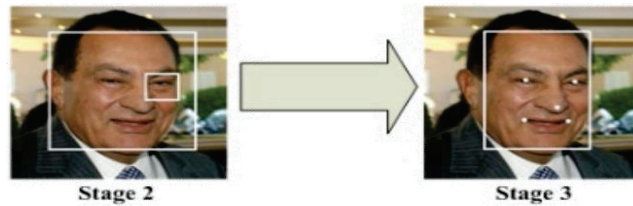


Figure 16. Work of Output-Network (O-Net) layer.

(<https://vincentblog.xyz/posts/an-overview-of-face-detection>)

Notice that the output layer of R-Net and O-Net is fully connected layers, but in P-Net is a fully convolutional neural network because we want P-Net to be able to accept any image size input in receiving images but when enabled, CNN only resizes images to 12x12.

2.3.3 Haar Cascade

in this algorithm Face images are assigned positive and non-face images are assigned negative. In order to train the classifier by calculating the Haar feature, it starts by converting the image to grayscale. This will make the calculations easier. Basically, there are three types of edge features of Haar-like features. line properties and features four rectangles. White bars represent pixels with different segments. of the image closer to the light source, thus "whitening" in grayscale images. The black bar is opposite. These are pixels whose image features are further away from the light source or obscured by other objects. This will make that area "blacker" in the grayscale image. The comparison between white and black pixels is the most important reason we convert images to grayscale as shown in Figure 17.

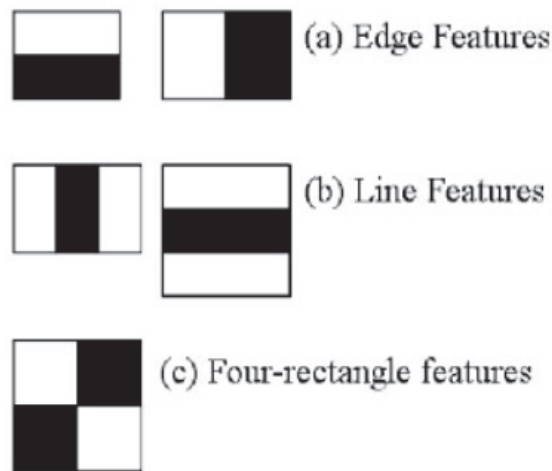


Figure 17. Haar-like features.

(<https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8>)

Integral Imaging is because we have to use Haar-like features in all possible image sizes and positions. This will ultimately result in a large number of feature maps to compute at once. The problem with calculating the new Haar feature is that we have to compute the mean of a given region multiple times, and the time complexity of these operations. We can use the integral method to achieve reduced running time. Pixels in an Integral image are defined as the sum of all pixels on the left and all pixels on the top shown in integral as shown in Figure 18.

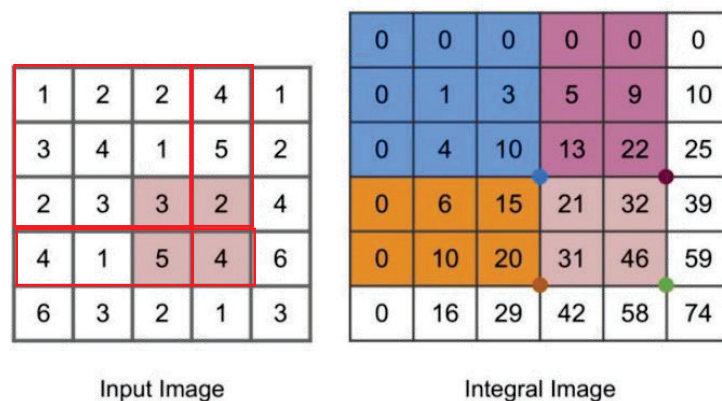


Figure 18. Imaging the image integral.

(<https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8>)

The work is from the input image in the pink frame by bringing all the pixels in the red box add up to be equal to the pink boxed pixels of the integral image, and in the working part of Adaboost training, Adaboost essentially selects the best features and trains the classifier to use it. Use a combination of weak classifiers to create a strong classifier that algorithms can use to detect objects the weak classifier is created by moving the red frame over the input image and calculating the Haar feature for each image subset. This difference was compared with the learned criteria which can distinguish non-objects in the image from objects because these are weak classifiers, many Haar features are required for accurate builds. A strong classifier is also known as Cascading Classifiers as shown in Figure 19.

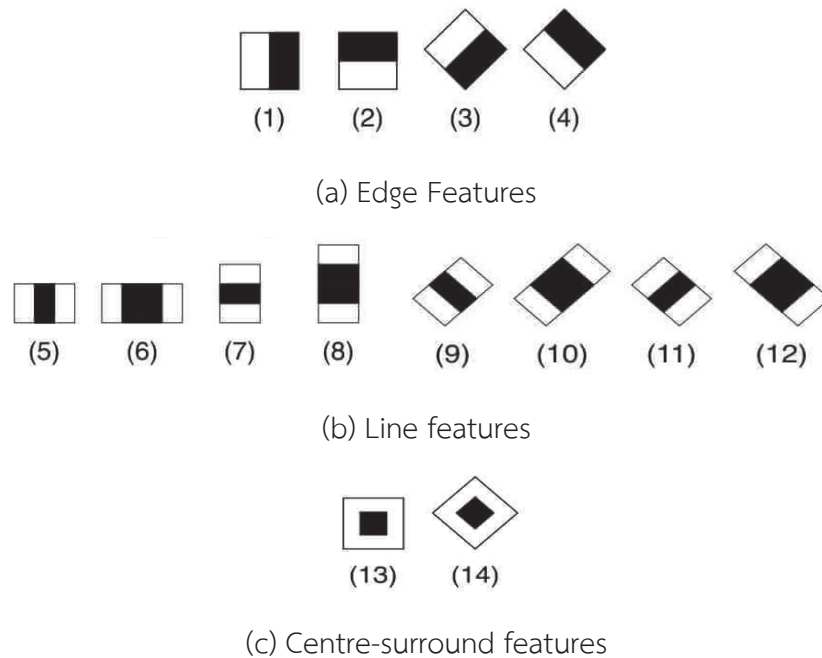


Figure 19. Cascading Classifiers. (<https://playelek.com/haar-cascade-create/>)



Figure 20. Demonstrate the work of the Haar Cascade.

(<https://cloudacademy.com/blog/google-vision-api-image-analysis/>)

To begin with, the Haar Cascade uses a red box to detect objects in an image. The image has all three Haar-like features overlaid on the image for all possible sizes. When work in the same area is finished, move the box to the next area and keep looping until you find objects that are different from the background of the image. When the loop completes, the image will increase the box size to find larger objects in the image. In other words, red boxes that can detect faces are considered weak classifiers. The Haar Cascade operation is to find the point where the most faces of the boxes are detected and generate a strong classifier.

2.3.4 Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients, also known as HOG, is used in computer vision and image processing for object detection purposes. This technique counts the occurrence of gradient misalignment in the translated part of the image, focusing on the structure or shape of the object. It is better than detecting any edge or corner because it uses the size and angle of the gradient to calculate the feature for the image area. The histogram is generated using the size and direction of the gradient. As a first step, the HOG description used for object detection is calculated from a 64x128 image. This solves the problem of images that may have multiple scales. Generally, images with multiple scales are analyzed at multiple image locations. The only limitation is that the image being analyzed has a fixed aspect ratio. In our case, the image must have an aspect ratio of 1:2. For example, it could be 100x200,

128×256, or 1000×2000. The large size was 720×475 and 100×200 was selected to reduce the size to calculate the HOG descriptor, so it was resized to 64×128 image as shown in Figure 21.

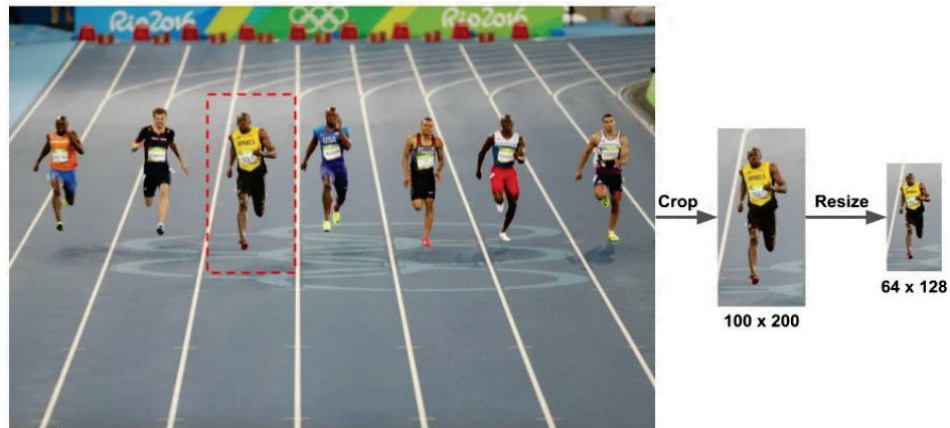


Figure 21. Reducing the size of the image for the use of HOG.
(<http://www.codestudyblog.com/cnb/0421120121.html>)

After that, we calculate the gradient Images calculate. We need to calculate the horizontal and vertical gradients first. And finally want to calculate the histogram of the gradient. This can be easily done by filtering the image as shown in Figure 22, resulting in the same results as Figure 23.



Figure 22. HOG gradient filter.



Figure 23. effect of gradient.

(<http://www.codestudyblog.com/cnb/0421120121.html>)

The image shows color level chasing that deletes a large amount of unnecessary data, such as the background of the image. However, it emphasizes the structure of the image from the image, it can be seen that chasing will increase the lines of the object in the image next is the calculation of the histogram of the color level chasing. At this step, the image will be divided into 8x8 cells, and the histogram of the color level chasing will be calculated for each 8x8 cell. The histogram will be defined as a vector or an array of 9 slots, which are numbers related to the angles 0, 20, 40, 60, 80, 100, 120, 140, and 160, representing the angles between 0 to 180 degrees instead of 0 to 360 degrees because the chasing and negative values will be represented by the same number, which is the arrow chasing level and another 180 degrees opposite to that arrow, which are considered the same.

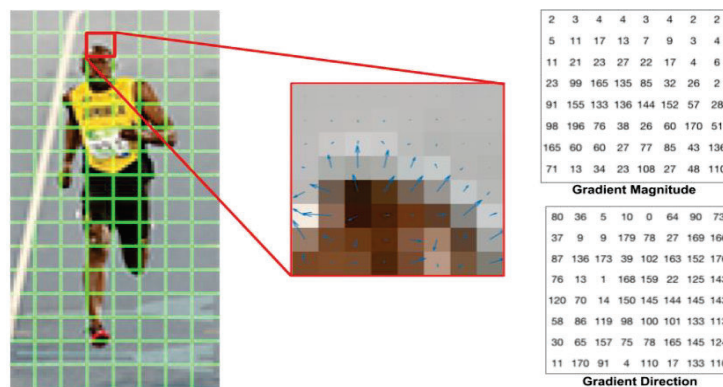


Figure 24. Gradient using directional arrows.

(<http://www.codestudyblog.com/cnb/0421120121.html>)

Notice that the arrows show the direction of the gradient and their lengths show the magnitude it can be observed that the direction of the arrow points to the direction of the change in intensity and the size shows how much they differ, the next step is to create a histogram of the gradients in these 8x8 cells. The histogram consists of 9 channels corresponding to angles 0, 20, 40, ... ,160. Figure 25 shows the process We are looking at the size and direction of the gradient of the same 8x8 patch as in Figure will be selected according to the size the pixel circled in blue has an angle (direction) of 80 degrees and a size of 2, so put a value in the 80-degree box. Next, the pixel circled in red has an angle of 10 degrees and a size of 4, since 10 degrees is halfway between 0 and 20. Vote with pixels split into two equal boxes.

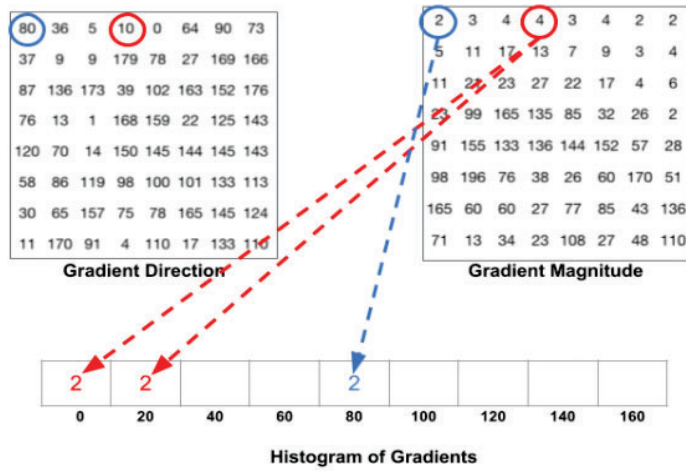


Figure 25. How to create a histogram of in-cell gradients.
 (<http://www.codestudyblog.com/cnb/0421120121.html>)

If the angle is greater than 160 degrees, it is between 160 and 180, and we can assume that the angle wraps around to equal 0 to 180. In Figure 26, pixels with an angle of 165 degrees are proportional to the 0-degree channel, and the channel size is 160 degrees, the size is divided into two equal channels.

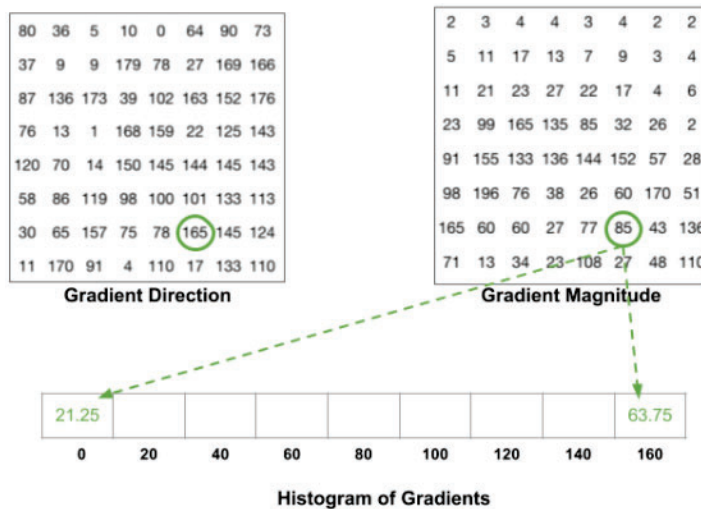


Figure 26. How to create a histogram of in-cell gradients in case of more than 160 degrees direction.
 (<http://www.codestudyblog.com/cnb/0421120121.html>)

Going through the entire process, the HOG operation can often be visualized by plotting the 9x1 normalized histogram in an 8x8 cell. Looking at the side image, it is noticed that the direction in which the changes can be seen in the histogram captures the shape of the person in the region of the torso and legs as shown in Figure 27.

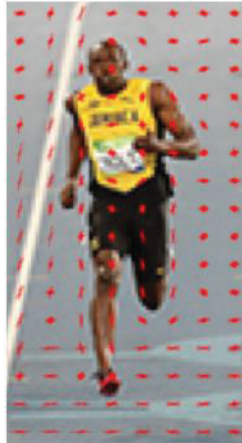


Figure 27. Result of HOG.

(<http://www.codestudyblog.com/cnb/0421120121.html>)

According to various research studies, SSD performance is suitable for improving system efficiency. Both detection accuracy, detection speed, and face detection at various angles of the face. Therefore, the most suitable algorithm for developing a face detection system is SSD.

The results of the model training are presented in precision, recall and AP (Average precision) values, known as the confusion matrix, which are used to evaluate performance and to measure machine learning's ability to classify and learn performance. The detection includes:

True Positive (TP) of what the model predicts and the answer is "True" and "True", respectively,

True Negative (TN) of what the model predicts and the answer is "True" and "False", respectively,

False Positive (FP) of what the model predicts and the answer is "False" and "True", respectively, and

False Negative (FN) of what the model predicts and the answer is "False" and "False", respectively.

Precision is a measure of how accurately our model can predict. or the rate of guessing correctly all guesses can be obtained from Equation (2), which is dividing the number that the model predicted by (TP) by the total model prediction.

$$\frac{TP}{TP+FP} \quad (2)$$

The recall is the number of correct guesses to the total number of Ground Truths (the number of true answers) in the event that the model cannot predict the object in the picture, but the indifferent has that object. Can be obtained from Equation (3), which is taking the number that the model predicted was divided by (TP) the total number of FN or Ground Truth (number of true answers).

$$\frac{TP}{TP+FN} \quad (3)$$

We also convert the data into a confusion matrix that can calculate Accuracy and F1 Score from Precision and Recall from Equation (4).

$$\text{F1 Score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (4)$$

The F1 Score is a combination of precision and recall (F1 was created as a single metric that measures model capability. No need to choose between precision, recall because averaging is already provided).

If we want to measure the performance of a model, we need to use only one value to represent the performance of our model. Normally, the Average Precision (AP) is used by finding the area under the precision and recall curves. The value is between 0 and 1. If it is 1, the model performs best. For example, the graph of Average Precision is square, that is, Precision is 1 and Recall is 1. Finding the area under the graph is equal to 1, meaning that the model can work best shown in Equation (5)

$$\sum_{k=1}^N P(k) \Delta r(k) \quad (5)$$

Where N is the total number of predicted images both correct and incorrect, $P(k)$ is the precision at that position, delta $r(k)$ is the change in the recall value, AP is the search for the area under the curve of precision-recall and the mAP (average mean precision) part is the mean of the AP.

In the object detection system, the prediction part is divided into boundary box prediction and class name prediction for each boundary frame we will measure the overlap between the predicted boundaries. And answer boxes were measured using IoU (intersection over union).

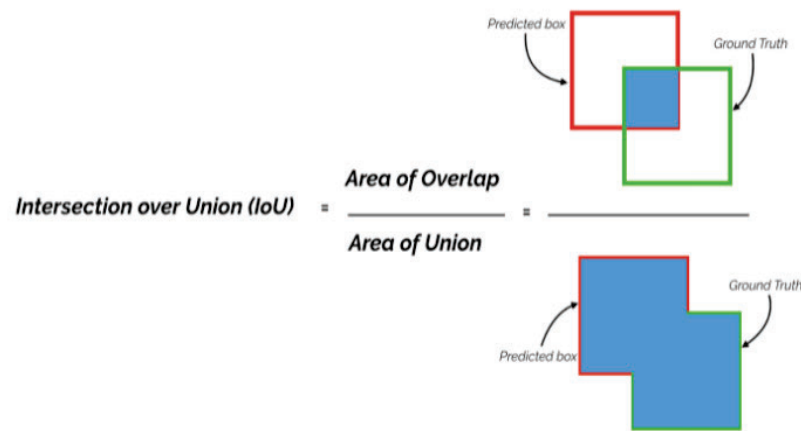


Figure 28. Example of finding IoU (intersection over union).

We normally assign a threshold value of 0.5. If the IoU for prediction is 0.7, we classify the prediction as True Positive (TF), and if the IoU is 0.3, we classify it as False Positive (FP).

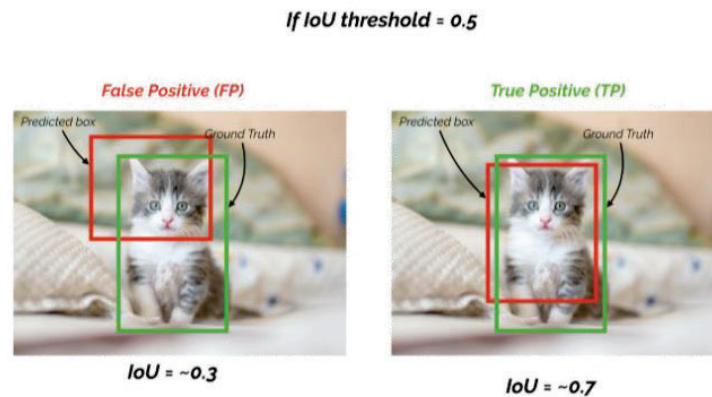


Figure 29. Threshold configuration.

