
त्रिभुवन
विश्वविद्यालय
TRIBHUVAN
UNIVERSITY ORGANISED 1956 A.D.
INCORPORATED 1959 A.D.
KATHMANDU
NEPAL

ACKNOWLEDGEMENT

I sincerely acknowledge with gratitude all the people's assistance that made my study a success. My special thanks go to my Principal Prof. **Dr. Jivandhar Gnawali**, BCA Coordinator as well as supervisor **Mr Bhupendra Ram Luhar** and Lecturer **Mr. Bhoj Raj Joshi** whose guidance, constructive suggestions, and encouragement that greatly contributed to my completing this report.

I am thankful to TU for giving me this opportunity to broadening my theoretical knowledge and putting it into practical use.

Rukmagat Kandel

Table of Contents

References

LIST OF ABBREVIATIONS

PY–Python

TK- tkinter

SVM – Support Vector Machine

PCA – Principle Component Analysis

UI – User Interface

UX – User Experience

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

The face mask detection system is an application that utilizes computer vision and machine learning techniques to automatically detect whether individuals in video frames are wearing face masks. It has gained significant attention and importance in recent times due to the global COVID-19 pandemic and the need to ensure adherence to safety protocols. [1]
The system aims to contribute to public health and safety by providing an automated and efficient method of monitoring and enforcing face mask usage. By leveraging the capabilities of artificial intelligence, the system can analyze images or video streams in real-time and identify whether individuals are properly wearing masks, improperly wearing masks, or not wearing masks at all.

The detection of face mask in Face Mask Detection is supported by Support Vector Machine (SVM). SVM are a set of supervised learning methods used for classification, regression and outliers detection. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

This System used OpenCV for real time computer vision which supports multiple platforms, including Windows, Linux, macOS, iOS, and Android. OpenCV's modular structure allows developers to easily incorporate its functionality into their projects, making it a valuable tool for tasks ranging from simple image processing to complex computer vision applications. In summary, OpenCV is a versatile and powerful computer vision library that provides developers with a wide range of tools and algorithms to process and analyze images and videos.

This system uses a Harr Cascade algorithm for face detection. It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001. The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them. [2]

1.2. Problem Statement

The problem in face mask detection system lies in automatically identifying individuals wearing face masks correctly, improperly, or not wearing masks at all. The system must perform multi-class classification on video frames, distinguishing between "Wearing Face Mask Correctly," "Wearing Face Mask Improperly" and "Not Wearing Face Mask" categories. It should achieve high accuracy in real-time processing, handling varying lighting conditions, different mask types, and diverse demographics. Privacy considerations are essential, ensuring no personal data or images are stored. The system should offer a user-friendly interface for easy deployment and integration with existing surveillance or security systems. By efficiently enforcing mask mandates and promoting public health, the system contributes to controlling the spread of contagious diseases in crowded places like airports, shopping malls, and public transportation.

1.3. Objective

- To identify whether individuals are wearing face masks.
- To provide real-time notifications.

1.4 Scope and Limitations

1.4.1 Scope of Project

- The system is useful for academic institutions, hospitals & commercial organizations
  Automatic alert system makes people alert to wear mask.
- Reduce spread of air borne diseases.
- Mass surveillance improves smarter border control

1.4.2 Limitations

- There are some limitations related to the project:
- It cannot correctly classify partially hidden faces.
- The irregularities in images, like those with insufficient light need proper attention.

1.5 Development Methodology

Considering that the proportion of face masks in the image is often not uniform in size, they often do not occupy the whole image. The features at the highest level have relatively wealthy semantic information, the candidate region’s features can only be obtained by pooling the

target region in the last convolution layer. Therefore, in order to solve the problem of multiscale detection, the feature pyramid network / waterfall model is introduced that takes a singlescale image of an arbitrary size as input, and outputs proportionally sized feature maps at multiple levels.

Dataset Creation: The dataset was created by using harr cascade algorithm and stored in array as numpy.

Training a model to detect face masks: A default OpenCV module was used to obtain faces followed by training a harr_cascade to identify face mask.

Detecting the person not wearing a mask: A open CV model was trained to detect the people who are not wearing masks.

1.6 Organization of Report

The material presented in the project is organized into five chapters.

Chapter 1 represents the problem statement, objectives, scope and limitations of the project.

Chapter 2 describes the fundamental theories and concepts as well as information about existing system, journals and references.

Chapter 3 summarizes the keynote on system analysis and design where description of use case diagram, performance and reliability, different feasibility analysis, diagrams, dataset as well as class diagram are set out.

Chapter 4 provides an account on implementation and testing, tools used for preparation of the project. Test cases for unit testing as well as integration testing are done. Implementation details of modules are traced.

Chapter 5 presents brief summaries of outcome of the project, conclusion, reviews as well as future recommendations, improvements that can be done on upcoming days and feedback of systems, stability of the project. The implementation of desktop-based application in the project are described.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Face mask detection systems have gained significant importance during the COVID-19 pandemic, as the wearing of masks in public places has become a crucial measure to mitigate the spread of the virus. These systems rely on computer vision and machine learning techniques to automatically detect whether individuals are wearing face masks correctly or not. The process involves face detection algorithms, such as Haar cascades, Viola-Jones, SSD, or Faster R-CNN, to locate and extract faces from images or video frames. Subsequently, machine learning models, particularly Convolutional Neural Networks (CNNs), are utilized to classify whether the detected face is wearing a mask or not. Transfer learning using pre-trained models like VGG, ResNet, or MobileNet can be employed to enhance model performance.

