Democratic and Popular Algerian Republic
Ministry of Higher Education and Scientific Research
Faculty of Sciences Yahia Fares University of Medea
Departement of Mathematics and Computer Science

Final Master Project

Sector : Computer science

Speciality : Systems Engineering and Web Technology

Early Recognition of Infant Discomfort Through Real Time Analysis of cries- A Digital Twin Based Approach

released by :
FARADJI Hamza
CHOUIH Brahim

Supervised by :
Dr.Rahim Messaoud
Co-Supervised by :
Mr.Boudjema Ali

Dedication

To Allah Almighty, who has illuminated my path and guided me with wisdom and clarity. To Prophet Mohammad, the epitome of inspiration, wisdom, and encouragement towards patience.

To my beloved parents, your unconditional love, sacrifices, and unwavering belief in my potential have paved the way for my success. Your presence and support have shaped me into the person I am today.

To my cherished family, each moment shared with you has enriched my life with joy, laughter, and strength. Your enduring love and unwavering support have been my rock.

To my dear friends, including Hamza, your companionship has been a constant source of joy and solidarity. Together, we've created memories that I cherish deeply.

CHOUIH Brahim

Dedication

In the name of Allah, the Most Gracious, the Most Merciful.

To my dear parents, your endless love, support, and sacrifices have been the foundation of all my achievements. Your belief in me has been my greatest strength and motivation.

To my treasured family, your love, encouragement, and unwavering support have been a source of constant strength and joy in my life. Every moment with you has been a blessing.

To my friends, especially Brahim, your companionship and support have made this journey unforgettable. The shared experiences and memories have enriched my life and made this project possible.

My heartfelt gratitude to each and every one of you. May Allah bless you all with happiness and success.

FERRADJI hamza

Acknowledgment

First of all, we wish to express our gratitude to Almighty God for giving us the strength, patience, and perseverance necessary to carry out this memory. Without His blessing and guidance, this work would not have been possible.

We would like to express our sincere gratitude to our advisor, ***Mr. RAHIM Messaoud****, for his wise advice, constant support, and encouragement throughout this work. His availability and valuable suggestions were decisive for the success of this project.*

We would also like to extend our thanks to our Co-Supervisor, ***Mr. BOUDJEMA Ali****, for his support and guidance during this project.*

We warmly thank the members of the jury who were kind enough to make an appreciation of this work and to participate in the jury of defense.

We are deeply grateful to our colleagues and friends for their encouragement and moral support, especially ***Fathia*** *and* ***Abderrahmen*** *and* ***Abdelmoutaleb*** *and* ***Hossem*** *. Their camaraderie was an invaluable source of motivation and comfort.*

Finally, we would like to extend our sincere thanks to our families, especially our parents, for their unconditional love, patience, and unwavering support throughout our studies. Their trust in us was a real inspiration.

To all, a big thank you.

Abstract

The ability to accurately interpret an infant's cries is crucial for providing appropriate and timely care. However, many caregivers find it challenging to understand the needs of a crying infant, leading to misinterpretations and delayed interventions. This project aims to address this issue by developing a system based on digital twin technology to enhance infant care through accurate cry detection and classification.

The concept of the digital twin is the foundation of this project, where a virtual model replicates the physical entity, in this case, the infant. By integrating Internet of Things (IoT) technologies, we establish real-time connectivity between the physical baby and its digital representation. This digital twin captures and processes data from sensors attached to the infant, providing a comprehensive and dynamic model of the baby's condition.

The digital twin serves as the core of the system, facilitating continuous monitoring and data collection. We employ advanced machine learning (ML) and deep learning (DL) techniques to analyze the collected data. Two primary models were developed within the system : one for cry detection and another for classifying the reasons for crying. These models are trained using a variety of neural network architectures, including Multi-Layer Perceptrons (MLP) and Long Short-Term Memory (LSTM) networks combined with Convolutional Neural Networks (CNN). Additionally, genetic algorithms are used to optimize the hyperparameters of these models, ensuring high accuracy and performance.

The results demonstrate the system's effectiveness through simulations and real-world tests, significantly improving the accuracy of cry interpretation and, consequently, infant care. The MLP model achieves an impressive accuracy rate of 96.67%, while the LSTM-CNN model attains a perfect accuracy rate of 100%.

Keywords : Digital Twin, Infant Care, Cry Detection, IoT, Machine Learning, Deep Learning, Neural Networks, Genetic Algorithms, MLP, LSTM-CNN.

Résumé

La capacité d'interpréter correctement les pleurs d'un nourrisson est essentielle pour fournir des soins appropriés et rapides. Cependant, de nombreux soignants trouvent difficile de comprendre les besoins d'un bébé qui pleure, ce qui conduit à des interprétations erronées et à des interventions tardives. Ce projet vise à résoudre ce problème en développant un système basé sur la technologie des jumeaux numériques pour améliorer les soins aux nourrissons grâce à une détection et une classification précises des pleurs.

Le concept de jumeau numérique est au cœur de ce projet, où un modèle virtuel réplique l'entité physique, en l'occurrence le nourrisson. En intégrant les technologies de l'Internet des objets (IoT), nous établissons une connectivité en temps réel entre le bébé physique et sa représentation numérique. Ce jumeau numérique capture et traite les données provenant des capteurs attachés au nourrisson, fournissant un modèle complet et dynamique de l'état du bébé.

Le jumeau numérique sert de noyau au système, facilitant la surveillance continue et la collecte de données. Nous employons des techniques avancées d'apprentissage automatique (ML) et d'apprentissage profond (DL) pour analyser les données collectées. Deux modèles principaux ont été développés au sein du système : l'un pour la détection des pleurs et l'autre pour la classification des raisons des pleurs. Ces modèles sont entraînés en utilisant diverses architectures de réseaux neuronaux, y compris les Perceptrons Multicouches (MLP) et les Réseaux de Mémoire à Long Court Terme (LSTM) combinés avec des Réseaux de Neurones Convolutionnels (CNN). De plus, des algorithmes génétiques sont utilisés pour optimiser les hyperparamètres de ces modèles, assurant ainsi une haute précision et performance.

Les résultats démontrent l'efficacité du système à travers des simulations et des tests réels, améliorant significativement la précision de l'interprétation des pleurs et, par conséquent, les soins aux nourrissons. Le modèle MLP atteint un taux de précision impressionnant de 96,67 %, tandis que le modèle LSTM-CNN atteint un taux de précision parfait de 100 %.

Mots-clés : Jumeau Numérique, Soins aux Nourrissons, Détection des Pleurs, IoT, Apprentissage Automatique, Apprentissage Profond, Réseaux Neuronaux, Algorithmes Génétiques, MLP, LSTM-CNN.

الملخص

تُعد القدرة على تفسير بكاء الرضيع بشكل دقيق أمراً بالغ الأهمية لتوفير الرعاية المناسبة وفي الوقت المناسب. ومع ذلك، يجد العديد من مقدمي الرعاية صعوبة في فهم احتياجات الطفل الباكي، مما يؤدي إلى تفسيرات خاطئة وتأخير التدخلات. يهدف هذا المشروع إلى معالجة هذه المشكلة من خلال تطوير نظام يعتمد على التوأمة الرقمية لتحسين رعاية الرضع عبر الكشف الدقيق عن أسباب بكائهم وتصنيفها.
يعد مفهوم التوأمة الرقمية أساس هذا المشروع، حيث يتم إنشاء نموذج افتراضي يعكس الكيان الفيزيائي، وهو الطفل في هذه الحالة. من خلال دمج تقنيات إنترنت الأشياء ،(IoT) نقوم بإنشاء اتصال في الوقت الفعلي بين الرضيع الفعلي وتمثيله الرقمي. يلتقط هذا التوأم الرقمي البيانات ويعالجها من المستشعرات المتصلة بالرضيع، مما يوفر نموذجاً شاملاً وديناميكياً لحالة الطفل.
يعمل التوأم الرقمي كنواة للنظام، مما يسهل المراقبة المستمرة وجمع البيانات. نستخدم تقنيات التعلم الآلي (ML) والتعلم العميق (DL) المتقدمة لتحليل البيانات المجمعة. تم تطوير نموذجين رئيسيين داخل النظام : أحدهما للكشف عن البكاء والآخر لتصنيف أسباب البكاء. يتم تدريب هذه النماذج باستخدام مجموعة متنوعة من بنى الشبكات العصبية، بما في ذلك الشبكات العصبية متعددة الطبقات (MLP) والشبكات العصبية التلافيفية (CNN) المدمجة مع شبكات الذاكرة طويلة وقصيرة الأجل .(LSTM) بالإضافة إلى ذلك، تُستخدم الخوارزميات الجينية لتحسين معايير هذه النماذج، مما يضمن دقة وأداء عاليين.
أظهرت النتائج فعالية النظام من خلال المحاكاة والاختبارات الواقعية، مما يُظهر قدرته على تحسين دقة تفسير البكاء بشكل كبير، وبالتالي رعاية الرضع. أظهر النموذج الأول للكشف عن البكاء باستخدام الشبكات العصبية متعددة الطبقات (MLP) دقة مذهلة بلغت 96.67%. أما النموذج الثاني، فقد تم تطويره لتصنيف أسباب البكاء باستخدام شبكات الذاكرة طويلة وقصيرة الأجل (LSTM) المدمجة مع الشبكات العصبية التلافيفية ،(CNN) حيث حقق دقة مثالية بلغت 100%.

الكلمات المفتاحية : التوأم الرقمي، رعاية الرضع، كشف البكاء، إنترنت الأشياء، التعلم الآلي، التعلم العميق، الشبكات العصبية، الخوارزميات الجينية، LSTM-CNN. MLP،

Contents

Generale Introduction **xiv**

1 Digital twins **1**
1.1 Introduction . . . 2
1.2 Digital Twins . . . 2
1.3 Internet Of Things IOT . . . 3
1.3.1 IoT technologies . . . 3
1.3.2 IoT Architecture . . . 3
1.3.3 IoT Applications . . . 4
1.3.4 IoT characteristics . . . 5
1.3.5 Advantages and Disadvantages of the IoT . . . 6
1.3.5.1 IoT advantages . . . 6
1.4 Digital twin types . . . 6
1.5 The Digital Twin Architecture . . . 7
1.6 Digital twin technologies . . . 8
1.7 Digital twins use cases and applications . . . 9
1.8 Digital twin benefits . . . 9
1.9 Technologies and tools used for digital twins . . . 10
1.9.1 Eclips Ditto . . . 10
1.9.2 The cloud tools . . . 11
1.9.2.1 Azur . . . 11
1.10 Conclusion . . . 11

2 Deep Learning **12**
2.1 Introduction . . . 13
2.2 Machine learning (ML) . . . 13
2.2.1 Supervised Learning . . . 13
2.2.2 Unsupervised Learning . . . 13
2.2.3 Semi-supervised Learning . . . 13
2.3 Deep Learning (DL) . . . 14
2.3.1 How Deep Learning Works . . . 14
2.3.2 Deep learning applications . . . 14
2.3.3 Deep Learning Types . . . 15
2.3.3.1 Neural Networks . . . 15
2.3.3.2 Convolutional Neural Networks . . . 16
2.3.3.3 Feed Forward Neural Network . . . 16
2.3.3.4 Recurrent Neural Networks . . . 17
2.3.3.5 Auto Encoders . . . 17
2.3.3.6 Recursive Neural Network . . . 18
2.3.3.7 Deep Belief Networks (DBN) . . . 18
2.4 Popular Deep learning Models . . . 18
2.4.1 Long Short-Term Memory (LSTM) Network . . . 18
2.4.2 Generative adversarial networks (GANs) . . . 19

2.4.2.1 Recurrent neural networks (RNNs) . . . 19
2.4.2.2 Radial basis function networks (RBFNs) . . . 20
2.4.2.3 Feedforward Neural Networks (FNN) . . . 20
2.4.2.4 Multilayer Perceptrons (MLPs) . . . 20
2.4.2.5 Convolutional Neural Network (CNN) . . . 21
2.5 Difference between Machine Learning and Deep Learning . . . 22
2.6 Challenges in Deep Learning . . . 22
2.7 Conclusion . . . 23

3 Conception **24**
3.1 Introduction . . . 25
3.2 Overall Architecture . . . 25
3.2.1 Digital Twin . . . 26
3.2.2 Synchronization services . . . 29
3.2.3 Artificial Intelligence Services . . . 29
3.3 Baby supervises . . . 31
3.3.1 Create a Digital twin . . . 31
3.3.2 Data processing . . . 32
3.4 Cry detection and classification . . . 32
3.4.1 Baby cry classification . . . 32
3.4.1.1 MLP Model . . . 33
3.4.1.2 CNN-LSTM . . . 35
3.4.2 Baby Cry Detection . . . 36
3.4.2.1 MLP model . . . 36
3.4.2.2 CNN-LSTM Model . . . 37
3.4.3 Hyperparameter Optimization with Genetic Algorithms . . . 39
3.4.3.1 Genetic Algorithm Implementation for Neural Network Optimization . . . 40
3.4.3.2 Optimal Neural Network Architectures Identified by Genetic Algorithms . . . 40
3.5 Conclusion . . . 41

4 Implementation **42**
4.1 Introduction . . . 43
4.2 Development Environment . . . 43
4.2.1 Google Colab . . . 43
4.2.2 Python . . . 43
4.2.3 TensorFlow . . . 43
4.2.4 Scikit-learn . . . 43
4.2.5 Imbalanced-learn (imblearn) . . . 44
4.2.6 DEAP (Distributed Evolutionary Algorithms in Python) . . . 44
4.2.7 Django . . . 44
4.2.8 Node-RED . . . 44
4.2.9 Flutter . . . 44
4.3 Implementation . . . 45
4.3.1 Baby digital twin . . . 45
4.3.2 Model Development . . . 45
4.3.2.1 Crying Detection Model . . . 46
4.3.2.2 Cry classification Model . . . 46
4.3.3 Results . . . 47

4.3.3.1 Result cause cry 47
4.3.3.2 Result detect cry 49
4.3.4 Performance Evaluation 51
4.3.5 Training Time Comparison for Genetic Algorithm Optimized Models . . . 51
4.3.6 Implementation of AI services 52
4.4 Realization Twining Services 52
4.5 Mobile App 53
4.5.1 Login Interface 53
4.5.2 Add Baby Interface 53
4.5.3 Child's Status Interface 53
4.5.4 Notifications Interface 53
4.5.5 Signature Correction by Caregivers Interface 54
4.6 Conclusion 57

List of Figures

1.1 Digital twin. 2
1.2 IoT Architecture 4
1.3 Digital twin types [8]. 7
1.4 Technologies of Digital Twins [14]. 8
1.5 Digital twin. 10

2.1 Represents the Relation between Artificial Intelligence, Machine Learning and Deep Learning 14
2.2 Simple Neural Network. 15
2.3 The Architecture of CNN [67] 16
2.4 Typical unfolded RNN diagram [68] 17
2.5 The Architecture of Auto Encoders [69] 18

3.1 Overall Architectur 25
3.2 Use Case Diagram 26
3.3 Class Diagram of Ditto [72]. 27
3.4 Block Diagram 28
3.5 Sequence Diagram for Baby Cry Detection API Interaction 30
3.6 Sequence Diagram for Baby Cry Classification 30
3.7 Create DT Sequence Diagram 31
3.8 Baby Supervision Sequence Diagram 32
3.9 Befor Balancing Data 33
3.10 After Balancing Data 33
3.11 Mlp Architucture for cry classification 34
3.12 CNN-LSTM Architucture for cry classification 35
3.13 Mlp Architucture for detect cry 37
3.14 LSTM+CNN Architucture for detect cry 38
3.15 Genetic Algorithms Process [79] 39

4.1 Eclipse Ditto Platform 45
4.2 MLP model for cry detection 46
4.3 CNN with LSTM Model for Detection Cry 46
4.4 MLP Model for Cry classification 46
4.5 CNN with LSTM Model for Cry classification 46
4.6 Accuracy and Loss of the MLP Model for Cry classification 47
4.7 Matrix confision for model MLP for Cry classification 48
4.8 Accuracy and Loss of The LSTM with CNN Model for Cry classification 48
4.9 Matrix confision for model CNN with LSTM for cause cry 49
4.10 Accuracy and Loss of the MLP Model for Detection cry 49
4.11 Matrix confision for model MLP for detect cry 50
4.12 Accuracy and loss of the CNN with LSTM Model for Detection Cry 50
4.13 Matrix confision for model CNN with LSTM for detect cry 51
4.14 Visualizing Sensor Data from a Baby Crib using Node-RED 53
4.15 Login Interface 54

4.16 Add Baby Interface 54
4.17 Child's Status Interface 55
4.18 Child's Status Interface After Crying 55
4.19 Signature Correction by Caregivers Interface 56
4.20 Notifications Interface 56

List of Tables

4.1 The Performance of suggested models 51
4.2 Training duration for each model using genetic algorithm 52

List of Acronyms

AI : Artificial Inteligence
ANN : Artificial Neural Networks
API : Application Programming Interface
Bi-LSTM : Bidirectional LSTM
CNN : Convolutional Neural Network
CSS : Cascading Style Sheets
DL : Deep Learning
DNN : Deep Neural Network
GAP : Global Average Pooling
GPU : Graphics Processing Unit
HAR : Human Activity Recognition
HTML : Hypertext Markup Language
HTTP : Hypertext Transfer Protocol
LSTM : Long Short Term Memory
ML : Machine Learning
MSE : Mean Squared Error
NLP : Natural Language Processing
PAMAP2 : Physical Activity Monitoring of Older Adults, version 2
PDO : Php Data Objects
RAM : Random Access Memory
ReLU : Rectified Linear Unit
SNN : Simulated Neural Network
REST API : Representational State Transfer Application Programming Interface
RNN : Recurrenl Neural Network
SGD : Stochastic Gradient Descent
SQL : Structured Query Language
SVM : Support vector machine
WISDM : Wireless Sensor Data Mining

Generale Introduction

In the realm of modern technological advancements the intersection of digital twins and artificial intelligence (AI) has given rise to innovative solutions aimed at improving various aspects of human life. One notable application of this integration is the development of systems for monitoring and analyzing infant behavior. This specifically involves the detection and classification of baby cries. This Project delves into the intricate architecture and methodologies employed in creating a digital twin framework designed to supervise infants. Special focus is placed on recognizing and interpreting their cries.

Understanding and responding to an infant's cries is a significant challenge for both nurses and parents. Differentiating between cries indicating hunger, discomfort pain, or other needs is often difficult. This leads to delays in appropriate intervention. This challenge is compounded by the lack of immediate and accurate feedback. This feedback is crucial for timely and effective care. The inability to quickly ascertain the cause of a baby's cry can result in prolonged distress for the infant. It also heightens anxiety for caregivers•

Through a digital twin, caregivers can remotely monitor their babies, gaining insights into baby comfort, sleep quality, and vital signs. It facilitates real-time data collection from various sensors embedded in the bed, such as Temperature sensors, sound sensors, pressure sensors, heart rate monitors, CO2 sensors. These sensors provide valuable information about the patient’s condition and the bed’s performance. Due to these benefits of the digital twin, this study treats the problematic issue of how designing a digital twin-based solution that allows the control and supervision of babies.

This problem can be subdivided into the following questions :

- How to design a digital twin model that can replicate the Baby’s health state ?
- How can supervision and control functionalities be provided for the caregivers in a secure and convivial manner ?
- How can the physical system be connected to the digital twin ?
- How can we provide analytic services concerning the meaning of baby's cry ?

The goal of our project Develop a digital twin-based solution to supervise and monitoring babies, With AI services for the detection and classification of babies’ cry The remainder of the document is structured as follow :

- Chapter 1: Introduction to Digital Twins This chapter provides an overview of digital twin technology, its integration with IoT, and its applications in baby healthcare. It explains the concept, functionalities, types, and elements of digital twins. It also highlights how digital twins, updated with real-time data and using simulations, machine learning, and reasoning, can support decision-making and improve baby monitoring.
- Chapter 2: Literature Review on Deep Learning and Machine Learning This chapter reviews deep learning and machine learning technologies, focusing on their applications in infant monitoring. It covers types of learning (supervised, unsupervised, reinforcement) and neural networks (RNNs, CNNs). It discusses how these technologies address complex problems like pattern recognition and signal processing, and their potential in healthcare.

- Chapter 3: System Concept and Design This chapter details the architecture and functionalities of the proposed baby monitoring system. It describes the system's components : digital twin, AI services, and the end-user application. It covers sensor integration for monitoring vital signs and movements, and the use of AI models (MLP, LSTM+CNN) for cry classification, aiming to identify specific needs like hunger or discomfort.
- Chapter 4: Sensor Integration and Data Collection This chapter focuses on integrating sensors for data collection in the baby monitoring system. It explains the roles of various sensors (temperature, sound, pressure, heart rate, CO2) and their integration for real-time data processing. It also discusses using Node-RED for managing data flow and integrating deep learning models with sensor data for accurate cry classification and monitoring.

1 Digital twins

Content

1.1 Introduction 3
1.2 Digital Twins 3
1.3 Internet Of Things IOT 4
1.4 Digital twin types 7
1.5 The Digital Twin Architecture 8
1.6 Digital twin technologies 9
1.7 Digital twins use cases and applications 10
1.8 Digital twin benefits 10
1.9 Technologies and tools used for digital twins 11
1.10 Conclusion 12

1.1 Introduction

A digital twin is a virtual representation of an object or system designed to reflect a physical object accurately. Digital twins integrated with IoT provide valuable insights, optimize operations, and support decision-making across various domains, is updated from real-time data and uses simulation, machine learning and reasoning to help make decisions.

The aim of this chapter is to present a comprehensive view of "digital twins" technology. This involves examining the fundamental concept behind this technology and delving into its functionalities, constituent elements, diverse types, practical applications, and the significant transformative influence it exerts on baby healthcare. Furthermore, as we delve into the capabilities of digital twin technology, this chapter also sheds light on the concept of the Internet of Things (IoT) .

1.2 Digital Twins

Digital twins are precise virtual representations of physical objects, processes or systems. They are created by synchronizing real-time data between the real object and its virtual version, thus allowing to simulate, analyze and predict the behavior of the physical object. These virtual models are used in various fields such as industry, health, product design, and resource management to improve understanding, decision-making and operational efficiency. Digital twins integrate geographic and temporal data to reflect changes over time, providing a comprehensive view of past, current and future states of the objects or systems they represent [1][2]Ḋigital twins offer a cost-efficient method for comprehending the performance attributes of a real-world physical entity prior to investing in tangible prototypes. This significantly reduces development time and ensures the eventual process or product achieves optimal quality [3] . Three main components must be included in a digital twin [4] :

- Physical objects in real space.
- Digital twins in the form of software.
- Data that links the first two elements together.