2.4 Studies on facial recognition

In the Face Recognition section, we looked at some of the most popular architectures, VGG (Simonyan, et al.,2014), Inception Resnet V2 (Baldassarre, et al, 2017), ResNet50 (Targ et al,2016), and pre-trained EfficientNet (Tan, Mingxing, and Quoc Le,2019), for computer vision tasks such as visualization neural pattern transfer image classification, captioning, anomaly detection, etc.

- Transfer Learning is a technique that reduces the training time of deep learning models by removing parts of the already trained model. In practice, very few people practice Convolutional Neural Networks in the beginning. Because there is not

a big enough dataset, most people use the modeling method. When training on a large dataset, we take that model as a starting model to continue training on a small dataset for a specific task or to extract features for a specific task. (Data Science Milan, 2020)

- FaceNet (Schroff, et al,2015) is a face recognition startup. By checking and neural network clustering It is a 22-layer deep neural network capable of extracting image properties up to 128-dimensional embedding, and the loss function, triplet loss, is used to manage the data. In the FaceNet working method, it is to find features from face images as Embedding values, then it can be used to search groups of similar faces to find Embedding from images, the deep neural network is used as Inception ResNet V2. We can modify in addition, FaceNet will enable deep neural network values and use L2 normalization to help reduce the distance between images of similar faces and to calculate the distance between Face Net images, the triplet loss function is used to calculate. Calculations of the distance between images, such as using Support Vector Machine (SVM) or K-Nearest Neighbor (KNN) algorithms to speed up grouping faces on demand, are shown in Figure 30. At present, auticulture network at Face Net used mainly Inception Resnet V2.

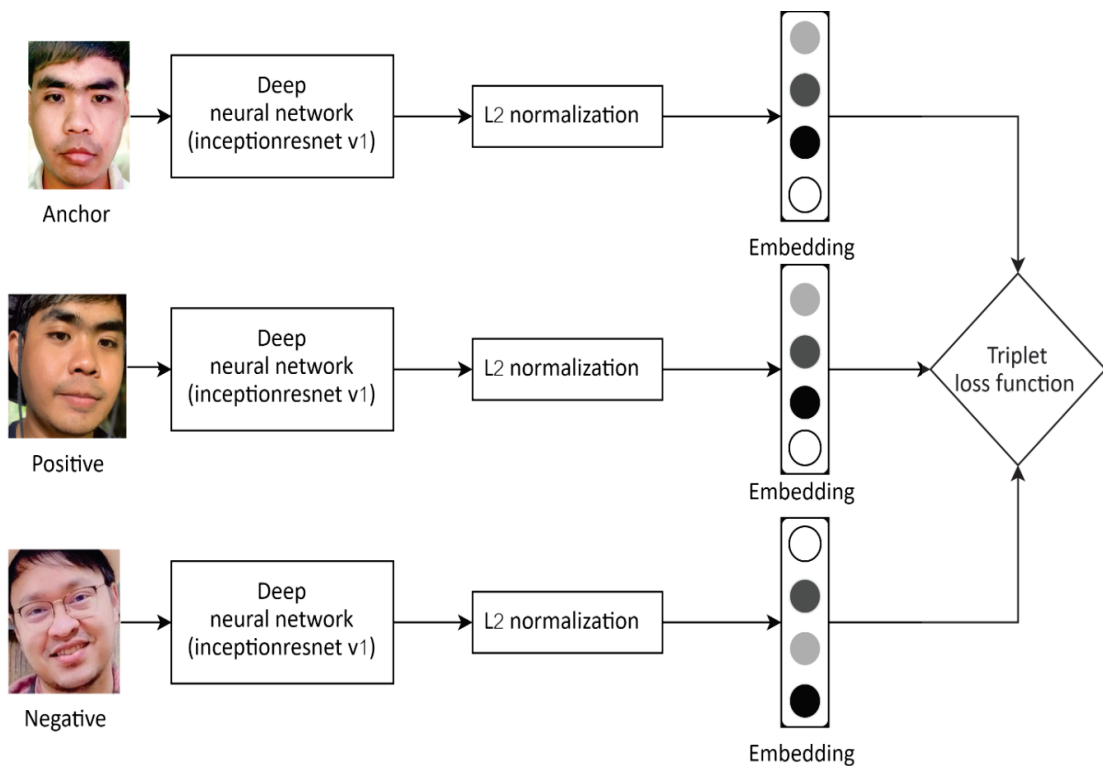


Figure 30. Workflow of FaceNet.

Embedding means (Will Koehrsen,2018) defining features on the face and converting them to numbers. FaceNet uses neural networks to convert a human face into a vector that can represent that image (we call this vector an encoding vector or embedding vector).

The loss function is the need for indicators is a single numeric value that can tell how well our machine learning model works. By comparing the output of the model with the sample output data, both values at the same Input, the data that we will use to calculate through the loss function how much the model has failed, if Loss = 0, meaning no error at all. Optimization can also be used to find ways to minimize the output of the loss function (Christophe Pere,2020).

- **Triplet loss** is a grouping of data based on 3 considerations, which are divided into

Anchor: The default example used in the comparison.

Positive: An example of the same face as Anchor.

Negative: An example of a different face than Anchor.

Triplet loss acts to optimize the distance between positive and negative anchors. The objective is to reduce the distance between the image anchor and the image positive (they are the same person) and increase the distance between the anchor and the negative (they are different people), and then the distance on the embedding space is the loss of the triplet and can be written as a formula,

$$L = \max(d(a, p) - d(a, n) + \text{margin}, 0) \quad (6)$$

where a = anchor input,

p = positive input,

n = negative input,

margin = the margin between $d(a, p)$ and $d(a, n)$.

- **Visual Geometry Group (VGG)** is an architecture simple and widely used Convolutional Neural Network (CNN), the VGG16 architecture was developed and recommended by K. Simonyan and A. Zisserman (2014), and the name VGG is an acronym for the Visual Geometry Group, a group of university researchers of Oxford developed this architecture, and '16' indicates that this architecture has 16 layers. In 2014, the AlexNet architecture was reworked by replacing large kernel-size filters (11 and 5 in the convolutional layer) first and second respectively) with three x three

kernel size filters individually. The VGG16 was trained over several weeks using NVIDIA Titan Black GPUs. The VGG16 is used in many deep-learning image classification techniques and it is popular because it's easy to use. Which remains one of the best vision architectures to date.

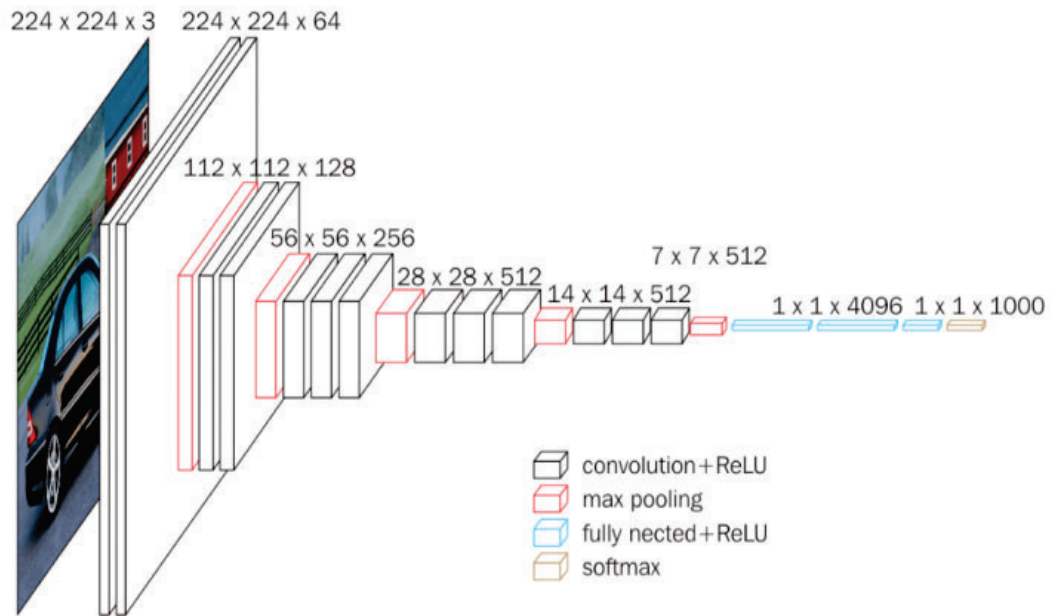


Figure 31. VGG architecture.

(<https://neurohive.io/en/popular-networks/vgg16/>)

The work of VGG is that the image is sent to the convolution1 layer with dimensions 224 x 224 and 64 is the width of the convolution layer and gradually increases by 2 times when reaching the next convolution 512, as well as the size of the image, will continue to decrease until it reaches 7 x 7 in the last layer. VGG's implementation uses a 3 x 3 filter across the entire architecture, and inter-layer pooling is used to reduce the size of the properties of the previous layer finally, it will use fully connected + ReLu, which is from walking in CNN. fully connected will compare the information obtained from various feature maps, Like VGG, ReLu function has been added to help make processing more accurate. The architecture of VGG is shown in Figure. 32.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 32. Structure of VGG.

(https://medium.com/@amir_hf8/implementing-vgg13-for-mnist-dataset-in-tensorflow-abc1460e2b93)

ReLU stands for Rectified Linear Unit, which is a rectified straight-line function that is not S-shaped. ReLU is a simple function because if the input is positive, the slope is equal to 1, if negative, equal to 0. As a result, we train the model much faster. ReLU is implemented because normal CNN processing will get different values, resulting in more processing time. (Surapong, 2019)

- **Inception ResNet V2** is an image-intensive neural network. Based on the ImageNet database, the network has a depth of 164 layers and can predict images into 1,000 categories of objects. The input image size for the network is 299 × 299, and the output is a list of class probabilities determined by the combination of

Inception and Residual connections in The Inception-Resnet block uses a multi-size convolutional filter, which is combined with the rest of the connections or ResNet operations. And it also reduces training time . Figure 33 shows the basic architecture of Inception Resnet V2, and it can be seen that residual connections are maintained throughout the steps of the Inception Resnet V2.

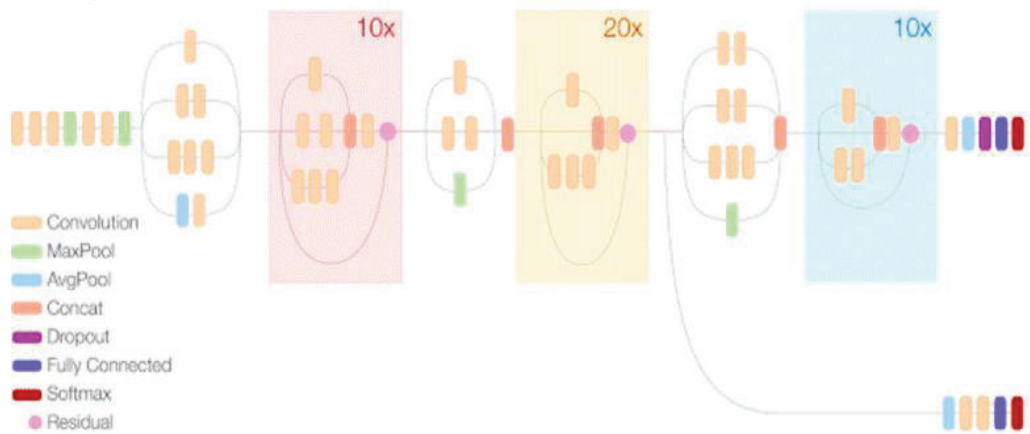


Figure 33. The architecture of Inception-Resnet-v2.

In the workflow, Inception expands the properties of the image. As shown in Figure 33, the properties are divided into three stages: starting with 1x1 filters and gradually adding more resolution with 1x1, 3x3, and 1x1, 3x3, 3x3 filters, then combining all filters with 1x1 filters. Finally, the Resnet method is used, where the result of the previous layer is added to the result of the current layer, as shown in Figure 34. This is a condensed diagram of the function of Resnet Inception v2.

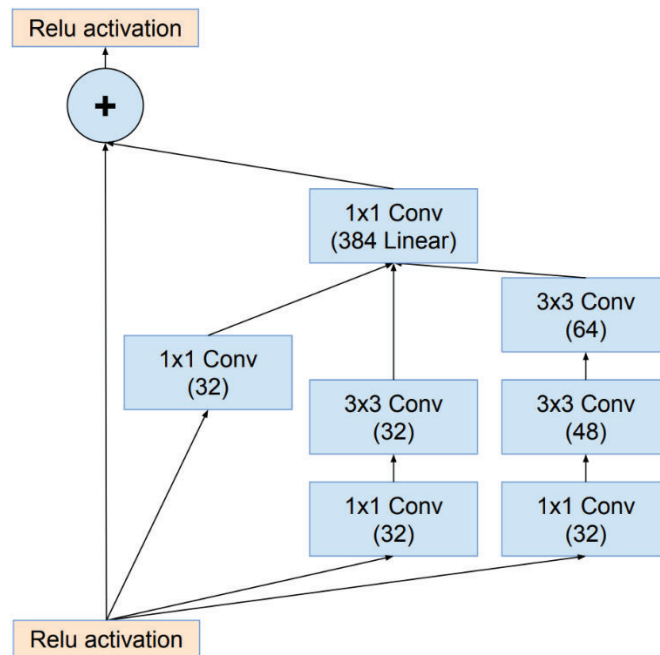


Figure 34. Operation of Inception Resnet V2.

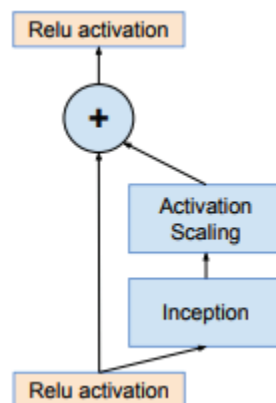


Figure 35. Abbreviated implementation of Inception Resnet V2.

- **ResNet50** or Deep Residual Network, presents a solution to the problem of vanishing gradient in networks where there is quite a lot of depth. Inserting a shortcut (shortcut) in the neural network Vanishing Gradient is the case where there are a large number of layers of convolution layers, causing blurring of the quality. Solved this problem by creating a shortcut to combine the results of the previous layer with the results of the next layer.

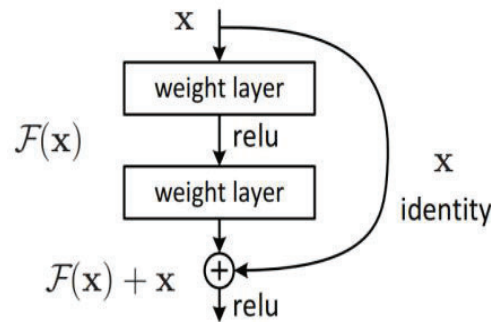


Figure 36. Resnet block.

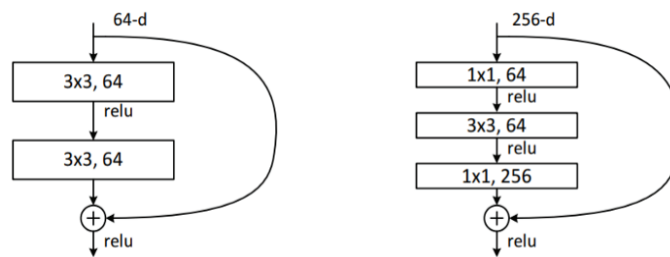


Figure 37. Bottleneck building block.

Adding layers together size must be the same. It can be seen that the two networks on the left side, the number of feature maps is equal to 64 throughout the string, so they can be added immediately, but on the right side, it starts to have 256 and then decreases to 64. We adjust the number of feature maps by using a 1x1 filter, this recalculation structure is known as a bottleneck building block. All these 1x1 or 3x3 filters work on the same principle only using a 1x1 filter combines the properties to reduce the size.

And the origin of ResNet-50, the number 50 comes from the fact that this network consists of 4 large blocks from Figure 38, namely conv2_x, conv3_x, conv4_x, and conv5_x, called stages. The total number of parameters for training becomes the number of layers we use to name it. For example, ResNet50 means that there are 48 layers in the big block + the convolutional layer adjacent to the input layer + the Dense layer adjacent to the output layer. In the research, it is described as [3, 4, 6, 3], which means the number of pairs in all 4 blocks. Taking ResNet50 as an example, the first layer is conv1, which uses a 77 filter to filter the image and gets a size of 64. In the conv2_x layer, there are three layers [11, 64], [33, 64], and [11, 256]. These three layers are repeated three times, so we have 9 layers.

In the conv3_x layer, there are three layers [11, 128], [33, 128], and [11, 512]. These three layers are repeated four times, so we have 12 layers. In the conv4_x layer, there are three layers [11, 256], [33, 256], and [11, 1024]. These three layers are repeated six times, so we have 18 layers. Finally, in the conv5_x layer, there are three layers [11, 512], [33, 512], and [1*1, 2048]. These three layers are repeated three times, so we have 9 layers. After that, we count the Average Pool layer and end with a Flatten layer and a softmax function. We get one more layer. In summary, ResNet(1+9+12+18+9+1=50). The most popular structure usually has a number of layers of 18, 34, 50, 101, 152 (Natthawat, 2020).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 38. Bottleneck building block for any layer.

(<https://iq.opengenus.org/resnet50-architecture/>)

- **EfficientNet** is a novel approach to adjusting the size of Convolutional Neural Network models, presented by the Google Brain team in their paper "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". The driving force behind this research was the need to increase the efficiency and accuracy of models. Previous studies have shown that increasing the size of the network Be it width, depth or resolution that can help improve accuracy. However, most models simply increased dimensions without considering efficiency. Many researchers have discovered a technique called Compound Scaling that can effectively and efficiently increase both the accuracy and performance of a model. This involves increasing width, depth, and resolution together with a constant ratio. Figure 39. shows the differences in Convolutional Neural Networks using EfficientNet

models from B0 to B7, where each version increases the constant ratio (θ) in order (Aakash Nain, 2019).

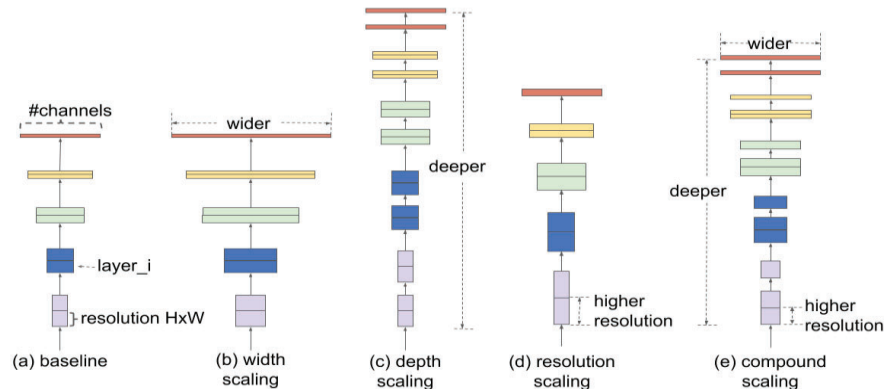


Figure 39. Model Scaling in different ways.

