: N° d'ordre
: N° de série

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

UNIVERSITY OF ECHAHID HAMMA LAKHDAR - EL OUED

FACULTY OF EXACT SCIENCES
Computer Science Department

Thesis
submitted in partial fulfilment of the requirements for the degree of

ACADEMIC MASTER

Field: **Mathematics and Computer Science**
Option: **Computer Science**
Specialty: **Distributed Systems and Artificial Intelligence**

Presented by :

- **Mennana Mouna**
- **Bousbia salah loudjine**

Title

IoT Edge deployment for an automatic recognition of face masks

Defended on June 16 th 2022

Examination Commitee :

M.	……	MCA	Chair
M.	……	MAA	Examinator
M.	Allal Imed	MAA	Supervisor
M	Khelaifa Abdennacer	MAA	Co-Supervisor

Academic year: 2021-2022

Acknowledgements

My thanks go first to Almighty GOD who gave us patience, courage and health to complete this work. Research work can by no means be brought to light without the help of people around us.For this, we would like to thank our supervisors

Mr.Imed Allal and *Mr.Khelaifa Abdennacer*

for their commitment and persistence with us. Similar gratitude addressed to our teachers for their academic generosity throughout five years of devotion. We are also pleased to thank everyone who advised, guided, or contributed with us in preparing this research. Finally, our thanks go to the members of the jury for accepting to discuss and evaluate our work

Dedicate

The first thanks is to Allah Almighty

For their endless love, support and encouragement , A special feeling of gratitude to my loving parents, Samir and Nacira whose words of encouragement and push for tenacity ring in my ears.

My sisters Sounia, Roumaissa, Houda, Djouhina, Belkis and Salima and

my brothers Abdelmadjide and Nizar rayen

always supported and motivated me to go forward .

I also dedicate this work to my many friends and big family, To all our dear nephews.

I give special thanks to my best friend , I will always appreciate all they have done

Mennana mouna Berir mebarkah Chellig marwa

I am also pleased to extend my thanks to everyone who advised me, guided me, directed me, or contributed to the preparation of this research by connecting me to the required references and sources at any stage of it

Thank you verry well

Loudjine bousbia salah

Abstract

Edge Al is a combination of Edge Computing and Artificial Intelligence. Edge computing consists of multiple techniques, wich are: data collection, analysis, and processing to the network's edge. This means that the computing power and data storage are located where the actual data collection happens.Due to, sending all that device-generated data to a centralized data center or to the cloud it can caused issues bandwidth, storage, processing and latency. So,the challenge is how to adapt the existing model to the edge environment? In this study, a system are being created to meet the various needs of industrial applications. The main contribution of this research is the development of a robust real-time face mask detection system using a deep learning SSD MobileNetV2 model in order to, design and implement a Proof Of Concept (PoC) of an IoT Edge deployment to address the use case of automatic recognition of face masks. For the performance evaluation of this deep learning system, we used the device Raspberry Pi to deploy in edge.

Key words : Edge AI,Edge Computing,Artificial Intelligence,face mask, deep learning,SSD MobileNetV2 v2,Raspberry Pi.

Résume

Edge Al est une combinaison d'Edge Computing et d'intelligence artificielle. L'Edge Computing se compose de plusieurs techniques, à savoir: la collecte, l'analyse et le traitement des données jusqu'à la périphérie du réseau. Cela signifie que la puissance de calcul et le stockage des données sont situés là où la collecte de données réelle a lieu. En raison de l'envoi de toutes ces données générées par l'appareil vers un centre de données centralisé ou vers le cloud, cela peut entraîner des problèmes de bande passante, de stockage, de traitement et de latence. Alors, le défi est de savoir comment adapter le modèle existant à l'environnement de pointe ? Dans cette étude, un système est en cours de création pour répondre aux différents besoins des applications industrielles. La principale contribution de cette recherche est le développement d'un système robuste de détection de masque facial en temps réel utilisant un modèle d'apprentissage en profondeur SSD MobileNetV2 afin de concevoir et de mettre en œuvre une preuve de concept (PoC) d'un déploiement IoT Edge pour répondre au cas d'utilisation. de reconnaissance automatique des masques faciaux. Pour l'évaluation des performances de ce système de deep learning, nous avons utilisé l'appareil Raspberry Pi à déployer en périphérie.

Mots clés: Edge AI, Edge Computing, Intelligence artificielle, masque facial, apprentissage en profondeur, ssd mobilenet v2, Raspberry Pi.

الملخص

Edge AI عبارة عن مزيج من الحوسبة المتطورة والذكاء الصناعي. تتكون حوسبة الحافة من تقنيات متعددة، تتمثل في جمع البيانات وتحليلها ومعالجتها في حافة الشبكة. هذا يعني ان قوة الحوسبة وتخزين البيانات يقعان في مكان حدوث جمع البيانات. نظرا لإرسال جميع البيانات التي تم انشاؤها بواسطة الجهاز الى مركز البيانات المركزي، او الى السحابة، فقد يتسبب ذلك في حدوث مشكلات في النطاق الترددي والتخزين والمعالجة ووقت الاستجابة. إذن، التحدي هو كيف يتكيف النموذج الحالي مع بيئة الحافة؟ في هذه الدراسة، يتم إنشاء نظام لتلبية الاحتياجات المختلفة للتطبيقات الصناعية. تتمثل المساهمة الرئيسية لهذا البحث في تطوير نظام قوي للكشف عن قناع الوجه في الوقت الفعلي باستخدام نموذج التعلم العميق SSD Mobilenet v2 من اجل تصميم وتنفيذ مفهوم ال (PoC) لنشر Iot Edge لمعالجة حالة استخدام التعرف التلقائي على أقنعة الوجه. لتقييم أداء نظام التعلم العميق هذا استخدمنا جهاز Rasberry Pi لنشره في الحافة.

الكلمات المفتاحية: Edge AI، الحوسبة الطرفية، الذكاء الصناعي، قناع الوجه، التعلم العميق،Rasberry Pi, SSD Mobilenet v2.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

5G Fifth Generation

AP Access Point

API Application Program Interface : Interface de Programme d'Application

BS Base Station

CCC Central Cloud Computing

CNN Convolutional Neural Network

COM1 COMmunications port 1

CPU Central Processing Unit

D2D device-to-device

DL Deep Learning

DNN Deep Neural Network

Edge-AI Edge Artificial Intelligence

FN False Negatives

FP False Positives

fps Frames per second

GPU Graphics processing unit

HTTP Hyper Text Transfert Protocole

IOT Internet of Things

LCD Liquid-crystal display

MCC Mobile Cloud Computing

MEC Mobile Edge Computing

ML Machine Learning

ROI Region of interest

SDN Software-defined network

SSD Single Shot MultiBox Detector

TN True Negatives

TP True Positives

VM Virtual Machine

VSPE VIRTUAL SERIAL PORTS EMULATOR

GENERAL INTRODUCTION

IoT edge computing eliminates latency difficulties associated with the cloud. Along with lower latency, IoT edge architecture improves safety and provides a more seamless end-user experience. IoT edge can be used to handle enormous amounts of data very instantly on a high-throughput network, such as 5G, producing a more immersive, comprehensive experience for the user. At the same time, IoT edge can make machines and other equipment that impact human safety work faster, keeping operators and others safer, even when relatively tiny amounts of data are communicated [23].

Today, a massive number of smart devices and objects are connected with sensors. However, the Internet is insufficiently scalable and efficient to handle IoT big data. at the same time, transmitting the big amounts of data to Claude is costly, it requiries a large amount of bandwidth, energy, and time. IoT Edge is proposed by placing computer resources closer to IoT devices in order to reduce traffic in the core network and to minimize delaying between computing resources and IoT devices.[55] Mobile edge computing (MEC) is a significant technology in the emerging of fifth generation (5G) network that enables a wide range of IoT applications,which includ facial recognition, face mask recognition, augmented reality, and immersive media.

The target is that to propose, design and implement a Proof Of Concept (PoC) of an IoT Edge deployment and to address the use case of automatic recognition of face masks. The proposed method is an independent face mask detection system, which is ready to be used in edge computing devices like the Raspberry Pi.

We will discuss the structure of this thesis as follows:

Chapter I: IOT And Mobile Edge Concepts. Throughout Chapter I, we will discuss the fundamental concepts of IoT and Edge computing.

Chapter II: State Of The Art For Face Mask Detection. During Chapter II, we introduce the related works and some concepts of artificial intelligence.

Chapter III: Proposed Solution. As for chapter III, we will present our method that used for the detection face mask and optimized model from Deci to apply concept Edge AI.

Chapter IV: Implementation And Results. We will devote Chapter IV to the evaluation of the results that obtained following the implementation of the proposed approach, and we will conclud this thesis with a general conclusion describing the pivots of the contribution, no doubt, followed by the perspectives drawn.

Chapter 1

IOT and mobile edge concepts

Introduction

Internet of Things (IoT) refers to the stringent connectedness between the digital and physical world, Various researchers have described IoT in a multitude of forms. Today, we live in a time of the Internet of Things (IoT), a new paradigm that has accelerated the development of contemporary wireless communications. The main idea underlying this notion is that there are many items (sensors, actuators, mobile phones...) all around us that may communicate and collaborate to achieve a shared purpose through a system of unique addressing. Edge computing is a relatively new paradigm that aims to bring computational power in close proximity to IoT sensors, smartphones, and connected technologies. So, let's explore concepts of edge computing and IoT through some real-world use cases.

1.1 Internet of things

1.1.1 Deinition IoT

The Internet of Things, or IoT, refers to billions of physical objects that are connected to the internet and collect and exchange data all over the world. Because of the introduction of low-cost computer processors and the widespread availability of wireless networks,it is now possible to turn anything.Connecting all of these disparate items and attaching sensors to them provides digital intelligence to machines that would otherwise be stupid,allowing them to communicate real-time

data without requiring a person.The Internet of Things connects the digital and physical worlds,making the world around us smarter and more responsive[5].

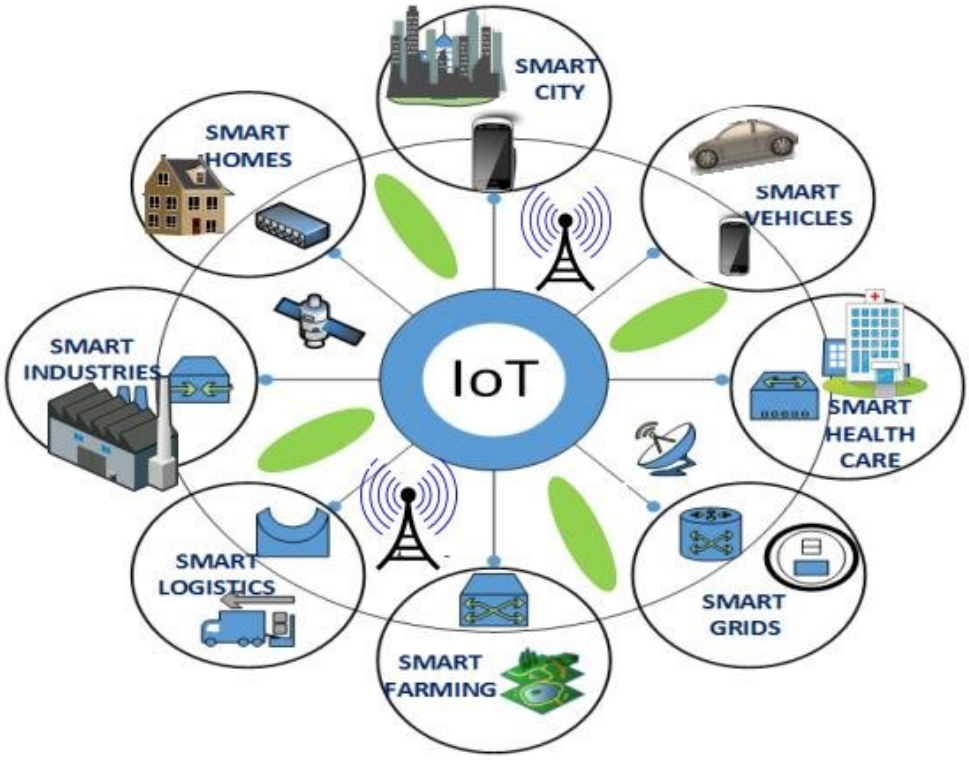


Fig. 1.1 Application domains internet of Things [34]

1.1.2 Internet of things characteristics

some internet of things Characteristics of are:

1. **Connectivity** Connectivity makes possible network accessibility and Compatibility.There needs to be a connection between various levels. Anything can connected with sensors and other electronics and connected hardware and control systems

2. **Things** Anything that can be tagged or is designed to be connected as such.Sensors and home appliances to tagged livestock.

3. **Data** Data is the Internet of Things' glue,the first step toward action and intelligence[32].

4. **Communication** Devices are linked so that they can exchange data, which can then be examined.Communication can take place across short distances or over long to extremely long distances.Wi-Fi,LPWA network technologies such as LoRa or NB-IoT are examples.

5. **Intelligence** IoT features inbuilt artificial intelligence that improves all aspects, including smart and dynamic, such as sensing capabilities in IoT devices and intelligence derived from big data analytics.

6. **Distributed control** Because of the large number of nodes, control in the internet of things is dispersed and exhibits emergent features and behaviors that necessitate proper distributed control [58].

1.1.3 IoT components

Here are four basic components of an IoT system that explain how it works:1.2

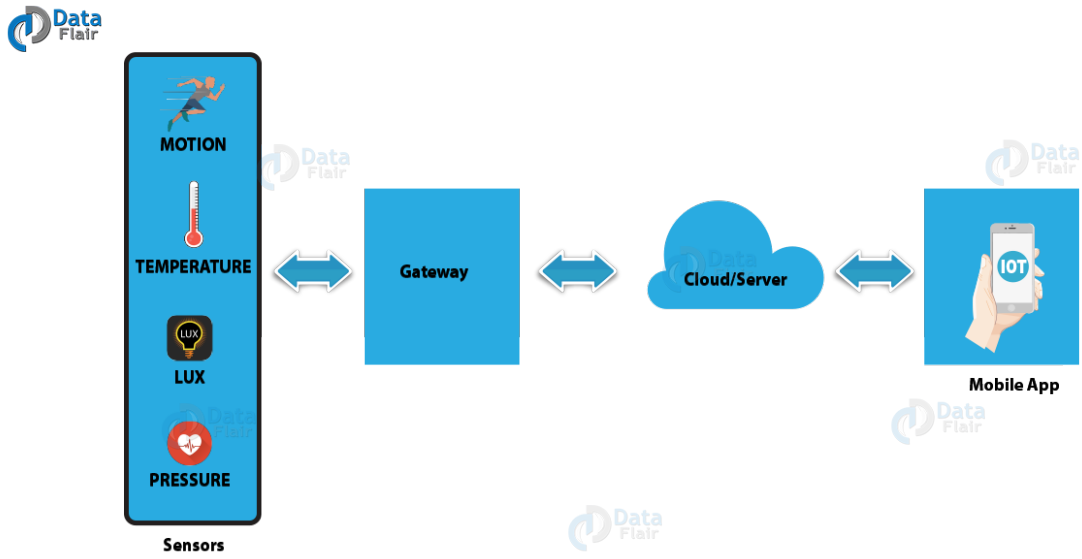


Fig. 1.2 IoT Components [4]

1. **Sensors/Devices** Sensors or gadgets assist in the collection of extremely very minute data from the environment. From a basic temperature monitoring sensor to a complicated full video stream, all of this acquired data might have varying degrees of complications. Multiple sensors on a gadget can be bundled together to perform more than merely feel things1.3 [4].

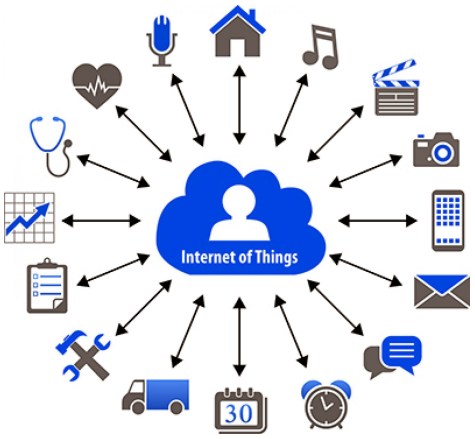


Fig. 1.3 Devices [4]

2. **Connectivity** Data is captured and transmitted to a cloud infrastructure,but it must be transported.Sensors can communicate with the cloud via a variety of communication and transport methods, including cellular networks,satellite networks, wide-area networks (WAN), low-power wide-area networks,Wi-Fi, Bluetooth and others fig1.4[4]

Fig. 1.4 connectivity [4]

3. **Data processing**The software processes the acquired data once it has been collected and transferred to the cloud.This might be as basic as ensuring that

the temperature reading on devices like air conditioners or heaters is within an acceptable range[4].

4. **User interface** These IoT Elements establish the fundamentals of nearly every IoT system on the planet.They are still segregated into several architecture layers to further enhance the overall IoT network [4].

1.1.4 IoT layers

These Elements of IoT define the fundamentals of almost every IoT system on the globe.Still,they are divided into multiple architecture layers to further refine the overall IoT network[39].