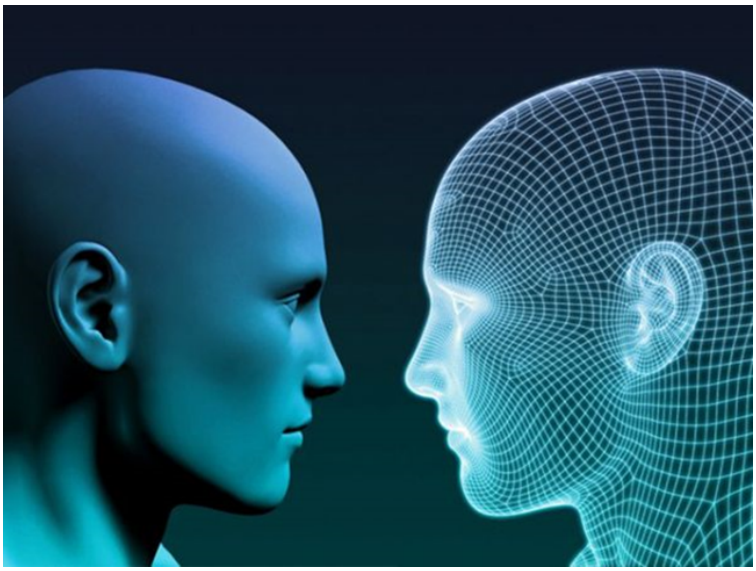


Fig. 1.1: Digital twin.

Digital twins can help manufacturers and engineers complete many tasks, such as accelerating design optimization, reducing product development time, improving product quality, predicting performance, facilitating rapid testing and simulation, enhancing training opportunities, optimizing maintenance, monitoring equipment health in real-time, and enabling predictive insights into machine behavior and performance [5] [6] [7] .

1.3 Internet Of Things IOT

The Internet of Things (IoT) refers to a network of physical devices, vehicles, appliances, and other physical objects that are embedded with sensors, software, and network connectivity, allowing them to collect and share data [19].

1.3.1 IoT technologies

IoT technologies encompass the diverse set of tools that support and drive Internet of Things devices and networks. Here are several examples of IoT technologies :

- **Sensors and actuators :** Sensors are devices that can detect changes in the environment, such as temperature, humidity, light, motion, or pressure. Actuators are devices that can cause physical changes in the environment, such as opening or closing a valve or turning on a motor. These devices are at the heart of IoT, as they allow machines and devices to interact with the physical world. Automation is possible when sensors and actuators work to resolve issues without human intervention [19].
- **Connectivity technologies :** To transmit IoT data from sensors and actuators to the cloud, IoT devices need to be connected to the internet. There are several connectivity technologies that are used in IoT, including wifi, Bluetooth, cellular, Zigbee, and LoRaWAN.[19].
- **Security and privacy technologies :** As IoT deployments become more widespread, IoT security and privacy become increasingly important. Technologies such as encryption, access controls and intrusion detection systems are used to protect IoT devices and the data they generate from cyber threats[19].
- **Cloud computing :** The increase in the availability of cloud platforms enables both businesses and consumers to access the infrastructure they need to scale up without actually having to manage it all[20].
- **Machine learning and analytics :** With advances in machine learning and analytics, along with access to varied and vast amounts of data stored in the cloud, businesses can gather insights faster and more easily. The emergence of these allied technologies continues to push the boundaries of IoT and the data produced by IoT feeds these technologies[20].

1.3.2 IoT Architecture

IoT architecture refers to the tangle of components such as sensors, actuators, cloud services, Protocols, and layers that make up IoT networking systems. In general, it is divided into layers that allow administrators to evaluate, monitor, and maintain the integrity of the system. The architecture of IoT is a four-step process through which data flows from devices connected to sensors, through a network, and then through the cloud for processing, analysis, and storage. With further development, the Internet of Things is poised to grow even further, providing users with new and improved experiences. [21] The key building blocks of any IoT structure are always the same in [22] namely :

- Smart things.
- Networks and gateways enabling low-power devices (which is often the case in IoT) to enter the big Internet.
- The middleware or IoT platforms providing data storage spaces and advanced computing engines along with analytical capabilities.
- Applications, allowing end users to benefit from IoT and manipulate the physical world. These components form the foundation for a robust multi-layered architecture, upon which

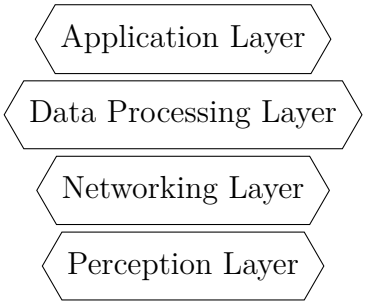


FIG. 1.2: IoT Architecture

any IoT system can be built. Typically, these layers include

- Perception layer : It is the foremost layer that is also known as the sensor layer. It implies all types of sensors that can be used in IoT devices. They gauge particular parameters. Then, they transform signals from the environment sensors interact with them into digital information. Sensors can help create smart premises maintenance systems. The most popular ones that represent the first IoT architecture layer are temperature, pressure, and infrared sensors [23].
- Network Layer : It is known as a gateway layer that helps establish the logical design of IoT by connecting devices with a platform. The data captured by sensors get transmitted by wires or radio to back-end IoT services. The transferred data is secured to prevent unauthorized access. In most cases, the transmitted information is encrypted. The most popular IoT communication models are the following : Ethernet, WiFi, NFC (Near Field Communication), Bluetooth [23].
- Processing layer : is also known as the middleware layer. It is a layer that gathers all the data provided by the perception layers through a network. All the information is stored and analyzed. It is one of the most important layers of IoT because it makes decisions based on data analysis. Also, the application layer interacts with a user to manage and operate their IoT devices. This layer is placed between Network and Application layers in the logical design of IoT architecture[23].
- Application layer : The application layer includes all the software necessary to offer a specific service. In this level, the data from the previous levels are stored, aggregated, filtered, and processed, and databases, analysis software, etc., are used. As a result of this processing process, the data is made available to real IoT applications (smart wearable, smart car, etc.)[24].

1.3.3 IoT Applications

IoT applications span a wide range of domains like :

- Smart Home Applications : Refers to all of the linked objects that can be contained in a house : wired lamps, smart fridges, thermostats with controlling sensors, and so on. The Internet of Things is bringing the house of the future to life[[25].
- Industrial Automation : With IoT applications, it's also possible to re-engineer products and their packaging for improved cost and consumer service[25].
- Health Care : IoT has revolutionized healthcare by enabling remote monitoring of patients, improving the accuracy of diagnoses, and enhancing the overall quality of care. IoT devices and sensors are used in hospitals, clinics, and homes to collect health-related data and transmit it to healthcare providers in real-time. IoT devices include wearables, telemedicine platforms, smart pill dispensers, and hospital asset trackers[26].
- Smart Cities : In a smart city IoT, application, various devices and systems are interconnected to enhance urban living. Smart traffic lights, environmental sensors, waste disposal systems, and public transportation all contribute to the IoT ecosystem. Data from these devices are collected, analyzed, and used to optimize traffic flow, reduce pollution, and improve the overall quality of life in the city. Residents can interact with smart city services through mobile applications, demonstrating the involvement of human interaction in IoT[26].
- Agriculture : Smart Agriculture, an IoT application, employs sensors to collect data on soil conditions, weather, crop health, and livestock. This data-driven approach enables precise irrigation, pest control, and resource allocation. Farmers use analytics for informed decisions, automation for efficiency, and remote monitoring for convenience. Smart Agriculture promotes sustainability by reducing waste and environmental impact while enhancing crop yields and reducing costs[26].

1.3.4 IoT characteristics

The Internet of Things (IoT) is characterized by the following key features that are mentioned below [27][28] :

- Connectivity : Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connect, this should be guaranteed at all times. For example, the connection between people through Internet devices like mobile phones, and other gadgets, also a connection between Internet devices such as routers, gateways, sensors, etc
- Intelligence and Identity : The extraction of knowledge from the generated data is very important. For example, a sensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status
- Safety : There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also be at risk. Therefore, equipment safety is also critical.
- Self-Configuring : This is one of the most important characteristics of IoT. IoT devices are able to upgrade their software in accordance with requirements with a minimum of user participation. Additionally, they can set up the network, allowing for the addition of new devices to an already-existing network[27].

- Scalability : The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as an outcome is enormous, and it should be handled appropriately.
- Heterogeneity : The devices within the IoT exhibit heterogeneity due to their reliance on diverse hardware platforms and networks. They have the capability to communicate with other devices or service platforms across various networks.
- Things-related services : Within the limitations of things, the Internet of Things can offer thing-related services like semantic consistency between actual objects and the virtual objects they are connected with, as well as privacy protection. Both the physical and information worlds' technologies will evolve to give thing-related services within the limitations of things [28].

1.3.5 Advantages and Disadvantages of the IoT

The Internet of Things (IoT) presents a myriad of advantages and disadvantages that impact various aspects of our lives and industries

1.3.5.1 IoT advantages

A few advantages associated with the IoT are :

- Through IoT, information can be accessed from anywhere with any device and at any time.
- There is improved communication amongst the devices that are connected.
- The data packets can be transferred through the connected network that saves money and time.
- Through the automated task in the organizations, the quality of the services offered by the businesses has improved, thereby reducing the interference or the need for human intervention [29]. II.1.5.2 IoT disadvantages A few disadvantages associated with the IoT are :
- In the IoT, there is an increase in the number of devices and the connectivity between the devices. As a result, there is more sharing of information between the devices. This might lead to hacking cases as there is a potential of stealing important confidential information from a hacker. Therefore, the chances of hacking increase.
- With the application of IoT, the data in the organization increases. Therefore, the organization has to deal with the massive amount of generated data. This stands as a challenge as collecting such a large amount of data generated from all the connected devices is not easy.
- The presence of bugs in any system will corrupt every other connected device in the IoT.
- If the connected devices are from different manufacturers, there might be a compatibility problem between the devices in the IoT [29].

1.4 Digital twin types

Digital twin models offer the capability to simulate everything from individual components to complete systems. Although all virtual twin types essentially perform the same function of digitally replicating real-world objects or systems, their intentions and extent differ significantly. The four main types of digital twins include

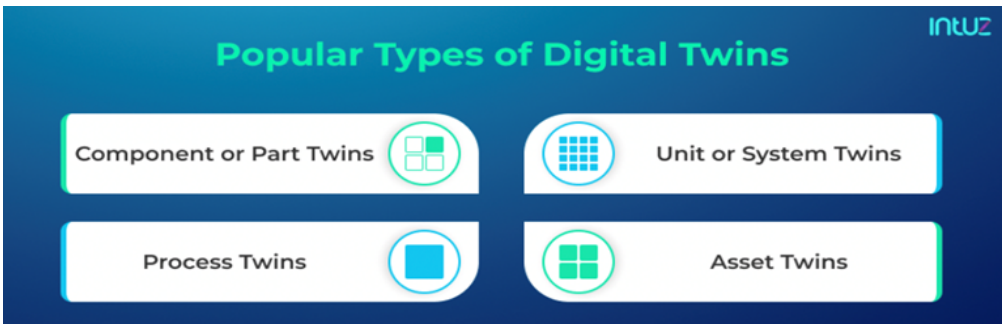


FIG. 1.3: Digital twin types [8].

- Component twins : Component twins are digital renditions of specific parts within a system or product, such as gears or screws. Instead of merely replicating every single part of a product, component twins focus on modeling crucial elements, especially those subjected to specific stresses or high temperatures. By digitally simulating these components and subjecting them to dynamic scenarios, designers and engineers gain insights into potential enhancements required to maintain their integrity in probable conditions [9].

- Asset twins : Asset Twins represent an advanced stage in digital twin technology, showcasing the collaborative functioning of individual components as a cohesive asset – exemplified by entities like engines or pumps. They can integrate data from Component Twins or encompass multiple Component Twins themselves. While Component Twins focus primarily on the stability and endurance of individual parts, Asset Twins provide a holistic view of entire systems. This enables the examination of the synergy between individual parts and identifies areas for enhancement without necessitating physical manipulation of real engines or machine gearboxes. Consequently, Asset Twins contribute to virtual – and consequently real-world – enhancements, reducing mean time between failures and repairs, minimizing fuel consumption, and enhancing performance factors [73].

- System or unit twins : System twins, also known as unit twins, operate at a higher level by amalgamating individual Asset Twins, allowing you to observe how these assets collaborate – akin to Asset Twins, which merge Component Twins. To illustrate, think of a car : the System Twin combines all assets related to propulsion, electricity, bodywork, etc., offering versatility across various applications. In a simpler scenario, such as a car factory, a System Twin consolidates units required for producing a car component. The goal of System Twins is to optimize collaboration among assets, resulting in maximum performance with minimal wear and tear or time consumption [10],[1].

- Process twins : which provide a big-picture view, show how different systems collaborate to run a whole production facility. They help us see if these systems are working together smoothly or if delays in one part affect others. By using process twins, we can figure out the best timing for each part of the process, which ultimately affects how well everything works [1].

1.5 The Digital Twin Architecture

General DT architecture consists of three main elements : the physical world, the virtual world, and the connectivity between them.

- Each element integrates various components based on designer's needs.

- Basic components include sensors for data collection, a physical twin, edge processing capabilities, data security, the digital twin itself, data processing capabilities (enabled by ML, AI, big data, etc.), and communication interfaces like internet, Bluetooth, satellite, etc.
- DT architecture includes data visualization for user comprehension [11].

1.6 Digital twin technologies

The three main aspects of Digital Twins are data acquisition, data modeling, and data application [12]. Digital Twin uses four technologies to collect and store real-time data, obtain information to provide valuable insights, and create a digital representation of a physical object. These technologies include :

- Internet of Things (IoT) : IoT refers to a giant network of connected "things". The connection is between things–things, people–things, or people–people [13]. IoT uses sensors to collect data from real-world objects. The data transmitted by IoT is used to create a digital duplication of a physical object. The digital version then can then be analyzed, manipulated, and optimized. IoT constantly updates data and helps Digital Twin applications create a real-time virtual representation of a physical object. Therefore, every Digital Twin application uses IoT as a primary technology [14].
- Cloud computing : Cloud computing provides Digital Twins with data computing technology and cloud data storage technology. Cloud computing allows Digital Twin, with large volumes of data, to store data in the virtual Cloud and easily access the required information from any location. Cloud computing enables Digital Twins to effectively reduce the computation time of complex systems and overcome the difficulties of storing large amounts of data [15].
- Artificial Intelligence (AI) : AI can help Digital Twins by offering an advanced analytical tool that can automatically analyze the collected data and provide valuable insights, to make predictions about the results and to propose suggestions to avoid possible problems. [12].
- Extended Reality (XR) : XR generates digital representations of objects where digital and real objects coexist and interact in real time. Digital Twins leverage XR capabilities to digitize physical elements, allowing users to interact with digital content [14].

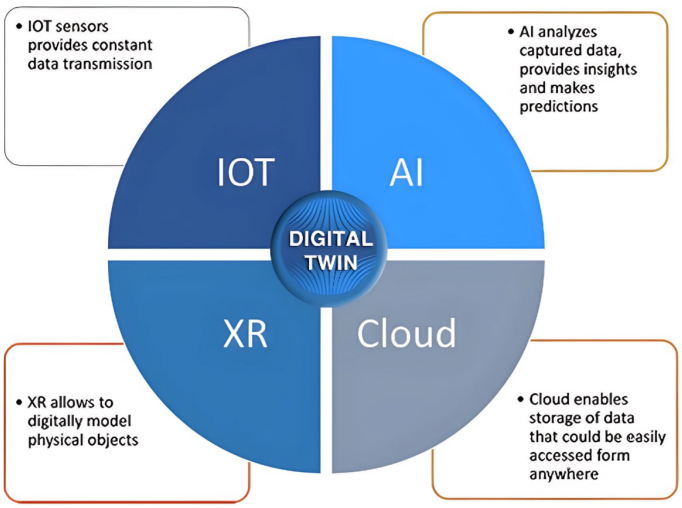


FIG. 1.4: Technologies of Digital Twins [14].

1.7 Digital twins use cases and applications

Today, engineering and manufacturing predominantly use Digital Twins to provide accurate virtual representations of objects and simulations of operational processes. Digital Twins applications in operations and supply chain management, especially the role of Digital Twins in terms of operations traceability, transport maintenance, remote assistance, asset visualization, and design customization [14].
• **Applications :** Digital twins are used in a wide variety of industries for a range of applications and purposes. Some notable examples include :

- Power-generation equipment : Large engines—including turbojets, locomotive engines, and power generation turbines—benefit greatly from the use of digital twins, including to help establish regular maintenance schedules [1].
- Manufacturing operations : Digital twins can make manufacturing more productive and streamlined while reducing throughput times [18].
- Automotive industry : One example of where digital twins are used in the automotive industry is to gather and analyse operational data from a vehicle in order to assess its status in real time and inform product improvements [18].
- Urban planning : Civil engineers and others involved in urban planning activities are aided significantly by the use of digital twins, which can show 3D and 4D spatial data in real time and incorporate augmented reality systems into built environments [1].
- Healthcare services : Just as products can be profiled by using digital twins, so can patients receiving healthcare services. Band-aid-sized sensors send health information back to a digital twin used to monitor and predict a patient's wellbeing [16].

1.8 Digital twin benefits

There are as many benefits to using digital twins [17] :

- Increased reliability of equipment and production lines.
- Improved OEE through reduced downtime and improved performance
- Improved productivity
- Reduced risk in various areas, including product availability, marketplace reputation, and more
- Lower maintenance costs by predicting maintenance issues before breakdowns occur
- Faster production times
- New business opportunities such as mass customisation, mixed manufacturing, small-batch manufacturing, and more
- Improved customer service as customers can remotely configure customised products
- Improved product quality and enhanced insight into the performance of your products in multiple real-time applications and environments
- More efficient supply and delivery chains
- Improved profits.

1.9 Technologies and tools used for digital twins

The technologies and tools commonly used for digital twins : Eclipse Ditto and the cloud tools (IBM, Azur,…).

1.9.1 Eclips Ditto

Eclipse Ditto is an open-source framework designed to create digital twins of internet-connected devices, suitable for various IoT domains including industrial, residential, and agricultural. As IoT middleware, Ditto provides an abstraction layer that interacts with physical devices using the digital twin pattern, integrating devices through connectivity layers like Eclipse Hono™ or MQTT brokers such as Eclipse Mosquitto™. It also integrates with other existing backend systems using various protocols like AMQP, MQTT, Apache Kafka, and HTTP, allowing bidirectional data exchange and payload mapping [30]. Its architecture includes components for policy enforcement, thing management, change tracking, and connectivity, all running within the same Pekko cluster to facilitate inter-component communication without additional message brokers [31]. Eclipse Ditto architecture consists of Policies, Things, Things-Search, Gateway, and Connectivity services, These services communicate with each other for data persistence, authorization, and search functionalities [31]. Ditto supports several protocols : CoAP for lightweight device communication using UDP and DTLS for security ; MQTT for reliable, TCP-based data streams in remote IoT monitoring ; AMQP for asynchronous message queuing ; and DDS for secure, real-time data distribution using a Global Data Space and QoS contracts, critical for industrial IoT applications [32] [33].

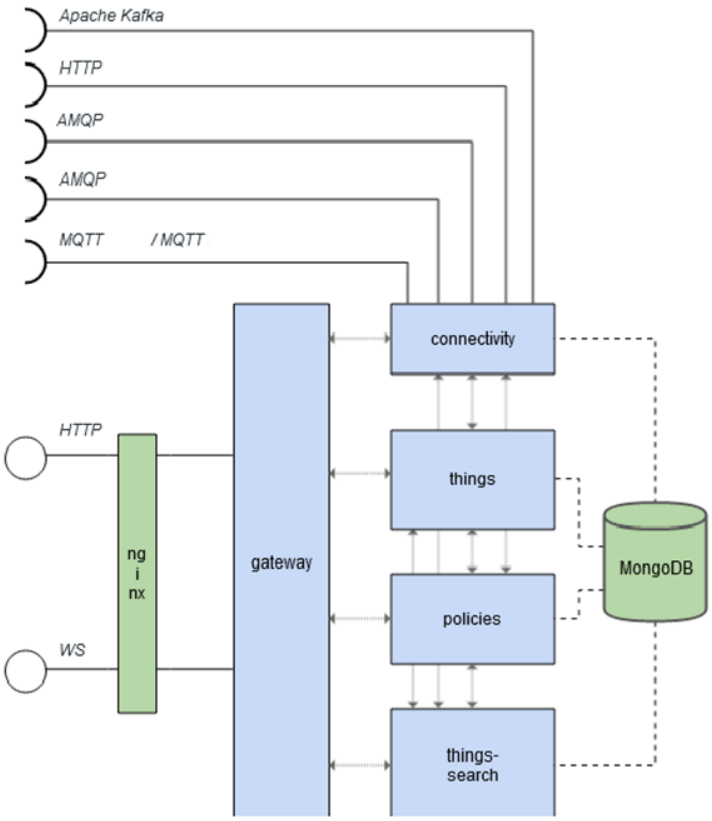


Fig. 1.5: Digital twin.

1.9.2 The cloud tools

1.9.2.1 Azur

Azure Digital Twins is a digital twin software platform that presents organizations with the foundation to build the next generation of IoT solutions. They create data-driven workspaces that combine all the necessary sources and develop new production technologies. Azure Digital Twins can be connected to Azure Stream Analytics, Azure AI, Azure Maps, Azure Storage, Microsoft Mixed Reality, Office 365, or Dynamics 365. IoT and AI platforms significantly reduce the complexity of creating digital copy solutions by enabling functions that trigger automatic endpoint actions based on incoming information. Azure Digital Twins utilizes all environments, such as warehouses, offices, schools, hospitals, and banks. It can even be applied to factories, parking, stadiums, parks, intelligent grids, and cities [34]. III.2.2 IBM : IBM is another example of digital twin applications that have had a priceless impact on business in this high-tech novelty revolution. IBM's Watson Internet of Things system unites system design, engineering lifecycle optimization, and Rational Lifecycle Integration capabilities. IBM uses digital twin technology for virtual creation, testing, and monitoring products and processes. The extended IBM Engineering Lifecycle Management (ELM) solution, with integrations to many third-party tools, helps reduce latency in the feedback loop between design and operation. The IBM Engineering Systems Design Rhapsody (Rational Rhapsody) products offer a comprehensive solution for design, development, and test environments for system engineers. The help of UML, SysML, and AUTOSAR makes it easier for software engineers to do their jobs [34].

1.10 Conclusion

In summary, this chapter provided a comprehensive overview of digital twins and their integration with the Internet of Things (IoT). We explored the concept of digital twins, their types, architecture, technologies, and applications across various domains. Additionally, we discussed the foundational aspects of IoT, including its technologies, architecture, applications, advantages, and disadvantages. Finally, we examined the tools and technologies commonly used for implementing digital twins, such as Eclipse Ditto ,IBM and Azur.

2

Deep Learning

Content

2.1 Introduction 14
2.2 Machine learning (ML) 14
2.3 Deep Learning (DL) 15
2.4 Popular Deep learning Models 19
2.5 Difference between Machine Learning and Deep Learning 23
2.6 Challenges in Deep Learning 23
2.7 Conclusion 24

2.1 Introduction

Machine learning is a field of study that attempts to make a machine learn by itself without being explicitly instructed. A subset of ML called deep learning (DL) attempts to address and resolve more intricate issues, such as pattern recognition, signal processing, memorization, and generalizing to more complex problems with more depth and greater accuracy, all of which are based on the biological neural mechanism. This is converted into an artificial neural network [76]. This chapter delves into the basic ideas and principles that underpin deep learning and machine learning, encompassing several kinds of learning like reinforcement learning, supervised learning, and unsupervised learning. We explore the inner workings of neural networks, including recurrent neural networks (RNNs), convolutional neural networks (CNNs) and other models. We also look at the many areas that these technologies may be used to, including healthcare, and the challenges that must be addressed to fully leverage their potential.