(<https://medium.com/@nainaakash012/efficientnet-rethinking-model-scaling-for-convolutional-neural-networks-92941c5bfb95>)

- **The parameters of neural network/deep learning models** are quite numerous. The important step in building these models is the need to select parameter values or fine-tune them in order to achieve good accuracy. Examples of parameters that need to be fine-tuned regardless of the type of model include the number of layers in the model, type of activation function, learning rate, and loss function. The process of fine-tuning parameters is also known as hyperparameter optimization, and grid search is the easiest and most traditional method. This involves setting the desired values for each parameter and testing them by running the model with all possible sets of parameter values. This method can be used for small models, but it may take a long time to fine-tune large models. For example, if a normal model takes 10 minutes to run, and we have 5 parameters and want to test 10 values for each parameter, using this method to fine-tune the model could take up to 2 years. Therefore, using transfer learning is the best way to reduce the amount of time it takes to work.

Table 1 The performance of the architecture used in facial recognition.

Architecture	Size	Accuracy	Number of Parameters	Depth
VGG-19	549 MB	90.0%	143,667,240	26
Inceptionv3	92 MB	93.7 %	23,851,784	159
ResNet50	98 MB	92.1 %	25,636,712	-
EfficientNet	29 MB	93.3 %	5,300,000	159

From Table 1, evaluating model size, accuracy, number of parameters and depth, Face Net architecture is a suitable architecture for facial recognition implementation.

2.5 Hardware and software equipment used in the operation of the system

- **Hardware**

The main equipment used to control various devices is the NVIDIA Jetson Nano Developer Kit microcontroller board, which is a project by NVIDIA that enables the development of many small-scale AI systems. Additionally, when compared to other projects for neural networks, the price and energy usage are relatively low. With the Jetson Xavier NX board, many inventions can be created, such as IoT applications ranging from small robots to other intelligent systems. All of these have energy-efficient usage, ranging from 5w to 10w.

Furthermore, NVIDIA has also developed similar products to the Jetson Xavier NX, such as the Jetson Nano, Jetson AGX Xavier, and Jetson TX2. The mentioned models have higher performance than the Jetson Xavier NX, except for the Jetson Nano. The reason for choosing Jetson Xavier is that it meets the system's requirements and can help reduce costs as well, as shown in Figure 40 (JETSON XAVIER NX, 2023).

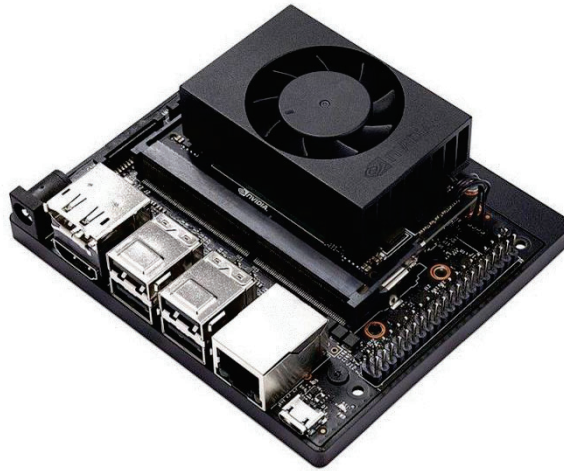


Figure 40. NVIDIA Jetson Xavier NX microcontroller board.

Heat detection cannot be detected by normal cameras, so we must use a camera for heat detection. Non-contact infrared temperature measurement module, using chip MLX90614ESF for Arduino, 3V-5V power supply, I2C connection, using only 2 lines to control It can measure the target temperature without contact at -70 to 380 $^{\circ}\text{C}$, and can also measure the ambient temperature at -40 to 125 $^{\circ}\text{C}$, with a resolution of 0.02 $^{\circ}\text{C}$. (MLX90614 infrared thermometer, 2023)

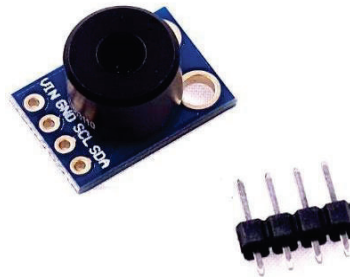


Figure 41. Thermal camera MLX90614ESF Infrared camera.

The camera used to detect faces and recognize faces will use a Camara Webcam 4k camera. In this case, we are concerned about the quality of the image processed by the system. We want the sharpness to be as high as possible so that there are no post-processing problems. And this camera is suitable for this camera can be connected to NVIDIA Jetson Xavier NX via USB port as shown in Figure 42. (Camara Webcam 4k, 2023).



Figure 42. Camara Webcam 4k.

Temperature sensor ultrasonic sensor HC-SR04 distance measuring module using ultrasonic sensor. This ultrasonic module is a device for measuring distances without making contact with the measuring location. Measures from 2 cm to 400 cm by sending an ultrasonic signal at a frequency of 40 kHz to the object to be measured and receiving the reflected signal along with a timer to be used to calculate the distance. There are two main parts: A wave emitter that generates a sound wave for each distance measurement ("Ping") when it hits an object or obstacle. The sound waves are reflected back to the receiver and processed by the electronic circuitry inside the module. If time is measured in the travel of a sound wave in a back-and-forth direction and if the speed of sound in air is known will be able to calculate the distance from the obstructing object as shown in Figure 43 (Ultrasonic Sensor, 2023).

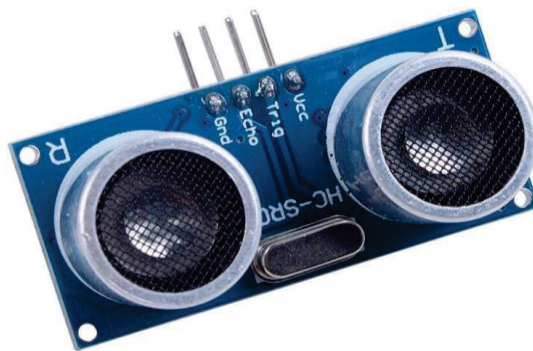


Figure 43. Ultrasonic Sensor HC-SR04.

Arduino Uno R3, the first model produced, has a size of about 68.6x53.4 mm. It is the most popular standard board. Because it is the perfect size for starting to learn Arduino and has more shields to choose from than other Arduino boards that are more specifically designed. The Arduino Uno board has been developed since model R2. R3 and its sub-models replace the IC chip with SMD type, it is the most popular Arduino board because it is inexpensive and most of the projects and libraries developed Support will mainly refer to this board (Arduino, 2022).

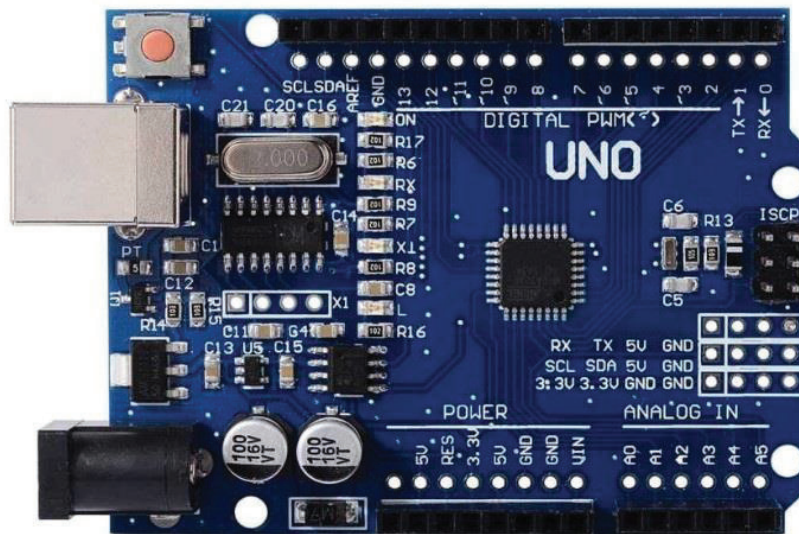


Figure 44. Arduino board.

Table 2 Hardware Features.

NVIDIA Jetson Xavier NX	GPU	NVIDIA Volta architecture with 384 NVIDIA CUDA cores and 48 Tensor cores
	CPU	6-core NVIDIA Carmel ARM v8.2 64-bit CPU
	DL Accelerator	2x NVDLA Engines
	Vision Accelerator	7-Way VLIW Vision Processor
	memory	8 GB 128-bit LPDDR4x @ 51.2GB/s
	storage	microSD
	video encoding	2x 4K @ 30 6x 1080p @ 60 14x 1080p @ 30 (H.265/H.264)

	video decoding	32x 1080p @ 30 (H.265)
	connectivity	Gigabit Ethernet, M.2 Key E (WiFi/BT included), M.2 Key M (NVMe)
	screen	HDMI 2.0 or DP 1.2 eDP 1.4 DSI (1 x 2) 2 simultaneously
	USB	4x USB 3.1, USB 2.0 Micro – B
	size	69, 6.45 mm x mm
	mechanical	103 mm x 90.5 mm x 34.66 mm
Board Arduino	Microcontroller	ATmega328
	Power supply	5V
	Input power (recommended)	7-12V
	Power input (limited to)	6-20V
	Digital I/O pins	14 ခု (6 output PWM)
	Analog input pin	6 ခု
	DC current per I/O pin	40 mA
	DC 3.3V	50 mA
	Flash Memory	32 KB (ATmega328)
	RAM	2 KB (ATmega328)
	EEP ROM	1 KB (ATmega328)
	Clock Speed	16 MHz
Camara	Camara Webcam	4k resolution can be connected to the board via USB.
Infrared	Power Supply	3V - 5V

camera MLX90614 ESF	Connection	I2C
	Measures environmental temperature	-40 to 125 degrees Celsius.
Ultrasonic Sensor HC- SR04	Working voltage	DC 5 V
	Static current	3 mA
	Working temperature	Working temperature: 0 ~ 70
	Output way	GPIO
	Induction Angle	Less than 15
	Detection range	2 cm to 4 m
	Detecting precision	0.3 cm
	Sensor size	45 x 20 x 1.6mm

- **Software**

- 1. Google Colab**

Google Colab, full name is Google Colaboratory, is a Software as a Service (SaaS) service hosting Jupyter Notebook programs on the Cloud from Google. We can use Google Colab to program the Python language and provide GPU, TPU services to access 12 hours at a time if you want more sustained use, you'll need to upgrade. Colab's GPU offerings will be randomly deployed with three GPUs: the Nvidia Tesla K80, the Nvidia Tesla T4, and the Nvidia Tesla P100 (Google Colab, 2023).

- 2. Jetpack version 4.6**

jetpack is a central SDK for the entire NVIDIA Jetson family of products. It supports both training and inferences for AI, and supports multiple frameworks including TensorFlow, PyTorch, Caffe, and MXNet. It also includes Linux desktop and rendering capabilities. High-level results available immediately (Nvidia developer, 2023).

3. Python 3.6.9

The Python programming language is a computer programming language. It was designed to be an easy-to-read scripting language. By removing the complexity of the structure and grammar of the language in terms of converting a set of instructions that we write to machine language, Python has an Interpreter function, meaning it translates a set of instructions line by line to be fed into the processor for the computer to work as we want. In addition, the Python programming language can be used to write various types of programs without being limited to any specific work (General-purpose language), thus making it widely used in many large global organizations such as Google, YouTube, Instagram, Dropbox and NASA, etc. The reason for choosing version 3.6.9 is because it is the default operating system ubuntu 18.04 (Ubuntu packages, 2022).

4. Tensorflow-gpu 1.13.1

TensorFlow is an open-source machine learning library developed by Google. It provides a platform for running mathematical operations on various hardware devices, including Google's CPU, GPU, and Tensor Processing Units (TPUs). GPUs are commonly used to train machine learning models. The main purpose of TensorFlow is used different mathematical models to enable efficient training of deep neural networks. These networks have the ability to learn patterns and make predictions from data without explicit human instructions, making them highly effective compared to traditional approaches. Additionally, TensorFlow supports distributed computing across multiple machines with different operating systems and GPUs. This enables efficient and scalable processing of machine learning tasks. (Derrick Mwit, 2023).

5. Opencv 3.4.2

OpenCV is a software program developed with the support of Intel Corporation Limited for image processing. It allows for the easy development of various programs in computer vision. OpenCV can be used on both Linux and Windows operating systems and supports multiple programming languages. It provides functionalities for processing still images and videos, along with pre-built functions for managing image data and performing basic image processing tasks such as edge detection and image filtering (Ken RobotSiam, 2022).

6. Matplotlib 3.1.1

Matplotlib is used to generate visualizations. It can be used to plot different types of plots, write scatter diagrams, histograms, bar charts, error charts, box plots, and plots. Leverages NumPy, a library that provides extensions. Numerical arithmetic for Python (Devdocs, 2018).

7. Numpy 1.16.6

NumPy is a Python arithmetic library. Written in C, it is fast and efficient, capable of handling multidimensional arrays and matrix data. NumPy can be easily installed through the Python package installer with the command `pip install numpy`. The benefits of NumPy are that it is easy to create and manage multidimensional arrays in a number of ways, such as specifying the size of an array and providing the data inside it. Through NumPy functions, specifying the version of NumPy allows the system to work efficiently (Borntodev,2020).

Table 3 Software Features.

Name	Version
ubuntu	18.04
Jetpack	4.6
Python	3.6.9
Tensorflow	1.13.1
Opencv	3.4.2
Mathplotlib	3.1.1
Numpy	1.16.6

2.6 Related Works.

2.6.1 Face mask detection

The research by Yishi Guo and Burkhard C. Wünsche (2020) examined the comparison of four common face detection algorithms: Haar Cascade, HOG, SSD, MTCNN. The test was to bring different algorithms. The same face detection is

categorized into Frontal faces, Blurred faces, Slightly Rotated faces, Side faces, Upside down faces, lying down faces and partially occluded faces type. It also performs best with 89.28% detection accuracy and a frame rate of nearly 10 fps, and for MTCNN, it has the highest accuracy of 89.62%, which is slightly more. But cannot work in real time. And then Thilinda Edirisooriya and Eranda Jayatunga (2021) on this comparison used five face detection methods: ViolaJones, HOG-SVM, MTCNN, SSD and Maxmargin Object Detection (MMOD). Different light intensity, face angle, face size and different occlusion types, divided into Illumination Intensity, Angle of Face, Scale of Face, Occlusion of Face WIDERFACE video data and image samples were used for analysis. Test results have shown that SSD performs better at face detection with 98.7% accuracy and highest detection speed of 17.33 FPS, while MMOD has the lowest performance and Viola- Jones gives the lowest accuracy. Md. Iftekher Hossain et al. (2021) research on the enhancement of facial recognition performance using Mean Embeddings, in which the paper discussed face detection testing, includes Haar Cascade, HOG, MTCNN and SSD. The highest detection accuracy of 95.22% and speed of 10 FPS, followed by MTCNN, has the highest detection accuracy of 80.03% and speed of 1 FPS, which is not classified as Realtime detection.

Andrey V. Savchenko's (2021) investigated is a multi-task learning study of convolutional neural networks to identify faces and classify facial features (age, gender, ethnicity) trained using cropped faces without borders. To make predictions, facial expressions are highlighted more prominently face recognition architectures such as Inceptionv3, EfficientNet, VGG and Res Net were presented. Experiments using emotion classification on the Affect Net dataset showed that Inceptionv3 was the best predictor with 8 accuracy classes was 59.65% and the 7 classes accuracy was 63.1%. It is an efficient facial feature separator and greatly shortens the work time. Sheikh Rufsan Reza et al (2021) conducted research on face mask detection using deep learning techniques on IoT devices, especially the Convolutional Neural Network (CNN) with four network architectures including Mobile Net V2, Inception V3, VGG 16, and Res Net 50, which were used for face mask detection and testing by inference on IoT devices powered by GPUs, including NVIDIA Jetson TX2 and NVIDIA Jetson Nano. The comparison and analysis of performance were conducted using various training data sizes, with Inception V3 achieving the highest accuracy. Next Malik, Hassaan, and Tayyaba Anees (2022) applied the classification algorithm using deep learning to identify Covid-19, pulmonary edema, and lung cancer from chest X-ray images. They proposed a model that combines VGG-19 and Convolutional Neural Network (CNN) called BDC Net to be used with various standardized databases for

the diagnosis of Covid-19 and other respiratory diseases. Moreover, they calculated and compared the accuracy of the model categorization with pretrained models such as Res Net-50, VGG-16, VGG-19, and Inception V3 for the classification of diseases in different chest areas, where CNN, VGG-16, VGG-19, Res Net-50, and Inception restnet V2 achieved accuracy of 97.35%, 97.14%, 97.15%, and 95.10%, respectively. And then, Poonam Verma et al. (2021) compared the accuracy of deep learning architectures with different learning rates using various performance-enhancing tools to reduce the problem of overfitting. In addition, they proposed a method to classify COVID-19 images using a group of two-layer Convolutional Neural Networks with Transfer learning, which reduced the time taken to classify the images and achieved an accuracy of almost 90.45%. They focused on architectures such as VGG-16, Res Net 50, Inception V3, and EfficientNet with learning rates equal to 0.0001. EfficientNet had an accuracy of 70.09%, while Inception V3 had an accuracy of 70.30% with learning rates equal to 0.0004, which is not significantly different. In the medical applications, Karim Karou Diallo and Yun Ju (2020) investigated the efficient detection of COVID-19 through chest X-rays at a higher level than that of new radiologists. This algorithm uses the EfficientNet architecture, which expands and is named K-EfficientNet. K-EfficientNet is related to advanced resizing, which resizes images from 112×112 to 224×224 during the training process. When combining six publicly available datasets, a large dataset named K-COVID with 14,124 X-ray images of patients affected by pneumonia or COVID-19 and patients with normal X-ray images was created. The use of transfer learning on the dataset and adding ImageNet data enabled us to achieve 97.3% accuracy and 100% speed and positive prediction for COVID-19 detection. Furthermore, it was compared to VGG-19, ResNet-50, and EfficientNet. The accuracy of EfficientNet was slightly lower than the presented model at 91.5%.

Notice that the accuracy of each model varies greatly due to factors such as image data, processing time and using different learning rates. So, we used information from Papers with Code, a website that collects information about various architectures and tested on the same database, ImageNet, for better performance comparisons as shown in Figure 45 (from the FaceNet image we are interested in studying using the Inception RestNet V1 model).

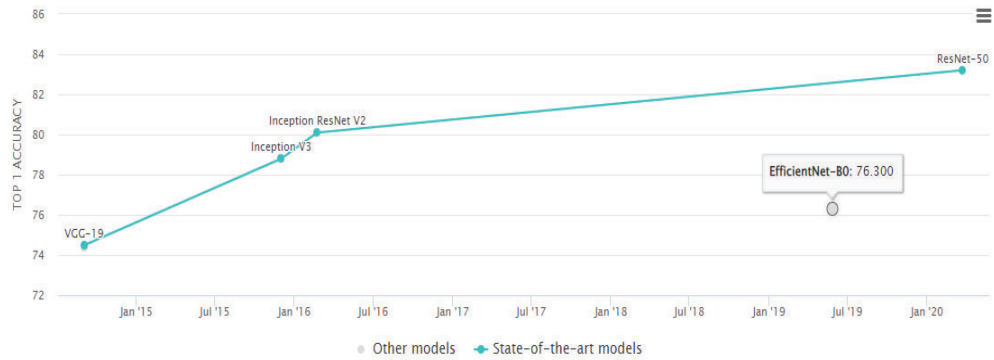


Figure 45. Compare architectures on ImageNet.
(<https://paperswithcode.com/>)

We gathered data from the literature review to include factors related to the development of various devices, including face mask detection and facial recognition systems. Including equipment used that contains details of the research used in the preparation of the thesis as shown in Table 2.

Table 4 Summary of the models used in related works.

Paper name	Year	Authors	Model				Comment
			Haar	HOG	SSD	MTCNN	
Face Detection Comparison of Face Detection Algorithms on Mobile Devices	2020	Yishi Guo and Burkhard C. Wünsche	✓	✓	✓	✓	The SSD can detect all kinds of faces. It also performs best with a detection accuracy of 89.28%.
Comparative Study of Face Detection Methods for Robust Face Recognition Systems	2022	Thilinda Edirisooriya and Eranda Jayatunga		✓	✓	✓	The SSD did a better job at detecting faces with 98.7% accuracy.
An Efficient Way to Recognize Faces Using Mean Embeddings	2021	M. I. Hossain, Sama-E- Shan and H. Kabir,	✓	✓	✓	✓	SSD has the highest detection accuracy of 95.22%.