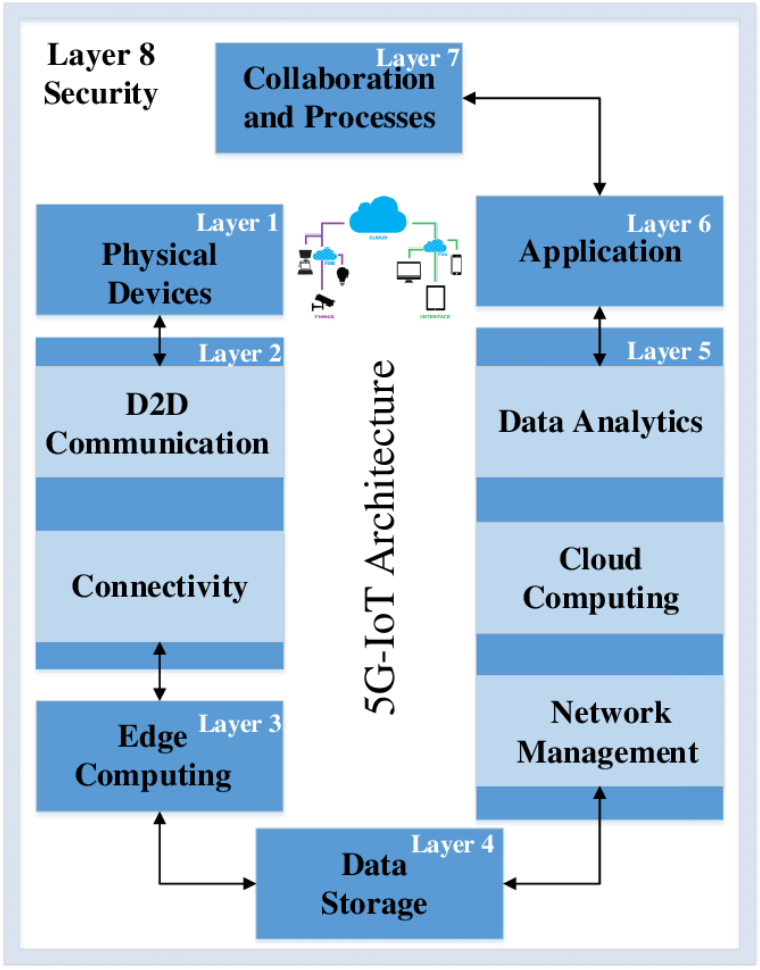


Fig. 1.5 The proposed 5G-IoT architecture [39]

1. **Physical device layer**

This layer is made up of wireless sensors, actuators,and controllers, which are the "things" of the Internet of Things.Physical devices are a layer that is shared by all designs.Small size devices, such as Nano-Chips,will be used in this layer to boost computational processing power while reducing power consumption.Nano-Chips can generate a large volume of first processed data that is suited for Big Data at the data analytic layer (layer 5).

2. **Communication layer**

   There are two sub-layers in this layer.Layers of D2D communication and connectivity.

   - **Device-to-device communication (D2D)**

     Direct device-to-device communication sublayer (D2D). Because of increases in processing power and intelligence, physical devices (nodes) now have their own identity and personality,as well as the ability to generate their own data. In order to improve the efficiency and capabilities of IoT systems,these devices need form a HetNet to connect with one another.This sub-layer employs the most recent wireless sensor network (WSN) communication standards. Nodes can cluster, and they can even elect a leader (cluster head) to ensure proper networking.One of the most important technologies for upgrading this sub-layer is mmWave. Furthermore, 5G is an optional technology at this sub-layer that can boost D2D communication. 5G has been regarded as a serious candidate for MTC device connectivity.5G-Plus-HetNet has been identified as a potential technology solution in the anticipated 5G-IoT architecture,which provides low latency to IoT, due to MTC's high data rate support and other distinctive qualities.

   - **Connectivity sub-layer**

     Devices in this sublayer are linked to communication hubs such as BSs.They also use the Intranet connection to the storage unit to send and analyze data through the centers.This IoT sub-layer currently has several limitations:it can only handle a limited number of device connections; data exchange for a variety of data types is inapplicable in applications such as autonomous vehicles; and high volume data cannot be processed in real-time due to high communication latency. In the near future,5G deployment will see major advancements in terms of reliability, performance, and agility at this sub-layer.As previously noted, another

technology of this sub-layer is Advanced SSIM, which allows IoT devices to select appropriate spectrum (frequency bands) with less interference. In fact, SSIM approaches can be used to capitalize on the potential of cognitive radio-based spectrum sharing. This design's sublayer provides low latency, connection durability, and compatibility for a wide range of data formats.

3. **Edge (Fog) computing layer** This tier's nodes or leaders edge process data in order to make decisions at the edge.With the advent of 5G technology and the proliferation of mobile devices (such as smartphones),MEC technology will be more powerful in resolving challenges, and will contribute considerably at this layer.

4. **Data storage layer** This layer contains data storage units that hold both raw data and information obtained from physical device edge processing.This layer requires additional security protection and should be able to handle the large amounts of data and traffic generated by future apps.

5. **Management service layer** This layer consist of three sub-layers as follows.

   - **Sub-Layer for network management**

     Network administration includes changing the type of communication between devices and data centers.The most important technology in this sublayer is WNFV. To improve the quality of the IoT structure, the WNFV can simultaneously update the network topology and communication protocols,such as 5G-IoT or ZigBee. Another important technology at this layer is WSDN. Instead of traditional network monitoring for performance enhancement,the WSDN keeps the IoT network running and allows for network reconfiguration.

   - **Sub-Layer for cloud computing**

     In this sub-layer, data and information from edge computing are (re)processed in the cloud,allowing the final processed information to be obtained.Because of the adoption of 5G technology,mobile devices can now perform this type of computing,known as MCC, in realtime.As a result,processing tasks will be distributed in parallel across mobile devices, increasing the efficiency,scalability,and speed of the IoT system.

   - **Sub-Layer for data analytics**

In this sublayer,new data analytics methodologies are employed to extract value (manipulable information) from raw data.Any advances in Big Data approaches will assist this sub-layer in processing data more efficiently. In actuality, as the volume of data collected grows as a result of the convergence of 5G and IoT,the role of this sublayer will become increasingly crucial in the near future.

6. **Application layer**

   Because software interacts with preceding levels and data that is at rest at this tier,network speed is not required.The Applications layer,in fact, enables business people to do the right thing with the right data at the right time.

7. **Collaboration and processes layer**

   Unless and until the IoT system generates an action, the data it gets from the preceding levels is meaningless.People are empowered by applications that perform business logic.People use applications and data to satisfy their individual needs. At the same time,multiple persons may use the same software for various objectives.Individuals must be able to engage and communicate for the Internet of Things to be useful.

8. **Security layer**

   This layer is addressed independently.In reality, this layer shields and covers all previous levels,although each segment (where this layer meets with another layer) serves a distinct purpose.The security layer of the proposed architecture includes data encryption, user authentication,network access control,and cloud security.Furthermore,the security layer detects and anticipates threats and cyberattacks [51].

1.1.5 Internet of things applications

IoT application domains are composed of a variety of services with varying traffic characteristics.As a result, there is a trade-off between generality and specificity.The characterization of IoT traffic is compounded further by the fact that there are very few publicly available datasets for academics to use and build upon.They divide IoT use cases into four primary application domains 1.6[41].

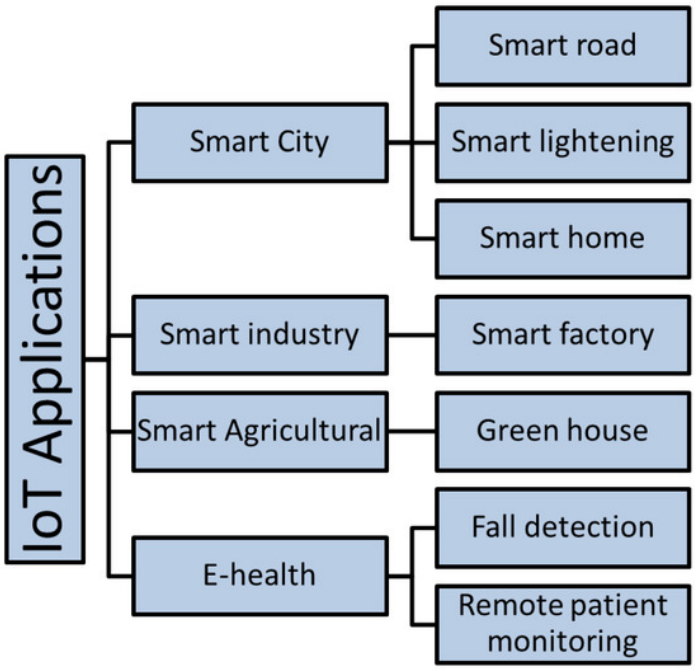


Fig. 1.6 Internet of things applications [41]

1.2 Cloud computing

1.2.1 What's cloud computing?

Simply described,cloud computing is the transmission of computing services over the Internet ("the cloud") to enable faster innovation, more flexible resources,and scalability.You normally only pay for the cloud services you use,which allows you to save money, run your infrastructure more efficiently,and scale as your company grows [30].

1.2.2 Types of cloud computing

Organizations of all sizes have come to rely on the cloud to protect and cost-effectively manage their data infrastructure.They use cloud services to manage, store, and deliver data. Choosing which service to use requires a sufficient strategy and planning for choosing between public,private,and hybrid cloud installations.

1.2.2.1 Public cloud

The public cloud is the most typical approach to install cloud services.A third-party service provider owns and maintains a public cloud service and is in charge of

maintaining the cloud services and infrastructure.Public cloud services are available over the internet and are appropriate for small to medium-sized organizations. Microsoft Azure and Amazon EC2 are two of the most well-known public cloud services.Public clouds are employed when data compliance and control are not a major issue for the consumer.

1.2.2.2 Private clouds

Private clouds are owned and operated by a single company or entity. In a private cloud environment, the hardware, software, and any supporting infrastructure are either located in the organization's data center or in a service provider's regulated environment. In terms of data flexibility and control, private clouds differ from public clouds. Private clouds, by definition, cannot be provided as a service. Microsoft and HP Data Centers are two well-known private cloud instances. Private clouds are preferred by government agencies, financial institutions such as banks, mid- to large-sized corporations, and any other body dealing with sensitive information.

1.2.2.3 The hybrid cloud

The hybrid cloud is a cloud computing architecture that synchronizes private and public cloud services. This cloud allows customers to move data and programs between public and private clouds. Most firms favor hybrid cloud architecture since it provides numerous business benefits such as meeting regulatory and data requirements, addressing low latency challenges, and so on [15].

1.2.3 Main cloud service models

IaaS, PaaS, and SaaS are the three main cloud computing service models. Each cloud service model meets unique user and business requirements while providing variable levels of control, security, and scalability.

1.2.3.1 SaaS (Software as a Service)

In this cloud service model, software is hosted online and made available to customers via subscription or purchase. SaaS cloud providers host apps in their network, which users can access from a range of devices via a browser or app. Software as a Service is another name for cloud application services.

1.2.3.2 PaaS (Platform as a Service)

PaaS is a cloud service paradigm that provides a ready-to-use development environment that allows developers to focus on developing and executing high-quality code to construct bespoke apps.

1.2.3.3 IaaS (Infrastructure as a Service)

IaaS is a cloud computing system that provides and manages computing resources such as servers, storage, networking, and virtualization via the Internet. [12].

1.2.4 Main cloud computing platforms

1.2.4.1 Amazon Web Services

Amazon Web Services (AWS) is an amazon subsidiary (a leading Commerce corporation). Amazon offers on-demand cloud computing platforms such as storage and data analysis. Amazon provides services to individuals, corporations, and governments. Members of Amazon Web Services can obtain a full-fledged virtual cluster of computers at any time, depending on their needs. The entire service is available via the internet [8].

1.2.4.2 Google cloud platform

Google Cloud Platform (GCP) refers to Google's public cloud computing products. It offers services in every major domain, including computation, networking, storage, machine learning (ML), and the internet of things (IoT). It also offers cloud management, security, and development tools. Google Cloud Storage is a highly dynamic storage solution capable of storing SQL (Cloud SQL) and NoSQL (Cloud Data store) databases.

1.2.4.3 Microsoft Azure

The company's cloud computing solution is Microsoft Azure (formerly Windows Azure). This service, which is only offered through Microsoft-managed data centers, has proven to be a reliable choice, especially for Microsoft evangelists. It assists in the development, testing, deployment, and management of applications and services, as do the preceding solutions. For web development, it supports PHP, ASP.net, and Node.js[8].

1.2.4.4 IBM Bluemix

IBM Bluemix is an IBM cloud computing system that offers platform as a service (PaaS) as well as infrastructure as a service (IaaS). Bluemix IaaS users can deploy and utilize virtualized computational power, storage, and networking via the internet. Depending on the needs of the company, IBM service offerings can be used in a public, private, or hybrid style [8].

1.2.4.5 Alibaba

Alibaba Cloud is the cloud initiative of the Chinese eCommerce juggernaut Alibaba Group. With headquarters in Hangzhou, China, Alibaba's services dominate the Chinese market and have global origins. It was established in 2009, just three years after Amazon Web Services[8].

1.2.5 What is mobile cloud computing?

Mobile Cloud Computing (MCC) makes substantial computational resources available to mobile users over wireless networks. MCC's goal is to enable the execution of sophisticated mobile apps on a wide range of mobile devices [42].

1.3 Edge computing

1.3.1 Definition edge computing

Edge computing is an unique distributed computing architecture that moves data storage, services, and computing applications from centralized nodes to the network's edge. IoT devices generate enormous amounts of data. As a result, issues with bandwidth, storage, and processing have arisen. In real-time applications, edge computing technology minimizes latency and reaction time. The network is managed by the cloud, while edge computing maintains service continuity. The edge device delivers storage, control, and communication in close proximity to the end user [41].

As a result, deploying edge computing in a network improves the network in a variety of ways.

- **Efficiency** By dispersing storage, compute, and control functions to resources situated anywhere between the end-user and the cloud, an edge device makes

the best use of available resources. This allows IoT devices to take advantage of pooled edge-computing resources

- **Cognition** An edge device is mindful of its consumers' requirements. IoT devices in an e-health system, for example, monitor patients' physical health, especially in emergency patient scenarios, and computation-resource allocation is adjusted based on users' health-risk grade.
- **Agility** Because data processing and storage are done close to the end user, experimenting with edge devices and clients is faster and less expensive.
- **Latency** edge computing helps time-critical applications by allowing data analysis and processing close to the end-user, allowing IoT apps to make better and faster decisions [41].

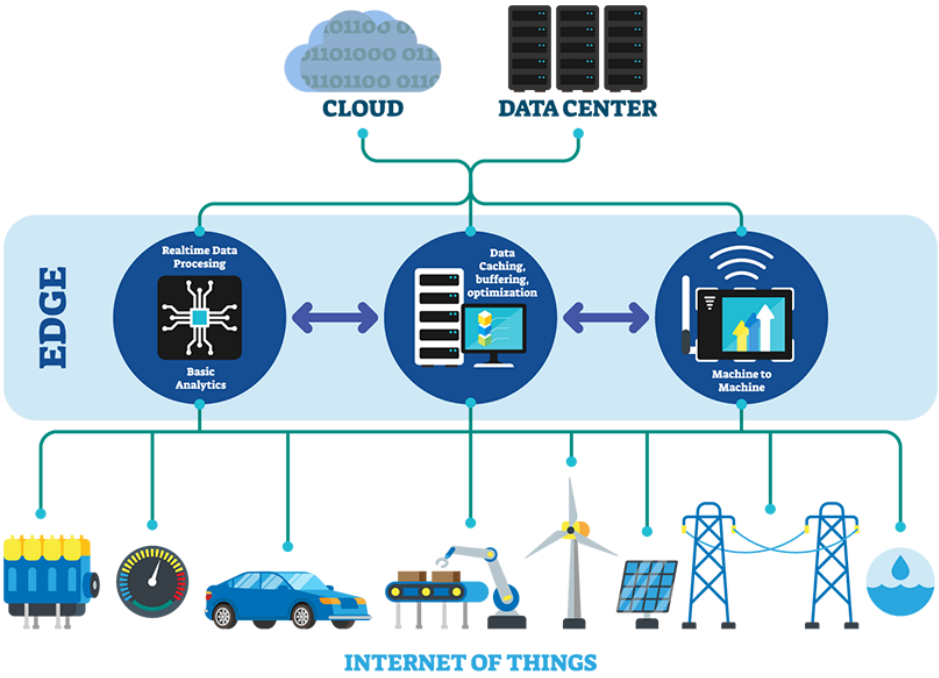


Fig. 1.7 Edge computing [27]

1.3.2 Edge-Computing implementation

1.3.2.1 Cloudlet

Cloudlet nodes are a collection of computers that function as a miniature data center dedicated to providing services to IoT devices in the same geographic area. This

resolves the previously unresolved issue of end-to-end responsiveness between mobile devices and the cloud[41].

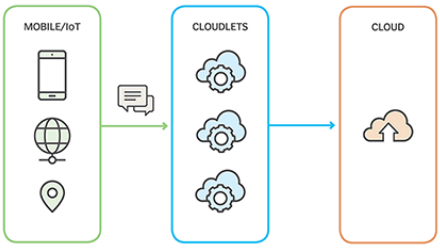


Fig. 1.8 Cloudlet for IoT [1]

1.3.2.2 Fog computing

Fog computing is a decentralized architecture of computing nodes that provide end-user services between themselves and the cloud. Because fog-computing nodes are heterogeneous, they can include a wide range of devices such as switches, industrial controls, and access points. Because fog computing nodes can be deployed anywhere between end-users and the cloud, it is versatile[41].