2.2 Machine learning (ML)

Machine learning is a computer programming technique that uses statistical probabilities to give computers the ability to learn for themselves without explicit programming. The basic objective of machine learning is to "learn to learn" computers – and then to act and react – as humans do, by improving their learning mode and knowledge autonomously over time. The ultimate goal would be for computers to act and react without being explicitly programmed for these actions and reactions. Machine learning uses development programs that adjust each time they are exposed to different types of input data [37]. There are three main types of machine learning : supervised, unsupervised, and semi-supervised learning

2.2.1 Supervised Learning

Supervised learning is an approach to creating artificial intelligence (AI) where a computer algorithm is trained on input data that has been labeled for a particular output. The model is trained until it can detect the underlying patterns and relationships between the input data and the output labels, enabling it to yield accurate labeling results when presented with never-before-seen data [38].

2.2.2 Unsupervised Learning

In the context of unsupervised machine learning, the algorithm determines the structure of the input itself (no label is applied to the algorithm). This approach can be a goal in itself (that allows to discover structures buried in the data) or a means to achieve a certain goal. This approach is also called feature learning [39].

2.2.3 Semi-supervised Learning

Semi-supervised learning is a type of machine learning that falls in between supervised and unsupervised learning. It is a method that uses a small amount of labeled data and a large amount of unlabeled data to train a model. The goal of semi-supervised learning is to learn a function that can accurately predict the output variable based on the input variables, similar to supervised learning. However, unlike supervised learning, the algorithm is trained on a dataset that contains both labeled and unlabeled data [40].

2.3 Deep Learning (DL)

Deep learning is a subset of machine learning that uses multi-layered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain. Some form of deep learning powers most of the artificial intelligence (AI) in our 4 lives today [41].

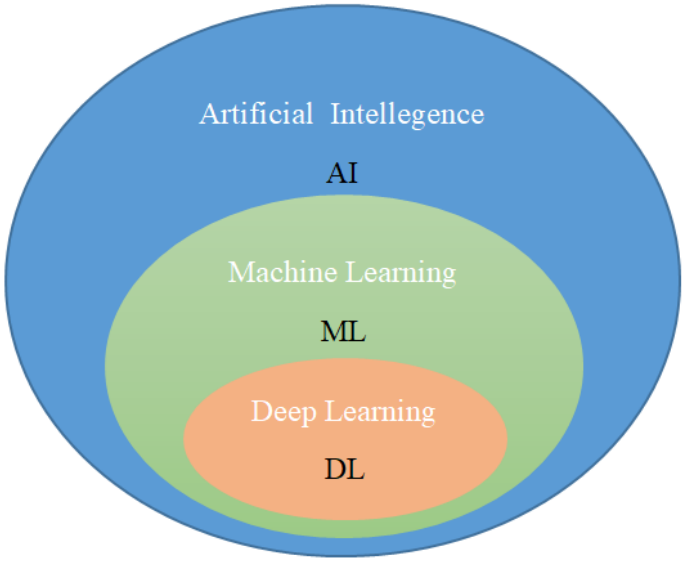


FIG. 2.1: Represents the Relation between Artificial Intelligence, Machine Learning and Deep Learning

2.3.1 How Deep Learning Works

Deep learning is a branch of machine learning that enables machines to learn how to perform classification tasks directly from raw data such as images, text, or audio. Based on a hierarchical learning principle, deep learning involves non-linear transformations between each layer, where each layer receives the output of the previous layer as input. What sets deep learning apart from classical approaches is its replacement of the manual feature extraction phase with hierarchical feature extraction algorithms. Although the concept of deep learning emerged in the 1980s, its capabilities were only fully realized recently, mainly due to two key factors : the availability of large annotated datasets for training and the need for considerable computing power. The term "deep" refers to the number of hidden layers; unlike multilayer neural networks, which typically have only 2 to 3 layers, deep networks can have up to hundreds of hidden layers. Classification systems based on deep learning have demonstrated exceptional recognition rates by leveraging extensive datasets for training and using models of artificial neural networks with multiple hidden layers [42].

2.3.2 Deep learning applications

Currently, deep learning is in widespread use across several industries such as law enforcement [41], where it analyzes transactional data to detect fraud and criminal activities, and investigative analysis through speech recognition and computer vision. In the financial sector [41], it powers algorithmic trading, evaluates business risks, detects fraud, and manages credit and investment portfolios. Machine translation [43]. is significantly advanced with the invention of tools like Google Translate that simplify language translation. Other applications of deep learning are seen in virtual

assistants [43]. such as Siri, Alexa, and Google Assistant, which all rely on deep learning to understand natural interaction. Self-driving cars [43], such as those coming out of Uber's AI lab, are possible thanks to deep learning. The healthcare industry has benefited greatly from deep learning capabilities ever since the digitization of hospital records and images. Image recognition applications can support medical imaging specialists and radiologists, helping them analyze and assess more images in less time [41]. Also, DL has been applied in various ways to improve baby care, particularly in the areas of computer vision and intelligent baby monitoring systems. One of the most common uses of computer vision in baby care is baby monitoring. Baby monitors are devices that allow parents and caregivers to monitor and listen to their babies remotely, usually through a smartphone app. Some baby monitors use computer vision to improve their features, such as detecting baby's movements, breathing, temperature, heart rate, and sleep habits. These features can alert parents and caregivers if the baby is in distress, needs attention or has a health problem. Computer vision can also help analyze the baby's facial expressions and emotions, and provide feedback and suggestions for improving the baby's mood and well-being [74].

2.3.3 Deep Learning Types

This section introduces some types of deep learning :

2.3.3.1 Neural Networks

A neural network is a machine learning program, or model, that makes decisions in a manner similar to the human brain, by using processes that mimic the way biological neurons work together to identify phenomena, weigh options and arrive at conclusions. Every neural network consists of layers of nodes, or artificial neurons—an input layer, one or more hidden layers, and an output layer. Each node connects to others, and has its own associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. Neural networks rely on training data to learn and improve their accuracy over time. Once they are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the best-known examples of a neural network is Google's search algorithm. Neural networks are sometimes called artificial neural networks (ANNs) or simulated neural networks (SNNs). They are a subset of machine learning, and at the heart of deep learning models [44].

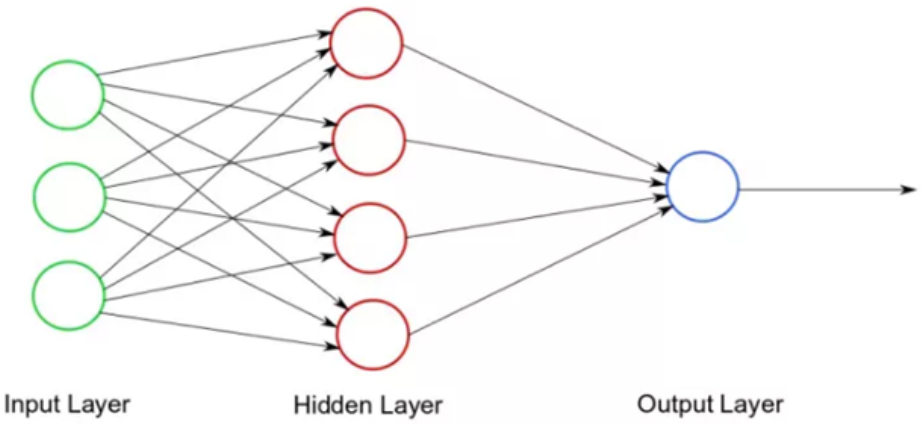


FIG. 2.2: Simple Neural Network.

- **Basics of Neural Networks :**

  Neural networks help in finance Development of processes such as time series forecasting, algorithmic trading, security classification, credit risk modeling, and proprietary indicator creation Price Derivatives. Neural networks contain layers of interconnected nodes. Each node is called a perceptron, similar to multiple linear regression. Perceptron feed signal Generate activation functions that may be nonlinear through multiple linear regression.

- **Multi-Layered Perceptron :**

  In a multilayer perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has a classification or output signal that can be associated with the input pattern. For example, these patterns can include a quantitative list of security technical indicators ; possible outcomes Can be "Buy", "Hold" or "Sell".

- **Hidden layers :** Adjust the input weights until the neural network's fault tolerance is minimal. Assume that the hidden layer infers salient features in the input data Ability to predict outcomes. This describes feature extraction, which provides similar utility to statistical techniques such as principal component analysis [45].

2.3.3.2 Convolutional Neural Networks

Convolutional neural networks (CNN) are another variation of feedforward multilayer perceptrons. It is a feedforward neural network in which individual neurons are arranged to respond to all overlapping areas in the visual area. This is a network consisting of learned weights and biases. Each input layer consists of a sequence of neurons where a dot product is performed on each input, further extending the concept of nonlinearity. It is a fully connected network that uses the SVM/-SoftMax function as the loss function. CNNs are at the heart of the most sophisticated computer vision applications, facial safety features, and automatic license plate recognition [46].

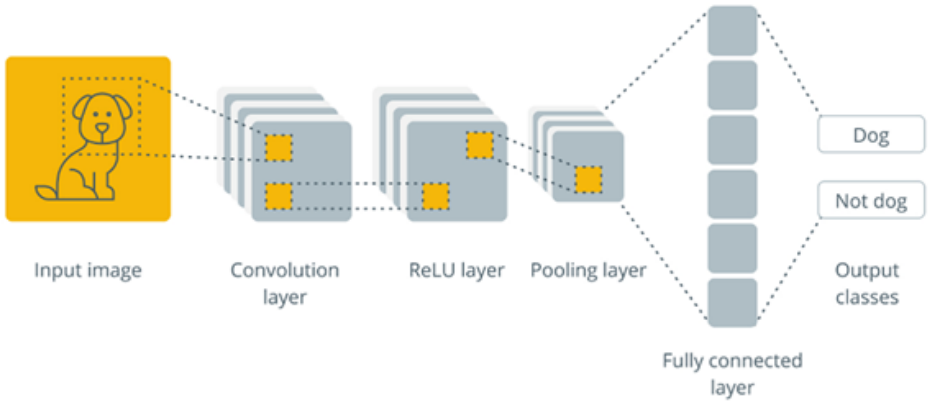


FIG. 2.3: The Architecture of CNN [67]

2.3.3.3 Feed Forward Neural Network

This type of neural network is the most basic type of neural network, where flow control occurs from the input layer to the output layer. This type of network consists of only a single layer or only one hidden layer. Since the data only moves in one direction, there is no backpropagation technique here network. In this network, the sum of weights present in the input is fed to the input layer. Such networks are used in computers' facial recognition algorithms Imagine [46].

2.3.3.4 Recurrent Neural Networks

RNNs are a commonly employed and familiar algorithm in the discipline of DL RNN is mainly applied in the area of speech processing and NLP contexts Unlike conventional networks, RNN uses sequential data in the network. Since the embedded structure in the sequence of the data delivers valuable information, this feature is fundamental to a range of different applications. For instance, it is important to understand the context of the sentence in order to determine the meaning of a specific word in it. Thus, it is possible to consider the RNN as a unit of short-term memory, where x represents the input layer, y is the output layer, and s represents the state (hidden) layer [48]. For a given input sequence, a typical unfolded RNN diagram is illustrated in 2.4 Pascanu et al. [48]

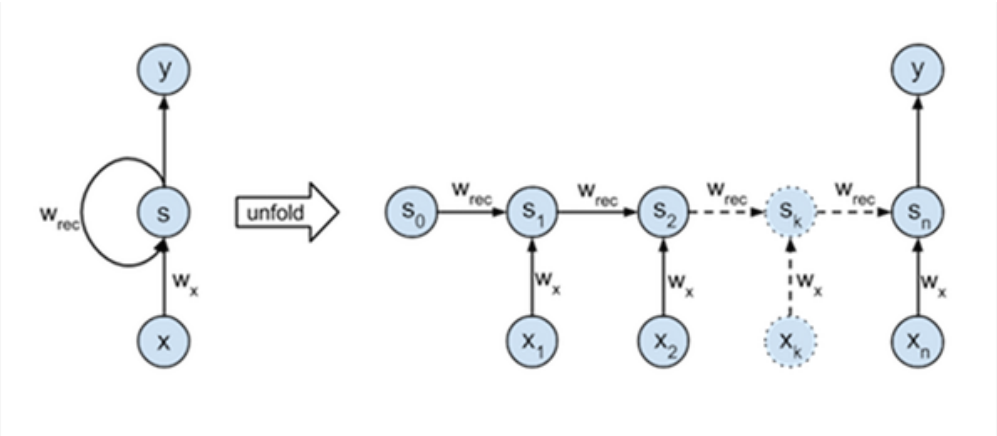


FIG. 2.4: Typical unfolded RNN diagram [68]

introduced three different types of deep RNN techniques, namely “Hidden-to-Hidden”, “Hidden-to-Output”, and “Input-to-Hidden”.Deep RNN is introduced, which reduces the learning difficulty of deep networks and provides the advantages of deep RNN based on these three technologies.

2.3.3.5 Auto Encoders

An autoencoder is an artificial neural network that can learn different encoding patterns. The simple form of an autoencoder is similar to a multilayer perceptron, containing an input layer or one or more hidden layers or an output layer. The main difference between a typical multilayer perceptron and feedforward neural networks and autoencoders is the number of nodes in the output layer. In an autoencoder, the output layer contains the same number of nodes as the input layer. Instead of predicting a target value based on an output vector, the auto encoder must predict its input [49] . The AutoEncoder network has two blocks :

1. **Encoder :** includes layers that encode a compressed representation of input data by dimensionality reduction. In a typical auto-encoder, the hidden layers of the neural network contain a number of nodes that gradually shrink compared to the input layer : as the data passes through the layers of the encoder, they are compressed into a small number of dimensions

2. **Decoder :** Hidden layers whose number of nodes gradually increases decompress (or decode) the encoded representation of the data, eventually reconstructing the data in its original pre-encoding form. This reconstructed output is then compared to the « ground truth », in most cases, the original input, to evaluate the efficiency of the auto-encoder. The difference between output and field truth is what is called reconstruction error [50].

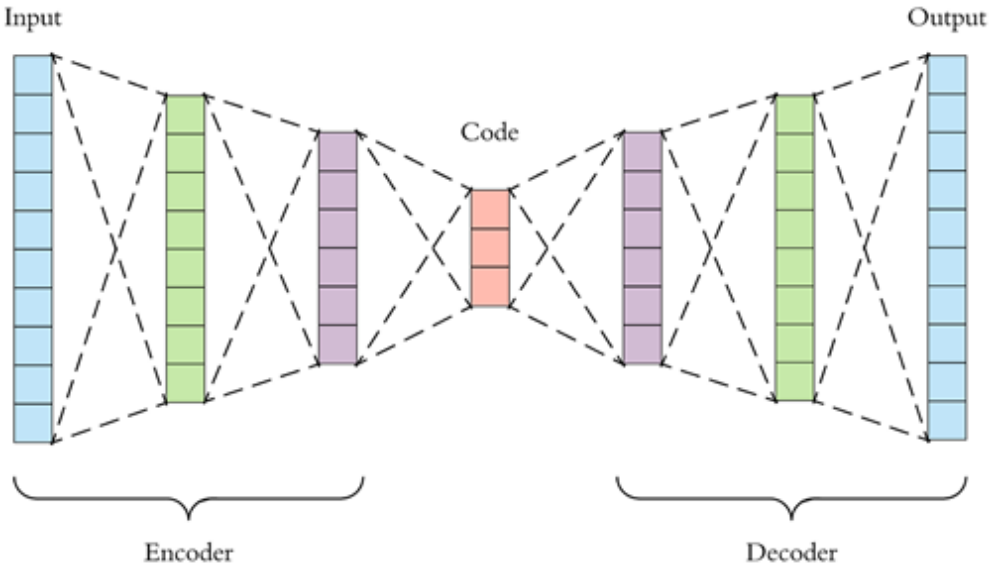


FIG. 2.5: The Architecture of Auto Encoders [69]

2.3.3.6 Recursive Neural Network

Recursive Neural Networks are a type of neural network architecture that is specially designed to process hierarchical structures and capture dependencies within recursively structured data. Unlike traditional feedforward neural networks (RNNs), Recursive Neural Networks or RvNN can efficiently handle tree-structured inputs which makes them suitable for tasks involving nested and hierarchical relationships [51].

2.3.3.7 Deep Belief Networks (DBN)

Deep belief networks are a solution to the problem of non-convex objective functions and local minima using typical multi-layer perceptrons. It is another type of deep learning that consists of multiple layers of latent variables with connections between the layers. Deep belief networks can be viewed as restricted Boltzmann machines (RBMs), where the hidden layer of each sub-network serves as the visible input layer for adjacent layers of the network. It uses the lowest visible layer as the training set for adjacent layers of the network. This way, each layer of the network is trained independently and greedily. Hidden variables are used as observed variables to train each layer of the deep structure [58].

2.4 Popular Deep learning Models

2.4.1 Long Short-Term Memory (LSTM) Network

LSTMs Long Short-Term Memory is a type of RNNs Recurrent Neural Network that can detain long-term dependencies in sequential data. LSTMs are able to process and analyze sequential data, such as time series, text, and speech. They use a memory cell and gates to control the flow of information, allowing them to selectively retain or discard information as needed and thus avoid the vanishing gradient problem that plagues traditional RNNs. LSTMs are widely used in various applications such as natural language processing, speech recognition, and time series forecasting [52]. Long Short-Term Memory (LSTM) is a highly effective Recurrent Neural Network (RNN) that has been used in various applications such as Language Simulation where Language support vector machines (LSTMs) have been used for natural language processing tasks like machine translation,

language modeling, and text summarization. By understanding the relationships between words in the sentence, they can be trained to construct meaningful and grammatically correct sentences. They are also used in the field of Voice Recognition where they are used for speech recognition tasks such as speech-to-text-to-text-transcription and command recognition. They can be taught to recognize and match speech patterns with appropriate text, as well as Sentiment Analysis LSTMs can be used to classify text sentiment as positive, negative, or neutral by learning the relationships between words and associated feelings. Time Series Prediction they can be used to predict future values in a time series by learning the relationships between past values and future values. Finally, they can be used in Video Analysis by learning the relationships between frames and their associated actions, objects, and scenes and recognize handwriting by learning the relationships between images and the corresponding text [55].

2.4.2 Generative adversarial networks (GANs)

Deep learning generative algorithms called GANs produce new data points that mimic the training data. GAN consists of two parts : a generator that learns to produce false data and a discriminator that absorbs the false data into its learning process. Over time, GANs have become more often used. They can be employed for dark-matter studies to replicate gravitational lensing and enhance astronomy photos. In order to recreate low-resolution, 2D textures at classic video games in 4K or higher resolutions, video game producers use GANs. GANs help produce cartoon characters and realistic images, take pictures of real humans, and display 3D objects and generating synthetic data from real human pictures. GANs function by having a discriminator that learns to distinguish between real data samples and the generator's fake data, Initially, the generator creates poor-quality fake data, during early training, and the characteristic quickly captures the fact that it is incorrect. In order to update the model, GAN presents the results to the featured and generator [53]. Applications of GANs include generating new examples of image data sets in various fields, such as medical imaging, satellite images and natural language processing. By creating artificial data, researchers can increase existing data sets and improve the performance of machine learning models. translating images from one domain to another (e.g., converting photos into drawings or paintings), predicting future video frames based on past frames to generate realistic video sequences, and generating frontal views of faces from angled images to enhance face recognition or create new facial images [59].

2.4.2.1 Recurrent neural networks (RNNs)

The outcomes from the LSTM can be provided as input to the current stage thanks to the directed cycles formed by connections between RNNs. The LSTM's output becomes an input for the current phase and, thanks to its internal memory, may remember prior inputs. Natural language processing, time series analysis, handwriting recognition, and computational linguistics are all common applications for RNNs. Recurrent Neural Networks (RNNs) function by feeding the output at time t-1 into inputs at time t, and similarly, the output is timely feeding t into inputs at t + 1 time, where any length of inputs can be handled by RNNs taking into account historical data without increasing the model size as the volume of inputs grows [53]. RNNs are widely used in the following domains/ applications : Speech Recognition, RNNs can be used for predicting phonetic segments considering sound waves from a medium as an input source. The set of inputs consists of phoneme or acoustic signals from an audio which are processed in a proper manner and taken as inputs. The RNN network will compute the phonemes and then produce a phonetic segment along with the likelihood of output. In text Summarization, they help in summarizing content from literatures and customising them for delivery within applications which cannot support large volumes of text. For example, if a publisher wants to display the summary of one of his books on its backpage to help the readers get an idea of the content present within, Text Summarization would be helpful. in generating image

descriptions, A combination of CNNs and RNNs are used to provide a description of what exactly is happening inside an image. CNN does the segmentation part and RNN then uses the segmented data to recreate the description. and in language modeling and text generation, taking a sequence of words as input, we try to predict the possibility of the next word. This can be considered to be one of the most useful approaches for translation since the most likely sentence would be the one that is correct. In this method, the probability of the output of a particular time-step is used to sample the words in the next iteration [60].

2.4.2.2 Radial basis function networks (RBFNs)

Radial basis functions are a unique class of neural network feedforward networks (RBFNs) that are used as activation functions. They are typically used for classifications, regression, as well as time-series prediction and have input data, a hidden layer, and a convolution layer. Radial Basis Function Networks (RBFNs) function by classifying data through comparison of input instances to samples from the training set. The process starts with feeding an input vector. They contain an RBF neural layer. One node per group or type of data is present in the output layer, where the function calculates the weighted total of the inputs. Each neuron in the hidden layer uses a Gaussian transfer function, with outputs inversely related to the distance from the neuron's center. in the end, Radial-basis functions from the input and the neuron's parameters are combined linearly to form the network's output [53].

2.4.2.3 Feedforward Neural Networks (FNN)

A feedforward neural network is one of the simplest types of artificial neural networks devised. In this network, the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any), and to the output nodes. There are no cycles or loops in the network. Feedforward neural networks were the first type of artificial neural network invented and are simpler than their counterparts like recurrent neural networks and convolutional neural networks. The working of a feedforward neural network involves two phases : the feedforward phase and the backpropagation phase.

- Feedforward Phase : In this phase, the input data is fed into the network, and it propagates forward through the network. At each hidden layer, the weighted sum of the inputs is calculated and passed through an activation function, which introduces non-linearity into the model. This process continues until the output layer is reached, and a prediction is made.
- Backpropagation Phase : Once a prediction is made, the error (difference between the predicted output and the actual output) is calculated. This error is then propagated back through the network, and the weights are adjusted to minimize this error. The process of adjusting weights is typically done using a gradient descent optimization algorithm [54].

