```
\documentclass{book}
\sloppy
\usepackage[bookmarksnumbered=true]{hyperref}

\usepackage[utf8]{inputenc}
\usepackage{hyperref}
\usepackage[backend=biber, style=numeric]{biblatex}  % Ορίζουμε το στυλ ως
numeric
\addbibresource{bibl.bib}

\usepackage[a4paper, left=1in, right=1in, top=1in, bottom=1in]{geometry}
\usepackage{listings}
\usepackage{fancyhdr}
\usepackage[backend=biber]{biblatex}
\addbibresource{bibl.bib}
\usepackage{fancyhdr}
\usepackage{graphicx}
\usepackage{setspace}
\usepackage{multirow}
 \usepackage{amsmath}
\usepackage[x11names,table]{xcolor}
\usepackage{listings}
\setlength{\headheight}{22.54448pt}
\addtolength{\topmargin}{-7.34448pt}
\lhead[\textbf{\thesection.\quad \rightmark}]{}
\rhead[]{\textbf{\chaptername\ \thechapter.\quad \leftmark}}




\begin{document}

\mainmatter  % Ξεκινάει τα κύρια κεφάλαια

\chapter{Introduction: IoHT Insights Unveilingthe World of Fetal Heart Rate
Monitoring}
\label{chap:introduction}

\begin{flushright}
\textit{Tools serve, Knowledge leads, and Intelligence transforms.} \\
\small — A.G. Skrivanos
\end{flushright}

% Abstracts (εδώ μπορείς να προσθέσεις περιεχόμενο για τα abstract)
\begin{figure}[htbp]
    \centering
    \includegraphics[width=0.5\textwidth]{dissertation_flowchart.pdf}  %
Εισαγωγή PDF
    \caption{ROADMAP.}
\end{figure}
```

This thesis is structured into thematic sections covering the process from the initial conception of the idea to the implementation and evaluation of the proposed solution. The diagram below summarizes the flow and logical sequence of the chapters.

\section*{When Life Begins: Intelligent Monitoring and Prediction of Cardiac Events Using Embedded AI}

My interest in heart rate monitoring did not arise solely from academic curiosity. Rather, it has deep personal roots, dating back to my wife's pregnancy experience. The sense of responsibility and natural concern of every parent for the health of their unborn child led to the formulation of the following fundamental question:

\begin{quote}
\textbf{\textit{"How can I ensure that the fetus develops safely, even outside the hospital setting?"}}
\end{quote}

This question laid the foundation for the development of a portable and automated monitoring system, designed to operate in real time, offering user-friendliness along with the ability to send alerts or trigger interventions when necessary.

The initial challenge was identifying the right technology. An extensive literature review was conducted in areas such as electrocardiography (ECG), photoplethysmography (PPG), biomedical signal acquisition systems, and artificial intelligence-based diagnostic algorithms.

The final system was developed using a heart rate sensor integrated into an Arduino-based platform. It included a PPG sensor capable of accurately recording pulse signals, processing the data locally, and transmitting to the cloud via the ThingSpeak platform.
\sloppy
Data was uploaded every 30 seconds, allowing authorized healthcare providers or family members to remotely monitor the fetal condition. The system was optimized for low power consumption and continuous operation, validating its practicality for daily use.

This application was successfully tested in real conditions during my wife's pregnancy, in collaboration with the General Hospital of Kalamata and the gynecologist Dr. Georgios Petrakos.

However, as the pregnancy progressed, new concerns arose:

\begin{quote}
\textbf{\textit{"What if complications arise in the child's cardiac function later in life?"}}
\end{quote}

This concern led to a new line of research focused on predictive cardiology.

Specifically, the present study introduces an electrocardiogram (ECG) baseline detection system designed to predict the spontaneous termination of atrial fibrillation (AF). The proposed deep neural network (DNN) classifier system combined with Hjorth parameters extracted from ECG signals. The detection process consists of two stages: first, the extraction of Hjorth features and second, the classification of the signals using the DNN model. The clinical data used to validate the proposed scheme were obtained from the publicly available database of Shandong Provincial Hospital (SPHD). The confusion matrix and receiver operating characteristic (ROC) analyses confirmed that the proposed model achieves high accuracy while maintaining low computational complexity, making it suitable for application in portable consumer electronic devices. Offering materials is called help, offering knowledge and offering material is called function

\section{Wireless Sensor Networks and AI in IoHT: Towards Intelligent, Low-Power and Life-Saving Systems}

In recent years, the convergence of Wireless Sensor Networks (WSNs), Artificial Intelligence (AI), and the Internet of Things (IoT) has transformed the healthcare domain, resulting in the Internet of Health Care Things (IoHCT). This emerging paradigm enables real-time monitoring, early disease prediction, and adaptive responses to health-related events delivered through compact, low-power, and cost-effective systems %\cite{CardioGuard}.

Among these, electrocardiography (ECG) systems have experienced a remarkable shift. ECG monitoring has become a proactive tool in personalized medicine, ranging from bulky clinical machines to portable, wearable, and AI-enhanced sensors. AI algorithms can interpret complex ECG signals, detect arrhythmia's and predict spontaneous termination of atrial fibrillation (AF) as a life-threatening condition %\cite{CardioGuard}.
The proposed CardioGuard system, introduces a novel detection framework using Hjorth parameters and deep neural networks (DNNs) to achieve a high diagnostic accuracy with minimal resource consumption %\cite{CardioGuard}.

Simultaneously, IoT-based architectures redefine how the health information is collected, processed, and accessed. Plug-and-play wearable devices seamlessly integrate with body-area networks and redefine remote care by offering effortless operation, anyone-anywhere access, and intelligent adaptability \cite{28}. In emergency scenarios, whether inside an ambulance or in a home-care setting, such systems become critical life-saving enable is high lighting the true potential of IoT-powered healthcare.\cite{skrivanosesp8266}

This dissertation presents hardware-optimized WSN implementations for multiple IoHcT applications, focusing on energy efficiency, signal processing, and edge intelligence. These implementations have been successfully validated through projects such as the:

\textbf{AI-powered ECG classification systems} %\cite{CardioGuard},

\textbf{Smart fetal heart rate monitoring platforms} \cite{28,skrivanosesp8266}

and \textbf{ intelligent habitat monitoring for endangered species }%\cite{TurtleIoT}.

Although each use case addresses different application domains, the underlying architecture remains common: low-cost, scalable sensor networks integrated with machine learning intelligence, optimized for real-time decision-making and power-aware operation.

By advancing these technologies, this study aims to bridge the gap between academic innovation and practical deployment of intelligent IoHcT systems, laying the groundwork for a safer, smarter, and more responsive healthcare ecosystem.

{\textbf{keywords}}
\textbf{ECG}: Electrocardiogram, heart rate, QRS complex, P wave, T wave, arrhythmia, cardiovascular disease.

\textbf{Neural Networks}: Deep learning, artificial intelligence, backpropagation, convolutional neural networks (CNN), recurrent neural networks (RNN), supervised learning, unsupervised learning.

\textbf{DWT}: Discrete wavelet transform, wavelet decomposition, multi-resolution analysis, signal processing, image compression.

\textbf{IoT}: Internet of Things, smart devices, connected devices, sensor networks, data management, cloud computing, real-time data analysis, smart homes, smart cities.

\textbf{BWSN}: Body wireless sensor networks, wireless health monitoring,wireless medical devices, wearable devices, telemedicine, wireless communication protocols.

\textbf{Cardiologist}: Heart disease, cardiovascular disease, arrhythmia, heart attack, cardiac catheterization, angiography, echo-cardiography, electrophysiology.

\textbf{Fetus}: Prenatal development, fetal heart rate, fetal ECG, fetal monitoring, ultrasound, fetal health, fetal movement, fetal position.

\textbf{IOHT}: Internet of Healthcare Things, remote patient monitoring, telemedicine, healthcare IoT, medical devices, mHealth, eHealth, health informatics, connected health.
\section{Internet of Things}

The \textit{Internet of Things (IoT)} refers to an extensive network of physical and digital devices equipped with sensors, software, and connectivity capabilities, allowing the collection and exchange of data in real time. The idea of "smart objects" has its roots as early as the 1980s with early applications in industrial networks, while the term "Internet of Things" was coined by Kevin Ashton in 1999 at the MIT Auto-ID Center \cite{ashton2009iot}.

The technological maturity of sensors, wireless communications (such as WiFi, Bluetooth, ZigBee), low power consumption, and IPv6 protocols have contributed significantly to the practical implementation of the IoT since the mid-2010s \cite{zanella2014internet}. Today, IoT is evolving into one of the key

foundations of the Digital Age, with applications in areas such as smart health, transportation, cities, energy, and Industry 4.0 \cite{atzori2017evolution}.

The core of IoT is based on a combination of technologies: embedded systems, cloud computing, machine learning, and security protocols for sharing data between heterogeneous devices. Current trends include the integration of artificial intelligence in IoT (\textit{AIoT}), decentralized processing with edge/fog computing, and specialized application in environments such as \textit{Internet of Medical Things (IoMT)} and \textit{Internet of Health-centric Things (IoHcT)} \cite{gubbi2013internet, islam2021aiiot}.

As connected infrastructure expands, the importance of interoperability, security, and scalability of IoT systems becomes crucial for their future viability.

\subsection{Internet of Health Care Things}

The application of the Internet of Medical Things (IoMT) and the broader Internet of Healthcare Things (IoHcT) has paved the way for the development of wearable low-power systems for real-time health monitoring \cite{R1}. Particularly in fetal health monitoring, the integration of wearable sensors, ECG/PPG technologies, and cloud-based data transmission systems has enabled continuous, non-invasive supervision of fetal development, even in remote or home-based environments. \cite{Mohanty2019,Yang2024}.

Despite technological advancements, current commercial solutions often fall short of accessibility, energy efficiency, and real-time data availability requirements. There is growing demand for systems that are affordable, reliable, user-friendly, and capable of 24/7 monitoring, particularly in cases where direct medical care is not readily available\cite{Raj2018}.

\section{Artificial Intelligence}

Artificial Intelligence (AI) represents a transformative scientific field that mimics and enhances human cognitive functions through computational intelligence, allowing systems to perceive, reason, learn, and act autonomously according to \cite{marr2022}. The idea was based on the basic principle of training humans from the earliest stages of their lives. These artificial systems are rule-based, AI has evolved into data-driven architectures supported by neural networks and deep learning algorithms that achieve cutting-edge performance in areas such as medical imaging, robotics, and real-time analytics \cite{rajkomar2019machine, lecun2015deep}.

In recent years, AI with embedded systems and increased computational evolution has swept the emergence of the Internet of Things (IoT), enabling intelligent, distributed, and adaptive environments to be integrated into consumer electronics \cite{xu2014} .
One of the most effective applications of this integration lies in the Internet of Things in Healthcare (IoHcT), where AI-enabled medical sensors, wearables, and cloud-based analytics empower continuous, real-time, and patient-centric monitoring and diagnosis \cite{bhardwaj2022} . This paradigm shift is transforming traditional healthcare into a preventive, predictive, and personalized system.

The success of AI-based IoHcT systems relies heavily on advances in wireless

communications, particularly the evolution from 3G/4G to 5G and beyond. Ultra-reliable low-latency communications (URLLC), massive machine-to-machine communication (mMTC), and high data throughput, as offered by 5G and the anticipated 6G technologies, enable seamless interaction between edge devices and central signal processing infrastructures \cite{haque2023, dang2020should}. These advances ensure timely delivery of critical health data, real-time decision support, and telemedicine services powered by AI models operating at the edge of the planet or in the cloud \cite{mansour2021}. The synergy between Artificial Intelligence, IoHcT and telecommunications innovations is laying the foundation for a next-generation smart healthcare ecosystem.

\subsection{Machine learning}

Machine Learning (ML) is a subfield of artificial intelligence that allows systems to learn patterns from data and improve their performance on specific tasks without being explicitly programmed. ML algorithms are designed to build models that generalize from a set of examples, making decisions or predictions on new, unseen data \cite{jordan2015machine, domingos2012few}.

The three main paradigms of machine learning are supervised learning, where algorithms are trained using labeled input output pairs, unsupervised learning, which focuses on discovering hidden patterns or structures in unlabeled data, and reinforcement learning, where agents learn to act in an environment by maximizing cumulative reward through trial-and-error interactions \cite{li2017deep, lecue2019}. With the exponential increase in the availability of data and computational resources, ML has become fundamental to modern applications, including natural language processing, computer vision, bioinformatics, and autonomous systems \cite{lecun2015deep, goodfellow2016deep}. Recent developments have also introduced hybrid learning schemes, such as self-supervised learning and few-attempt learning, which address the limitations of traditional machine learning (ML) in data-poor scenarios \cite{chen2020simple, brown2020language}. Overall, machine learning continues to emerge as a critical technology for data-driven innovation across scientific, industrial, and healthcare sectors, as will be applied in this study.

\subsection{Historical Evolution of Neural Networks}

The evolution of neural networks spans over eight decades, marked by key theoretical milestones and technological breakthroughs. The field began with the foundational model of McCulloch and Pitts \cite{mcculloch1943logical}, who introduced the neuron as a binary computational unit and described mechanisms such as excitatory and inhibitory signals, along with feedback loops, forming the logical structure of early neural computation \cite{pitts1947mathematical}. Their work laid the groundwork for connecting neurobiology with formal systems, inspiring von Neumann's subsequent exploration of computation and the architecture of the brain \cite{vonneumann1958computer}. Hebb’s theory of

synaptic plasticity \cite{hebb1949organization}, known as Hebbian learning, further emphasized the importance of adaptive connection weights, establishing the conceptual basis for learning in neural models. In the 1950s and 60s, Rosenblatt's perceptron \cite{rosenblatt1962principles} and the Adaline/Madaline architectures by Widrow and Hoff \cite{widrow1960adaptive} emerged as the first physical and functional neural models with practical applications, although their limited capabilities—most notably the inability to solve non-linearly separable problems—were exposed by Minsky and Papert \cite{minsky1969perceptrons}. These limitations led to a temporary decline in research until the 1980s, when Hopfield \cite{hopfield1982neural} revitalized interest with recurrent networks, and the backpropagation algorithm formalized by McClelland and Rumelhart \cite{mcclelland1986parallel} opened the way for deep multilayer networks. This period marked the formal institutionalization of the field, with the emergence of dedicated conferences, journals, and organizations like the International Neural Network Society, led by figures such as Grossberg, Kohonen, and Amari \cite{grossberg1988competitive, amari1997information}. Despite initial commercial enthusiasm, tempered growth followed, shifting expectations toward sustained research and application development \cite{lecun2015deep, schmidhuber2015deep}. Today, neural networks represent a robust interdisciplinary domain, driven by both the scientific aspiration to model cognitive processes \cite{hebb1949organization, hopfield1982neural} and the engineering goal of surpassing classical computational architectures \cite{mcculloch1943logical}.