Support Vector Machine (SVM) is another machine learning algorithm that has been applied to face mask detection. SVM is a binary classification algorithm that works well with linearly separable data and can handle high-dimensional feature spaces effectively. In the context of face mask detection, SVM can be used as an alternative to CNNs for binary classification tasks. Feature extraction techniques, like Histogram of Oriented Gradients (HOG) or Local Binary Patterns (LBP), are often employed to represent face images as feature vectors for SVM classification. SVM has shown promise in face mask detection, especially when computational resources are limited, and it can achieve reasonable accuracy with smaller datasets. [3] To train accurate face mask detection models, large and diverse datasets have been curated during the pandemic, which include samples of people wearing masks correctly, incorrectly, and without masks. The successful implementation of these systems contributes to the enforcement of mask-wearing protocols in various public spaces, improving public health and safety measures. Whether using CNNs or SVM, these face mask detection systems play a crucial role in automating the process of monitoring mask compliance and reducing the burden on human personnel in public settings.

2.2 Literature Review

For this project, research and reviews some of the related websites and applications are done. In 2001, Viola and Jones proposed a real-time object detection called the Viola-Jones object

detection framework. This process can determine competitive object detection rates in realtime to solve various detection problems. Moreover, it was used primarily in face 6 detection. Although this algorithm was very slow in training, it could detect faces in realtime with impressive speed. The method divided the image into many smaller sub-regions and required checking many different positions and scales because an image had many faces of different sizes. An effective and economic approach to the use of AI in a manufacturing setting to build a secure environment. Using a face mask detection dataset, the system will use Open CV to perform real-time face detection from a live stream from our webcam. Using harr cascade , SVM, Python, and Open CV, and, it will build a COVID-19 face mask detector with computer vision. Using computer vision and CNN, it is aimed to decide whether or not the person in the image or video streaming is wear a mask. [4]

Existing works on face mask detection can be categorized into conventional machine learning (ML) methods, deep learning (DL) based methods, and hybrid methods. The deep learningbased methods were utilized in the majority of studies on face mask detection, while conventional ML-based methods by Nieto-Rodríguez are limited. In NietoRodríguez et al, an automated system was designed that activates an alarm in the operational room when the healthcare workers do not wear the face mask. This system used the Viola and Jones face detector for face detection and Gentle AdaBoost for face mask detection [5]

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. [6]

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

In the object-oriented approach, the focus is on capturing the structure and behaviour of information systems into small modules that combines both data and process. The main aim of Object-Oriented Design (OOD) is to improve the quality and productivity of system analysis and design by making it more usable.

3.1.1 Requirement Analysis

Requirement analysis results in the specification of operational characteristics of software: indicates interface of software with other system elements and establishes constrains the software must meet. The requirement analysis is mainly categorized into two types functional and non-functional:

i. Functional Requirement

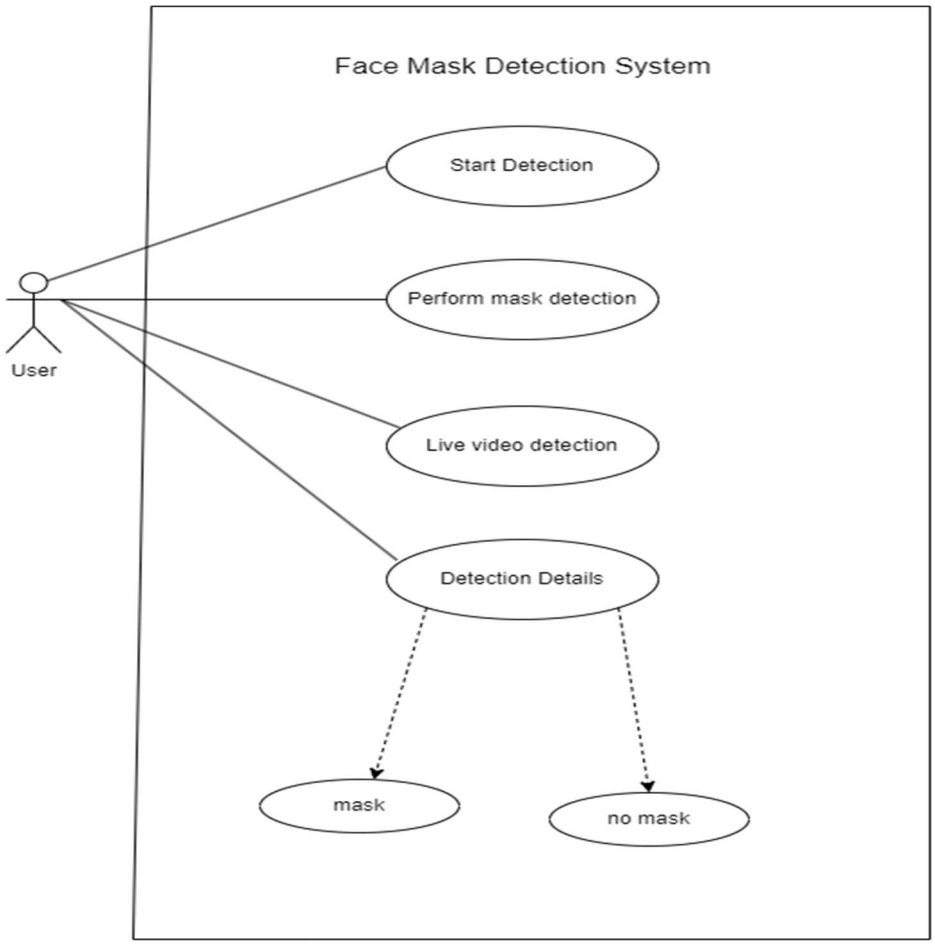

Figure 3.1 Use Case Diagram of Face Mask Detection System

Figure 3.1 shows use case diagram of Face Mask Detection System. The System read the visual input performed by user and process the video to find out either user is wearing mask or not and output the message based on detection.

ii. Non-functional Requirement

Availability: The system is developed as a desktop application.

Security: The system is secure and data isn't shared with third party. Likewise, only authorized entities have access to the data.

Performance: The system is designed for smooth performance with optimization.

Reliability: The system is reliable and gives the same result irrespective of the user if same operations are done.

3.1.2 Feasibility Analysis

- **Technical Feasibility**

  Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements. It requires a computer with python anaconda installed. Today every organization has computer, so it is not an extra cost.

- **Economic Feasibility**

  In this project work, the system developed is a webcam application; which requires all the basic hardware and software support as required by other application. To integrate real surveillance may require software and manpower with developing skills. The proposed model is cost effective.

- **Schedule Feasibility**

  The project to be completed, realistic and achievable under a deadline according to strategy. It is developed within time limit. Hence, it is feasible in respective schedule.

 Operational Feasibility

The proposed system performs effective outcome. It used to dig into the data and quickly conduct experiments to establish baseline performance on a task. The system recognizes a person without wearing a mask which requires no human involvements.

3.1.3 Object Modelling: Object & Class Diagram

Class diagram shows your classes and their relationships. An Object Model Diagram shows the interaction between objects at some point, during run time. A Class Diagram will show what the Objects in your system consist of (members) and what they are capable of doing (methods) mostly static.

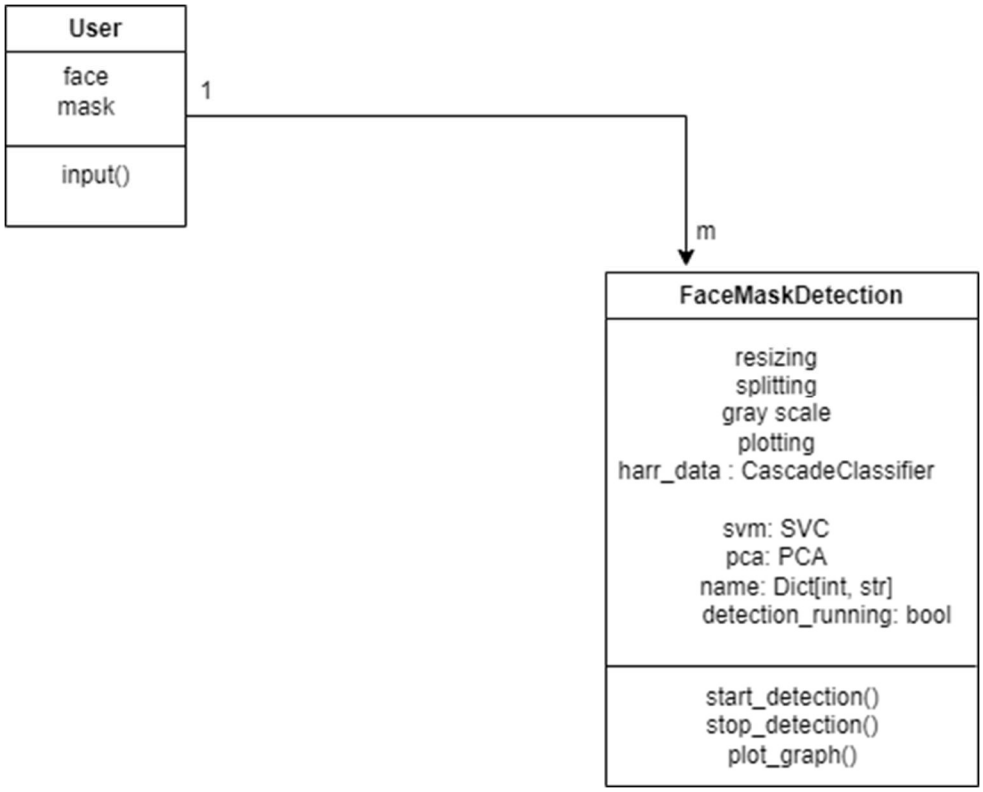

Figure 3.2 Class Diagram of Face Mask Detection System

Figure 3.2 shows the class diagram of the Face Mask Detection System. There are two classes user and FaceMaskDetection in this system. The relationship between these classes is one-tomany which means one user can perform detection multiple times. The class User has a method input() and a face attribute, while the class FaceMaskDetection has three methods: start_detection() for starting the detection system and stop_detection for stopping the detection system and plot_graph() for plotting graph.