2.4.2.4 Multilayer Perceptrons (MLPs)

MLPs are a kind of neural network feedforward that contain many layers of activation-function-equipped perceptrons. A completely coupled input data and a hidden layer make up MLPs They can be used to create speech recognition, picture recognition, and machine translation software since they have the same amount of input and output stages but may have several hidden layers. In a Multilayer Perceptron, data flows through interconnected layers of neurons. The network's input layer receives the data from MLPs. To ensure that the signal only travels in one direction, the strata of neurons are connected in a graph. The weights between the main layer and the concealed layers are used by MLPs to calculate the input. Activation functions, such as ReLU, sigmoid, and tanh, determine which neurons to activate, adding non-linearity to the model. A training data set is used by

MLPs to train the model to recognize correlations and discover dependencies between independent and target variables [53]. Multilayer Perceptrons (MLPs) are versatile and powerful machine learning tools that can be applied in various fields. This section reviews key applications of MLP and its contribution to advancements in several fields. in Image Recognition and Classification MLP is widely used in image recognition tasks such as object detection, face recognition, and digit classification. By processing pixel values as input features, MLPs can learn to recognize patterns and features in images, making them an important part of computer vision systems. it's can be applied to natural language processing (NLP), including sentiment analysis, language translation, and text classification. By processing text data into sequences of words or characters, MLP can learn to extract semantic information and understand the underlying meaning of the text content. In healthcare and biomedicine, MLPs are used for disease diagnosis, medical image analysis, and drug development. By analyzing patient data, medical images, and genetic sequences, can also help healthcare professionals make accurate diagnoses and develop personalized treatment plans. in Robotics and Autonomous Systems, MLP is used for sensor fusion, motion planning, and control tasks in robotics and autonomous systems. MLP enables robots and autonomous vehicles to navigate complex environments and perform tasks autonomously by processing sensor data and environmental inputs [56]. and in speech emotion recognition, Multi-Layer Perceptron model is considered to the suitable model for speech emotion recognition. This is because, MLP classifier is best suitable for complex structures datasets when compared to other models like SVM, KNN etc. MLP classifier is an in-built classifier model available as default in scikit learn library which contains many in-built classifier models. The output dataset from the feature extraction module is given as input to MLP classifier model [57].

2.4.2.5 Convolutional Neural Network (CNN)

CNNs, often referred to as ConvNets, have several layers and are mostly used for object detection and image processing. When it was still known as LeNet in 1988, Yann LeCun created the first CNN. It was used to identify characters such as ZIP codes and numbers. The identification of satellite photos, processing of medical images, forecasting of time series, and anomaly detection all make use of CNNs. The processing and feature extraction levels in CNNs are numerous :

- Convolution Layer A convolution layer on CNN uses a number of filters to carry out the convolution operation.
- ReLU layers in Rectified Linear Unit (ReLU) CNNs are used to manipulate elements. An improved feature map is the result.
- Layering Pools The pooling layer receives the adjusted feature map after that. Pooling is a downsampling procedure that shrinks the feature map's dimensions.
- The pooling layer flattens the two-dimensional arrays produced by the convolutional feature map into a solitary, lengthy, continuous linear vector.
- Fully Connected Layer The pooling layer's flattened matrix, which is provided as an input, creates a fully linked layer that categorizes and labels the images [54].

Applications of CNN :

- Image Classification : Image classification is a major business application of convolutional neural networks (CNNs) because it enables computers to automatically categorize and understand visual content, which has numerous applications in a wide range of industries.
- Face Recognition : Face recognition is a subset of image recognition that specifically focuses on detecting and identifying human faces within images or videos.

- Recommender Systems : Another application of CNN is recommendation engines, which use image data to recommend products or services to customers[61].

2.5 Difference between Machine Learning and Deep Learning

Machine Learning and Deep Learning are the two main concepts of Data Science and the subsets of Artificial Intelligence. Most of the people think the machine learning, deep learning, and as well as artificial intelligence as the same buzzwords. But in actuality, all these terms are different but related to each other. The key differences between these two terms based on different parameters :

- Data Dependency : Although machine learning depends on the huge amount of data, it can work with a smaller amount of data, Deep Learning algorithms highly depend on a large amount of data, so we need to feed a large amount of data for good performance.
- Execution time : Machine learning algorithm takes less time to train the model than deep learning, but it takes a long-time duration to test the model, Deep Learning takes a long execution time to train the model, but less time to test the model.
- Type of data : Machine learning models mostly require data in a structured form. Whereas Deep Learning models can work with structured and unstructured data both as they rely on the layers of the Artificial neural network.
- Hardware Dependencies : Since machine learning models do not need much amount of data, so they can work on low-end machines while the deep learning model needs a huge amount of data to work efficiently, so they need GPU's and hence the high-end machine.
- Feature Engineering :Machine learning models need a step of feature extraction by the expert, and then it proceeds further on the other hand Deep learning is the enhanced version of machine learning, so it does not need to develop the feature extractor for each problem; instead, it tries to learn high-level features from the data on its own.
- Problem-solving approach : To solve a given problem, the traditional ML model breaks the problem in sub-parts, and after solving each part, produces the final result. The problem-solving approach of a deep learning model is different from the traditional ML model, as it takes input for a given problem, and produce the end result. Hence it follows the end-to-end approach [?].

2.6 Challenges in Deep Learning

Here are some of the challenges in deep learning :

- Data requirements : One of the primary requirements for deep learning is data. Data is the most critical component in constructing a highly accurate model. In several cases, deep neural networks typically require large amounts of data in order to prevent overfitting and perform well. The data requirements for object detection tasks might require more data for a model to detect different objects with high accuracy.
- Computational resources : Apart from a large amount of data, one must also consider the high computational cost of computing the deep neural network. Models like the Generative Pre-trained Transformer 3 (GPT-3) have 175 billion parameters, for example. The compilation

and training of models for complex tasks will require a resourceful GPU. Models can often be trained more efficiently on GPUs or TPUs rather than CPUs. For extremely complex tasks, the system requirements range higher, requiring more resources for a particular task.

- Trainings issus : During training, the model might also encounter issues such as underfitting or overfitting. Underfitting usually occurs due to a lack of data, while overfitting is a more prominent issue that occurs due to training data consistently improving while the test data remains constant. Hence, the training accuracy is high, but the validation accuracy is low, leading to a highly unstable model that does not yield the best results [62].

- Interpretability : Deep learning models can be complex and difficult to interpret. This can make it difficult to understand how the model is making its predictions and to identify potential biases in the model [64].

2.7 Conclusion

Lastly, we present a thorough analysis of deep learning (DL) in this chapter, emphasizing how it has emerged as a revolutionary branch of machine learning (ML). By investigating several deep learning models and architectures, such as : B. Generative Adversarial Networks (GAN), Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNN). Deep learning is still driving innovation and improvement in a variety of fields, including banking, healthcare, law enforcement, and more, despite obstacles with data needs, interpretability, and computational power. Deep learning is becoming more widely available and provides exciting prospects to address challenging issues and have a big societal impact. This is due to advancements in processing power and the availability of massive data sets.

3

Conception

Content

3.1 Introduction 26
3.2 Overall Architecture 26
3.3 Baby supervises 32
3.4 Cry detection and classification 33
3.5 Conclusion 39

3.1 Introduction

Monitoring and understanding the needs of infants are crucial to their well-being and development. With advances in artificial intelligence (AI) and sensor technologies, it is now possible to create sophisticated systems that can continuously monitor babies' vital signs, movements and crying. These systems can provide valuable information to caregivers and enable rapid intervention when needed. The application of AI in this area opens up new perspectives for the classification of baby crying, making it possible to accurately determine specific needs such as hunger, discomfort, or fatigue. In this chapter, we delve into the detailed architecture and functionalities of the proposed baby bed system. We begin by explaining the overall system design and its components, such as the digital twin, AI services, and the end-user application. We then focus on the digital twin concept, detailing its construction and the role of various sensors. Following this, we discuss the AI services layer, including the dataset used for training cry classification models and the implementation of genetic algorithms for hyperparameter optimization. Finally, we explore the use of Multi-Layer Perceptron (MLP) and hybrid LSTM+CNN models for baby cry classification.

3.2 Overall Architecture

The system architecture comprises an AI-powered baby bed system that monitoring and analyzing baby data on movement, vital signs, and sleep through a collection of sensors, its AI algorithms, and a cloud base platform. The system is composed of : a physical bed with sensors inside, a digital twin ; carrying the data coming from the bed, AI services to detect and classify different baby cries (e.g., hungry, hot, cold, uncomfortable...), a database where the acquired data is stored, and an End User App to let parents follow up on their kid well-being indicators on their mobile device. The proposed architecture is shown in Figure 3.1

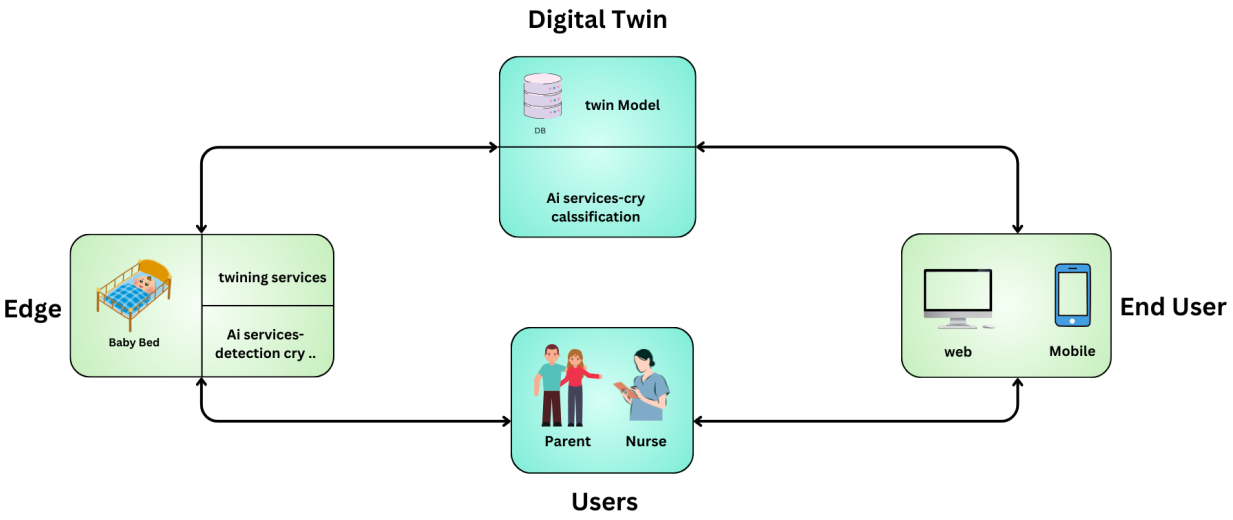


FIG. 3.1: Overall Architectur

To illustrate the interaction between the system and its user, we explain the system's key actors :

- **Caregiver** : The caregiver is the primary user of the system, is the person responsible for overseeing and caring for the baby. They interact with the system through a dedicated caregiver app.
- **Baby** : The individual being monitored by the system. The baby is placed in a specially designed bed with sensors to collect data about him.

In the (figure 3.2) a use case diagram representing the actors and their functionalities and their interaction with the suggested solution For Child Care and supervision.

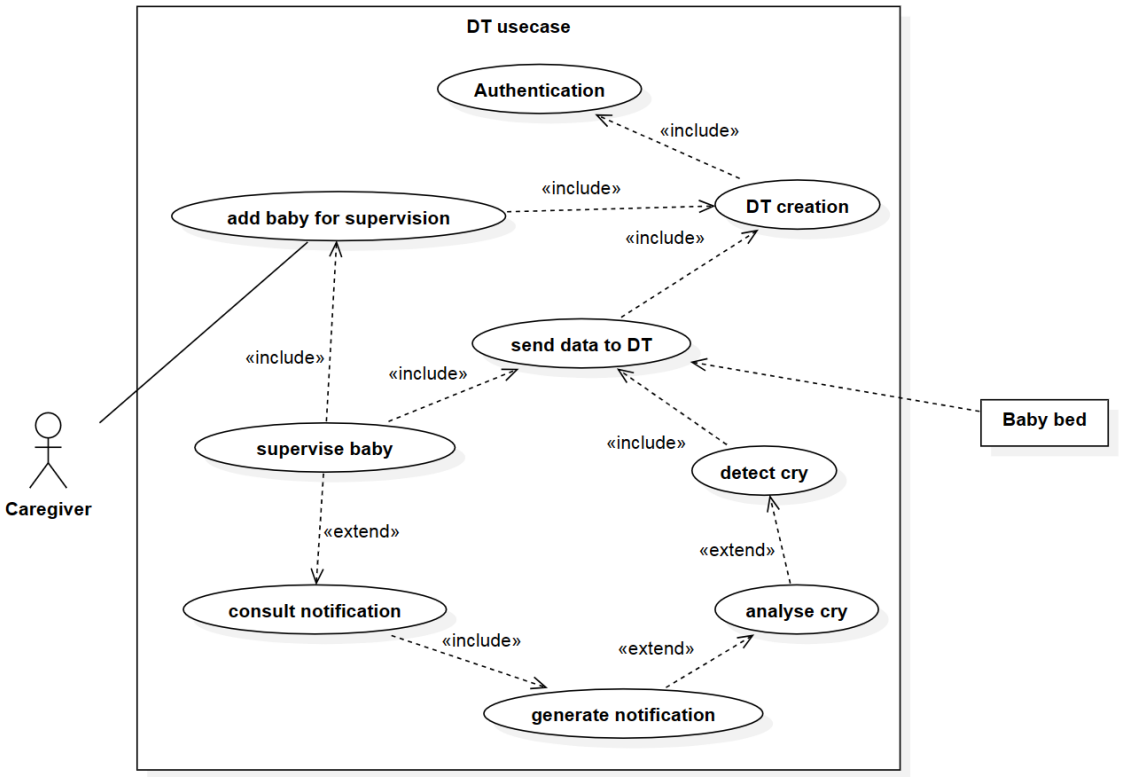


FIG. 3.2: Use Case Diagram

The technique for adding a baby to a database and enabling its supervision is depicted in this diagram. The only actor interacting with the system is a caregiver. To guarantee safe access, the caregiver must first authenticate themselves. They then start the process of adding the infant for oversight. This causes the system to create the baby's virtual profile, or digital twin (DT). The system captures the cries of the baby and interpret them to determine if they are caused by hunger or discomfort. The technology provides insights from the cry analysis and notifies the caregiver based on these capabilities.

3.2.1 Digital Twin

designing the digital twin of a baby bed's entails building a virtual model of the bed together with all of its connected sensors, actuators, and interfaces. This virtual representation is created using technology that provides the model framework and manages the digital twin. A "Thing" is the primary thing used to represent the digital twin.

- **Things** are very generic entities and are mostly used as a "handle" for multiple features belonging to this Thing [70].
- **A Feature** is used to manage all data and functionality of a Thing that can be clustered in an outlined technical context. For different contexts or aspects of a Thing different Features can be used which are all belonging to the same Thing and do not exist without this Thing [71].

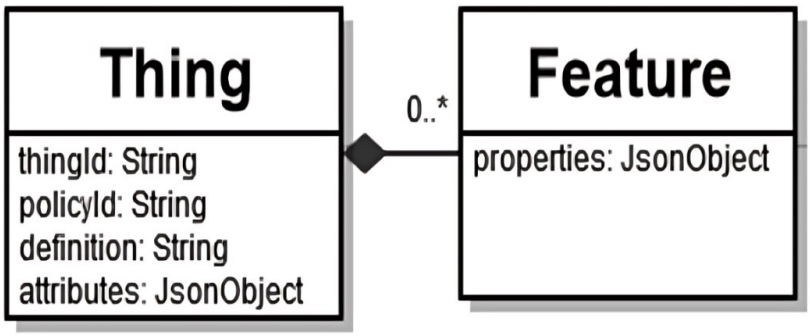


FIG. 3.3: Class Diagram of Ditto [72].

Things are identified by :

- **ThingId :** Unique identifier representing the thing [70].
- **PolicyId :** Links to the ID of an existing Policy which contains the authorization information applied for this thing [70].
- **Definition** : A Thing may contain a definition. The definition can also be used to find Things. The definition is used to link a thing to a corresponding model defining the capabilities/features of it [70].
- **Attributes :** describe the Thing in more detail and can be of any type. Attributes can also be used to find Things. Attributes are typically used to model rather static properties at the Thing level. Static means that the values do not change as frequently as property values of Features [70].

The thing in our case is a "baby" identified by :
ThingId is a unique identifier for the data object, PolicyId is a unique identifier for the security policy. and definition, the definition key defines the type of the thing, can also be used to find it. And its attributes are the name of the baby (first name and last name), its birthday, And the caregiver responsible for him. We define the features of the baby, such as its microphone sensor, blood pressure sensor, heart rate, Co2 and temperature sensor. The microphone has a property which indicates whether or not the baby is crying, the heart rate sensor provides valuable data about the baby's physiological state like discomfort, pain, or even illness. Sudden changes in heart rate might suggest baby's stress or discomfort due to factors like hunger, teething, or environmental changes. The CO2 sensor keeps an eye on the amount of carbon dioxide in the baby's surroundings to guarantee that the air quality is safe. If CO2 levels become too high, it notifies caretakers, urging them to increase ventilation and keep the baby's environment safe. The temperature sensor has a property which indicates the baby’s temperature. In this section, we represent the relationships and interactions between the components of this system, providing an overview of the key elements involved in the functionality of a baby bed.
The figure 3.4 represents a block diagram of a digital twin system for a baby bed

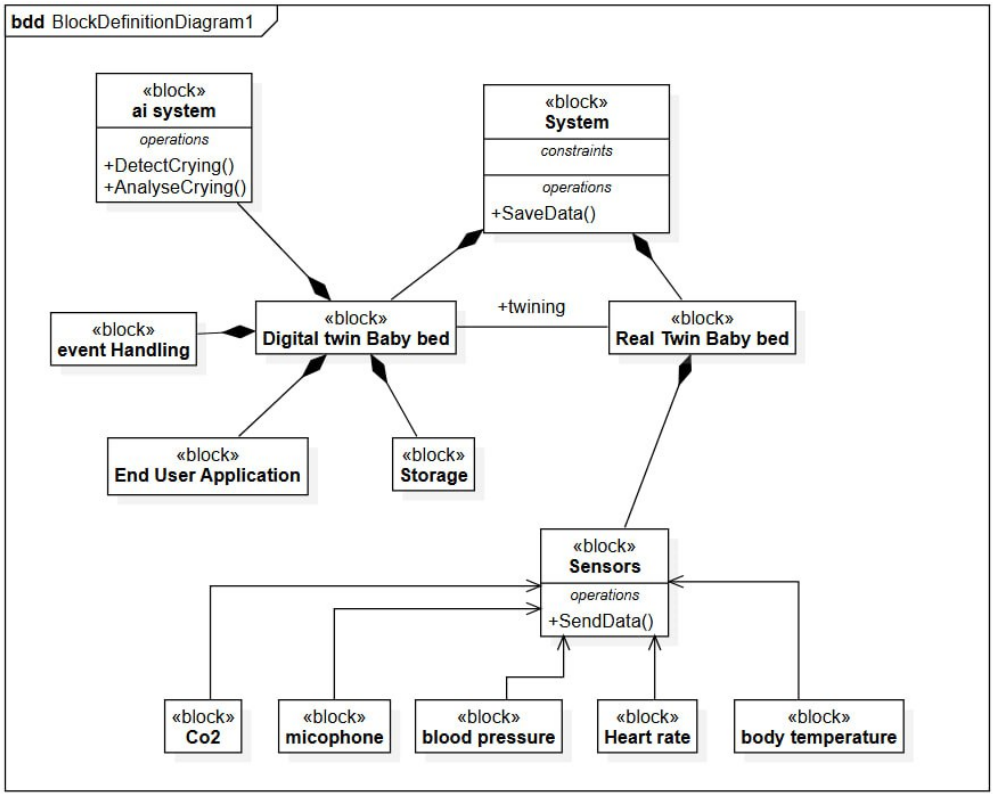


FIG. 3.4: Block Diagram

The block diagram shows the main components of a digital twin system for a baby bed. These components include :

- **Digital Twin of Baby bed :** Represents the physical baby bed that is being represented by the digital twin. The baby bed is equipped with sensors that Stores sensor data, AI analysis results, and other relevant information. Provides a digital replica of the baby crib's state and conditions.
- **Sensors :** Collect data on the baby's environment, including temperature, heart rate, blood pressure, and Collect data on the baby's environment, including temperature, heart rate, blood pressure, and Microphone which Records sounds from the baby bed, capturing crying sounds.
- **Artificial Intelligence (AI) System :** Utilizes machine learning algorithms to Detect crying sounds from the recorded audio and analyze the crying patterns to identify potential causes (e.g., hunger, discomfort, boredom).
- **End User Application :** Provides a user interface for parents or caregiver to access and interact with the system, Displays real-time data from the sensors and AI analysis. also Sends notifications in case of potential issues or concerns and allows parents to personalize baby care based on the collected data.

3.2.2 Synchronization services

Sensors continuously collect data from the baby's environment. The microphone records sound from the baby bed, capturing crying sounds and other audio cues. Sensor data and microphone recordings are sent to the AI system. Upon receiving the data, the AI system analyzes the crying sounds and provides insights into the baby's needs or discomfort. Moreover, sensor data, AI analysis results, and other relevant information are stored in the digital twin of the baby bed. Finally, parents or guardians can access the end-user application to view real-time data, receive notifications, and personalize baby care based on the insights gained from the system.

3.2.3 Artificial Intelligence Services

This layer comprises artificial intelligence (AI) services responsible for data analysis, pattern recognition, and decision-making. which includes a variety of AI technologies such as machine learning algorithms. These algorithms are used to analyze data and identifying potential issues, we employed two models, MLP and CNN-LSTM (In section IV, we will discuss its specifics.) in order to forecast the meanings behind children's cries and to generate recommendations for caregivers. We used it on two different kinds of databases : balanced and imbalanced.
AI APIs are application programming interfaces that utilize artificial intelligence to provide advanced capabilities beyond typical request-response APIs. While standard APIs allow applications to exchange data, AI APIs take things a step further by using machine learning and natural language processing to understand requests, generate relevant responses, and complete tasks [76]. AI planning tools use time series data to estimate future developments for many industries, such as healthcare with the advent of AI application programming interfaces (APIs) that allow other software applications to interact with and utilize the capabilities of the machine learning model [77]. Some common features of AI APIs include :

- Voice capabilities : Accept and respond to voice inputs.
- Machine learning : Improve responses over time based on user interactions.
- Context awareness : Understand user context and intent to provide relevant results.
- Recommendations : Recommend next best actions, products, content etc. based on data [76].

The baby cry detection and classification process involve a seamless interplay between the Sensor, Digital Twin, Junction Layer, Detection Ai Api 1, and Ai Api 2, as depicted in (Figures 3.5) and (Figure 3.6). The Sensor continuously captures sound data from the baby's environment, which is then processed by the Junction Layer. This processed data is sent to the Detection Ai Api to determine whether a baby is crying. If a cry is detected, the Detection Ai Api notifies the Digital Twin, which then sends the cry data to the Ai Api for classification. The Ai Api analyzes the cry and sends back the classification results to the end user app and save the cry data and classification results for future reference. The end user plays a crucial role in monitoring the system's output, and providing feedback to improve its accuracy.