\subsubsection{Neural Networks}
Neural networks represent a modern and rapidly evolving research field within the natural sciences. Over the past four decades, research and development in this area have intensified significantly \cite{anderson1995introduction, haykin1999neural, bose1996neural, lippmann1987introduction, obermeier1987time, hecht1988neurocomputing, kohonen1987adaptive}. The first neural network model proposing that neurons are the basic unit of the network was introduced in 1943 by McCulloch and Pitts \cite{mcculloch1943logical}. In their seminal work, they presented for the first time the idea that a neural network consists of a collection of a large number of interconnected neurons, and they demonstrated how these neurons could operate through their connections. This is historically considered the first formal representation of a neural network. McCulloch and Pitts envisioned neurons and their connections as a model analogous to an electrical circuit. Notably, McCulloch was a neurophysiologist, while Pitts was an 18-year-old first-year mathematics student. In 1947, they advanced this model further, focusing on pattern recognition \cite{mcculloch1947letter}. This progress is reflected not only in the increasing number of researchers engaged in the study of neural networks but also in the remarkable achievements attained through their application, which have brought this technology to the attention of a broad scientific and technological audience.

Their importance is particularly pronounced in the technological sciences, as the fundamental principles and operational mechanisms of neural networks are inspired by the nervous systems of living organisms, including that of humans. Moreover, their use has extended far beyond the realm of biology, finding application in a wide array of computational problems and challenges.

In contrast to classical computing systems, neural networks aim to integrate elements of human cognition with advanced mathematical structures. Concepts such

as learning, memory, and forgetting, which until recently were attributed exclusively to human cognitive processes, have now become central components in the operation of neural networks. At the same time, complex mathematical functions, analytical methods, and computational models are employed to achieve their objectives.

Scientists active in this field come from a wide range of disciplines, including medicine, engineering, physics, chemistry, mathematics, computer science, and electrical engineering. This interdisciplinary nature makes the field of neural networks one of the most intriguing and demanding areas of modern research, with applications extending from medicine and biology to technology.

In contrast to classical computing systems, neural networks aim to integrate elements of human cognition with advanced mathematical structures \cite{haykin1999neural}. Concepts such as learning, memory, and forgetting, which until recently were attributed exclusively to human cognitive processes, have now become central components in the operation of neural networks \cite{goodfellow2016deep, lecun2015deep}. At the same time, complex mathematical functions, analytical methods, and computational models are employed to achieve their objectives \cite{bishop1995neural}.

The fundamental idea behind the development of neural networks originates from the study of the nervous system in animals and humans. Living organisms possess nervous systems responsible for processes such as interaction with the environment, learning, and information storage. The brain, as the central structure, consists of a network of neurons that function as the core computational units. Neurons continuously process information through the exchange of electrical signals, providing the biological model upon which artificial networks are based. Understanding these processes is the focus of intensive research, aiming to achieve deeper insight into complex brain functions such as cognition and memory. Biological neural networks are extensively studied in the fields of biology and medicine, as they are fundamental to all living organisms (with the exception of plants) \cite{gerstner2002spiking, kandel2013principles, marblestone2016toward, lillicrap2020backpropagation}.

Numerous efforts have been made to integrate the concept of neural systems into computer science. The ability of electronic computers to replicate functions of the human brain, such as image and speech recognition, has been a longstanding subject of research. Despite their substantial computational power, conventional computers struggle to achieve the flexibility and adaptability of the human mind, primarily because their architecture differs significantly from that of biological systems. This gap has served as the foundation for the development of artificial neural networks (ANNs), which aim to emulate the functioning of the human brain \cite{marblestone2016toward, lillicrap2020backpropagation}.

Artificial neural networks (ANNs) differ fundamentally from traditional computational systems, as they acquire knowledge through training and experience rather than relying on strictly predefined algorithms. Their training process involves the presentation of data sets (patterns) for which the desired output is known. As the network internally adjusts its parameters to achieve correct input-output mappings, it develops the ability to generalize to new, unseen data, provided that these data share similar characteristics with the training examples. Although this process may appear ambitious, progress in the field over

the past thirty years has been remarkable, establishing artificial neural networks as a key tool in a wide range of applications \cite{goodfellow2016deep, lecun2015deep}.

The primary purpose of an artificial neural network is to autonomously perform specific tasks, such as image recognition, provided it has been appropriately trained beforehand. Each network receives certain inputs and produces corresponding outputs (input output). The training process involves presenting the network with a set of patterns, which are representative or similar to those the network is expected to learn. This means providing the network with input data for which the desired output is already known, in other words, the target or correct response to each input is specified. Essentially, it is akin to supplying the network with both the question and the correct answer.

Using this information, the network adjusts its internal structure to match the input-output relationships provided during training. Once the network has identified an appropriate internal configuration, it can generalize to solve new, similar problems that it has not encountered before, even though it was not explicitly trained on those particular examples. However, it is important to note that the new problems must share similar characteristics and belong to the same domain as the training data. While this process may appear ambitious, it represents the most common approach to training neural networks, although, as will be discussed later, various alternative training methods also exist \cite{goodfellow2016deep, lecun2015deep}.
Recent advances in supervised, unsupervised, and reinforcement learning have significantly expanded the capabilities of neural networks \cite{raffel2020exploring, chen2020simple, schulman2022proximal, badia2020agent57}.

In summary, artificial neural networks represent an innovative computational framework that integrates principles from biology, mathematics, and computer science, offering solutions to complex problems that were once considered the exclusive domain of human intelligence.

\subsubsection{The Human Brain as a Neural Circuit}
The human nervous system can be conceptualized as a three-stage system, where receptors convert stimuli into electrical signals, the neural network (brain) processes the information, and effectors produce appropriate responses \cite{arbib1987brains, bassett2017network, sporns2011networks}. Understanding brain function began with the pioneering work of Ramón y Cajál, who identified neurons as the brain’s structural units \cite{ramon1911histology, fields2002new}. Although neurons operate much more slowly than electronic gates, their immense connectivity and number (~10 billion neurons and ~60 trillion synapses) make the brain an extraordinarily energy-efficient system \cite{shepherd2003synaptic, fornito2016fundamentals, bassett2017network}.

Synapses are the fundamental communication elements between neurons, possessing both chemical and electrical properties, while brain plasticity enables the creation or modification of connections \cite{eggermont2007role, churchland1992computational, fields2002new}. At both micro- and macro-levels, brain organization follows hierarchies: from synapses and microcircuits to local

and interregional neural interactions, which create mapped sensory processing areas in the cerebral cortex (e.g., Brodmann maps) \cite{brodal1981central, fornito2016fundamentals, yamins2016using}.

Despite advances in artificial neural networks, they remain primitive compared to the biological complexity of the brain \cite{anderson1995introduction, marblestone2016toward, lecun2015deep}. However, modern approaches in computational neuroscience, inspired by biology, continue to enrich the understanding and applications of artificial networks \cite{schmidhuber2015deep, bassett2017network, fornito2016fundamentals}.

Synapses, or nerve endings, are fundamental structural and functional units mediating interactions between neurons \cite{shepherd2003synaptic, fields2002new}. The most common type of synapse is the chemical synapse, where a presynaptic process releases a neurotransmitter that diffuses across the synaptic cleft and acts on the postsynaptic region \cite{churchland1992computational}. Thus, the synapse converts a presynaptic electrical signal into a chemical signal and then back into a postsynaptic electrical signal \cite{anderson1995introduction, sporns2011networks}.

Regarding information transmission, it is determined not by the signal type but by the neural pathway through which the signal passes across distinct communicating neurons \cite{yamins2016using, lecun2015deep}. There are two distinct signal states:
the resting potential (-65 mV) and the action potential (+30 mV).
To generate a signal, the neuron receives inputs that modulate its potential, once the threshold (-55 mV) is exceeded, it fires and produces an electrical signal \cite{marblestone2016toward, bassett2017network}. The signal is always transmitted in a predictable and stable direction \cite{fornito2016fundamentals, sporns2011networks}.

Ultimately, one might wonder: Could the human brain be considered a bio-digital electronic circuit? \cite{lecun2015deep, schmidhuber2015deep, yamins2016using}

\subsubsection{Neural Networks and Computers}

Figure 1 shows the simplest possible neural network, consisting of a single neuron. More complex neural networks are constructed from multiple neurons connected together. The structure of such networks can become extremely complex, referred to as the network architecture, which is a central topic in the study of artificial neural networks \cite{ beniaguev2021single}. The architecture of neural networks is different from that of traditional computers, which are based on a central processing unit (CPU). Conventional computers operate serially, following early von Neumann principles, and can execute a set of well-defined instructions, such as arithmetic operations, according to an internal clock.

In contrast, neural networks do not operate serially, but rather in a manner similar to parallel processing, where a task is distributed across different parts of the network and across its neurons. Thus, neural networks are often described as \textit{parallel distributed processing} \cite{ wang2025hybrid} systems. This parallelism provides significant computational speed, as if

multiple processors were operating simultaneously. However, the architecture of neural networks differs from that of parallel processors because the simple units of neural networks (i.e., neurons) have a large number of interconnections, usually far exceeding the number of neurons themselves. In contrast, in parallel computers, the number of processors usually exceeds the number of interconnections, and their overall complexity still follows the von Neumann model.

Neural network units are much simpler, performing only basic functions such as summing input signals and adjusting connection weights. In addition, neurons operate independently of each other without requiring synchronization. This feature gives neural networks robustness and fault tolerance \cite{li2025layer, sun2023approxabft, kim2025naper}.

In a neural network, information is distributed across many neurons, whereas in a conventional computer, data is stored in specific memory locations in binary form. When a neural network successfully solves a problem, we often do not understand precisely \textit{why} or \textit{how} the solution was achieved. Neural networks do not break a problem down into small logical parts, but solve it holistically, a method that is difficult for the human mind to understand through conventional logic. Nevertheless, the correctness of the solution can be verified, making this technique extremely efficient.

Another novel property of neural networks is fault tolerance. This means that if a small part of the network fails, the rest of the network continues to function, albeit with some minor errors \cite{beniaguev2021single, park2025neuromorphic}. Similarly, if part of the input data is incorrect, the network can still produce the correct output, albeit with minor inaccuracies. In contrast, computers behave very differently. For example, if an error in a computer program results in division by zero, the system immediately stops execution and returns an error message, even if the rest of the program is flawless. A neural network, on the other hand, recognizes such an operation as invalid, bypasses it with some degree of error in the final output, and proceeds to solve the problem.

\begin{table}[]
\begin{tabular}{|c|c|c|cc}
\cline{1-3}
\textbf{Topic}   & \textbf{Neural Networks}   & \textbf{Computer} & & \\
\cline{1-3}
\textbf{\begin{tabular}[c]{@{}c@{}}Process\\ synchronization\end{tabular}} &
\begin{tabular}[c]{@{}c@{}}With synchronous\\ operation\end{tabular}   &
\begin{tabular}[c]{@{}c@{}}With asynchronous\\ operation\end{tabular}   & &
\\ \cline{1-3}
\textbf{Processing} & Parallel processing   & Serial processing & & \\
\cline{1-3}
\textbf{Programming}   & \begin{tabular}[c]{@{}c@{}}Trained with examples \\
by adjusting the weights \\ of their connections\end{tabular} &
\begin{tabular}[c]{@{}c@{}}Programmed with \\ logical instructions\\
(if-then)\end{tabular}   & & \\ \cline{1-3}
\textbf{ Operation structure} & \begin{tabular}[c]{@{}c@{}}Memory, networks, \\
and processing units coexist\end{tabular}   &
\begin{tabular}[c]{@{}c@{}}Memory and information \\ processing are
separated\end{tabular}   & & \\ \cline{1-3}

```
\textbf{Fault tolerance} & Fault tolerance  & No fault tolerance     &  &  \\
\cline{1-3}
\textbf{Self-organization}   & \begin{tabular}[c]{@{}c@{}}Self-organization
during\\  the training process\end{tabular} &
\begin{tabular}[c]{@{}c@{}}Entirely dependent on\\  the provided
software\end{tabular}   &  &  \\ \cline{1-3}
\textbf{Information storage} & \begin{tabular}[c]{@{}c@{}}Information is stored
in the\\  weights of the connections\end{tabular}  &
\begin{tabular}[c]{@{}c@{}}Information is stored in\\  addressed memory
locations\end{tabular} &  &  \\ \cline{1-3}
\textbf{Cycle time} & \begin{tabular}[c]{@{}c@{}}Cycle time is on the order\\
of milliseconds (msec)\end{tabular}  & \begin{tabular}[c]{@{}c@{}}Cycle time is
on the order of \\ nanoseconds (nsec)\end{tabular}    &  &  \\ \cline{1-3}
\end{tabular}
\caption{Comparison between Neural Networks and Computers}
\end{table}
```

Fault tolerance is a unique feature of neural networks that is not commonly found in standard computational methods. This feature can sometimes provide practical solutions efficiently, especially when approximate rather than exact answers are acceptable. However, this is not universally applicable, and it is important to recognize that neural networks are not a solution to all currently unsolved problems. In fact, there are cases where their use is not recommended.

```
\subsubsection{Neural Networks Learnings}
```

A neural network is composed of a number of elements called neurons. Each neuron receives a set of input signals, has several possible internal states, and produces a single output, which is a function of the input signals (see Figure 1). Each input is associated with a weight value, indicating the strength of the connection between two neurons. Typically, these weights range between -1 and 1. The meaning of the weight is analogous to a chemical bond between two atoms in a molecule, indicating the strength of the connection between the two units \cite{goodfellow2016deep}.