3.1.4 Dynamic Modelling: State & Sequence Diagram

A sequence diagram or system sequence diagram (SSD) shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

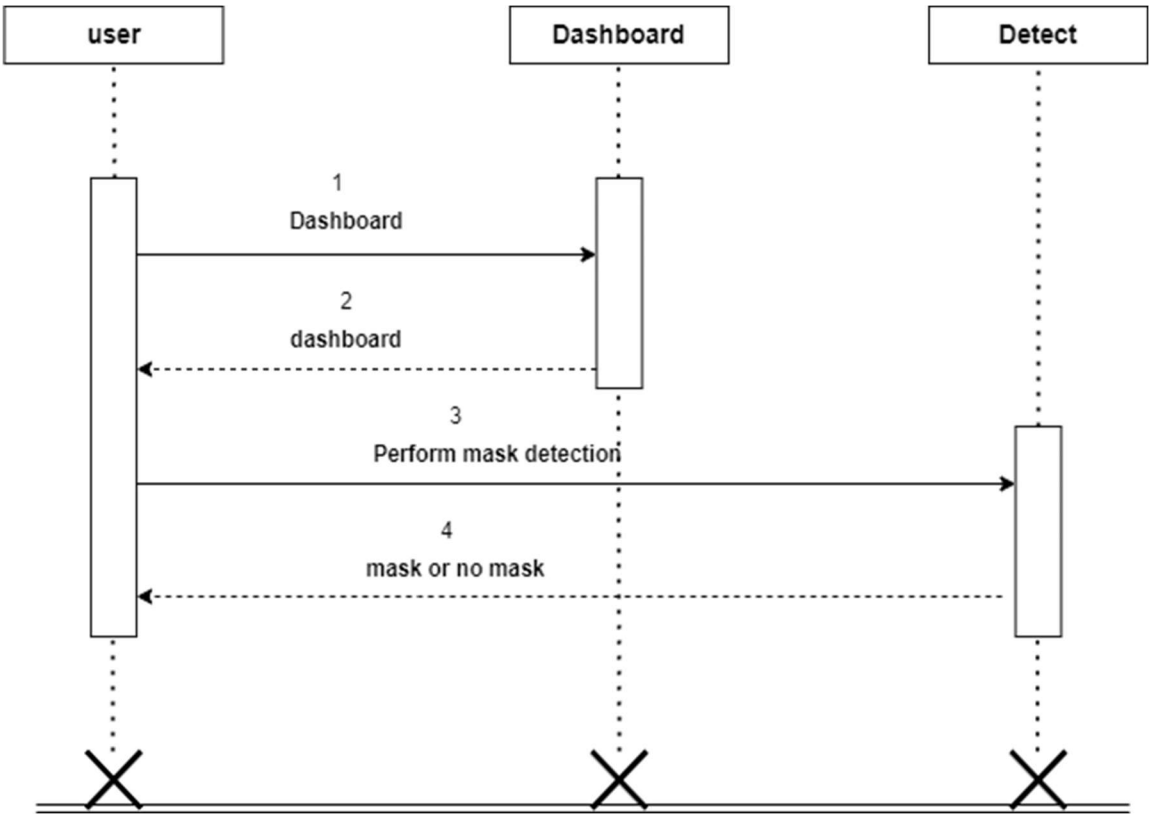

Figure 3.3 State and Sequence Diagram of Face Mask Detection System

Figure 3.3 shows the state and sequence diagram of the Face Mask Detection System. There are three objects: user, dashboard, and detect. The user first gets access to the dashboard, and the dashboard responds to the user. The user then performs mask detection on the system and gets a response.

3.1.5 Process Modelling: Activity Diagram

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. It describes how activities are coordinated to provide a

service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modelling how a collection of use cases coordinates to represent business workflows. Model workflows between/within use cases. Model complex workflows in operations on objects. Model in detail complex activities in high-level activity. Activity diagrams are often used in business process modelling. They can also describe the steps in a use case diagram.

