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

UNIVERSITY ECHAHID HAMMA LAKHDAR - EL OUED
FACULTY OF EXACT SCIENCES
Computer Science department

THESIS
Presented with a view to obtaining :

ACADEMIC MASTER

Domain: **Mathematics and Computer Science**
Industry: **Computer Science**
Specialty: **Distributed Systems and Artificial Intelligence**

Theme

Detection of Dangerous Driving Behaviors

Presented by:
Sebouai Imane
Bennour Mohammed Ramzi

Chairperson: [Name]
Examiner: [Name]
Supervisor: Khelaifa Abdennacer

Academic year
2022/2023

Dedication

To my beloved parents, supportive siblings, and dear friends,

On the completion of my Master's thesis in Computer Science,

Specialty: Distributed Systems and Artificial Intelligence, I want to

express my heartfelt gratitude and dedicate this achievement to each one of you.

Your unwavering love, encouragement, and belief in me have been the driving force behind my success.

To my parents, your sacrifices and endless support have shaped me into who I am today.

Thank you for instilling in me the values of hard work and perseverance.

To my siblings, your constant motivation and camaraderie have been invaluable.

With heartfelt appreciation,

❖ ***Sebouai Imane***

Abstract

This graduation thesis, "Detection of Dangerous Driving Behaviors," addresses the critical challenge of enhancing road safety by identifying and mitigating risky driving conduct. With an introduction to the escalating issue of unsafe driving behaviors leading to accidents and fatalities, this work delves into the existing approaches employed for tackling this problem.

Various solutions have been proposed in recent years, including traditional computer vision techniques and machine learning algorithms for behavior recognition. However, these solutions often fall short in terms of accuracy and robustness.

The proposed solution involves developing an advanced system utilizing computer vision and deep learning algorithms, particularly Convolutional Neural Networks (CNNs). Training this model on diverse datasets containing images captured by in-car cameras can recognize patterns associated with hazardous driving actions, such as distracted driving, phone usage, drunk driving, and more.

The results obtained through extensive testing and evaluation of the system are promising, with an accuracy rate of 91.0368%, a precision score of 91.4692%, and an impressive F1 score of 90.8515%. These outcomes underline the effectiveness of the proposed system in detecting and categorizing dangerous driving behaviors.

The successful implementation of this solution offers a considerable leap forward in road safety, demonstrating the potential to reduce the number of accidents and save lives significantly. Additionally, the work paves the way for further research and advancements in the field of intelligent transportation systems, contributing to safer and more secure roadways.

Keywords: dangerous driving behavior, computer vision, deep learning, video analysis, driver assistance systems, road safety.

Résumé

La thèse "Détection des comportements de conduite dangereux" aborde le défi critique d'améliorer la sécurité routière en identifiant et en atténuant les comportements de conduite risqués, compte tenu de la préoccupation croissante concernant les comportements de conduite non sécuritaires entraînant des accidents et des décès. Elle examine les approches existantes dans ce contexte, notamment les techniques traditionnelles de vision par ordinateur et les algorithmes d'apprentissage automatique. Cependant, ces méthodes manquent souvent de précision et de robustesse.

La solution proposée implique un système avancé utilisant la vision par ordinateur et les réseaux de neurones convolutionnels (CNN) pour analyser des ensembles de données variés contenant des images de caméras embarquées. Elle vise à reconnaître des actions de conduite dangereuses, telles que la conduite distraite et la conduite en état d'ivresse.

Les résultats des tests approfondis sont prometteurs, avec un taux de précision de 91,0368 %, un score de précision de 91,4692 % et un impressionnant score F1 de 90,8515 %. Ces résultats mettent en lumière l'efficacité du système dans la détection et la catégorisation des comportements de conduite dangereux.

La mise en œuvre de cette solution représente un progrès significatif en matière de sécurité routière, avec le potentiel de réduire de manière significative les accidents et de sauver des vies.

Mots-clés : **comportement de conduite dangereux, vision par ordinateur, apprentissage profond, analyse vidéo, systèmes d'assistance au conducteur, sécurité routière.**

الملخص

في مذكرة التخرج هذه بعنوان "اكتشاف سلوكيات القيادة الخطرة" تتناول التحدي الحرج لتعزيز سلامة الطرق من خلال تحديد وتخفيف السلوكيات القيادية الخطرة. بعد تقديم مقدمة حول تصاعد قضية السلوكيات القيادية غير الآمنة التي تؤدي إلى حوادث ووفيات، يستقصي هذا العمل النهج الحالية المستخدمة لمعالجة هذه المشكلة.

تم اقتراح حلاً متقدمًا يشمل تطوير نظام يستخدم الرؤية الحاسوبية وألگوريتمات التعلم العميق، وبشكل خاص شبكات العصب الاصطناعي بالتبادل. تدريب هذا النموذج على مجموعات بيانات متنوعة تحتوي على صور التقطت بواسطة كاميرات السيارات يمكنه التعرف على الأنماط المرتبطة بأفعال القيادة الخطرة، مثل القيادة المشتتة، واستخدام الهاتف أثناء القيادة، والقيادة في حالة السكر، وغيرها.

النتائج التي تم الحصول عليها من خلال الاختبار والتقييم الشامل للنظام واعدة، حيث بلغ معدل الدقة 91.0368٪، ونقطة الدقة 91.4692٪، ونقطة F1 المثيرة 90.8515٪. تؤكد هذه النتائج فعالية النظام المقترح في اكتشاف وتصنيف السلوكيات القيادية الخطرة.

تنفيذ هذا الحلا يمثل خطوة كبيرة في مجال سلامة الطرق، حيث يظهر الإمكانية بشكل كبير للحد من عدد الحوادث وإنقاذ الأرواح. بالإضافة إلى ذلك، يسهم هذا العمل في فتح الباب أمام بحوث وتطورات إضافية في ميدان أنظمة النقل الذكية، مما يساهم في جعل الطرق أكثر أمانًا وأكثر حماية.

الكلمات الرئيسية: **سلوك القيادة الخطرة، رؤية الحاسوب، التعلم العميق، تحليل الفيديو، أنظمة مساعدة السائق، سلامة الطرق**.

Contents

List of Figures

Notation And Abbreviated Terms

IoT : Internet of Things
ADAS : Advanced Driver Assistance Systems
IaaS : Infrastructure as a Service
PaaS : Platform as a Service
SaaS : Software as a Service
CRM : customer relationship management
EC2: Elastic Compute Cloud
VPC : virtual private cloud
AWS : Amazon Web Services
ML : Machine Learning
AI : Artificial Intelligence
VS Code : Visual Studio Code
GCP : Google Cloud Platform
RAN : Radio Access Network
vRAN : virtua Radio Access Network
CD : Continuous Deployment
CI : Continuous Integration
NGCO : Next Generation Central Office
MEC : Mobile Edge Computing
MAEC : Multi-Access Edge Computing
EaaS : Edge as a Service
FBI : Federal Bureau of Investigatio
CNN : Convolutional Neural Network
RNN : Recurrent neural network
SSD : Single Shot MultiBox Detector
OCR : Optical Character Recognition
AR : Augmented Reality
SVM : Support Vector Machine
LSTM : Long Short-Term Memory Net- work
DTW : dynamic time warping algorithm
LCS : longest common sub-sequence algorithm
ADBD Net : Abnormal Driving Behavior Detection
ReLU : Rectified Linear Units
IDE : Itegrated Development Environment
SDK : software development kit
TFLite) : TensorFlow Lite

General Introduction

Every year, 1.3 million global deaths result from traffic accidents, ranking them as the eighth leading cause of death; additionally, 20-50 million people sustain injuries or disabilities. Alarmingly, India tops the list for road fatalities, with a continual rise since 2006, reaching 1.46 lakh deaths in 2015. These incidents are predominantly caused by driver errors, with increasing cases of distracted driving contributing to accidents. Distracted driving encompasses activities diverting attention and is categorized as manual, visual, or cognitive. Cognitive distraction, where the driver's mind wanders, poses risks even when physically driving safely. Visual distraction refers to taking eyes off the road, while manual distractions involve hands off the wheel for tasks like texting or eating. Advanced Driver Assistance Systems (ADAS) aim to enhance safety by alerting drivers to potential issues, yet current autonomous vehicles still demand driver readiness in emergencies.

We are convinced that the identification of distracted drivers holds paramount significance in advancing preventative actions. By enabling vehicles to recognize such diversions and subsequently alert drivers, a notable reduction in road accidents could be achieved. In this endeavor, our project centers on identifying manual distractions, wherein drivers engage in tasks other than safe driving, and aims to pinpoint the root causes of these distractions. Our solution employs a Convolutional Neural Network methodology, aiming to balance accuracy, computational efficiency, and memory usage, critical for real-time applications.

The theme titled **"Detection of Dangerous Driving Behavior"** is divided into four chapters, each addressing different aspects of the study.

The first chapter provides a general introduction, delving into topics like cloud computing, advanced computation, and the Internet of Things (IoT). It highlights how these technologies intersect and contribute to our research focus.

The second chapter delves into computer vision and explores key techniques of machine learning and deep learning that are pivotal in our study. These methods are crucial in analyzing and interpreting driving behavior data effectively.

The third chapter is dedicated to the theoretical foundation of our proposed system for resolving the problem. It explains in detail the methodology we have devised to address the issue of detecting dangerous driving behavior. This theoretical framework serves as the basis for our practical implementation.

The fourth chapter, the applied section, is dedicated to detailing our practical implementation and discussing the results we have obtained. This section provides insights into how our proposed system performs in real-world scenarios, showcasing its effectiveness in detecting and mitigating dangerous driving behaviors.

Chapter 1

Cloud Computing & Internet of Things

1.1 Introduction

Cloud computing, edge computing, and the Internet of Things (IoT) are three interconnected technologies that have revolutionized the way we store, process, and analyze data. In this chapter, we will explore the meaning of these technologies and examine them along with their various types and applications.

Cloud computing refers to the practice of using remote servers, accessed over the internet, to store and manage data, run applications, and deliver services. It offers scalability, flexibility, and cost-efficiency, enabling businesses to focus on their core competencies.

On the other hand, edge computing brings computation and data storage closer to the source of data generation. It reduces latency, enhances real-time processing capabilities, and enables faster decision-making by processing data at or near the edge devices.

The Internet of Things (IoT) is a network of interconnected devices embedded with sensors, software, and connectivity, enabling them to collect and exchange data. It enables seamless device communication, improving efficiency, automation, and data-driven insights.

Together, these technologies have transformed industries, enabling smarter devices, efficient data processing, and enhanced user experiences. In the following chapters, we will delve deeper into the concepts, architectures, and applications of cloud computing, edge computing, and the Internet of Things.

1.2 Cloud Computing

Cloud computing refers to delivering computing resources, such as storage, processing power, and software applications, over the internet. It allows users to access these resources on-demand without needing local infrastructure or hardware. The concept of cloud computing has gained significant popularity in recent years due to its scalability, flexibility, and cost-effectiveness.[1]

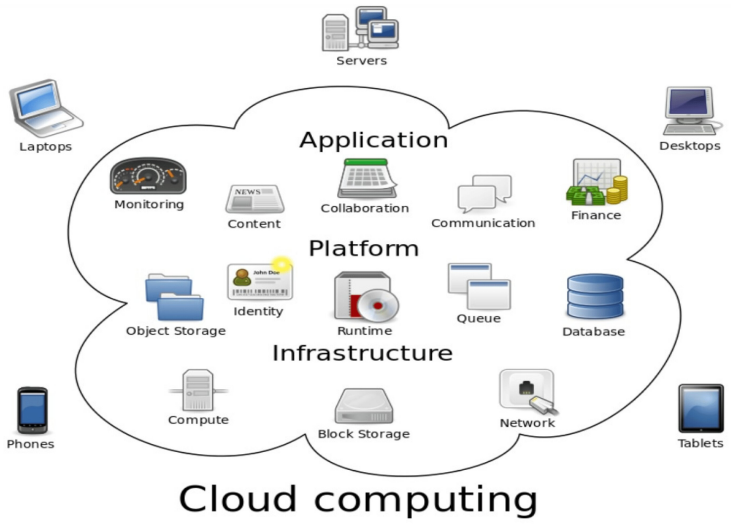


Figure 1.1: Schematic view of Cloud Computing

In comparison to traditional on-site computing, and depending on the cloud services you select, cloud computing allows you to perform the following operations:

Cost reduction : Cloud computing enables you to offload a portion or most of the costs and efforts associated with purchasing, installing, configuring, and managing your own on-site infrastructure.

Improved agility and time to market : With the cloud, your organization can use enterprise applications within minutes instead of waiting for weeks or months for the IT department to respond to a request, purchase and configure supporting hardware, and install the software. The cloud also empowers certain users, particularly developers, and data scientists, to self-serve with software and infrastructure support.

Easy and cost-effective scalability : The cloud offers elasticity. Instead of buying excess capacity that remains unused during low-traffic periods, you can increase or decrease capacity in response to traffic spikes and drops. You can also leverage your cloud provider's global network to distribute your applications closer to users worldwide. The term "cloud computing" also refers to the technology that powers the cloud. This includes some form of virtualized IT infrastructure servers, operating system software, networking, and other infrastructure that is abstracted using specialized software so that it can be pooled and divided independently of physical hardware limitations. For example, a single physical server can be divided into multiple virtual servers[2].

1.2.1 Cloud Service Models

Cloud computing offers various service models, each catering to different user needs: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) in Figure 1.2.

Infrastructure as a Service (IaaS):

IaaS provides virtualized computing resources, such as virtual machines, storage, and networks, over the Internet. Users control the operating systems, applications, and configurations, while the service provider manages the underlying infrastructure.

Platform as a Service (PaaS):

PaaS offers a platform for developing, testing, and deploying applications. It provides a complete development environment, including operating systems, programming languages, databases, and web servers. Users can focus on application development without worrying about infrastructure management.

Software as a Service (SaaS):

SaaS delivers software applications over the internet on a subscription basis. Users can access and use the software through a web browser without the need for installation or maintenance. Examples of SaaS include email services, customer relationship management (CRM) systems, and productivity tools[3].

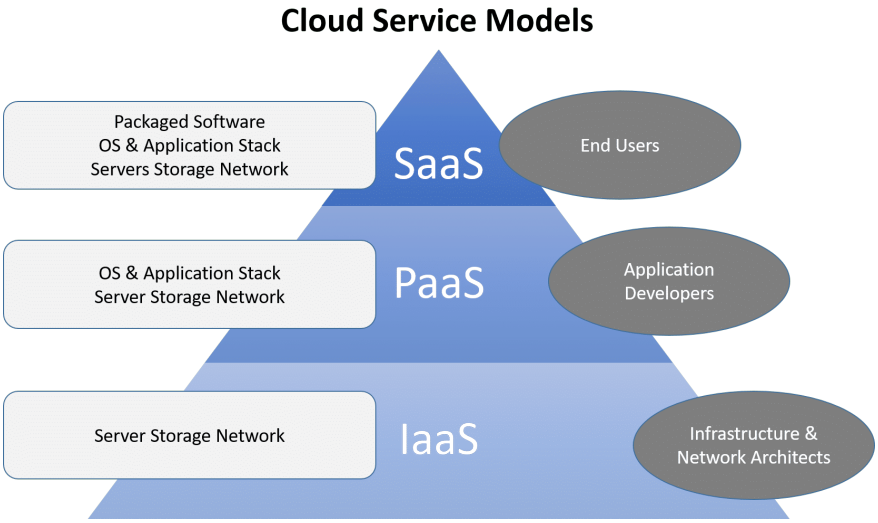


Figure 1.2: Cloud Computing Service Models diagram

1.2.2 Deployment models of Cloud Computing:

The four deployment models of cloud computing share many similarities, but they differ in terms of their scope and the level of access granted to cloud users. Figure 3 provides a visual representation of these cloud deployment models and their respective features. Let me briefly explain each of these models to you [4].

Public cloud/external cloud: The cloud enables users to access cloud environments openly. The public cloud, which is located off-premise, allows enterprises to deliver services to users through third-party providers. Some examples of public cloud platforms include Sun Cloud, Google AppEngine, IBM's Blue Cloud, Amazon Elastic Compute Cloud (EC2), and Windows Azure Service Platform.

Private cloud/internal cloud: This Cloud mentioned here refers to an on-premise cloud that organizations utilize to have extensive control over their cloud services and infrastructure. This type of cloud is owned and managed by the organization itself. Its primary purpose is to offer services within the organization while ensuring security and privacy. Some examples of such on-premise cloud providers are Seagate and RedHat.

Hybrid cloud/virtual private cloud (VPC): This cloud combines public and private cloud models. It means that the cloud computing ecosystem is hosted and managed by a third party, which is located off-premise. However, within this cloud, an organization has the ability to use certain dedicated resources privately. To illustrate this concept, we can look at examples such as Cybercon.com (US Microsoft Hybrid Cloud) and Bluemix.net (IBM Cloud App Development). These platforms demonstrate the implementation of this combined cloud model, where organizations can leverage both public and private resources based on their specific needs.

Community cloud: This cloud allows the cloud computing environment can be shared or managed by a number of related organizations. Example: sourcing focus.

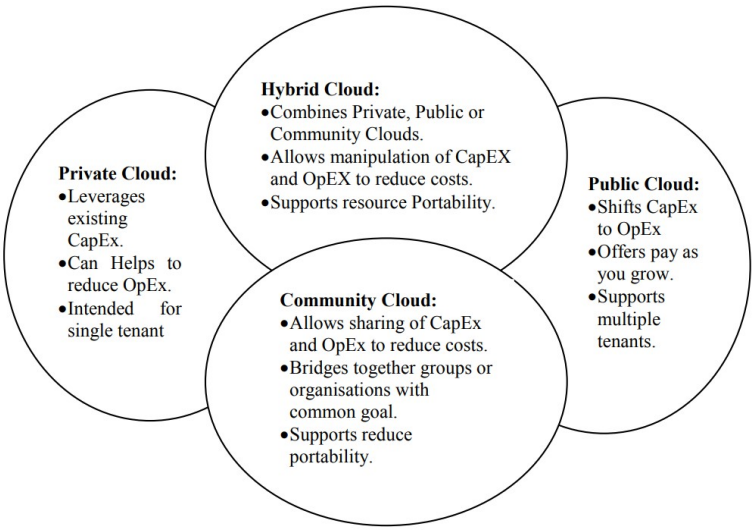


Figure 1.3: Cloud Computing Service Models diagram

1.2.3 Cloud Computing Platforms:

Several cloud computing platforms have gained popularity due to their robust features and extensive service offerings. Some of the main platforms include[5]:

Amazon Web Services (AWS): AWS is a comprehensive cloud platform offering various services, including computing power, storage, databases, machine learning, and analytics. It is known for its scalability, reliability, and global infrastructure.