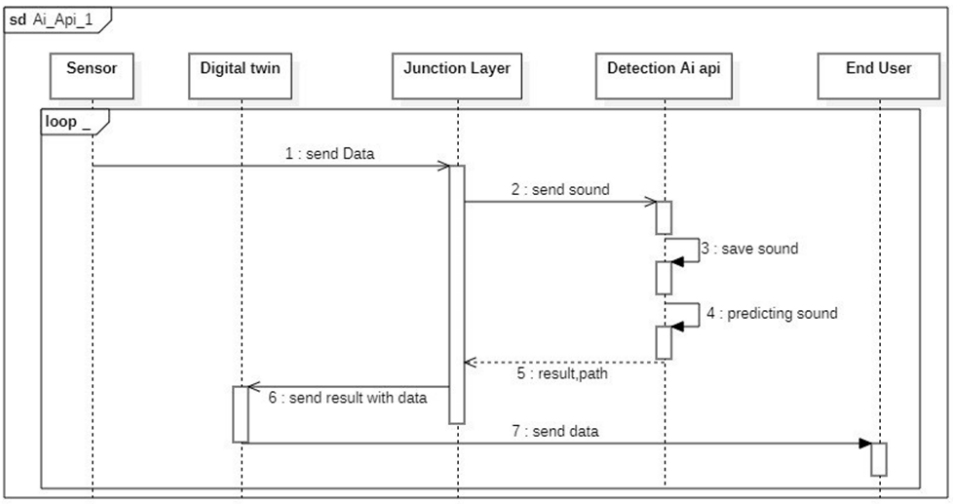


FIG. 3.5: Sequence Diagram for Baby Cry Detection API Interaction

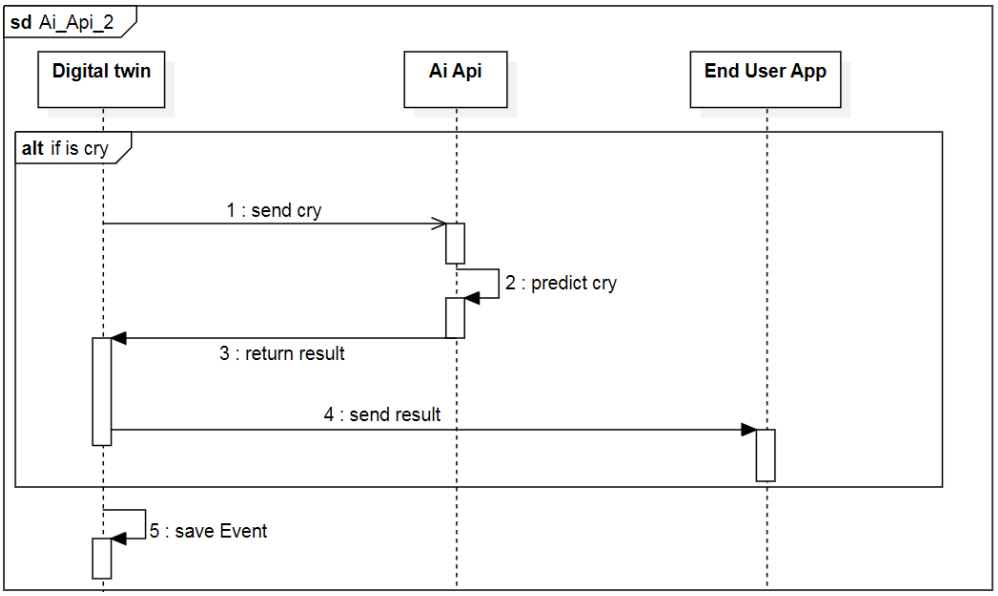


FIG. 3.6: Sequence Diagram for Baby Cry Classification

3.3 Baby supervises

In the context of infant care, ensuring constant supervision and timely responses to a baby's needs is paramount. The Baby Supervise system leverages advanced technology to monitor and analyze various aspects of a baby's environment and actions, giving caregivers vital information and alerts. This section explores the main elements and features of the system, such as the digital twin generation process, authentication procedures, and supervision mechanisms—all aimed at improving the overall experience of providing care.

3.3.1 Create a Digital twin

The sequence diagram provides a clear overview of how to add a baby to the "DT" system, a digital tracking and monitoring system for babies, it illustrates the interactions between the caregiver, the app, the baby bed sensor, and the system's authentication server.
First, the caregiver uses the app to scan a QR code linked to the baby's bed. This code connects the physical bed to its digital counterpart in the system. The system then gathers data from the bed sensor, including temperature, sound, CO2, other sensors. This data helps create a digital representation of the baby's environment. Next, the caregiver inputs the baby's details, such as their name and date of birth, to associate them with the digital twin. If everything goes smoothly, the system generates a special identifier for the baby's bed and confirms the creation of the digital twin. This allows the caregiver to monitor the baby's bed environment, including sensor readings, through the app's interface.
The (figure 3.7) represents create DT sequence diagram :

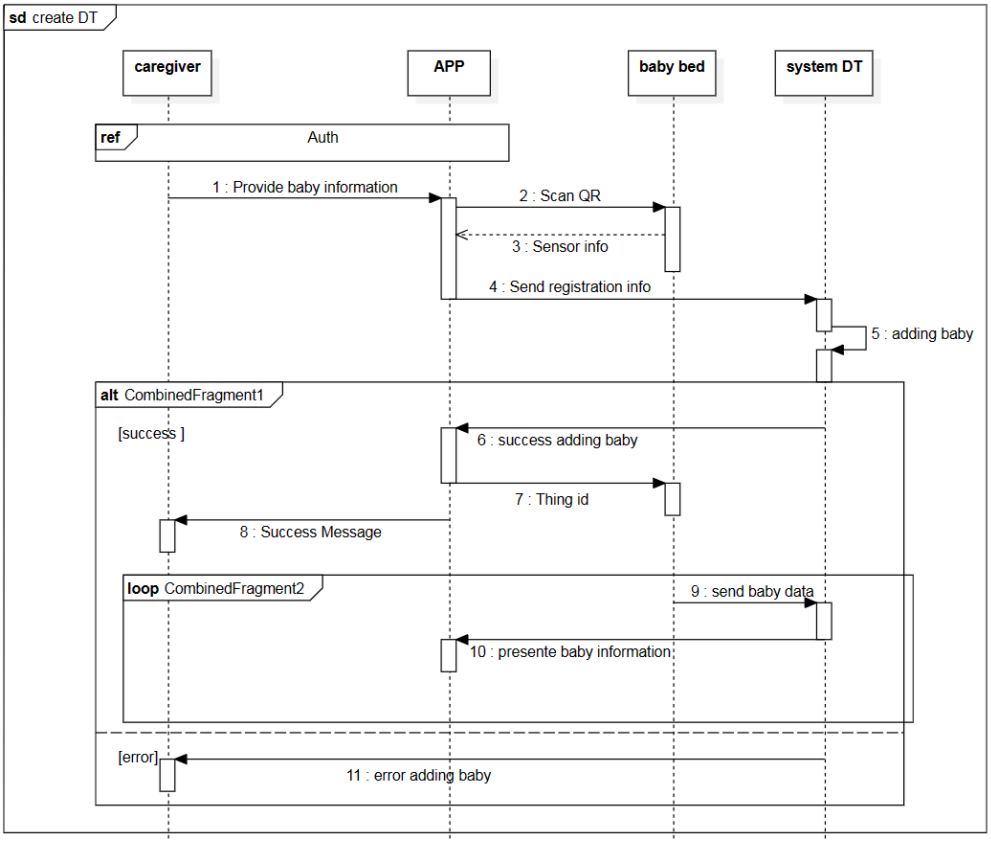


FIG. 3.7: Create DT Sequence Diagram

3.3.2 Data processing

The Sequence diagram depicts information processing in the baby's cry notification system. It starts with the baby crying, which is detected by sensors in the crib that capture sound and movement data. This data is then sent to a monitoring device that analyzes it to determine if it signifies crying. If a crying pattern is identified, the monitoring device sends a notification to the system. The system further examines the crying data to assess its intensity, duration, and potential cause. Upon confirming crying, a notification message is sent to the caregiver's smartphone, alerting them to their baby's distress. The caregiver then responds by checking on the baby, addressing their needs, and potentially adjusting the monitoring settings for future alerts.
The (figure 3.8) represents baby supervision sequence diagram :

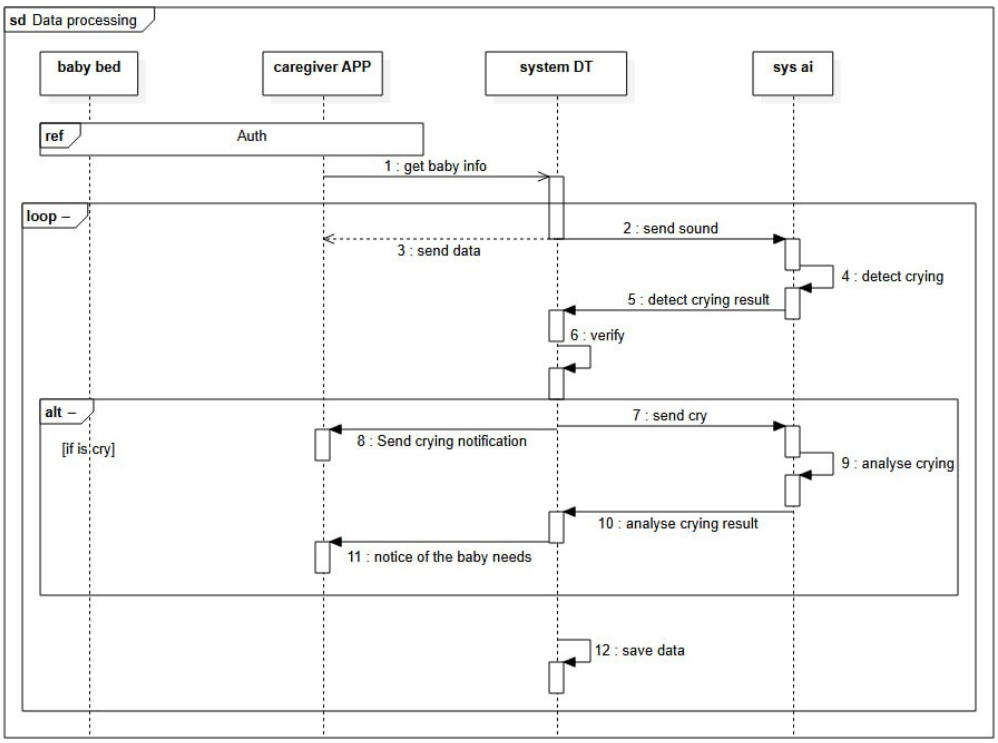


FIG. 3.8: Baby Supervision Sequence Diagram

3.4 Cry detection and classification

In this section, we delve into the methodologies and models employed for detecting and classifying infant cries. This includes data preprocessing, feature extraction, and the implementation of different machine learning models to accurately identify and categorize cry sounds. We focus on using Mel-Frequency Cepstral Coefficients (MFCCs) for feature extraction and balancing the dataset to improve model performance. We explore the application of a Multi-Layer Perceptron (MLP) model and a hybrid model combining Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks. Additionally, we employ genetic algorithms for hyperparameter optimization to further enhance model performance.

3.4.1 Baby cry classification

The first data used in this research work were available in WAV format, contains recordings of cries from children under two years old, aiming to discern their needs based on cry sounds. It is

categorized into five classes : Belly Pain (16 files), Burping (8 files), Discomfort (27 files), Hungry (382 files), and Tired (16 files). The next step is feature extraction, to effectively capture the acoustic characteristics of baby cries, in our approach, we use Mel-Frequency Cepstral Coefficients (MFCCs), as the primary feature extraction technique. MFCCs provide a compact yet informative representation of the audio signal, capturing essential temporal and frequency variations in the cry recordings. By extracting MFCC features from the preprocessed audio data, we create a feature matrix that serves as input for our classification models. Once you've extracted features from the audio files, we organize the data into a DataFrame using pandas. Each row of the DataFrame represents an audio file, with columns for the file path, extracted features (MFCCs). To ensure the effectiveness of the model, the next critical step is balancing the dataset. An imbalance in classes may result in models that are biased, since the classifier may be more likely to predict the class that is in the majority. In order to solve this, we balance the dataset by increasing the number of samples in the minority classes through oversampling using the RandomOverSampler from the imbalanced-learn module.

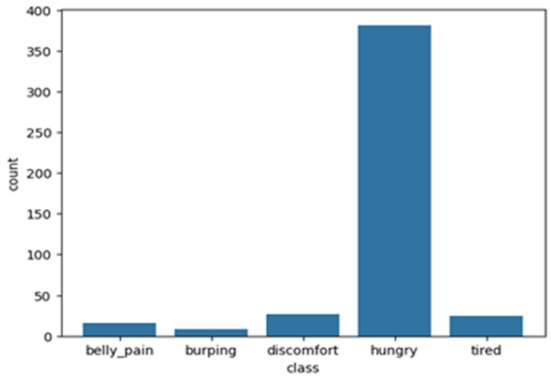


FIG. 3.9: Befor Balancing Data

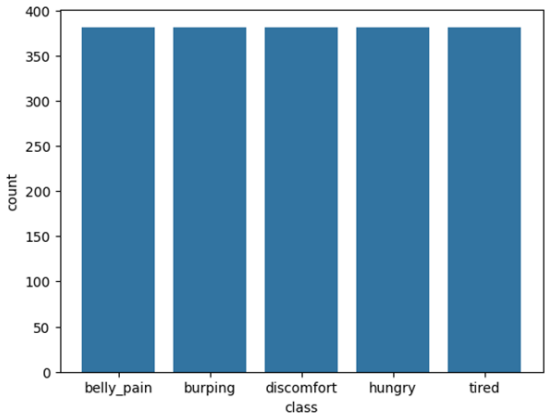


FIG. 3.10: After Balancing Data

3.4.1.1 MLP Model

In this section we explore how a Multi-Layer Perceptron (MLP) could be applied for classifying infant cries and identifying what they need. Multi-layer perceptron (MLP) is a kind of Artificial Neural Network which is good at classification tasks because it has the ability to recognize intricate data patterns. The process begins with data preparation, Where the dataset was split into training set and testing set using an 80-20 split. The features were scaled using StandardScaler to ensure they have a mean of 0 and a standard deviation of 1. The labels are converted into one-hot encoded vectors, which is necessary for the categorical cross-entropy loss function used in multi-class classification.

The architecture includes an input layer with 261 neurons, corresponding to the number of features, two hidden layers with 251 and 68 neurons respectively, using ReLU activation functions, batch normalization layers after the dense layers to improve training speed and stability, and an output layer with 5 neurons (one for each class) and a softmax activation function to produce probability distributions for each class. The model was compiled using the Adam optimizer and the categorical cross-entropy loss function, which is suitable for multiclass classification. To avoid overfitting, a method of early stopping was used ; this adaptive learning rate helps the model converge faster and achieve better results during training. the figure 3.11

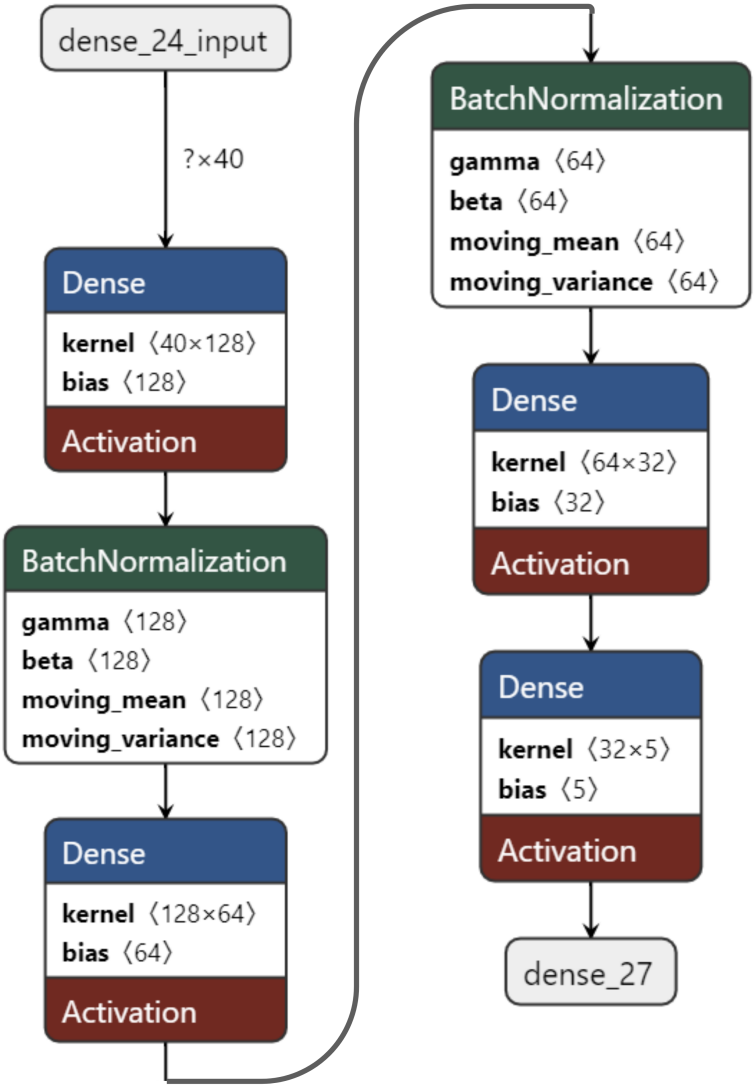


FIG. 3.11: Mlp Architucture for cry classification

3.4.1.2 CNN-LSTM

The other proposition is to use a hybrid model of long short-term memory (LSTM) and convolutional neural network (CNN) to classify infant cries to determine their needs. The combination of LSTM and CNN exploits the temporal and spatial features in the data and is therefore well suited for this classification task. The hybrid CNN-LSTM model was chosen for several reasons : LSTMs are effective at capturing temporal dependencies in sequential data, which is crucial for time-series or sound analysis. CNNs excel at extracting local features and patterns from data, making them useful for processing raw input before passing it to LSTMs. Combining the complementary strengths of CNNs and LSTMs allows the model to capture both spatial (local features) and temporal (sequential patterns) aspects of the data, thus improving classification accuracy. The dataset was scaled and reshaped appropriately for the CNN-LSTM model using StandardScaler, and labels were one-hot encoded. The input features were reshaped to fit the expected input shape for the LSTM layer.
The model architecture comprises a Conv1D layer to extract local features from the input sequences, followed by MaxPooling1D and Dropout layers for downsampling and regularization, respectively. Two LSTM layers are integrated to capture temporal dependencies, supplemented by dropout layers for regularization purposes. Dense layers are then employed to process the combined features extracted by the CNN and LSTM components. Finally, an output layer with 5 neurons and a softmax activation function facilitates multi-class classification. The model was trained with early stopping to prevent overfitting. Early stopping monitors the validation loss and stops training when the model's performance on the validation set stops improving.

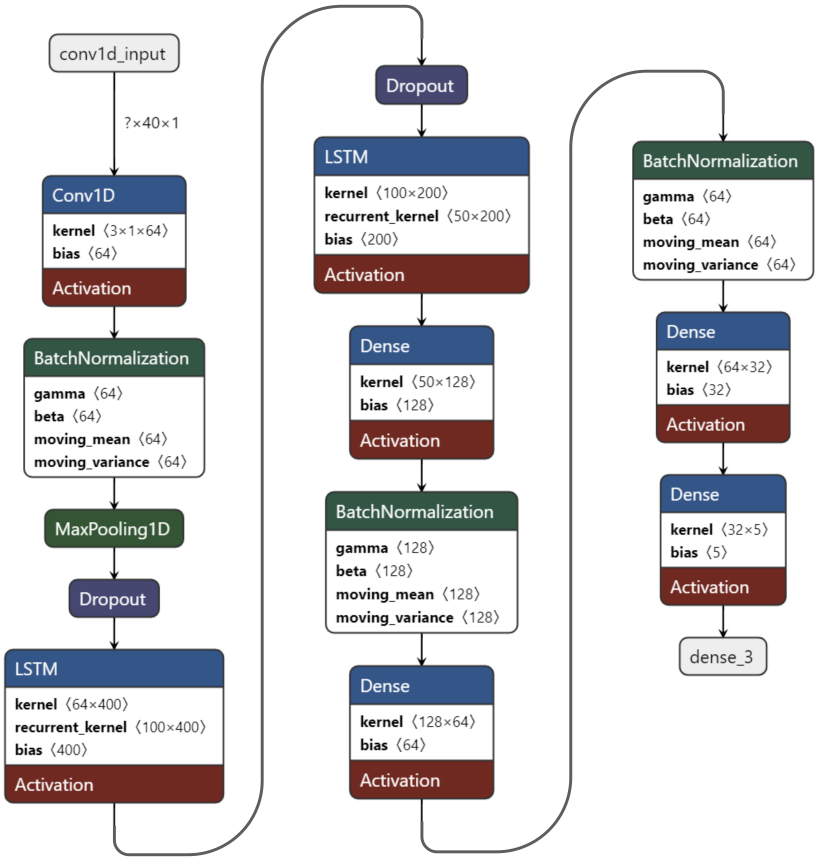


FIG. 3.12: CNN-LSTM Architucture for cry classification

Following this, we split the dataset into training and testing sets using train_test_split from scikit-learn. This ensures that we have separate datasets for training the model and evaluating its

performance. We use 80% of the data for training (X_train, y_train) and 20% for testing (X_test, y_test). After splitting the data, it is essential to standardize the features to have a mean of 0 and a standard deviation of 1. This step is crucial for neural networks as it ensures that the model trains more efficiently. This process is also known as feature scaling.

The standardization formula :

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

where :

z : is the standardized value (the output after scaling).
x : is the original value of the feature.
μ : is the mean of the feature values.
σ : is the standard deviation of the feature values.