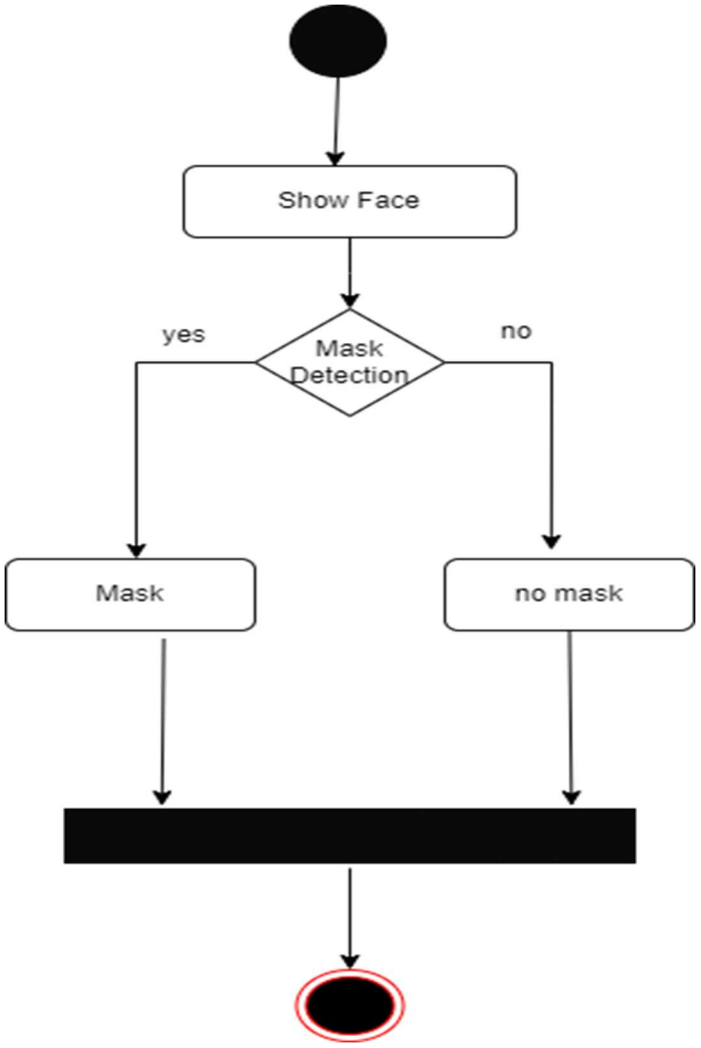

Figure 3.4 Activity Diagram of Face Mask Detection System

Figure 3.4 shows the activity diagram of the Face Mask Detection System. The system first gets started, and then the user shows a face on the system, and the system starts performing

mask detection and provides a response with a message mask or no mask according to the user's appearance.

3.2 System Design

3.2.1 Refinement of Classes and Object

Generalisation is the refinement of a class into more refined classes. Generalisation allows the developer to model objects into hierarchical structures based on their similarities. The class being refined is called super-class and the refined versions of it are called sub-classes. Each sub-class inherits the attributes and operations from their super-class. Methods and attributes can then be refined and the sub-class also adds specific attributes and operations. A discriminator is a variable of enumeration type, which indicate which property of an object is being abstracted. The most important use of inheritance is the conceptual simplification it makes trough the reduction of independent features in the system. By defining a feature with the same name a subclass can override a superclass feature. Overriding is a process of refining and replacing the overridden feature. Generalisation is used for extension and restriction.

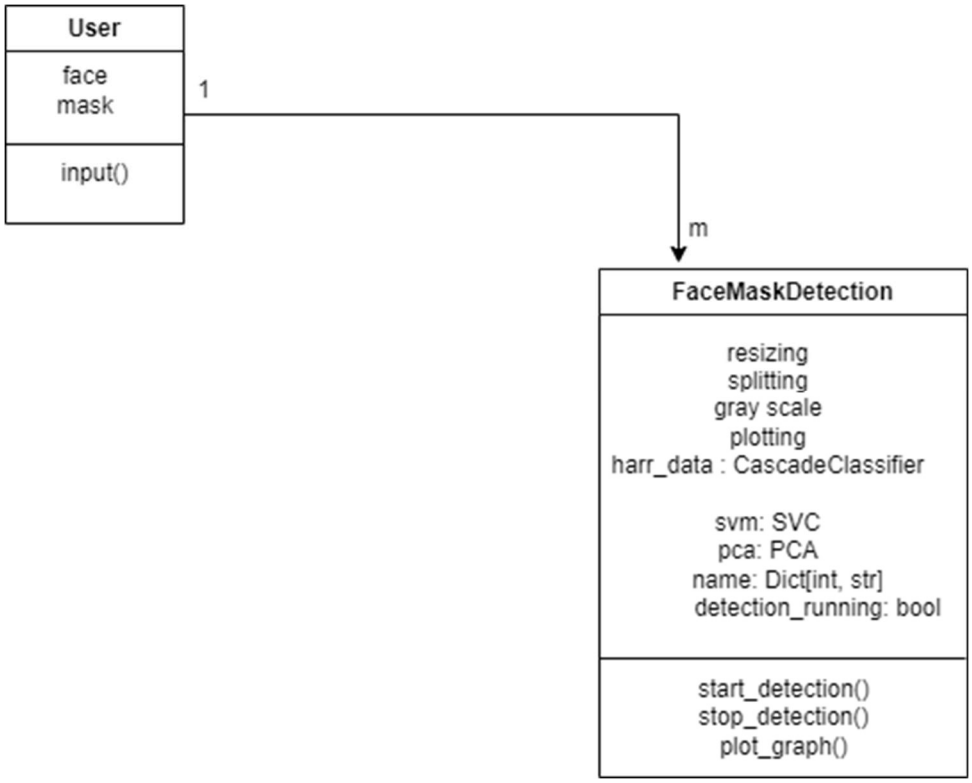

Figure 3.5 Refinement of class diagram

3.2.2 Component Diagram

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development. The purpose of this diagram is different. Component diagrams are used during the implementation phase of an application. However, it is prepared well in advance to visualize the implementation details. Initially, the system is designed using different UML diagrams and then when the artifacts are ready, component diagrams are used to get an idea of the implementation.