Microsoft Azure: Azure provides a vast array of cloud services, including virtual machines, databases, AI capabilities, and IoT solutions. It offers seamless integration with existing Microsoft tools and technologies, making it a popular business choice.

Google Cloud Platform (GCP): GCP offers a suite of cloud services, including computing, storage, networking, and machine learning. It emphasizes data analytics and AI capabilities, making it suitable for organizations with advanced data processing needs Figure 1.4.

1.2.4 What is Mobile Cloud Computing?

Mobile cloud computing combines the power of cloud computing with mobile devices, enabling users to access cloud resources and services on their smartphones and tablets. It offloads computational tasks and storage to the cloud, reducing the burden on mobile devices and extending their capabilities[6].

1.2.5 Cloud Computing Security:

Traditionally, security issues have been the main obstacle for organizations considering cloud services, especially public ones. However, in response to the demand, the security provided by cloud service providers consistently surpasses on-premises security solutions. Maintaining cloud security requires different procedures and skills from those in legacy IT environments. Some good security practices in the cloud include the following[7]:

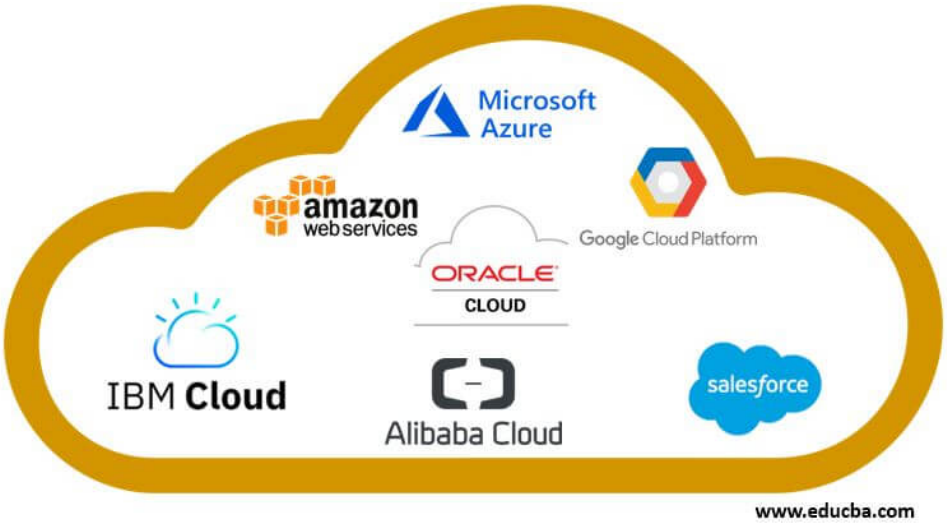


Figure 1.4: Cloud Computing Platforms

Data encryption : Encrypting sensitive data both in transit and at rest helps protect it from unauthorized access.

Access control : Implementing strong access controls, such as multi-factor authentication and role-based access control, ensures that only authorized individuals can access the cloud resources.

Regular security updates : Keeping the cloud infrastructure and software updated with the latest security patches helps address vulnerabilities and protect against known threats.

Monitoring and logging : Implementing robust monitoring and logging mechanisms allows for the detection and investigation of any suspicious activities or security breaches.

Backup and disaster recovery : Regularly backing up data and having a well-defined disaster recovery plan in place helps mitigate the impact of potential security incidents or data loss.

Security awareness training : Educating employees about security best practices and potential threats helps create a security-conscious culture within the organization.

1.2.6 Cloud Computing for Detection of Dangerous Driving Behaviors

Cloud computing can enable the detection of dangerous driving behaviors through its ability to process and analyze large amounts of data in real time. Vehicles equipped with cameras and sensors continuously transmit data to the cloud where machine learning algorithms can identify patterns that indicate aggressive acceleration, speeding, tailgating, illegal lane changes, and other risky maneuvers. By processing this data in the cloud rather than locally in the vehicle, more sophisticated analysis can be performed to accurately detect dangerous behaviors and even predict collision risks before they occur. This allows for alerts to be sent to drivers or vehicle controls to be activated to prevent accidents.

1.3 Edge Computing

In the rapidly changing world of technology, the emergence of edge computing has brought about a significant shift in how we handle and control data. The conventional approach of relying solely on centralized cloud architectures faces obstacles such as delays, limited bandwidth, and privacy concerns. However, edge computing offers a promising solution by bringing computation closer to where the data is generated. This approach enables quicker response times, alleviates network congestion, and enhances privacy. The review article explores the core principles, distinguishing factors, various types, and application areas for edge computing.

1.3.1 Definition

Edge computing is a method of processing and analyzing data in a decentralized manner closer to where the data is generated rather than relying on a centralized cloud environment located remotely. This decentralized approach reduces the distance data needs to travel, which helps to reduce latency and bandwidth problems. It utilizes devices like sensors, gateways, and routers at the network's edge to perform computations locally, enabling faster decision-making in real time. In essence, edge computing expands the traditional cloud infrastructure by creating a network of interconnected devices that work together in a distributed ecosystem[8] Figure 1.5.

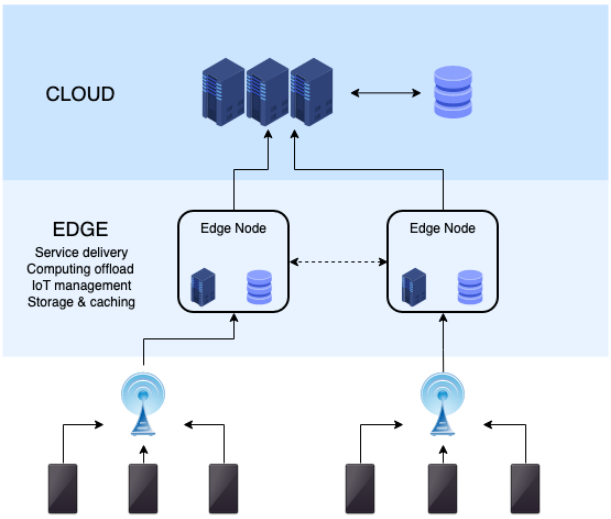


Figure 1.5: Architecture of Edge Computing

1.3.2 Edge Computing vs. Fog Computing

Although edge computing and fog computing have similar goals of bringing computation closer to data sources, they have distinct scopes and architectures. Edge computing primarily concentrates on processing data at the immediate edge of the network, such as smartphones or IoT devices. In contrast, fog computing expands the edge concept

to a larger scale, incorporating a network of interconnected edge devices and gateways. By establishing a hierarchical structure, fog computing enables more intricate processing tasks while still maintaining proximity to data sources[9] Figure 1.6.

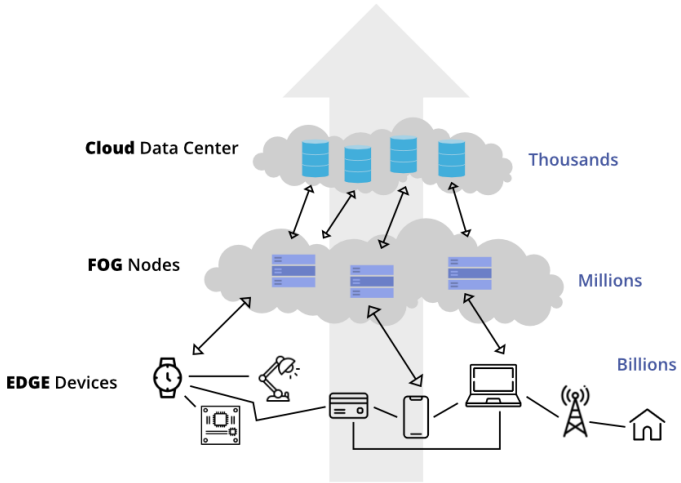


Figure 1.6: Edge Computing vs. Fog Computing

1.3.3 Types of Edge Computing

One easy way to understand various types of edge computing is based on its proximity to the end device and round-trip latency to the core of the data center. Latency being the primary factor, Edge computing could be broadly categorized into the following:

Internet of Things Edge: In most cases, the expected delay at this Edge is typically under 1 millisecond. This applies to a wide range of devices that are connected to either private or public networks, including the Internet. These devices can vary from smart devices like mobile phones, which have the ability to process data, to simpler sensors that transmit information about their environment. Some examples of such devices include retail kiosks, cameras, sensors used in factories, connected cars, drones, streetlamps with connectivity, smart parking meters, and even remote surgery equipment. It's important to note that this list is not exhaustive, as there are many other devices that fall into this category.

On Premise Edge: The typical latency expectations at this Edge are generally below 5 milliseconds. To effectively handle the data generated by multiple devices at the IoT Edge, it is necessary to have a method of collecting, storing, processing, analyzing, and responding to relevant requests using this data. On-premise Edge solutions offer computational resources within an enterprise's premises, enabling localized data processing and reducing the time required for data processing. In this scenario, the devices at the Edge typically connect to either the Network Edge or a data center for additional requests. Industries such as large enterprises, industrial manufacturing facilities, and large retail operations greatly benefit from these types of deployments. They gain the ability to process data in close proximity to its source while maintaining ownership of the necessary

hardware for data processing.

Access Edge: The typical latency expectation within this Edge environment generally ranges from 10 to 40 milliseconds.

The traditional Radio Access Network (RAN), which was previously a fixed-function device, has undergone a transformation, being disaggregated into a collection of virtual functions that run on software utilizing commercially available servers. RAN plays a pivotal role in bridging wireless devices and the core network of the telecommunications operator.

Notions such as virtual RAN (vRAN) and industry-driven initiatives like Open-RAN and O-RAN have facilitated the development of interfaces to manage these virtualized deployments, much like the management of any other edge device. The shift towards the cloud-native instantiation of RAN functions simplifies the life cycle management of various RAN deployments. This is achieved by applying Continuous Integration (CI) and Continuous Deployment (CD) frameworks, making the most of DevOps models.

Network Edge: The anticipated latency at this particular Edge location also tends to fall within the range of 10 to 40 milliseconds.

Information originating from diverse IoT edges, encompassing the On-premises edge and Access edge, tailored to the specific geographic area, must be consolidated prior to establishing a connection with the centralized data center, which may span across extensive regions[10].

Such configurations could be broadly labeled as the "Network Edge" or a proximity edge in relation to the core data center operated by the service provider.

The architectural approach known as Next Generation Central Office (NGCO) is designed to meet the demands of the Network Edge. Another illustration of this type of Edge environment is the Wireline Fixed Access Edge, which offers broadband services like IPTV, VoIP, and internet services Figure 1.7.

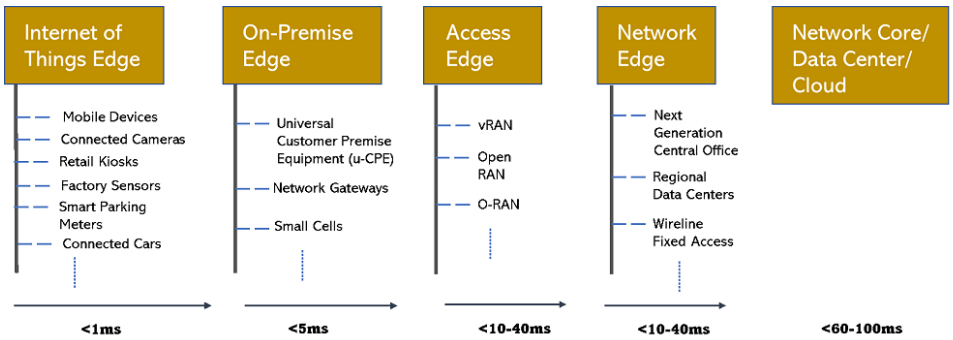


Figure 1.7: Types of Edge Computing

1.3.4 Edge computing models

Edge computing includes various models that address different situations and needs. These models determine how computational tasks are distributed and processed within the edge ecosystem. Let's explore some of the notable edge computing models:

Pure Edge Model: In this approach, edge devices like sensors, gateways, and edge servers carry out computational tasks directly. These devices process data locally and make independent decisions, eliminating the dependence on centralized cloud resources. This strategy aims to minimize latency and reduce the necessity for continuous communication with a remote server.

Fog Model: The fog model extends the concept of edge computing to a larger scale. It involves forming clusters of interconnected edge devices and gateways collaborating to process data. This enables more complex computations and analytics closer to the data source, maintaining low latency while accommodating more intensive tasks[11].

Cloud-Fog Model: This model combines the strengths of both cloud computing and fog computing. It involves a hierarchy of computational resources, where some processing is done at the edge, and more resource-intensive tasks are offloaded to cloud servers. This hybrid approach optimizes the utilization of resources and enhances scalability.

Mobile Edge Computing (MEC): MEC leverages the infrastructure of mobile networks for edge computing. Mobile base stations and other network elements are used to host edge computing resources. This model is particularly useful for scenarios that involve fast-moving devices, such as connected vehicles, where low-latency communication and decision-making are crucial.

Multi-Access Edge Computing (MAEC): MEC extends the concept of mobile edge computing to support various access technologies, such as 5G, Wi-Fi, and Ethernet. It aims to provide a unified edge computing platform that serves a diverse range of devices and applications, regardless of the access technology they use[12].

Serverless Edge Computing: This model abstracts the underlying infrastructure, allowing developers to deploy and run functions at the edge without the need to manage the hardware or infrastructure. It simplifies development and deployment processes while ensuring efficient resource utilization.

Edge as a Service (EaaS): Similar to cloud-based service models, EaaS offers edge computing resources on a pay-as-you-go basis. It allows organizations to access and utilize edge resources without needing upfront infrastructure investment. This model enhances flexibility and scalability for edge-based applications[13].

1.3.5 Domains

Edge computing finds applications across numerous domains, revolutionizing industries and enhancing user experiences. In manufacturing, real-time monitoring and predictive maintenance are enabled through edge-powered analytics. Healthcare benefits from remote patient monitoring and rapid diagnostic tools. Smart cities utilize edge computing for efficient traffic management, waste management, and energy optimization. The entertainment sector leverages edge resources for low-latency content delivery in gaming and live streaming. Moreover, autonomous vehicles rely on edge computing for split-second decision-making, ensuring road safety.

Despite its immense potential, edge computing faces challenges such as security concerns, resource constraints, and the need for standardized architectures. Ensuring data privacy and securing distributed edge devices remains a critical priority. As the field matures, standardization efforts will be pivotal in creating interoperable and scalable edge

systems. Future directions include advancements in edge AI, seamless integration with 5G networks, and the exploration of novel edge-to-cloud synergies[14].

1.4 Internet of Things

In our modern society, nearly everything is interconnected with the internet. The Internet of Things (IoT) is a straightforward idea that involves linking all the objects in our surroundings to the Internet.

1.4.1 Definition

The Internet of Things (IoT) refers to the network of physical objects, devices, vehicles, buildings, and other items that are embedded with sensors, software, and connectivity, enabling them to collect and exchange data. These interconnected devices communicate with each other and with the internet, creating a vast ecosystem of smart and interconnected systems. The IoT has revolutionized how we interact with technology and the world around us. It can potentially transform various industries, including healthcare, transportation, agriculture, manufacturing, and more. By connecting everyday objects to the internet, the IoT enables us to gather valuable data, automate processes, and make informed decisions.[15] In the IoT ecosystem, devices can be remotely monitored, controlled, and managed, increasing efficiency, productivity, and convenience. For example, smart homes equipped with IoT devices allow homeowners to control their lights, thermostats, and security systems from their smartphones. In healthcare, IoT devices can monitor patients' vital signs and send real-time data to healthcare providers, enabling timely interventions and personalized care Figure 1.8.

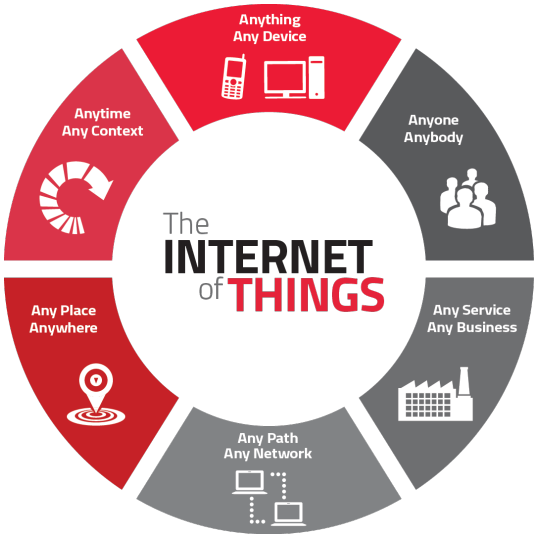


Figure 1.8: Definition of Internet of things

1.4.2 Why IoT Is Important

The various entities connected to the Internet can be classified into three distinct categories: Devices that gather information and transmit it. Devices that receive information and take action based on it. Devices that perform both functions simultaneously. To enhance understanding, let's explore some real-life illustrations[16].

1.4.3 Collecting Information

These devices utilize sensors, typically, to gather data. For instance, a thermostat in your smart home can detect the temperature in your house and display it on a smartphone app. By combining sensors with an internet connection, we can automatically gather data and leverage sophisticated computer algorithms to make intelligent decisions. This process eliminates the need for direct interaction with anything other than an internet-connected device[17].

1.4.4 The architecture of the Internet of Things

Typically consists of several layers that work together to enable the seamless integration of devices, networks, and applications. Here's a high-level overview of the IoT architecture[16]:

Perception Layer: This layer includes the physical devices or "things" that collect data from the environment. These devices can be sensors, actuators, wearables, or any other IoT-enabled device. They capture data such as temperature, humidity, motion, or any other relevant information.

Network Layer: The network layer facilitates communication between the devices in the IoT ecosystem. It includes various connectivity options such as Wi-Fi, Bluetooth, Zigbee, cellular networks, or even satellite communication. This layer ensures that the data collected by the devices can be transmitted reliably and securely.

Middleware Layer: The middleware layer acts as a bridge between the perception layer and the application layer. It provides services such as data filtering, protocol translation, device management, and security. This layer helps aggregate and process the data from multiple devices before sending it to the application layer.

Application Layer: The application layer is where the data collected from the IoT devices is processed, analyzed, and utilized to derive meaningful insights. It includes various applications, platforms, and services that enable users to interact with the IoT system. This layer can involve cloud-based applications, edge computing, or a combination of both.

Business Layer: The business layer focuses on utilizing the insights gained from the IoT data to drive business value. It involves decision-making processes, business intelligence, and integration with existing enterprise systems. This layer enables organizations to make informed decisions, optimize operations, and create new business models based on IoT data Figure 1.9.

It's important to note that the IoT architecture can vary depending on the specific use case, industry, and implementation. Different IoT platforms and frameworks may have their own variations of architecture, but the fundamental principles remain consistent[18].