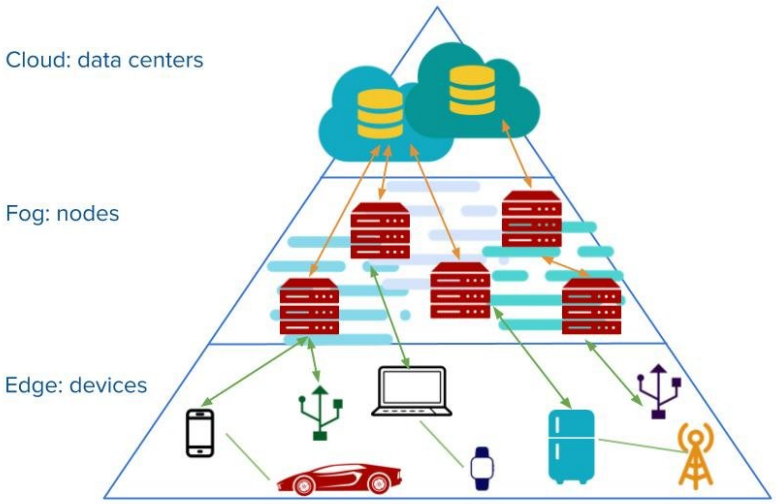


Fig. 1.9 Fog computing [2]

1.3.2.3 Mobile edge computing (MEC)

Mobile edge computing (MEC), also known as multi-access edge computing, is analogous to the outermost edge in that it consists of our ubiquitous mobile devices and applications, as well as the huge amounts of data they consume and generate. This figure will rise with the launch of 5G[18].

1.3.2.4 Key drivers for MEC

Edge computing is being driven by the Internet of Things (IoT), today's 4G networks, and next-generation 5G networks. Due to the exponential increase in traffic, particularly video, and the proliferation of connected devices, network infrastructures will need to expand successfully in order to provide increasing volumes of data. To meet these requirements, MEC brings the flexibility and agility of the cloud closer to the client.

1.3.2.5 Characteristics of MEC

- **Close proximity** Edge captures crucial information for analytics and processing since it is close to the source of data, eliminating the requirement to backhaul data to central locations[19].
- **Real-time**Applications that require near-real-time or real-time decision processing and outcomes benefit from MEC[19].
- **Low latency** Typically has a latency of less than 20 milliseconds. This results in a faster response time and a better user experience[19].
- **Continuous operations** Edge applications are localized, which means they may operate independently of the rest of the network – even if they are cut off from the core[19].
- **Interoperability** MEC eliminates the need for application adoption or migration to the new environment, making development and deployment more efficient[19].
- **High Bandwidth** MEC offers high bandwidth because data and tasks are transferred to a nearby MEC server at the edge network rather than the central server via the core network, avoiding network congestion and allowing data

transmission between the MEC server and user terminals to use the entire bandwidth of access networks at high speeds[47].

- **Location Awareness** By evaluating the information received from user devices within the local access network, MEC can determine where users are [47].
- **Mobility Support** Mobility is an evident aspect of user devices, which is supported by MEC because work can be offloaded to different MEC servers that communicate with each other when users travel[47].
- **Security and Privacy Protection** Data can be delivered to MEC servers rather than central servers over the core network, ensuring security and privacy at the edge and reducing the risk of data leakage during transmission [47].

1.3.2.6 MEC Architecture

As shown in Fig 1.10 MEC Architecture is partition into three layers.

1. Cloud layer.
2. MEC Layer.
3. Device Layer.

Millions of gadgets, including mobile phones, IoT devices, and sensors, are linked to various base stations located near their position on the device layer. MEC servers with cloud and IT services capability are installed at MEC layer base stations. It receives and processes requests from numerous devices, as well as offering compute capabilities or demanding resources at the base station. Decentralization is accomplished in this manner because devices are linked and transmit requests to the nearest MEC server, as opposed to the cloud, where all needs are centralized. The core network connects all MEC servers to the cloud[60].

1.3.3 Benefits of edge computing

Moving some data services such as storage, processing, and analysis away from the cloud and closer to where data is generated might provide several major advantages:

- **Increased speed and lower latency:** Reduced latency and increased speed. Moving data processing and analysis to the edge improves system response time, allowing for faster transactions and better experiences, which are critical in near-real-time applications such as autonomous car operation.

Fig. 1.10 MEC architecture [19]

- **Improved network traffic management:** The bandwidth and expenses of sending and storing huge amounts of data can be reduced by reducing the amount of data delivered over the network to the cloud.
- **Greater reliability:** The amount of data that may be transmitted by a network at one time is restricted. The ability to store and process data at the edge enhances reliability when the cloud connection is broken for locations with poor internet connections.
- **Enhanced security:** An edge computing system, when properly implemented, can improve data security by limiting data transfer over the internet[10].

1.3.4 Disadvantages of edge computing

1. Security In a distributed edge context, ensuring proper security might be difficult. Because data processing takes place at the network's perimeter, there is a high danger of identity theft and cyber security breaches.

2. Incomplete Data Small amounts of data can only be processed and interpreted by edge computing. The rest of the data is just ignored. As a result, businesses may lose a significant amount of essential information.

3. Investment Cost Edge computing on IoT devices requires the utilization of more local hardware to function. Overall, this has the potential to boost efficiency, but it will necessitate a significant investment.

4. Maintenance Edge computing, as opposed to centralized cloud architecture, is a distributed system. In comparison to centralized infrastructure, this requires more upkeep.

1.3.5 Mobile edge computing & 5G

In the future 5G application scenario, 5G networks will be required to respond swiftly and efficiently to complex and changing services by utilizing cloud-based resource pools and fully achieving their performance criteria. In this regard, flexibility is a crucial quality for 5G networks since it allows them to expand network resources and improve service provisioning. Using existing mobile network architecture to provide the flexibility required by the 5G scenario, on the other hand, is a difficult task. MEC has been praised as a promising solution for 5G networks, with the ability to change how resources are managed, distributed, and maximized. MEC has been praised as a promising solution for 5G networks, with the ability to change how resources are managed, distributed, and maximized. MEC can be used by mobile operators to vary their operation modes, cut operational expenses (OPEX), and improve revenue capability. In contrast to the usual centralized data center, MEC is located at the periphery of 5G networks. The resources could be dynamically updated and optimized based on the services and applications coupled with MEC. Furthermore, mobile device computational resources may be considered in the MEC architecture to perform the essential operations, resulting in a significant boost in system performance. The confluence of 5G networks and MEC creates a new commercial environment for network operators, service providers, and end users[43].

1.3.5.1 What is 5G?

5 G is the most recent version of cellular technology, designed to greatly increase the speed and responsiveness of wireless networks. Data transmitted through wireless broadband connections can currently travel at multigigabit rates, with some estimates claiming peak speeds of up to 20 gigabits per second (Gbps). These speeds are quicker than traditional landline network rates and have a latency of 1 millisecond (ms) or less, making them ideal for real-time applications. 5G will enable a huge increase in the volume of data carried over wireless systems due to increased available bandwidth and improved antenna technology[28].

1.3.5.2 MEC for 5G and IOT : enabling technologies

Cloud computing, software-defined networking (SDN), network function virtualization (NFV), virtual machine (VM) and container technology, smart devices, network slicing, and compute offloading are just a few of the important technologies that allow MEC to be combined with 5G and IoT in this field [35].

1.4 Conclusion

In this chapter we had defined the basic elements and concepts involved in this memoire .Trying to clarify and offer a background, by discussing the different technologies involved. Next chapter we will provide State of the art for face mask detection of into existing systems with view previous works .

Chapter 2

State of the art for face mask detection

2.1 Introduction

Wearing masks has been shown to be an effective method of preventing COVID-19 transmission and is now required in most public areas, increasing demand for automatic real-time mask detection devices to eliminate person reminders. However, there have been few investigations into face mask detection. Deep learning has gained popularity in object detection and has been applied to human identification, such as the development of a face mask detection tool that can determine whether or not a person is wearing a mask. This may be accomplished by examining real-time camera streams to evaluate the categorization findings. We will demonstrate earlier work for face mask detecting embedded devices in this area.

2.2 Machine learning (ML)

Machine Learning (ML) is the study of computer algorithms that learn on their own over time. It is thought to be a subset of artificial intelligence.Machine learning algorithms construct a mathematical model using sample data, referred to as "training data," in order to create predictions or judgements without being expressly programmed to do so. When constructing standard algorithms to do the essential tasks that would be difficult or impossible, machine learning techniques are used. Machine learning is closely related to computational statistics, which focuses on using computers to anticipate outcomes. Mathematical optimization research offers machine learning with tools, theory, and application fields. Data mining is a similar field of study that focuses on exploratory data analysis using unsupervised

learning. When applied to commercial problems, machine learning is frequently referred to as predictive analytic[24].

Machine learning can be split in three main approaches:

2.2.1 Supervised learning

To train the machine learning model in supervised learning, we require labeled data. We feed the input into the model, and its output is compared to the right label using user-defined criteria known as the loss function. Supervised learning is demonstrated by object detection and tracking. Because all Deep Learning DL object identification models are taught using this method, most cutting-edge tracking approaches rely on supervised learning[48].

2.2.2 Unsupervised learning

Unsupervised learning is the polar opposite of supervised learning. The goal here is to uncover structure within the given input that corresponds to the problem to be anticipated, but starting from unlabeled data, the absence of annotated data, or the absence of a learning base, which indicates unsupervised learning [48].

2.2.3 Reinforcement learning

Reinforcement learning enables learning based on input from interactions with the outside environment. It is widely used in robotics, gaming, and navigation. For example, an agent interacts with its environment and learns its behavior depending on the input it gets. Reward feedback, also known as a reinforcement signal, is essential for the agent to learn its behavior. The agent must learn to operate in a way that maximizes the overall expected reward[48].

2.2.4 Deep learning

Deep learning is an artificial intelligence function that simulates the processing of data by the human brain to identify objects, recognize voice, translate languages, and make decisions.

AI can learn without human involvement utilizing both ordered and unlabeled data with in-depth schooling. Deep learning approaches attempt to learn feature hierarchies made of lower-level features and higher-level features. Instead of

depending completely on human-crafted features, a system may learn complicated functions mapping input to output directly from data by automatically learning features at various levels of abstraction.
Deep learning algorithms seek to identify acceptable representations by leveraging the unknown structure in the input distribution, usually at many levels, with higher-level learnt features defined in terms of lower-level features.
In this, Face mask identification is achieved by the use of Convolution Neural Networks, a Deep Learning approach (CNN)[13].

2.2.5 Convolution neural network (CNN)

A Convolution Neural Network (CNN) is a form of artificial neural network that is made up of neurons that have learnable weights and biases. Each neuron takes a large number of inputs, computes a weighted sum, runs it through an activation function, and then creates an output. The network as a whole has a loss function, and all of the principles we learned for neural networks still apply to CNNs. One of the most visible uses of architecture is image classification. CNNs are used for a variety of tasks, including image and video recognition, recommender systems, and natural language processing[13].

2.3 Related works

2.3.1 Proposed R.Mohandas and all

R. Mohandas and others recommended using automated face mask detecting devices at the office building and lab entry points. This subsystem provides real-time face mask identification with over 89 percent accuracy and greater than 90 percent confidence at detection speeds of less than 3ms.

The proposed solution is a self-contained face mask detection system that may be run on edge computing devices like the Raspberry Pi. This is a single stage detector that operates in real-time with the Raspberry Pi camera module's video stream. real-time with the video feed from the Raspberry Pi camera module.

The technology has been shown to function consistently with minimum human input and may be easily installed in labs or other employee facilities where face masks are necessary for employee safety. The proposed method consists of a single

stage object detector that uses ROI to specifically target a specific region in an image/video input, as well as an adaptive training dataset.

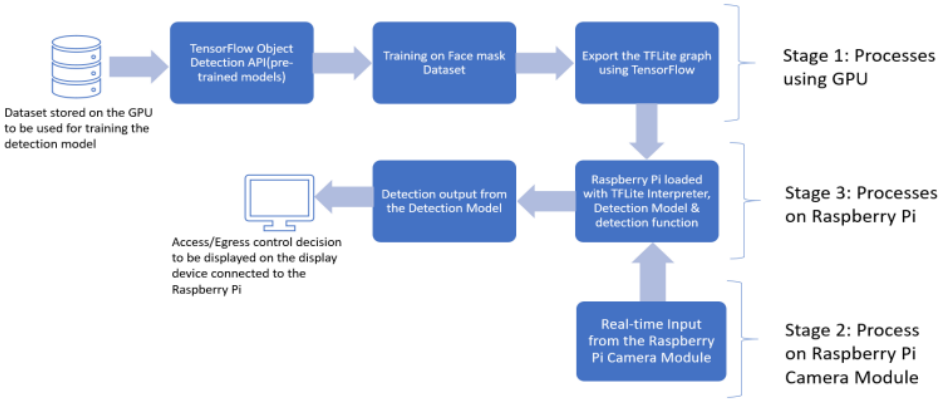


Fig. 2.1 The stages of the face mask detection and access/egress control process[49]

The Deep Learning Detection Model is trained during the Model Stage 1 Detection Stage. In this study, the SSD Inception V2 Deep Learning model was used, which was originally trained on a GPU.

In Stage 2 of the procedure, the input video stream is received. This is gathered by the Raspberry Pi Camera Module v2 and sent to the Raspberry Pi over the serial port.

Stage 3 includes the Raspberry Pi processes. At this stage, the Raspberry Pi serves as a resource-constrained Edge Computing device connected to a display that displays the output of the Raspberry Pi Camera Module as well as the final access/egress decision based on the face mask detection algorithm[49].

2.3.2 Proposed B.Varshini and all

B Varshini and colleagues created an IoT-enabled smart door that uses a machine learning model to monitor body temperature and detect face masks in this study. To recognize face masks, monitor temperature, and count the number of people present at any one time, the model leverages a real-time deep learning system based on a Raspberry Pi. The device performs brilliantly in terms of temperature measurement and mask detection; the trained model received a score of 97 percent. The test findings reveal a high degree of accuracy in recognizing persons wearing and not wearing facemasks, as well as monitored and recorded notifications. The

proposed method may be used in any commercial mall, hotel, or residential doorway. As a consequence, a cost-effective and trustworthy method of creating a healthy environment employing AI and sensors was devised. The suggested system is evaluated using the Face Mask Detection approach, which is implemented in the TensorFlow software library. Deep learning is used to conduct face mask recognition

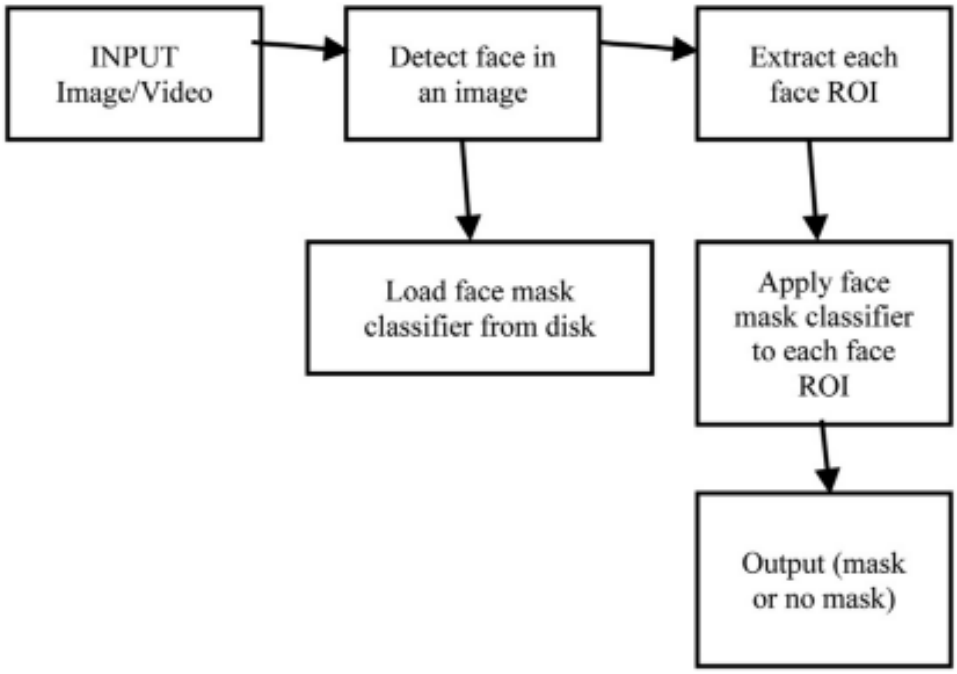


Fig. 2.2 Face mask detection [59]

and Convolution Neural Networks (CNN) classification. The TensorFlow software library and the face mask detection technique are used to assess the proposed framework. The Mask detection model is trained using Keras and TensorFlow [59].

2.3.3 Proposed R.Tanuja and all

Tanuja R and all ,they proposed Real World Face Mask Detection . The MultiTask Cascaded Convolutional Neural Network (MTCNN) is used in this work to detect and identify faces. For mask detection, MobileNetV2 is employed as an object detector.

For this project, 3833 photos from various data sources were picked. This is later replicated using a Raspberry Pi and a pi cam; this system feeds live video data from a remote location, allowing the mask to be predicted to be worn.