Paper name	Year	Authors	Model				Comment
			VGG	Inception ResNet V2	ResNet-50	Efficient net	
DFFMD: A Deepfake Face Mask Dataset for Infectious Disease Era with Deepfake Detection Algorithms	2020	Alnaim, Norah M. et al.	✓	✓			Testing on the VGG19, InceptionResNetV2 and CNN InceptionResNetV2 models. It has the highest accuracy of 99.39% compared to CNN and VGG19 models. It was also highest among other metrics such as accuracy, recall, and f1 score, and the presented CNN model had a lower accuracy than the transfer learning model at 77.80%.
Search to Distill: Pearls are Everywhere but not the Eyes	2020	Yu Liu et al.		✓		✓	EfficientNet with Input size 600 and accuracy is about 4.4% higher than Inception-ResNet-v2 with Input size 299.
Multimodal Biometric Authentication: Deep Learning Approach	2021	Jena, Pabitra Priyadarshini et al.		✓	✓	✓	Biometric authentication through a variety of biometrics Authentication is done with both fingerprint and face, with transfer learning used to recognize all three fingerprint models. Inception ResNet V2 performs best in real-time fingerprint recognition.
Multiple skin lesions diagnostics via integrated deep convolutional networks for segmentation and classification	2020	Mohammed A. Al-masni et al.		✓	✓		It is a comparison of categorizing skin diseases, with Inception ResNet V2 having the best accuracy out of the 3 models at 87.16%.

Table 5 Summary of the equipments used in related works.

Paper Name	Year	Authors	Equipment			Comment
			Hardware	Best model	Fps	
A Low-Cost Embedded Security System for UAV-Based Face Mask Detector Using IoT and Deep Learning to Reduce COVID-19	2022	Adhan Othman, Nashwan and Aydin, Ilhan	Jetson Nano	MobileNetV2	8.72	has proposed applying Jetson Nano to camera drones which the detection efficiency is 99%
Facemask Wearing Alert System Based on Simple Architecture with Low-Computing Devices	2022	D. -L. Nguyen, M. D. Putro and K. -H. Jo	Jetson Nano	Res Net-50	26.18	The results of the Jetson Nano experiments showed that as the number of users increased, the detection speed continued to decrease.
Real-time Face Mask Detection System on Edge using Deep Learning and Hardware Accelerators	2021	S. Ruparelia, M. Jethva and R. Gajjar,	Jetson Nano and Jetson Xavier NX	Yolo V5	NX 30 Nano 12	Using it on the Jetson Xavier NX gave 30 fps while using it on the Jetson Nano gave it 12 fps. For the YOLOv5 NX detection model it gave a higher fps compared to the Nano.

Paper Name	Year	Authors	Equipment			Comment
			Hardware	Best model	Fps	
Face Mask Detection Based on Machine Learning and Edge Computing	2022	I. Jovović, D. Babić, S. Čakić, T. Popović, S. Krčo and P. Knežević,	Jetson Nano Raspberry Pi3	MobileNetV2	64	It's a comparison between Jetson Nano and Raspberry Pi3, Jetson Nano is more powerful.
Real-time Face Mask Detection System on Edge using Deep Learning and Hardware Accelerators	2021	S. Ruparelia, M. Jethva and R. Gajjar,	Jetson Nano and Jetson Xavier NX	Yolo V5	NX 30 Nano 12	Using it on the Jetson Xavier NX gave 30 fps while using it on the Jetson Nano gave it 12 fps. For the YOLOv5 NX detection model it gave a higher fps compared to the Nano.

Chapter 3

Research Methodology

3.1 System

In our research studies, we have found that the SSD model performs the best for face detection, with the highest accuracy compared to the other models. In terms of processing speed, the SSD model also outperforms the other models, achieving higher frames per second (FPS). Therefore, we have chosen the SSD model for face detection. For facial recognition, we utilize FaceNet to compare faces in the database and extract their features. When comparing different architectures, we observed that the Inception Resnet V2 architecture performs similarly to EfficientNet in terms of accuracy. Since FaceNet originally used the Inception Resnet V2 architecture for training, we have selected it as the core architecture for facial recognition. This choice is also supported by the concept of transfer learning.

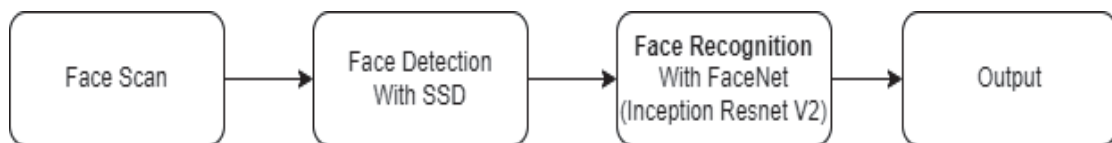


Figure 46. System Overview.

The spread of the COVID-19 virus affects the effect of recording the time attendance of personnel today, making the traditional attendance pose a risk of spreading germs. Causing the need to add facilities that are mask detection systems and face recognition while wearing masks used for time attendance recording that can greatly facilitate the user and can reduce the spread of the virus while recording time in and out of work as well.

This chapter discusses building a low-cost system. including the use of prototypes and introductions to various devices and the overall operation of the system. We can find a prototype in section 3.1 describing the device components and device properties. Both in terms of hardware and software, in Section 3.2, the development of a mask detection system, and in Section 3.3, the development of a face recognition system while wearing a mask, and finally, Section 3.4 is the process of connecting the camera module to the system and transmitting notification to LINE application.

We use a Jetson Nano board to bring the MLX90614ESF Infrared camera used for temperature detection and the Raspberry Camera to detect face masks and compare the detected faces with the figure database. Figure 47 shows the structure of the system.

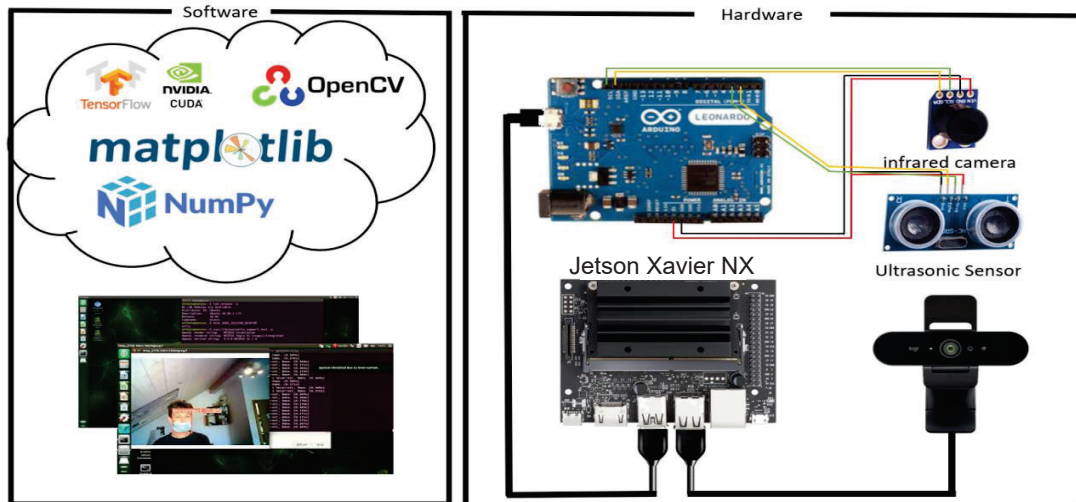


Figure 47. System software and hardware.

Designing an infrared camera system is a challenging task. Because connecting an infrared camera to the Jetson board is complicated because the camera is more developed than the Arduino board, so we design the ultrasonic sensor to work with the infrared camera using the Arduino board, then we pass the values to the Jetson Xavier NX board which is designed specifically for the blind to increase detection efficiency. We have placed a video camera, infrared camera and ultrasonic sensor in the same plane taking into account the efficiency of mask detection face recognition range and temperature detection accuracy range which we set at 1 meter. Figure 48 shows the prototype design and Figure 49 shows the device connection for actual use.

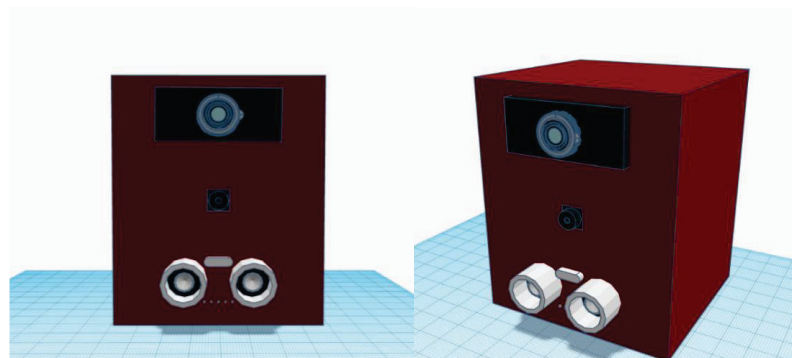


Figure 48. The prototype designs.

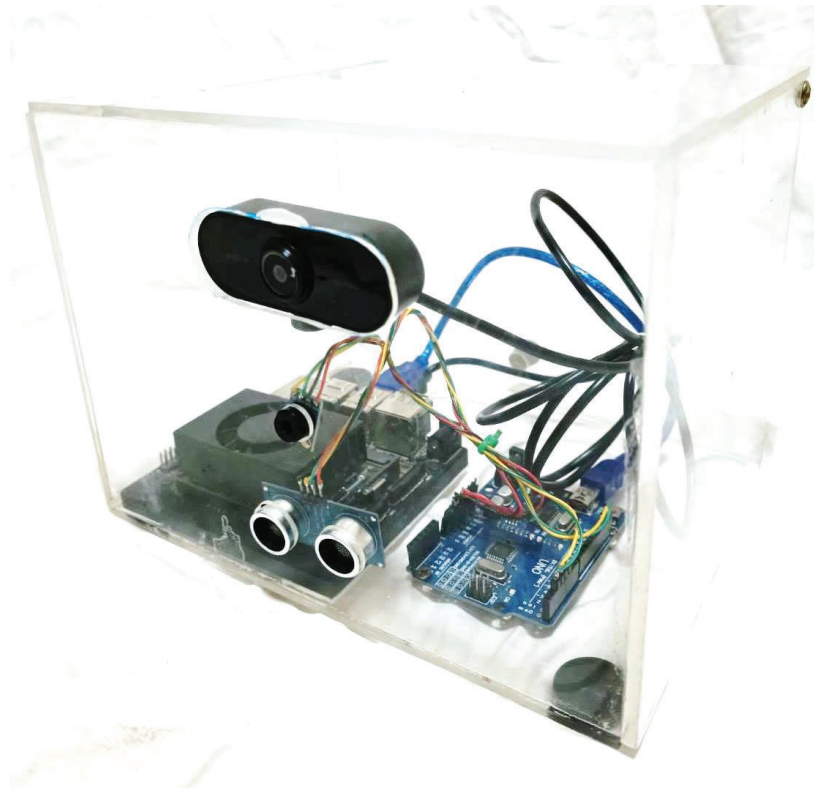


Figure 49. Equipment connection for actual use.

When connecting equipment, we take into account the ease of installation. We have designed both cameras and video recording. The temperature-sensing camera and sensor ultrasonic can receive values simultaneously when the user walks through the camera. It also connects to a LAN cable to send data to the user's phone.

3.2 Development of mask detection system

In the process of creating a mask detection model, we aimed to increase the variety of face masks used in our dataset to enable the system to recognize more patterns of mask-wearing and improve the accuracy of mask detection. To achieve this, we included image data from the Real-World Masked Face Dataset (RMFD), which captures masked and non-masked faces. We selected 10,000 images, each depicting various ways of wearing face masks, including images of people wearing masks correctly and improperly, as well as group portraits of people wearing masks and not wearing masks, or with an object covering their mouth area. Although these images may result in discrepancies in mask predictions, with enough practice time, the model will be able to identify objects that obscure the face and are not masks. An example of the Real-World Masked Face Dataset (RMFD) is shown in Figure 50.



Figure 50. Real-World Masked Face Dataset (RMFD) database example.

Then, we proceeded to label the image, which means indicating the position of objects in the image and identifying what that position represents. This process is similar to teaching a child to recognize something, just as we teach a computer to recognize objects. We used the Label Image version 1.8.0 program to label the images. We first imported the image into the program, then drew a rectangular box around the area we wanted to label in the image. We needed to identify the face with a mask. Once we finished drawing the box, the program prompted us to enter the class name. Based on Figure 51 , we assigned the class name "Mask" and "Not_mask."

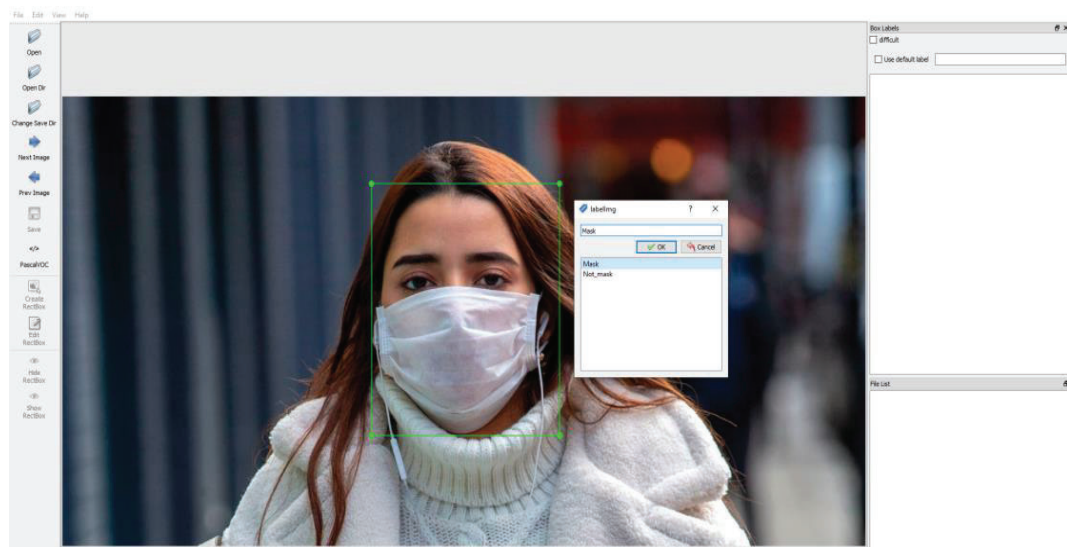


Figure 51. Cropping faces in Label image program.

After that, we get an .xml file that is used to indicate the position of objects and their class names. The file contains multiple objects in the image, and the red box includes a description starting from the Class Name: Mask, Not_mask, and the position of the bounding box (Bndbox), including Xmin, Ymin, Xmax, Ymax of each box that we have placed over the face area as shown in Figure 52.

```

<?xml version="1.0"?>
- <annotation>
  <folder>images</folder>
  <filename>1.png</filename>
  <path>C:\Users\Mario\Desktop\images\1.png</path>
  - <source>
    <database>Unknown</database>
  </source>
  - <size>
    <width>630</width>
    <height>424</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  - <object>
    <name>Mask</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    - <bndbox>
      <xmin>80</xmin>
      <ymin>93</ymin>
      <xmax>218</xmax>
      <ymax>278</ymax>
    </bndbox>
  </object>
  - <object>
    <name>Not_mask</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    - <bndbox>
      <xmin>398</xmin>
      <ymin>109</ymin>
      <xmax>544</xmax>
      <ymax>278</ymax>
    </bndbox>
  </object>
</annotation>

```

Figure 52. .xml file result of making Label image.

Next, we feed all the images to the deep learning model training process so that the mask can be detected. In the first step, we will divide all the images into 3 sets: Train, Validation, and Test, with a split of 70:20:10 respectively. The images are divided as follows: 7,000 images for training, 2,000 images for validation, and 1,000 images for testing. We display the various variable settings used in the training process in Table 6, with each set being accompanied by figures of individuals wearing and not wearing face masks, as well as the .xml files of each image. Next, we will train the model using the Train and Validation datasets.

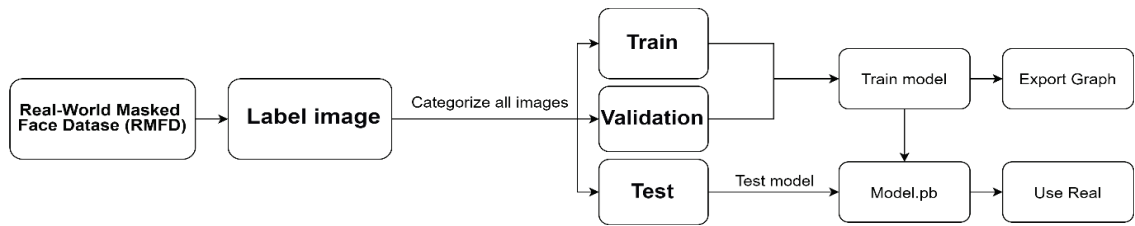


Figure 53. Training operation of the face mask detection model.

```

#---image processing
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) // converse เปลี่ยนแปลงรูปแบบค่าสีจาก blue green red ให้กลายเป็น red green blue
img_rgb = img_rgb.astype(np.float32) // ทำการเปลี่ยนค่าภาพที่อ่านได้จากภาพ red green blue ให้เป็นค่าจุดทศนิยมทศนิยมขนาด 32 bit
img_rgb /= 255 // ทำการหาค่า binary จากเลขทศนิยม 32 bit โดยการหาค่าเฉลี่ยโดยหารด้วย 255

#---face detection
img_fd = cv2.resize(img_rgb, fmd.img_size) // ตรวจสอบใบหน้าโดยการเปรียบเทียบจากไฟล์ xml รูปพรรณสัณฐานของใบหน้า
img_fd = np.expand_dims(img_fd, axis=0) // ทำการเปลี่ยนขนาดของภาพที่ทำการอ่านได้จากตัวกล้อง

bboxes, re_confidence, re_classes, re_mask_id = fmd.inference(img_fd, height, width) // ติกรอบล้อมรอบใบหน้าด้วยกล่องตามค่าสีที่ดึงไว้
if len(bboxes) > 0: // ถ้ากล่องที่ครอบใบหน้ามีค่ามากกว่า 0
    for num, bbox in enumerate(bboxes): // ทำการนับจำนวนกล่องที่ตรวจจับใบหน้าได้ทั้งหมด
        class_id = re_mask_id[num] // เปลี่ยนค่าภาพใบหน้าที่อยู่ในกล่องให้เป็นค่าขาวดำโดยการหาค่าเฉลี่ยหารด้วย 255
        if class_id == 0:
            color = (0, 255, 0) # (B,G,R) --> Green(with masks)
        else:
            color = (0, 0, 255) # (B,G,R) --> Red(without masks)
        cv2.rectangle(img, (bbox[0], bbox[1]), (bbox[0] + bbox[2], bbox[1] + bbox[3]), color, 2) // ติกรอบล้อมรอบใบหน้าทั้งหมด
        # cv2.putText(img, "%s: %.2f" % (id2class[class_id], re_confidence[num]), (bbox[0] + 2, bbox[1] - 2),
        # cv2.FONT_HERSHEY_SIMPLEX, 0.8, color)
  
```

Figure 54. Coding of Face Detection.

A comparison study of facial detection models, including Haar, SSD, HOG and MTCNN, found that SSDs can detect objects faster than other models, and with similar accuracy, SSDs are attractive models. Most interested in this experiment the speed of detection is critical to system performance and the table shows the settings of the SSD model used to train mask detection.