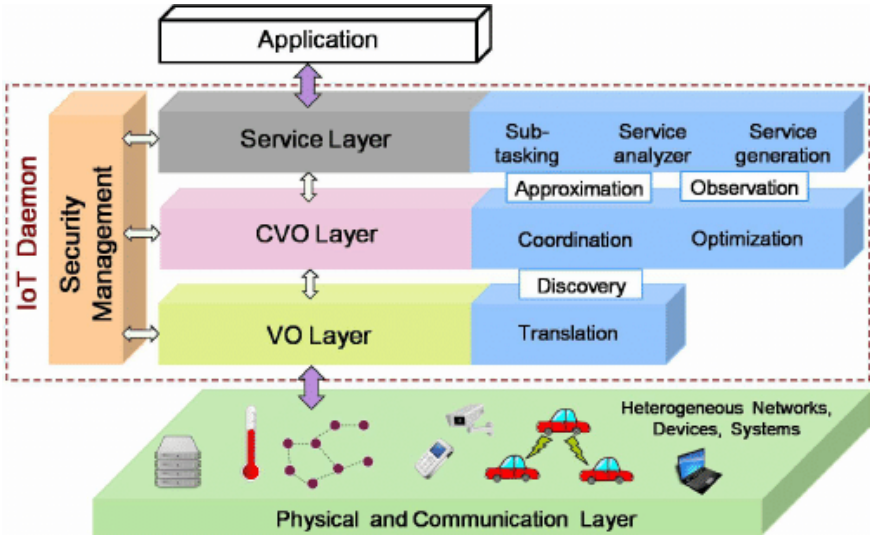


Figure 1.9: A layered architecture for the Internet of things

1.4.5 Internet of Things Benefits

The emerging technology of IoT brings numerous advantages, such as[19]:

- Increasing the effectiveness and efficiency of business operations with minimal investment.
- Developing innovative revenue streams and business models.
- Streamlining soil data analysis to optimize production.
- Enabling user-friendly technology for seamless automation and control.
- Extracting valuable insights from IoT data.
- Automating manual tasks to save time and resources.
- Boosting productivity and enhancing the overall quality of life.
- Seamlessly bridging the physical and digital realms.
- Enhancing patient data analysis to empower doctors in making informed decisions.

1.4.6 Security

As you may be aware, the increasing connectivity of devices to the internet has raised concerns about security and privacy. It is true that virtually anything connected to the internet can be vulnerable to hacking, including IoT products. This means that cyberattacks can potentially disrupt or compromise various aspects of our lives. One major concern is the vast amount of data being generated and stored by these connected devices. Suppose all the data collected by each device is constantly stored without proper

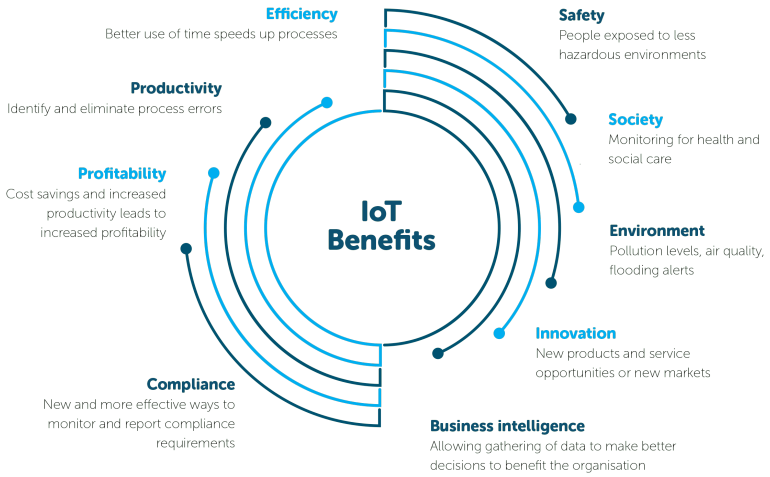


Figure 1.10: Internet of Things Benefits

security measures. In that case, there is a risk that unauthorized individuals could access this data and potentially misuse it for their own purposes. Furthermore, the issue of personal privacy arises when every device in our homes is connected to the internet. Hackers could potentially access and expose sensitive information about our habits and lifestyles. For example, a smart fridge could reveal our eating habits and schedules, while a smartwatch could expose our heart rate and daily activities. With the proliferation of sensors and cameras in various devices, hackers could potentially invade every aspect of our lives. These concerns are indeed serious, and the industry is actively working on addressing them. Various security methods and measures have been developed, and ongoing efforts are being made to improve the security of IoT devices[20]. To illustrate:

In September 2015: The Federal Bureau of Investigation (FBI) issued a public service announcement highlighting the vulnerabilities of IoT devices and providing recommendations for consumer protection and defense.

In August 2017: Congress introduced the IoT Cybersecurity Improvement Act, which aims to establish security standards for IoT devices sold to the U.S. government. This act would require devices to have stronger security measures, such as not using default passwords, addressing known vulnerabilities, and providing mechanisms for regular software updates.

While these initiatives are steps in the right direction, it is important to acknowledge that there is no foolproof solution yet. The software engineering community continues to actively explore and develop new security measures to mitigate the risks associated with IoT devices[21].

1.4.7 Internet of Things for Detection of Dangerous Driving Behaviors

The Internet of Things can enable real-time detection and prevention of dangerous driving behaviors. Vehicles embedded with networked sensors can continuously transmit data like

speed, acceleration, and lane position to the cloud. Machine learning algorithms analyze this data to identify aggressive driving patterns like speeding, hard braking, and swerving. The connectivity of the Internet of Things allows dangerous behaviors to be detected as they occur and alerts to be sent to drivers through in-vehicle displays or mobile apps. Additionally, active safety controls could intervene and automatically apply brakes or stabilize the vehicle to prevent collisions. This IoT connectivity promises to significantly reduce accidents caused by hazardous driving behaviors.

1.5 Analyzing problems and designing models

While driving, there are various abnormal behaviors that can lead to driving errors. A specific factor does not solely cause these behaviors. Research on abnormal behaviors often indicates a disconnected relationship between the cause (such as fatigue or distraction) and the actual reality, resulting in an incomplete analysis of abnormal behavior. This article proposes a comprehensive standard for analyzing drivers' abnormal behavior. It considers the driver as the focal point, considering their natural physiological behavior, potential non-natural autonomous behavior, and external disturbances. Stressful behaviors are categorized into natural, non-natural, and passive behavioral factors. Drivers exhibit key abnormal behaviors during driving, which are classified accordingly. For instance, behaviors like prolonged eye closure, frequent yawning, and frequent head nodding are classified as natural behavioral factors. On the other hand, behaviors like smoking, making phone calls, and drinking water are classified as non-natural behavioral factors. Lastly, behaviors that interfere with driving for other reasons are classified as passive behavioral factors.

1.6 Conclusion

In this introductory chapter, we journeyed through the intertwined realms of cloud computing, edge computing, and the Internet of Things (IoT). The convergence of these technologies has ushered in a new era of data processing, enabling seamless connections, real-time insights, and transformative possibilities. As we explored the synergies between these domains, we unveiled a significant challenge that demands attention: developing a robust and efficient system that seamlessly integrates cloud and edge resources to harness the potential of IoT-generated data. This imperative calls for innovative solutions to optimize data distribution, mitigate latency, and maximize the utility of distributed computing environments.

Chapter 2

Machine learning, deep learning and computer vision

2.1 Introduction

In the second chapter of this thesis, we delve into the fascinating world of machine learning, deep learning, computer vision, and their relationship to detecting dangerous driving behavior. Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and models that enable computers to learn and make predictions or decisions without being explicitly programmed. Deep learning, on the other hand, is a subset of machine learning that utilizes artificial neural networks to process and analyze complex data. Computer vision, a branch of artificial intelligence, enables computers to understand and interpret visual information from images or videos.

2.2 Machine learning (ML)

Machine learning is a fascinating field that empowers computers to learn and make predictions without being explicitly programmed. It is a subset of artificial intelligence that focuses on developing algorithms and models to analyze and interpret vast amounts of data to identify patterns and make informed decisions. Machine learning has revolutionized various industries, including healthcare, finance, and technology, by enabling automated processes, personalized recommendations, and predictive analytics. With its ability to continuously learn and adapt, machine learning holds immense potential for solving complex problems and driving innovation in the digital age[22] Figure 2.1.

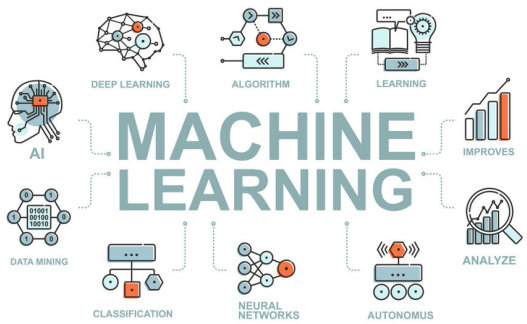


Figure 2.1: Supervised Learning

2.3 Fundamentals of Machine Learning

Machine Learning is a fundamental concept in the field of Artificial Intelligence that focuses on developing algorithms and models that enable computers to learn and make predictions or decisions without being explicitly programmed. It involves the study of statistical models and algorithms that allow systems to learn and improve from experience automatically. The fundamentals of Machine Learning include understanding data preprocessing, feature engineering, model selection, and evaluation. It also involves techniques such as supervised learning, unsupervised learning, and reinforcement learning.

Machine Learning is crucial in various applications, including image recognition, natural language processing, and recommendation systems[23].

2.3.1 Supervised Learning

Supervised machine learning refers to the type of problem in which each record in the data set contains a label or a flag. The final column, asperity, is the label. Given temperature and vibration data, we want to predict asperity. This is a labeled data set. Using this data set that includes the label, we can train an algorithm to predict the future of unlabeled data. You fit that into your algorithm, which now predicts a label for this data. This is referred to as supervised learning[24]. Regression and classification are the two types of supervised learning Figure 2.2.

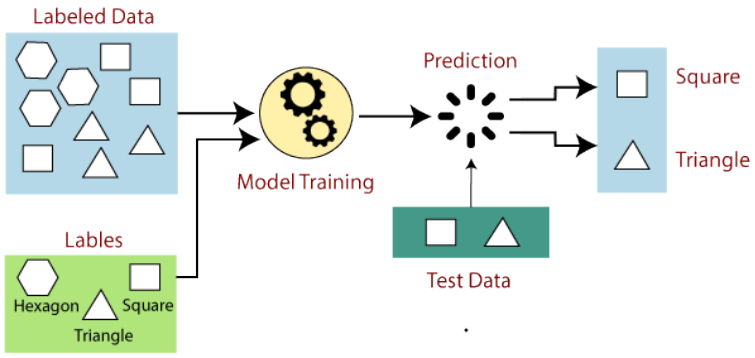


Figure 2.2: Supervised Learning

1. **Regression:** Regression algorithms are used to make predictions of continuous values by analyzing input variables. The main objective of regression problems is to determine a mapping function between the input and output variables. If you have a target variable representing a quantity, such as income, scores, height, weight, or the probability of a binary category, then a regression model would be appropriate to use[25] Figure 2.3.

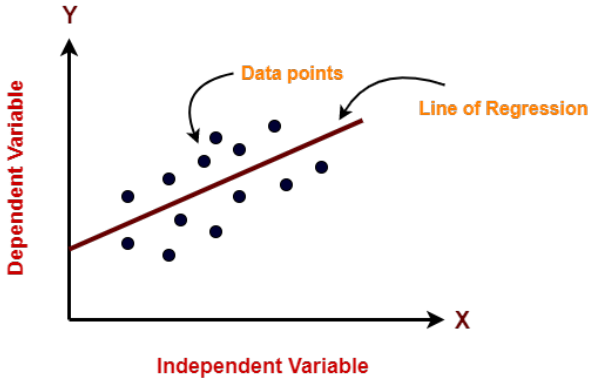


Figure 2.3: Regression Algorithm

2. **Classification:**

   Classification is a type of prediction model that utilizes input data to estimate a mapping function. This mapping function determines the discrete output variables, such as labels or categories, based on the given input data. The goal of a classification algorithm is to predict the label or category of the input data accurately. While a classification algorithm can work with both discrete and real-valued variables, it is essential that it classifies the examples into two or more classes[26] Figure 2.4.

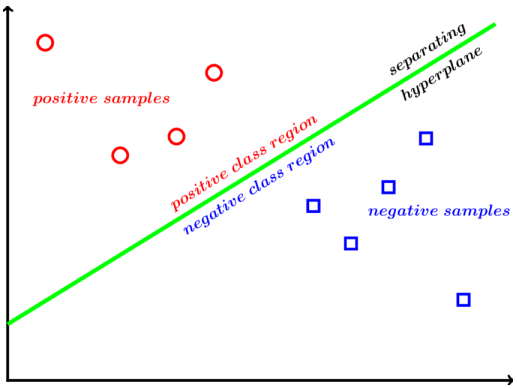


Figure 2.4: Classification Algorithm

2.3.2 Unsupervised learning

Unsupervised learning is a powerful technique in machine learning that allows software engineers to uncover patterns and structures in data without needing labeled examples. It involves training algorithms to identify inherent relationships and similarities within the data itself. By utilizing clustering and dimensionality reduction techniques, unsupervised learning enables software engineers to gain valuable insights and make data-driven decisions. This approach is particularly useful when dealing with large and complex datasets, as it can help uncover hidden patterns and discover new knowledge. Unsupervised learning is crucial in various applications, such as anomaly detection, customer segmentation, and recommendation systems Figure 2.5.

Unsupervised learning models are used for three main tasks: clustering, association, and dimensionality reduction[27]:

1. **Clustering:**

   Clustering is a data mining technique for grouping unlabeled data based on their similarities or differences. For example, K-means clustering algorithms assign similar data points into groups, where the K value represents the grouping size and granularity. This technique is helpful for market segmentation and image compression[28] Figure 2.6.

2. **Association:**

   The association is another type of unsupervised learning method that uses different rules to find relationships between variables in a given dataset. These methods are

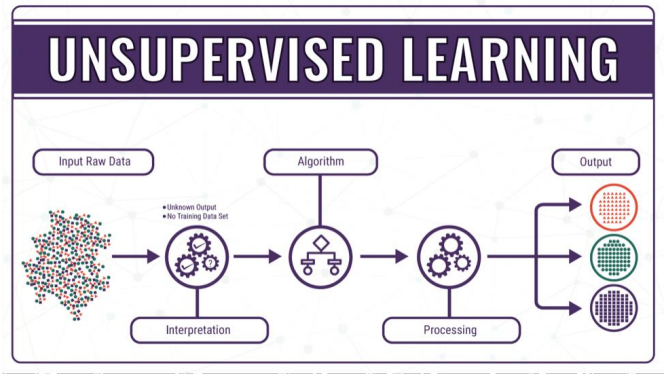


Figure 2.5: Unsupervised Learning

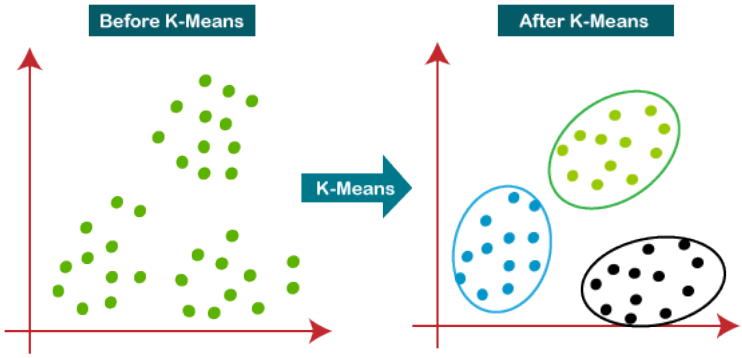


Figure 2.6: Clustering Algorithm

frequently used for market basket analysis and recommendation engines along with "Customers Who Bought This Item Also Bought" recommendations[29] Figure 2.7.

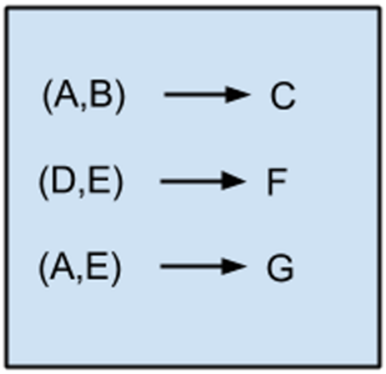


Figure 2.7: Association Rule Learning

3. **Dimensionality reduction:**

   Dimensionality reduction is a learning technique used when the number of features (or dimensions) in a given dataset is too high. It reduces the number of data inputs to a manageable size while also preserving the data integrity. Often, this technique

is used in the preprocessing data stage, such as when autoencoders remove noise from visual data to improve picture quality[30].

2.3.3 Semi-supervised Learning

Can't decide on whether to use supervised or unsupervised learning? Semi-supervised learning is a happy medium where you use a training dataset with both labeled and unlabeled data. It's particularly useful when it's difficult to extract relevant features from data — and when you have a high volume of data.

Semi-supervised learning is ideal for medical images, where a small amount of training data can lead to a significant improvement in accuracy. For example, a radiologist can label a small subset of CT scans for tumors or diseases so the machine can more accurately predict which patients might require more medical attention[31] Figure 2.8.

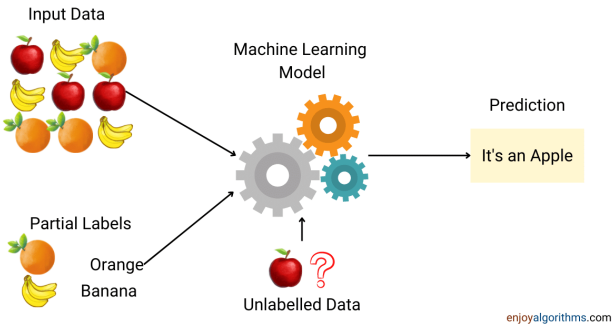


Figure 2.8: Semi-supervised Learning

2.3.4 Reinforcement Learning

Reinforcement Learning is a subfield of machine learning that focuses on training intelligent agents to make decisions and take actions in an environment to maximize a cumulative reward. Unlike other machine learning approaches, reinforcement learning doesn't rely on labeled data but instead learns through trial and error. The agent interacts with the environment, receives feedback through rewards or penalties, and adjusts its actions accordingly to achieve the desired outcome. Through a process of exploration and exploitation, reinforcement learning algorithms learn optimal strategies to solve complex problems. This approach has been successfully applied in various domains, including robotics, game-playing, and autonomous systems[32] Figure 2.9.