At the 20th epoch, the amount of information lost in the process is steadily reduced to 0.0199. The accuracy with which mask/no mask detection is improved

The obtained training model has a 99% accuracy and a 99 percent recall. The accuracy for mask detection in the cascade frame is 86.6% and 87.8%. When comparing the cascade framework for mask detection to the suggested framework, the proposed framework has higher accuracy.

The proposed methodology attempts to detect whether or not people are wearing masks in video footage of a public location. To do so, we first recognize the person's face before evaluating whether or not a facial mask is evident.

The proposed framework aims to identify the aforementioned behaviour from a public location. To do so, we first identify the person's face before deciding whether or not a mask is present. For this assignment, the Multitask Cascaded Convolutional Neural Network (MTCNN) was employed as the approach. The input image is first scaled in different dimensions to form an image stream, which is then fed into the network in three phases.

In the first stage, a Fully Convolutional Network (FCN) is utilized to identify possible facial areas based on the input. In the second stage, a CNN dubbed Refine Network (R-Net) is utilized to filter these candidate windows. R-Net eliminates a large number of false positives and calibrates the candidates

discovered using bounding box regression. In the third step, an O-Net CNN is used to produce facial landmark locations such as the eyes, nose, and mouth. Face landmark detection is a regression problem, comparable to bounding box regression. The Raspberry Pi and pi cam were the best hardware choices for implementing face mask classification in the real world[56].

2.3.4 Proposed W. Boulila, and all

The first stage was to develop a deep learning (DL) model capable of detecting and locating facemasks in real-time and determining if they are suitably worn. They recommended using MobileNetV2 to identify and assure facemask in real-time, as well as light-weight DL suitable for edge devices.

This technique has the benefit of being rapid and suitable to edge devices; the dataset is acquired using public datasets as well as their own; the datasets utilized in this study are as follows: Face Mask Lite Dataset, Real-World-Masked-Face-Dataset, MaskedFace-Net, and face mask detection also describe and analyze experimental results using numerous cutting-edge models such as ResNet50, DenseNet, and

VGG16. Several experiments are carried out, and the proposed technique performs admirably (99 percent for training and testing accuracy)[37].

DL model	Training time (Hours: minutes)
DenseNet	4:24
ResNet	5:58
VGG	18:51
MobileNetV2	**1:26**

Fig. 2.3 Comparison of the training time of Models

2.3.5 Proposed R. K. Shinde , and all

Intelligent Internet of Things (IoT) Device for Detecting COVID-19 Infections Using Sensor Fusion (SF) Algorithm and Real-Time Mask Detection This work uses a novel SF technique to detect a COVID-19 suspect and the upgraded MobileNetV2 model is employed for face mask identification on an Internet-of-Things (IoT) platform.

The SF algorithm avoids inaccurate questionable predictions. The ThingSpeak cloud server continually monitors and records health data. Three separate cloud servers are used in this strategy to implement the various functions. The cloud gives a write API key for saving data and a read API key for retrieving data. On the Raspberry Pi, we installed the basic mail transmission protocol (SMTP).

The SMTP server sends an alert email to a healthcare practitioner including critical health data and the GPS location of a suspect. The edge device has SF and notification servers. Real-time face identification (using a spy camera) predicts an output using the learned deep learning model.

To determine optimum mask placement, a lightweight and quick deep learning model are utilized; this limits virus transmission. Before pre-processing, the real-world masked face dataset (RMFD) (which comprises 2165 photos with asks and 1930 without masks) was separated into 80 percent for training and 20 percent for testing subsets. The improved MobileNetV2 neural network is ideal for the Raspberry Pi. Our IoT device and deep learning model are 99.26 % and 98.50 % accurate, respectively, and the time required for face mask evaluation is 31.1 milliseconds.

The suggested gadget can be used to remotely monitor covid patients. As a result, the approach will be used in the identification of COVID-19-positive patients. The gadget can also be worn[54].

2.3.6 Proposed P. Nagrath , and all

Many face detection models have been constructed using various methods and methodologies, including a real-time DNN-based face mask detection system employing a single shot multibox detector and MobileNetV2 (SSDMNV2). To detect face masks, the suggested method in this research employs deep learning, Tensorflow, Keras, and OpenCV. It includes a 'Single Shot Multibox Detector ' (SSD), which enables real-time detection with little resource use. This method aids in the detection of faces in real-time, even on embedded devices such as the Raspberry Pi. The following classifier employs a pre-trained model MobileNetV2, which is extremely lightweight and may even be used in embedded devices to identify masks in real-time. The approach used in this study results in an accuracy score of 0.9264 % and an F1 score of 0.93% [50].

2.4 Conclusion

CNN algorithms have a great success in image classification problem. after we talked about machine learning and Related works in this section ,In the next chapter, we will present our system. The method and the approach we used.

Chapter 3

Proposed solution

3.1 Introudection

Face mask detection is the detection of whether or not a person is wearing a mask. It is an important area in computer vision and pattern recognition. High complexity in feature design and low detection accuracy are among the issues faced with this approach. Face mask detection approaches based on deep convolutional neural networks (CNN) have been widely developed in recent years to increase detection performance [53]. In thesis study, face mask detection is performed utilizing a Deep Learning system and the photo classification approach. The SSDMobilnetV2 architecture model was used.

3.2 Model used

3.2.1 Why to use SSDMobileNetv2

- By training the SSDMNV2 model on the same dataset as other pre-existing models such as LeNet-5, AlexNet, VGG-16, and ResNet-50, the suggested model outperforms the other models in terms of accuracy, F1 score, and FPS parameter. As a result, the SSDMNV2 model is simple to deploy on embedded devices, whereas heavier models require significant computational power to do real-time detection [50].

- Mobile Net requires relatively few computational resources. This results in a perfect match with less detailed results for mobile devices, embedded systems, and computers lacking a GPU or with limited computing capacity [61].

- The architecture MobileNetV2 shows an acceptable performance with low computational power, this makes this model suitable for embedded [36].
- The validation accuracy and loss results of MobileNetV2, ResNet50, DenseNet, and VGG16 are compared. The findings reveal that the MobileNetV2 model performs best in terms of training time, performance, validation accuracy and loss, which validates its adoption in our application [38].
- This table 3.1 displays the resource values for the various neural network models where:
    - When compared to other models, MobileNetv2 consumes fewer resources because its weight does not exceed 14MB.
    - It has the smallest parameters of 3.5M, making it lighter and more adaptable to embedded systems with limited computational power.

Table 3.1 The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset

Model	**Size (MB)**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**	**Time (ms) per inference step (CPU)**	**Time (ms) per inference step (GPU)**
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
Inception-ResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7

3.2.2 Single Shot MultiBox Detector(SSD)

SSD is a deep neural network-based approach for recognizing objects in pictures. It is easy to train and is frequently employed in systems that require a detecting component. The SSD, like Faster-RCNN, predicts which box contains an object by using boxes with varied aspect ratios and then makes modifications to better match the geometry of the object. The bounding box output space will be discretized into a number of default boxes, each with a particular aspect ratio and scale[40].

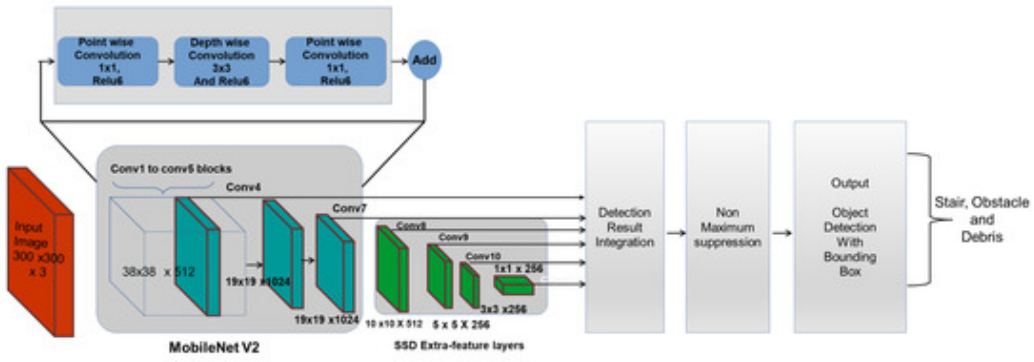


Fig. 3.1 SSD MobileNet V2 Object Detection Framework [52]

3.2.3 MobileNetV2

Mobilenet v2 is a light weight feature extractor that can be used in real-time object detection applications as well as low-power embedded device applications.They may be used for detection, classification, embeddings, and segmentation in the same way that other popular large scale models can. MobileNetv2 is a Significant Improvement over MobileNetV1. The MobileNetV1 and V2 structures are shown below in Fig 3.2.

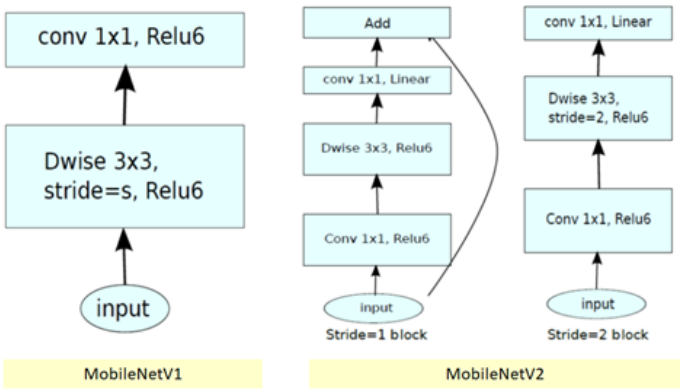


Fig. 3.2 MobileNet [3]

Its architecture is made up of a number of residual bottleneck layers. These layers employ 3 x 3 depthwise convolution layers, which are more efficient than conventional convolution layers. Instead, they use 1 x 1 convolutions. ReLU6 layers, which are ReLU layers with a maximum depth of 6, are also used in the architecture. The upper limit prevents the outputs from scaling to very large values, lowering the calculation cost as well. Furthermore, the model delivers better accuracy for the

design than non-residual structures due to the residual link [52]. The structure of MobileNetV2 has been shown in 3.3

- Convolutional layer This layer serves as the foundation for the Convolutional Neural Network. Convolution is the mathematical process of merging two functions to form a third function. It employs a sliding window technique to assist in the extraction of characteristics from images. This contributes to the generation of feature maps. The convolution of two functional matrices, one of which is the input image matrix A and the other is the convolutional kernel B, results in the output C shown below.

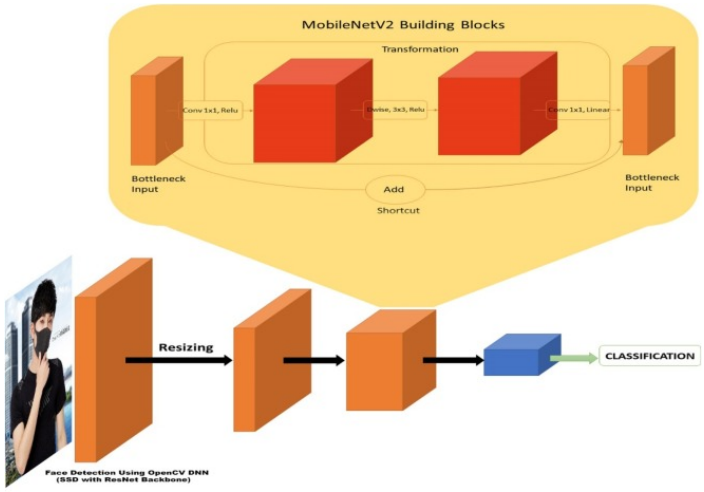


Fig. 3.3 Architecture of MobileNetV2 [50]

- Pooling layer

  Pooling operations can speed up calculations by allowing the size of the input matrix to be reduced without sacrificing many features. pooling processes of several types can be used, some of which are described below:

  1. Max pooling: The maximum value in the designated region where the kernel is currently placed is used as the value for the matrix value output for that cell.
  2. Average pooling: It takes the average of all the values in the region where the kernel is currently placed and uses that value as the matrix value for that cell's output. The operation of average-pooling is depicted.

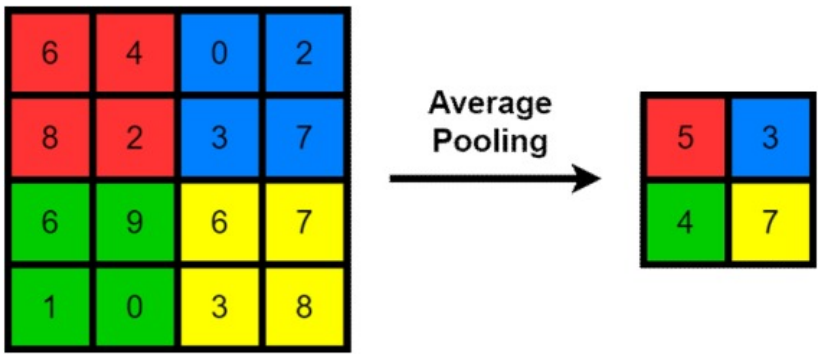


Fig. 3.4 Convolutional operation [50]

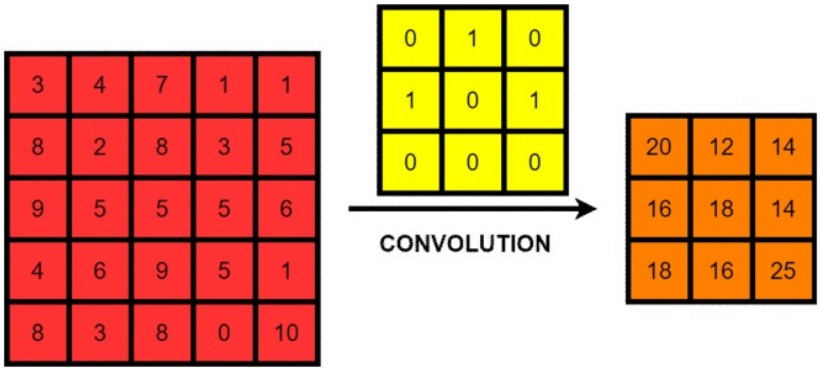


Fig. 3.5 Average-pooling operation [50]

- Dropout layer By removing random biased neurons from the model, this lowers overfitting that may occur during training. These neurons can be found at both the hidden and visible layers of the brain. To change the likelihood of a neuron being dropped, the dropout ratio can be changed.

- Non-Linear layer

  These layers are usually inserted after the convolutional layers. The most commonly used non-linear functions are various versions of Rectified Linear Units (ReLU). Functions include the sigmoid and tanh functions. Non-linear functions of many forms and their equations are given below.

- Fully-Connected layer

  These layers are added to the model and have complete links to the activation layers. These layers aid in the classification of pictures in multi-class or binary

$$Sigmoid : \sigma(x) = \frac{1}{1+e^{-x}} \tag{2}$$

$$LeakyRelu : f(x) = \max(0.1x, x) \tag{3}$$

$$Tanh : f(x) = \tanh(x) \tag{4}$$

$$Maxout : f(x) = \max\left(w_1^T x + b_1, w_2^T x + b_2\right) \tag{5}$$

$$Relu : f(x) = \max(0, x) \tag{6}$$

$$ELU : f(x) = \begin{cases} xx \geq 0 \\ \alpha(e^x - 1)x < 0 \end{cases} \tag{7}$$

Fig. 3.6 Non-Linear functions

classification. SoftMax is an example of an activation function utilized in these layers, and it offers the likelihood of anticipated output classes.

- Linear Bottlenecks

  Because many matrix multiplications cannot be reduced to a single numerical operation, non-linear activation functions such as ReLU6 are used in neural networks to easily remove several disparities. A multilayer neural network can be constructed using this method. Because the ReLU activation function discards values smaller than zero. The network's dimensions grow by increasing the number of channels to combat information loss.

Layers of the blocks are compressed for a reversed residual block, and the opposite method is performed as described above. This is done at the point where the skip connections are joined; this may impede network execution. To address this, the linear bottleneck concept was created, in which the last convolution of the left-over block is given a linear output before adding the block to initial activation[50].

3.3 Our methode

An automated system replaces the human inspection of a person's face mask in the proposed system. The proposed method is ready for implementation on edge computing devices (Raspberry Pi). Figure 3.7 depicts the system's block diagram. This system is installed alongside a video-capture Pi camera. Initially, it detects the face in the captured video. This study employs a Caffe model, which is a deep neural network face detector in OpenCV, to detect faces. Only when the face is detected does the mask detection occur.

In this work, the MobileNetv2 architecture was used to develop the model that assesses whether or not a person is wearing a face mask. The created model takes into account the video stream frame and splits the output into two classes: with and without a mask. The classification result is presented on the Raspberry Pi's LCD screen.