Table 6 Model training settings for SSD.

Model	SSD
Image size	512*512
batch	8
Epochs	100
Learning rate	0.004

3.3 Face recognition system while wearing a mask

Due to the COVID-19 outbreak, we are unable to collect user data directly. As an alternative, we have decided to use Google Forms to store user data. Users can upload an image as shown in the Google Forms example, and then we will use the Form Builder to process the data when the user enters their name in the given name box. The box will be set to the folder name of the image, as shown in Figure 55.



Figure 55. Using Google form to collect data.

To collect comprehensive information about faces, we aimed to capture faces from different angles and cover every part of the face. To achieve this, we included both masked and non-masked faces in the images, consisting of a straight face, a left-right face at 45 degrees, a left-right face at 90 degrees, a face looking up, a face looking down, and a face with the mouth covered by hands (Xiaoguang Lu and Jain, A.K. 2006), as shown in Figure 56. Our data collection targeted Personnel of PSU. Wittayanusorn Surat Thani School, comprising 76 individuals, with 14 photos each, resulting in a total of 1064 photos, as shown in Figure 57.



Figure 56. Different angles of the face were used to collect data.

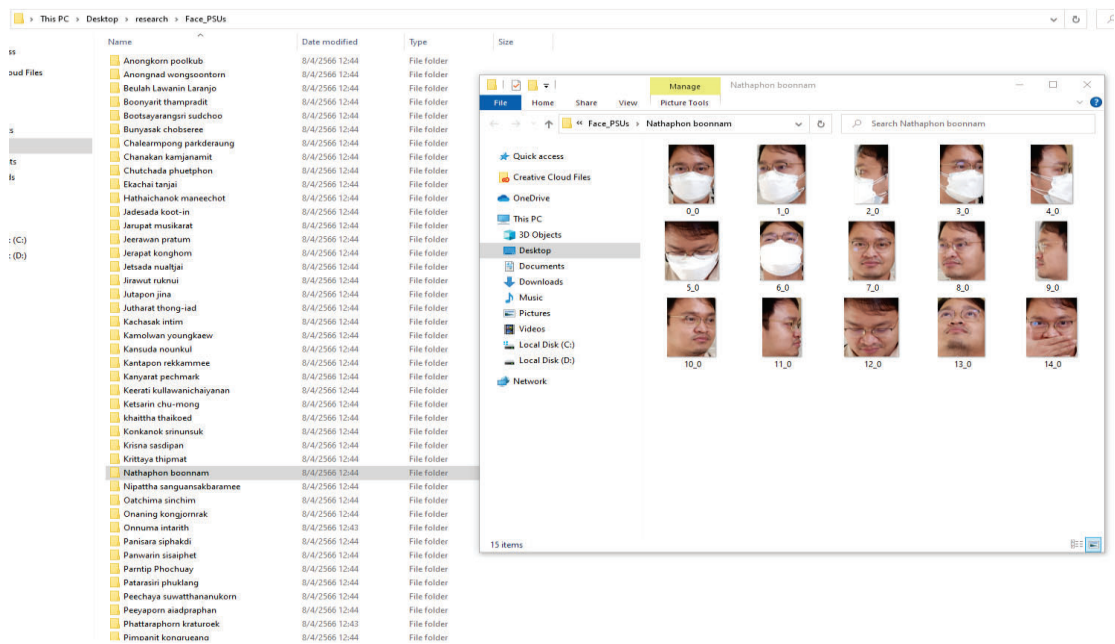


Figure 57. Dataset Personnel of PSU. Wittayanusorn Surat Thani School.

After collecting the data, the resulting images are not immediately usable. We need to perform face alignment, which involves using algorithms to detect faces in the images and cropping out only the faces (to reduce unnecessary processing). We have set the value for face alignment to 20 (margin = 20), leaving a small amount of the surroundings of the face image. We also require the background of the image because we need to capture the total area of the face. If the margin is set to zero, there will be no space left in the image background as shown in Figure 58.

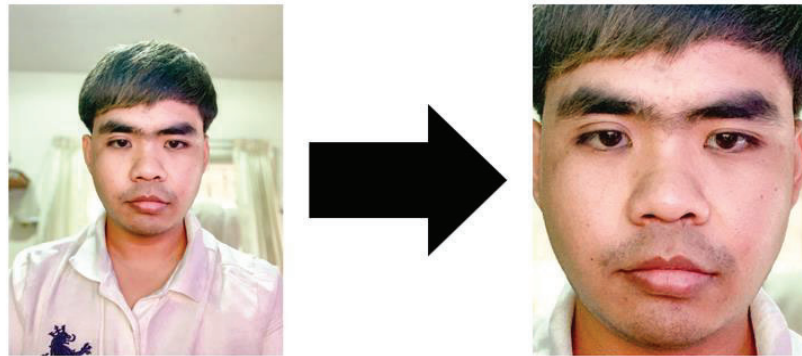


Figure 58. Result of Face Alignment work.

We also included user face data generated by our own system. We took the non-masked faces obtained from all collected data and added other types of masks to those faces. Using the finalized model shown in Figure 59, we plotted points on the face, defining all points located around the nose and mouth. We then added prepared face masks in .png format, a total of 16 randomly added to the user's face, as shown in Figure 60.



Figure 59. Face masks for use in adding image data.

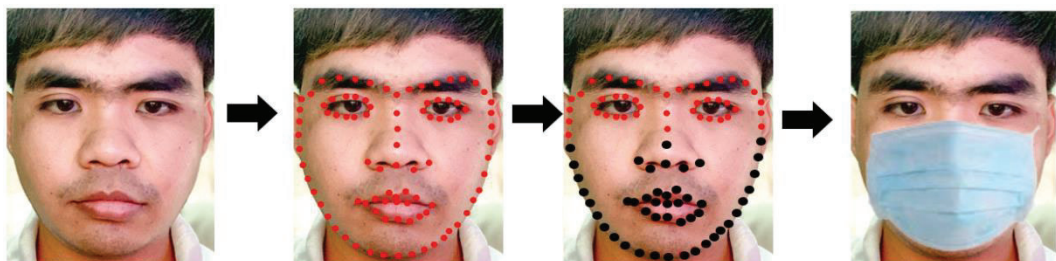


Figure 60. Operation of adding a mask to a non-masked face.



Figure 61. Result of adding masks.

In terms of creating a face recognition system because the model we use already has the ability to find embedding values and when we face aligning the image, the background information of the image is not a problem for embedding values, but we want to increase the accuracy of the model to be used again the working method of training the face recognition model is shown in Figure 62.

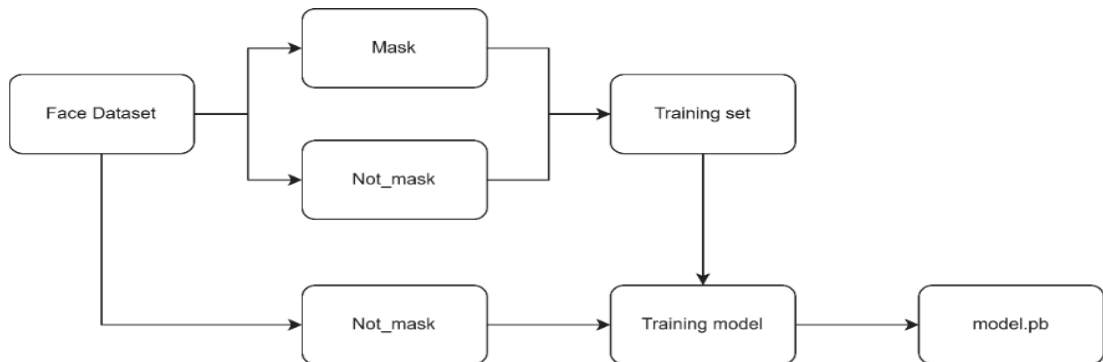


Figure 62. Training a face recognition model.

In terms of creating a face recognition system, for training the model, we use human dataset to train the model. We divide the images into two sets: images with masks and images without masks. The set with masks is used for training, while the set without masks is used for testing. Its purpose is to enable the system to learn how to recognize faces with and without masks. The working principle of a facial recognition system involves the use of FaceNet, which sets up an embed using a CNN to extract features from all images in the training dataset.

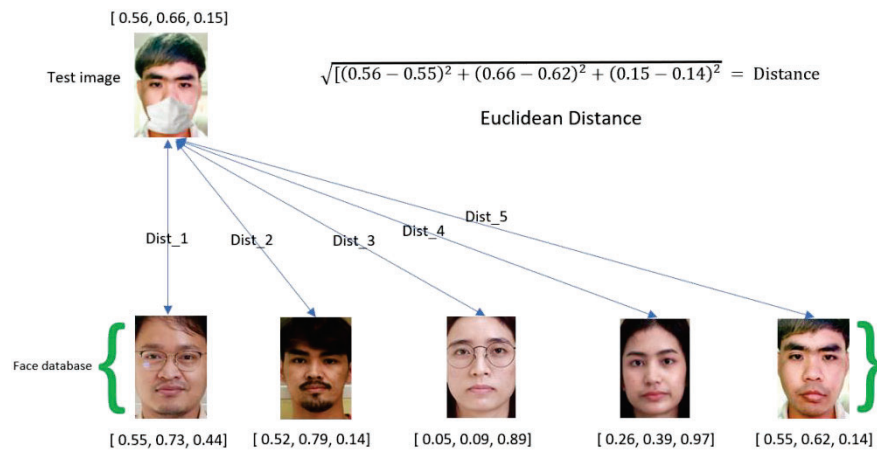


Figure 63. Operation of the Euclidean distance function in the triplet loss function.

```
# ---face recognition
name = ""
if len_ref_path > 0: // ทำการเปรียบเทียบใบหน้าทีตรวจจับได้จาก open CV เปรียบเทียบในฐานข้อมูลว่าใบหน้านั้นเป็นใคร
img_fr = img_rgb[bbox[1]:bbox[1] + bbox[3], bbox[0]:bbox[0] + bbox[2], :] # crop // ทำการปรับ
img_fr = cv2.resize(img_fr, (model_shape[2], model_shape[1])) # resize // ทำการปรับขอบบนใบหน้า
img_fr = np.expand_dims(img_fr, axis=0) # make 4 dimensions

feed_dict[tf_input] = img_fr // เก็บชื่อของบุคคลที่ตรวจจับได้โดยการเปรียบเทียบฐานข้อมูลใบหน้าไว้ในตัวแปร
embeddings_tar = sess.run(tf_embeddings, feed_dict=feed_dict)
feed_dict_2[tf_tar] = embeddings_tar[0]
distance = sess_cal.run(tf_dis, feed_dict=feed_dict_2)
arg = np.argmin(distance) # index of the smallest distance // หาค่าระยะห่างระหว่างใบหน้ากับตัวกล้องและ

if distance[arg] < threshold: // แสดงชื่อของบุคคลเจ้าของใบหน้าทีทำการตรวจจับได้ชี้หน้าจอ
name = paths[arg].split("\\")[-2].split(".")[0]

##### New Serial #####
port.write(0x01) // ส่งรหัสคำสั่งผ่าน serial port โดยใช้โปรโตคอล USB เพื่อทำหน้าที่ในการอ่านค่าอุณหภูมิ
#s_temp = readlineCR(port) // สร้างตัวแปรเพื่อรับค่าอุณหภูมิที่อ่านได้จากตัวเซ็นเซอร์
s_temp = port.readline() // สร้างตัวแปรเพื่อรับค่าอุณหภูมิที่อ่านได้จากตัวเซ็นเซอร์
print(s_temp) // แสดงค่าอุณหภูมิที่อ่านได้ผ่านหน้าจอ
#####
```

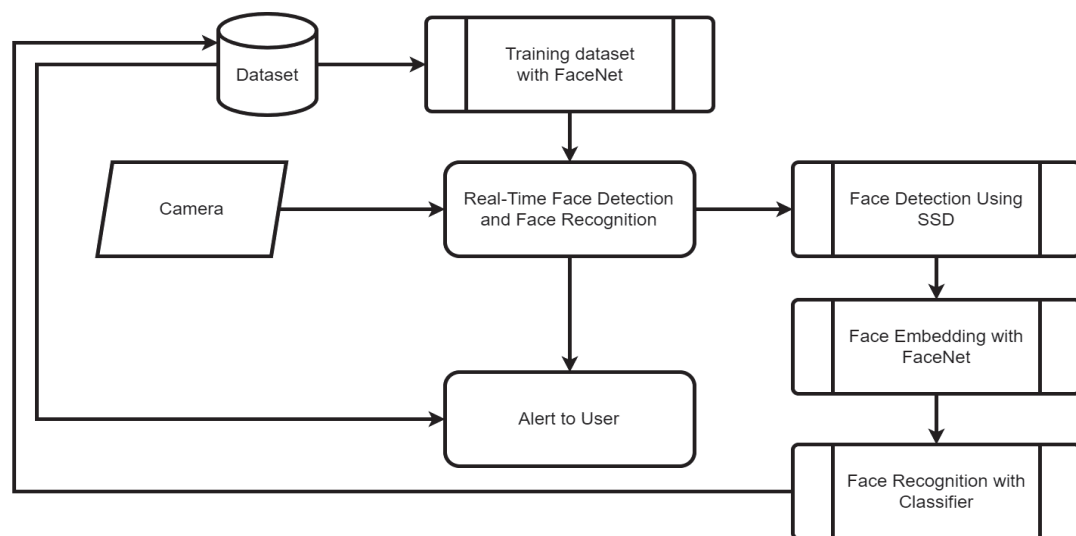
Figure 64. Coding of Face Recognition.

In a research study comparing face recognition models, including VGG, Inceptionv3, ResNet50, and EfficientNet, it was observed that ResNet50 achieved the highest facial recognition accuracy, followed by Inception_ResNet_v2. Referring to Figure 63, where all models are compared using the FaceNet learning method for recognizing faces, which is a pre-trained model, we can significantly reduce the operation time by utilizing the processing power of Inception_ResNet_v2. Table 7 shows the FaceNet learning settings used to train face recognition while wearing a mask. And euclidean distance was used to compare the embedding values of all images in the Face database, and the image with the smallest Euclidean value was considered the training answer. This training was conducted using images from 76 personnel to demonstrate.

Table 7 Model training settings for FaceNet.

Architecture	inception_resnet_v2 (FaceNet)
Image size	512*512
Batch Size	32
Epochs	100
Learning rate	0.004

To make the system more efficient in recognizing faces, the model is a Pre-train model to be trained with a database that will be used for another 100 Epochs. The structure of this model is shown in Figure 65.

**Figure 65.** Block Diagram of Face Detection and Face Recognition System.

Chapter 4

Result and Discussion

4.1 Training the face mask detection model

From the graph, it can be seen that as the training time elapses, the values of Precision and Recall approach 1. The model's accuracy increases, and it can be observed that Recall is higher than Precision. During training, the model was able to learn and detect masks very well compared to images. After training, we tested the fully trained model with a validation image dataset shown in Figure 66.

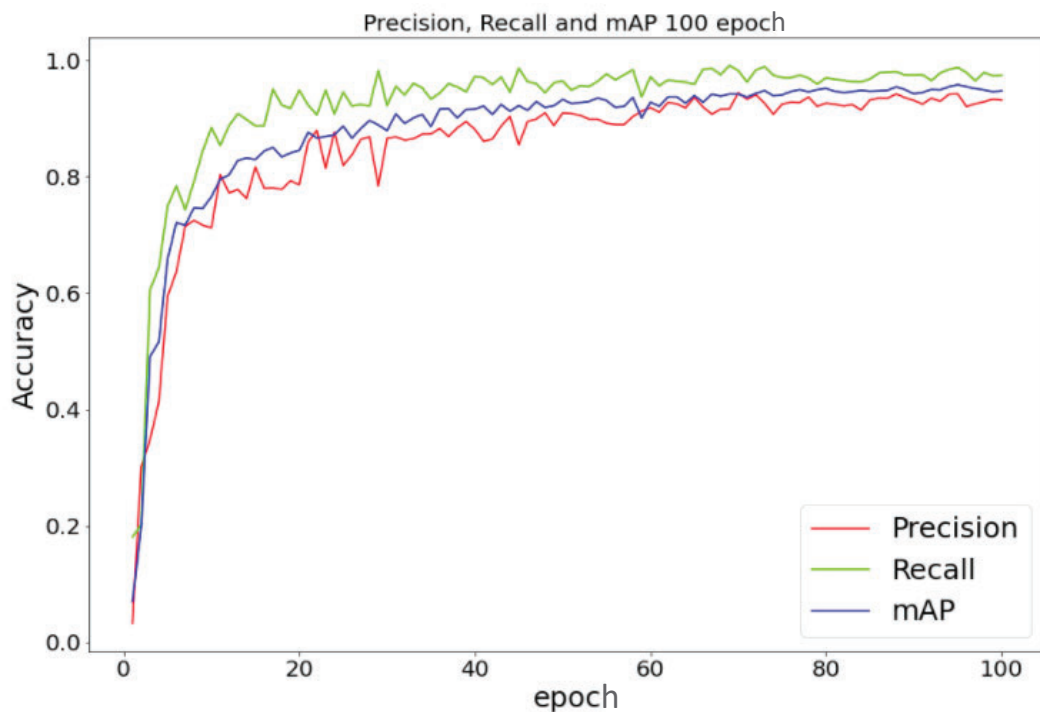


Figure 66. Showing the results of training a face detection model.

Then we convert the obtained data into a confusion matrix, which can be seen that the model was able to predict mask wearing more accurately. Prediction of faces without masks was small because of the face that does not wear a mask. There may be only a small amount of non-masked face coverings, allowing the model to predict being non-masked.

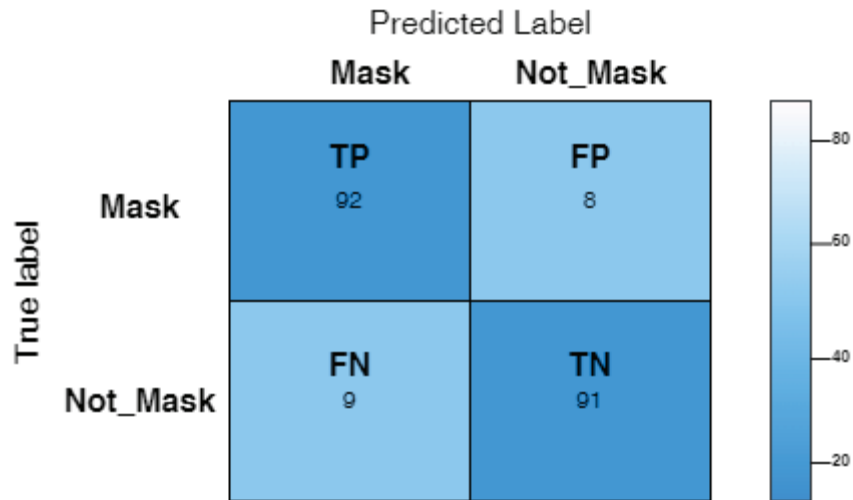


Figure 67. Confusion matrix of the face detection model.

Apparently, the model was able to successfully detect masked faces. And cannot identify other objects as masks. Models can also differentiate between images of faces with and without masks. It is important to note that the use of a large number of training images and appropriate pre-training settings are crucial for the successful learning of the mask detection model.



Figure 68. Face detection model training test results.

From the face detection results after the training, it was found that the system was able to detect the mask well. The image shown in Figure 68 is an image from the Test dataset, which is divided from the Real-World Masked Face Dataset (RMFD). It can be observed from the white bottom corner of the image. The model can detect even if the face mask is not worn correctly.