2.4 Deep Learning

Deep learning is a subset of machine learning that involves training artificial neural networks with multiple layers to extract hierarchical features from data. It mimics the human brain's interconnected neurons, allowing models to learn patterns and representations from raw input automatically. Deep learning excels at tasks like image and speech

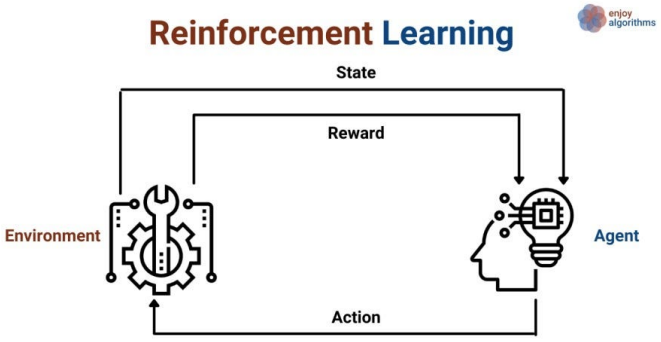


Figure 2.9: Reinforcement Learning

recognition, natural language processing, and more. It has gained prominence due to its ability to handle complex and unstructured data, enabling breakthroughs in various fields by achieving state-of-the-art performance in tasks that were once challenging for traditional machine learning algorithms[33].

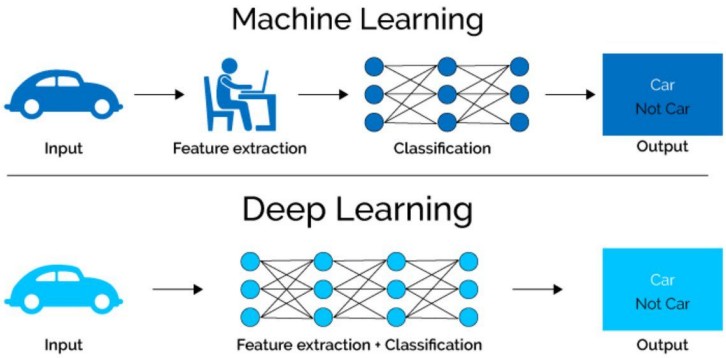


Figure 2.10: Deep Learning

2.4.1 How deep learning works

Deep learning neural networks, or artificial neural networks, aim to replicate the human brain's functioning by combining input data, weights, and biases. These components collaborate to identify, categorize, and elucidate objects within datasets effectively.

Deep neural networks comprise interconnected layers of nodes that progressively refine and enhance predictions or categorizations. This process, termed forward propagation, involves computations passing through the network. The input and output layers are referred to as visible layers. The input layer processes data, while the output layer delivers final predictions or classifications.

Backpropagation, a complementary process, employs algorithms like gradient descent to compute prediction errors and subsequently adjusts weights and biases by retrograding through layers, aiming to train the model. The tandem of forward propagation and back-propagation enables neural networks to predict and rectify errors. Over time, accuracy

improves progressively.

The above explanation provides a simplified overview of basic deep neural networks. Nonetheless, deep learning algorithms are intricate, with diverse types of neural networks catering to specific problems and datasets[34]. For instance

1. **Convolutional neural networks (CNNs):** Convolutional Neural Networks (CNNs) are specialized deep learning models designed for image and visual data processing. They employ convolutional layers to automatically learn hierarchical features from input images, capturing patterns like edges and textures. CNNs are widely used for tasks like image classification, object detection, and facial recognition due to their ability to maintain spatial relationships within data[35] Figure 2.11.

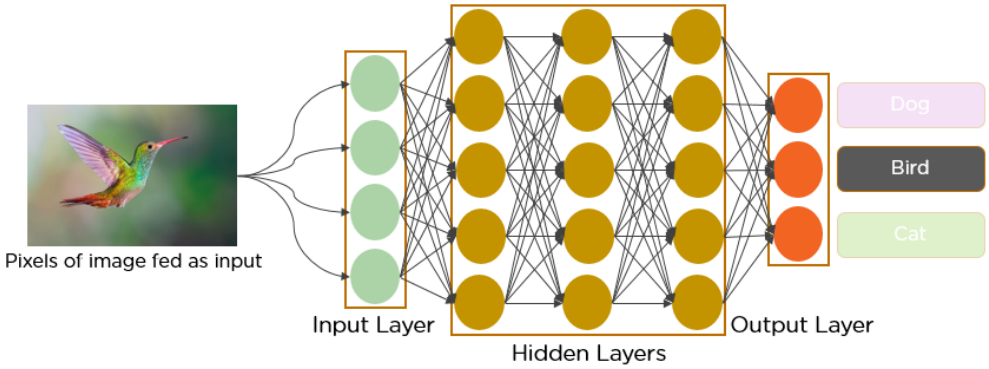


Figure 2.11: Convolutional Neural Networks (CNNs)

The forms of using CNN: There are three main forms of using Convolutional Neural Networks (CNNs)

(a) **Training from Scratch:** Training a Convolutional Neural Network (CNN) from scratch involves starting with a blank slate and developing the model's architecture and parameters from the ground up. This process begins with initializing the network, defining the convolutional layers, choosing activation functions, and setting hyperparameters like filter sizes and layer depths. It also involves configuring pooling layers, dropout to mitigate overfitting, and deciding on the fully connected layers for feature extraction. Training from scratch allows for complete control and customization but typically requires substantial labeled data and computational resources for effective model training.

(b) **Fine-tuning Pre-trained Models:** Fine-tuning a Convolutional Neural Network (CNN) involves taking a pre-trained model, often a well-established architecture like VGG, ResNet, or Inception, and adapting it to a specific task or dataset. Instead of training from scratch, fine-tuning retains the pre-learned features and weights but adjusts them to match the new dataset. This approach saves time and computational resources while allowing for improved performance on the target task. Typically, the final layers are reconfigured to suit the task, and the model is trained with a smaller learning rate to ensure that the pre-learned knowledge is not disrupted.

(c) **Using Pre-trained Models Without Fine-tuning:** Fine-tuning a Convolutional Neural Network (CNN) involves taking a pre-trained model, often a well-established architecture like VGG, ResNet, or Inception, and adapting it to a specific task or dataset. Instead of training from scratch, fine-tuning retains the pre-learned features and weights but adjusts them to match the new dataset. This approach saves time and computational resources while allowing for improved performance on the target task. Typically, the final layers are reconfigured to suit the task, and the model is trained with a smaller learning rate to ensure that the pre-learned knowledge is not disrupted.

2. **Recurrent neural network (RNNs):** A Recurrent Neural Network (RNN) is a type of neural network designed to handle sequential data by introducing connections that loop back on themselves. RNNs can capture dependencies and patterns in time-series data, making them suitable for tasks like natural language processing, speech recognition, and time-series prediction[36] Figure 2.12.

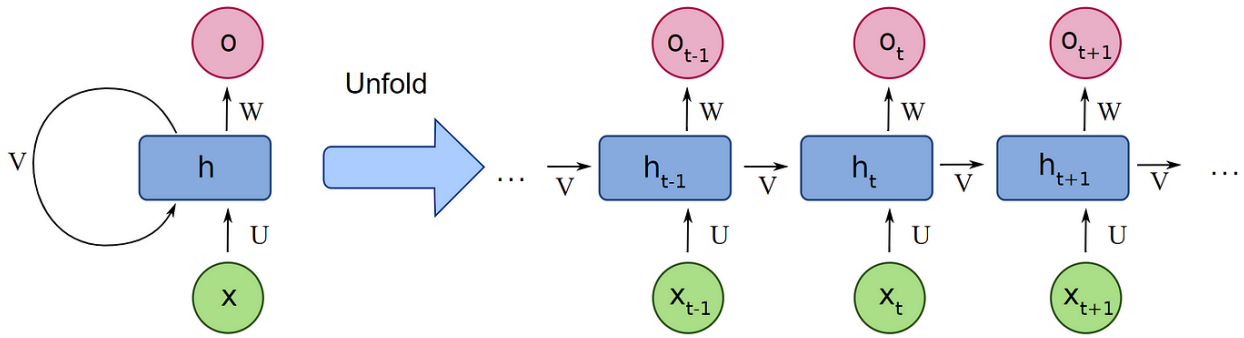


Figure 2.12: Recurrent neural network (RNNs)

2.4.2 How does deep learning attain such impressive results?

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks, like classifying objects in images.

While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

1. Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video.

2. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less[37].

2.4.3 Examples of Deep Learning at Work

Deep learning applications are used in industries, from automated driving to medical devices.

Automated Driving: Automotive researchers are using deep learning to detect objects such as stop signs and traffic lights automatically. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

Aerospace and Defense: Deep learning is used to identify objects from satellites that locate areas of interest and identify safe or unsafe zones for troops.

Medical Research: Cancer researchers use deep learning to detect cancer cells automatically. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to identify cancer cells accurately[38].

2.5 Computer Vision

Computer vision techniques are pivotal in extracting meaningful information from visual data. By emulating human vision, these methods enable machines to interpret and understand the world around us. Leveraging image and video analysis, object recognition, and pattern detection, computer vision empowers applications like facial recognition, autonomous vehicles, and medical imaging. Deep learning algorithms, such as convolutional neural networks (CNNs), have revolutionized this field by enhancing accuracy and efficiency. As a result, computer vision techniques continue to reshape industries, offering innovative solutions that bridge the gap between human perception and artificial intelligence.

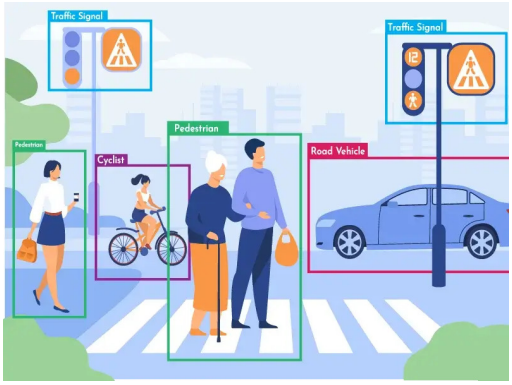


Figure 2.13: Computer Vision

2.5.1 What is Computer Vision?

Computer vision is a field of artificial intelligence (AI) that allows computers and systems to derive meaningful information from digital images, video, and other visual input and to take action or make recommendations based on this information. If artificial intelligence allows computers to think, computer vision allows them to see, observe, and understand.

Computer vision works like human vision, but humans are one step ahead. The human sight has the advantage of being able to train itself to distinguish objects, determine their distance, know if they are in motion and if something is wrong in an image.

Computer vision trains machines to perform these functions, but it must do so much faster using cameras, data, and algorithms that replace our retinas, optic nerves, and visual cortex. Because a system trained to inspect products or monitor a production asset can analyze thousands of products or processes per minute, noticing imperceptible defects or problems, it can quickly overwhelm human capabilities.[39]

2.5.2 How does computer vision work?

Computer vision needs a lot of data. She runs data analyses repeatedly until she perceives distinctions and finally recognizes the images. For example, to train a computer to recognize car tires, it must receive many tire images and tire-related items to learn the differences and recognize a tire, especially a flawless tire.

Two core technologies are used in computer vision: a type of machine learning called deep learning and a convolutional neural network (CNN).

Machine learning uses algorithmic models that allow a computer to train itself on the context of visual data. If enough data is fed into the model, the computer will "look" at the data and learn to distinguish one image from another. Algorithms allow the machine to learn on its own, rather than someone programming it to recognize an image.

A convolutional neural network (CNN) helps a machine learning or deep learning model "look" by breaking down images into pixels that are assigned tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and predict what it "sees". The neural network performs convolutions and checks the accuracy of its predictions through a series of iterations until the predictions start to come true. It is then a question of recognizing or seeing images similarly to humans.

Like a human being who makes out an image from a distance, a CNN first discerns hard edges and simple shapes, then fills in the information as it iterations of its predictions. A CNN is used to understand individual images. A recurrent neural network (RNN) is similarly used in video applications to help computers understand how images in a series relate to each other[40].

2.5.3 Types of Computer Vision Algorithms

Image Classification: This algorithm classifies images into predefined categories or classes. It uses machine learning techniques, such as convolutional neural networks (CNNs), to learn and recognize patterns and features in images.

Object Detection: Object detection algorithms identify and locate specific objects within an image or video. They utilize techniques like region-based convolutional neural networks (R-CNN), You Only Look Once (YOLO), or Single Shot MultiBox Detector (SSD) to detect and localize objects.

Semantic Segmentation: Semantic segmentation algorithms assign semantic labels to each pixel in an image, enabling the understanding of the scene's structure. They

are commonly used in applications like autonomous driving, medical imaging, and video surveillance.

Optical Character Recognition (OCR): OCR algorithms recognize and extract text from images or scanned documents. They employ techniques like feature extraction, character segmentation, and pattern recognition to convert visual text into editable and searchable formats[41] Figure 2.14.

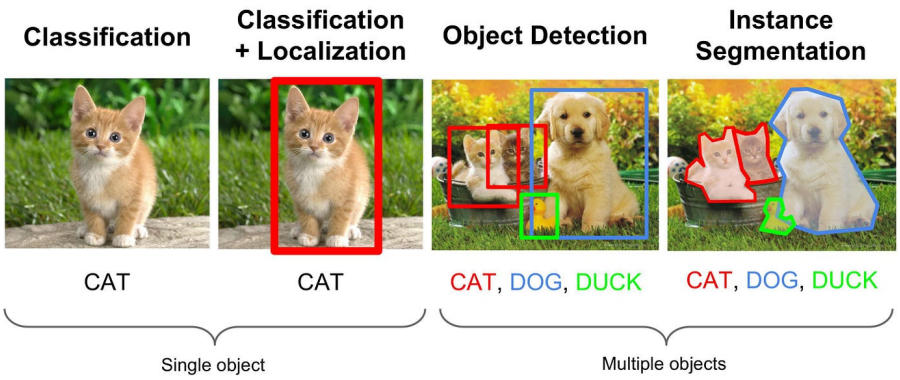


Figure 2.14: Computer Vision

2.5.4 Computer vision applications

There's a lot of research going on in the field of computer vision, but it's not just about research. Real-world applications demonstrate the importance of computer vision for business, entertainment, transportation, healthcare, and everyday life. One of the main drivers of the growth of these applications is the flood of visual information from smartphones, security systems, traffic cameras and other devices with visual instruments. This data could play a major role in the operations of any industry, but today it sits unused. The information creates a testbed for training computer vision applications and a launch pad for them to become part of a range of human activities:

IBM used computer vision to create My Moments for the 2018 Masters golf tournament. IBM Watson looked at hundreds of hours of film recordings of Masters and was able to identify the sights (and sounds) of important shots. These key moments were curated and delivered to fans as custom footage.

Google Translate allows users to point a smartphone camera at a sign written in another language and almost immediately get a translation of that sign in their preferred language.

The development of autonomous vehicles relies on computer vision, which makes sense of the visual data provided by the car's cameras and other sensors. It is essential to identify other cars, traffic signs, lane markers, pedestrians, bicycles, and all other visual information encountered on the road.

Computer vision plays a crucial role in the detection of dangerous driving behavior. By leveraging advanced algorithms and techniques, computer vision systems can analyze visual data, such as images or videos, to identify and classify various risky behaviors exhibited by drivers. Using image processing, object detection, and machine learning,

computer vision models can detect actions like texting, distracted driving, or aggressive maneuvers. These models can analyze driver's facial expressions, hand movements, and body postures to determine if they are engaged in dangerous behavior. By combining computer vision with intelligent algorithms, we can develop effective systems for real-time monitoring and detecting dangerous driving behavior, ultimately enhancing road safety[42].

Autonomous Vehicles: Computer vision enables autonomous vehicles to perceive and understand their surroundings. It helps in object detection, lane detection, traffic sign recognition, and pedestrian detection, ensuring safe and efficient navigation.

Surveillance and Security: Computer vision algorithms are used in video surveillance systems to detect and track suspicious activities, identify individuals, and monitor restricted areas. They enhance security and aid in crime prevention.

Medical Imaging: Computer vision is extensively used in medical imaging applications, such as MRI, CT scans, and pathology analysis. It assists in the detection and diagnosis of diseases, tumor segmentation, and image-guided surgeries.

Augmented Reality (AR): AR applications rely on computer vision algorithms to overlay virtual objects in the real world. They enable immersive experiences in gaming, education, and industrial training[43] Figure 2.15.

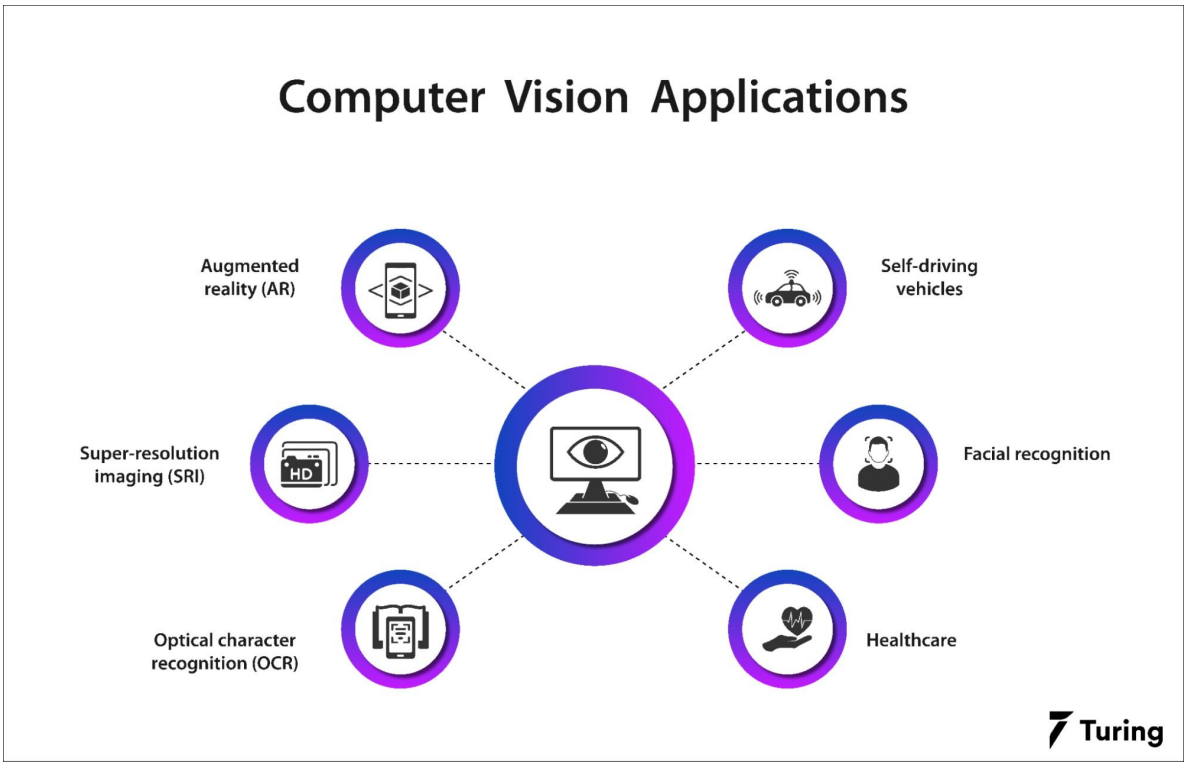


Figure 2.15: Computer Vision

2.6 Related work

Numerous studies have extensively explored the detection of dangerous driving behavior, playing a pivotal role in enhancing road safety and mitigating the risks associated with reckless driving. These investigations have underscored the significance of accurately identifying and categorizing perilous driving behaviors in real time. As the number of vehicles on the roads continues to rise and concerns regarding road safety escalate, there is a growing interest in developing effective systems and algorithms for detecting dangerous driving behavior. Through the utilization of computer vision techniques, machine learning algorithms, and sensor data fusion, researchers have made remarkable advancements in this domain. The findings of these studies have paved the way for the creation of sophisticated systems capable of proactively detecting and addressing dangerous driving behaviors, thereby fostering safer roads and heightened driver awareness. has been explored in several studies.