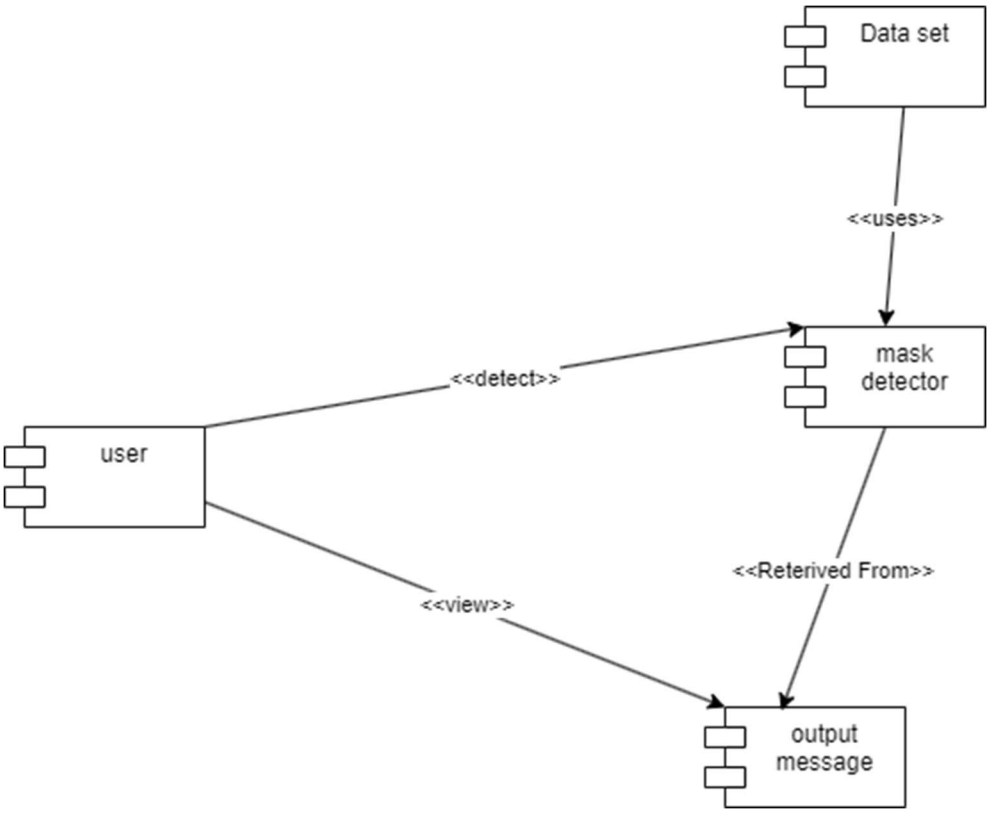

Figure 3.6 Component Diagram of Face Mask Detection System

Figure 3.6 shows the component diagram of the Face Mask Detection System. The diagram contains three components: user, mask detector, and output message. The user component performs mask detection and gets the response output from the output message components.

3.2.3 Deployment Diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

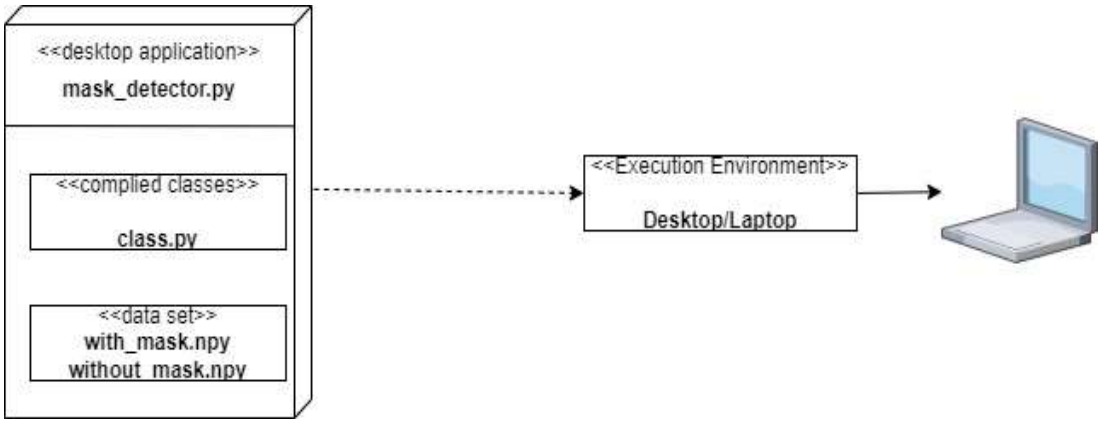

Figure 3.7 Deployment Diagram of Face Mask Detection System

Figure 3.7 shows the deployment diagram of the Face Mask Detection System. The diagram shows that the system for face mask detection is deployed on the laptop or desktop. This means the system Face Mask Detection is a desktop-based application.

3.3 Algorithm Details

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships.

In this study, the SVM method was used as an image classification machine that can classify images of people wearing face masksor not. SVM works by finding the best hyperplane by maximizing the distance between classes. The distance between classes is the maximum distance between data objects in each class. An illustration of determining the hyperplane in SVM is shown in figure below.