4.2 Training a face recognition model

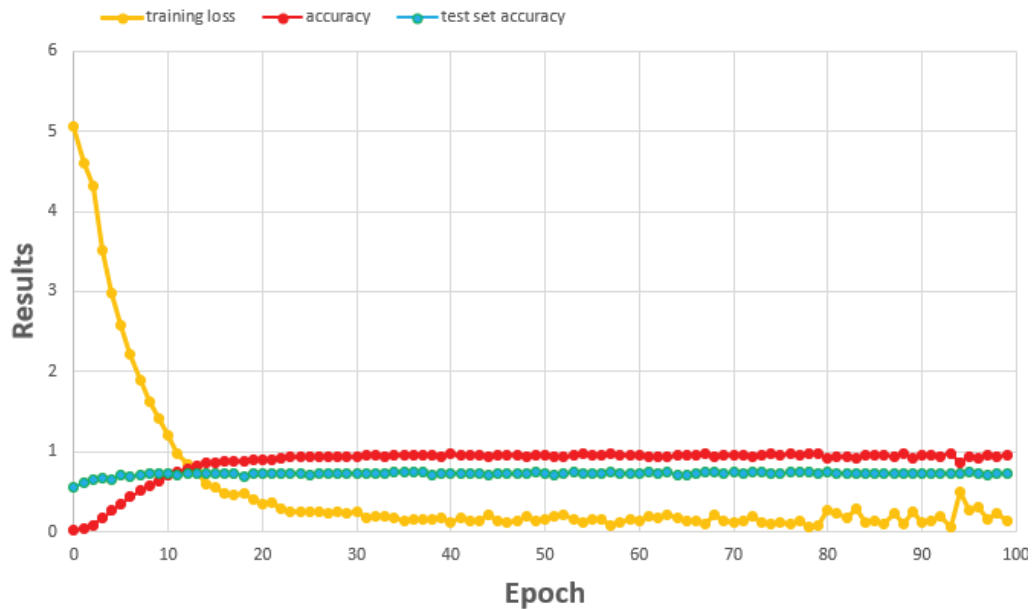


Figure 69. Results of facial recognition training.

In training the model for 100 Epochs, the system was able to distinguish between masked and non-masked faces and matched them to the unmasked faces in the test set. The training loss continued to decrease, indicating that the model was learning to match masked and non-masked faces better over time. This is reflected in the accuracy curves and test accuracy line, which approach 1 as the number of epochs increases, representing the maximum accuracy of the model. In this training, the final accuracy achieved was 0.98.

4.3 Practical testing

In the actual trial, 76 participants were tested during the mask detection phase, and the system was able to differentiate well between masked participants, accurately identifying all faces. During the face recognition process, the system can recognize faces even when participants wear masks, but there are some cases where faces cannot be recognized. Specifically, the system can recognize 64 faces but cannot recognize 12 while wearing a mask. Meanwhile, the system can recognize 70 out of 76 faces while the user is not wearing a mask, as shown in Table 8. Additionally, the temperature sensing part of the system can also accurately measure the temperature.

Table 8 Evaluation table of actual use performance from all 76 trial participants.

Participants	Face Detection		Face Recognition		Detect Temperature
	Mask	Not_Mask	Mask	Not_Mask	
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	0	0	1
5	1	1	1	1	1
6	1	1	1	1	1
7	1	1	0	1	1
8	1	1	1	1	1
9	1	1	0	1	1
10	1	1	1	1	1
11	1	1	1	1	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1
16	1	1	0	1	1
17	1	1	1	1	1
18	1	1	1	1	1
19	1	1	1	1	1
20	1	1	1	1	1
21	1	1	1	1	1
22	1	1	1	1	1
23	1	1	1	1	1
24	1	1	1	1	1
25	1	1	0	0	1
26	1	1	1	1	1
27	1	1	1	1	1
28	1	1	1	1	1

Participants	Face Detection		Face Recognition		Detect Temperature
	Mask	Not_Mask	Mask	Not_Mask	
29	1	1	0	1	1
30	1	1	0	1	1
31	1	1	1	1	1
32	1	1	1	1	1
33	1	1	1	1	1
34	1	1	1	1	1
35	1	1	1	1	1
36	1	1	1	1	1
37	1	1	1	0	1
38	1	1	1	1	1
39	1	1	1	1	1
40	1	1	1	1	1
41	1	1	1	1	1
42	1	1	1	1	1
43	1	1	1	1	1
44	1	1	0	0	1
45	1	1	1	1	1
46	1	1	1	1	1
47	1	1	1	1	1
48	1	1	1	1	1
49	1	1	1	1	1
50	1	1	1	1	1
51	1	1	1	1	1
52	1	1	0	1	1
53	1	1	1	1	1
54	1	1	1	1	1
55	1	1	1	1	1
56	1	1	1	0	1
57	1	1	1	1	1

Participants	Face Detection		Face Recognition		Detect Temperature
	Mask	Not_Mask	Mask	Not_Mask	
58	1	1	1	1	1
59	1	1	1	1	1
60	1	1	1	1	1
61	1	1	1	1	1
62	1	1	1	1	1
63	1	1	1	1	1
64	1	1	0	1	1
65	1	1	0	1	1
66	1	1	1	1	1
67	1	1	1	1	1
68	1	1	1	1	1
69	1	1	1	1	1
70	1	1	1	0	1
71	1	1	1	1	1
72	1	1	1	1	1
73	1	1	0	1	1
74	1	1	1	1	1
75	1	1	1	1	1
76	1	1	1	1	1
Total	76	76	64	70	76

The performance comparison between the system's training phases and actual test results indicate that the latter has lower accuracy due to several visual factors, such as inappropriate lighting or insufficient image data in the database, which hinder the system's ability to differentiate faces.

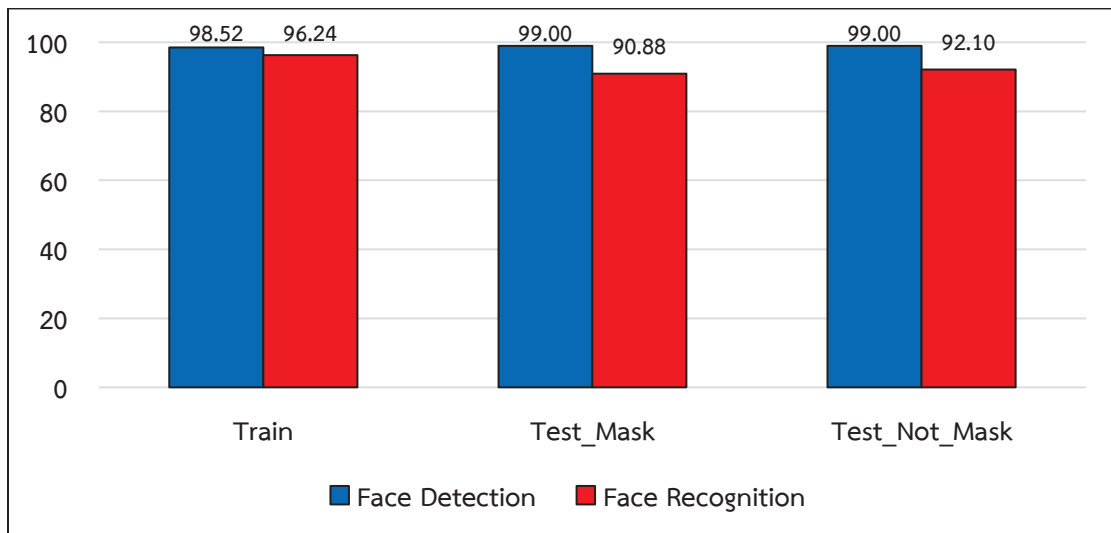


Figure 70. Comparison of training and testing data.

Figure 70 shows in the process of the face detection and recognition-training data, the model perform very well since the image is a preset image. And in the actual test, for the detection part, the face wearing mask can be detected very well. And for the recognition part, there are some limitations that cannot recognize some faces. As for recognizing faces while wearing masks, the model was able to recognize less than a face without a mask.

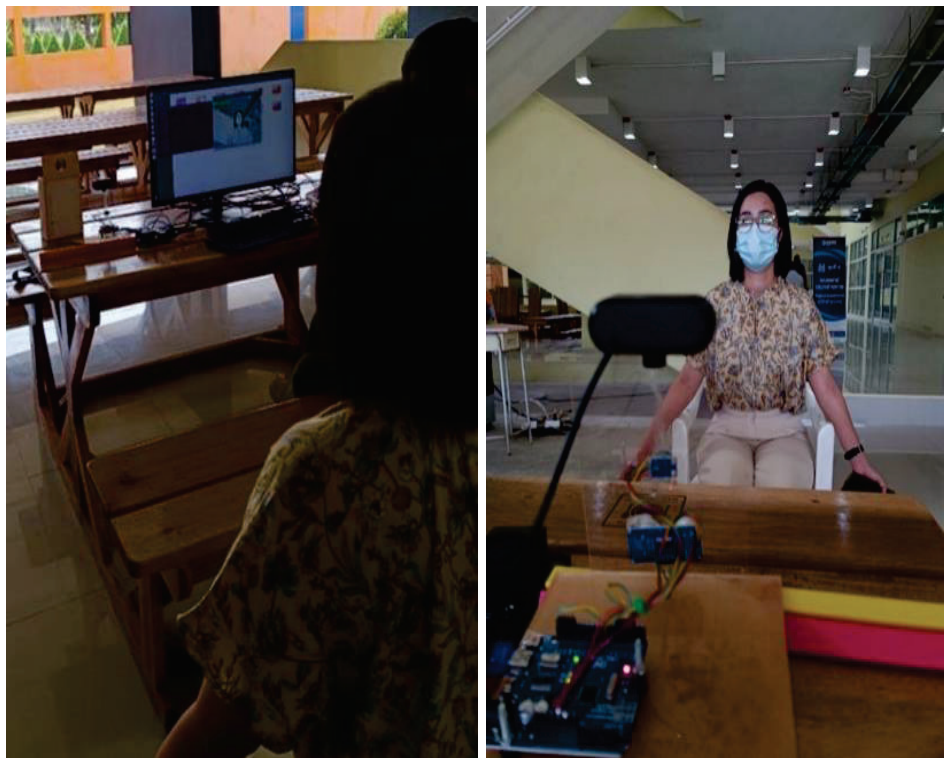


Figure 71. Real use testing.

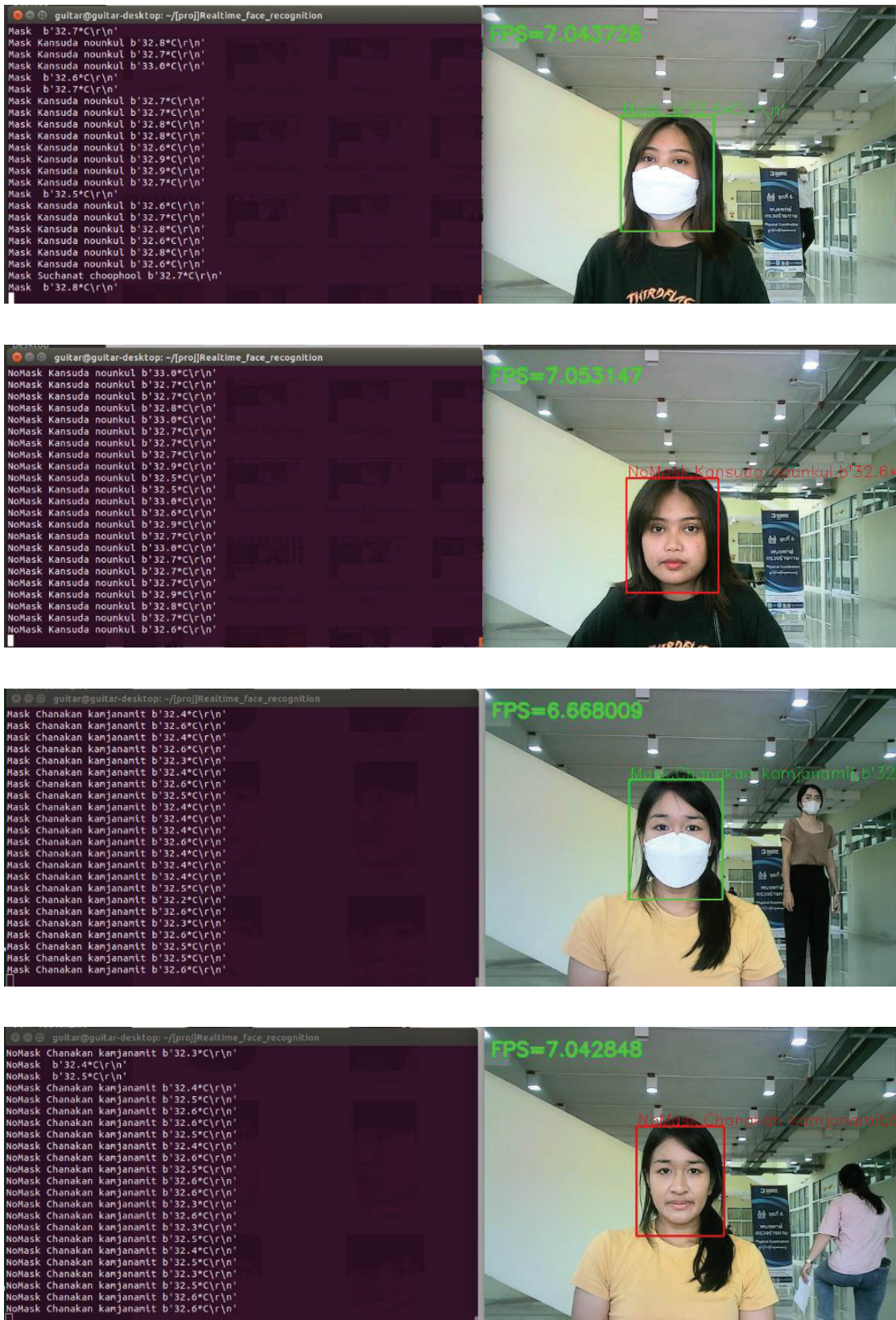


Figure 72. The system can detect masks and recognize faces correctly.

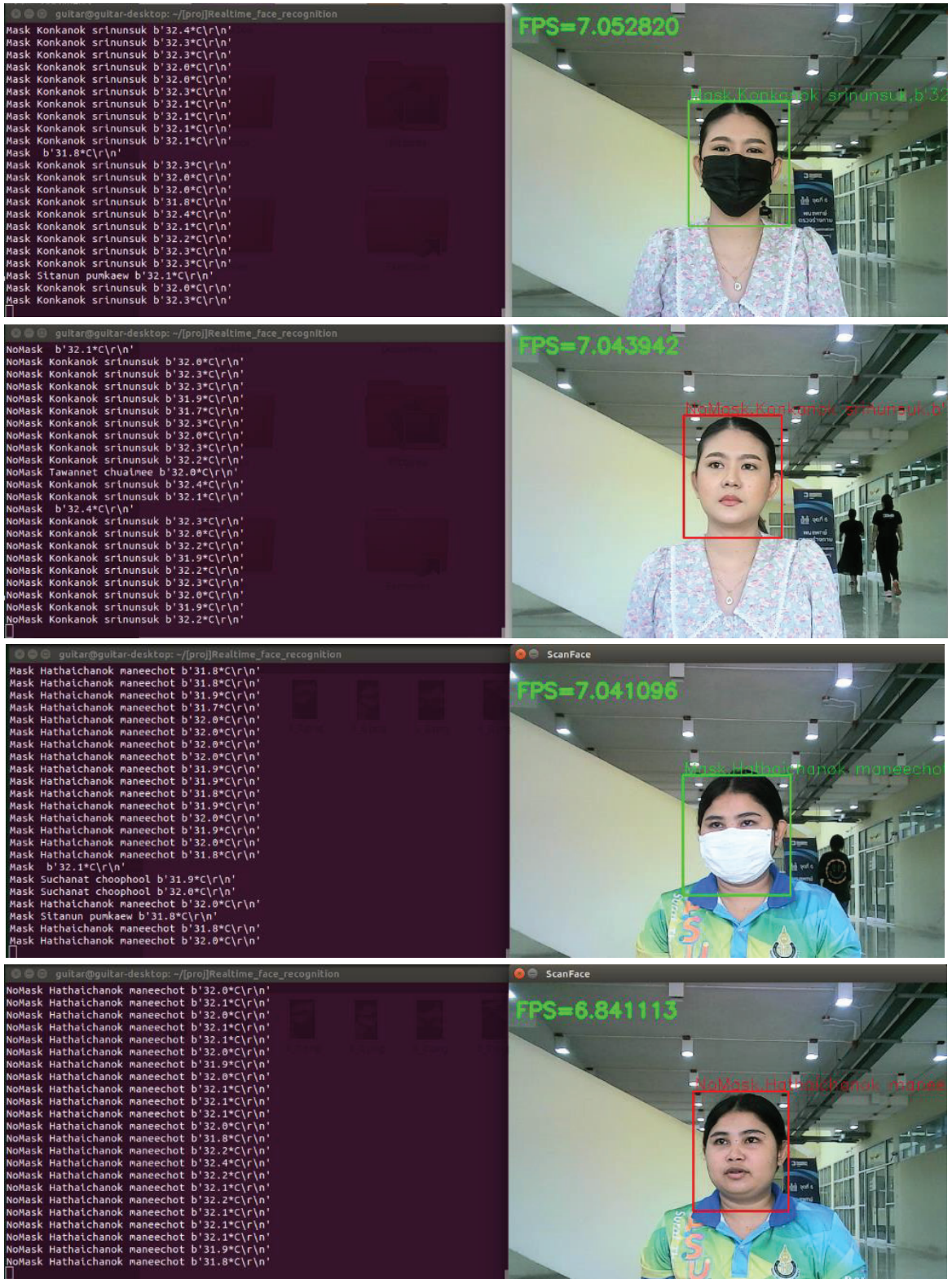


Figure 73. System error when detecting incomplete face angles.

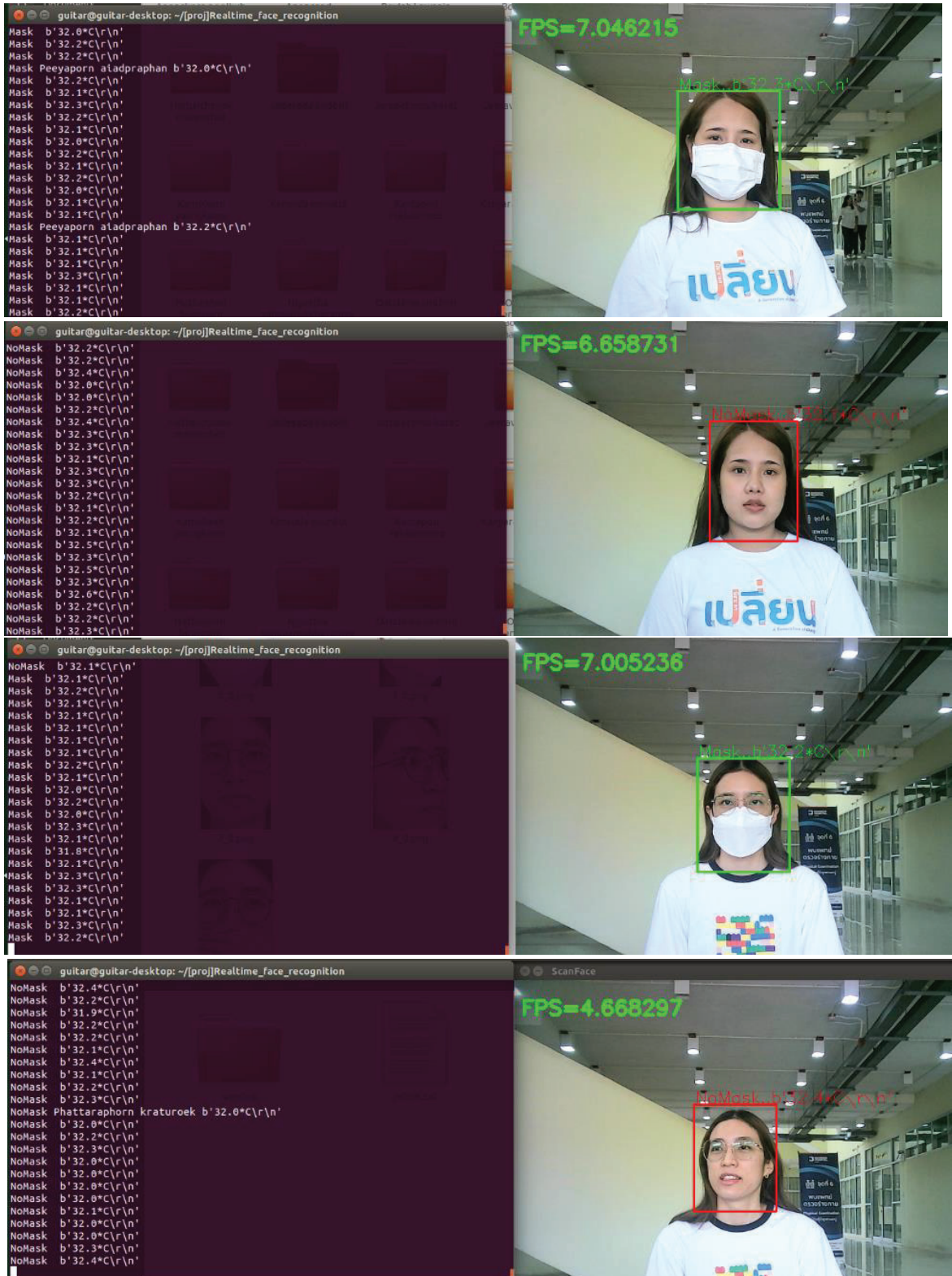


Figure 74. The system cannot recognize faces.