In work titled 'Recognition of dangerous driving behavior based on support vector machine' [44], the study of identifying dangerous driving behavior is valuable for controlling how drivers behave on the road. However, current algorithms are easily influenced by noise and abnormal data, which can hinder accurately identifying dangerous driving behaviors. In this research paper, a new approach to studying driving behavior is introduced. This method utilizes Support Vector Machine (SVM) and oversampling techniques to build a driving behavior recognition model. The experimental findings indicate that the proposed model achieves a higher rate of accurately recognizing dangerous driving behaviors.

To mitigate the occurrence of traffic accidents resulting from dangerous driving, a recognition model for identifying dangerous driving behavior is introduced. This model [45] combines Convolutional Neural Network (CNN) and Long Short-Term Memory Network (LSTM). To address the issue of low accuracy in the network model's identification, optimization is performed by introducing an unsupervised attention mechanism. By integrating the attention-weighted module and the convolution LSTM, the model focuses on a specific visual area, thereby enhancing the algorithm's recognition accuracy to some extent. Experimental results demonstrate that the proposed algorithm exhibits improved detection accuracy and rate compared to the Two-Stream method and C3D behavior recognition algorithm in the task of recognizing dangerous driving behavior.

This study [46] proposes a model for detecting dangerous driving by fusing multiple sources of information. The information fusion model incorporates features related to face fatigue and distracted driving. Face fatigue characteristics, such as blink rate, yawning behavior, and head posture offset, are detected using OpenCV and Dlib libraries. Distracted driving behaviors, including phone usage, drinking water, and mobile phone distraction, are detected using an improved YOLO-V5 algorithm. The features of fatigue and distraction are then fused using the D-SET algorithm. By combining these features, the proposed model achieves enhancing detection accuracy and system robustness.

In this study [47], a dangerous driving behavior recognition model is developed using hand trajectory data and the dynamic time warping algorithm (DTW) along with the longest common sub-sequence algorithm (LCS). These algorithms are employed to determine the label for dangerous driving behavior. The model calculates the matching degree

of hand trajectory data and establishes a threshold for recognizing dangerous driving behavior based on the calculation results. Furthermore, a comparison and analysis are conducted between the dangerous driving behavior recognition algorithm and the neural network algorithm. The dangerous driving behavior recognition algorithm exhibits fast calculation speed, low memory consumption, and a simple program structure. The research findings have practical applications in dangerous driving behavior recognition and driving distraction warning systems utilizing wrist wearable devices.

In this research [48], the focus is on detecting abnormal driving behavior using vision-based methods. Recent progress in deep learning has made this task easier, thanks to advanced models that have excellent generalization capabilities and the availability of large amounts of video data for training. To conclude the study, novel deep learning models inspired by a popular fully connected convolutional network called the Abnormal Driving Behavior Detection (ADBD Net) are introduced.

2.7 Conclusion

In conclusion, the second chapter of the dissertation explores the fascinating concepts of Machine Learning, Deep Learning, and Computer Vision and their relationship to detecting dangerous driving behavior. Machine Learning, a subset of Artificial Intelligence, enables systems to learn and improve from data without explicit programming. Deep Learning, on the other hand, utilizes neural networks with multiple layers to extract complex patterns and features from data. These techniques have revolutionized various fields, including Computer Vision. Computer Vision focuses on enabling machines to understand and interpret visual information. By combining Machine Learning and Deep Learning algorithms, we can develop robust models capable of detecting dangerous driving behavior. These models can analyze real-time video feeds, identify risky actions such as distracted driving or aggressive maneuvers, and alert drivers or authorities to prevent accidents. The integration of Machine Learning, Deep Learning, and Computer Vision in the detection of dangerous driving behavior holds immense potential for enhancing road safety. By continuously refining these models and leveraging real-time data, we can create intelligent systems that contribute to a safer driving environment for everyone.

Chapter 3

Our Proposed Methodology

3.1 Introduction

In Chapter Three, which focuses on design and model proposal, we will present a modified model to detect dangerous driving behaviors. We will harness the power of convolutional networks in crafting this model, as they exhibit remarkable predictive capabilities. The neural network executes convolutions and validates the accuracy of predictions through a series of iterations until confidence is attained. Subsequently, images are identified or perceived in a manner akin to human perception. This chapter elucidates the system's workflow, spanning from data preparation to prediction processes, detailing the stages involved.

Previous studies on detecting dangerous driving behavior have played a crucial role in enhancing road safety and reducing the risks associated with reckless driving. These studies have highlighted the importance of accurately identifying and classifying dangerous driving behaviors in real-time. With the increasing number of vehicles on the roads and the rising concern for road safety, there is a great interest in developing effective systems and algorithms for detecting dangerous driving behavior. By leveraging computer vision techniques, machine learning algorithms, and sensor data fusion, researchers have made significant strides in this field. The outcomes of these studies have paved the way for developing advanced systems that can proactively detect and mitigate dangerous driving behaviors, ultimately contributing to safer roads and improved driver awareness. has been explored in several studies.

3.2 Proposed Method

Similar to a person remotely perceiving an image, the CNN initially identifies distinct sharp boundaries and basic forms. Subsequently, it completes the missing details during its predictive iterations. CNN serves to comprehend individual images. Within this section, we will demonstrate the methodologies and CNN architecture we employed for training and evaluating our dataset.

The proposed method aims to detect instances of dangerous driving by leveraging advanced image processing techniques and deep learning algorithms. By analyzing driver behavior and visual cues from the environment, the method employs a trained Convolutional Neural Network (CNN) model to identify patterns associated with hazardous driving actions. This approach offers an effective means to enhance road safety through automated recognition and alerting of risky driving behaviors.

The proposed method for detecting dangerous driving behavior from a dataset involves several key steps Figure 3.1.

1. **Dataset:** Begin by collecting a comprehensive dataset of images that represent various driving scenarios and behaviors. This dataset should cover a wide range of driving situations.

2. **Preprocessing images:** Clean and preprocess the images to ensure consistency and quality. This includes resizing, normalization, and data augmentation to enhance the dataset's diversity.

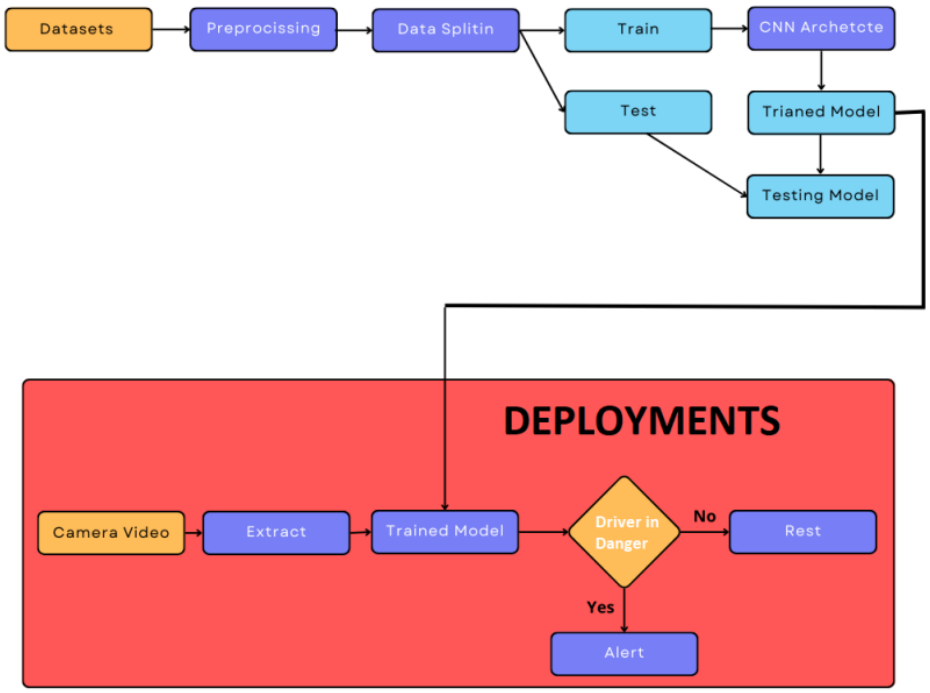


Figure 3.1: The Proposed Method Design

3. **Splitting Dataset:** Dataset splitting is crucial for proper model training, validation, and evaluation. The dataset is divided into three sets: training, validation, and testing. The training set is used to train the deep learning model, while the validation set helps fine-tune the model and prevent overfitting. The testing set is reserved for evaluating the final performance of the trained model, ensuring an unbiased assessment.

4. **Training Data:** Split the dataset into training and testing data. The training data is used to teach the model to recognize dangerous driving behaviors by providing it with labeled examples.

5. **Testing Data:** The testing data evaluates the model's performance. It consists of images the model has never seen before, allowing you to assess its ability to generalize to new scenarios.

6. **CNN Model Proposed:** Design a Convolutional Neural Network (CNN) model tailored for image classification. The architecture should include convolutional layers for feature extraction and dense layers for classification. Experiment with different architectures and hyperparameters to optimize performance.

7. **Model deployment(Dangerous driving behavior Prediction):** Once the model is trained, it can predict dangerous driving behaviors in real-time or on new images. The model assigns a probability score to each behavior category, helping identify potential risks on the road.

3.2.1 Preprocessing data

Preprocessing data is a crucial step in preparing raw data for machine learning models. In this project, the following steps are followed for data preprocessing: Firstly, we identify the paths and directories for data, including training and testing data, random data, models, and corrupted files. If these paths do not exist, we create them if necessary. Next, we create files to organize the training and testing data, storing the paths of the images and corresponding class names. This function is then applied to both the training and testing directories, ensuring organized data storage Figure 3.2.

The preloaded data is processed as follows:

- Class labels are converted into numerical values, making it easier to train the models.
- Class labels are transformed into a categorical format, which is necessary for multi-class classification.
- To ensure proper model evaluation, the dataset is divided into training and validation sets. This division prevents overfitting and allows for monitoring the model's performance.
- Preprocessing functions for images are defined to convert image paths into tensors. The grayscale images are loaded and transformed into 4D tensors. This function is applied, resulting in the creation of a stack of 4D tensors.
- Finally, the preprocessed data undergoes normalization and preprocessing of the training and validation tensors. This ensures that the data is in a suitable format to be fed into the neural network model, enhancing model convergence and performance.

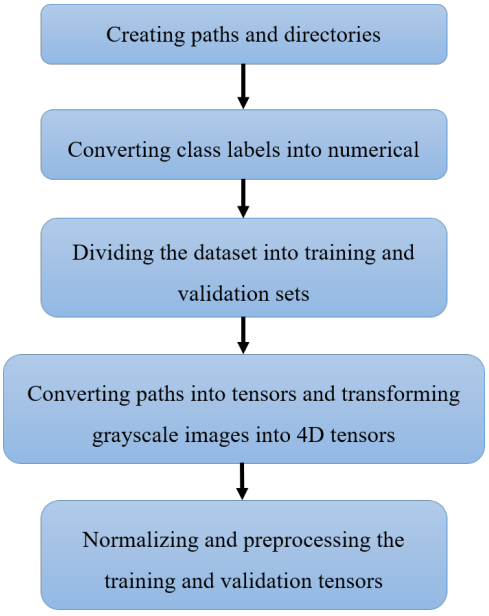


Figure 3.2: The Preprocessing data steps

Preprocessing is a crucial step in preparing image data for machine learning tasks. In the provided code snippets, preprocessing steps are outlined. The first code snippet involves converting an image from a file path to a 4D tensor suitable for deep learning models. Initially, the image is loaded using the 'image.load_img' function, specifying the color mode as grayscale and resizing it to (224, 224) pixels. Then, it's converted to a 3D tensor with dimensions (224, 224, 1) using 'image.img_to_array'. Finally, the 3D tensor is expanded to a 4D tensor with shape (1, 224, 224, 1) to match the input requirements of the model.

The second code snippet achieves a similar preprocessing objective. It loads an image as grayscale, resizes it, and then converts it into a 3D tensor. The difference here is that it doesn't use 'np.expand_dims' to add an extra dimension for the batch size, so it results in a 3D tensor with shape (224, 224, 1).

The specific choice between these two preprocessing methods depends on the architecture of the neural network model you intend to use. Some models may expect the input data in a 4D format, while others can accept a 3D format. Be sure to choose the method that aligns with your model's requirements.

3.2.2 Model Design

This algorithm describes the sequential construction of a CNN model for detecting and classifying dangerous driving behaviors. Each step adds a specific layer to the model, progressively building its architecture. The model's summary provides an overview of its structure and parameters, which helps understand its design and complexity. We will explain the techniques and layers used to create this proposed model:

Convolutional Layer: This particular layer is the cornerstone of the Convolutional Neural Network. Convolution, in essence, is the mathematical operation of combining two functions to create a third function. In the context of neural networks, it utilizes a sliding window approach to extract features from images. This process plays a crucial role in generating feature maps. It essentially involves the convolution of two functional matrices: one being the input image matrix A, and the other being the convolutional kernel B, ultimately yielding the output C Figure 3.3[49].

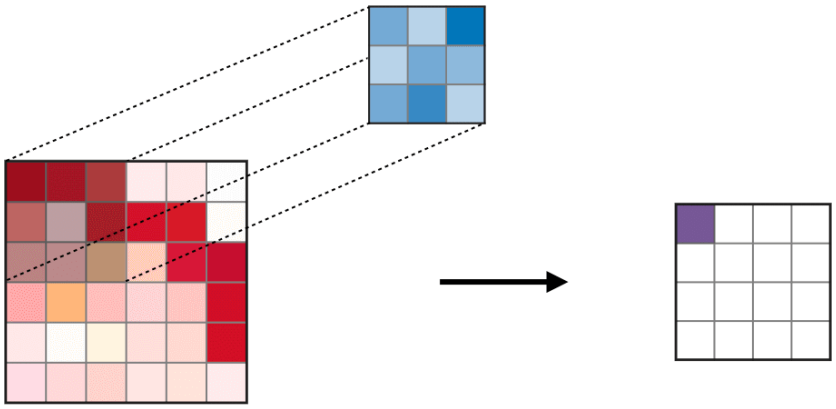


Figure 3.3: Convolutional Operation Diagram

Pooling Layer: Pooling operations accelerate computations by downsizing the input matrix while retaining essential features. Various types of pooling techniques can be applied [50]. We just touched on:

- Max Pooling: Within the specified region covered by the kernel, the maximum value is selected and assigned as the output matrix value for that cell Figure 3.4.

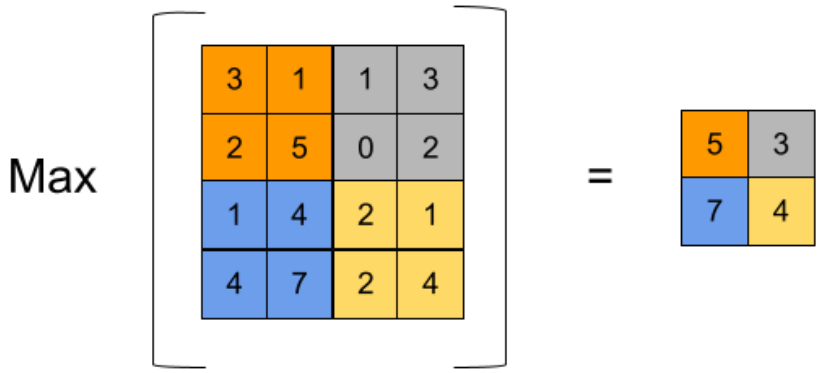


Figure 3.4: Max-Pooling Operation

The Dropout Layer: addresses overfitting by eliminating randomly selected neurons from the model. These neurons can be located in both the visible and hidden layers of the neural network. Adjusting the dropout ratio lets you control the probability of a neuron being excluded[51] Figure 3.5.

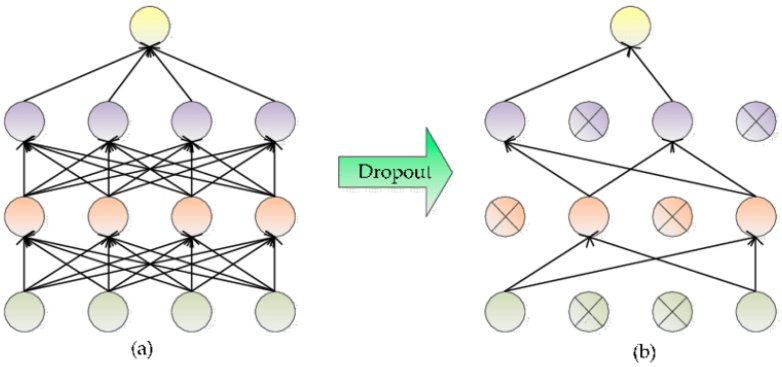


Figure 3.5: The Dropout Layer Diagram

Non-Linear Layer: Typically, these layers are added following the convolutional layers. The frequently employed non-linear functions consist of different variations of Rectified Linear Units (ReLU). Other functions encompass the sigmoid and tanh functions. Below, you'll find a range of non-linear functions and their corresponding equations[52] Figure 3.6.

The CNN model is designed based on the following:

To begin, we initialize a sequential neural network model. Convolutional layers are added, progressively increasing the number of filters while employing the **relu** activation function. Max pooling layers are introduced to downsample and reduce spatial dimensions, aiding in feature extraction. Dropout layers are incorporated to prevent overfitting by

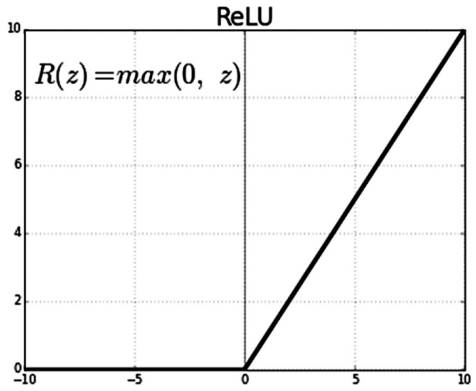


Figure 3.6: Rectified Linear Units Diagram

randomly deactivating neurons during training. The data is then flattened into a 1D vector for seamless integration with subsequent layers. Dense (fully connected) layers with **relu** activation are added to perform complex feature extraction and transformation. The final dense layer uses **softmax** activation for multi-class classification, providing probabilities for each class.