3.4.2 Baby Cry Detection

The dataset for infant cry sound includes a total of 750 audio files, initially categorized into four classes—Cry (108 files), Laugh (108 files), Noise (108 files), and Silence (108 files)—the dataset was later restructured by merging Laugh, Noise, and Silence into a single "nocry" category, resulting in Nocry (324 files) and Cry (108 files). The Cry category was then supplemented with additional files from the baby cry detection dataset, bringing its total to 426 files. To effectively capture the acoustic characteristics of baby cries, we use Mel-Frequency Cepstral Coefficients (MFCCs) as the primary feature extraction technique. MFCCs provide a compact yet informative representation of the audio signal, capturing essential temporal and frequency variations in the recordings. The extracted MFCC features are organized into a DataFrame using pandas, where each row represents an audio file with columns for the file path and extracted MFCC features. After extracting features and constructing the DataFrame, we combine the processed DataFrames from both datasets. The combined dataset is then split into training (80%) and testing (20%) sets using train_test_split from scikitlearn. This ensures that we have separate datasets for training the model and evaluating its performance, maintaining the balanced distribution achieved through oversampling. To optimize the training process, particularly for neural networks, it is essential to standardize the features (equation 01)

3.4.2.1 MLP model

The proposed Multi-Layer Perceptron (MLP) model for infant cry detection is designed to classify audio files into "Cry" and "No Cry" categories. The architecture of the MLP model consists of several fully connected (dense) layers interspersed with batch normalization layers to enhance training efficiency and stability. The input to the model consists of extracted audio features, such as Mel-frequency cepstral coefficients (MFCCs), which are fed into the first dense layer. This first dense layer has 244 neurons with a total of 10,004 parameters, applying a linear transformation to the input features. Following this, a batch normalization layer with 976 parameters is applied to normalize the activations, helping to speed up the training process and providing some regularization. The second dense layer contains 158 neurons and 38,710 parameters, providing another level of transformation to the inputs. After this layer, another batch normalization layer with 632 parameters is applied to stabilize and normalize the output. The third dense layer, with 102 neurons and 16,218 parameters, continues the transformation of the input data, followed by a batch normalization layer with 408 parameters to ensure normalized activations. Next, the fourth dense layer includes 87 neurons and 8,961 parameters, further refining the data through linear transformation. This is followed by another batch normalization layer with 348 parameters for normalization purposes. Finally, the

output layer consists of a single neuron with 88 parameters, producing a binary output for classification (either "Cry" or "No Cry"). The MLP model's architecture, with its sequence of dense layers and batch normalization layers, is designed to effectively capture and learn the complex patterns in the audio features that distinguish infant cries from other sounds. Each dense layer progressively transforms the input data, while the batch normalization layers help in stabilizing the learning process, making the model robust and efficient. This architecture is well-suited for the classification task given the dataset of 750 audio files, where it can leverage the learned features to accurately predict the category of each audio file.

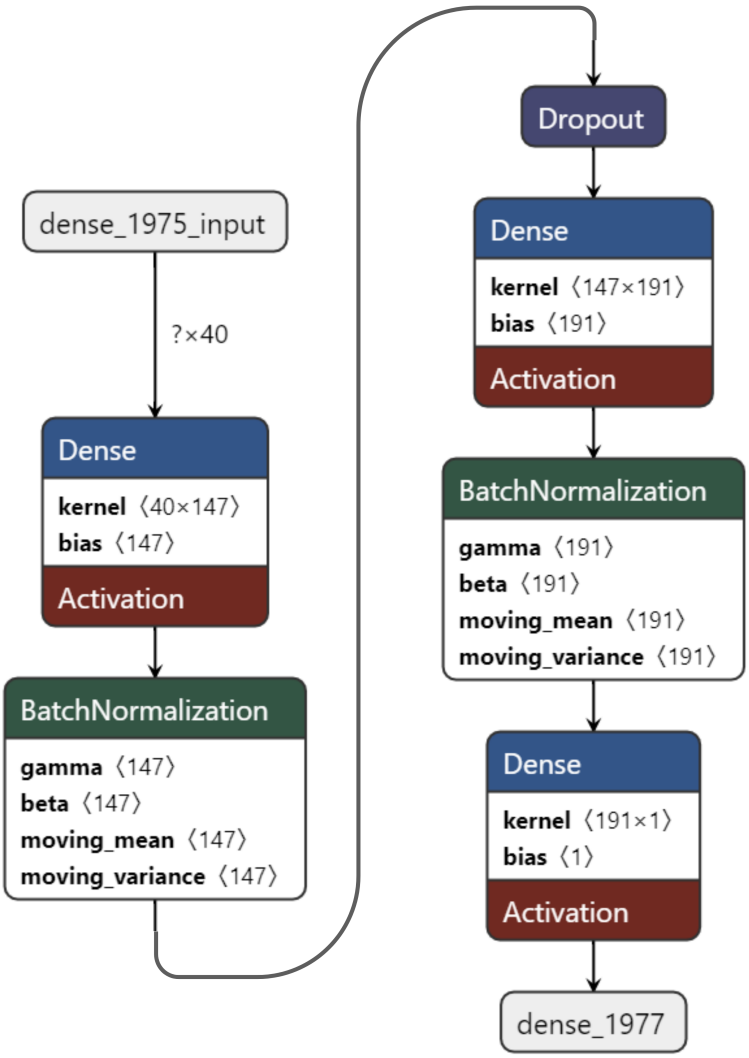


FIG. 3.13: Mlp Architucture for detect cry

3.4.2.2 CNN-LSTM Model

The proposed hybrid model for infant cry detection combines Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks. This architecture leverages the strengths of CNNs in feature extraction and LSTMs in handling sequential data, making it well-suited for audio classification tasks. The input to the model consists of raw audio features, which are processed by the initial Conv1D layer. The first convolutional layer (Conv1D) has 64 filters, each of size 3, resulting in an output shape of (None, 38, 64) and 256 parameters. It performs convolution ope-

rations to extract local features from the input audio data. This is followed by the first max pooling layer, which reduces the dimensionality of the feature maps, outputting (None, 19, 64), thereby reducing computational load and controlling overfitting. The first batch normalization layer, with 256 parameters, normalizes the output of the previous layer to have a mean of 0 and a variance of 1, stabilizing and speeding up the training process. The second convolutional layer (Conv1D) has 261 filters, each of size 3, resulting in an output shape of (None, 17, 261) and 50,373 parameters. It further extracts more complex features from the previous layer's output. This is followed by the second max pooling layer, which reduces the dimensionality of the feature maps further, outputting (None, 8, 261), continuing to reduce computational load and help in feature abstraction. The second batch normalization layer, with 1,044 parameters, normalizes the output of the second Conv1D layer to ensure normalized activations for stable and efficient training. A flatten layer then converts the 2D feature maps into a 1D vector of size 2088, preparing the data for the LSTM layers. The repeat vector layer adds a time step dimension, resulting in an output shape of (None, 1, 2088). The first LSTM layer has 100 units and 875,600 parameters, with the return sequence set to true, capturing the temporal dependencies in the sequential data. The second LSTM layer, with 100 units and 80,400 parameters, continues to learn temporal dependencies and outputs a sequence of fixed size. The first dense layer, with 251 neurons and 25,351 parameters, performs a linear transformation on the LSTM output to reduce dimensionality. A dropout layer introduces dropout regularization to help prevent overfitting by randomly setting a fraction of input units to 0 during training. Finally, the output layer (dense) has 1 neuron and 252 parameters, producing the final binary classification output, which is the predicted probability for the "Cry" class. This hybrid model's architecture effectively captures and learns complex patterns in audio features, distinguishing infant cries from other sounds through its combination of CNN and LSTM layers. Each layer progressively transforms and refines the input data, ensuring robust and efficient learning for accurate classification.

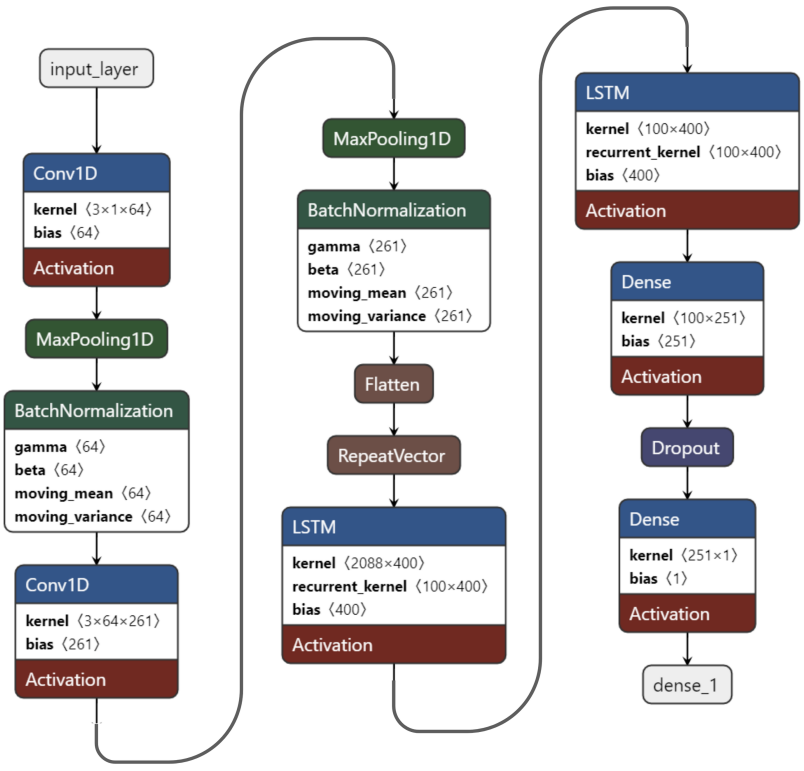


Fig. 3.14: LSTM+CNN Architucture for detect cry

3.4.3 Hyperparameter Optimization with Genetic Algorithms

The genetic algorithm is a kind of a heuristic algorithm inspired by the theory of evolution. It is widely used in search problems and optimization problems [78],[80]. By performing biological heuristic operators such as selection, mutation and crossover. The GA becomes a useful tool in many areas [78]-[80]. The genetic algorithms can be presented in the (Figure 3.15) :

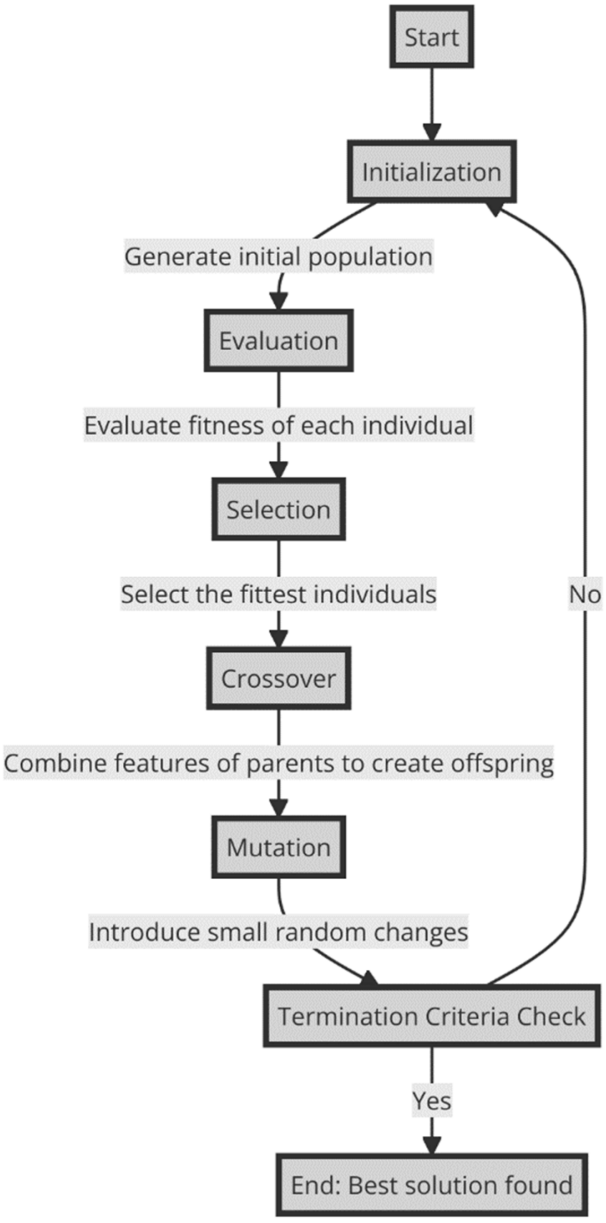


FIG. 3.15: Genetic Algorithms Process [79]

These algorithms are based on several key components, which can be cited as follows :

- Initialization : A population of potential solutions is generated. These solutions are usually represented as strings of binary numbers, but they can also be other data structures.
- Evaluation : The objective function, represented by f(x) = y, evaluates the performance of individuals about the defined criterion. Each member of the population is evaluated based on a fitness function. The fitness function determines how close a given solution is to the optimum solution of the problem being solved.

- Selection : This step selects individuals or solutions based on their fitness. The better the solutions, the more chances they have to be selected for reproduction. Selection operators based on techniques such as roulette wheel selection, tournament selection, or rank selection are commonly used to determine the probability of selection of individuals for reproduction.
- Crossover : Also known as recombination, this step combines the genetic information of two parents to generate new offspring. It aims to produce a better solution by combining the best traits of the parents.
- Mutation : This step introduces random changes to the offspring's genetic makeup to maintain genetic diversity within the population, helping to prevent the algorithm from becoming too homogeneous and getting stuck in local optima.
- Termination Criteria Check : This step determines whether the algorithm has met its termination criteria, which could be a specific number of generations, a plateau in fitness levels, or a satisfactory fitness level.
- End : If the termination criteria are met, the algorithm ends, and the best solution found is returned [81].

3.4.3.1 Genetic Algorithm Implementation for Neural Network Optimization

Each individual in the population represents a potential neural network architecture. The individuals are encoded as lists of integers, where each integer specifies the number of units in a layer or the configuration of a specific network component. For the MLP model, the individual represents the number of units in each hidden layer. For the LSTM+CNN model, the individual includes the number of filters in the Conv1D layer, the number of units in the LSTM layers, the number of units in the Dense layer, and the dropout rate. The fitness evaluation function constructs, trains, and evaluates the neural network based on the architecture defined by an individual. The steps include : Model Construction, which builds the neural network according to the individual's parameters ; Compilation, which compiles the model with the Adam optimizer and categorical cross-entropy loss function ; Training, which trains the model using early stopping to prevent overfitting ; and Evaluation, which evaluates the model's accuracy on the validation set, which is used as the fitness value. Genetic operators in this context include Crossover (Mating), which combines two parents to produce offspring using two-point crossover ; Mutation, which introduces variability by altering the units or configuration in the network layers, with uniform integer mutation being used ; and Selection, which selects the best individuals for the next generation using tournament selection. The genetic algorithm hyperparameters include a Population Size of 50, Number of Generations of 20, Crossover Probability (cxpb) of 0.5, Mutation Probability (mutpb) of 0.2, and Tournament Size of 3.

3.4.3.2 Optimal Neural Network Architectures Identified by Genetic Algorithms

The best MLP architecture found by the GA might be [128, 64, 32], indicating three hidden layers with 128, 64, and 32 units respectively. This architecture achieves a high accuracy on the validation set, demonstrating the effectiveness of the GA in finding optimal configurations. Similarly, the best LSTM+CNN architecture might be [64, 128, 32, 0.3], representing 64 filters in the Conv1D layer, 128 units in the LSTM layer, 32 units in the Dense layer, and a dropout rate of 0.3. This model also achieves high accuracy on the validation set.

3.5 Conclusion

Finally, this chapter details the architecture and components of the smart baby monitoring system. By integrating advanced sensors and artificial intelligence technologies, the system aims to improve the quality of infant care. The design of digital twins enables centralized data collection and in-depth analysis, providing valuable information about the child's condition. The importance of AI services in baby cry detection and classification has been highlighted by using advanced data processing and machine learning techniques. The use of genetic algorithms to optimize the hyperparameters of neural network models has been shown to be effective in improving the accuracy of these models. Classification models, whether multi-layer perceptron (MLP) or hybrid LSTM+CNN, have demonstrated their ability to identify different types of cries, helping to respond quickly and appropriately to the needs of infants.

4 Implementation

Content

4.1 Introduction 41
4.2 Development Environment 41
4.3 Implementation 43
4.4 Realization Twining Services 50
4.5 Mobile App 51
4.6 Conclusion 55

4.1 Introduction

The "Baby Supervise" project aims to create an advanced digital twin system that allows for the early identification of a baby's needs through the recognition of their cries. This system integrates multiple components, including deep learning models, IoT sensors, middleware, and easy-to-use application, to provide a comprehensive solution for real-time baby monitoring. The core objective is to leverage state-of-the-art technologies to enhance the caregiving process, ensuring that any discomfort or need of the baby is promptly and accurately identified and addressed. Our method comprised building strong deep learning models that could tell the difference between cries and non-tears in addition to pinpointing the exact reasons for the cries. To attain high accuracy and avoid overfitting, these models were trained and refined through the use of genetic algorithms and a sizable dataset. A digital twin system (DITTO) in a Docker environment was utilized to handle the data flow from sensors to the AI models and the end user. Node-RED was included into the DITTO to facilitate smooth data processing.

4.2 Development Environment

4.2.1 Google Colab

Google Colab is a free cloud service that supports Jupyter notebooks and provides free access to GPUs and TPUs, making it ideal for machine learning and deep learning tasks. It allows developers to write, run, and debug Python code in a web-based interface, with the added benefit of leveraging powerful hardware acceleration. In this project, Google Colab was used extensively for developing, training, and testing the baby cry classification models, enabling rapid prototyping and iteration with ease.

4.2.2 Python

Python is a high-level programming language known for its readability and support for multiple programming paradigms, including procedural, object-oriented, and functional programming. Its extensive standard library and community-contributed modules make it a versatile tool for various applications. In this project, Python served as the primary programming language for developing the machine learning models and server-side applications, providing a robust and flexible environment for implementing complex algorithms and integrations.

4.2.3 TensorFlow

TensorFlow is an open-source library for numerical computation and large-scale machine learning, developed by the Google Brain team. It supports a wide range of deep learning and neural network models, enabling efficient training and deployment across different platforms. TensorFlow was used in this project to build and train the neural network models for cry classification and need identification, leveraging its powerful tools and APIs to ensure high performance and scalability

4.2.4 Scikit-learn

Scikit-learn is a machine learning library for Python that provides simple and efficient tools for data mining and data analysis, built on top of NumPy, SciPy, and matplotlib. It includes various algorithms for classification, regression, clustering, and dimensionality reduction. In this project, Scikit-learn was used for preprocessing data, splitting datasets, and evaluating model performance, ensuring that the data pipeline was robust and the models were well-validated

4.2.5 Imbalanced-learn (imblearn)

Imbalanced-learn is a Python package that provides tools for dealing with imbalanced datasets, offering resampling techniques such as oversampling and undersampling. It is designed to work seamlessly with Scikit-learn. In this project, Imbalanced-learn was utilized to handle class imbalances in the dataset, ensuring that the machine learning models were trained effectively without bias towards more frequent classes, thus improving their ability to accurately predict less common labels.

4.2.6 DEAP (Distributed Evolutionary Algorithms in Python)

DEAP is a Python library for rapid prototyping and testing of evolutionary algorithms. It provides a comprehensive set of tools to implement genetic algorithms, genetic programming, and other evolutionary techniques. In this project, DEAP was used to optimize the hyperparameters of the machine learning models, leveraging evolutionary strategies to improve model performance. By using DEAP, the project could efficiently explore a wide range of hyperparameter configurations and identify the most effective settings for the classification tasks

4.2.7 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It provides a robust set of tools for building web applications, including an ORM, authentication mechanisms, and a template engine. In this project, Django was used to host the server-side applications for crying verification and need identification, managing data storage, processing incoming audio data, and integrating machine learning models to provide real-time analysis and notifications.

4.2.8 Node-RED

Node-RED is a flow-based development tool for visual programming, primarily used to wire together devices, APIs, and online services. It provides a browser-based editor that makes it easy to create event-driven applications by connecting pre-built nodes. In this project, Node-RED acted as an intermediary layer that processed data from sensors and communicated with Django servers and Ditto. It facilitated the routing of sensor data, ensuring seamless integration and data flow between various components of the system.

4.2.9 Flutter

Flutter is an open-source UI software development kit created by Google, used to develop natively compiled applications for mobile, web, and desktop from a single codebase. It provides a rich set of pre-designed widgets and tools for building responsive and visually appealing user interfaces. In this project, Flutter was used to develop the mobile application that displayed real-time monitoring data and notifications to caregivers. It provided an intuitive and responsive user interface, enabling users to easily interact with the system and receive timely alerts regarding the infant's needs.

4.3 Implementation

4.3.1 Baby digital twin

The infant monitoring system uses a JSON format to organize and store the collected data, including baby's name, date of birth, microphone data, body temperature, CO2 level, heart rate and blood pressure.
Below, is an example of JSON data representing information collected and monitored for a specific infant.

flow.json > ...

```json
{
    "thingId": "org.eclipse.ditto:d4a7efd9-116f-4d6d-b84d-9df753fe7d0e",
    "definition": "baby:bed:0.0.1",
    "attributes": {
        "firstName": "Baby3",
        "lastName": "Baby3",
        "birthday": "2024-05-03T00:00:00Z",
        "caregiver": 2
    },
    "features": {
        "microphone": {
            "properties": {
                "isCrying": true,
                "audioPath": "/media/audios/audio_2024-06-07-183334.708415.wav"
            }
        },
        "temperature": {
            "properties": {
                "value": 37
            }
        },
        "co2": {
            "properties": {
                "value": 400
            }
        },
        "heartRate": {
            "properties": {
                "value": 60
            }
        },
        "bloodPressure": {
            "properties": {
                "sp": 70,
                "dp": 60
            }
        }
    }
}
```

FIG. 4.1: Eclipse Ditto Platform

This JSON structure shows how the system monitors various aspects of the baby's health and environment, allowing the caregiver to receive alerts and react quickly to the infant's needs.

4.3.2 Model Development

This study focuses on the development and evaluation of deep learning models for early recognition of an infant's needs through cry detection. Two datasets were used : one for distinguishing

between crying and non-crying sounds, and another for identifying the reasons behind crying. We implemented and compared the performance of Multi-Layer Perceptron (MLP) and CNN-LSTM hybrid models for both tasks. The results highlight the efficacy of these models in accurately classifying crying events and determining the specific needs of an infant based on their cries.

4.3.2.1 Crying Detection Model

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 244)	10004
batch_normalization (Batch Normalization)	(None, 244)	976
dense_1 (Dense)	(None, 158)	38710
batch_normalization_1 (Bat chNormalization)	(None, 158)	632
dense_2 (Dense)	(None, 102)	16218
batch_normalization_2 (Bat chNormalization)	(None, 102)	408
dense_3 (Dense)	(None, 87)	8961
batch_normalization_3 (Bat chNormalization)	(None, 87)	348
dense_4 (Dense)	(None, 1)	88

Total params: 76345 (298.22 KB)
Trainable params: 75163 (293.61 KB)
Non-trainable params: 1182 (4.62 KB)

FIG. 4.2: MLP model for cry detection

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 38, 64)	256
max_pooling1d (MaxPooling1 D)	(None, 19, 64)	0
batch_normalization_3 (Bat chNormalization)	(None, 19, 64)	256
conv1d_1 (Conv1D)	(None, 17, 261)	50373
max_pooling1d_1 (MaxPoolin g1D)	(None, 8, 261)	0
batch_normalization_4 (Bat chNormalization)	(None, 8, 261)	1044
flatten (Flatten)	(None, 2088)	0
repeat_vector (RepeatVecto r)	(None, 1, 2088)	0
lstm (LSTM)	(None, 1, 100)	875600
lstm_1 (LSTM)	(None, 100)	80400
dense_6 (Dense)	(None, 251)	25351
dropout_2 (Dropout)	(None, 251)	0
dense_7 (Dense)	(None, 1)	252

Total params: 1033532 (3.94 MB)
Trainable params: 1032882 (3.94 MB)
Non-trainable params: 650 (2.54 KB)

FIG. 4.3: CNN with LSTM Model for Detection Cry

4.3.2.2 Cry classification Model

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 261)	10701
batch_normalization (Batch Normalization)	(None, 261)	1044
dense_1 (Dense)	(None, 251)	65762
batch_normalization_1 (Bat chNormalization)	(None, 251)	1004
dense_2 (Dense)	(None, 68)	17136
dense_3 (Dense)	(None, 5)	345

Total params: 95992 (374.97 KB)
Trainable params: 94968 (370.97 KB)
Non-trainable params: 1024 (4.00 KB)

FIG. 4.4: MLP Model for Cry classification

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 38, 64)	256
batch_normalization_2 (Bat chNormalization)	(None, 38, 64)	256
max_pooling1d (MaxPooling1 D)	(None, 19, 64)	0
dropout (Dropout)	(None, 19, 64)	0
lstm (LSTM)	(None, 19, 100)	66000
dropout_1 (Dropout)	(None, 19, 100)	0
lstm_1 (LSTM)	(None, 50)	30200
dense_4 (Dense)	(None, 128)	6528
batch_normalization_3 (Bat chNormalization)	(None, 128)	512
dense_5 (Dense)	(None, 64)	8256
batch_normalization_4 (Bat chNormalization)	(None, 64)	256
dense_6 (Dense)	(None, 32)	2080
dense_7 (Dense)	(None, 5)	165

Total params: 114509 (447.30 KB)
Trainable params: 113997 (445.30 KB)
Non-trainable params: 512 (2.00 KB)

FIG. 4.5: CNN with LSTM Model for Cry classification

4.3.3 Results

In this section, we report the results of our tests with two different models—the MLP (Multi-Layer Perceptron) model and the LSTM-CNN (Long Short-Term Memory with Convolutional Neural Network) model—to identify the reason behind crying and identify crying noises. We offer a thorough examination of the training and validation outcomes for every model, emphasizing its generalizability, accuracy, and overall performance.