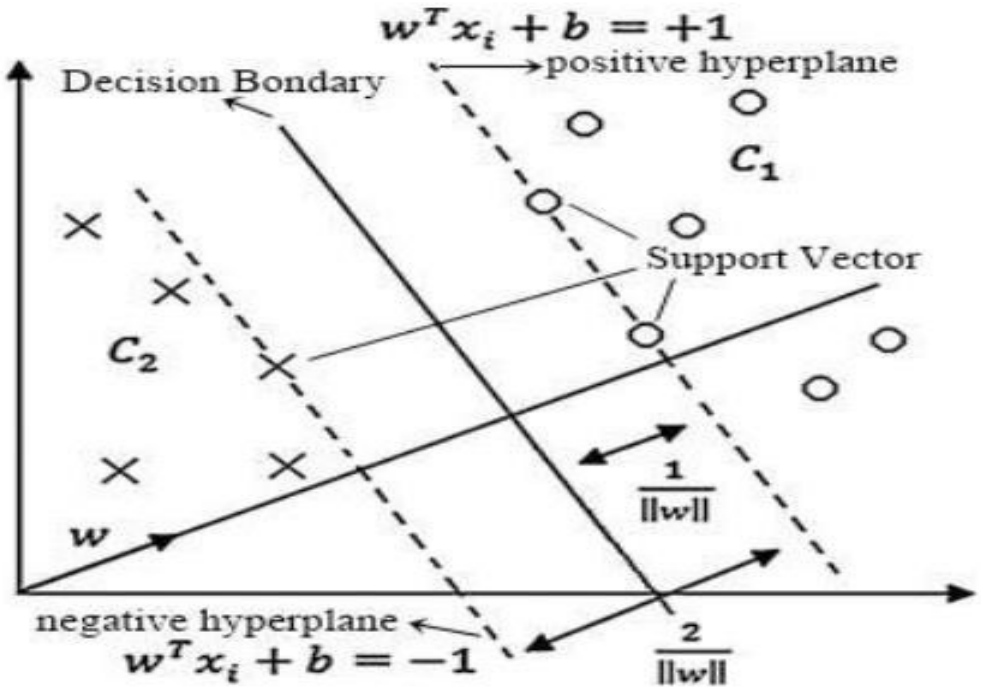

Figure 3.8 SVM hyperplane for classification

In figure 3.3 the patterns X and O are representations of classes 0 and 1. Pattern Xdescribes the distribution of class 0 data and pattern Odescribes class 1. SVM then worksto find the best dividing line (decision boundary) between the two groups of data, this process becomes the learning process in SVM . The result of the process of finding the best dividing line is called a hyperplane.

This research uses SVM with a linear kernel. This setup is chosen because it only aims to detect two classes: images with face mask and without face mask. The SVM learning process in this study can be described. For example, the data is denoted as $x,\in Rn$and the label of each data is notated $yi\in\{1,0\}$for i=1,2,...,$n$where $n$mis the number of data,and $b\in R$. The class assumption used is 1 and 0 which can be completely separated by a hyperplane on dimension $d$defined as in Equation1.

$wTx2+b$=0 ---------->(1)

The pattern of $x$data that belongs to class 1 can be formulated as a pattern that meets Equation2.

$wTx2+b$=+1 ---------->(2)

While the data pattern $x$which belongs to class 0 is formulated by Equation3.

$wTx2+b$=−1 ---------->(3)

The largest margin of the two classes can be found by maximizing the value of the distance between the hyperplane and its closest point through Equation4.

$1\|w\|$ ----------->(4)

For linear classification cases, the SVM conducts optimization on primal space using Equation5.

$min 1/2\|w\|2$ for ----------

---------->(5)

$(wxi+b)\geq 1,=1,...l$

Where xi is the input data, yi is the output of $xi$and w, are the parameters whose values need to be known. The goal to be achieved from the equation is the minimization of the objective function $1/2\|w\|2$or maximize quantity $\|w\|2(wTw)$ with due regard to the limits $yi(xi+b)\geq 1$for output $yi=+1$and $yi(xi+b)\leq -1$for output$yi=-1$. [7]

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Implementation

4.1.1 Tools Used

Python:

Python is a high-level, interpreted programming language known for its simplicity and readability. It offers extensive libraries and frameworks that make it versatile for various applications, including desktop development, data analysis, machine learning, and automation. Python's syntax is clear and concise, making it easy for developers to write and maintain code. Its open-source nature fosters a vibrant community and encourages collaboration, resulting in a wealth of resources and support for Python users worldwide. Due to its user-friendly design and powerful capabilities, Python has become one of the most popular programming languages for both beginners and experienced developers alike.

Draw.io

This is used to generate diagrams for system analysis and design of face mask detection system. Diagrams were created using this tool in order to save time since all components are available with drag and drop function.

4.1.2 Implementation Details of Modules

i. User dashboard module

The user dashboard module of a face mask detection system is a crucial user-facing interface that serves as the primary entry point for the application. This module offers various essential elements to enhance the user experience and facilitate effective face mask detection. This module provide the start and stop button for starting and stopping detection system.

ii. Algorithm module

In a face mask detection system, the Support Vector Machine (SVM) algorithm plays a crucial role in classifying images of faces into two categories: those with masks and those without masks. This module begins by collecting a dataset of labeled images, where each image is preprocessed to extract relevant features. These features are used to train the SVM classifier, which seeks to find the optimal hyperplane that maximizes the separation between masked and unmasked faces in the feature space. Through a process of iterative optimization, SVM learns to make decisions based on this hyperplane, effectively distinguishing between the two classes.