When a neuron is activated, it computes a function over all the input data and compares the result to a threshold value characteristic of that neuron. If the computed function exceeds the threshold, the neuron produces an output, which is passed on as input to the next neuron(s). During the training process, the only component that changes is the set of connection weights between neurons. The way these weights are adjusted depends significantly on the learning method employed \cite{lecun2015deep}.

```
\subsubsection*{Learning}
```

```
Weight adjustments typically follow one of three approaches: supervised
learning, unsupervised learning, or self-supervised learning
\cite{schmidhuber2015deep}.
\begin{figure}[t]
\centering
\includegraphics[width=0.7\columnwidth]{neural.pdf}
\caption{Neural Network}\label{fig:Board1}
\end{figure}


\begin{itemize}
    \item \textbf{Supervised learning}: Training starts with random weight
values, and both the input data and desired target outputs are provided to the
network. During training, the network updates the weights to minimize the error
between its output and the target output.
    \item \textbf{Unsupervised learning}: Input data are provided without any
corresponding target outputs, and the network attempts to identify underlying
structures or patterns in the data.
    \item \textbf{Self-supervised learning}: The network autonomously monitors
and adjusts its own outputs, using feedback mechanisms to correct errors in the
data.
\end{itemize}

In all cases, when the network stops updating its weights, training is
considered complete. This typically occurs when the output error reaches zero or
approaches zero, indicating that the network has successfully learned the
desired mapping or representation.



\begin{table}[h]
\centering
\begin{tabular}{|l|p{8cm}|}
\hline
\textbf{Learning Paradigm} & \textbf{Description} \\
\hline
Supervised Learning & Learns from labeled examples maps inputs to known outputs.
\\
\hline
Unsupervised Learning & Identifies patterns or clusters in input data without
labeled outputs. \\
\hline
Self-Supervised Learning & Creates its own labels from the input data and uses
feedback to improve predictions. \\
\hline
\end{tabular}
\caption{Summary of neural network learning paradigms}
\label{tab:learning_paradigms}
\end{table}

%me kathe epifilajh
```

\subsubsection{Types of Neural Networks}

Artificial Neural Networks (ANNs) are core tools of machine learning, inspired by the biological structure and function of the human brain. There are various types of neural networks, each designed to address specific kinds of problems and data. Multilayer Perceptrons (MLPs) are among the most classical and widely used models, where information flows unidirectionally from the input to the output through hidden layers \cite{shrestha2019review}. Convolutional Neural Networks (CNNs) are particularly efficient for image and visual data processing, as they detect local patterns using filters and convolutional layers \cite{liu2017survey}. Recurrent Neural Networks (RNNs), including variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), are designed for sequential data and time-series processing, incorporating memory mechanisms that enable the modeling of long-term dependencies \cite{yu2019review}. Transformer networks represent a breakthrough in sequence modeling, particularly for natural language processing tasks, by employing self-attention mechanisms that allow parallel and global dependency modeling \cite{ wang2020survey}. Lastly, Generative Adversarial Networks (GANs) provide a novel approach to data generation by training two networks—the generator and the discriminator—in a zero-sum game framework \cite{creswell2018generative}. Each of these architectures has distinct strengths and limitations, supporting the selection of appropriate models based on the problem domain and data type.

\begin{table*}[ht]
\centering
\scriptsize
\begin{tabular}{|p{2cm}|p{2.5cm}|p{2.5cm}|p{2.5cm}|p{2cm}|}
\hline
\textbf{Network Type} & \textbf{Description} & \textbf{Typical ECG Applications} & \textbf{Advantages} & \textbf{Disadvantages} \\
\hline
\textbf{DNN \cite{raj2022ecg}} & Fully connected feedforward network with multiple dense layers & AF classification using Hjorth parameters; suitable for low-power devices & Simple, fast training, easy to implement & Limited sequence modeling capabilities \\
\hline
\textbf{CNN \cite{li2019cnn}, \cite{sundaravadivel2018smart}} & Convolutional filters for spatial feature extraction & Arrhythmia detection using spectrograms or 1D ECG filters & Automatic feature learning, noise-robust & High computational requirements \\
\hline
\textbf{RNN \cite{khan2022lstm}} & Sequential model with internal memory state & Heart rate variability (HRV), temporal ECG signal analysis & Captures temporal dependencies & Vanishing gradients, complex training \\
\hline
\textbf{LSTM \cite{farady2024ecg}} & Gated RNN for long-term dependencies & Beat prediction, long-duration ECG signal modeling & Learns long-range patterns effectively & Slower training, more parameters \\
\hline
\textbf{GRU \cite{cho2014gru}} & Simplified version of LSTM with two gates & Low-power ECG sequence modeling & Faster and lighter than LSTM & Slightly less

```
expressive than LSTM \\
\hline
\textbf{Transformer \cite{liu2024dpnet}} & Attention-based architecture without
recurrence & Multi-lead ECG classification with attention mechanisms & High
scalability, long-range pattern handling & Memory- and compute-intensive \\
\hline
\textbf{Autoencoder \cite{santos2021smart}} & Encoder–decoder architecture for
representation learning & ECG denoising, unsupervised feature extraction & Data
compression, anomaly detection & Not directly suitable for classification \\
\hline
\end{tabular}
\caption{Comparison of Neural Network Types in ECG Applications}
\end{table*}
```

```
\subsubsection{Deep learning}

Deep Neural Networks (DNNs) represent a fundamental class of machine learning
models, characterized by their multiple hidden layers that enable the
hierarchical extraction of complex patterns from data. These architectures have
demonstrated high accuracy and generalization capability, particularly in
structured data classification tasks and biomedical applications. In recent
years, DNNs have been widely applied to physiological signal analysis, such as
electrocardiograms (ECGs), where they facilitate the automated detection of
cardiac abnormalities with minimal preprocessing requirements. According to Raj
and Ray (2022), a hybrid DNN-based framework effectively distinguished atrial
fibrillation events using Hjorth parameters derived from ECG signals, achieving
high accuracy in low-complexity environments \cite{raj2022ecg}. Furthermore,
emerging studies highlight the role of DNNs in edge-AI scenarios, where
lightweight models are deployed on portable or wearable devices for real-time
health monitoring (Liu et al., 2021) \cite{huang2024efficient}. Despite their
advantages, DNNs are limited in modeling temporal dependencies compared to
sequential architectures, which often necessitates hybridization with time-aware
models for time-series biomedical data.


\section{Performance Management}
Performance management is a critical organizational process that drives the
system towards achieving optimal outcomes and enhancing overall efficiency and
productivity. It encompasses activities such as setting clear performance goals,
continuous monitoring, providing timely feedback, and evaluating performance,
thereby ensuring employee contributions align with strategic objectives
\cite{de2016performance}.

Modern approaches to performance management integrate advanced technologies such
as data analytics and artificial intelligence to deliver actionable insights,
```

forecast trends, and personalize development programs, thereby improving decision-making and workforce agility \cite{boudreau2017analytics, nankervis2006performance}. These innovations enable proactive identification of performance gaps and implementation of targeted interventions, fostering a culture of continuous learning and improvement \cite{pulakos2009performance}.

Overall, performance management acts as a bridge between strategic execution and organizational development, ensuring sustainable success in a dynamic and competitive environment.

\subsubsection{Evaluation Techniques in Neural Network Classification: Emphasizing ROC and Confusion Matrix Analysis}
Performance evaluation of artificial neural networks is a critical process for validating the accuracy and reliability of the model. Key evaluation tools include the confusion matrix and the Receiver Operating Characteristic (ROC) curve. The confusion matrix offers a detailed representation of true versus predicted classes, enabling the analysis of Type I (false positives) and Type II (false negatives) errors, as well as the calculation of metrics such as precision and recall \cite{swaminathan2024confusion}.
The ROC curve illustrates the trade off between the true positive rate and false positive rate across various decision thresholds, with the area under the curve (AUC) serving as a measure of the model's discriminative ability \cite{rumelhart1986learning}. Beyond these, other evaluation metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), F1 score, accuracy, and log loss are commonly employed to comprehensively assess neural networks, depending on the nature of the task—classification or regression \cite{rocha2007evolution, patel2022evaluation}. The combined use of these methods enables a thorough assessment and optimization of neural network performance in modern applications.

\subsection{Roc Curved}
The Receiver Operating Characteristic (ROC) curve was first used, introduced during World War II for the analysis of radar signals, and was developed as a tool to assess the ability of radar operators to distinguish between enemy and non-enemy aircraft in their radar screens. The pioneers of Roc Curved signal detection theory, which was initially developed in the 1940s and 1950s by researchers like Norbert Wiener and Julian Bigelow respectively.\cite{72}

In the next few years, the ROC curve has become more widely known and adopted in the fields of statistics and machine learning, as it has proven to be a valuable tool for evaluating the performance of binary classification models.

Since then, the ROC curve has become a standard method in various fields, including medicine, machine learning, and signal processing, for assessing and visualizing the trade-off between sensitivity and specificity in classification tasks.\cite{72,73,74,75}

The Receiver Operating Characteristic (ROC) curve is a key tool in the realm of statistics and machine learning, offering a nuanced evaluation of binary classification models. Visually depicting the trade-off between sensitivity and specificity at various threshold settings, the ROC curve plots the true positive

rate (sensitivity) versus the false positive rate (1 - specificity). Each point on this curve represents a distinct balance between accurately identifying positive cases and misclassify negative cases. The area under the ROC curve was denoted as AUC, quantify the overall performance of the predictive model. A larger value in the area occupied by the AUC region indicates superior discrimination ability. Using this graph helps users choose the optimal threshold values tailored to their specific application requirements because it provides valuable insights into the accuracy and reliability of the model in distinguishing between positive and negative cases.

The ROC system has applications in various fields, serving as a critical tool for evaluating and visualizing the performance of binary classification models. In many fields, especially in healthcare, ROC curves are used extensively to evaluate the effectiveness of diagnostic tests, thereby helping to determine optimal cutoffs for medical conditions.

As in machine learning, curves play a central role in providing deep insights into the performance of classification algorithms, helping practitioners make informed decisions about model thresholds. This is particularly evident when applied to healthcare, where the incorporation of machine learning and the use of an ROC curve synergistically contributes to improved decision-making processes.

In addition, receiver operating characteristic (ROC) curves are used in information retrieval systems to evaluate ranking algorithms and quality control procedures to optimize defect detection.

Finally, it extends to areas such as biology and bioinformatics and even to areas such as criminal justice and various other fields where the precise distinction between positive and negative hypotheses is paramount.

The ROC curve is a valuable and adaptable tool that contributes to improved decision-making and model evaluation in various fields.

Based on the aforementioned evidence, the ROC curve was used individually, and in and on each achievement separately, playing a crucial role in the comprehensive theoretical validation of our results. Its implementation is a key element that contributes substantially to the verification of results.
provide remote monitoring and diagnosis of heart conditions.
\cite{75,78,79,80,81,82}

\subsubsection{Receiver Operating Characteristic (ROC) Curve: Theory and Applications}

The Receiver Operating Characteristic (ROC) curve was developed during World War II to assess the ability of radar operators to distinguish between hostile and non-hostile aircraft\cite{72}. Norbert Wiener and Julian Bigelow pioneered the theoretical foundations for what would later become signal detection theory. In

recent decades, ROC curves have been widely applied in statistics, biomedical engineering, and machine learning as vital tools for evaluating the performance of binary classification models\cite{72,73,74,75}. Currently, this is the standard in fields, such as medicine, information retrieval, and quality control. This information was used to validate and process the data.

\subsubsection{Understanding the ROC Curve}

The ROC curve plots the \textbf{True Positive Rate (TPR)} on the y-axis against the \textbf{False Positive Rate (FPR)} on the x-axis with various classification threshold levels. This visual representation demonstrates the trade-off between sensitivity and specificity, and offers insight into the diagnostic or predictive accuracy of the model.

Each point on the ROC curve corresponds to a particular decision threshold. A model with excellent discrimination will have a curve that bows toward the top-left corner, indicating high sensitivity and a low false-positive rate. Conversely, a diagonal line from (0,0) to (1,1) indicates no discriminative ability (i.e., random guessing).

\subsubsection{Key Metrics}

\begin{table}[]
\centering
\begin{tabular}{|cc|cc|}
\hline
\multicolumn{2}{|c|}{}   & \multicolumn{2}{c|}{{\color[HTML]{000000} \textbf{Actual}}} \\ \cline{3-4}
\multicolumn{2}{|c|}{\multirow{-2}{*}{}}   &
\multicolumn{1}{c|}{{\color[HTML]{333333} Positive}} & {\color[HTML]{333333} Negative} \\ \hline
\multicolumn{1}{|c|}{}   & Positive &
\multicolumn{1}{c|}{\begin{tabular}[c]{@{}c@{}}True \\ Positive\end{tabular}} &
\begin{tabular}[c]{@{}c@{}}False\\ Negative\end{tabular} \\ \cline{2-4}
\multicolumn{1}{|c|}{\multirow{-2}{*}{\textbf{PREDICTED}}} & Negative &
\multicolumn{1}{c|}{\begin{tabular}[c]{@{}c@{}}False \\ Negative\end{tabular}} &
\begin{tabular}[c]{@{}c@{}}False\\ Negative\end{tabular} \\ \hline
\end{tabular}
\end{table}

\begin{itemize}
  \item \textbf{True Positive Rate (TPR)}, also known as sensitivity or recall, is defined as:
  \[
  TPR = \frac{TP}{TP + FN}
  \]
  where TP denotes true positives and FN denotes false negatives. The TPR measures the proportion of actual positives correctly identified by the model\cite{82,83,85,86}.

  \item \textbf{False Positive Rate (FPR)} is given by:
  \[
  FPR = \frac{FP}{FP + TN}

```
  \]
  where $FP$ denotes false positives and $TN$ denotes true negatives. The FPR
quantifies the proportion of actual negatives incorrectly identified as
positive\cite{82,83,85,86}.
\end{itemize}

\begin{figure}[htbp]
    \centering
    \includegraphics[width=1\textwidth]{predictionauc.pdf}
   \caption{TPR and FPR.}
\end{figure}

\begin{figure}[htbp]
    \centering
    \includegraphics[width=1\textwidth]{roc_curve_plot.pdf}
   \caption{ Random Roc Curved Plot}
\end{figure}

\subsubsection{Area Under the Curve (AUC)}

The \textbf{Area Under the Curve (AUC)} quantifies the overall ability of the
model to discriminate between positive and negative classes. AUC ranges from 0
to 1:
\begin{itemize}
  \item AUC = 1: perfect classification
  \item AUC = 0.5: no discriminative power (equivalent to random guessing)
  \item AUC < 0.5: worse than random
\end{itemize}

AUC is:
\begin{itemize}
  \item \textbf{Threshold-invariant:} It evaluates model performance over all
possible thresholds.
  \item \textbf{Scale-invariant:} It measures how well predictions are ranked,
rather than their absolute values\cite{134,135}.
\end{itemize}

\subsubsection{Applications of the ROC Curve}

ROC curves are widely used, as follows:
\begin{itemize}
  \item \textbf{Healthcare:} To evaluate diagnostic tests and choose optimal
thresholds for disease prediction.
  \item \textbf{Machine Learning:} For classifier evaluation and model
selection.
  \item \textbf{Information Retrieval:} To assess ranking algorithms.
  \item \textbf{Bioinformatics, criminal justice, and finance:} Where binary
classification is critical.
\end{itemize}



In this study, ROC curve analysis played a central role in the theoretical
validation of our fetal heart monitoring system, which was used to verify the
```

ability of the model to accurately distinguish between valid and invalid fetal pulse detections. The integration of ROC analysis substantially contributes to confirming the reliability of the measurements obtained through both ECG and PPG signals\cite{75,78,79,80,81,82}.