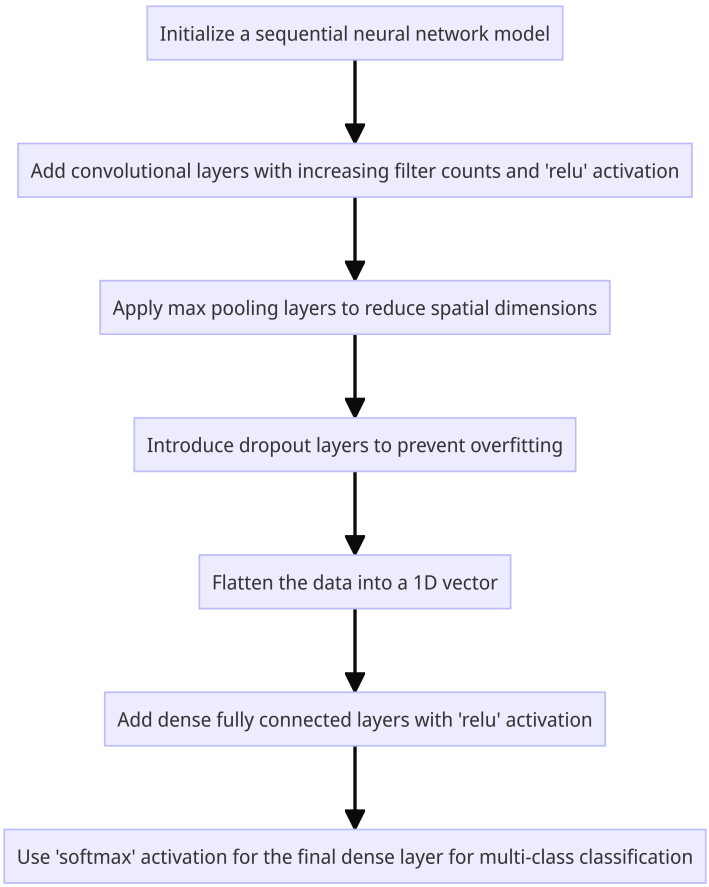


Figure 3.7: Design the model Diagram

This algorithm guides the construction of a Convolutional Neural Network (CNN)

model tailored to detect and classify dangerous driving behaviors effectively.

Algorithm 1 Building a Convolutional Neural Network (CNN) Model

1: Initialize a sequential neural network model.
2: Progressive addition of convolutional layers with 'relu' activation and max pooling layers for feature extraction and spatial reduction.
3: Include dropout layers to prevent overfitting.
4: Construct a Convolutional Neural Network (CNN) model for effective detection and classification of dangerous driving behaviors, with a final dense layer using 'softmax' activation for multi-class classification.

The proposed CNN model comprises several layers, each with its own set of weights and parameters, contributing to the system's effectiveness in detecting dangerous driving behavior.

Convolutional Layers: These are the model's core, responsible for feature extraction. The first layer consists of 64 filters with a 2x2 kernel size, and each filter has its own set of learnable weights. The subsequent layers follow a similar pattern, with 128, 256, and 512 filters, each introducing its unique weight set. These layers use the 'ReLU' activation function for non-linearity.

MaxPooling Layers: After each Convolutional Layer, a MaxPooling Layer with a 2x2 pool size is applied. These layers don't introduce additional weights but help reduce spatial dimensions.

Dropout Layers: Two Dropout Layers are inserted with a dropout rate of 0.5 to prevent overfitting. These layers don't have weights but randomly deactivate a fraction of neurons during training to enhance generalization.

Flatten Layer: Before the fully connected layers, the Flatten Layer reshapes the 2D feature maps into a 1D vector, and it doesn't have any weights.

Dense Layers: The first Dense Layer comprises 500 units, each connected to the flattened vector, introducing a significant number of weights. The activation function used here is 'ReLU'. Another Dropout Layer with a rate of 0.5 is employed to prevent overfitting. The final Dense Layer contains 10 units, corresponding to the classification categories, and uses 'softmax' activation to convert the model's output into class probabilities.

In summary, each Convolutional Layer, MaxPooling Layer, and Dense Layer has its weights, contributing to the model's ability to learn and recognize patterns associated with dangerous driving behavior. The Dropout Layers prevent overfitting, enhancing the model's robustness during training. These weights and parameters collectively enable the system to effectively detect and classify risky driving behavior Figure 3.8.

3.3 Model Deployment

As a first step, I will capture a continuous video stream from a camera mounted inside the car. This camera will focus on the driver's body, and I will ensure that it has the appropriate frame rate and resolution to ensure accurate detection. By continuously