4.3.3.1 Result cause cry

This subsection focuses on the results of our models in determining the cause of crying

- **MLP model**
  The training and validation results for the MLP model in determining the cause of crying demonstrate excellent performance. The training loss consistently decreases over the epochs, indicating effective learning, while the validation loss initially decreases and then stabilizes, suggesting good generalization to unseen data without significant overfitting. The training accuracy shows a steady increase, reaching high levels early in the training process, and the validation accuracy also increases, achieving high and stable values. This close alignment between training and validation accuracy indicates that the model performs well on both the training and validation sets. Overall, the model's final accuracy of around 97% and stable performance across epochs reflect a well-trained model with strong generalization capabilities, making it a reliable choice for determining the cause of crying.

  The (Figure 4.6) represente the accuracy and loss of the MLP Model for Cause Cry.

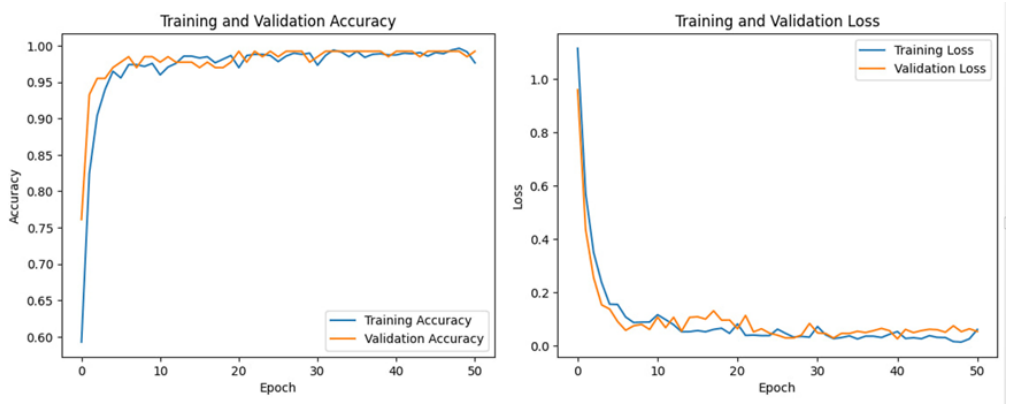


FIG. 4.6: Accuracy and Loss of the MLP Model for Cry classification

Figure 4.7 shows the confusion matrix of MLP model

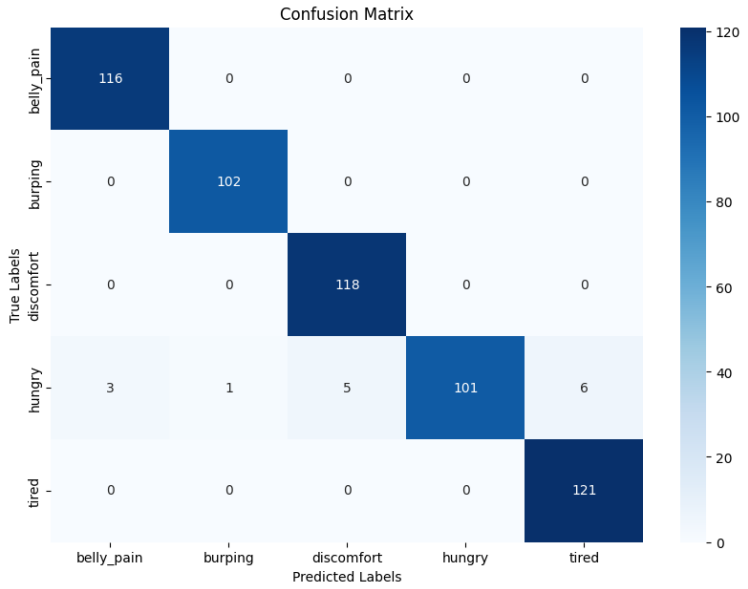


FIG. 4.7: Matrix confision for model MLP for Cry classification

- **CNN with LSTM model**
  The training and validation results for the CNN-LSTM model in determining the cause of crying show a solid performance progression. Initially, the model struggled with high training and validation losses and low accuracy, reflecting difficulty in learning the complex patterns in the early epochs. However, as training progressed, both training and validation losses consistently decreased while accuracy increased significantly. This trend indicates that the model effectively learned and generalized well to unseen data. By the end of the training, the model achieved a high validation accuracy, peaking around 96%, with stable and low validation loss, suggesting strong generalization and effective learning of the input features. The alignment between training and validation metrics also indicates minimal overfitting, showcasing the model's robustness in determining the cause of crying. The (Figure) represente the accuracy and Loss of The CNN with LSTM Model for Cause Cry.



FIG. 4.8: Accuracy and Loss of The LSTM with CNN Model for Cry classification

Figure 4.9 shows the confusion matrix of CNN with LSTM model

While both models perform exceptionally well, the MLP model slightly outperforms the CNN-LSTM model in terms of accuracy (97% vs. 96%). Additionally, the MLP model's simpler architecture allows for faster training and less computational overhead, making it more efficient for this task. Thus, the MLP model is the better choice based on the given results, considering both accuracy and computational efficiency.

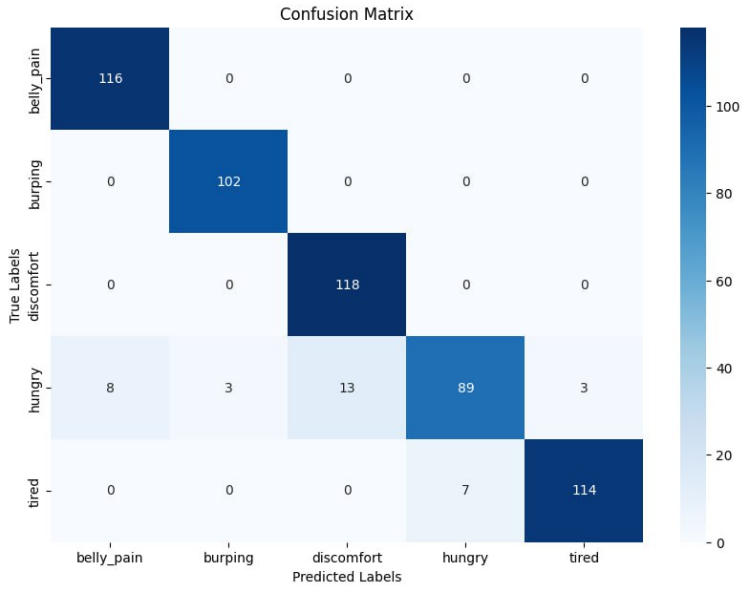


FIG. 4.9: Matrix confision for model CNN with LSTM for cause cry

4.3.3.2 Result detect cry

Here, we analyze the performance of the models in detecting crying sounds.

- **MLP model**
  The MLP model trained on the cry detection dataset exhibits strong performance, achieving an accuracy that steadily improves throughout training. Starting from 78.52% in the first epoch, the model quickly progresses to 90% accuracy by the 4th epoch and maintains high validation accuracy, peaking at 96.67% by epoch 12. The inclusion of batch normalization layers aids in stabilizing and enhancing training, resulting in a reliable model capable of accurately identifying crying sounds. The (Figure 4.10) represente the accuracy and Loss of the MLP Model for detection cry

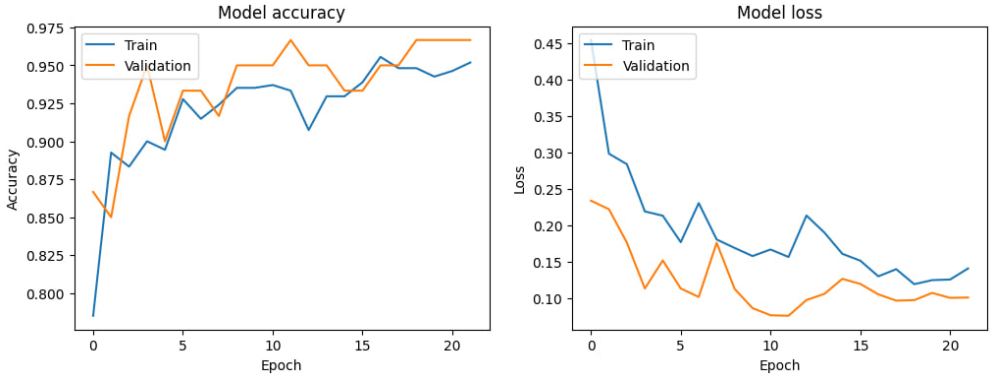


FIG. 4.10: Accuracy and Loss of the MLP Model for Detection cry

Figure 4.11 shows the confusion matrix of MLP model

- **LSTM with CNN model**
  The CNN-LSTM model for cry detection achieves impressive performance, swiftly reaching 100% accuracy by the 10th epoch during training and maintaining 98.33% accuracy during validation. It effectively combines Conv1D and max-pooling layers for feature extraction

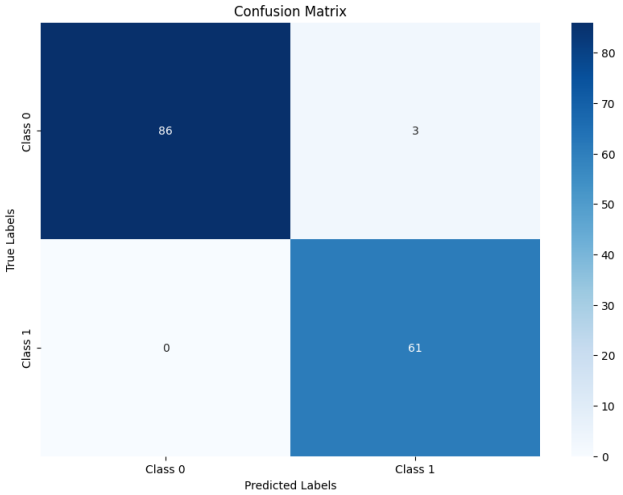


FIG. 4.11: Matrix confision for model MLP for detect cry

and LSTM layers for capturing temporal dependencies. Batch normalization ensures stable training, while dropout regularization enhances generalization. Overall, the model demonstrates robust learning dynamics and high efficiency in detecting cry patterns, making it highly effective for practical applications. The (Figure4.12) represente the Accuracy and loss of the CNN with LSTM model for detection cry

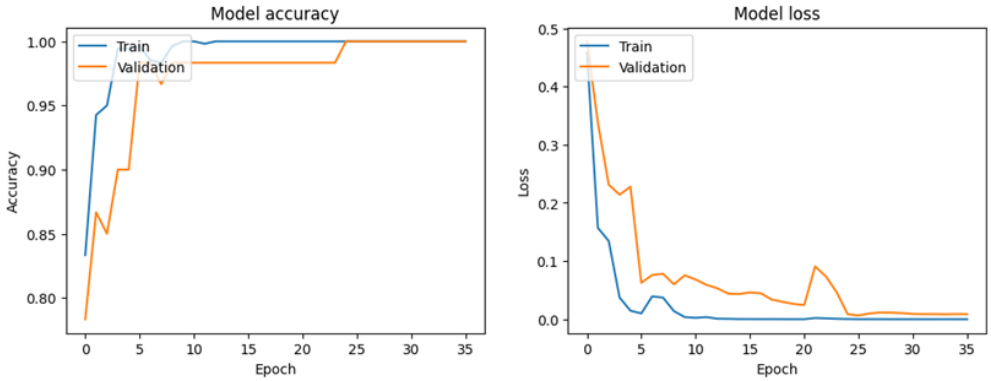


FIG. 4.12: Accuracy and loss of the CNN with LSTM Model for Detection Cry

The Figure 4.13 show the confusion matrix of CNN with LSTM model

The CNN-LSTM model performs better than the MLP model for crying detection. It achieves a higher validation accuracy of 98.33% compared to the MLP model's 96.67%. The CNN-LSTM's architecture, leveraging Conv1D and LSTM layers, allows it to effectively capture both spatial and temporal features, making it more suitable for the sequential nature of cry detection data. Therefore, the CNN with LSTM model is recommended for this dataset due to its superior performance and architecture.

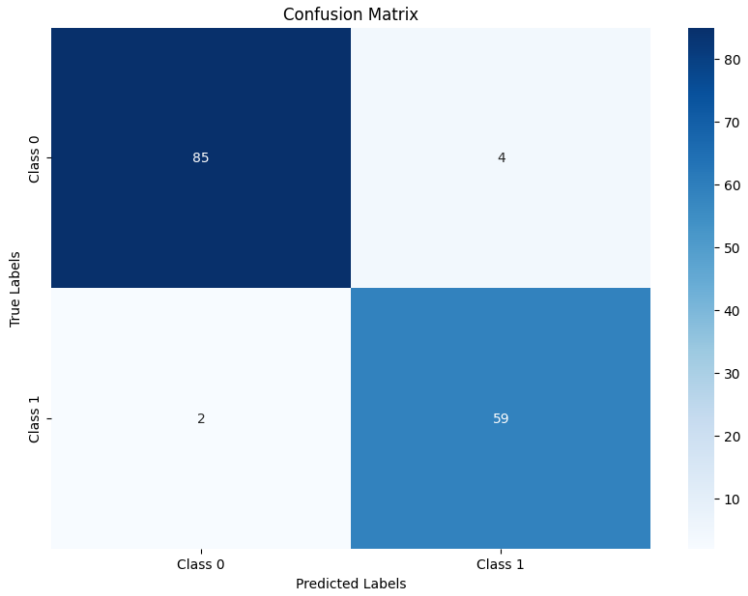


FIG. 4.13: Matrix confision for model CNN with LSTM for detect cry

4.3.4 Performance Evaluation

In this section, we provide a comparative evaluation of the performance metrics for the MLP and CNN+LSTM models in two tasks : determining the cause of crying and detecting crying sounds. The key metrics used for this evaluation are accuracy, precision, recall, and F1-score. These metrics give a comprehensive view of each model's performance, allowing us to understand their strengths and potential areas for improvement. The table below summarizes the performance metrics for both tasks and models.

	Model	Accuracy	Precision	Recall	F1-Score
Cause Cry	MLP	0.97	0.97	0.97	0.97
	CNN+LSTM	0.9424	0.948	0.944	0.94
Detect Cry	MLP	0.98	0.98	0.98	0.98
	CNN+LSTM	0.96	0.96	0.96	0.96

TAB. 4.1: The Performance of suggested models

4.3.5 Training Time Comparison for Genetic Algorithm Optimized Models

The table shows the training time for different generations (6th, 20th, and final) of two neural network architectures (MLP and CNN+LSTM) on two tasks (Cause Cry and Detect Cry). For the "Cause Cry" datasets :

	Model	1 gen	6 gen	20 gen
Cause Cry	MLP	10m :43s	1h :3m	3h :37m
	CNN+LSTM	2h :15m	12h	
Detect Cry	MLP	4m :36s	27m :4s	1h :26m
	CNN+LSTM			

TAB. 4.2: Training duration for each model using genetic algorithm

In general, the more complex CNN+LSTM architecture takes significantly longer to train compared to the simpler MLP, especially for later generations and more complex tasks like "Cause Cry." Training time increases with the number of generations as the genetic algorithm continues optimizing the models.

4.3.6 Implementation of AI services

In this section, we will explore in an integrated way the complete process of detecting and classifying baby crying through several essential steps, The process begins with the sensors capturing audio data and sending it to the junction layer (Node-RED). The junction layer forwards this data to the Detection AI API, upon receiving the audio data from the junction layer, the Detection AI API processes the data. This involves saving the audio file and creating a path for it, conducting training to analyze the sound, and determining whether the baby is crying and returns the result along with the audio file path. The junction layer then sends this information to Ditto, which uses WebSocket to relay the data to the end user. API (AI Api_1) performs a thorough processing including saving the audio file, creating a path for it, as well as model training for sound analysis and determination of the presence of baby crying. If a cry is detected, AI Api_1 returns the audio file path and indicates the presence of crying. Then, the second API (AI Api_2) comes after the first. the AI Api_1 return a result (if is crying), AI Api_2 requests to send the data to our model for further classification. Our model predicts and classifies crying based on the data received. The result of the classification is sent to the end-user application and is also saved.

4.4 Realization Twining Services

We used the Node-RED tool to create connectivity between the actuators, sensors, and physical system. The system's various components could be linked and integrated with ease thanks to this strong tool, which also made it possible to retrieve sensor data and send it to the digital twin for additional processing and communication. The smooth transfer of data between the digital representation of the physical system and its physical counterpart was made possible in large part by Node-RED.
The (Figure 4.14) represent Visualizing Sensor Data from a Baby Crib using Node-RED This Node-RED designed for monitoring a baby and sending relevant data to a platform called Ditto. It involves capturing sensor data (temperature, CO2, heart rate, blood pressure), analyzing audio for cry detection, and sending both sensor and audio data to Ditto. Here is an explanation of some nodes :

- Inject triggers the flow.
- cryingAudio : Represents the audio data captured by a microphone in the baby's crib.
- setDataInToContext : Stores sensor data (temperature, CO2, heart rate, blood pressure) in the context use or visualization.

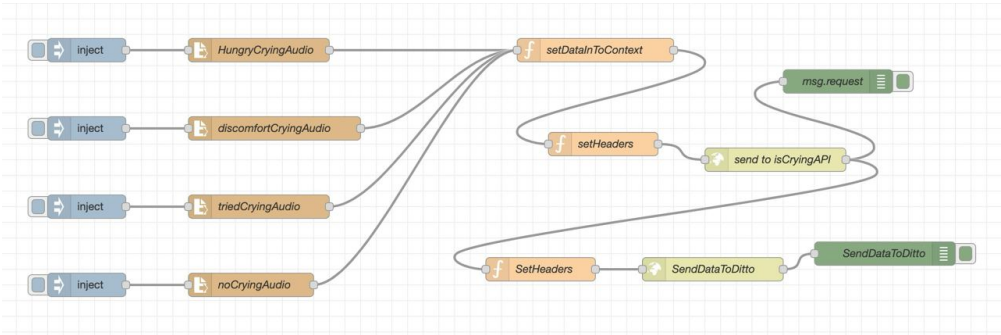


FIG. 4.14: Visualizing Sensor Data from a Baby Crib using Node-RED

- setHeaders (first) : Prepares audio data for cry detection and combines it with a thingId from a global variable. for create the form for sending to the "is_crying" API.
- SendDataToDitto : Sends data (audio and sensor data) to Ditto. Retrieves thingId from global and combines it with sensor data from context. Sends the prepared data to Ditto for storage or processing.
- setHeaders (second) : prepares sensor data and sends it to Ditto.