\subsection{Confusion Matrix}

The confusion matrix is a fundamental tool for evaluating the performance of classification models, including neural networks. It organizes the correct and incorrect predictions for each class by distinguishing true positives, false positives, true negatives, and false negatives \cite{swaminathan2024confusion}. Through the confusion matrix, key metrics such as accuracy, recall (sensitivity), specificity, and F1-score can be calculated, providing a comprehensive assessment of the model's performance across different scenarios \cite{rumelhart1986learning}. The use of the confusion matrix is especially critical in imbalanced class problems, where simple accuracy measurements can be misleading. Furthermore, analyzing the confusion matrix allows targeted improvements, such as addressing false negatives, which can have severe consequences in critical applications like medical diagnosis \cite{patel2022evaluation}. Overall, the confusion matrix remains an indispensable tool for the evaluation and optimization of neural network models.

\subsubsection{Confusion Matrix and Performance Metrics}

The confusion matrix is a fundamental tool for evaluating the performance of classification models, especially in binary classification tasks. It is typically represented as:

\begin{figure}[htbp]
  \centering
  \includegraphics[width=\linewidth]{ConfusionMatrix2.pdf}
  \caption{Confusion Matrix.}
  \label{fig:myfigure2}
\end{figure}

\[
\begin{bmatrix}
\text{TP} & \text{FP} \\
\text{FN} & \text{TN}
\end{bmatrix}
\]

Where:
\begin{itemize}
    \item \textbf{TP (True Positives):} Correctly predicted positive instances.
    \item \textbf{FP (False Positives):} Incorrectly predicted positive instances.
    \item \textbf{FN (False Negatives):} Incorrectly predicted negative instances (missed positives).
    \item \textbf{TN (True Negatives):} Correctly predicted negative instances.
\end{itemize}

Using these values, we can derive key performance metrics:

\begin{itemize}
    \item \textbf{Accuracy:}
    \[
    \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
    \]

    \item \textbf{Precision (Positive Predictive Value):}
    \[
    \text{Precision} = \frac{TP}{TP + FP}
    \]

    \item \textbf{Recall (Sensitivity or True Positive Rate):}
    \[
    \text{Recall} = \frac{TP}{TP + FN}
    \]

    \item \textbf{F1 Score (Harmonic Mean of Precision and Recall):}
    \[
    F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
    \]
\end{itemize}

According to the above confusion matrix figure,the confusion matrix provided the following values:
\[
TP = 53,\quad TN = 24,\quad FP = 1,\quad FN = 2
\]

Based on these:
\[
\text{Accuracy} = \frac{53 + 24}{53 + 24 + 1 + 2} = \frac{77}{80} = 96.25\%
\]
\[
\text{Precision} = \frac{53}{53 + 1} = \frac{53}{54} \approx 98.15\%
\]
\[
\text{Recall} = \frac{53}{53 + 2} = \frac{53}{55} \approx 96.36\%
\]
\[
F1 = 2 \cdot \frac{0.9815 \cdot 0.9636}{0.9815 + 0.9636} \approx 97.25\%
\]

These metrics confirm the system's strong classification capability in identifying heart disease instances accurately.

\section{Consumer Electronics}

\subsection*{Historical Background}

The development of consumer electronics (\textit{Consumer Electronics}) has its roots in the early 20th century, with the development of radio receivers and the widespread use of vacuum tubes. The invention of the transistor in 1947 by Bardeen, Brattain, and Shockley was a milestone, enabling the creation of portable radios and televisions, and marking the beginning of the semiconductor era \cite{ieee_milestones_transistor}.

The introduction of integrated circuits (ICs) in the 1960s brought about significant changes, enabling the development of more advanced and affordable devices, such as televisions, pocket computers, and, later, video game consoles and personal computers \cite{ieee_milestones_ic}.

The digitization of devices, starting in the 1980s with the advent of personal computers, has radically changed the operation and performance of consumer electronics, leading to improved image and sound quality \cite{immink_cd_history}.

\subsection{Modern Developments}

Today, consumer electronics incorporate advanced technologies, such as artificial intelligence (AI), the Internet of Things (IoT), edge computing, and 5G connectivity. These technologies enable the creation of smart devices that offer personalized experiences to users \cite{sundaravadivel2018smart}.

The global consumer electronics market is experiencing continuous growth, with an emphasis on energy-efficient, durable, and compact devices. The integration of AI into products such as televisions, home appliances, and wearable health devices is becoming increasingly widespread, responding to growing consumer demands \cite{marr2022}

\subsection{Integration of Advanced Technologies into Consumer Electronics}

Rapid advances in communication and computing technologies have radically transformed the consumer electronics sector, with particular emphasis on systems based on the \textit{Internet of Health-Centric Things (IoHcT)}. IoHcT systems enable remote monitoring of physiological parameters, disease prediction through artificial intelligence, and integration with mobile networks and cutting-edge computing units \cite{raj2020}.

The evolution of wireless technologies, from 3G networks to the emerging 6G, provides increased bandwidth, low latency, and mass device interconnection. 5G networks are already a key infrastructure for applications in smart cities, biomedical monitoring, and real-time AI computing at the edge (\textit{edge computing}) \cite{taleb20165G, IEEN6G2023}. The addition of WiFi 6 and WiFi technologies further improves stability and capacity for applications in home environments.

At the processing level, artificial neural networks are now being integrated into low-power devices using algorithms optimized to run on computers such as \textbf{Raspberry Pi}, \textbf{Arduino}, and \textbf{NVIDIA Jetson Nano /

Xavier} \cite{maji2021, alajlan2022}. These platforms are widely used to develop smart consumer applications, such as:

\begin{itemize}
\item Emotion detection with cameras and CNNs on Raspberry Pi.
\item Heart rate anomaly prediction using IoHcT sensors with ESP32 and data transfer over WiFi/4G.
\item Running YOLOv5 or EfficientNet neural models on Jetson Xavier for real-time image recognition.
\end{itemize}

The increased computing power combined with \textit{model quantization, pruning, and edge-optimized neural inference} techniques make it possible to implement advanced algorithms without the need for cloud infrastructure \cite{plastiras2018}. Also, new 6G networking standards emphasize \textit{semantic communication}, reducing latency and increasing efficiency in consumer environments \cite{IEEN6G2023,plastiras2018}.

\section{Developments boards}
Embarking on the journey of innovation, developers encounter a plethora of development boards, each distinguished by their unique features and capabilities. Arduino boards, known for their versatility, provide accessible entry into the realm of embedded systems. In contrast, Raspberry Pi is a robust miniature computer, fueling a diverse range of projects. Introducing a novel platform, BeagleBone excels in real-time connectivity and processing, while ESP32 caters to the Internet of Things (IoT) domain with its wireless capabilities. The NVIDIA Jetson series, renowned for its proficiency in artificial intelligence and machine learning, adds another dimension to the landscape. Whether opting for simplicity or pushing technological boundaries, the panorama of development boards becomes creators to craft their innovative narratives, intertwining distinct threads of ingenuity.
\subsection{Arduino uno R3}
Arduino Uno R3 is a cornerstone in the realm of microcontroller development boards. Powered by the versatile ATmega328P microcontroller, it boasts 14 digital input/output pins, 6 analog inputs, and a clock speed of 16 MHz. Its programmable nature, facilitated by the Arduino IDE, makes it accessible to both beginners and experienced developers. Equipped with USB connectivity, it enables seamless interface with computers for programming and power supply. The Uno R3's simplicity, combined with its open-source nature and a vibrant community, renders it an ideal platform for diverse projects, spanning from simple prototypes to intricate electronic applications.\cite{27,55,40}

\begin{enumerate}
    \item Microcontroller: Atmel ATmega328P.
    \item Clock Speed: 16 MHz.
    \item Flash Memory: 32 KB (ATmega328P) of which 0.5 KB is used for the boot loader.
    \item SRAM: 2 KB.
    \item EEPROM: 1 KB.
    \item Digital I/O Pins: 14 (of which 6 provide PWM output).
    \item Analog Input Pins: 6.
    \item DC Current per I/O Pin: 20 mA.

```
    \item DC Current for 3.3V Pin: 50 mA.
    \item Voltage Operating Range: 5V.
    \item Input Voltage (recommended): 7-12V.
    \item Input Voltage (limits): 6-20V.
    \item Onboard USB Interface: Type B.
    \item Power Supply: External (7-12V), USB (5V), or Battery.
    \item Operating Voltage: 5V.
    \item Analog Reference Voltage: 5V.
    \item Built-in LED: 13 (connected to digital pin 13).
    \item Size: 68.6mm x 53.4mm.
    \item Weight: 25g.

\end{enumerate}
\subsection{ESP- 8266}

ESP-8266 is a compact and powerful Wi-Fi module that has revolutionized the development of (Internet of Things) development. Its core integrates a Tensilica L106 32-bit microcontroller and offers a clock speed of 80 MHz. What sets the ESP8266 apart from its built-in Wi-Fi connectivity, making it an excellent choice for projects requiring wireless communication. GPIO pins for digital input/output and support for both I2C and SPI communication protocols provide versatility for various applications. The module also includes onboard memory, which is crucial for storing the firmware and data. Its low-cost, low-power consumption, and compatibility with the Arduino IDE have contributed to its popularity among hobbyists and professionals, enabling the creation of connected devices and smart applications with relative ease.\cite{41,70}
\begin{enumerate}
    \item Microcontroller: Tensilica L106 32-bit microcontroller.
    \item Clock Speed: 80 MHz.
    \item Operating Voltage: 3.3V.
    \item Digital I/O Pins: 17 GPIO pins.
    \item Analog Input Pins: 1 (3.3V max input).
    \item Flash Memory: 4 MB.
    \item Wi-Fi Connectivity: Integrated 802.11 b/g/n Wi-Fi.
    \item Wi-Fi Modes: Station, SoftAP, and Station+SoftAP.
    \item Wi-Fi Security: WPA/WPA2.
    \item Networking Protocol Support: TCP/IP.
    \item Serial Communication: UART.
    \item SPI Communication: Yes.
    \item I2C Communication: Yes.
    \item GPIO Output Current: 12 mA.
    \item Operating Temperature Range: -40°C to +125°C.
    \item Power Consumption: Varies depending on operation; can be very low in deep sleep mode.
    \item Size: Varies depending on the specific ESP8266 module or development board.
    \item Programming: Can be programmed using the Arduino IDE with the help of the ESP8266 board package.

\end{enumerate}

\subsection{NVIDIA Jetson Xavier}
The NVIDIA Jetson Xavier is a high-performance system-on-module (SoM) designed specifically for artificial intelligence (AI) and deep learning applications. It
```

is part of the NVIDIA Jetson family, which focuses on providing powerful computing solutions for edge devices. Here are some key specifications of the NVIDIA Jetson Xavier: \cite{129}
\begin{itemize}
    \item GPU Architecture: Equipped with an integrated NVIDIA Volta GPU with Tensor Cores, designed for accelerated deep learning and AI processing.
    \item CPU: Octa-core ARMv8.2 64-bit CPU complex, providing high-performance processing capabilities.
    \item DLA (Deep Learning Accelerator): Dedicated hardware accelerators designed to optimize and accelerate deep neural network inference.
    \item Memory: 16 GB 256-bit LPDDR4x RAM, providing substantial memory bandwidth for handling complex AI workloads.
    \item Storage: 32 GB eMMC 5.1 onboard storage for firmware and software.
    \item Connectivity: Multiple high-speed I/O interfaces, including PCIe, USB 3.1, and Gigabit Ethernet.
    \item Camera Inputs: Multiple MIPI CSI-2 camera interfaces for connecting cameras and performing computer vision tasks.
    \item Power Efficiency: Designed with a focus on energy efficiency to meet the requirements of edge devices.
    \item Form Factor: Compact form factor suitable for integration into various edge devices, such as robots, drones, and intelligent cameras.
NVIDIA Jetson Xavier is widely used in applications that require real-time AI processing on the edge, including robotics, autonomous vehicles, and smart surveillance systems. Its high-performance capabilities make it a popular choice for developers and researchers working on AI-driven projects at the edge.\cite{129}
\end{itemize}

\subsection{Exploring Add-On Modules for Development Boards}
Development boards often support additional modules called shields, which are add-on boards that provide additional functionality to the main board. Some examples of extra shields commonly used with development boards are as follows:
\begin{enumerate}
    \item Arduino Shields\cite{27,55,40}:
\begin{itemize}

    \item Motor Shield: Allows control of motors (servo, DC, or stepper) directly from an Arduino.

    \item Ethernet Shield:
    Ethernet connectivity is added to an Arduino for IoT projects.
    \item Bluetooth Shield: Enables wireless communication via Bluetooth.
GPS Shield: Allows integration of GPS functionality into Arduino projects.

    \item LCD Display Shield: Provides an easy way to add an LCD display to projects.

    \item SD Card Shield: Adds SD card storage capability to Arduino projects.
\end{itemize}
 \item Raspberry Pi HATs (Hardware Attached on Top)\cite{130}:
 \begin{itemize}
    \item Sense HAT: Equips the Raspberry Pi with environmental sensors and an

8x8 LED matrix.
    \item PoE HAT: Enables Power over Ethernet for the Raspberry Pi.
    \item Motor Driver HAT: Allows control of motors with the Raspberry Pi.
    \item Camera Module: Attaches to the Raspberry Pi for capturing images and video.
    \end{itemize}
 \item NVIDIA Jetson Modules \cite{129}:
\begin{itemize}
    \item Jetson GPIO Expansion Header: Provides additional GPIO pins for Jetson Nano.
    \item Jetson Camera Module: Connects to the MIPI CSI-2 interface for adding a camera to Jetson boards.
    \end{itemize}
 \item ESP8266 and ESP32 Modules:
\begin{itemize}
 \item ESP8266/ESP32 Relay Module: Enables control of high-power devices using relays.
 \item ESP8266/ESP32 TFT LCD Module: Adds color display to ESP8266 or ESP32 projects.
 \item ESP8266/ESP32 Sensor Modules: Various sensor modules for temperature, humidity, motion, etc.\cite{41,70}

These shields and modules extend the capabilities of development boards, allowing users to easily integrate new features into their projects without requiring extensive wiring or hardware modifications. This streamlines the development process and makes it more accessible to professionals working on diverse applications.

However, it is important to keep in mind that compatibility may vary. Therefore, it is essential to check whether a shield is compatible with a specific development board before integration.

\end{itemize}
\end{enumerate}

\subsection{Raspberry Pi 3}

The Raspberry Pi 3 has emerged as an outstanding development board, powered by a quad-core ARM Cortex-A53 processor clocked at 1.2 GHz. It is a compact yet high-performance computer that combines affordability with versatility.

Seamlessly bridging the worlds of the Internet of Things (IoT) and general-purpose computing, it features built-in wireless capabilities, including both Wi-Fi and Bluetooth, which significantly broaden its connectivity options.