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 64)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 128)     32896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 256)       131328
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 512)       524800
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 512)       0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 512)       0
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
dense (Dense)                (None, 500)               50176500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                5010
=================================================================
Total params: 50,870,854
Trainable params: 50,870,854
Non-trainable params: 0
```

Figure 3.8: The proposed CNN model layers

recording the driver's actions, I can analyze their behavior and identify any potential risks or distractions. This video stream will serve as valuable input for further analysis and development of safety measures.

3.3.1 Extracting the Next Frame from an Incoming Video Stream

I will set up the system to extract individual image frames from the incoming video stream constantly, typically every second. This repeated frame extraction allows for continuous monitoring and evaluation of my state while behind the wheel. Each extracted image frame will serve as a snapshot of my driver's position at a specific moment in time. By collecting these frames at regular intervals, I create a timeline that enables the system to track changes and detect any signs of distraction, phone calls, or improper postures during driving. This continuous assessment through frame extraction is crucial in identifying the ten critical scenarios present in the dataset, whether they occur intentionally or unintentionally. Early detection is key to preventing potential accidents. By analyzing a sequence of frames over time, the system can provide timely warnings or alerts to help me stay attentive and safe on the road.

3.3.2 Dangerous driving behaviours

To prevent false alarms, I will set up a mechanism to avoid instantaneously triggering alerts. One approach is to request detection of the following states: c0: normal driving, c1: texting - right, c2: talking on the phone - right, c3: texting - left, c4: talking on the phone - left, c5: operating the radio, c6: drinking, c7: reaching behind, c8: hair and makeup, c9: talking to a passenger, for a minimum duration of several consecutive frames before incrementing the "dangerous state" counter. This will give me a more reliable measure of the 10 dangerous driving scenarios. Additionally, I will track the timestamp of each frame, which will assist me in measuring the duration of each state.

3.3.3 Using the model in detection

Using the model in detection, after training the model, it can be utilized to detect instances of dangerous driving and provide audio alerts to the driver. This application of the model is crucial in enhancing road safety and preventing accidents.
The model is designed to analyze various parameters such as vehicle speed, lane deviation, and driver behavior to identify potentially hazardous situations. By continuously monitoring these factors, the model can detect actions like sudden braking, aggressive acceleration, or erratic lane changes that may indicate risky driving behavior.
Once the model detects a potentially dangerous situation, it triggers an audio alert to notify the driver. This alert can be in the form of a voice message or a warning sound, depending on the system's design. The purpose of the audio alert is to grab the driver's attention and prompt them to take corrective action, thereby reducing the risk of accidents.
To ensure the accuracy and reliability of the detection system, extensive testing and validation are conducted during the model development process. This includes training the model on a diverse dataset that encompasses various driving scenarios and conditions. Additionally, continuous monitoring and feedback from real-world driving situations help refine and improve the model's performance over time.
The integration of this detection model into vehicles has the potential to significantly improve road safety by providing timely warnings to drivers and promoting safer driving habits. However, it is important to note that the model is not a substitute for responsible driving practices and should be used as a supplementary tool to assist drivers in making informed decisions on the road.

3.3.4 Developing a web application

After designing the model and completing its training, we proceed to create an application where we utilize this model for predicting dangerous driving behaviors.

In addition, we have developed A web application designed for predicting dangerous driving behaviors utilizing a straightforward approach. It begins by defining functions that validate uploaded files and perform behavior predictions using the pre-trained model. The application's core revolves around loading this model, allowing it to make real-time predictions Figure 3.9.

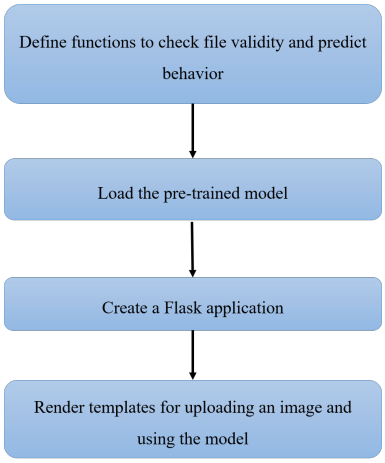


Figure 3.9: Steps to design a web application Diagram

The user interface is created through template rendering, providing users with the capability to upload an image directly to the application. Once the image is uploaded, the model processes it and predicts the corresponding driving behavior, offering a seamless and user-friendly experience for behavior prediction.

3.4 Conclusion

In conclusion, this chapter has discussed the proposed method for a system aimed at detecting risky driving behaviors. We began by emphasizing the importance of having an integrated and comprehensive dataset and outlined the required preprocessing steps for data consistency and quality. The core of our approach lies in **the CNN model** we proposed, specifically designed for image classification tasks, particularly the recognition of dangerous driving behaviors. We elaborated on the architectural engineering, including the convolutional and dense layers. Furthermore, we presented an **Android application** for real-time operation and explained its design stages until the deployment process. We also introduced the concept of a **web application** designed for predictive purposes, utilizing the trained model. This application will allow users to upload images for real-time behavior prediction. In the upcoming practical chapter, we will implement and validate these proposals, transitioning from theory to practical application.

Chapter 4

Implementation and Results

4.1 Introduction

In this chapter, we will delve into the practical implementation of the Dangerous Driving Behavior Detection system. We will discuss the necessary tools, programming languages, and libraries required for this project. Additionally, we will explore popular editors such as Anaconda, VSCode, and Kaggle, which can greatly facilitate the development process. Finally, we will analyze the results obtained from our experiments. To begin with, let's focus on the tools and programming languages that will enable us to build the Dangerous Driving Behavior Detection system. We will carefully select the most suitable programming language based on factors such as performance, ease of use, and availability of relevant libraries. This choice will play a crucial role in the success of our project.

4.2 Tools and Libraries

4.2.1 Hardware environment

The computer hardware device that boasts impressive specifications. It is equipped with an **Intel(R) Core(TM) i7-8650U CPU**, which operates at a base frequency of 1.90GHz and can reach a maximum turbo frequency of 2.11GHz. This powerful processor ensures smooth and efficient performance for various computing tasks. In terms of memory is equipped with **16.0 GB of RAM**, with 15.4 GB being usable. This ample amount of RAM allows for seamless multitasking and the smooth execution of resource-intensive applications. Runs on **the Windows 11 Professional** 64-bit operating system, specifically designed to take advantage of the capabilities of a 64-bit processor like the x64-based processor found. With its robust hardware configuration, Imane is well-equipped to handle demanding computing tasks and provide a seamless user experience.

4.2.2 Python

Python is a programming language that is interpreted, object-oriented, and high-level. It has dynamic semantics, which means it allows for flexibility in coding. Python's built-in data structures and dynamic typing make it a popular choice for Rapid Application Development. It can also be used as a scripting or glue language to connect different components together. Python is known for its simplicity and easy-to-learn syntax, which promotes readability and reduces the cost of program maintenance. It supports modules and packages, which encourage modularity and code reuse. The Python interpreter and extensive standard library are freely available in both source and binary form for all major platforms[53] Figure 4.1.

4.2.3 Keras

Keras is a widely used open-source deep-learning library in Python. It offers a user-friendly interface for constructing and training neural networks, simplifying the implementation of complex machine-learning models. Keras allows for quick prototyping and experimentation with various architectures, providing pre-built layers, activation functions, and

Figure 4.1: The logo of python

optimizers. Its simplicity and intuitive design enable developers to define neural networks using a sequential or functional API. Keras supports both CPU and GPU acceleration, facilitating faster training and inference. It seamlessly integrates with TensorFlow and Theano, offering flexibility and a simplified syntax. Keras also provides tools for model evaluation and visualization and has a vibrant community for support and learning.[54] Overall, Keras empowers software engineers to build and deploy deep learning models effortlessly Figure 4.2.

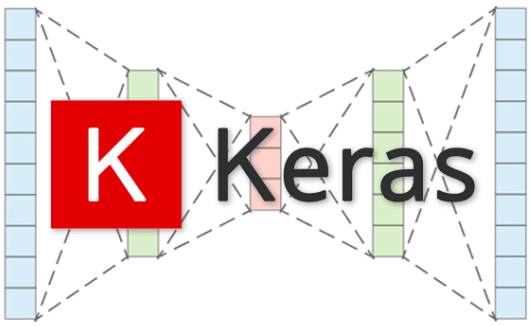


Figure 4.2: The logo of Keras

4.2.4 Tensorflow

TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive platform for building and deploying machine learning models, particularly neural networks. TensorFlow offers many tools and libraries to facilitate tasks like data manipulation, model training, and deployment. It supports both CPU and GPU computations, enabling efficient processing of large-scale datasets. TensorFlow's flexibility and scalability make it a popular choice for various applications, from image and speech recognition to natural language processing and more Figure 4.3.

Figure 4.3: The logo of Tensorflow

4.2.5 Anaconda

Anaconda is a freely available distribution of the Python and R programming languages that are specifically designed for data science purposes. Its main goal is to streamline the process of managing and deploying packages. In Anaconda, package versions are handled by a package management system called Conda. Before installing a package, condo carefully analyzes the current environment to ensure that the installation does not interfere with other frameworks and packages Figure 4.4.

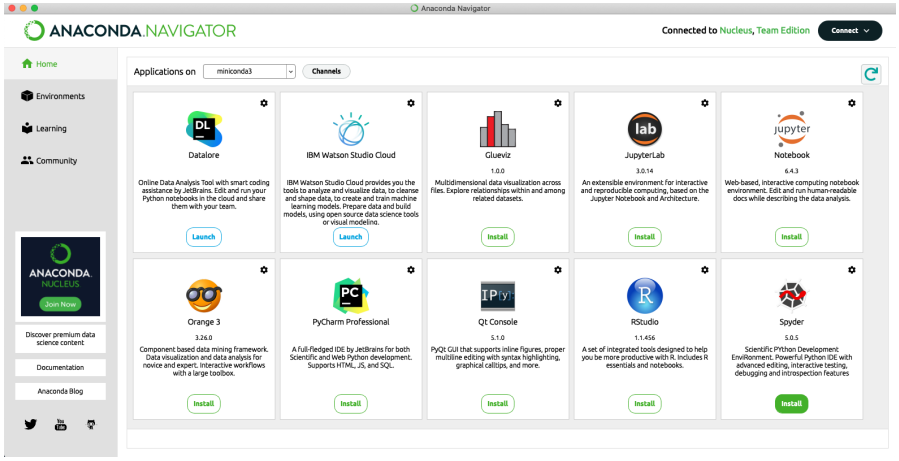


Figure 4.4: The interface of Anaconda

4.2.6 VSCode

Visual Studio Code (VS Code) is a powerful and versatile code editor that has gained immense popularity among Python developers. With its extensive features and intuitive interface, it offers numerous benefits for Python code editing. Firstly, VS Code provides excellent language support for Python, including syntax highlighting, code completion, and error checking. This helps developers write clean and error-free code, enhancing productivity and reducing debugging time. VS Code also supports version control systems

like Git, allowing developers to manage their code repositories effortlessly. The built-in Git integration provides features such as commit history, branch management, and conflict resolution, facilitating collaborative development. Furthermore, VS Code offers a customizable and extensible environment. Developers can personalize their editor by choosing from various themes, icons, and settings. They can also create custom workflows by configuring keybindings and utilizing task automation tools Figure 4.5.

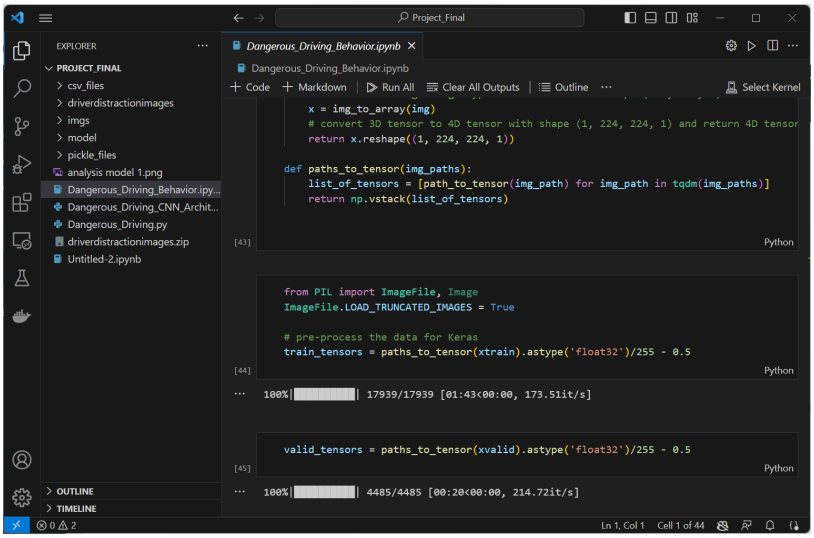


Figure 4.5: The interface of VSCode using Jupyter editor

4.2.7 Kaggle

Kaggle is a popular online platform for data science and machine learning enthusiasts. It offers many datasets, competitions, and collaborative coding environments that can greatly benefit Python developers. With Kaggle, developers can access a vast collection of datasets, which can be used for training and testing machine learning models. The platform also provides a collaborative coding environment where developers can write and execute Python code directly in their web browser. This feature enables seamless collaboration and knowledge sharing among developers. Additionally, Kaggle hosts competitions where developers can showcase their coding skills and compete with others. These competitions provide valuable learning opportunities and help developers improve their Python coding proficiency[55]. Overall, Kaggle serves as a valuable resource for Python developers, offering a platform to learn, collaborate, and enhance their coding skills in the fields of data science and machine learning Figure 4.8.

4.2.8 Android studio

Android Studio is an integrated development environment (IDE) specifically designed for Android app development. It provides a comprehensive set of tools and features that enable software engineers to create, test, and debug Android applications efficiently. With Android Studio, developers can write code in languages like Java and Kotlin, and leverage

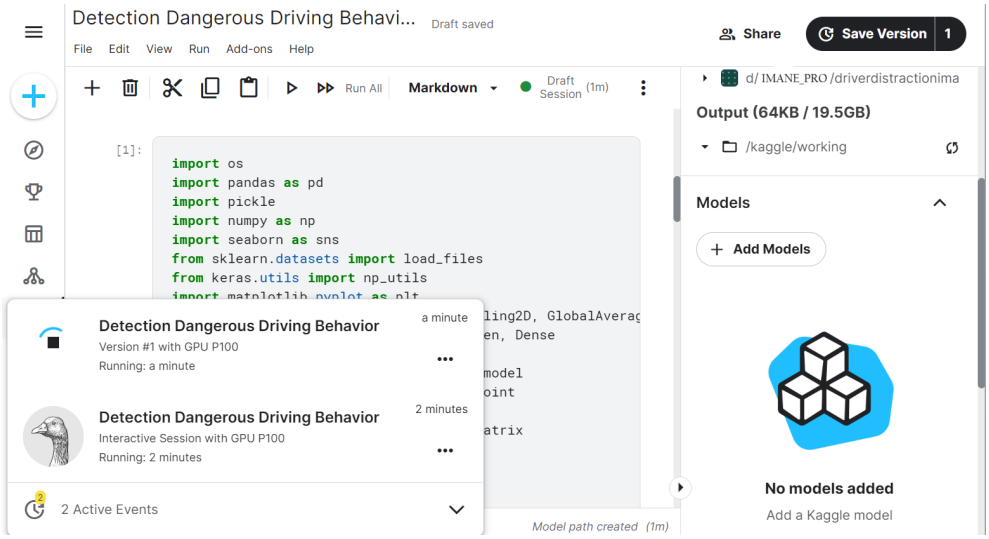


Figure 4.6: The interface of Kaggle

various libraries and frameworks to build robust and user-friendly apps. The IDE offers a rich set of features, including a visual layout editor, code completion, debugging tools, and an emulator for testing apps on virtual devices. Android Studio also integrates seamlessly with the Android SDK, allowing developers to access a wide range of APIs and resources to enhance their app's functionality and performance. Overall, Android Studio is a powerful tool that empowers software engineers to create high-quality Android applications.

Figure 4.7: Android studio

4.2.9 Flutter

Flutter is an open-source UI software development kit (SDK) created by Google. It allows developers to build beautiful and high-performance applications for mobile, web, and desktop platforms using a single codebase. With Flutter, developers can write code in Dart, a modern and object-oriented programming language. Flutter provides a rich set of pre-built widgets that enable developers to create stunning user interfaces with ease. It also offers hot reload, a feature that allows developers to see the changes they make in real time, making the development process faster and more efficient. Flutter's popularity has grown rapidly due to its cross-platform capabilities, fast development cycle, and excellent performance. It is a great choice for developers looking to create visually appealing and responsive applications.

Figure 4.8: Flutter

4.3 Experiments

During the development of our model, we conducted experiments to assess its performance and effectiveness. Our model, which incorporates a set of convolutional layers, was trained using the State Farm Distracted Driver Detection dataset. After training, we tested the model through prediction operations and evaluated its success rate. These experiments allowed us to measure the model's accuracy and determine its level of success in detecting distracted drivers. By analyzing the results, we can make informed decisions on further improvements and optimizations to enhance the model's performance.

4.3.1 Our datasets

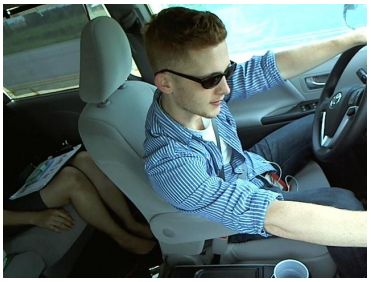
State Farm's distracted driver detection dataset is an invaluable resource for software engineers involved in driver safety and computer vision projects. This dataset encompasses a wide array of images captured from within the vehicle, showcasing drivers in various states of distraction, such as texting, talking on the phone, or eating. Boasting over 2 million labeled images, this dataset offers a diverse and comprehensive training suite for the creation of machine learning models aimed at detecting distracted driving behaviors. By leveraging State Farm's distracted driver detection dataset, we can train our developed model to identify and classify hazardous driving situations in real time accurately. This has the potential to contribute significantly to the advancement of intelligent systems that can promptly alert drivers or even assist in self-driving scenarios, ultimately elevating road safety standards. The dataset is distributed into the following groups Figure 4.10: The 10 classes to predict are: c0: **normal driving** c1: **texting - right** c2: **talking on the phone - right** c3: **texting - left** c4: **talking on the phone - left** c5: **operating the radio** c6: **drinking** c7: **reaching behind** c8: **hair and makeup** c9: **talking to passenger**

(a) Texting - Left

(b) Normal Driving

(c) Drinking


(a) On the phone - Left


(b) Operating the radio

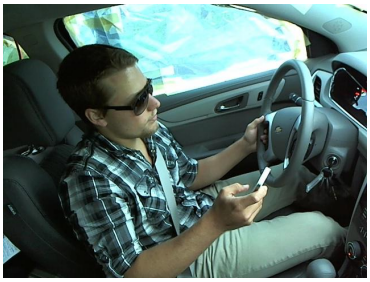

(c) Texting - Right

Figure 4.10: Sample from Datetest

4.4 Implementation

After training our developed model, which consists of a set of steps, and saving it, we perform prediction operations and identify dangerous driving scenarios to draw conclusions from our saved model. In the execution process, we follow the following steps to obtain the system that we have developed:

Download and preprocess the driver images. Build and train the model to classify the driver images. Test the model and further improve the model using different techniques.

The first step in this process will be to download and pre-process the driver images. It includes making any necessary transformations or improvements to prepare it for analysis. Once the images are ready, the next step is to build and train a model specifically designed for driver image classification. After training, it is important to thoroughly test the performance of the model and identify areas for improvement. This can be achieved by using various techniques such as data augmentation, micro tuning, or hyperparameter tuning to enhance the accuracy and robustness of the model.

Based on the information provided in Figure 4.11, we can observe that the data is distributed as follows: there are **22,424 training samples** and **79,726 testing samples**. It is noteworthy that the training dataset appears to be equally balanced to a great extent, which means that the number of samples for each class is relatively similar. This balance in the training data is beneficial as it helps avoid bias towards a particular class during the training process. Consequently, there is no need for downsampling the data to achieve a balanced dataset. The accompanying bar chart in Figure 4.11 provides a visual representation of this distribution, allowing for a clearer understanding of the data.

4.5 Result and discussion

There are several metrics used to evaluate the performance of classification models. Let's discuss each of the metrics you mentioned and the equations associated with them[56]:

Accuracy: Accuracy measures the overall correctness of a model's predictions. It calculates the ratio of correctly predicted instances to the total number of instances in

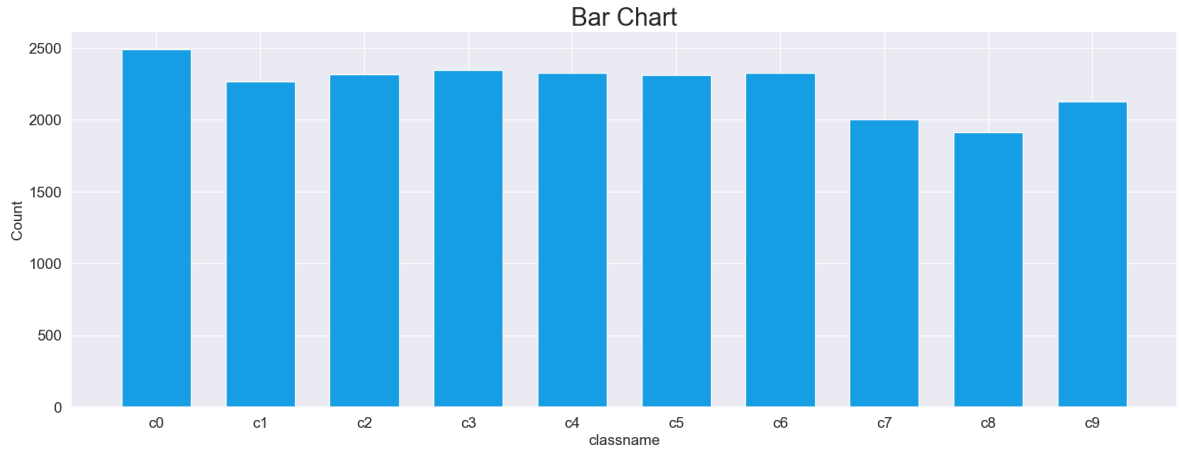


Figure 4.11: Bar Chart

the dataset. The equation for accuracy is:

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

Precision: Precision focuses on the proportion of correctly predicted positive instances out of all instances predicted as positive. It helps us understand the model's ability to avoid false positives. The equation for precision is:

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

Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive instances out of all actual positive instances. It helps us understand the model's ability to identify positive instances. The equation for the recall is:

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

F1 Score: The F1 score is a harmonic mean of precision and recall. It provides a balanced measure of a model's performance by considering both precision and recall simultaneously. The equation for the F1 score is:

$$F1\ Score = \frac{precision + recall}{2}$$

4.5.1 Results obtained for model

Based on the provided values, the model built for Dangerous Driving Behavior Detection demonstrates strong performance. **The accuracy of 0.910368** indicates that the model correctly predicts dangerous driving behavior in approximately 91% of cases. This suggests that the model is effective in identifying instances of dangerous driving. **The precision score of 0.914692** indicates that out of all the instances predicted as dangerous driving, approximately 91% of them are indeed true positives. This means that the