4.2 Testing

4.2.1 Test Cases for Unit Testing

Unit testing is a critical software testing approach that focuses on evaluating individual units of code, such as functions or methods, in isolation. The primary objective of unit testing is to verify that each unit performs as expected, given specific inputs, and produces the correct outputs. These tests are automated, enabling developers to quickly and consistently obtain feedback on their code changes. By catching and resolving bugs at an early stage of development, unit testing enhances the overall reliability and maintainability of the software. Additionally, unit tests act as a safety net during code refactoring, ensuring that existing functionality remains unaffected. As a vital component of a comprehensive testing strategy, unit testing plays a pivotal role in building robust and error-free software systems.

Table 4.1 Test cases for unit testing

SN	Test Case	Input	Expected Output	Actual Output	Result
1	Access home page	-	Access home page	Home page was accessed successfully	PASS
2	Mask detection with mask on face	Face	Display mask as an output message above the face frame.	mask	PASS
3	Mask detection with no mask	Face	Display no mask as a output message above the face frame	no mask	PASS

4	Graph	-	Plot graph according to the accuracy score in real time.	Graph plotted on detection	PASS

4.2.2 Test Cases for System Testing

System testing is a comprehensive software testing phase that assesses the entire integrated system's functionality to ensure it meets the specified requirements and performs as expected in a real-world environment. It involves testing the interactions between different components and subsystems to validate their seamless integration. System tests focus on evaluating the system as a whole rather than individual units, aiming to identify any inconsistencies or issues that may arise during the system's execution. This testing phase is essential for verifying the system's overall compliance with the intended design and user expectations before deployment, providing confidence in the system's reliability and readiness for production use. ***Table 4.2 Test cases for system testing***

SN	Test case	Input	Expected Output	Actual Output	Result
1	Access home page	-	Access home page	Home page was accessed successfully	PASS
2	Perform face mask detection	Face	‘marks’ or ‘no mask’ message	mask	PASS
3	Graph		Plot graph according to the accuracy score in real time.	Graph plotted on detection	PASS

4.3 Result Analysis

The evaluation of outcomes obtained from a face mask detection system relies on a parameter called the accuracy score. This score functions as a quantifiable measure of the certainty or reliability associated with detections made by machine learning models or algorithms. Typically expressed as a probability value ranging from 0 to 1, a high accuracy score close to 1 indicates a strong level of accuracy in the detection that a person is wearing or not wearing a mask, while a low score near 0 suggests uncertainty. In the context of machine learning classification tasks for face mask detection, the accuracy score is provided alongside the detected result, providing users with valuable insights into how much the model trusts its output. This information is crucial for decision-making and risk assessment, allowing users to take appropriate actions based on the model's accuracy level. In practical scenarios, detections with high accuracy scores are more reliable and may not require further validation, whereas detections with low accuracy scores might necessitate additional scrutiny or manual verification. By leveraging accuracy scores, users can make more informed decisions and gain a better understanding of the reliability of detections produced by machine learning models in face mask detection applications.

Table 4.3 Result analysis of Face Mask Detection System

Algorithm	Dataset	Accuracy score
SVM	400	0.9916666666666667
	600	1
	800	0.99375

The table shows the result analysis of face mask detection system. This system uses svm for face mask detection. There are multiple dataset with their respective accuracy score.

CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATIONS

5.1 Conclusion

At the end of this project, the experience of working single has been got. The use and implementation of the desktop-based application has been learnt by this project. This project is developed using python in visual studio code.

With the completion of this project it has been possible to detect face mask. The face mask detection system is an application that utilizes computer vision and machine learning techniques to automatically detect whether individuals in video frames are wearing. The user can also see the graph plotted according to the accuracy score.

5.2 Future Recommendations

The system for face mask detection can be more efficient for the user. The system can be developed as a mobile app. This system can be integrated with access control systems, such as automatic doors or turnstiles, to allow or deny entry based on mask detection results. This system can also integrate with alert sounds based on mask detection results.

References

[1] S. H. S. Rajat Sachdeva, "Face Mask Detection System," https://www.ijser.org/research paper/Face-Mask-Detection-System.pdf.

[2] G. S. Behera, "towardsdatascience," [Online]. Available: https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08#:~:text=So%20what%20is%20Haar%20Cascade,Simple%20Features%E2%80%9D%20published%20in%202001..

[3] P. M. Shah, "Face Detection using support vector machine," https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1322&context=etd_projects.

[4] B. Ko, "A Brief Review of Facial Emotion," Research Gate, January, 2018.

[5] E. M, "A Review of Emotion Recognition," Semantics Scholar, 28 June 2018,.

[6] "Support Vector Machine Algorithm," javatpoint.

[7] M. N. Y. Utomo, "Face Mask-e Wearing Detection Using Soft-Margin Support Vector Machine (SVM)," Dec, 2021.