An HDMI port provides high-definition video output, while four USB ports allow the connection of various peripherals. Additionally, the Raspberry Pi 3 includes a 40-pin GPIO header, offering extensive I/O capabilities for hardware interfacing and prototyping.

Its flexibility makes it ideal for a wide range of applications—from educational platforms and media centers to experimental projects by makers and professionals

alike. With its small form factor, low power consumption, and strong community support, the Raspberry Pi 3 remains a cornerstone of accessible innovation in embedded systems and IoT development.
 \cite{130,131}

\begin{enumerate}
    \item Processor:

Quad-core ARM Cortex-A53.
Operating Frequency: 1.2 GHz.
    \item Memory (RAM):

1 GB LPDDR2.
    \item Wireless Connectivity:

802.11n Wireless LAN (Wi-Fi).
Bluetooth 4.2.
    \item Ethernet:

10/100 BaseT Ethernet socket.
    \item USB Ports:

4 x USB 2.0 ports.
    \item Video Output:

1 x HDMI (rev 1.3 \& 1.4).
    \item Audio Output:

3.5mm jack, HDMI.
    \item GPIO Pins:

40-pin GPIO header for general-purpose input/output.
    \item Storage is provided via a microSD card slot, which is used to store the operating system as well as user data.
    \item Power Supply:

5V/2.5A DC via a micro USB connector.
    \item Various operating systems are supported, including Raspberry Pi OS (formerly known as Raspbian) and NOOBS, among others.
    \item Dimensions:

85.6mm x 56.5mm.
\end{enumerate}The Raspberry Pi 3 is a versatile and widely used single-board computer, known for its affordability and compatibility with a variety of applications, ranging from DIY projects to educational initiatives.

\subsection{Raspberry Pi 4}

The Raspberry Pi 4 is the evolution of the Raspberry Pi 3 development platform, redefining expectations with its significantly enhanced specifications. It is powered by a quad-core Cortex-A72 processor running at 1.5 GHz and is available in versions with 2 GB, 4 GB, or 8 GB of RAM, making it a true computing powerhouse.

It features dual micro-HDMI outputs that support up to 4K resolution, enabling high-definition displays for advanced visual applications. Additionally, USB 3.0 ports and a Gigabit Ethernet interface significantly improve data transfer and connectivity speeds.

The Raspberry Pi 4 retains the familiar 40-pin GPIO header, supporting hardware interfacing and prototyping. A notable improvement is the transition to a USB-C power supply, which provides a more stable and reliable power source compared to its predecessors.

Raspberry Pi 4 remains at the forefront of low-cost computing innovation, powering a wide range of projects—from DIY electronics and educational tools to advanced applications in fields like neural networks, computer vision, facial recognition, and color detection.
\cite{130,131} \cite{132}

\section{ECG and PPG Historical and Clinical Perspectives of the Electrocardiogram }

The electrocardiogram (ECG) represents a fundamental milestone in the evolution of cardiovascular diagnostics. Its genesis can be traced to the pioneering efforts of Dutch physiologist Willem Einthoven, who, in the late 19th century, developed the first practical device for recording cardiac electrical activity—the string galvanometer \cite{57}. This innovation enabled the detection and amplification of the heart's bioelectrical signals using a fine conductive filament suspended within a magnetic field. Einthoven's seminal publication in 1903 formally introduced the nomenclature of the ECG waveform components—P, Q, R, S, and T waves—thereby laying the foundation for modern cardiac electrophysiology. His contribution was recognized with the Nobel Prize in Physiology or Medicine in 1924. Prior to Einthoven, Augustus Waller had recorded the first human ECG in 1887, and Thomas Lewis significantly advanced the clinical interpretation of electrocardiographic data \cite{57}.

Today, the ECG remains an indispensable, noninvasive diagnostic modality for the evaluation of the heart's electrical activity. By capturing the voltage fluctuations associated with cardiac depolarization and repolarization, ECG enables clinicians to assess heart rate, rhythm, and conduction, as well as detect a spectrum of pathophysiological conditions including myocardial ischemia, infarction, electrolyte disturbances, and structural abnormalities such as cardiomyopathies and hypertensive heart disease \cite{59,60,61,62,63,64,65,66}. Standard ECG acquisition involves the placement of surface electrodes on the chest and limbs, allowing for real-time waveform visualization in various lead configurations. Its simplicity, accessibility and diagnostic scope make it a vital solution in both acute and preventive cardiology, facilitating early detection, risk stratification and evidence-based clinical management.

\subsection{Signal Processing and Filtering Techniques in ECG and PPG: A

Comparative Perspective}

Electrocardiography (ECG) and photoplethysmography (PPG) are two fundamental non-invasive methods for monitoring cardiac function. While ECG captures the electrical activity of the heart using surface electrodes, PPG measures blood volume changes through light absorption using optical sensors. Both signals are susceptible to various types of noise—such as power-line interference, motion artifacts, and baseline wander—necessitating the application of filtering techniques to ensure diagnostic accuracy \cite{charlton2021ppg, khalid2023ppg}.

To reduce noise and enhance signal quality, several digital filtering methods are employed. Butterworth filters are widely used due to their flat frequency response in the passband and their minimal signal distortion, making them ideal for removing low- or high-frequency noise from ECG and PPG signals \cite{dauda2020comparative}. Chebyshev filters, particularly of the first kind, offer sharper roll-off characteristics at the expense of passband ripple. For signals with significant variability like PPG, Savitzky-Golay filters provide smoothing while preserving morphological features \cite{dauda2020comparative}. Additionally, notch filters—especially around 50/60 Hz—are implemented to eliminate power-line interference. More advanced approaches such as wavelet transforms allow for multi-resolution analysis, enabling simultaneous time and frequency localization of noise and signal components \cite{romero2018baseline}.

Comparatively, ECG offers superior diagnostic precision in detecting arrhythmias, conduction disorders, and ischemic events, while PPG is more prone to external disturbances such as motion, ambient light variation, and poor sensor contact. However, PPG excels in ease of integration into wearable systems and supports continuous, long-term monitoring in ambulatory settings \cite{charlton2021ppg}.

In conclusion, effective signal processing for ECG and PPG requires tailored filtering techniques based on signal characteristics and application context. The combination of these filtering methods with modern computational tools enables robust feature extraction and enhances the reliability of physiological monitoring in intelligent healthcare environments \cite{khalid2023ppg, romero2018baseline}.

\subsection {Comparison of ECG and PPG-Based Systems in Biomedical Applications}
Electrocardiography (ECG) and photoplethysmography (PPG) are among the most widely used non-invasive biosignals for monitoring cardiovascular health. While ECG directly measures the electrical activity of the heart, PPG relies on the optical detection of volumetric changes in blood flow. Each modality offers unique advantages: ECG provides precise temporal features for arrhythmia detection, while PPG enables low-cost, wearable solutions for continuous pulse monitoring. The following table presents a comparative overview of the technical features and biomedical applications of these signals.

\subsection {ECG and PPG Sensors: A Comparative Framework for Biomedical Monitoring}

\begin{figure}[htbp]
  \centering
  \includegraphics[width=\linewidth]{PGRST_Signal.pdf}

```
  \caption{PQRST Signal}
  \label{fig:myfigure}
\end{figure}


\begin{figure}[ht]
\centering
\includegraphics[width=1\textwidth]{PPG_Sens.png}
\caption{Working principle of PPG sensors~\cite{elgendi2012}.}
\label{fig:ppg_principle}
\end{figure}
```

Electrocardiography (ECG) and photoplethysmography (PPG) represent two foundational techniques in non-invasive cardiovascular monitoring, each offering distinct advantages across clinical and wearable health contexts. ECG, widely regarded as the gold standard, captures the electrical depolarization of myocardial tissue, yielding high-fidelity information such as P, QRS, and T waves, PR and QT intervals, and supports detailed analysis of arrhythmias and heart rate variability (HRV) \cite{kaur2021, narotamo2024}. Conversely, PPG is an optical method that detects volumetric changes in peripheral blood flow using light-emitting diodes and photodetectors, offering a compact, low-cost solution ideal for wearable applications and continuous pulse monitoring \cite{zhou2023, song2021}. In the context of maternal health monitoring, the PPG sensor has demonstrated utility in fetal heart rate tracking with the added benefit of enabling maternal mobility, while maintaining consistency when compared to ECG-derived measurements \cite{xu2024}. Technically, ECG operates in a wider frequency range (0.05–100 Hz) with higher sampling rates (250–1000 Hz), whereas PPG functions within 0.5–5 Hz and is typically sampled at 30–250 Hz \cite{kaur2021, peng2014}. However, PPG is more susceptible to motion artifacts, whereas ECG requires stable electrode contact. The combined use of both signals—especially in IoHT (Internet of Health Things) systems—enables robust multimodal monitoring and diagnostic augmentation. Recent studies utilizing deep learning and transformer-based models have shown that ECG-PPG fusion improves reliability in real-time health analytics and cardiovascular anomaly detection \cite{xu2024}. Thus, integrating these biosignals contributes significantly to the evolution of intelligent, adaptive, and context-aware biomedical monitoring systems.

```
\begin{table*}
[htbp]
\centering
\label{tab:ecg_ppg_comparison}
\resizebox{\linewidth}{!}{%
\begin{tabular}{|p{3.5cm}|p{5.5cm}|p{5.5cm}|}
\hline
\textbf{Feature} & \textbf{ECG} & \textbf{PPG} \\
\hline
\textbf{Signal Origin} & Electrical depolarization of the myocardium & Optical detection of blood volume changes \\
\hline
\textbf{Typical Frequency Range} & 0.05 – 100 Hz & 0.5 – 5 Hz \\
\hline
```

\textbf{Sampling Rate} & 250 – 1000 Hz \cite{kaur2021} & 30 – 250 Hz \cite{peng2014} \\
\hline
\textbf{Key Components} & P, QRS, T waves; PR and QT intervals \cite{narotamo2024} & AC/DC components; waveform morphology \cite{zhou2023} \\
\hline
\textbf{Motion Sensitivity} & Moderate (electrode contact loss) & High (movement artifacts) \cite{peng2014} \\
\hline
\textbf{Form Factor} & Requires contact-based multi-lead setup & Easily embedded in wearable devices \cite{xu2024} \\
\hline
\textbf{Main Applications} & Arrhythmia detection, myocardial ischemia, HRV analysis \cite{kaur2021} & Heart rate estimation, blood oxygen saturation, blood pressure trends \cite{song2021} \\
\hline
\textbf{Combined Use Potential} & Multimodal biometric authentication, robust HR monitoring \cite{xu2024} & Improves accuracy when combined with ECG \cite{xu2024} \\
\hline
\end{tabular}
}
\caption{Comparison of ECG and PPG in terms of technical features and application domains}
\end{table*}

\section{Wireless Telecommunications and ECG Signals}
\label{intro:Wireless telecommunications and ECG signals}

The convergence of wireless telecommunications and electrocardiogram (ECG) signal processing has enabled powerful architectures for remote and continuous cardiac health monitoring. Modern systems utilize wireless transmission modules to send real-time ECG data from patients to cloud-based platforms or remote monitoring centers, where signal processing techniques—such as Discrete Wavelet Transform (DWT)—and machine learning models are employed for advanced interpretation \cite{erhani2020}.

The proliferation of Internet of Things (IoT) technologies further strengthens this paradigm by facilitating seamless integration of biomedical sensors and embedded systems, allowing uninterrupted acquisition and transmission of ECG signals over wireless networks. This real-time connectivity provides clinicians with timely access to critical data and supports the automated detection of cardiac anomalies, such as arrhythmias or ST-elevation myocardial infarction (STEMI), using intelligent algorithms \cite{huda2020}.

In particular, Internet of Health Things (IoHT) infrastructures leverage neural networks and deep learning models to classify ECG signals with high precision, often outperforming traditional rule-based diagnostics \cite{gokila2023}. Supervised learning frameworks, trained on large annotated ECG datasets, can distinguish normal and pathological heart rhythms, including atrial fibrillation, bradycardia, and tachycardia, with clinically acceptable accuracy .

The advent of sixth-generation (6G) wireless communication technology is

expected to bring disruptive advancements across various sectors, with healthcare—particularly smart hospitals—being a key beneficiary. This study explores the transformative potential of 6G in healthcare, focusing on its architectural foundations and enabling technologies. Through a comprehensive review of contemporary technological trends, integration frameworks, and system-level strategies, a novel smart hospital model is proposed. The model incorporates core innovations including the Internet of Things (IoT), artificial intelligence (AI), blockchain, robotics, telemedicine, and advanced data analytics.

Findings suggest that 6G's ultra-low latency, massive device connectivity, and enhanced data throughput can significantly improve patient care, real-time monitoring, and operational efficiency within hospital environments. However, despite its promising capabilities, several challenges persist, such as data privacy and security risks, system interoperability issues, and ethical concerns. The study highlights the urgent need for robust regulatory frameworks and standardized protocols to ensure the secure and ethical deployment of 6G technologies in healthcare contexts. \cite{kumar2025integrating}.

\textit{In summary}, the integration of wireless telecommunications, IoT infrastructure, and artificial intelligence into ECG monitoring creates a transformative ecosystem for cardiovascular healthcare. Such systems promote patient autonomy, optimize clinical workflows, and lay the groundwork for intelligent, scalable, and predictive remote healthcare platforms.

\subsection{Discrete Wavelet Transform (DWT)}

Following the theoretical discussion on the significance of the Discrete Wavelet Transform (DWT) in ECG signal analysis, we now focus on its practical implementation using MATLAB. MATLAB offers a powerful Wavelet Toolbox that provides a wide range of functions for DWT, enabling effective signal decomposition and feature extraction.

There are different ways to implement a DWT, but the most straightforward approach is to use a built-in library or toolbox. Popular environments that support the implementation include MATLAB's Wavelet Toolbox, Py-Wavelets in Python, \texttt{scipy.signal}, and \texttt{scikit-image}.