4.5 Mobile App

Our application is designed to help parents and caregivers quickly recognize and respond to an infant's needs through a user-friendly interface. Here's a brief explanation of each section of the app

4.5.1 Login Interface

The login interface allows users to securely access their accounts. Users enter their email and password to log in. New users have the option to sign up for an account.Look The figure 4.15

4.5.2 Add Baby Interface

The add child interface enables users to add a new child to the system. Users can input the child's name, date of birth, and other relevant details. This information helps personalize the notifications and status updates.Look The figure 4.16

4.5.3 Child's Status Interface

The child's status interface provides real-time updates on the child's current state, such as whether the child is crying, and the probable reason for the cry. This interface also displays historical data and trends.Look The figure 4.17 and 4.18

4.5.4 Notifications Interface

The notifications interface alerts users to important events, such as when the system detects that the child is crying or identifies a specific need. Notifications ensure that caregivers can promptly respond to their child's needs.Look The figure 4.19

4.5.5 Signature Correction by Caregivers Interface

This interface allows caregivers to provide feedback on the system's predictions. For example, if the system incorrectly identifies the reason for a child's cry, caregivers can correct it by selecting the actual reason. This feedback is used to improve the model's accuracy over time.Look The figure 4.20

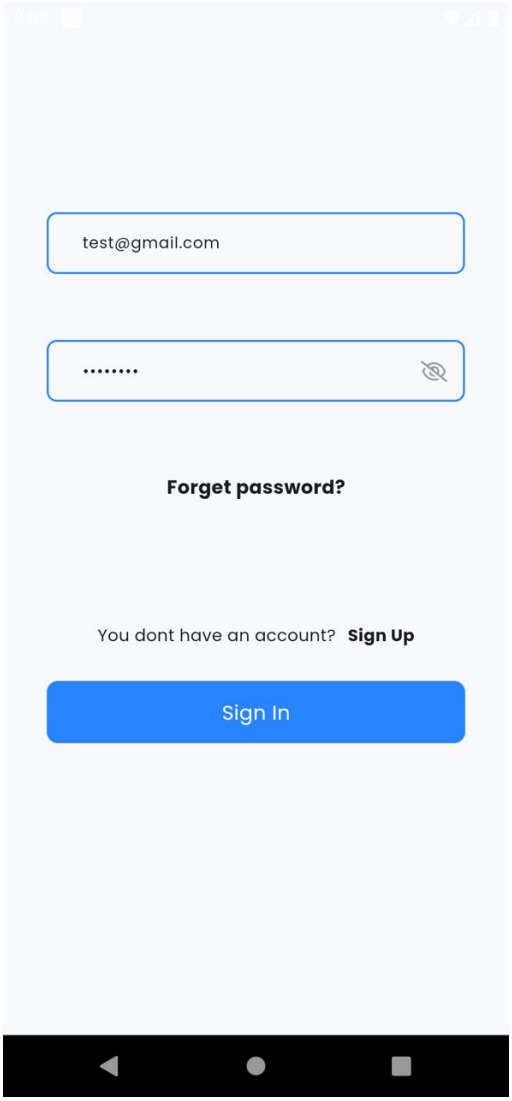


FIG. 4.15: Login Interface

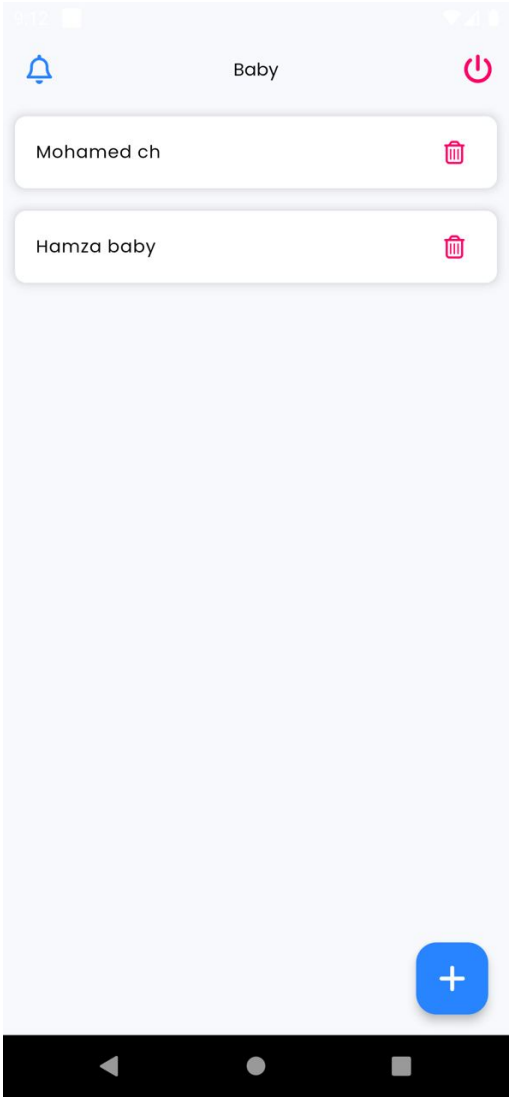


FIG. 4.16: Add Baby Interface



FIG. 4.17: Child's Status Interface

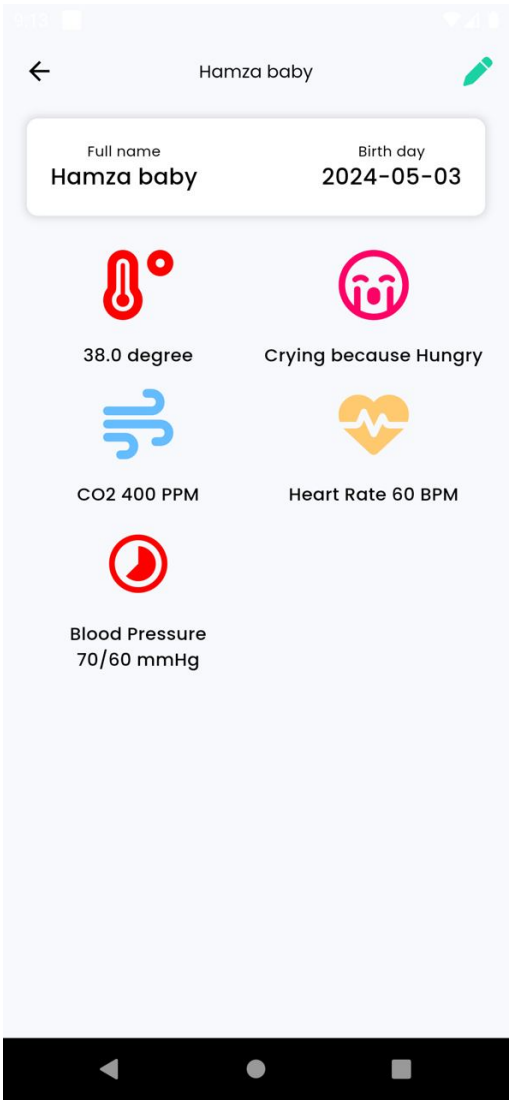


FIG. 4.18: Child's Status Interface After Crying

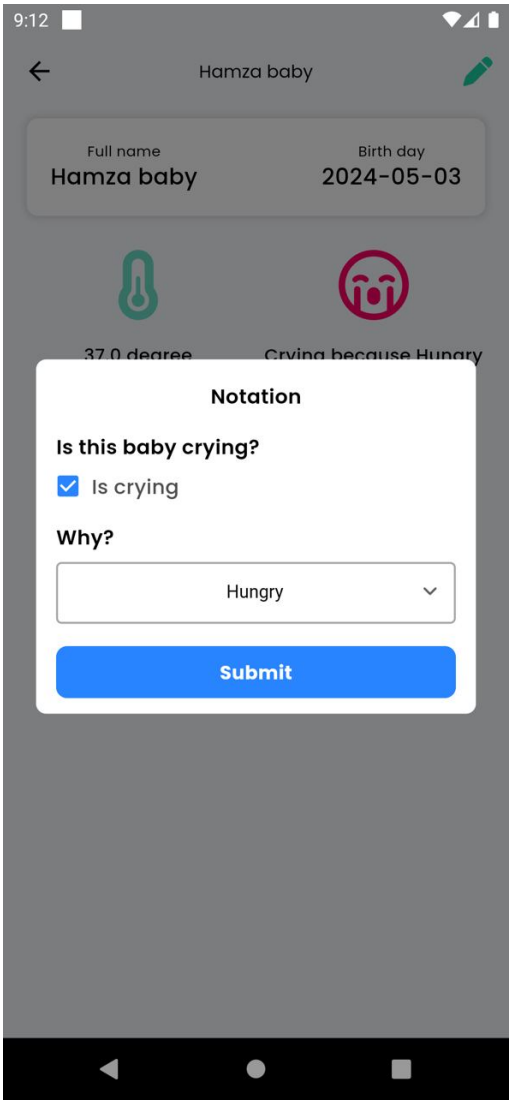


FIG. 4.19: Signature Correction by Caregivers Interface

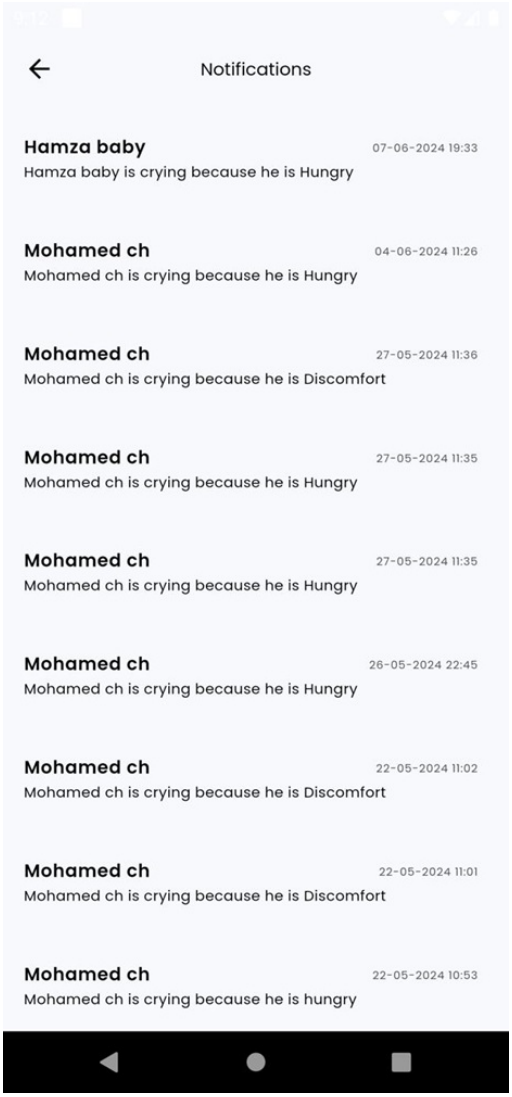


FIG. 4.20: Notifications Interface

4.6 Conclusion

The implementation of our project successfully integrated advanced digital twin technology, IoT tools, and deep learning models to create a robust and effective system for monitoring and analyzing baby cries. The use of Eclipse Ditto enabled seamless digital twin management, while Node-RED facilitated easy connectivity and data flow between sensors and the digital twin. Our deep learning models, optimized through rigorous tuning and validation, demonstrated high accuracy in detecting and classifying baby cries. This comprehensive approach ensures reliable performance, providing caregivers with valuable insights and enhancing the overall monitoring of infant well-being. Additionally, the application features a user-friendly interface designed to help parents and caregivers quickly recognize and respond to an infant's needs

General Conclusion

Our project on developing a digital twin for infant care involved several key steps and components to ensure a robust and efficient system• We started by selecting used tools, including IoT sensors, machine learning algorithms, and cloud platforms• These technologies were integrated to create a cohesive system capable of real-time data collection, processing, and analysis•

The system architecture comprises an AI-powered baby bed system that monitor and analyzing baby data on movement, vital signs, and sleep through a collection of sensors, its AI algorithms, and a cloud base platform. The system is composed of, a physical bed with sensors inside, a digital twin ; carrying the data coming from the bed, AI services to detect and classify different baby cries (e.g., hungry, hot, cold, uncomfortable...), a database where the acquired data is stored, and an End User application to let parents follow up using their kid well-being indicators on their mobile device Functionally, the digital twin allowed for real-time monitoring, cry detection and classification, and predictive alerts for potential health issues• We used machine learning models, specifically MLP and CNN-LSTM, to enhance the system's ability to accurately identify and respond to different cry patterns•

The development of the application was facilitated by the Eclipse Ditto platform for managing digital twin instances and Node-RED for ensuring effective connectivity and data exchange between the physical and digital components• This integration allowed for seamless data flow and real-time updates, improving the system's overall performance•

Additionally, the inclusion of AI services enabled predictive analysis based on historical and current data, providing valuable insights and timely alerts for caregivers• This aspect of the project was crucial in ensuring proactive responses to infant care•

In conclusion, the successful implementation of this digital twin-based system for infant care showcases the transformative potential of this technology in healthcare• By integrating IoT, machine learning, and cloud computing, we have created a system that not only improves the quality of care but also empowers caregivers with actionable insights and predictive capabilities•

Bibliographie

[1] https://www.ibm.com/fr-fr/topics/what-is-a-digital-twin

[2] https://www.talend.com/fr/resources/jumeau-numerique-definition/

[3] Sylvain, C, Définition Jumeau Numérique, Alliance Industrie du Futur – Janvier 2019

[4] Blidi, Ines, and Nour Elhouda Aoufi. "Automatic Approach Implementation for a Digital Twins Development for a Drinking Water Distribution System." Master's Thesis, Yahia Fares Medea University, Algeria, Faculty of Sciences, Department of Mathematics and Computer Science, 2020-2021 pp 5.

[5] Argolini, Roberto, Federico Bonalumi, Johannes Deichmann, and Stefania Pellegrinelli. "Digital twins : The key to smart product development." Article, July 31, 2023 Digital twins in manufacturing & product development | McKinsey

[6] The Digital Twin with Engineering - Engineering Industries eXcellence (indx.com)

[7] Singh, Deepak. "Why will intelligent digital twins become an industrial manufacturing must-have ?" Article, August 22, 2023 Digital twins become industrial manufacturing must-have (plantengineering.com)

[8] https://themanifest.com/big-data/blog/what-is-digital-twins [74]DIGITAL TWINS : THE 4 TYPES AND THEIR CHARACTERISTICS. Digital Twins : The 4 types & the characteristics | Tributech

[9] Coursera Staff. "What Is a Digital Twin ? Definition, Types, and Uses." Coursera, updated November 29, 2023. What Is a Digital Twin ? Definition, Types, and Uses | Coursera

[10] Plank, Thomas. "Digital Twins : The 4 Types and Their Characteristics." Tributech Blog, December 15, 2019. Digital Twins : The 4 types & the characteristics | Tributech

[11] Botín-Sanabria, Diego M., et al. "Digital Twin Technology Challenges and Applications : A Comprehensive Review." Remote Sensing 14, no. 6 (2022) : 1335.pp 04 https://doi.org/10.3390/rs14061335.

[12] Lv Z., Xie S. Artificial intelligence in the digital twins : State of the art, challenges, and future research topics (2021) Online : https://doi.org/10.12688/digitaltwin.17524.1

[13] Attaran M. The Internet of Things : Limitless opportunities for business and society J. Strateg. Innov. Sustain., 12 (1) (2017)

[14] Attaran, M., & Celik, B. G. (2023). Digital Twin : Benefits, use cases, challenges, and opportunities. Decision Analytics Journal, 6, 100165. pp 02 https://doi.org/10.1016/j.dajour.2023.100165

[15] Shu Z., Wan J., Zhang D. Cloud-integrated cyber–physical systems for complex industrial applications Mob. Netw. Appl., 21 (5) (2016), pp. 865-878

[16] Malakuti Somayeh ; Grüner Sten. 24 September 2018. Architectural aspects of digital twins in IIoT systems. European Conference on Software Architecture

[17] What is Digital Twin Technology and How It Benefits Manufacturing in the Industry 4.0 Era ? - SL Controls

[18] What is Digital Twin Technology and How Does it Work ? - TWI (twi-global.com)

[19] What is the Internet of Things (IoT)? | IBM

[20] What Is the Internet of Things (IoT)? (oracle.com)

[21] "IoT Architecture – Detailed Explanation." InterviewBit, January 2, 2024. Available at : https://www.interviewbit.com/blog/iot-architecture/

[22] [23] altexsoft. IoT architecture layers components. url : https://www.altexsoft.com/blog/iot-architecture-layers-components/.

[23] Itsekson, Alexey. "What Are the 7 Layers of IoT Architecture ?" Trends Business, Available at : https ://jelvix.com/blog/iot-architecture-layers.

[24] Lombardi, Marco, Francesco Pascale, and Domenico Santaniello. "Internet of Things : A General Overview between Architectures, Protocols and Applications." Information 12, no. 2 (2021) : 87. https ://doi.org/10.3390/info12020087.

[25] Blidi, Ines, and Nour Elhouda Aoufi. "Automatic approach implementation for a digital twins development for a drinking water distribution system." Master's Thesis, Yahia Fares Medea University, Faculty of Sciences, Department of Mathematics and Computer Science, 2020-2021. Pp 4.

[26] Yadav, Ujjwal. "Characteristics of Internet of Things (IoT)." Last updated : 3 Oct 2023. https://www.scaler.com/topics/characteristics-of-iot/

[27] Characteristics of Internet of Things - GeeksforGeeks

[28] Na, A., Isaac, W., Varshney, S., & Khan, E. (2017). An IoT based system for remote monitoring of soil characteristics. 2016 International Conference on Information Technology, InCITe 2016 - The Next Generation IT Summit on the Theme - Internet of Things : Connect Your Worlds, 316–320. https ://doi.org/10.1109/INCITE.2016.7857638

[29] Vadapalli, P. (2022, 22 avril). "What is IoT (Internet of Things) Technology ? Importance, Benefits & Pros and Cons" [Article de blog]. UpGrad. Récupéré à partir de https ://www.upgrad.com/blog/what-is-iot-internet-of-things-technology/

[30] Eclipse Ditto™ • open source framework for digital twins in the IoT

[31] Architecture overview • Eclipse Ditto™ • a digital twin framework

[32] Gour, R. (2018, 5 décembre). "4 Major IoT Protocols — MQTT, CoAP, AMQP, DDS" [Article]. Medium. URL https://medium.com/@rinu.gour123/4-major-iot-protocols-mqtt-coap-amqp-dds-46016897c3e9

[33] Lukman, N. (2021, 14 septembre). "CoAP, MQTT, AMQP, XMPP & DDS : Which Protocol Should You Choose for IoT ?" [Article]. NexPCB. URL https://www.nexpcb.com/blog/different-data-protocols-which-one-to-choose

[34] https://dashdevs.com/blog/product-owner-talks-20-digital-twins-service-companies/

[35] What Is NodeRED | OpenIAP Documentation

[36] Practical Node-RED Programming (packtpub.com)

[37] https://www.talend.com/fr/resources/what-is-machine-learning/

[38] Alexander S. Gillis, Technical Writer and Editor, David Peterssonm Source : TechTarget - Definition : Supervised Learning

[39] https://www.talend.com/fr/resources/what-is-machine-learning/

[40] https://www.geeksforgeeks.org/ml-semi-supervised-learning/

[41] https://www.ibm.com/topics/deep-learning

[42] Khelalef, Aziz. "Reconnaissance d'activités humaines en utilisant les descripteurs spatio-temporels 2D/3D." Thèse de doctorat, Université Batna 2 – Mostefa Ben Boulaïd, Faculté de Technologie, Département de l'électronique, Laboratoire d'Automatique Avancée et d'Analyse des Systèmes (LAAAS), 2020, p. 41. http://eprints.univ-batna2.dz/1909/1/these_PhD_khelalef_Aziz_VF_Corrected.PDF

[43] Aravind Pai.(2019,JUNE 10).Comprehensive Guide to Text Summarization using Deep Learning in Python.Retrieved from https://laptrinhx.com/comprehensiveguide-to-text-summarization-using-deep-learning-in-python-3082791976/ (Poster)

[44] https://www.ibm.com/topics/neural-networks.

[45] Simon Haykin. Neural Networks : A Comprehensive Foundation (3rd Edition). Prentice-Hall, Inc., USA, 2007.

[46] Editorial.(2020,March 15).Different types of Deep Learning models explained.Retrieved from https://roboticsbiz.com/different-types-of-deep-learningmodels-explained/ (Article)

[47] Alzubaidi, Laith, et al. "Review of Deep Learning : Concepts, CNN Architectures, Challenges, Applications, Future Directions." Journal of Big Data, vol. 8, 2021, article 53. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8

[48] Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2014). How to Construct Deep Recurrent Neural Networks. arXiv :1312.6026v5 [cs.NE]. https://arxiv.org/pdf/1312.6026

[49] Editorial. (2020, March 15).Different types of Deep Learning models explained.Retrieved from https://roboticsbiz.com/different-types-of-deep-learning-models-explained/ (Article)

[50] https://www.ibm.com/fr-fr/topics/autoencoder

[51] "Recursive Neural Network in Deep Learning." GeeksforGeeks, last modified January 6, 2024, https://www.geeksforgeeks.org/recursive-neural-network-in-deep-learning/ .

[52] Banoula, Mayank. "Introduction to Long Short-Term Memory (LSTM)." Simplilearn, last modified April 27, 2023, https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/lstm.

[53] Singh, Sumit. "5 Best Deep Learning models — When Should You Use Them ?" Labellerr, Nov 16, 2022, https ://www.labellerr.com/blog/5-deep-learning-models-when-should-you-use-them/

[54] https ://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[55] Banoula, Mayank. "Introduction to Long Short-Term Memory (LSTM)." Simplilearn, Apr 27,2023, [Lesson 13 of 23] https ://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/lstm#structure_of_lstm.

[56] https://spotintelligence.com/2024/02/20/multilayer-perceptronmlp/#Applications_of_Multilayer_Perceptron_MLP

[57] R, Rekha, and Tharani R S. "Speech Emotion Recognition using Multilayer Perceptron Classifier on Ravdess Dataset." ICCAP 2021, December 07-08, Chennai, India. EAI, 2021. DOI : 10.4108/eai.7-12-2021.2314726.

[58] Editorial.(2020,March 15).Different types of Deep Learning models explained.Retrieved from https ://roboticsbiz.com/different-types-of-deep-learning-models-explained/ (Article)

[59] Simplilearn. "List Of Generative Adversarial Networks Applications." Simplilearn, 13 Feb. 2023, https://www.simplilearn.com/generative-adversarial-networks-applications-article#top_generative_adversarial_networks_applications.

[60] https://iq.opengenus.org/applications-of-rnn/

[61] https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/

[62] "Introduction to Deep Neural Networks." DataCamp, Jul. 2023, www.datacamp.com/tutorial/introduction-to-deep-neural-networks?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720824&utm_adgroupid=157156376071&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative=684592140425&utm_targetid=dsa-2218886984380&utm_loc_interest_ms=&utm_loc_physical_ms=9069693&utm_content=&utm_campaign=230119_1-sea~dsa~tofu_2-b2c_3-row-p2_4-prc_5-na_6-na_7-le_8-pdsh-go_9-na_10-na_11-na&gad_source=1&gclid=CjwKCAjwrIixBhBbEiwACEqDJQOb2ukb2b9mxEF99ya59FPTgqKBppnlkf0-yAjrcqWdloOipYWMyhoCWnYBwE.

[63]

[64] https://www.quora.com/What-are-some-challenges-in-deep-learning

[65] https://www.javatpoint.com/machine-learning-vs-deep-learning

[66] Snoussi, Mouhcine. "Chapter 2.1: Self-Driving Car [Intro to Neural Network - The Perceptron]." LinkedIn, 19 Apr. 2020, https://www.linkedin.com/pulse/chapter-21-self-driving-car-intro-neural-network-mouhcine-snoussi.

[67] SKY ENGINE AI. "What is a Convolutional Neural Network?" Developer Blog, 09 June 2023, https://skyengine.ai/se/skyengine-blog/125-what-is-a-convolutional-neural-network.

[68] PangDae. "Memoir [BoostCamp Assignment] Backpropagation." 10 August 2021. BoostCamp_Assignment, https ://velog.io/@kimhwangdae/BoostCamp-Assignment-Backpropagation.

[69] Akbari, Kevin. "Demystifying Autoencoders : Understanding the Mathematics and Implementation." 10 April 2024, https://akbarikevin.medium.com/demystifying-autoencoders-understanding-the-mathematics-and-implementation-a9c1b3e

[70] Thing • Eclipse Ditto™ • a digital twin framework

[71] Feature • Eclipse Ditto™ • a digital twin framework

[72] Basic concepts overview • Eclipse Ditto™ • a digital twin framework

[73]

[74] LinkedIn. (23 février 2024). Comment la vision par ordinateur peut-elle améliorer les soins aux bébés? Généré par l'IA et la communauté LinkedIn. Consulté sur https://www.linkedin.com/advice/3/how-can-computer-vision-improve-baby-care-r3gzf

[75] Y. LeCun, "There can be no intelligence without learning.," 1 January 2020. [Online]. Available :https://twitter.com/ylecun/status/1212381611838361602.

[76] AI APIs : How AI is Revolutionizing Application Programming Interfaces, URL : https://www.mosaicx.com/blog/ai-apis-how-ai-is-revolutionizing-application-programming-interfaces

[77] What is AI planning and forecasting? URL : https ://www.domo.com/glossary/what-is-ai-planning-and-forecasting

[78] C. Reeves. A Genetic Algorithm for Flowshop Sequencing. Computers & Operations Research, 22(1) :5–13, 1995

[79] C. Houck, J. Joines, and M. Kay. A Genetic Algorithm for Function Optimization : A Matlab Implementation. Technical Report, North Carolina State University, 2009.

[80] L. Xie, A. Yuille. Genetic CNN. In : ICCV, 2017, pp. 1379–1388

[81] Mehdary, Adil, et al. "Hyperparameter Optimization with Genetic Algorithms and XGBoost : A Step Forward in Smart Grid Fraud Detection." Sensors, vol. 24, no. 4, 2024, p. 1230. MDPI, https://doi.org/10.3390/s24041230