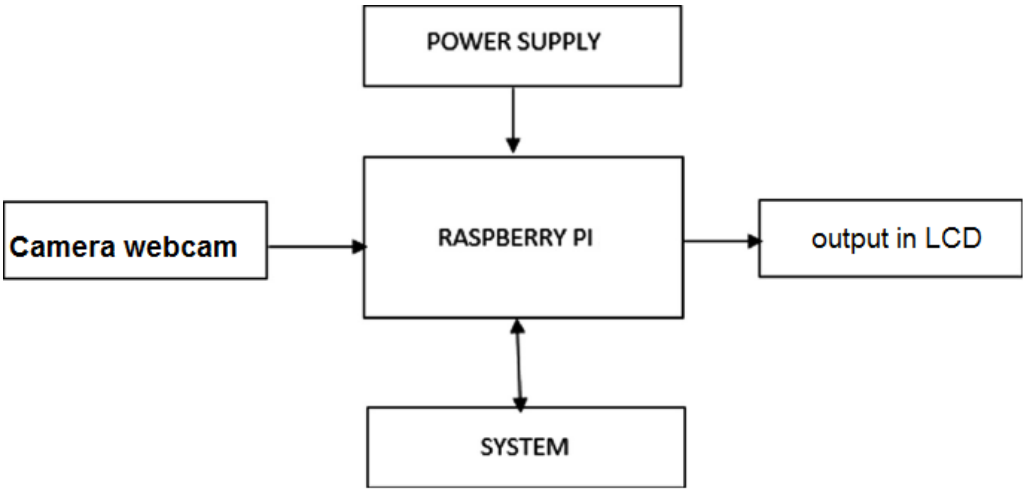


Fig. 3.7 Block diagram of the system

3.3.1 System requirements

The proposed system can be created by following the steps outlined below:

3.3.1.1 Creating or Storing a Face detection model

A model for detecting a face in an image must be established before going on to face mask detection. We utilize a DNN face detector in OpenCV since it can detect a face in a single pass. It is a Caffemodel that uses the ResNet architecture and a single shot detector framework. This module consists of two files. The prototxt file outlines the model's architecture, while the caffemodel file contains face detection layers. A set of bounding boxes centered on each recognized face is returned as output.

3.3.2 Developing, training, and testing a mask detection model

The model SSD MobileNet V2 can be created by following the procedures outlined below steps in algoritm 1.

1. data collecting.
2. pre-processing.
3. split the data.
4. building the model.
5. testing the model.

The complete steps as shown in 3.8.

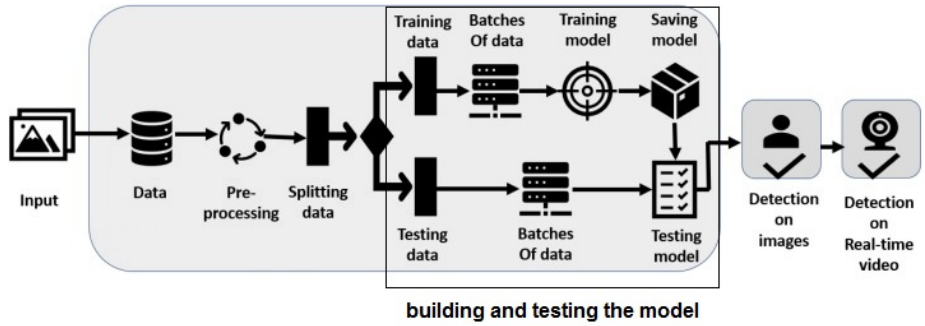


Fig. 3.8 The Proposed Architecture

Algorithm 1 Model Train

INPUT: Dataset.
OUTPUT: Trained model.
1: Load Images and their pixel values.
2: Process the images, resizing, normalization, conversion to a 1D array.
3: Load the filenames and their respective labels.
4: Split data into training and testing batches.
5: Augmentation training data.
6: Load MobileNetV2 network.
7: Compile Model using Adam optimizer.
8: Train MobileNetV2 on training data.
9: Make predictions for testing.
10:Save the model's face mask detection.

3.3.2.1 Dataset collection

The initial stage in building the Face Mask Recognition model is data collecting. The dataset is used to train data on persons who and do not wear masks. The model

can tell the difference between persons who are wearing masks and those who are not. Our dataset consists of 3529 Images *Imgs* from Kaggle, divided into two classes:

- with mask: 1721 images.
- without mask: 1808 images.

At this point, the image is cropped so that the only visible thing is the item's face. The following step is to label the data. After the data has been tagged, it is divided into two categories. The figure shows an example of a face wearing and not wearing a mask.

Fig. 3.9 Set of images

3.3.2.2 Pre-processing:augmentation

The pre-processing phase comes before the data training and testing.Pre-processing consists of four steps: shrinking image size, turning the image to an array, pre-processing input using MobileNetV2, and performing hot encoding on labels.

Because of the effectiveness of training models, image scaling is an important pre-processing step in computer vision. The model will perform better if the image is lower in size. In this step, scaling an image results in an image of 224 by 224 pixels.

The next step is to create an array from all of the photographs in the dataset. The picture is converted into an array so that it may be called by the loop function. Using MobileNetV2, the picture will then be utilized to pre-process input.
Because many machine learning algorithms cannot act instantly on data labeling, the final step in this phase is to do hot encoding on labels. All input and output

variables, including this process, must be numeric. The tagged data will be turned into a numerical label, which the program will understand and interpret.

3.3.2.3 Split the data

Following the pre-processing phase, the data is divided into two batches: training data (80%), and testing data (the remaining 20%). Each batch includes both with-mask and without-mask photos. This testing data is required to assess system performance against new data that was not included in the training data [57].

The following parameters were used in this study:

- The augmentation used is a rotation distance of 20
- zooming area in the range of 0.15
- side and top shift of 0.2
- and image shift of 0.15
- The batch size used is 32

3.3.2.4 Building the model

The model will be built in the next step. The model is built in six steps: generating the training picture generator for augmentation, the basic model with MobileNetV2, adding model parameters, compiling the model, training the model, and finally saving the model for future prediction.

- The following layers were used by the top model (MobileNetV2): Average-Pooling2D with a pool-size of (7.7), then Flatten, then Dense layer with a size of 128 with the Rely activation function, then the arrays are lowered by 0.5 and continued to the Dense layer of 2 with the Soft Max activation function. The Dense value of 2 is changed to provide as many as two outputs, namely the detection of whether or not masks are used.
- The batch size utilized in the fitting procedure is 32, and the steps per epoch are derived by dividing the amount of training data by the batch size.
- Then, in this fitting procedure, we define validation data with a number of validation steps equal to the number of validation data divided by the number of batch sizes defined.

- When the training model is finished, it is saved in a file so that it may be simply used to classify photos from cameras, photographs, and video images.

3.3.2.5 Testing the model

The model is tested in stages. Making predictions on the testing set is the initial stage. The outcome of 20 iterations of checking the model's loss and accuracy when training it. Following the training, validation, and testing phases, the model can deliver a high-accuracy percentage of people wearing a face mask. It gives us accuracy, testing F1-score, recall, and precision.

3.3.3 Proof of concept(Testing with dynamic input)

We begin by loading the face detection model, which is comprised of two files, Prototxt and Caffemodel. The learned facemask detection model is loaded. When you start streaming video, this Pi camera starts receiving it and translating it into frames, or pictures.
Using a face detection model, detect the face in each image. When a face is found, we enlarge the image to 224 * 224 dimensions and preprocess it as in Step 2.
With the face and mask detection models loaded, forecast whether the image is with or without a mask. If the image's predicted class is without a mask, the message displayed on the LCD is "not detect mask". Alternatively, if the image's predicted class is with a mask, then Displayed on the LCD is "detect mask" ,and steps in algorithm .

Algorithm 2 Deployment face mask detector

```
INPUT:deployment option.
OUTPUT: Classification or Show normal feed in Real-time.
Load Model Face Detection
Load Model Mask Face Detection
Open the video stream
if Face > 0 then
    if Mask > 0 then
        Labeling Mask
    else if then
        Labeling NoMask
    end if
end if
```

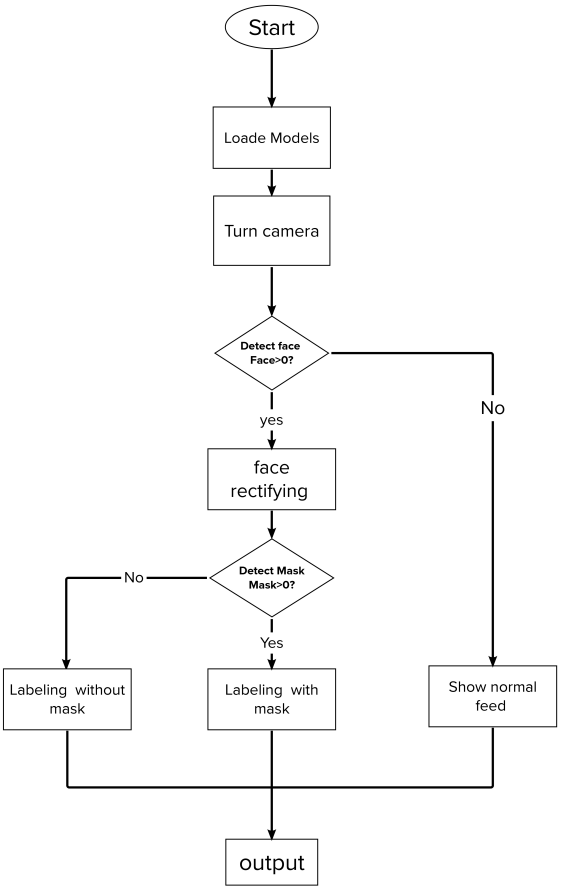


Fig. 3.10 Flowchart of face mask detection

3.4 Optimize the model

"Edge AI has recently emerged as a promising alternative to traditional cloud-assisted inference " [11] .According to the concept of edge AI, inference happens where data is gathered or at the nearest local location, such as on a local server. Implementing the original, uncompressed Model ML in edge devices with limited power, processing capabilities, and accessible memory or resources is a recipe for disaster. You'll wind up with a model that doesn't work correctly. We regularly require real-time applications, such as object identification software, to be installed on our edge, mobile devices [45].

This is why we must compress our initial deep neural network models in order for them to fit and operate with edge devices. This is a hotly debated topic right now. The field's pioneers have found numerous methods for compressing deep neural

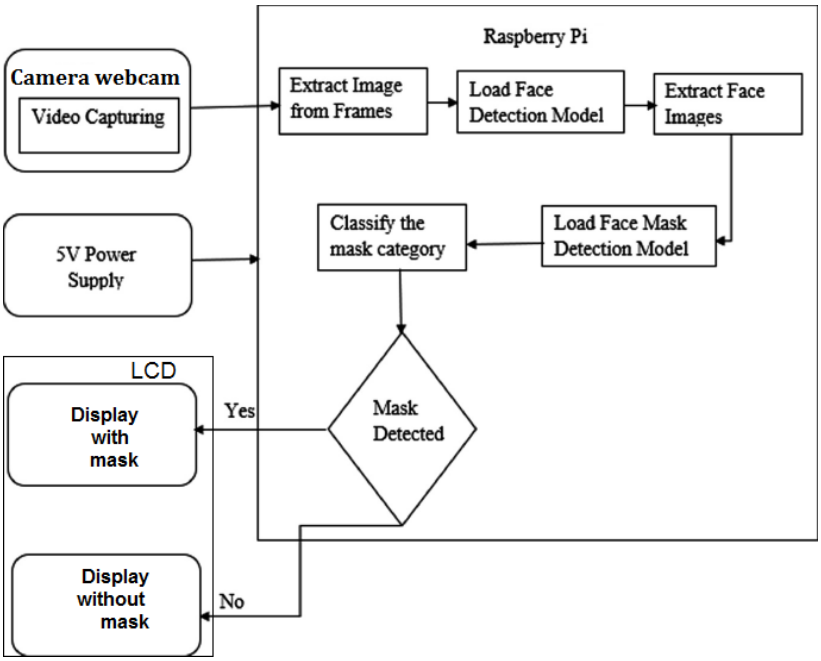


Fig. 3.11 Workflow of system face mask detection in hardware

networks so that they may fit into our resource-constrained edge devices or mobile devices.the compression techniques: knowledge distillation, pruning, quantization.

3.4.1 Sparsification

Sparsification is the removal of unnecessary information from an overprecise and over-parameterized deep learning model, resulting in a quicker and smaller model. Sparsification techniques range from producing sparsity via pruning and quantization to permitting naturally occurring sparsity via activation sparsity or Winograd/FFT. When utilized properly, these strategies result in much more performant and smaller models with little to no influence on the baseline measurements. Model compilation optimization is a post-training technique that can enhance model artifact and model execution alignment with underlying hardware [16].

3.4.2 Pruning

Network pruning is a useful method for lowering memory space and bandwidth. This made it easier to install neural networks in limited areas like embedded devices. Pruning removes unnecessary parameters or neurons that do not contribute significantly to the accuracy of the result. This circumstance happens when the

weight coefficients are zero, near to zero, or repeated. As a result, pruning decreases computing complexity. When pruned networks are retrained, they may be able to avoid a prior local minimum and enhance accuracy even more [46].

Pruning can be done on an element-by-element, row-by-row, column-by-column, filter-by-filter, or layer-by-layer basis. In most cases, element by element sparsity has the least influence and results in an unstructured model[46].

There are various types of modern pruning procedures, including:

- Pruning is classified into two types: structured pruning and unstructured pruning, depending on whether the pruned network is symmetric or not,
- Depending on the kind of pruned piece, neuron and connection pruning,
- and static and dynamic pruning.

3.4.3 Quantization

Model quantization is another strategy for enhancing speed and lowering memory needs by computing and storing tensors at lower bitwidths (such as INT8 or FLOAT16) than floating-point precision. This is especially useful during model deployment. Quantization is useful when large models must be served on machines with limited memory, or when switching between models is required and reducing I/O time is critical.

Quantization research is divided into two categories:

- quantization aware training (QAT)
- and post-training quantization (PTQ).

Post-training quantization is the process of taking a trained model, quantifying the weights, and then re-optimizing the model to obtain a quantized model with scales. Fine-tuning a stable full precision model or retraining the quantized model is required for quantization awareness training [26].

3.4.4 Platform used (Deci)

The Deci platform makes it simple to manage,deploy, optimize, and serve models in your production environment. You can continue to use popular deep learning frameworks like Keras, TensorFlow, ONNX and PyTorch. All you need is our web-based platform for your Python client to run it from your code[31].

- **Deci Score**
  Deci offers a standardized/normalized score to describe the efficiency of a model's runtime performance in a given production setting while maintaining accuracy. A model's performance can be divided into two categories: accuracy and efficiency. A Deci Score is a normalized score between one and ten that assesses a model's runtime efficiency on a given hardware and batch size configuration. It includes measurements and statistics for Accuracy, Throughput, Latency, Memory Footprint, and Model Size [20].

- **Quantization Level**
  Specify whether this model should go through a quantization process and the level of this quantization process. The lower the level you select here, the more accuracy may be compromised. Only 16-bit and 32-bit are available for the free version [25].

To improve our model we follow these steps:

- **Step 1 – convert basec model to model ONNX format** befor launching platform,will must convert our model "*mask_detection.model*" for to get an ONNX 1.8.0 format model and must support inference with dynamic batch size .

- **Step 2 – Launching the Deci Platform and adding our Model**
  Launching the Deci Platform and adding our Model The following describes how to upload my model to the Deci Platform. This model can then serve as the baseline model to be optimized.