In MATLAB, the \texttt{dwt} function was used to theorize a Discrete Wavelet Transform. This function uses two arguments: the input signal of the desired wavelet type. For example, the following code performs a DWT on a signal \texttt{x} using the \textbf{Daubechies} wavelet of order four:

```
\begin{lstlisting}[language=Matlab]
[c, l] = dwt(x, 'db4');
\end{lstlisting}
```

Alternatively, for structured decomposition into approximation and detail coefficients:

```
\begin{lstlisting}[language=Matlab]
[cA, cD] = dwt(x, 'db4');
\end{lstlisting}

Where:
\begin{itemize}
  \item \textbf{x} is the input signal,
  \item \textbf{wname} is the wavelet name (e.g., 'db4'),
  \item \textbf{cA} and \textbf{cD} are the approximation and detail
coefficients respectively.
\end{itemize}
\cite{102}

\subsubsection*{The DWT Decomposition Level}
The number of decomposition levels in the Discrete Wavelet Transform (DWT)
depends on the length of the input signal and the level of detail desired.
Mathematically, the \textbf{maximum number of decomposition levels} is given by:

\[
\log_2(N)
\]

where \(N\) denotes the length of the signal (i.e., the number of samples).

For example, if the signal contains 256 samples, the maximum number of levels is
calculated as:

\[
\log_2(256) = 8
\]

This means the signal can be decomposed into up to 8 levels, each corresponding
to a different frequency and time resolution.

In practice, fewer levels may be sufficient if the signals are simple.
Otherwise, higher levels provide more detail but at a higher computational cost.
In ECG signal retrieval, wavelets such as \textbf{Daubechies (db4, db8)} are
preferred due to their ability to detect the QRS complex, even at low levels of
decomposition.
\cite{103,104,105}

\subsubsection*{My Project}

In our case, following the PhysioNet.org library the sampling rate is set to 128
samples per second. For a 60 second signal we have:
\[
128 \times 60 = 7,680 \text{ samples}
\]

Therefore, the theoretical maximum decomposition level is:

\[
\log_2(7680) \approx 12.9 \Rightarrow \textbf{12 levels}
```

\]

\subsubsection{Wavelet Level Decomposition Considerations}

Choosing the optimal decomposition level involves considering the following factors, which guide the appropriate configuration:
\begin{itemize}
  \item \textbf{Signal-to-noise ratio (SNR):} Higher SNR permits deeper decomposition.
  \item \textbf{Compression:} More levels allow better quantization and coefficient thresholding.
  \item \textbf{Signal characteristics:} Different signals benefit from decomposition at specific scales.
  \item \textbf{Computation limits:} Higher levels demand more processing power The commond{itemize}

Common thresholding techniques include \textit{VisuShrink}, \textit{Minmax}, and \textit{SureShrink}. \cite{106,107,108}

\subsubsection{Wavelet Decomposition in ECG}

The application of wavelet analysis is particularly effective in ECG signal processing due to its ability to:

\begin{itemize}
\item Isolate the QRS complex, as well as the P and T waves.
\item Support accurate beat classification (e.g., premature ventricular contractions (PVCs), arrhythmias).
\item Enhance data compression and noise reduction (denoising).
\end{itemize}

\textbf{Daubechies wavelets} are especially well-suited for ECG analysis because of their sensitivity to sharp transitions, such as those found in QRS complexes. For optimal results, wavelet decomposition should be combined with complementary techniques such as filtering and feature extraction, particularly in applications like fetal ECG extraction and interpretation.\cite{110,111,112}

\subsubsection*{Conclusion}

The Decrece Wavelet Transform (DWT) is a powerful method for ECG analysis. Appropriate wavelet selection (e.g., db4) and decomposition levels allow for efficient cardiac signal examination, enabling real-time diagnostics and reliable feature extraction. It should be noted that signal processing using the discrete wavelet transform (DWT) is recommended for complex biosignals. This is due to the fact that it offers greater energy compression and improved localization in the time and frequency domains than traditional Hjorth parameters.
However, in the context of applications that are capable of functioning reliably (Hjorth Parameters), and when utilising elementary signals, its employment in signal analysis can be considered valid. An attempt was made to implement both signals at the laboratory level, and it was decided to reserve the signal Hjorth Parameters with the lower energy footprint.

\section{Hjorth Parameters for Biomedical Signal Analysis}
\label{sec:hjorth_parameters}

The Hjorth parameters constitute a compact yet powerful set of time-domain descriptors, initially introduced by Bo Hjorth in 1970, for the quantitative characterization of biomedical signals such as electroencephalogram (EEG) and electrocardiogram (ECG) signals~\cite{hjorth1970eeg}. These parameters include \textit{\textbf{Activity}}, \textit{\textbf{Mobility}}, and \textit{\textbf{Complexity}}, each representing distinct attributes of signal behavior: signal power, frequency content, and signal shape variability, respectively.

In contrast to frequency-based methods such as Fourier or wavelet analysis, Hjorth parameters offer a computationally efficient, interpretable approach that is particularly well suited for real-time systems and embedded health-monitoring devices~\cite{khan2025hybrid}. Their application has been extensively studied in biosignal classification tasks, such as seizure detection, arrhythmia classification, and sleep-stage analysis, often serving as input features for machine learning and deep learning models~\cite{alawee2023advancing}.

In ECG analysis, the Hjorth descriptors help quantify transient changes in heart dynamics, and when combined with other nonlinear or morphological features, they enhance model performance in identifying pathological conditions~\cite{rizal2015}. Recent studies have demonstrated their robustness under noise and motion artifacts, making them ideal for Internet of Health Things (IoHT) applications and wearable monitoring systems~\cite{khan2025hybrid}.

\textbf{In summary}, the Hjorth parameters offer a valuable, low-complexity feature extraction tool in biomedical signal processing, with proven utility in both traditional and AI-driven cardiovascular analysis pipelines.

\chapter{ Preparing Data}

\section*{Introduction}

The rapid advancement of the Internet of Things (IoT) has fundamentally transformed the way data is collected, transmitted, and analyzed from the physical environment. Sensors, development platforms, and networking technologies synergize to form distributed monitoring and automation systems, with applications across critical sectors such as healthcare, industry, agriculture, and smart cities \cite{garg2020iotcloud}. This section examines the fundamental architectural components of such systems, focusing on signal processing stages, the role of development platforms, connectivity technologies, and the integration of cloud services.

\subsection{Sensors and Signal Analysis in IoT}

Sensors constitute the primary data acquisition interface in IoT environments,

converting physical quantities into electrical signals. The processing flow comprises the following stages: (1) raw signal acquisition, (2) preprocessing, (3) feature extraction, and (4) interpretation or decision-making \cite{goyal2021iot}. Proper signal processing is essential for optimizing the system’s energy efficiency and computational performance \cite{ma2019survey}.

Accurate sensor calibration, the utilization of open-source software, and the application of signal processing techniques—such as filtering, decoding, and cognitive analysis—ensure measurement reliability \cite{perera2019iot}. Data aggregation is typically performed at a sink node and analyzed on development boards according to system specifications \cite{ahmed2018iotreview}.

\subsection{Development Platforms in IoT Systems}

Microcontrollers such as the ESP32, Raspberry Pi, and Arduino constitute the hardware backbone of IoT systems, enabling edge-level computational capabilities. The ESP32 is particularly distinguished for its low power consumption and integrated wireless communication features \cite{li2022energyaware}, while Arduino is suitable for real-time control applications and Raspberry Pi supports complex local processing and machine learning workloads \cite{malik2020raspberry}.

Communication with interfaces such as LEDs, APIs, and the Internet enables both local and remote utilization of the data \cite{bonomi2021iotcloud}, while ensuring that system security, power efficiency, and firmware robustness remain critical considerations \cite{ahmed2018iotreview}.

\subsection{Networking Technologies for IoT Platforms}

The performance and communication range of an IoT system are determined by the selected networking technology. Common options include Wi-Fi, cellular (3G/4G/5G), Ethernet, and LPWAN protocols (e.g., LoRaWAN, NB-IoT) \cite{elhadi2018comparative}. Each option offers advantages and limitations depending on the specific application requirements \cite{valecce2020nbiot, centenaro2019lora}.

In the experimental configurations of this study, Wi-Fi was selected due to its availability, high speed, and ease of deployment, enabling reliable transmission and cloud connectivity at minimal implementation cost \cite{zafar2018iot}.

\subsection{Cloud Integration in IoT Systems}

Connecting IoT devices to cloud infrastructures via protocols such as MQTT and HTTPS enables scalable storage, remote processing, and machine learning integration \cite{chen2020mqtt}. Platforms like ThingSpeak and AWS IoT provide capabilities for over-the-air (OTA) updates, monitoring, and real-time data analytics \cite{bonomi2021iotcloud}.

Furthermore, such systems incorporate rule engines, security mechanisms, and monitoring capabilities without relying on extensive local infrastructure \cite{garg2020iotcloud}.

\subsubsection{Artificial Intelligence Integration in IoT Applications}

Artificial Intelligence (AI) plays a critical role in signal analysis, particularly within biomedical domains. Following digitization and preprocessing, features such as Hjorth parameters, time- and frequency-domain statistical descriptors, and Discrete Wavelet Transform (DWT) coefficients can be extracted to serve as input to machine or deep learning models \cite{ma2019survey, oh2018automated}.

Training neural networks for biomedical tasks such as ECG classification requires supervised learning procedures using annotated datasets. Model weights are iteratively updated via the backpropagation algorithm, often in combination with stochastic gradient descent (SGD) or adaptive optimizers such as Adam and RMSprop \cite{kingma2014adam, goodfellow2016deep}. Loss functions such as binary or categorical cross-entropy are chosen based on the classification scope (binary or multi-class). Regularization techniques—including dropout, L2 penalties, and early stopping—are applied to reduce overfitting and promote generalization \cite{srivastava2014dropout}.

In the context of temporal signals like ECG, training efficacy can be further enhanced using mini-batch training, adaptive learning rate schedules, and signal-specific data augmentation (e.g., lead inversion, noise injection, or temporal stretching) \cite{hannun2019cardiologist}. K-fold cross-validation is widely adopted to improve robustness, especially when the training dataset is limited \cite{goldberger2000physiobank}. Moreover, transfer learning strategies—where pre-trained networks are fine-tuned on biomedical datasets—have demonstrated efficacy in boosting accuracy while reducing training time \cite{rajpurkar2017cardiologist}. Hybrid approaches combining manually extracted wavelet-based features with automatically learned representations further improve detection sensitivity and specificity in arrhythmia classification \cite{acharya2018deep}.

The article titled "A Machine Learning-Based Heart Disease Detection Scheme for Predicting Spontaneous Termination of Atrial Fibrillation" proposes a system for detecting and predicting the spontaneous termination of atrial fibrillation (AF) through ECG analysis and the use of an artificial neural network (DNN). Below I provide a detailed overview of the key points and details of the neural network. The network received Hjorth parameters as input and consisted of eight hidden layers with 32 ReLU neurons, an additional layer with 16 neurons, 50\% dropout, and a sigmoid output neuron. Binary cross-entropy was used as the loss function, and the Adam optimizer guided training. The model achieved 94\% accuracy, with inference latency ranging from 0.96 ms (GPU) to 550 ms (microcontroller), maintaining low energy consumption.

The proposed training methodology was carefully adapted to the problem’s complexity, data constraints, and model architecture to ensure high diagnostic relevance and deployment feasibility on low-cost embedded hardware.

\subsection{Conclusions}

The architecture of an IoT system is built upon the interaction of interconnected subsystems—sensors, development platforms, communication technologies, and cloud services. This study highlighted the importance of a holistic design approach, emphasizing reliable preprocessing, appropriate hardware selection, and secure, scalable, and energy-efficient communication infrastructure. Leveraging the computational power of edge platforms and

artificial intelligence is anticipated to further enhance the autonomy and intelligence of next-generation IoT applications.

\section{Machine Learning's Versatility:
A Cross-Industry Revolution}
Machine learning has witnessed remarkable applications across diverse domains, revolutionizing the manner in which tasks are addressed. It is instrumental for diagnostic accuracy, treatment optimization, and disease prediction.

The use of machine learning in finance is expected to increase the number of companies that realize the benefits of this technology.

The technology sector utilizes machine learning for recommendation systems, fraud detection, and natural language processing. Additionally, in autonomous vehicles, machine learning algorithms enhance navigation and object recognition. The versatility of machine learning continues to reshape industries, offering innovative solutions and driving unprecedented advancements. \cite{113,114,115}

\subsection{Neural Network Architectures for Effective ECG Signal Analysis
}

Various neural networks have demonstrated efficacy in the examination of ECG signals, and optimal selection depends on the specific task and data characteristics. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNN) are widely used in ECG signal processing. CNNs excel at analyzing signals and images, autonomously learning spatial hierarchies of features, and achieving success in ECG classification tasks such as distinguishing between normal and abnormal signals or different arrhythmia's. On the other hand, RNNs, particularly Long Short-Term Memory networks, directly handle sequential data such as ECG signals. Their capacity to maintain a hidden state enables them to recall previous inputs, making them suitable for tasks where data order is crucial, including ECG arrhythmia detection, beat classification, and time series forecasting.\cite{112,113,114,115,116}

In addition, although less common, other architectures such as autoencoders, generative adversarial networks (GANs), and attention mechanisms have also been employed in ECG signal analysis. The optimal choice of architecture largely depends on the complexity of the data and the specific problem at hand. Frequently, a combination of multiple architectures, or a hybrid model (e.g., combining convolutional neural networks (CNNs) with recurrent neural networks (RNNs)), is used to capture both spatial and temporal features of ECG signals.

Cross-validation and experimentation with different architectures are highly recommended in order to determine the most effective model for a particular use case.

A notable neural network architecture for ECG signal analysis is the one-dimensional Convolutional Neural Network (1D CNN), which is particularly well-suited for processing sequential biomedical data. This architecture has proven effective in tasks such as heartbeat classification, R-peak detection, and arrhythmia diagnosis.

An example is the Deep-ECGNet, a model specifically designed for ECG signals, which extracts informative features to aid in the classification process. A 1D CNN operates on one-dimensional data sequences, applying convolutional layers with 1D filters to extract local features and patterns. These layers are followed by nonlinear activations and pooling operations to enhance feature representation and reduce dimensionality.

Subsequently, fully connected layers perform classification or regression based on the learned features. One key advantage of 1D CNNs is their adaptability to input sequences of varying length, making them suitable not only for ECG analysis but also for applications in fields like natural language processing and speech recognition.
\cite{117,118,119,120}

\subsection{Training a neural network}
When training a neural network to classify ECG signals, an important pre-processing step is to decompose the signals into subbands using the Discrete Wavelet Transform (DWT).

This decomposition separates the signal into different frequency components, which can be used as input features for the neural network. Each level of DWT decomposition corresponds to a specific scale or frequency range of the ECG signal, and the resulting wavelet coefficients provide valuable information about the signal's spectral content.

By using multiple levels of DWT decomposition, more detailed information about the ECG signal can be extracted. For example, the lowest level of decomposition (known as the approximation coefficients) captures the overall trend of the signal, while the higher levels (detail coefficients) represent the small-scale variations and rapid transitions.

Decomposing the signal into various levels allows the extraction of features at different resolutions, which can then be input into the neural network. This multi-resolution analysis can help the network learn more complex and fine-grained representations of the ECG signal, potentially improving its classification and detection performance, especially in challenging recognition scenarios.