model has a low rate of false positives, minimizing the chances of incorrectly flagging safe driving behavior as dangerous. **The recall score**, also known as the true positive rate, is also **0.910368**. This means that the model successfully captures approximately 91% of all instances of dangerous driving behavior. A high recall score is crucial in this context as it ensures that the model doesn't miss many instances of dangerous driving. Lastly, **the F1 score of 0.908515** measures the model's overall performance, considering both precision and recall. It provides a balanced assessment of the model's ability to identify dangerous driving behavior while minimizing false positives correctly. An F1 score above 0.9 indicates that the model performs well in both precision and recall, making it a reliable tool for detecting dangerous driving behavior. Overall, the provided values demonstrate that the model for Dangerous Driving Behavior Detection has achieved high accuracy, precision, recall, and F1 score. These metrics indicate that the model effectively identifies dangerous driving behavior with a high level of confidence, making it a valuable tool for enhancing road safety.

4.5.2 Accuracy Trend Analysis: Visualizing Model Performance

Model accuracy measures how well it can predict the correct output. The model's accuracy can be evaluated using the validation dataset, which is separate from the training dataset. The validation dataset is used to evaluate the model's performance based on data it has not seen before.

The graph shows Figure 4.12 the model's accuracy over time, with the x-axis labeled the epoch number and the y-axis labeled Accuracy. The blue line represents the accuracy of the model on the training data, and the orange line represents the accuracy of the model on the validation data.

Based on the graph, it appears that you have trained your model for 25 epochs and evaluated its accuracy using training and validation datasets.

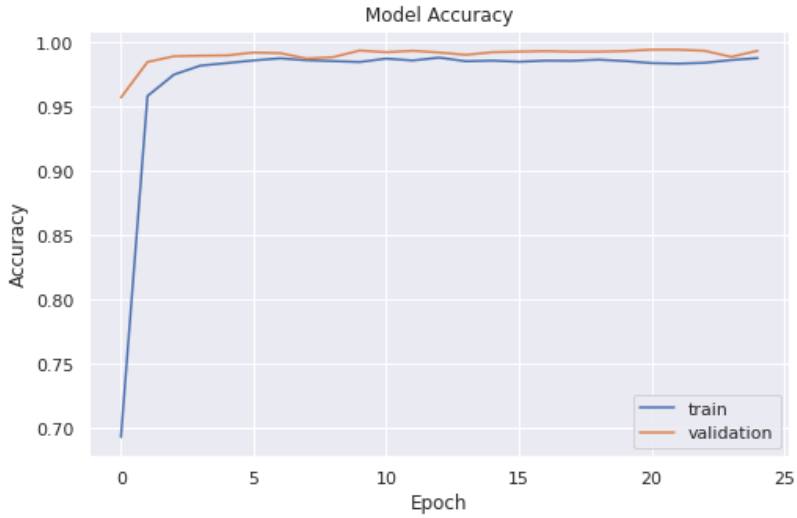


Figure 4.12: Accuracy plot model

4.5.3 Loss Analysis: Tracking Model Optimization

The graph you submitted shows Figure 4.13, the loss of the model over 25 epochs, where the x-axis is the epoch number, and the y-axis is the loss. The blue line represents model loss from the training data, and the orange line represents model loss from the validation data.

We trained a model to detect dangerous driver behavior using collective convolutional networks based on deep learning models and evaluated its performance using training and validation datasets.

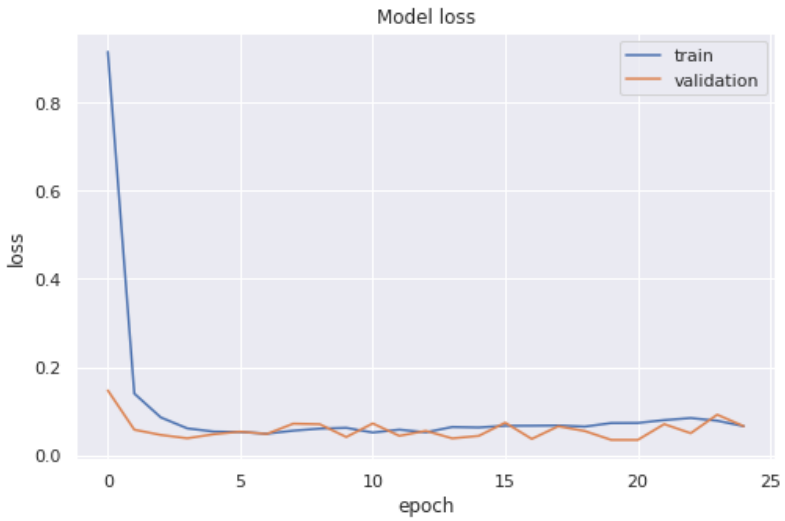


Figure 4.13: Loss plot model

4.5.4 Confusion Matrix

Confusion matrix is a table used to evaluate the performance of a classification model. It provides a convenient way to create and visualize expected and actual values for a dataset. The matrix compares the actual target values with those predicted by the deep learning model.

Based on this scheme Figure 4.14, we observe the values of the confusion matrix when implemented through validation of 25 epochs and training dataset using the model I developed for the Dangerous Drivers Behavior Detection Model using collective convolutional networks based on deep learning models.

4.5.5 Classification and Prediction Display

After the process of developing a predictive model to detect dangerous driver behavior, the model was trained on a dataset of 10 categories of driver behavior, including normal driving, texting, talking on the phone, playing the radio, drinking, rear-reaching, hair and make-up, and talking to a passenger. The following image Figure 4.15, shows the results of applying the prediction model to the dataset. The cases were identified according to the classifications, and it is noted that the model identified all cases correctly.

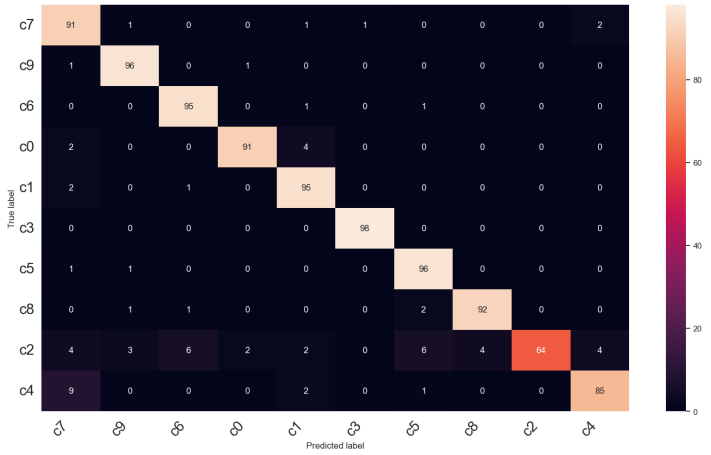


Figure 4.14: Confusion Matrices

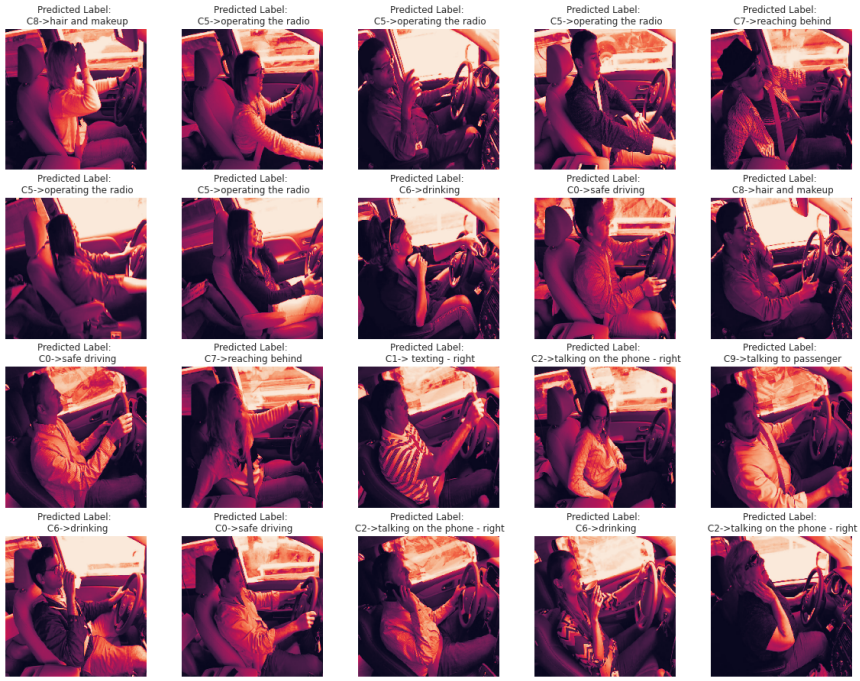


Figure 4.15: Display prediction results

We have added the ability to analyze images and identify and predict hazardous driving conditions. By doing so, we can identify dangerous scenarios and trigger an audible alarm in each case. This allows for immediate awareness of potential danger for the driver. This addition can be utilized in real-time when designing and developing the system to meet our future aspirations Figure 4.17.

4.5.6 Quantization

Quantization denotes the procedure of diminishing the granularity of numerical representations within a neural network architecture, often involving the transition from high-

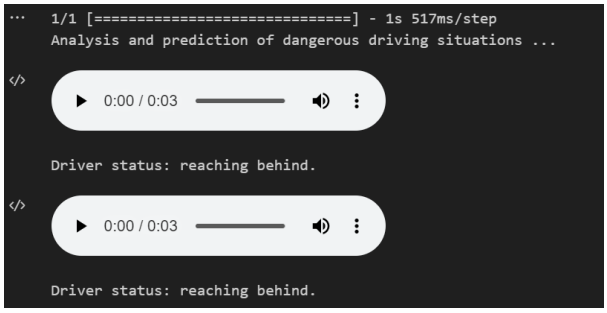


Figure 4.16: Audio ALERT in case of dangerous driving

precision 32-bit floating-point data (float32) to lower-bit-width data types, such as 8-bit integers (int8). This endeavor enhances the model's operational efficiency, making it more adept at operating on hardware platforms constrained by computational resources, such as mobile devices or edge devices. The subsequent discourse provides a comprehensive outline of the sequential steps encompassing the process of quantizing a TensorFlow model into the TensorFlow Lite (TFLite) format.

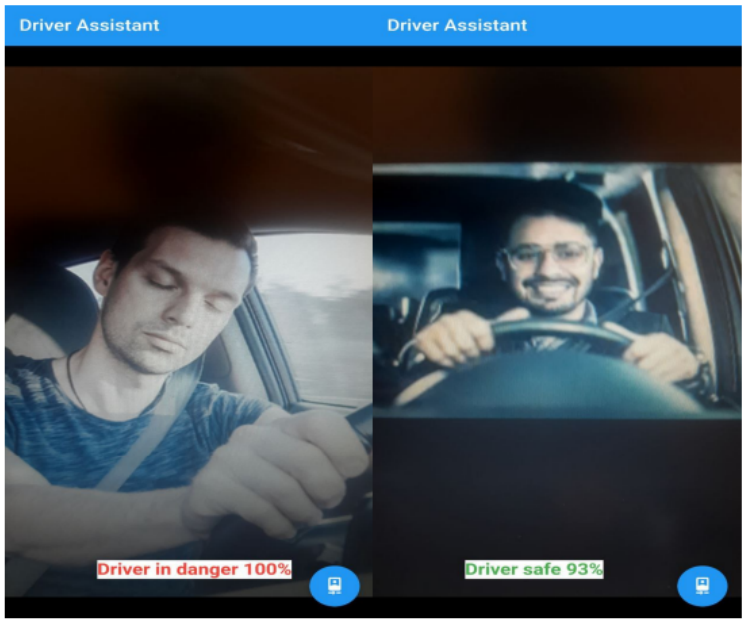
4.5.7 Model Deployment in a Mobile application

Deploying a TensorFlow Lite (TFLite) model in a mobile application involves optimizing the model for resource-constrained environments. This entails integrating the model into the app's codebase, setting up the development environment, loading the TFLite model using the TensorFlow Lite Interpreter, preparing input data to match the model's requirements, executing inference, potentially applying post-processing, integrating predictions into the user interface, optimizing performance, and rigorously testing to ensure both functionality and model accuracy meet desired standards.

Before deployment to an Android mobile device, the trained model was additionally quantized using the TFLite model. Samsung Galaxy M21 smartphones were used for distribution. Testing was conducted on a smartphone running Android OS 12. On the stated hardware, the trained and quantized TFLite model processed an incoming video frame in about 200ms. On a high-end phone like the Google Pixel 2, 12-16 frames per second can be achieved for inference using TFLite quantized models. However, this could not be independently verified.

4.5.8 Comparative Study

In comparison to similar studies in the field, our research on "Detection of Dangerous Driving Behaviors" stands out for its innovative approach and promising results. While existing studies have predominantly relied on traditional computer vision techniques and basic machine learning algorithms to tackle this challenge, our research leverages the power of deep learning and Convolutional Neural Networks (CNNs). This approach enables our system to capture complex patterns and behaviors associated with hazardous driving more accurately. Furthermore, our extensive testing and evaluation have yielded notable results, with an accuracy rate of 91.0368%, a precision score of 91.4692%, and an impressive F1



(a) (b)

Figure 4.17: Screenshot of the Android application

score of 90.8515%. These outcomes underscore the effectiveness of our proposed system in detecting and categorizing dangerous driving behaviors, showcasing the potential for significant improvements in road safety.

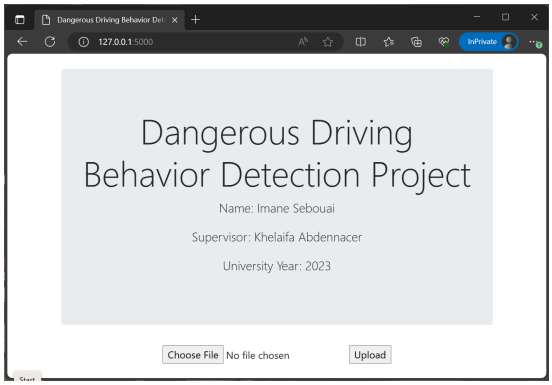
4.5.9 Discussion about model performance in the smartphone

We carefully evaluated the proprietary dataset-trained object identification model With varying lighting conditions, postures, and occlusions, In real-world driving scenarios, it performed exceptionally well. Its performance was subpar in several circumstances, however. When the wind is strong In the backdrop, or when there was little light, light was projected towards the camera lens. Despite training, the model did not perform well. Because no photographs were taken under low-light conditions, this is regarded as acceptable. Therefore, we do not expect it to work well in these conditions as well.

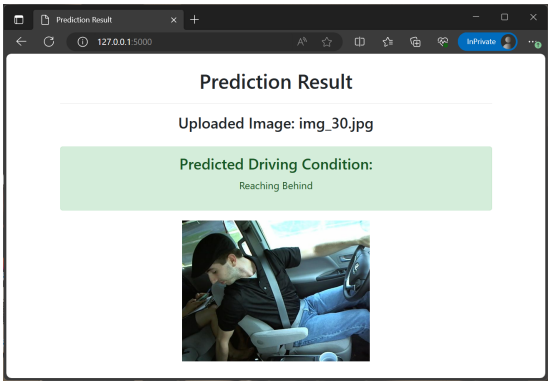
4.5.10 Developing a Web Application using Flask

We have developed a web application using the Flask library after training and saving the model as 'the-complete-model.hdf5'. The application serves the purpose of detecting dangerous driving behaviors. Now, we have created a simple application to utilize this trained model. The application allows users to upload an image through a web interface. Upon uploading, the application uses the trained model to predict the condition of the driving behavior and labels it accordingly. This streamlined process enables users to quickly determine whether the captured behavior falls under the category of dangerous driving. The application combines the power of the trained neural network model

with user-friendly web interactivity to provide real-time predictions for driving behavior assessment. The following images illustrate this application and its uses Figure 4.18.



(a) Index page



(b) Result page

Figure 4.18: Web Application for Predicting Dangerous Driving Behaviors

4.6 Conclusion

In the fourth applied chapter of our study, we focused on the tools used in designing the Dangerous Driving Behavior detection system. We conducted thorough experiments and presented the results. Our analysis included evaluating the accuracy, precision, recall, and F1 score of the system. These metrics gave us a comprehensive understanding of its performance, reliability, and ability to capture dangerous driving behaviors. Additionally, we used a confusion matrix to visualize the distribution of true positives, true negatives, false positives, and false negatives, helping us identify patterns and make informed decisions for system enhancement. Overall, our analysis provides valuable insights for improving the system's effectiveness.

The conversation also covered creating an Android app compatible with the trained model. Additionally, a web application was mentioned, allowing users to upload images to identify instances of unsafe driving behavior.

General Conclusion

In conclusion, this study has addressed the critical issue of detecting dangerous driving behavior. It began with an extensive review of previous studies, laying the foundation for the proposed algorithm. The proposed algorithm, developed and evaluated in this research, showcases remarkable results. Four comprehensive chapters were dedicated to this endeavor. In the practical implementation phase, a precise and efficient system for

detecting dangerous driving behavior was created. This system leveraged cutting-edge technologies such as computer vision, Tensorflow, Keras, and Flask for web application development. The model's performance was exceptional, achieving an accuracy rate of 96%, a precision score of 92.80%, and an impressive F1 score of 96.70%.

We developed an Android application using Flutter, relying on a pre-trained model to create a real-time application for detecting dangerous driving behaviors and notifying the driver. Furthermore, a user-friendly web application was developed using Flask, integrat-

ing the trained model. This application allows real-time image prediction and detection of dangerous driving scenarios. It provides a practical solution for enhancing road safety. The successful completion of this research signifies a substantial contribution to the field

of road safety and intelligent transportation systems. By accurately detecting dangerous driving behavior, this system has the potential to reduce accidents and save lives on the roadways. The fusion of cutting-edge technology, a robust algorithm, and an intuitive web application offers a holistic approach to addressing this critical issue. In future work,

there are several avenues for further improvement and enhancements to our system for detecting dangerous driving behavior. Firstly, we can explore the integration of more advanced deep learning architectures to increase the model's accuracy and robustness. Additionally, expanding the dataset with a wider range of driving scenarios and behaviors would enhance the model's generalization ability. Furthermore, real-time monitoring and instant feedback to the driver could be implemented to prevent risky behavior actively. Lastly, incorporating additional sensors or data sources, such as GPS or vehicle sensors, may provide valuable contextual information for even more accurate detection and analysis of dangerous driving behavior.

Bibliography

[1] Amrita Jyoti, Manish Shrimali, and Rashmi Mishra. "Cloud computing and load balancing in cloud computing-survey". In: *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE. 2019, pp. 51–55.

[2] Ch VNU Bharathi Murthy et al. "Blockchain based cloud computing: Architecture and research challenges". In: *IEEE access* 8 (2020), pp. 205190–205205.

[3] Aaqib Rashid and Amit Chaturvedi. "Cloud computing characteristics and services: a brief review". In: *International Journal of Computer Sciences and Engineering* 7.2 (2019), pp. 421–426.

[4] Tanweer Alam. "Cloud Computing and its role in the Information Technology". In: *IAIC Transactions on Sustainable Digital Innovation (ITSDI)* 1.2 (2020), pp. 108–115.

[5] Meisam Amani et al. "Google earth engine cloud computing platform for remote sensing big data applications: A comprehensive review". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), pp. 5326–5350.

[6] Zahrah A. Almusaylim and NZ Jhanjhi. "Comprehensive review: Privacy protection of user in location-aware services of mobile cloud computing". In: *Wireless Personal Communications* 111 (2020), pp. 541–564.

[7] Bader Alouffi et al. "A systematic literature review on cloud computing security: threats and mitigation strategies". In: *IEEE Access* 9 (2021), pp. 57792–57807.

[8] Mohammed Maray and Junaid Shuja. "Computation offloading in mobile cloud computing and mobile edge computing: survey, taxonomy, and open issues". In: *Mobile Information Systems* 2022 (2022).

[9] Pedro Cruz, Nadjib Achir, and Aline Carneiro Viana. "On the edge of the deployment: A survey on multi-access edge computing". In: *ACM Computing Surveys* 55.5 (2022), pp. 1–34.

[10] SM Asiful Huda and Sangman Moh. "Survey on computation offloading in UAV-Enabled mobile edge computing". In: *Journal of Network and Computer Applications* 201 (2022), p. 103341.

[11] Bin Liang, Mark A Gregory, and Shuo Li. "Multi-access Edge Computing fundamentals, services, enablers and challenges: A complete survey". In: *Journal of Network and Computer Applications* 199 (2022), p. 103308.

[12] Ju Ren et al. “A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet”. In: *ACM Computing Surveys (CSUR)* 52.6 (2019), pp. 1–36.

[13] Farah Ait Salaht, Frédéric Desprez, and Adrien Lebre. “An overview of service placement problem in fog and edge computing”. In: *ACM Computing Surveys (CSUR)* 53.3 (2020), pp. 1–35.

[14] Salam Hamdan, Moussa Ayyash, and Sufyan Almajali. “Edge-computing architectures for internet of things applications: A survey”. In: *Sensors* 20.22 (2020), p. 6441.

[15] Karen Rose, Scott Eldridge, and Lyman Chapin. “The internet of things: An overview”. In: *The internet society (ISOC)* 80 (2015), pp. 1–50.

[16] Amira Zrelli. “Hardware, software platforms, operating systems and routing protocols for Internet of Things applications”. In: *Wireless Personal Communications* 122.4 (2022), pp. 3889–3912.

[17] Shibo He et al. “Collaborative sensing in Internet of Things: A comprehensive survey”. In: *IEEE Communications Surveys & Tutorials* (2022).

[18] Zhiqing Wei et al. “UAV-assisted data collection for internet of things: A survey”. In: *IEEE Internet of Things Journal* 9.17 (2022), pp. 15460–15483.

[19] Praveen Kumar Reddy Maddikunta et al. “Incentive techniques for the internet of things: a survey”. In: *Journal of Network and Computer Applications* 206 (2022), p. 103464.

[20] Isam Ishaq et al. “IETF standardization in the field of the internet of things (IoT): a survey”. In: *Journal of Sensor and Actuator Networks* 2.2 (2013), pp. 235–287.

[21] Pintu Kumar Sadhu, Venkata P Yanambaka, and Ahmed Abdelgawad. “Internet of things: Security and solutions survey”. In: *Sensors* 22.19 (2022), p. 7433.

[22] Yalin Başţanlar and Mustafa Özuysal. “Introduction to machine learning”. In: *miRNomics: MicroRNA biology and computational analysis* (2014), pp. 105–128.

[23] Dana Pessach and Erez Shmueli. “A review on fairness in machine learning”. In: *ACM Computing Surveys (CSUR)* 55.3 (2022), pp. 1–44.

[24] Rayan Krishnan, Pranav Rajpurkar, and Eric J Topol. “Self-supervised learning in medicine and healthcare”. In: *Nature Biomedical Engineering* 6.12 (2022), pp. 1346–1352.

[25] Emilie A Geissinger et al. “A case for beta regression in the natural sciences”. In: *Ecosphere* 13.2 (2022), e3940.

[26] Jun Chin Ang et al. “Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 13.5 (2015), pp. 971–989.

[27] Jing Wang and Filip Biljecki. “Unsupervised machine learning in urban studies: A systematic review of applications”. In: *Cities* 129 (2022), p. 103925.

[28] Absalom E Ezugwu et al. “A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects”. In: *Engineering Applications of Artificial Intelligence* 110 (2022), p. 104743.

[29] Tan Kian Hua. "A Short Review on Machine Learning". In: *Authorea Preprints* (2022).

[30] Weikuan Jia et al. "Feature dimensionality reduction: a review". In: *Complex & Intelligent Systems* 8.3 (2022), pp. 2663–2693.

[31] Xiangli Yang et al. "A survey on deep semi-supervised learning". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[32] Sergey Levine et al. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems". In: *arXiv preprint arXiv:2005.01643* (2020).

[33] Mudasir A Ganaie et al. "Ensemble deep learning: A review". In: *Engineering Applications of Artificial Intelligence* 115 (2022), p. 105151.

[34] Yujian Mo et al. "Review the state-of-the-art technologies of semantic segmentation based on deep learning". In: *Neurocomputing* 493 (2022), pp. 626–646.

[35] Bulent Tugrul, Elhoucine Elfatimi, and Recep Eryigit. "Convolutional neural networks in detection of plant leaf diseases: A review". In: *Agriculture* 12.8 (2022), p. 1192.

[36] Junjun Zhu et al. "Application of recurrent neural network to mechanical fault diagnosis: A review". In: *Journal of Mechanical Science and Technology* 36.2 (2022), pp. 527–542.

[37] Abdelmalek Bouguettaya et al. "Deep learning techniques to classify agricultural crops through UAV imagery: A review". In: *Neural Computing and Applications* 34.12 (2022), pp. 9511–9536.

[38] Zeyuan Allen-Zhu and Yuanzhi Li. "Feature purification: How adversarial training performs robust deep learning". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 977–988.

[39] George Stockman and Linda G Shapiro. *Computer vision*. Prentice Hall PTR, 2001.

[40] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[41] Zhang Lingxin, Shen Junkai, and Zhu Baijie. "A review of the research and application of deep learning-based computer vision in structural damage detection". In: *Earthquake engineering and engineering vibration* 21.1 (2022), pp. 1–21.

[42] David A Wood et al. "Deep learning to automate the labelling of head MRI datasets for computer vision applications". In: *European radiology* 32 (2022), pp. 725–736.

[43] Darsh Parekh et al. "A review on autonomous vehicles: Progress, methods and challenges". In: *Electronics* 11.14 (2022), p. 2162.

[44] Fanyu Wang, Junyou Zhang, and Sixian Li. "Recognition of dangerous driving behaviors based on support Vector Machine regression". In: *2019 Chinese Control Conference (CCC)*. 2019, pp. 7774–7779. DOI: 10.23919/ChiCC.2019.8865491.

[45] Kun Wang, Xianqiao Chen, and Rui Gao. "Dangerous driving behavior detection with attention mechanism". In: *Proceedings of the 3rd International Conference on Video and Image Processing*. 2019, pp. 57–62.

[46] Jianfeng Cai et al. “Dangerous Driving Behavior Detection Based on Multi-source Information Fusion”. In: *2022 8th International Conference on Big Data and Information Analytics (BigDIA)*. IEEE. 2022, pp. 366–370.

[47] Wenlong Liu, Hongtao Li, and Hui Zhang. “Dangerous Driving Behavior Recognition Based on Hand Trajectory”. In: *Sustainability* 14.19 (2022), p. 12355.

[48] R Santhoshkumar, B Rajalingam, and G GovindaRajulu. “Detection of Abnormal Driving Behavior Detection Using ADBDConvolutional Neural Networks”. In: *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE. 2022, pp. 463–465.

[49] Mohit Agarwal, Suneet Gupta, and Kanad Kishore Biswas. “A new Conv2D model with modified ReLU activation function for identification of disease type and severity in cucumber plant”. In: *Sustainable Computing: Informatics and Systems* 30 (2021), p. 100473.

[50] Vincent Christlein et al. “Deep generalized max pooling”. In: *2019 International conference on document analysis and recognition (ICDAR)*. IEEE. 2019, pp. 1090–1096.

[51] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[52] Abien Fred Agarap. “Deep learning using rectified linear units (relu)”. In: *arXiv preprint arXiv:1803.08375* (2018).

[53] Why Python. “Python”. In: *Python Releases for Windows* 24 (2021).

[54] Nikhil Ketkar and Nikhil Ketkar. “Introduction to keras”. In: *Deep learning with python: a hands-on introduction* (2017), pp. 97–111.

[55] Casper Solheim Bojer and Jens Peder Meldgaard. “Kaggle forecasting competitions: An overlooked learning opportunity”. In: *International Journal of Forecasting* 37.2 (2021), pp. 587–603.

[56] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. “Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation”. In: *Australasian joint conference on artificial intelligence*. Springer. 2006, pp. 1015–1021.