  To add a model to be optimized (we are display step optimize our model)

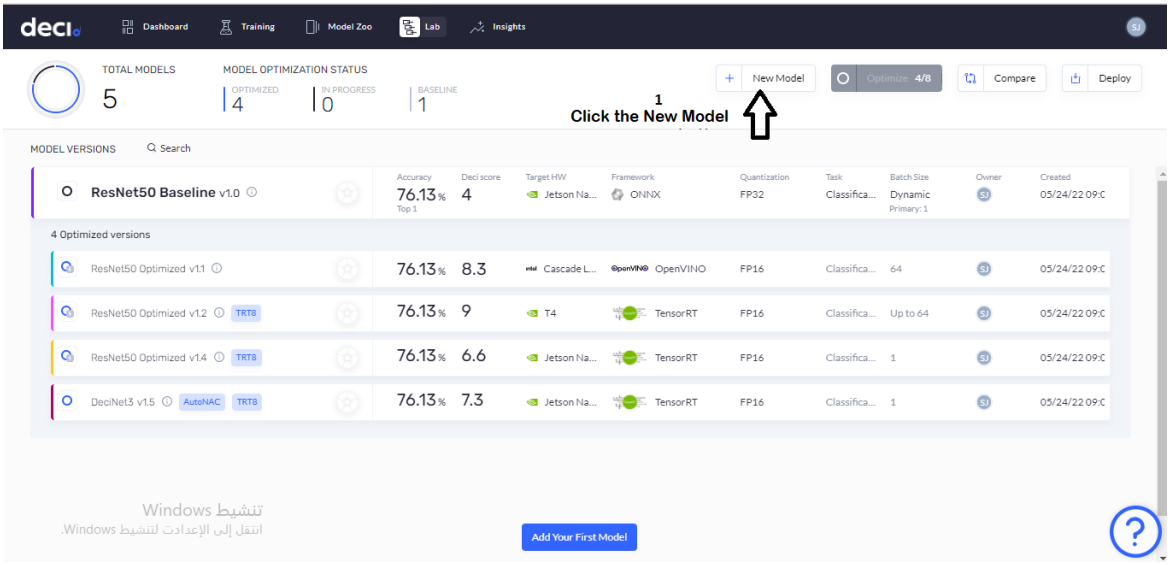


Fig. 3.12 Deci platform

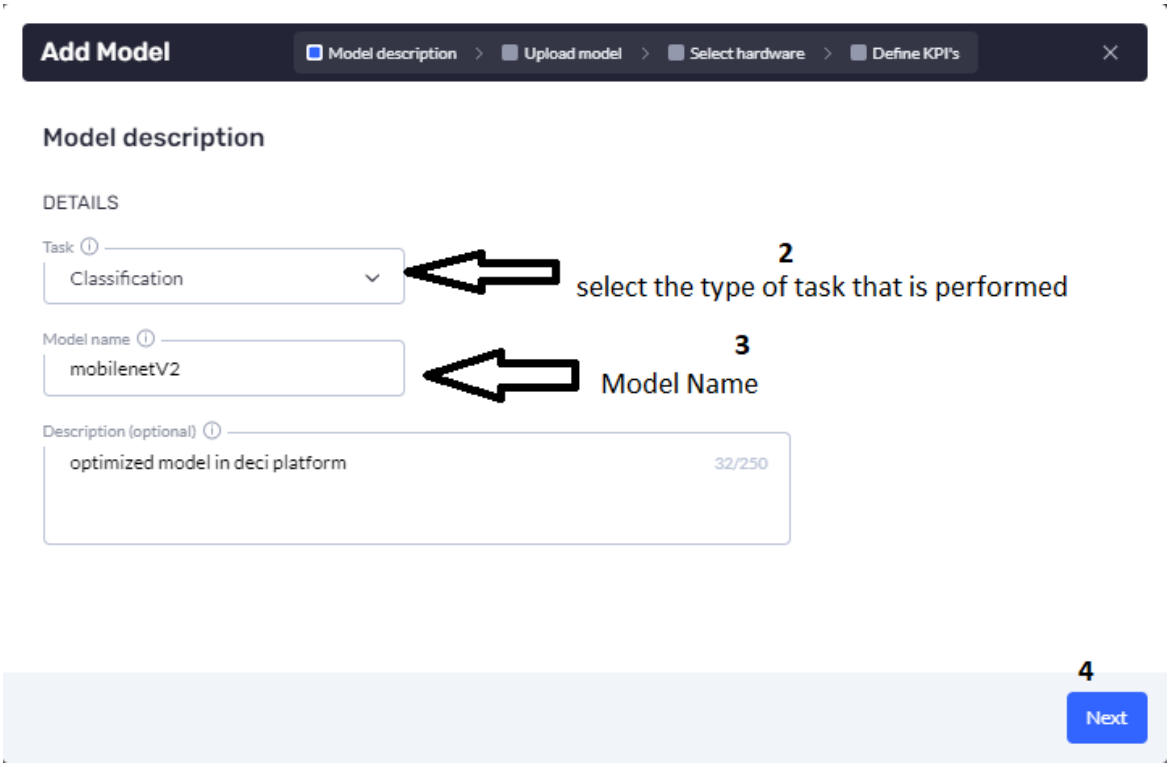


Fig. 3.13 Model description

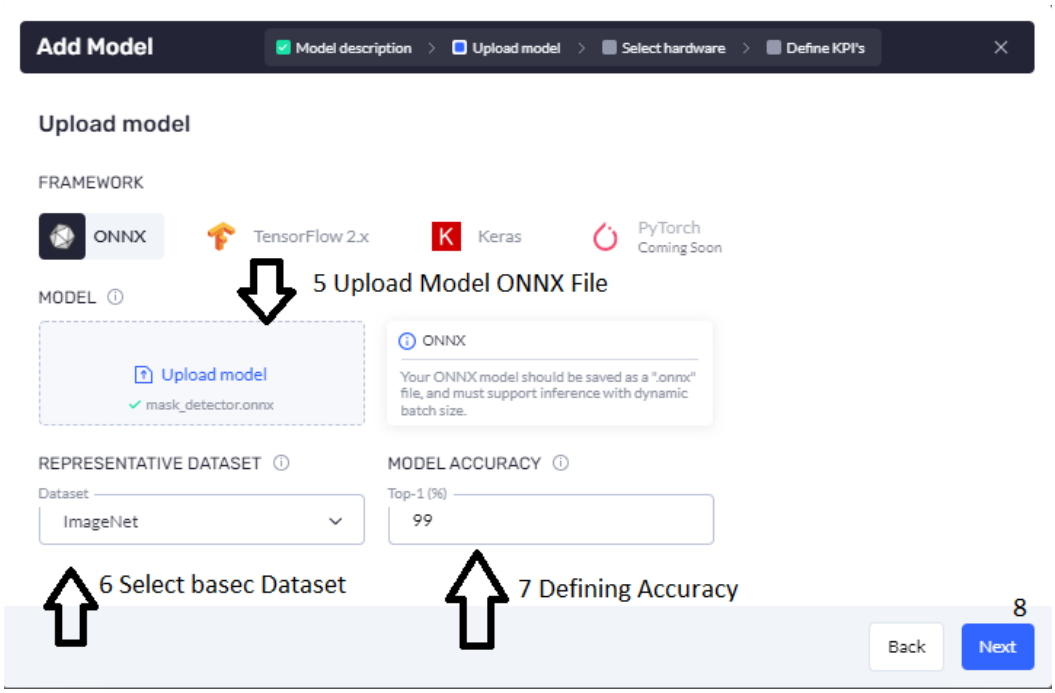


Fig. 3.14 Upload our model

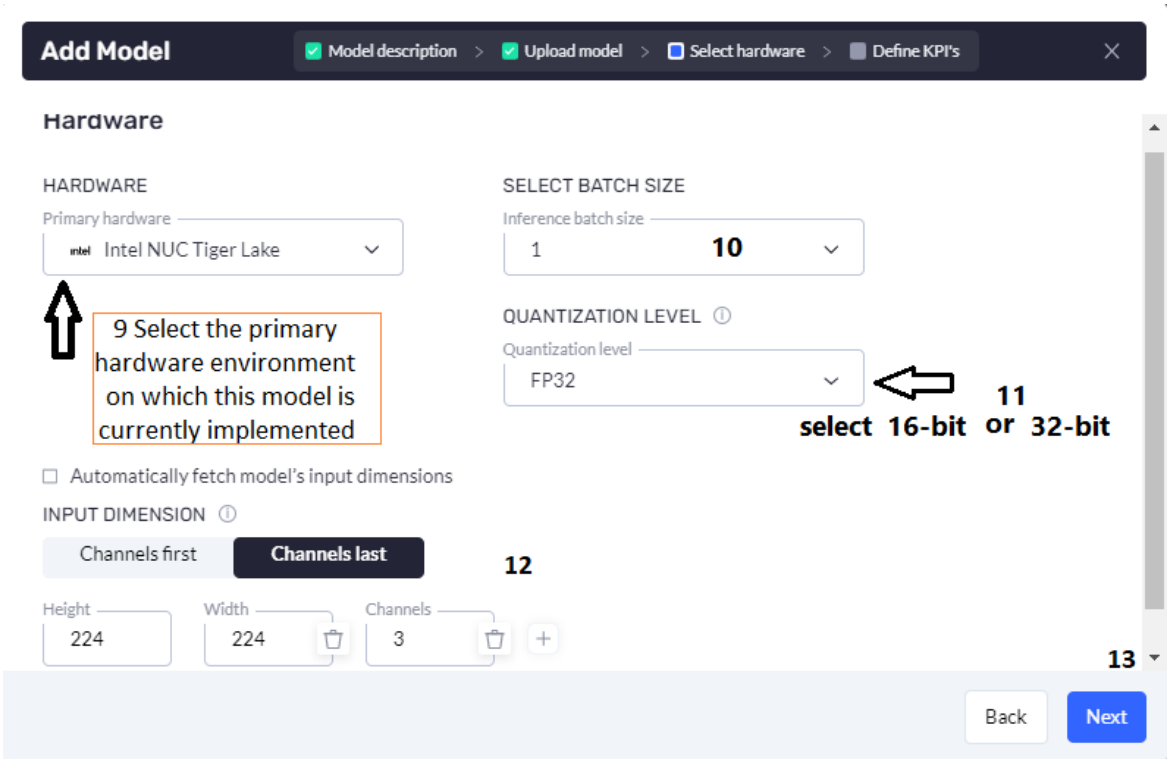


Fig. 3.15 Hardware

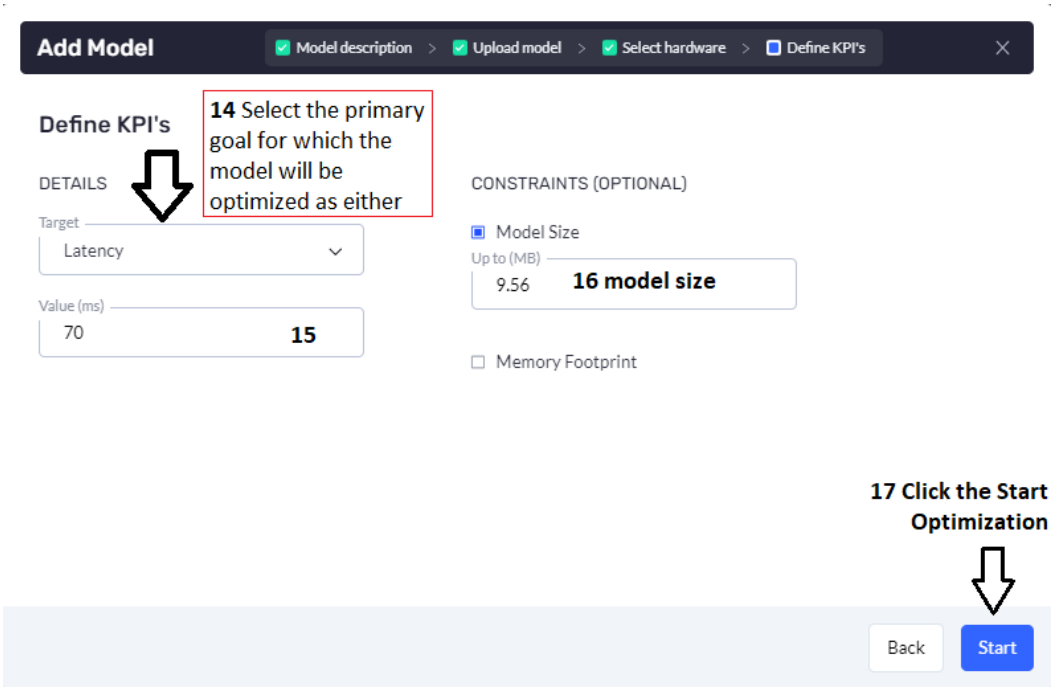


Fig. 3.16 Select goal optimize

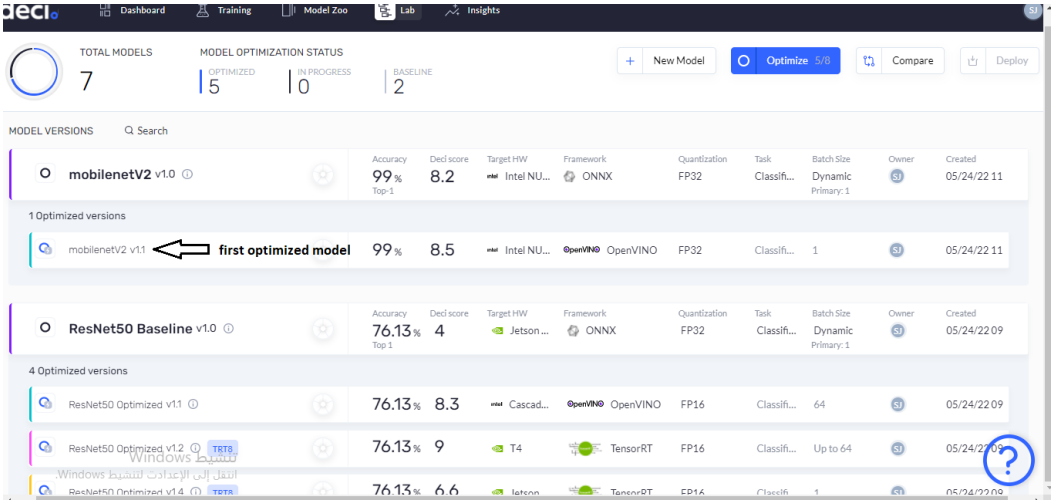


Fig. 3.17 optimized versions of that baseline model

3.5 conclusion

In this chapter, we talked about working methodology, deep learning algorithms and their uses for developing an intelligent system, the steps involved, training and deployment a model, as well as the architecture. In the next chapter, we will go over the implementation, discuss the results obtained, and other things.

Chapter 4

Implementation and results

4.1 Introduction

In this chapter, we will outline the project's objectives and needs. Furthermore, numerous processes will be taught step by step as to how these tasks are implemented in this system. We will define and describe the majority of the software that we will be utilizing in this project. We are collecting results and then discussing them.

4.2 Development environment

In this part, we will use the following software and hardware resources:

4.2.1 Software environment for model implementation

4.2.1.1 Open CV

OpenCV (Open Source Computer Vision Library) is a free and open-source computer vision and machine learning software library. OpenCV also supports a wide range of programming languages, including Python, C++, and Java, and is available on a variety of platforms, including Windows, OS X, Linux, iOS, and Android. CUDA and OpenCL-based interfaces for high-speed GPU operations [6].

4.2.1.2 TensorFlow

TensorFlow is an open-source software library for dataflow and differential programming for a variety of tasks. Similarly, neural networks use TensorFlow in machine

learning. TensorFlow was developed by Google in 2011 under the name DistBelief and was officially released for free in 2017. The library can run on multiple CPUs and GPUs and is available on a variety of platforms, including mobile [9].

4.2.1.3 Keras

Keras is a high-level neural network API that enables quick experimentation. The ability to move fast from idea to outcome is crucial for excellent research. It supports arbitrary network structures including multi-input/multi-output models, layer sharing, model sharing, and so on. As a result, Keras may be used to create nearly any deep learning model, from a memory network to a neural Turing machine [7].

4.2.1.4 Python

Python is an interpreted object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, together with dynamic typing and dynamic binding, make it an excellent choice for usage as a scripting or glue language to link existing components. Python's easy-to-learn syntax prioritizes readability, which reduces software maintenance expenses. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and substantial standard library are available in source or binary form for all major systems[33].

4.2.1.5 Anaconda

Anaconda is an open-source data science distribution for the Python and R programming languages that aims to simplify package management and deployment. Anaconda's package management system,conda, manages package versions by analyzing the current environment before executing an installation to avoid disrupting other frameworks and packages [29].

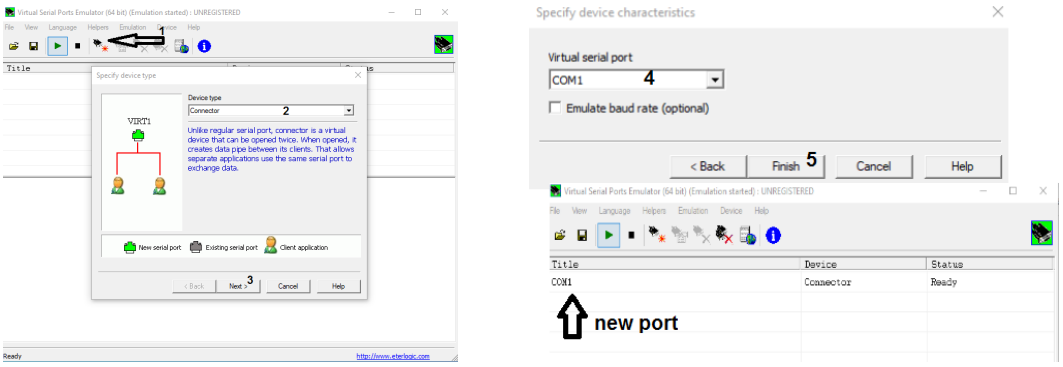
4.2.2 Hardware environment

The edge device used in this project is the Raspberry Pi 3. Since it was not available at this time, we used a software tool simulation called Proteus as an alternative.

4.2.2.1 VSPE

The Virtual Serial Ports Emuletor (VSPE) is designed to assist software engineers and developers in creating, debugging, and testing applications that use serial ports[17]. Its role is to create can create various virtual devices for data transmission and reception between Proteus and code source in IDLE python 3.9 . Below steps creation COM port in vspe.

Fig. 4.1 VSPE [17]



(a) (b)

Fig. 4.2 creation steps COM

4.2.2.2 Proteus Professional

Proteus is a computer-aided design software package for electronic circuits. This package, created by Labcenter Electronics, is a circuit simulation system. The ability to simulate the operation of programmable devices such as the Raspberry Pi is a unique feature of the Proteus Professional 8.13 package[14].