It is important to note that the optimal number of decomposition levels depends on the characteristics of the ECG signals and the specific task being addressed. DWT decomposition is just one step in the pre-processing pipeline for ECG signal analysis. Other techniques, such as normalization, data augmentation, and feature selection, are also commonly applied to enhance the performance of neural networks.
\cite{121,122,123}
\subsection{Creating a Neural Network Based on ECG Signals}

Creating a neural network to process ECG signals is a multi-step process. An overview of the general steps is as follows:

\textbf{1. Collect and preprocess the data:}
Obtain a dataset of ECG signals along with the corresponding labels (e.g.,

normal vs. abnormal beats). Perform essential pre-processing steps such as \textbf{filtering} to remove noise and \textit{resampling} to ensure a consistent sampling rate across signals.

\textbf{2. Define the architecture:}
An appropriate 1D Convolutional Neural Network (CNN) architecture must be selected for the specific task. This may include one or more 1D convolutional layers, followed by pooling layers, dropout layers, and one or more fully connected layers to extract and process spatial features from the ECG signals.

\textbf{3. Implement the neural network:}
Using a deep learning framework such as TensorFlow, Keras, or PyTorch, the selected architecture can be implemented. For example, in Keras, the \texttt{Sequential} API can be used to stack layers together in a straightforward manner \cite{121,122,123}.

\textbf{4. Train the network:}
In this step, the preprocessed data is used to train the implemented neural network. Various optimization techniques have been reported in the literature to improve convergence and generalization. For multiclass classification problems, categorical cross-entropy is commonly used as the loss function \cite{123}.

\textbf{5. Test the network:}
After the training process is completed, a separate test dataset of ECG signals is used to evaluate the network’s performance. Metrics such as accuracy, F1-score, and AUC are calculated to quantitatively assess the model's effectiveness.

\textbf{6. Fine-tune and optimize:}
At this stage, the most suitable architecture is selected, and hyper-parameters such as the learning rate, batch size, and number of layers are adjusted accordingly. This step aims to ensure that the network achieves the best possible performance.

\section{Tools for Implementing Convolutional Neural Networks}

The implementation of a Convolutional Neural Network (CNN) requires a set of tools and resources. The main tools typically include:

\begin{itemize}
  \item A programming language
  \item A deep learning library
  \item A dataset
  \item A computational platform
  \item A neural network visualization tool (optional)
  \item A package manager
\end{itemize}

\textbf{1. Programming language:}
A programming language is required to develop and implement CNNs. Python is the most widely used language in deep learning and is supported by major libraries such as TensorFlow, Keras, PyTorch, and Caffe.

\textbf{2. Deep learning library:}
Libraries such as TensorFlow, Keras, and PyTorch offer the necessary tools to define, train, and evaluate CNNs. Each library has its own strengths and is selected based on the user's preference and the specific project needs.

\textbf{3. Dataset:}
Training and testing of CNNs require a dataset. The data must be preprocessed and properly formatted (e.g., CSV, HDF5) for compatibility with the selected deep learning framework. Public datasets, such as those available on PhysioNet.org, are often used in biomedical applications.

\textbf{4. Computational platform:}
CNNs can be trained on local machines with suitable GPUs or using cloud-based services such as Google Colab, AWS, or Azure. For large-scale models or datasets, high-performance platforms like GPU clusters may be required.

\textbf{5. Neural network visualization tool:}
Tools like TensorBoard and Visdom are useful for visualizing training metrics, debugging issues, and monitoring the performance of neural networks in real-time.

\textbf{6. Package manager:}
Package managers like \texttt{pip} or \texttt{conda} are essential for installing and managing the necessary libraries and dependencies. The specific tools may vary based on the requirements of the project and personal preferences.

\subsubsection{Online Tools}

Several online platforms offer free or low-cost environments for developing and running CNNs, especially useful for those without access to high-end hardware:

\begin{itemize}
  \item \textbf{Google Colab:}
  A free cloud-based platform that allows execution of Python code using TensorFlow and Keras, with optional GPU acceleration.

  \item \textbf{Kaggle Kernels:}
  Provides a Jupyter Notebook environment for running Python code with built-in access to machine learning libraries like TensorFlow and Keras.

  \item \textbf{FloydHub:}
  A cloud platform for training and deploying deep learning models via a command-line interface, supporting both CPU and GPU backends.

  \item \textbf{Microsoft Azure:}
  Offers deep learning environments that support TensorFlow and Keras, with access to GPUs for accelerated training.

  \item \textbf{Amazon SageMaker (AWS):}
  A cloud-based machine learning service that supports the training and deployment of models using popular libraries including TensorFlow and PyTorch.
\end{itemize}

Running CNN models on the cloud reduces the need for powerful local hardware and eliminates the overhead of library installation and configuration. However, it is important to consider the potential cost of cloud services, as well as issues related to data privacy and security before use.

\subsection{Desktop-based Tools}

The steps required to set up an environment for training a neural network on a personal computer are as follows.

1. A Python distribution, such as Anaconda or Miniconda, needs to be installed on the computer to run a deep learning library, such as TensorFlow and Keras, which contain many of the necessary libraries pre-installed.

2.The installation of a deep learning library, such as TensorFlow, Keras, or PyTorch, using pip or conda package managers will be necessary.

3.Other necessary libraries, such as NumPy, SciPy, and Matplotlib, may need to be installed, depending on the specific requirements of the project.

4.The installation of a GPU driver may be necessary if the GPU is present as a computer to enable the utilization of its parallel processing capabilities during the training process.

5. These data are required for training and testing the neural network. Depending on the type of dataset, pre-processing, feature extraction, and formatting informative compatible mwit hfor the deep learning library being used, such as CSV, HDF5, or TFRecords, may be necessary.

6. The model can be trained once the environment has been set up by utilizing the API of the deep learning library to define the network trajectory and by utilizing the dataset for the training of the model.

\subsection{Colab}

Google Colab (short for "Colaboratory") is a cloud-based platform that allows machine learning models to be trained and deployed by clients without requiring powerful local hardware. Access to Colab is available through a web browser, and the virtual machine can be used by clients to run their code. Signing in with a Google Account is required to use Google Colab. After logging in, the user can create a new notebook to begin working on their project.

In the notebook environment, Python code can be written and run by client, and

documentation can be created using features such as syntax highlighting, code completion, and markdown cells.

Pre-installed libraries and dependencies, such as TensorFlow, Keras, and Py-Tool, were provided by Colab, eliminating the need for clients to install them. In addition, free GPUs and TPUs are available through Google Colab to accelerate the training process, making it a convenient option for deep learning tasks without access to high-performance local hardware.

Data can be uploaded and downloaded to and from Colab by the client using the file upload and download features, or by connecting to Google Drive. Colab notebooks can be shared by clients for collaborative purposes.

In conclusion, Google Colab is a convenient and accessible platform for clients looking to develop and deploy machine learning models without requiring powerful local hardware or complex setup processes.\cite{127}

\subsubsection{The Colab's stages }
The general stages for creating a neural network in Google Co.ab are as follows:

The necessary libraries for building and training the neural network are imported.\cite{127}
The dataset that will be used to train the neural network is loaded, either by uploading it to Colab or by loading it from an external source, such as Google Drive, or an online repository.
The dataset was pre-processed to prepare it for the neural network. This may involve tasks such as splitting the data into training and validation sets, normalizing the data, or converting the data into an appropriate format.
The architecture of the neural network is defined by specifying the layers of the neural network, including the input, hidden, and output layers. The activation functions, loss function, and optimizer used during the training may also be specified.
The neural network was trained on the training dataset by specifying the number of epochs and batch size for training as well as monitoring the model's performance on the validation set.
The performance of the model on a separate test dataset was evaluated to determine how well the model generalizes to the new data.
The model was fine-tuned based on the results of the evaluation by adjusting the hyperparameters or changing the architecture.
The model is deployed to predict new data, which may involve exporting the model to a file format that can be used by other applications or integrating the model into an existing software system.

Several alternatives to Google Colab can be considered depending on the client's needs and budget. Some free options include Microsoft Azure Notebooks, Jupyter Notebooks, and Kaggle Kernels. Microsoft Azure Notebooks are cloud-based platforms that allow clients to run Jupyter Notebooks in a free-hosted environment. Jupyter Notebook is an open-source web application that allows clients to create and share documents containing live code, equations, visualizations, and narrative text. Kaggle Kernels is a cloud-based platform that provides free access to compute resources, allowing clients to write and execute code in a Jupyter Notebook environment. On the other hand, there are also paid options, such as Amazon Sage Maker, IBM Watson Studio, and Data-bricks. These platforms provide more advanced features and better

performance, but come at a cost. Amazon SageMaker is a fully-managed service that provides clients with tools to build, train, and deploy machine learning models at scale. IBM Watson Studio is a cloud-based data science platform that allows clients to build and train machine learning models using a range of tools and frameworks. Data-bricks is a unified analytics platform that provides clients with a collaborative workspace for data engineering, machine learning, and analytics.
Analytical,
\textbf{Amazon SageMaker:} Similar to Colab, SageMaker is a cloud-based platform for building, training, and deploying machine-learning models. It provides a range of tools and services for data preparation, training, and hosting and supports popular machine learning frameworks such as TensorFlow, PyTorch, and MXNet.

IBM Watson Studio: Watson Studio is another cloud-based platform for building and training machine-learning models. It provides tools for data preparation, visualization, and model building and supports popular machine learning libraries such as scikit-learn, TensorFlow, and Keras.

Microsoft Azure Machine Learning: Azure Machine Learning is a cloud-based platform for building and deploying machine-learning models. It provides tools for data preparation, model building, and deployment and supports popular machine learning frameworks such as TensorFlow, PyTorch, and scikit-learn.

\textbf{Databricks}: Databricks is a cloud-based platform for building and deploying data-intensive applications, including machine learning models. It provides tools for data preparation, model building, and deployment and supports popular machine learning libraries such as TensorFlow, PyTorch, and scikit-learn.

These platforms offer similar functionalities to Google Colab and can be used to build and train machine-learning models in a cloud-based environment.

Google Colab is a free platform for creating and running machine learning models in a cloud-based environment. However, there are limitations to the resources available, such as computing power and memory, and usage is subject to fair usage policies.

However, some cloud-based machine learning platforms, such as Amazon SageMaker and Microsoft Azure Machine Learning, offer both free and paid plans. These paid plans typically provide access to more resources and advanced features as well as additional support and service-level agreements. The costs of these paid plans can vary depending on the level of usage and the specific features included.

\section{PhysioNet.org }

The PhysioNet.org Resource is a widely used resource for biomedical signal processing research. It provides a variety of databases and tools for analyzing and interpreting biomedical signals, such as electrocardiograms (ECGs), electroencephalograms (EEGs), and other physiological signals.
One of the main features of PhysioNet.org is the large collection of

physiological signals that it provides, including ECG, EEG, and other signals, collected from healthy individuals and patients with various cardiovascular and neurological conditions. These signals are available to researchers and clinicians for the development of new diagnostic and therapeutic tools as well as for educational purposes.
PhysioNet.org also provides several software tools and resources for analyzing and interpreting these signals. For example, the WFDB Software Package is a widely used open-source tool for manipulating and analyzing physiological signals. This package contains a set of command-line tools for reading and writing physiological signals in the WFDB format as well as a library of C-language functions for working with these signals.
Another important resource provided by PhysioNet.org is PhysioBank ATM, a web-based tool for exploring and analyzing large collections of physiological signals. This tool allows users to search and browse the PhysioBank databases, as well as to perform basic signal processing tasks such as filtering, resampling, and averaging.
Additionally, PhysioNet.org organizes the annual Computing in Cardiology Challenge, which aims to foster innovation in the field of biomedical signal processing by providing a common platform for researchers to share data, algorithms, and results, and to evaluate the performance of different methods against a common set of benchmark data.
Overall, PhysioNet.org is a valuable resource for researchers, clinicians, and educators working in the field of biomedical signal processing, and plays a vital role in advancing the development of new diagnostic and therapeutic tools for cardiovascular and neurological conditions. The data and tools provided by PhysioNet.org are widely used in research, education and clinical practice in multiple fields such as cardiology, neurology, and critical care.\cite{128}

\section{Integrating PhysioNet.org Data into a Neural Network}

Several methods exist for incorporating data from PhysioNet.org into a neural network, depending on the particular dataset and the type of network utilized. The following general steps can be used to insert the data from PhysioNet.org into a neural network:

1.The dataset from PhysioNet.org must first be downloaded from the PhysioNet.org website. The dataset is typically made available in formats such as the WFDB or CSV.

2.The data must be preprocessed to prepare it for input into the neural network. This pre-processing may include normalizing the data, removing outliers, and dividing the dataset into training, validation, and testing sets.

3.The data may need to be converted into a specific format depending on the type of neural network being utilized. For instance, if a convolutional neural network is used for image classification, the data must be converted into an image format.

4.The data can be input into the neural network once it is in the appropriate format. The input of the data can involve the use of a library, such as TensorFlow or Keras, to define the network architecture and train the network on the data, depending on the type of neural network utilized.

5.Finally, the performance of the network on the test data was evaluated and adjustments were made as necessary.

6.It should be noted that based on the dataset, the pre-processing step can be complicated and requires additional steps such as feature extraction, segmentation, and labeling of the data. In addition, the input format of the data may vary based on the neural network architecture.
It is important to mention that the preprocessed data, annotations, and tools for data analysis are already provided by the PhysioNet.org resource; therefore, utilizing these tools instead of pre-processing the data can be considered.

\section{Preparing Data for A Machine Learning-Based Heart Disease Detection Scheme for Predicting Spontaneous Termination of
Atrial Fibrillation}
This study presents the development of an advanced autonomous system named A Machine Learning-Based Heart Disease Detection Scheme for Predicting Spontaneous Termination of Atrial Fibrillation, designed to automatically retrieve electrocardiogram (ECG) data from PhysioNet and to execute a complete digital signal processing workflow.
The study, focuses on enhancing early detection and intervention capabilities for atrial fibrillation (AF) events.

That system processes ECG signals through a sequential pipeline consisting of filtering, segmentation, feature extraction, and classification. The primary objective is to predict whether an AF episode will terminate spontaneously.

The proposed ECG analysis framework integrates a deep neural network (DNN) classifier with feature extraction based on Hjorth parameters. The detection process operates in two main stages:

Stage 1: ECG signals are preprocessed and transformed into descriptive features through the calculation of Hjorth parameters.

Stage 2: These extracted features are input into the DNN classifier, which outputs a prediction regarding the spontaneous termination of AF.

Validation of the proposed detection scheme was conducted using clinical data obtained from the Shandong Provincial Hospital (SPHD) database. The system’s performance was assessed by analyzing confusion matrices and receiver operating characteristic (ROC) curves, providing insight into its classification accuracy and diagnostic effectiveness.

Experimental results demonstrate that A Machine Learning-Based Heart Disease Detection Scheme for Predicting Spontaneous Termination of Atrial Fibrillation achieves high prediction accuracy with low computational complexity, confirming its potential for integration into portable consumer healthcare devices.