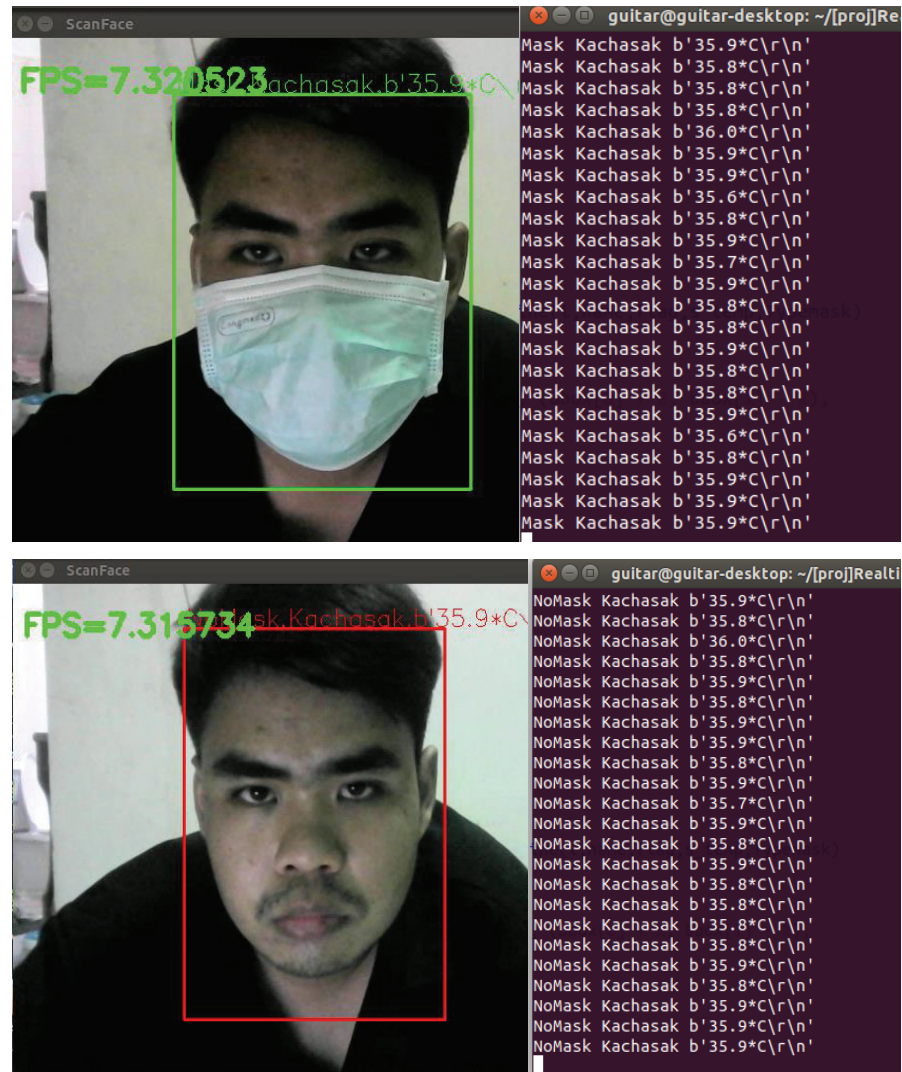


Figure 75. Test temperature detection.

4.4 Sending data to show results to users

In the process of sending detection results to the user, we have set the detection interval to every 5 seconds. The detected data, such as whether the person is wearing a mask or not and their temperature, will be sent to the user. The reason for this interval is that we do not want the system to send data more often than necessary, and we also value ease of installation and quick accessibility. Therefore, we use the LINE Application service to send information to users. The satisfactory results are shown in Figure 75, where the system can transmit data in real-time.

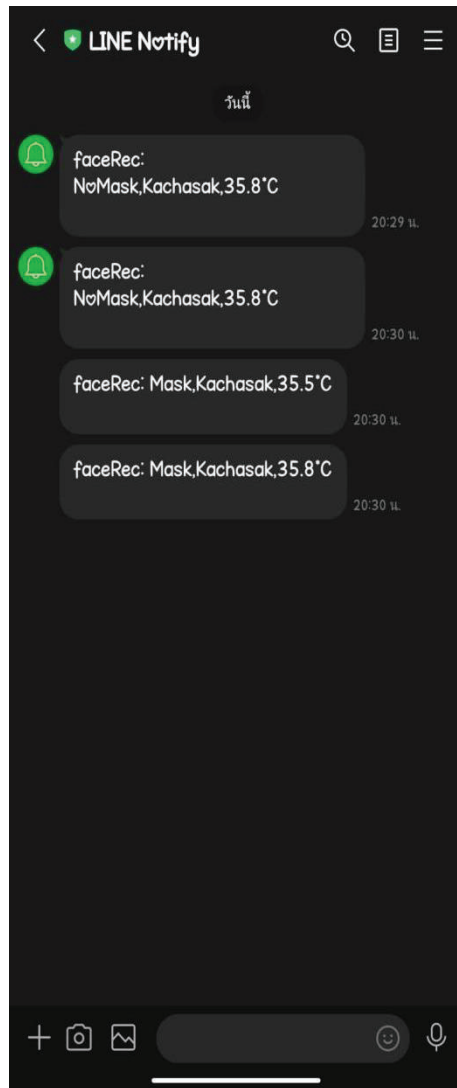


Figure 76. Sending data to the LINE application.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

For this study is to develop a mask detection system and facial recognition while wearing a mask. The aim is to create a device that can be practically used, incorporating new technologies in its operation. The study involves testing the performance of various models suitable for the mask detection system, face recognition system, and temperature detection to obtain a highly efficient system that can work in real-life situations.

We developed a Jetson Xavier NX on-board system for mask detection and facial recognition with the aim of improving attendance records at PSU Wittayanusorn Surat Thani School by replacing the current fingerprint scanning system. We reviewed different models for face and mask detection, including Haar Cascade, Single Shot Multibox Detector (SSD), Histogram of Oriented Gradients (HOG), and Multi-task Cascaded Convolutional Networks (MTCNN). We found that the Single Shot Multibox Detector was the best model for mask detection due to its speed, simplicity, and suitability for small devices. For face recognition, we evaluated the VGG, Inceptionv3, ResNet50, and EfficientNet models for extracting facial features, and we found that ResNet50 was the most accurate model. We also decided to use FaceNet as our facial recognition model, which uses Inception Resnet V2 as the principal model to extract features from images. We optimized the facial recognition system by collecting data via Google Forms with different face angles, with and without masks, for 76 individuals and trained the model to recognize those faces. We also integrated a temperature monitoring system to increase system efficiency. During testing, we found that the face mask detection system was 100% accurate in distinguishing between wearing and not wearing a mask. However, there were some face recognition errors. Out of 76 randomized participants, the system was able to identify 64, representing 90.88% accuracy. Additionally, an infrared camera connected to the system accurately measures temperature and sends the information to the user.

5.2 Future work

Currently, face detection and recognition technologies are being developed with the aspiration of improving society for better living. Apart from detecting faces, these technologies can also be used to detect small objects. Shi, Yanli, et al (2023)

applied object detection technology to detect traffic signs for smart vehicles using the YOLOv5 model, which detects three major categories of traffic signs: "road signs", "forbidden signs" and "warning signs". Similarly, Zhao, Hangyue et al. (2023) in maritime search and rescue, the YOLOv7 model was used to train object detection to detect people, boats, and other objects in open waters with an accuracy of 59%. Xu, Hongyan et al. (2022) used the Deep Spatial Pyramid Pooling (D-SPP) technology to detect the COVID-19 virus from chest x-ray images with a diagnostic accuracy of 88.12%. Furthermore, the integration of Karahan object recognition technology with object detection technology has been proposed by Mehmet et al. (2022) to detect and count moving objects and license plate recognition using a detection model Gaussian Mixture Models and Prewitt Operator for license plate recognition. Similarly, Ilyasi, Pervez Shoaib et al. used YOLO as a detection model for license plate recognition. Takayanagi, Miho, et al. proposed a relevance model for in-store product recognition and product scene recognition, and the results confirm the correct assessment of product stock status and angle type.

Finally, the next steps in the development of the system will involve its integration for use at the PSU. Wittayanusorn Surat Thani School. Firstly, a user manual will be created, and a facial data collection designed, including the gesture for collecting face images. Secondly, a function will be added to allow users to update their data in the database in case the system cannot recognize their faces. Lastly, the database will be expanded to include more users from the PSU. Wittayanusorn Surat Thani School.

References

- X. Zhu, Z. Lei, and S. Z. Li, "Out-of-Distribution Detection for Reliable Face Recognition," in *IEEE Signal Processing Letters*, vol. 27, 2020, pp. 710-714.
- W. Wang, X. Wang, W. Yang, and J. Liu, "Unsupervised Face Detection in the Dark," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, 1 Jan. 2023, pp. 1250-1266.
- Y. Martínez-Díaz, H. Méndez-Vázquez, L. S. Luevano, M. Nicolás-Díaz, L. Chang and M. González-Mendoza, "Towards Accurate and Lightweight Masked Face Recognition: An Experimental Evaluation," in *IEEE Access*, vol. 10, 2022002C pp. 7341-7353.
- STOCKMAN, George; SHAPIRO, Linda G. *Computer vision*. Prentice Hall PTR, 2001.
- S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017, pp. 1-6.
- SOO, Sander. *Object detection using Haar-cascade Classifier*. Institute of Computer Science, University of Tartu, 2014, 2.3: 1-12.
- LIU, Wei, et al. *ssd: Single shot multibox detector*. In: *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer International Publishing, 2016. p. 21-37.
- DALAL, Navneet; TRIGGS, Bill. *Histograms of oriented gradients for human detection*. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Ieee, 2005. p. 886-893.
- ZHANG, Kaipeng, et al. *Joint face detection and alignment using multitask cascaded convolutional networks*. *IEEE signal processing letters*, 2016, 23.10: 1499-1503.
- Eddie Forson. *Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning* online: <https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab> (accessed on 16 June 2022).

- Shivy Yohanandan. mAP (mean Average Precision) online: <https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2> (accessed on 16 June 2022).
- SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. (Accessed on 16 June 2022).
- BALDASSARRE, Federico; MORÍN, Diego González; RODÉS-GUIRAO, Lucas. Deep koalarization: Image colorization using cnns and inception-resnet-v2. arXiv preprint arXiv:1712.03400, 2017.
- TARG, Sasha; ALMEIDA, Diogo; LYMAN, Kevin. Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029, 2016.
- TAN, Mingxing; LE, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. PMLR, 2019. p. 6105-6114.
- Data Science Milan. An Introduction to Transfer Learning and HuggingFace online: <https://datasciencemilan.medium.com/an-introduction-to-transfer-learning-and-huggingface-cce345a97192> (accessed on 16 June 2022).
- SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 815-823.220
- Will Koehrsen. Neural Network Embeddings Explained online: <https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526> (accessed on 16 June 2022).
- Vihar Kurama. A Review of Popular Deep Learning Architectures: ResNet, InceptionV3, and SqueezeNet online: <https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/> (accessed on 18 June 2022).
- Surapong Kanoktipsatharporn. ReLU Function online: <https://www.bualabs.com/archives/1355/what-is-relu-function-why-popular-deep-learning-training-deep-neural-network-activation-function-ep-3/> (accessed on 2 Aug 2022).

- Natthawat Phongchit. Let's get to know ResNet better online: <https://shorturl.asia/f7EJj> (accessed on 2 Aug 2022).
- Aakash Nain. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks online: <https://medium.com/@nainaakash012/efficientnet-rethinking-model-scaling-for-convolutional-neural-networks-92941c5bfb95> (accessed on 2 Aug 2022).
- Christophe Pere. What are Loss Functions online: <https://towardsdatascience.com/what-is-loss-function-1e2605aeb904> (accessed on 10 Aug 2022).
- Aakash Kaushik. Understanding ResNet50 architecture online: <https://iq.opengenus.org/resnet50-architecture/> (accessed on 10 Aug 2022).
- GRAVITECH. JETSON XAVIER NX online: <https://shorturl.asia/if6wn> (accessed on 16 Aug 2022).
- Tasmota. MLX90614 infrared thermometer online: <https://tasmota.github.io/docs/MLX90614/#configuration> (accessed on 16 Aug 2022).
- Logitech. Camara Webcam 4k online: <https://www.logitech.com/th-th/products/webcams/brio-4k-hdr-webcam.960-001105.html> (accessed on 16 Aug 2022).
- Factomart. Ultrasonic Sensor online: <https://mall.factomart.com/what-is-ultrasonic-sensor/> (accessed on 16 Aug 2022).
- Sparkfun. What is an Arduino online: <https://learn.sparkfun.com/tutorials/what-is-an-arduino/all> (accessed on 16 Aug 2022).
- Google. Google Colab online: <https://research.google.com/colaboratory/faq.html> (accessed on 20 Aug 2022).
- Nvidia developer. JetPack SDK 4.6 Release Page online: <https://developer.nvidia.com/embedded/jetpack-sdk-46> (accessed on 20 Aug 2022).

- Ubuntu packages. Package: python3.6 online: <https://packages.ubuntu.com/bionic/python3.6> (accessed on 20 Aug 2022).
- Derrick Mwit. The Best ML Frameworks & Extensions for TensorFlow online: <https://neptune.ai/blog/extensions-for-tensorflow> (accessed on 20 Aug 2022).
- Ken RobotSiam. opencv-3.4.0 online: <https://rosthai.blogspot.com/2018/10/opencv-3-ros.html> (accessed on 20 Aug 2022).
- Borntodev. ฟังก์ชันการใช้ NumPy ใน Python 3 online: <https://www.borntodev.com/2020/04/16/-numpy--python-3/> (accessed on 20 Aug 2022).
- NumPy. NumPy 1.16.6 Release Notes online: <https://numpy.org/devdocs/release/1.16.6-notes.html> (accessed on 20 Aug 2022).
- Y. Guo and B. C. Wünsche, "Comparison of Face Detection Algorithms on Mobile Devices," 2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ), Wellington, New Zealand, 2020, pp. 1-6.
- T. Edirisooriya and E. Jayatunga, "Comparative Study of Face Detection Methods for Robust Face Recognition Systems," 2021 5th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI), Colombo, Sri Lanka, 2021, pp. 1-6.
- M. I. Hossain and M. S. Alam, "A Deep Learning Approach to Count people Using Facenet Architecture," 2021 Emerging Trends in Industry 4.0 (ETI 4.0), Raigarh, India, 2021, pp. 1-7.
- A. V. Savchenko, "Facial expression and attributes recognition based on multi-task learning of lightweight neural networks," 2021 IEEE 19th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2021, pp. 119-124.
- S. R. Reza, X. Dong and L. Qian, "Robust Face Mask Detection using Deep Learning on IoT Devices," 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 2021, pp. 1-6.

- VERMA, Poonam; TRIPATHI, Vikas; PANT, Bhaskar. Comparison of different optimizers implemented on the deep learning architectures for COVID-19 classification. *Materials Today: Proceedings*, 2021, 46: 11098-11102.
- P. A. K. K. Diallo and Y. Ju, "Accurate Detection of COVID-19 Using K-EfficientNet Deep Learning Image Classifier and K-COVID Chest X-Ray Images Dataset," 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 2020, pp. 1527-1531.
- Y. Guo and B. C. Wünsche, "Comparison of Face Detection Algorithms on Mobile Devices," 2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ), Wellington, New Zealand, 2020, pp. 1-6.
- T. Edirisooriya and E. Jayatunga, "Comparative Study of Face Detection Methods for Robust Face Recognition Systems," 2021 5th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI), Colombo, Sri Lanka, 2021, pp. 1-6.
- M. I. Hossain, Sama-E-Shan and H. Kabir, "An Efficient Way to Recognize Faces Using Mean Embeddings," 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2021, pp. 1-10.
- A. V. Savchenko, "Facial expression and attributes recognition based on multi-task learning of lightweight neural networks," 2021 IEEE 19th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2021, pp. 119-124.
- S. R. Reza, X. Dong and L. Qian, "Robust Face Mask Detection using Deep Learning on IoT Devices," 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 2021, pp. 1-6.
- MONSHI, Maram Mahmoud A., et al. CovidXrayNet: Optimizing data augmentation and CNN hyperparameters for improved COVID-19 detection from CXR. *Computers in biology and medicine*, 2021, 133: 104375.
- P. A. K. K. Diallo and Y. Ju, "Accurate Detection of COVID-19 Using K-EfficientNet Deep Learning Image Classifier and K-COVID Chest X-Ray Images Dataset," 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 2020, pp. 1527-1531.

- . Adnan Othman and I. Aydin, "A Low-Cost Embedded Security System for UAV-Based Face Mask Detector Using IoT and Deep Learning to Reduce COVID-19," 2022 International Conference on Decision Aid Sciences and Applications (DASA), Chiangrai, Thailand, 2022, pp. 693-697.
- D. -L. Nguyen, M. D. Putro and K. -H. Jo, "Facemask Wearing Alert System Based on Simple Architecture with Low-Computing Devices," in IEEE Access, vol. 10, 2022, pp. 29972-29981.
- S. Ruparelia, M. Jethva and R. Gajjar, "Real-time Face Mask Detection System on Edge using Deep Learning and Hardware Accelerators," 2021 2nd International Conference on Communication, Computing and Industry 4.0 (C2I4), Bangalore, India, 2021, pp. 1-6.
- I. Jovović, D. Babić, S. Čakić, T. Popović, S. Krčo and P. Knežević, "Face Mask Detection Based on Machine Learning and Edge Computing," 2022 21st International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2022, pp. 1-4.
- S. Ruparelia, M. Jethva and R. Gajjar, "Real-time Face Mask Detection System on Edge using Deep Learning and Hardware Accelerators," 2021 2nd International Conference on Communication, Computing and Industry 4.0 (C2I4), Bangalore, India, 2021, pp. 1-6.
- Xiaoguang Lu and A. K. Jain, "Automatic feature extraction for multiview 3D face recognition," 7th International Conference on Automatic Face and Gesture Recognition (FGR06), Southampton, UK, 2006, pp. 585-590
- Y. Shi, X. Li and M. Chen, "SC-YOLO: A Object Detection Model for Small Traffic Signs," in IEEE Access, vol. 11, 2023, pp. 11500-11510.
- H. Zhao, H. Zhang and Y. Zhao, "YOLOv7-sea: Object Detection of Maritime UAV Images based on Improved YOLOv7," 2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW), Waikoloa, HI, USA, 2023, pp. 233-238.
- H. Xu, D. Wang and A. Sowmya, "Multi-scale alignment and Spatial ROI Module for COVID-19 Diagnosis," 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 2022, pp. 1-9.