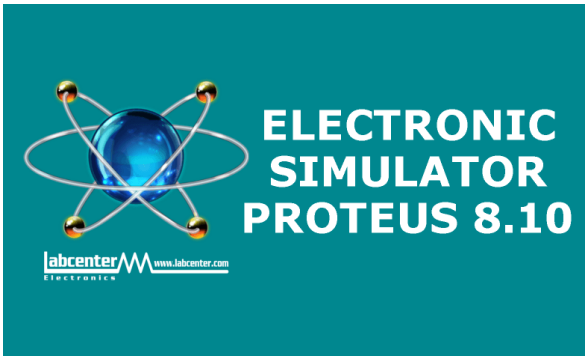


Fig. 4.3 Proteus 8 professional [14]

Installation proteus

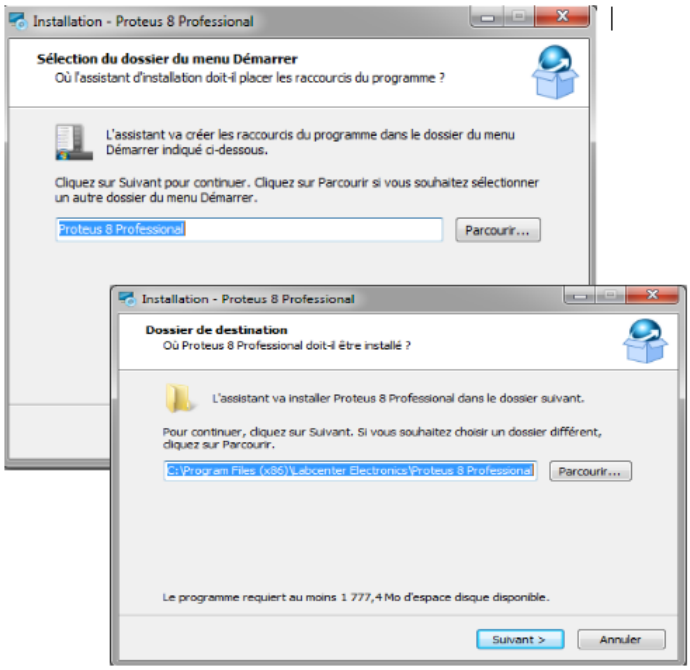


Fig. 4.4 Step 1: Installation proteus 8 professional

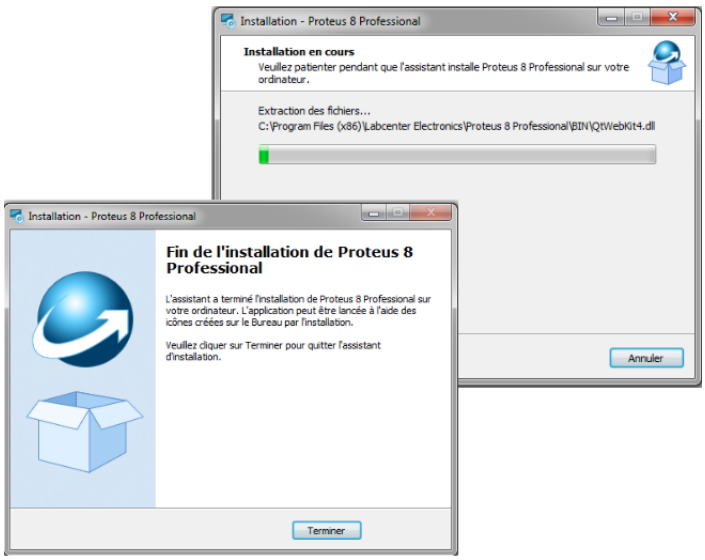


Fig. 4.5 Step 2: Installation proteus 8 professional

create first our project

firstly Start Proteus, To get started as a step 1,creating Project as illustrated in the following the Proteus Home Page is shown4.6,The next step of the New Project to be put name project "*RaspberryPI.pdsprj*" in 4.7,Firmware selection options for New Project in 4.8,The last stage of the new project The wizard shows an overview of the details for the newly established project4.9.Proteus will immediately launch the newly generated project once you click the Finish button 4.10.

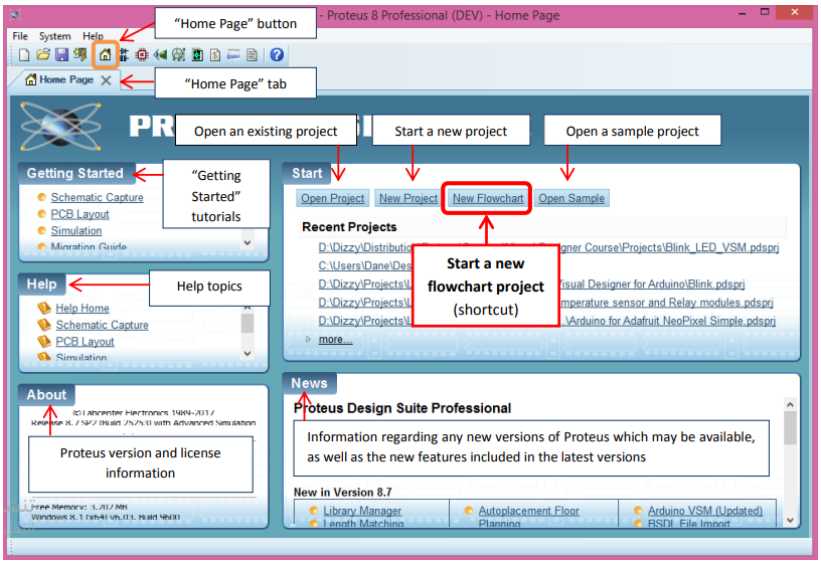


Fig. 4.6 Proteus home page

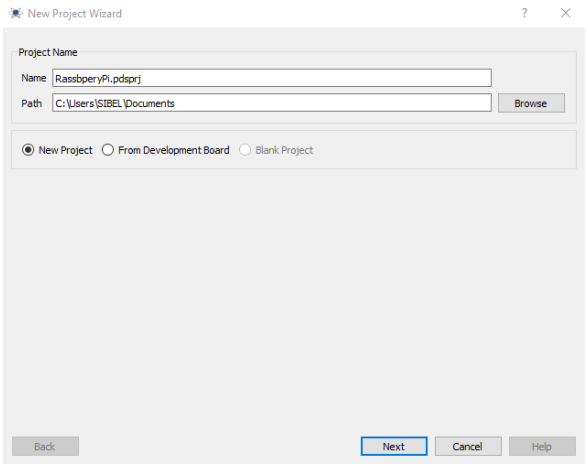


Fig. 4.7 New project wizard for starting a new project

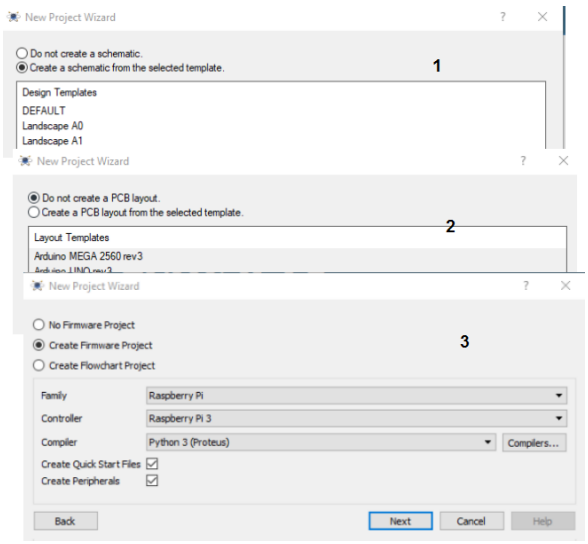


Fig. 4.8 Firmware selection options for new project

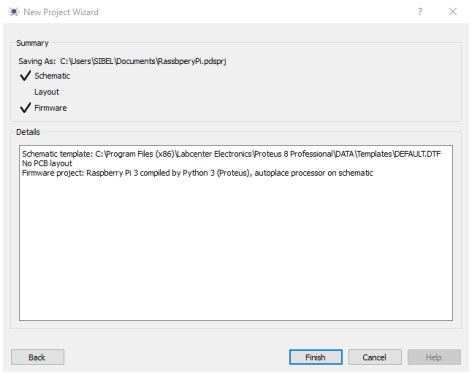


Fig. 4.9 Summary of details for new project

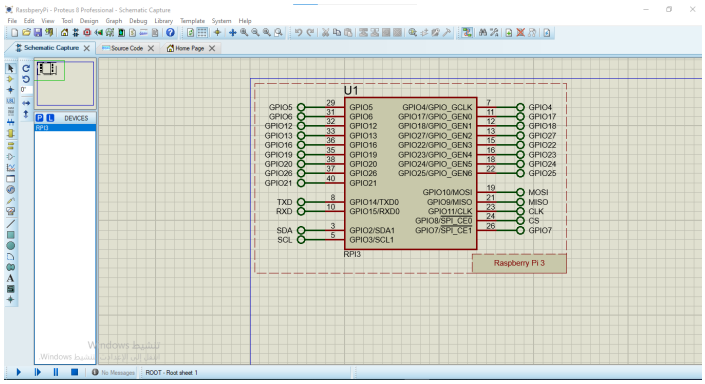


Fig. 4.10 New project

4.3 Mechanism the proposed system work

Proteus is allowing you to design, simulate and debug complete Raspberry Pi systems.so,users can create a Raspberry PI controlling program and then simulate and the entire system in software as explained in as shown in fige ,requirement the simulation system +0it is:

- Raspberry Pi 3
- COMPIM model is a physical interface model of a serial port. Incoming serial data is buffered and presented to the circuit as a digital signal at the PC's physical COM port (port COM1 which created in VSPE).
- LCD is the acronym for Liquid Crystal Display , It displays text messages on an LCD display 4.14.

After then,we was run Proteus project *RaspberryPI.pdsprj* , and run VSPE ,and open IDLE(python 3.9)and open *detect_mask_video.py* file ,then press "Run module"

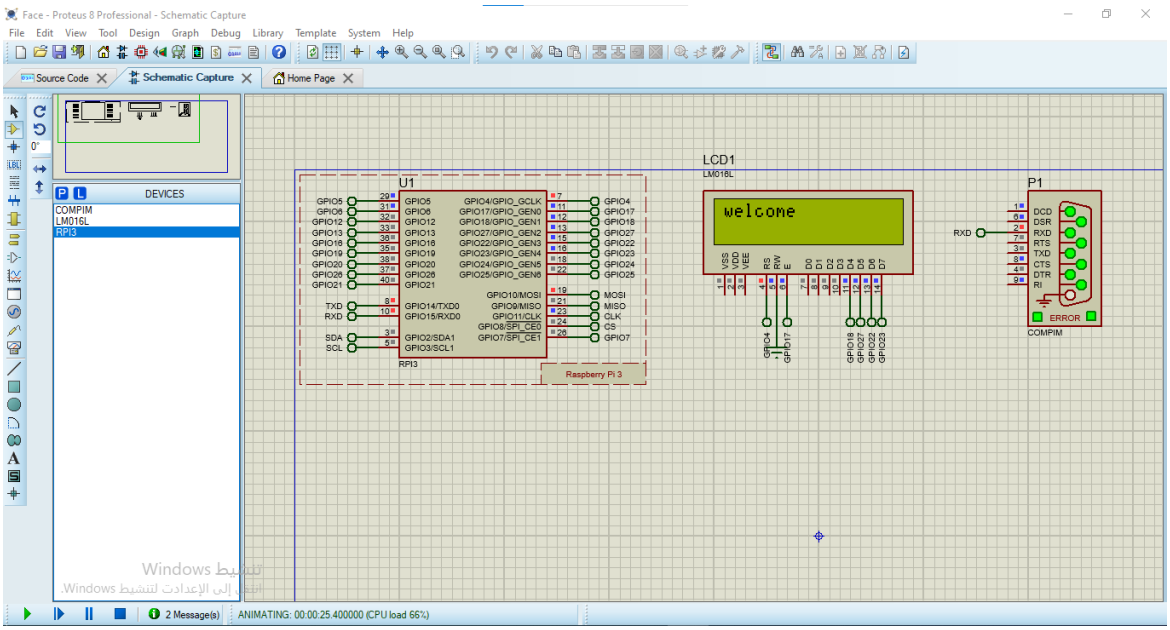


Fig. 4.11 Circuit our project.

.it will permission A new window appears 4.16. This script will open the camera sensor of the used device (laptop camera).to while then the recognition of face mask.

4.3.1 Loading and using models

load our serialized face detector model from disk,and the face mask detector model optimized from disk ,which we trained it on dataset

```
# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
# load the face mask detector model optimized  from disk
mode_optimize = infery.load(model_path='mobilenetV2_1_1.pkl',
    framework_type='openvino', inference_hardware='cpu')
```

Fig. 4.12 Load the models.

4.3.2 Load and Use our system to predict

it we write the following code to predict mask detection :

```
# only make a predictions if at face was detected
    faces = np.array(faces, dtype="float32")
    preds = maskNet.predict(faces)
```

Fig. 4.13 Predict Function.

detect faces in the frame and determine if they are wearing a face mask or not

```
# detect faces in the frame and determine if they are wearing a face mask or not
(locs, preds) = detect_and_predict_mask(frame, faceNet, mode_optimize)
```

Fig. 4.14 Detect Mask Function

4.3.3 Detector system based on the optimized model

We can notice that the proposed method succeeded to help achieve the main goals, this system we can detect the classification with efficiency, regardless to video quality

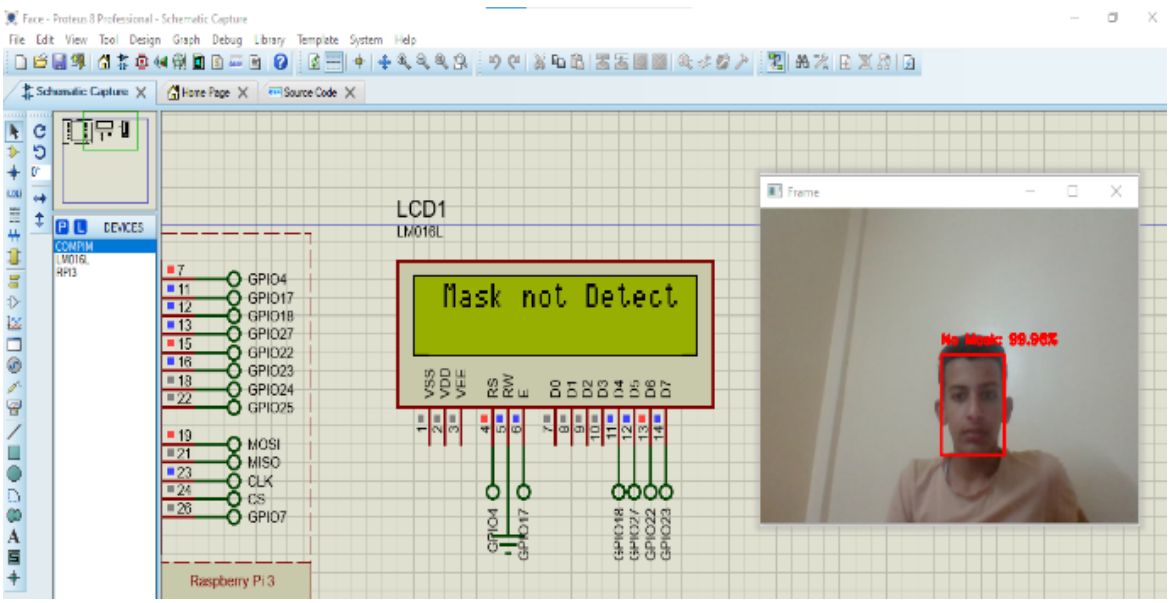


Fig. 4.15 Results of detection mask not detect

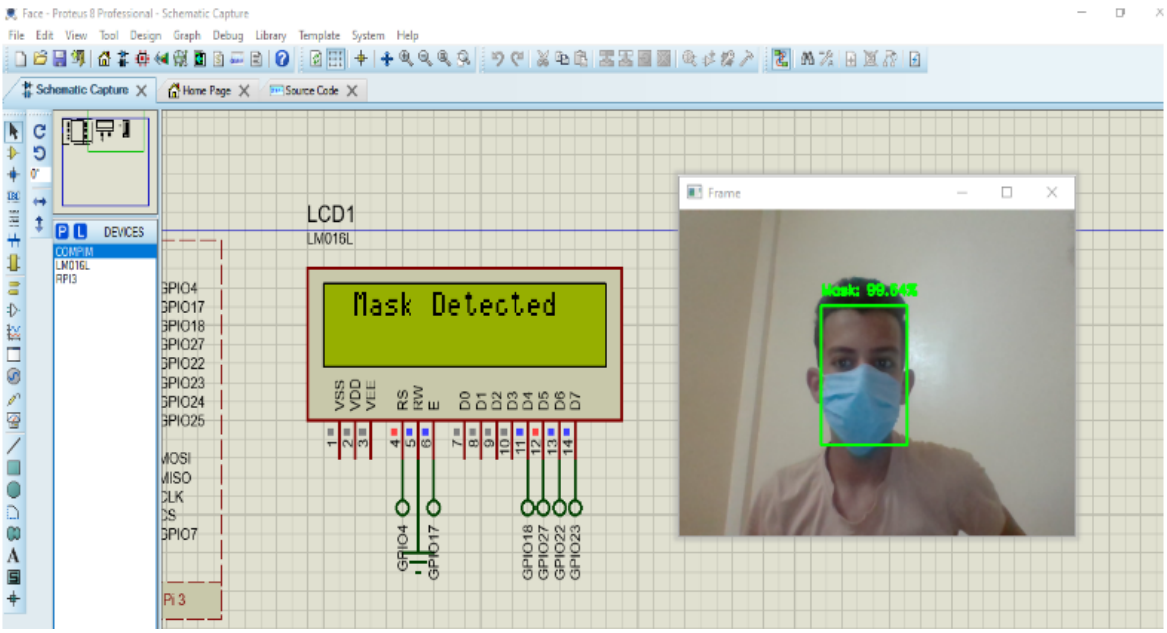


Fig. 4.16 Results of detection mask detect

4.4 Performance evaluation of the proposed approach

We trained the model on Google Collaboratory (colab) in the cloud, which provides a Jupiter notebook environment and allows the creation of notebooks with Python. Google colab has a free Jupiter while providing free access to computing resources such as GPUs[22]. Model training yielded loss and accuracy values. performance metrics of model training and does not take a long time to execute. Time spent for training is 1h 03m. The result appears in the figure4.17.

```
[INFO] training head...
Epoch 1/20
53/53 [==============================] - 81s 1s/step - loss: 0.4550 - accuracy: 0.8192 - val_loss: 0.1970 - val_accuracy: 0.9741
Epoch 2/20
53/53 [==============================] - 74s 1s/step - loss: 0.1823 - accuracy: 0.9628 - val_loss: 0.1016 - val_accuracy: 0.9835
Epoch 3/20
53/53 [==============================] - 76s 1s/step - loss: 0.1288 - accuracy: 0.9724 - val_loss: 0.0747 - val_accuracy: 0.9859
Epoch 4/20
53/53 [==============================] - 73s 1s/step - loss: 0.0889 - accuracy: 0.9790 - val_loss: 0.0634 - val_accuracy: 0.9859
Epoch 5/20
53/53 [==============================] - 73s 1s/step - loss: 0.0742 - accuracy: 0.9808 - val_loss: 0.0524 - val_accuracy: 0.9859
Epoch 6/20
53/53 [==============================] - 74s 1s/step - loss: 0.0707 - accuracy: 0.9790 - val_loss: 0.0513 - val_accuracy: 0.9859
Epoch 7/20
53/53 [==============================] - 74s 1s/step - loss: 0.0586 - accuracy: 0.9844 - val_loss: 0.0483 - val_accuracy: 0.9859
Epoch 8/20
53/53 [==============================] - 74s 1s/step - loss: 0.0493 - accuracy: 0.9886 - val_loss: 0.0444 - val_accuracy: 0.9882
Epoch 9/20
53/53 [==============================] - 74s 1s/step - loss: 0.0535 - accuracy: 0.9856 - val_loss: 0.0461 - val_accuracy: 0.9859
Epoch 10/20
53/53 [==============================] - 74s 1s/step - loss: 0.0407 - accuracy: 0.9910 - val_loss: 0.0452 - val_accuracy: 0.9859
Epoch 11/20
53/53 [==============================] - 74s 1s/step - loss: 0.0418 - accuracy: 0.9898 - val_loss: 0.0471 - val_accuracy: 0.9859
Epoch 12/20
53/53 [==============================] - 74s 1s/step - loss: 0.0323 - accuracy: 0.9928 - val_loss: 0.0444 - val_accuracy: 0.9859
Epoch 13/20
53/53 [==============================] - 74s 1s/step - loss: 0.0364 - accuracy: 0.9886 - val_loss: 0.0376 - val_accuracy: 0.9929
Epoch 14/20
53/53 [==============================] - 74s 1s/step - loss: 0.0341 - accuracy: 0.9928 - val_loss: 0.0414 - val_accuracy: 0.9859
Epoch 15/20
53/53 [==============================] - 74s 1s/step - loss: 0.0313 - accuracy: 0.9922 - val_loss: 0.0337 - val_accuracy: 0.9906
Epoch 16/20
53/53 [==============================] - 74s 1s/step - loss: 0.0294 - accuracy: 0.9934 - val_loss: 0.0329 - val_accuracy: 0.9906
Epoch 17/20
53/53 [==============================] - 74s 1s/step - loss: 0.0330 - accuracy: 0.9904 - val_loss: 0.0339 - val_accuracy: 0.9906
Epoch 18/20
53/53 [==============================] - 74s 1s/step - loss: 0.0293 - accuracy: 0.9892 - val_loss: 0.0369 - val_accuracy: 0.9929
Epoch 19/20
53/53 [==============================] - 74s 1s/step - loss: 0.0235 - accuracy: 0.9934 - val_loss: 0.0360 - val_accuracy: 0.9929
Epoch 20/20
53/53 [==============================] - 74s 1s/step - loss: 0.0319 - accuracy: 0.9928 - val_loss: 0.0353 - val_accuracy: 0.9929
```

Fig. 4.17 Train the model

It can be seen that the network trained for 20 epochs and we achieved high accuracy (99%) and low loss following training loss as shown in the Graph 4.18 below.

Fig. 4.18 Graph training loss and accuracy

Accuracy is defined as the proportion of correct predictions made to the total number of predictions made [44]. This system has an accuracy of nearly 99 percent. The formula for calculating accuracy is as follows:

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

The ratio of correct positive predictions to total positive predictions is defined as precision[44]. The achieved precision is 0.99, which is the optimal precision value. The following is the precision formula:

$$Precision = \frac{TP}{TP + FP}$$

The recall is defined as the ratio of correct positive predictions to total positive examples[44]. The recall value achieved is similarly 0.99, which is a high and respectable result . Here is the recall formula:

$$Recall = \frac{TP}{TP + FN}$$

The F1-score is the mean of the accuracy and recall scores[44]. The resulting F1-score is 0.99. The formula for calculating F1-score is:

$$F1 - score = \frac{precision + recall}{2}$$

Table 4.1 Evaluating network

	precision	recall	f1-score	support
with_mask	0.99	1.00	0.99	511
without_mask	0.99	0.99	0.99	387
accuracy			0.99	898
macro avg	0.99	0.99	0.99	898
weighted avg	0.99	0.99	0.99	898

4.5 Results and discussion

Table 4.2 shows the comparison of thees optimized model with baseline model which are implemented under the working of the system was tested with the dynamic input.

The comparison shows that the proposed optimized model has the same accuracy as the other baseline model. Deci currently does not validate the model's accuracy. It just displays the precision that we declared when the model was uploaded [21]. The best latency and Throughput is obtained with improvement 1.8x from the basic. The primary goal for which the model was optimized as either Throughput or Latency ,it was optimizing the model to process the most requests and to ensure the quickest response.In addition to,optimized Model size is less 1.9x with size 4.93 MB . which we was noticed in when detect mask face with detector.

Table 4.2 Benchmarking performance for models .

measures	baseline model mobilenetv2	optimize model mobilnetv2	Performance
Accuracy Top1	99%	99%	0%
Troughput	266.4 fps	470.7 fps	1.8x
Latency	3.754 ms	2.1243 ms	Improvement 1.8x
Model size	9.56 MB	4.93 MB	reduction 1.9x
Deci scor	8.2	8.5	0.4

4.6 conclusion

In this last chapter we shed light on the work environment, and the components we will use with the simulator Proteus, We analyzed the results of the tests on this classifier that we did from the webcam. An Edge-AI approach is widely expanded these days according to its efficiency and usability, the aim of this work is to protecting people by deploying a system that has an inference to detect and classify a mask face case, this project provides high performance in real-time of face mask detecting.

GENERAL CONCLUSION

At the end of this study, and after performing the required research for this project which we covered in our thesis. We spent a lot of time reading publications and articles related previously to see what is the best model about classification and to be able to apply model used by us. We reached our objective from this study, which was to build system IOT Edge deployment to address the use case of automatic recognition of face masks on edge computing devices such as :Raspberry Pi. The system implementation was made by Anaconda Environment, Python, TensorFlow and Open CV. Also, what distinguishes our system from other systems is that, the good accuracy rate. During the different chapters we described and clarify the system architecture. The goals achieved are:

- Fast training model
- We compressed our trained model so that it became smaller and faster in answering using the quantization technique.
- We focused on quantization techniques that worked on the size of the model that used, so that it decreased in a semantic way without having a significant impact on the accuracy, in the way the model classifies the mask.
- Quantization techniques adapt the model to the edge computing properties without decreasing the detection accuracy of the model itself.
- Fast and real time detection of mask.
- Accuracy and efficient system.

In addition to, we should mention the major difficulty we faced in, we could not apply technique pruning to optimize and to compress model in order to used usually at edge device computing with limited resources. Also, we did not apply our system at real device ,for extracting real and actual result. Finally, Our work can be used in airport systems or hospitals or at the entrance of companies for surveillance. As an

addition and an initial idea for another project, it is possible to use other categories to determine the type of mask or to determine a person's age or gender.

References

[1] (2018). Cloudlets extend cloud power to edge with virtualized delivery. [online] https://www.techtarget.com/searchcloudcomputing/tip/Cloudlets-extend-cloud-power-to-edge-with-virtualized-delivery.

[2] (2018). Fog computing vs edge computing. [online] https://erpinnews.com/fog-computing-vs-edge-computing/.

[3] (2018). Inference with mobilenet. [online] https://developer.ridgerun.com/wiki/index.php?title=GstInference/Supported_architectures/MobileNet.

[4] (2019). How iot works. [online] https://data-flair.training/blogs/how-iot-works/.

[5] (2020). 5g: What it means for iot ? [online] https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/.

[6] (2020). About opencv. [online] https://opencv.org/about/.

[7] (2020). Keras. [online] https://keras.rstudio.com/.

[8] (2020). Top 5 cloud platforms and solutions to choose from. [online] https://www.newgenapps.com/blogs/top-5-cloud-platforms-and-solutions-to-choose-from/.

[9] (2020). What is tensorflow? [online] https://deepai.org/machine-learning-glossary-and-terms/tensorflow.

[10] (2020). what is the network edge. [online] https://www.intel.ca/content/www/ca/fr/edge-computing/what-is-the-network-edge.html#articleparagraph_452.

[11] (2021). Edge ai. [online] https://neuralnetlab.com/edge-ai/.

[12] (2021). Main cloud service models: Iaas, paas and saas. [online] https://www.stackscale.com/blog/cloud-service-models/#IaaS_PaaS_and_Saas.

[13] (2021). A project report submitted in partial fulfillment of the requirements for the award of the degree of. [online] http://cse.anits.edu.in/projects/projects2021C4.pdf.

[14] (2021). Proteus professional. [online] https://en.taiwebs.com/windows/download-proteus-professional-2164.html.

[15] (2021). Public vs private vs hybrid: Which cloud models do companies prefer? [online] https://www.mygreatlearning.com/blog/public-vs-private-vs-hybrid-which-cloud-models-do-companies-prefer/.

[16] (2021). Sparsification.

[17] (2021). Virtual serial ports emulator. [online] http://www.eterlogic.com/Products.VSPE.html.

[18] (2021a). What is mobile edge computing (mec)? [online] https://enterprisersproject.com/article/2021/2/what-mobile-edge-computing-mec.

[19] (2021b). What is multi-access edge computing? [online] https://www.ibm.com/cloud/blog/what-is-multi-access-edge-computing.

[20] (2022). Concepts and terms. [online] https://docs.deci.ai/docs/concepts-and-terms.

[21] (2022). Exploring an example project. [online] https://docs.deci.ai/docs/step-2.

[22] (2022). Frequently asked questions. [online] https://research.google.com/colaboratory/faq.html.

[23] (2022). Iot edge. [online] https://www.fortinet.com/resources/cyberglossary/iot-edge.

[24] (2022). machine-learning. WebPage. [online] https://sensiml.com/entity/machine-learning/.

[25] (2022a). Optimizing your model. [online] https://docs.deci.ai/docs/step-4-optimizing-your-model.

[26] (2022b). pruning and quantization.

[27] (2022). Real-life use cases for edge computing. [online] https://innovationatwork.ieee.org/real-life-edge-computing-use-cases/.

[28] (2022). What is 5g? WebPage. [online] https://www.techtarget.com/searchnetworking/definition/5G.

[29] (2022). What is anaconda? [online] https://www.dominodatalab.com/data-science-dictionary/anaconda.

[30] (2022). What is cloud computing? WebPage. [online] https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/.

[31] (2022). What is deci? [online] https://docs.deci.ai/docs/what-is-deci.

[32] (2022). What is iot? the internet of things defined and explained. WebPage. [online]https://www.i-scoop.eu/internet-of-things-iot/internet-of-things-what-definition/.

[33] (2022). What is python? executive summary. [online] https://www.python.org/doc/essays/blurb/.

[34] Alissa, S. and Zarka, N. (2017). Next generation 5g wireless network.

[35] Barakabitze, A. A., Ahmad, A., Mijumbi, R., and Hines, A. (2020). 5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 167:106984.

[36] Benitez Baltazar, V. H., Pacheco Ramírez, J. H., Moreno Ruiz, J. R., and Nuñez Gurrola, C. (2021). Autonomic face mask detection with deep learning: an iot application. *Mexican Journal of Biomedical Engineering*, 42(2):160–170.

[37] Boulila, W., Alzahem, A., Almoudi, A., Afifi, M., Alturki, I., and Driss, M. (2021a). A deep learning-based approach for real-time facemask detection. *CoRR*, abs/2110.08732.

[38] Boulila, W., Alzahem, A., Almoudi, A., Afifi, M., Alturki, I., and Driss, M. (2021b). A deep learning-based approach for real-time facemask detection. *CoRR*, abs/2110.08732.

[39] Burhan, M., Rehman, R. A., Khan, B., and Kim, B.-S. (2018). Iot elements, layered architectures and security issues: A comprehensive survey. *Sensors*, 18(9).

[40] Giron, N. N. F., Billones, R. K. C., Fillone, A. M., Del Rosario, J. R., Bandala, A. A., and Dadios, E. P. (2020). Classification between pedestrians and motorcycles using fasterrcnn inception and ssd mobilenetv2. In *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pages 1–6.

[41] Hamdan, S., Ayyash, M., and Almajali, S. (2020). Edge-computing architectures for internet of things applications: A survey. *Sensors*, 20(22).

[42] Huang, D. and Wu, H. (2018). Chapter 1 - mobile cloud computing taxonomy. In Huang, D. and Wu, H., editors, *Mobile Cloud Computing*, pages 5–29. Morgan Kaufmann.

[43] Huang, H., Miao, W., Min, G., and Luo, C. (2019). Mobile edge computing for the 5g internet of things. pages 143–161.

[44] Kommaraju, R., Humanvitha Sai Dharani, C., Mansoor, S., and Pujitha, S. (2022). Real-time face mask detection at gateway using opencv and iot. In Pandian, A. P., Palanisamy, R., Narayanan, M., and Senjyu, T., editors, *Proceedings of Third International Conference on Intelligent Computing, Information and Control Systems*, pages 519–531, Singapore. Springer Nature Singapore.

[45] Li, E., Zeng, L., Zhou, Z., and Chen, X. (2019). Edge ai: On-demand accelerating deep neural network inference via edge computing.

[46] Liang, T., Glossner, J., Wang, L., Shi, S., and Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403.

[47] Liu, Y., Peng, M., Shou, G., Chen, Y., and Chen, S. (2020). Toward edge intelligence: Multiaccess edge computing for 5g and internet of things. *IEEE Internet of Things Journal*, 7(8):6722–6747.

[48] Lundervold, A. S. and Lundervold, A. (2019). An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127. Special Issue: Deep Learning in Medical Physics.

[49] Mohandas, R., Bhattacharya, M., Penica, M., Van Camp, K., and Hayes, M. J. (2021). On the use of deep learning enabled face mask detection for access/egress control using tensorflow lite based edge deployment on a raspberry pi. pages 1–6.

[50] Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., and Hemanth, J. (2021). Ssdmnv2: A real time dnn-based face mask detection system using single shot multibox detector and mobilenetv2. *Sustainable Cities and Society*, 66:102692.

[51] Rahimi, H., Zibaeenejad, A., and Safavi, A. A. (2018). A novel iot architecture based on 5g-iot and next generation technologies. pages 81–88.

[52] Ramalingam, B., Elara Mohan, R., Balakrishnan, S., Elangovan, K., Félix Gómez, B., Pathmakumar, T., Devarassu, M., Mohan Rayaguru, M., and Baskar, C. (2021). stetro-deep learning powered staircase cleaning and maintenance reconfigurable robot. *Sensors*, 21(18).

[53] Sethi, S., Kathuria, M., and Kaushik, T. (2021). Face mask detection using deep learning: An approach to reduce risk of coronavirus spread. *J Biomed Inform*, 120:103848.

[54] Shinde, R. K., Alam, M. S., Park, S. G., Park, S. M., and Kim, N. (2022). Intelligent iot (iiot) device to identifying suspected covid-19 infections using sensor fusion algorithm and real-time mask detection based on the enhanced mobilenetv2 model. *Healthcare*, 10(3).

[55] Sun, X. and Ansari, N. (2016). Edgeiot: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, 54:22–29.

[56] Tanuja R., Mahesh S., S. M. and R, V. K. (October 2021). Real world face mask detection using mobilenetv2 andraspberry pi. *International Journal of Engineering Research and Technology (IJERT)*, 11.

[57] Teboulbi, S., Messaoud, S., Hajjaji, M. A., and Mtibaa, A. (2021). Real-time implementation of ai-based face mask detection and social distancing measuring system for covid-19 prevention. *Scientific Programming*, 2021:8340779.

[58] Tzafestas, S. G. (2018). The internet of things: A conceptual guided tour. *European Journal of Advances in Engineering and Technology*, 5(10):745–767.

[59] Varshini, B., Yogesh, H., Pasha, S. D., Suhail, M., Madhumitha, V., and Sasi, A. (2021). Iot-enabled smart doors for monitoring body temperature and face mask detection. *Global Transitions Proceedings*, 2(2):246–254. International Conference on Computing System and its Applications (ICCSA- 2021).

[60] Vhora, F. and Gandhi, J. (2020). A comprehensive survey on mobile edge computing: Challenges, tools, applications. pages 49–55.

[61] Vijitkunsawat, W. and Chantngarm, P. (2020). Study of the performance of machine learning algorithms for face mask detection. pages 39–43.