- M. Karahan, H. Kurt and C. Kasnakoglu, "Moving Object Detection and Counting in Traffic with Gaussian Mixture Models and Vehicle License Plate Recognition with Prewitt Method," 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2022, pp. 32-36.
- P. S. Ilyasi, G. Gupta, M. S. Sai, K. Saatwik, B. S. Kumar and D. Vij, "Object-Text Detection and Recognition System," 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART), MORADABAD, India, 2021, pp. 539-545.

APPENDIX

ITC-CSCC2022

278

COVID-19 Face Mask Detection and Identification using FaceNet

1st Kachasak Intim
*Faculty of Science and Industrial
 Technology*
*Prince of Songkla University,
 Surat Thani Campus,
 Surat Thani, Thailand*
 6340320501@psu.ac.th

2nd Apirat Wanichsombat
*Faculty of Science and Industrial
 Technology*
*Prince of Songkla University,
 Surat Thani Campus,
 Surat Thani, Thailand*
 apirat.w@psu.ac.th

3rd Nathaphon Boonnam
*Faculty of Science and Industrial
 Technology*
*Prince of Songkla University,
 Surat Thani Campus,
 Surat Thani, Thailand*
 nathaphon.b@psu.ac.th

Abstract— All over the world, the COVID-19 pandemic has alarmed humans for the past two years. The spread of this virus continues to mutate and more. It makes preventing the spread of the epidemic a challenge. Therefore, we need to plan to deal with more mutations in the future. We consider the current time attendance record. Many organizations still use the old-time attendance method, namely fingerprint scanning, offering a method for detecting faces and identifying the faces of mask wearers. This study proposes constructing a mask detection system using a single shot multi-box detector as a pre-train model to distinguish masks and compare faces to identify personnel with FaceNet trained using facial data from school personnel. The system is practical in face detection and face comparison, accurate at 98%.

Keywords— *COVID-19, Mask Detection, Face Recognition, FaceNet, Single-shot Multi-box Detector.*

I. INTRODUCTION

Many years ago, object detection technology has been developed with greatly increased accuracy. As a result, facial recognition technology is gaining attention and playing an active role in today's society and business. The public and private sectors have widely adopted this object detection technology in their organizations, including counting people in and out, detecting people's moods at work or detecting masks, etc., in the COVID-19 outbreak. A government agency organizes society to ensure safety and reduce its spread [1]. It is not mandatory to wear masks in public. It is effective in protecting users and the people around them. On the contrary, most people are vaccinated against germs, but people cannot take off the mask-like before. Therefore, the vaccines that people in the country currently receive cannot protect against the virus. People, who have recovered from infection, can be reinfected because of the mutation of this virus. This makes people insecure about taking off their masks, so they have to wear masks in everyday life.

Recording time in and out of current jobs in most government organizations still uses fingerprint recording. We, therefore, see that in the situation of the spread of the COVID-19 virus, transmitting through contact or being in close proximity to each other only 2 meters. The traditional entry-exit time recording system is a breeding ground for disease. And there is a high risk if it is not disinfected before use or cleaned. Therefore, technology that can develop a method for recording time in-out is better than before and is suitable for the current situation. An object detection technology is used to detect a person's face, whether they are wearing a mask or not. It can be decreased the spread of the virus, but wearing a mask makes it very difficult to identify. Because the person's face is half-closed, it cannot detect

much important information on the face. We have to find new ways to develop a more efficient facial recognition system in the nose and mouth area.

The whole process uses convolutional neural networks (CNN) in terms of object detection. This technology adapts to various human needs such as agriculture in [2] industry in [3] medicine in [4] and many others. We are now focusing on applications related to COVID-19. This paper presents a facial recognition system for those who wear masks and those who do not wear masks. This study aims to create a system for detecting the masked wear of school personnel and identifying them by their faces while wearing masks.

II. RELATED WORK

Based on results from [5], [6], [7], faces were partially detected while wearing a mask and applied to the identification represented by Shilpa Sethi [8]. As we mentioned above, we are interested in implementing a detection model such as a single-shot multi-box detector (SSD) in [9] which is very popular in object detection applications. In particular, J. Talahua [10] studies FaceNet-based facial recognition in [11] that can accurately identify a person.

A. Single Shot Multi-Box Detector

SSD operation takes a single image to detect multiple objects within the image. SSD object detection consists of a separate feature tree and uses the formation filter to find architecture-based SSD objects. VGG16 in [12] extracts image feature maps. Objects are then detected using Conv4/3 layers. Notice that SSDs detecting multiple objects simultaneously in a single process can map multiple property levels, with the SSD adding six more layers on VGG16. Overall, the SSD made 8,732 predictions using six layers, as shown in Fig. 1. As we can see on the arrows of each layer, the front layer is sent to the final object detection.

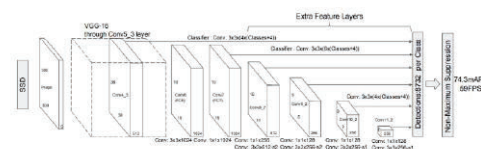


Fig. 1. SSD working process.

When it comes to SSD usage, as seen in [13], MobileNetV2 SSD models are compared to LeNet in [14], AlexNet in [15], VGG16 and ResNet-50 in [16], detecting a self-created database mask called SSDMobileNetV2. It is the best done with 15.71 frames per second (FPS) detection speed

and 92.64% accuracy. It was also in [17] with this SSD used on the Raspberry Pi board in [18] detection at 28.07 FPS and accuracy of 91.7%. It can be seen that this model is suitable for modifications, whether it is a PC or a microcontroller board. A 30 FPS detection speed is the default speed to count as real-time detection.

B. FaceNet

For face recognition, we are interested in using FaceNet [19] even if it is not the best version currently. But it is the most widely accepted form of research. Before, the image is involved in the process of extracting image properties. We use a multitask cascaded convolutional network (MTCNN) algorithm in the face alignment process. This can greatly reduce the processing time. Since it shrinks the image and selects only the necessary parts, the FaceNet has Inception ResNetV1 as its main architecture, as shown in Fig. 2. From another study in ResNetV1 [20], it was performed face recognition experiments using FaceNet and finding faces with images using MTCNN. Working on the Jetson TX2 board, the system recognized the faces of test participants. It has an accuracy of 97%

The way of FaceNet works is to create an embed feature from an image of a face in the Euclidean area, and it can be used to search for groups of similar faces as shown in Fig. 2. as embedded features. They can be obtained by using a deep neural network. The current pre-trained model has Inception ResNetV1 [21] as the core architecture of FaceNet. We can modify the architecture appropriately before embedding it through another function, L2 normalization. It reduces the distance between similar embeddings and in Euclidean space. FaceNet uses the triplet loss function. Loss calculations are divided into an anchor, the face used in the comparison; vivacious is the face that is the same as an anchor; and harmful is the face that is a window leaf with an anchor. In the case of an individual same, the distance between the anchor and the positive is reduced. The two faces are not the same, so the distance between the anchor and the negative increases. In addition, machine learning techniques such as support vector machine (SVM) [22] or K-nearest neighbor (KNN) algorithm in [23] can be used to embed features from FaceNet to speed up the grouping of faces on demand.

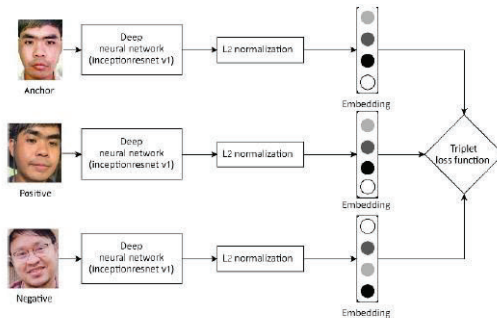


Fig. 2. FaceNet workflow.

III. MATERIALS AND METHODS

To develop the system, we used TensorFlow 1.13, Python 3.6.9, OpenCV 3.4.2, NumPy 1.16.1, and Matplotlib 3.1.1 and developed it on the PyCharm package. A person's face under the mask is building a database using Google Forms to

create forms for collecting images from users. For face detection, we used the SSD per-train model to differentiate between masked and non-masked. For face recognition, we applied FaceNet's Inception ResNetV1 to train the model to identify people wearing masks, as shown in Fig. 3.

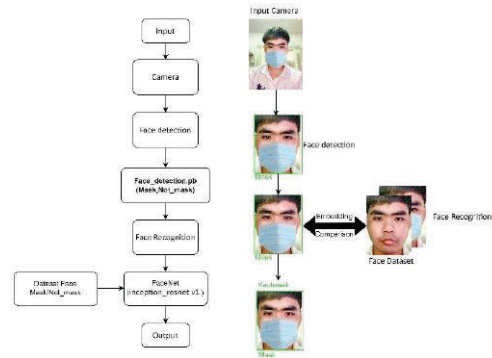


Fig. 3. Workflow of image data used in the System.

The first step in storing images is to include images stored in a folder using the folder name as the username. Each user has a total of seven pictures. We used a total of 76 users, for a total of 532 faces. The next step was to make the faces smaller to simplify the process. In extracting image properties, if the processed image contains only facial structures such as eyes, nose, and mouth, the process of extracting image properties is more efficiently detected. We have set the face alignment to 20 (margin = 20) because we need a little background from the original image. If the margin is equal to zero, there will be no background images left. To save time in managing image data, we used the method of adding a mask to the image of the face. In this step, we applied a face landmark detection model from Dlib. The model diagram creates a point at the edge of the face structure. We considered the areas around the nose and mouth to add a mask. We only need a face that is not hidden from the user, as shown in Fig. 4.

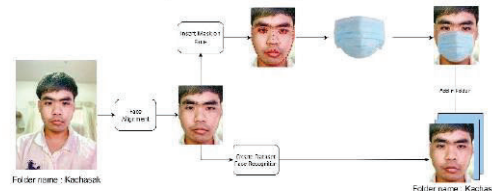


Fig. 4. Face alignment

From adding a mask to a picture of a face, the amount of image data increased to 14 faces per user, a total of 1064 images, and the image size was 1280x960. In the practice of facial recognition, we collected photos of people with masks and without masks, using datasets with and without masks to train. The dataset without a mask is a set of tests. We set the batch size at 64 and the number of training cycles at 100 epochs for training the model. Fig. 5 shows the working of the model training.

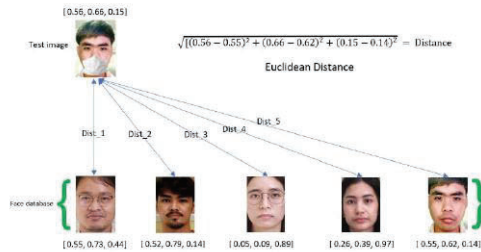


Fig. 5. Measuring distance in training.

To determine the accuracy of a training model is to use the test dataset by randomly selecting faces. In the test dataset, one face is the test image, and the other 10 faces consist of 9 different faces and one full face. The same person is called the support set. The model predicts the distance between the test image and each image in the support set. It is considered correct if the model predicts the least distance between the same pages. Then, repeating for the specified number of training cycles, the values obtained from each training cycle will be the model's accuracy. Finally, the step is to take a photograph obtaining information only from a face detection process, which receives information only for masked or masked faces. In comparing images sourced from a database, FaceNet uses them to retrieve images from the database for comparison. In the last step, it can be seen that masked images are also added to the database, allowing FaceNet to recognize images in the database. Face detection distinguishes between face detection and face recognition. Face detection detects a face and determines whether it is a mask or not. The working part of face recognition compares only the images in the database to find a matching face by embedding values representing facial features and using them to compare images from the lowest distance from the database using Euclidean distance as shown in Fig. 5.

IV. RESULTS

After the actual test, the results showed that the system can detect faces while wearing or not wearing a mask. From Fig. 6, we can see that the green box indicates the mask is correctly applied. And the red box indicates that the mask is wrong. Mask and Not_Mask keywords and display the face comparison name as the person's name being detected with a percentage of accuracy.

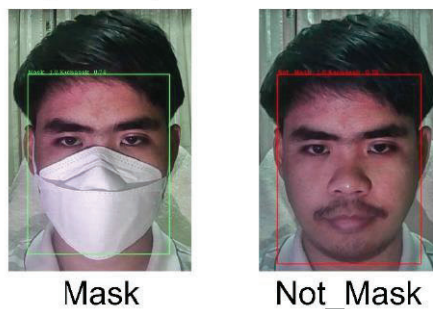


Fig. 6. Example of the results.

In data training, the system compared the training faces to the faces used in the test. The 100 sessions of training results as shown in Fig. 7 show that the training loss line approaches 0 relative to the approaching training accuracy curve at 0.98. It shows that the test loss line approaches 0 and is less than the training loss line relative to the test precision line less than the training precision line, which is estimated to be 0.74. This is normal if the test accuracy value is less than the training value. Fig. 6 depicts the actual test condition. In summary, the system's face comparison is highly accurate. It works when combined with a pre-training model used for mask detection.

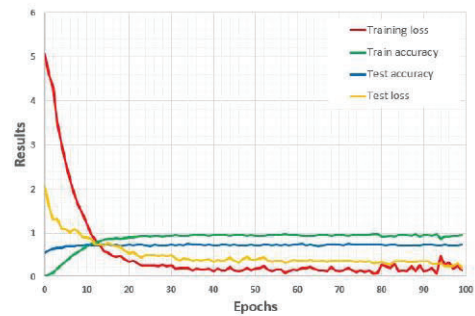


Fig. 7. System performance results.

V. CONCLUSIONS AND FUTURE WORKS

This paper was created to reduce social distance and shift from traditional employee attendance records. It has become a facial recognition system that can recognize the faces of those with and without masks. In terms of data collection, the current state of online data collection is convenient for users and also saves time in collecting data. We used Google Forms as a medium for collecting image data. From users which our system was divided into two parts: to detect masks, we use a pre-train model of SSD, the detection results are very accurate. The system can distinguish between with-mask and without-mask very well. For face recognition, we use FaceNet by taking the collected Google Forms images and adding masks to all of them. When all the images are trained for 100 epochs, the result is that the system can recognize the person. While using real-time, our system may encounter some errors with uncontrollables, especially the brightness and sharpness of the user's camera. Finally, this work was supported by the Graduate Research Development Project from the National Research Council of Thailand in the fiscal year 2021 and Prince of Songkla University, Surat Thani Campus for financial support for this trial. The authors express their support for this experiment.

REFERENCES

- [1] Thirumalaisamy P Velavan and Christian G Meyer. "The COVID-19 epidemic," *Tropical medicine international health*, 2020, vol.25, 3: 278.
- [2] A. Kamilaris and F. X. Prenafeta-Boldú. "A review of the use of convolutional neural networks in agriculture," *The Journal of Agricultural Science*, 2018, vol.156, no.3, pp.312-322.
- [3] Zhi Yang, Wei Yu, Pengwei Liang, Hanqi Guo, Likun Xia, Feng Zhang, Yong Ma, and Jiayi Ma. "Deep transfer learning for military object recognition under small training set condition," *Neural Computing and Applications*, 2019, vol.31, pp.6469-6478.
- [4] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, 2018, vol.9, pp.611-629.

- [5] Amit Chavda, Jason Dsouza, Sumeet Badgujar, and Ankit Damani. "Multi-stage cnn architecture for face mask detection," 2021 6th International Conference for Convergence in Technology (I2CT), 2021, pp. 1-8.
- [6] Xinqi Fan and Mingjie Jiang. "RetinaFaceMask: A Single Stage Face Mask Detector for Assisting Control of the COVID-19 Pandemic," 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2021. pp.832-837.
- [7] Erick Sebastián Lozano Roa, Juan Sebastián Urrea López, and Isai Daniel Chacón Silva. "Real time face mask detection with SSD," 2021 IEEE 2nd International Congress of Biomedical Engineering and Bioengineering (CI-IB&BD), 2021, pp.1-4
- [8] Shilpa Sethi, Mamta Kathuria, and Trilok Kaushik. "Face mask detection using deep learning: An approach to reduce risk of Coronavirus spread," Journal of biomedical informatics, vol.120, 2021: 103848.
- [9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single shot multibox detector," European conference on computer vision, Springer, Cham, 2016. pp. 21-37.
- [10] Jonathan S. Talahua, Jorge Buele, P. Calvopiña, and José Varela-Aldás. "Facial Recognition System for People with and without Face Mask in Times of the COVID-19 Pandemic," Sustainability, 2021, vol.13, no.12: 6900.
- [11] Florian Schroff, Dmitry Kalemichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823.
- [12] Hussam Qassim, Abhishek Verma, David Feinzimer. "Compressed residual-VGG16 CNN model for big data places image recognition," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2018, pp. 169-175.
- [13] Preeti Nagrath, Rachna Jain, Agam Madan, Rohan Arora, Piyush Kataria, and Jude Hemanth. "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2," Sustainable cities and society, 2021, vol.66, 2021: 102692.
- [14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 1998, vol.86, no.11: 2278-2324.
- [15] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size," arXiv preprint arXiv:1602.07360, 2016.
- [16] Sasha Targ, Diogo Almeida, and Kevin Lyman. "Resnet in resnet: Generalizing residual architectures," arXiv preprint arXiv:1603.08029, 2016.
- [17] Shashi Yadav. "Deep learning based safe social distancing and face mask detection in public areas for covid-19 safety guidelines adherence," International Journal for Research in Applied Science and Engineering Technology (IJRASET), 2020, vol.8, no.7, pp.1368-1375.
- [18] Eben Upton and Gareth Halfacree. "Raspberry Pi user guide," John Wiley & Sons, 2014.
- [19] I. William, D. R. Ignatius Moses Setiadi, E. H. Rachmawanto, H. A. Santoso, and C. A. Sari. "Face recognition using FaceNet (survey, performance test, and comparison)," 2019 Fourth International Conference on Informatics and Computing (ICIC), 2019, pp. 1-6.
- [20] Edwin Jose, Greeshma M., Mithun T. P. Haridas, and M.H. Supriya. "Face Recognition based Surveillance System Using FaceNet and MTCNN on Jetson TX2," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019, pp.608-613.
- [21] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning," In: Thirty-first AAAI conference on artificial intelligence, 2017.
- [22] S.V.M. Vishwanathan and M. Narasimha Murty. "SSVM: a simple SVM algorithm," Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290), 2002, pp.2393-2398.
- [23] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. "KNN model-based approach in classification," OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", 2003, pp.986-996.

VITAE

Name Mr.Kachasak Intim

Student ID 6340320501

Educational Attainment

Degree	Name of Institution	Year of Graduation
B.Sc (Information Technology)	Prince of Songkla University	2019

Scholarship Awards during Enrolment

1. Graduate tuition research fund fiscal year 2020 from National Research Agency

List of Publication and Proceeding

K. Intim, A. Wanichsombat and N. Boonnam, "COVID-19 Face Mask Detection and Identification using FaceNet," 2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Phuket, Thailand, 2022, pp. 1-4, doi: 10.1109/ITC-CSCC55581.2022.9894928.