\subsection{ Automated AF Termination: Insights from 2004 Cardiology Challenge}

The exploration of this project, the primary phase involved meticulous data acquisition. Data sourcing was performed using PhysioNet.org. and the AF

Termination Challenge database, a specialized resource tailored for the 2004 Computers in Cardiology Challenge. This challenge, directed at enhancing the automated prediction of spontaneous termination of atrial fibrillation (AF), underscores the significance of the two-channel recording database.

Delving into the dataset, wastes were distinctly categorized into a training set, characterized by records labeled n*, s*, and t*, and two test sets encompassing records denoted as a* and b*.

The dataset encapsulates one-minute segments of atrial fibrillation, each featuring two ECG signals sampled at samples/second. Originating from extensive 20-24 hour ECG recordings, these segments serve as fundamental components for the Computers in Cardiology Challenge 20 Integral signal to the dataset are automated QRS annotations, attributing normalcy to all detected beats, including ectopic occurrences, albeit with the caveat that these annotations remain unchecked and may harbor minor errors.

Within the learning set, 30 participants were meticulously distributed among the groups.

The atrial fibrillation (AF) termination prediction dataset is categorized into three primary groups based on the progression and resolution of AF episodes within long-term electrocardiogram (ECG) recordings:

Group N (n01–n10): Characterized by sustained, non-terminating AF persisting throughout the duration of the recording.

Group S (s01–s10): Includes episodes where AF terminates exactly one minute after the conclusion of the ECG recording.

Group T (t01–t10): Comprises cases where AF terminates immediately upon recording cessation.

Notably, Group T samples are sequentially linked to those in Group S, as they originate from the same prolonged ECG recordings (e.g., t01 follows s01). The training dataset was composed of signals from 20 subjects, evenly distributed between Group N and Groups S/T.

The testing subset, referred to as Set A, consists of 30 ECG segments (a01–a30) obtained from individuals not included in the training or secondary test sets. Approximately half of the samples exhibit sustained AF (Group N), while the rest are associated with imminent AF termination (Group T). The primary classification objective is to distinguish which of the a01–a30 records belong to Group T, thereby enabling prediction of spontaneous AF termination~\cite{moody2004predicting}.

Test set B encompasses 20 records (b01 to b20), featuring pairs from 10 subjects, not part of the learning set or test set A. Each pair integrates one record from Group S and another from Group T, with brief intervals between some pairs. The sub-challenge entails identifying records affiliated with Group T.

\subsection{Data Organization and Preparation for MATLAB Analysis}
During the data pre processing stage, raw signal data were structured into a format compatible with MATLAB's computational environment. Specifically, six training matrices were constructed, with dimensions of either 10 × 1280 and 20×1280, depending on the dataset subset. In these matrices, each row represents a single ECG record, while each column corresponds to one of the 1280 uniformly sampled time points. This standardized representation facilitates matrix-based operations and enables consistent input for subsequent signal analysis and classification algorithms.

The following base code is given to create hjorth parameters

\begin{lstlisting}[language=Python]

```
S = transpose(ECG1B);
%converting each row into a table column 1280 samples
for each patient (10)

[N,K] = size(S);
% number of electrodes K, number of samples N

%m0 = mean(sumsq(S,2));
d0 = S;
%m1 = mean(sumsq(diff(S,[],1),2));
d1 = diff([zeros(1,K);S ],[],1);
d2 = diff([zeros(1,K);d1],[],1);

FLAG_ReplaceNaN = 0;

if addme<2,
 UC = 0;
end;
if addme<3;
 if UC==0,

 elseif UC>=1,
  B = ones(1,UC);
  A = UC;
 elseif UC<1,
  FLAG_ReplaceNaN = 1;
  B = UC;
  A = [1, UC-1];
 end;
else
 B = UC;
end;

if ~UC,
 m0 = mean(d0.^2);
 m1 = mean(d1.^2);
 m2 = mean(d2.^2);
else
 if FLAG_ReplaceNaN;
```

```
  d0(isnan(d0)) = 0;
  d1(isnan(d1)) = 0;
  d2(isnan(d2)) = 0;
 end;
 m0 = filter)(B,A, and d0.^2)./filter(B,A,double(~isnan(d0)));
 m1 = filter(B,A,d1.^2)./filter(B,A,double(~isnan(d1)));
 m2 = filter(B,A,d2).^2)./filter(B,A,double(~isnan(d2)));
end;

ACTIVITY   = m0;

%create a table 1*10 with name activity
MOBILITY   = sqrt(m1. /m0);

%create a table 1*10 with name Mobility
COMPLEXITY = sqrt(m2./m1)./MOBILITY;

%create a table 1*10 with name Complexity

\end{lstlisting}
```

\subsection*{Training and Validation of the Neural Network Using Hjorth Parameters}

From the PhysioNet central repository, a total of 2400 ECG-derived entries were retrieved and subsequently divided into two datasets: Table A with dimensions 1920×4, and Table B with dimensions
480×4. Table A served as the primary training set for the neural network, while Table B was used exclusively for post-training validation by comparing known outcomes with model predictions.

Table A was further subdivided, with 1536 records allocated for training and 384 for internal validation during model development. Each record in Table A comprises three Hjorth parameters—Activity, Mobility, and Complexity—used as features, and a fourth column labeled target, which designates the binary classification label: 0 for non-atrial fibrillation (Non-AF) and 1 for atrial fibrillation (AF).

After conducting 600 training iterations, the model achieved a final training accuracy of 95.64\%, with a validation accuracy of 93.75\%. When tested against Table B (480 independent samples), the model correctly classified 455 entries, yielding an overall classification accuracy of 94.7\%.

This approach underscores the effectiveness of using Hjorth parameters as discriminative features for ECG signal classification in atrial fibrillation detection tasks. The dataset and further methodological details are available at the following link: [www.drhack.gr].

\subsection{Code Explanation}

This section provides an in-depth explanation of the Python code used for training and evaluating a neural network model with TensorFlow and Keras. The code is structured into several components, which are explained below.

\subsection{Importing Necessary Libraries}

The following libraries are imported at the beginning of the code:
\begin{itemize}
    \item \texttt{os}: Used for interacting with the operating system's file system.
    \item \texttt{tensorflow}: The main library used for building and training machine learning models.
    \item \texttt{pandas}: A library for data manipulation and analysis, particularly for working with data in tabular form.
    \item \texttt{keras}: A high-level neural networks API, running on top of TensorFlow, used for creating and training models.
    \item \texttt{TensorBoard}: A callback used for visualizing the training process and model metrics during the training.
\end{itemize}

```
\begin{verbatim}
import os
import tensorflow as tf
import pandas as pd
import keras
from keras import layers
from keras.callbacks import TensorBoard  # Import TensorBoard callback
\end{verbatim}
```

\subsection{Setting Up TensorBoard Logs Directory}

The following code sets the directory where TensorBoard will store the training logs:

```
\begin{verbatim}
log_dir = "./logs"
tensorboard_callback =
TensorBoard(log_dir=log_dir, histogram_freq=1, profile_batch=0)
\end{verbatim}
```

This defines the \texttt{log_dir} directory, which will hold the logs from the training process. The \texttt{TensorBoard} callback is used to record these logs for visualization.

\subsection{Loading and Preparing the Data}

The code loads the dataset from a CSV file hosted on a remote server. The data is then stored in a Pandas DataFrame, which makes it easy to inspect and manipulate.

```
\begin{verbatim}
file_url =
"https://drhack.gr/wp-content/uploads/2024/10/Learning_SET_1920x4.csv"
dataframe = pd.read_csv(file_url)
```

```
dataframe.shape
dataframe.head()
\end{verbatim}

The \texttt{shape} method returns the dimensions of the dataset, while
\texttt{head()} displays the first few rows of the data for inspection.

\subsection{Splitting the Data into Training and Validation Sets}

The data is split into two sets: one for training the model and the other for
validating it. The validation set represents 20\% of the total data, and the
training set consists of the remaining 80\%.

\begin{verbatim}
val_dataframe = dataframe.sample(frac=0.2, random_state=1337)
train_dataframe = dataframe.drop(val_dataframe.index)

print(
    f"Using {len(train_dataframe)} samples for training "
    f"and {len(val_dataframe)} for validation"
)
\end{verbatim}

This split ensures that the model is evaluated on unseen data, improving its
generalization ability.

\subsection{Converting DataFrames to TensorFlow Datasets}

The function \texttt{dataframe\_to\_dataset} converts the Pandas DataFrame into
a TensorFlow dataset, which can be efficiently used for training a neural
network model.

\begin{verbatim}
def dataframe_to_dataset(dataframe):
    dataframe = dataframe.copy()
    labels = dataframe.pop("target")
    ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
    ds = ds.shuffle(buffer_size=len(dataframe))
    return ds

train_ds = dataframe_to_dataset(train_dataframe)
val_ds = dataframe_to_dataset(val_dataframe)
\end{verbatim}

\texttt{train\_ds} and \texttt{val\_ds} are the training and validation
datasets, respectively. The data is shuffled to ensure randomness during
training.

\subsection{Feature Normalization}

The code normalizes the numerical features using the Keras
\texttt{Normalization} layer, which scales the data to have a mean of zero and a
standard deviation of one.
```

```
\begin{verbatim}
def encode_numerical_feature(feature, name, dataset):
    normalizer = layers.Normalization()
    feature_ds = dataset.map(lambda x, y: x[name])
    feature_ds = feature_ds.map(lambda x: tf.expand_dims(x, -1))
    normalizer.adapt(feature_ds)
    encoded_feature = normalizer(feature)
    return encoded_feature
\end{verbatim}
```

Normalization helps improve the training process by ensuring that all features are on a similar scale, making it easier for the model to converge.

```
\subsection{Building the Neural Network Model}
```

The neural network model is defined using the Keras API. It consists of multiple layers, including dense layers with ReLU activation functions and a dropout layer for regularization.

```
\begin{verbatim}
HjorthBPM1 = keras.Input(shape=(1,), name="HjorthBPM1")
HjorthBPM2 = keras.Input(shape=(1,), name="HjorthBPM2")
HjorthBPM3 = keras.Input(shape=(1,), name="HjorthBPM3")

all_inputs = [HjorthBPM1, HjorthBPM2, HjorthBPM3]

HjorthBPM1_encoded =
encode_numerical_feature(HjorthBPM1, "HjorthBPM1", train_ds)
HjorthBPM2_encoded =
encode_numerical_feature(HjorthBPM2, "HjorthBPM2", train_ds)
HjorthBPM3_encoded =
encode_numerical_feature(HjorthBPM3, "HjorthBPM3", train_ds)

all_features =
layers.concatenate([HjorthBPM1_encoded,
HjorthBPM2_encoded, HjorthBPM3_encoded])

x = layers.Dense(32, activation="relu")(all_features)
x = layers.Dense(32, activation="relu")(x)
x = layers.Dropout(0.5)(x)
output = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(all_inputs, output)
model.compile("adam", "binary_crossentropy", metrics=["accuracy"])
\end{verbatim}
```

Here, the model is designed to take three input features (HjorthBPM1, HjorthBPM2, and HjorthBPM3). It passes through several dense layers, with ReLU activation functions and a dropout layer for regularization. The output layer uses the sigmoid activation function, suitable for binary classification tasks.

```
\subsection{Training the Model}
```

The model is trained using the \texttt{fit} method, with the training data,

validation data, and TensorBoard callback to visualize the training process.

```
\begin{verbatim}
model.fit(train_ds, epochs=600,
validation_data=val_ds, callbacks=[tensorboard_callback])
\end{verbatim}
```

This code trains the model for 600 epochs, with the validation data used to assess the model's performance at each epoch.

Making Predictions

After training, the model is used to make predictions on new data. The predictions are then stored and processed for further evaluation.

```
\begin{verbatim}
predictions = model.predict(input_dict)
predictions_list.append(100 * predictions[0][0])
\end{verbatim}
```

For each input sample, the model outputs a prediction, which is then converted into a percentage and stored in the list \texttt{predictions_list}.

Rounding the Predictions

To make the predictions more interpretable, they are rounded to the nearest integer (either 0 or 1), which corresponds to the two possible classes for binary classification.

```
\begin{verbatim}
rounded_array = np.round(predictions_array)
\end{verbatim}
```

This step is useful for presenting the final results as binary outcomes.

Visualizing with TensorBoard

Finally, TensorBoard is used to visualize the training process. The following commands load and start TensorBoard:

```
\begin{verbatim}
%load_ext tensorboard
%tensorboard --logdir=./logs
\end{verbatim}
```

This command opens the TensorBoard interface, where various metrics such as loss and accuracy are displayed during training.

Conclusion

This code outlines the process of building, training, and evaluating a neural network for binary classification. The use of TensorFlow and Keras allows for the creation of a flexible and efficient model. The incorporation of TensorBoard provides valuable insights into the training process, helping to identify

potential issues and track the model's progress over time.

\section{Repairing Data for Fetus heart rate}

\section{Revolutionizing IoHT Based Monitoring: Low-Cost Fetal Cardiac Sensing via ESP8266 Node}

Over the past decade, the integration of Internet of Health Things (IoHT) technologies has profoundly reshaped healthcare delivery, particularly in the domains of real-time patient monitoring and telemedicine. Among the most impactful developments is the creation of smart body sensor network systems, which facilitate noninvasive health tracking, cloud-based data analytics, and patient-centric care.

In our recent research\cite{70}, we designed and implemented a low-cost wearable fetal heart rate (FHR) monitoring system based on an ESP8266 microcontroller. The proposed platform, developed by our team, combines photoplethysmography (PPG), which is commonly found in pulse oximeters, with IoT-based wireless communication modules. The system enables real-time, continuous monitoring of both maternal and fetal cardiac parameters, and securely transmits data over IEEE 802.11 b/g/n wireless networks to a remote server for analysis and visualization.

With a total hardware cost of approximately \$10, the device offers a scalable solution suitable for both clinical and rural applications. It features integrated sensors for heart rate, temperature, and humidity, making it ideal for use during pregnancy, where fetal well-being must be monitored around the clock. A key advantage of our implementation is the minimal error rate, which is less than 1\% when benchmarked against standard hospital-grade cardiotocograph systems.

Our system was rigorously tested in collaboration with medical professionals and validated against a clinical gold standard. The results were disseminated through the publication titled Home Healthcare Technologies and Services: Heart Rate Monitoring System Using an MCU ESP8266 Node, presented at the 2022 Panhellenic Conference on Electronics and Telecommunications (PACET)\cite{70}.

Importantly, the platform supports remote access to patient data through a cloud dashboard, enabling healthcare professionals to receive alerts and trends in real-time. This feature is particularly beneficial in situations that require physical distancing, such as during the COVID-19 pandemic, where home-based monitoring is essential.

Furthermore, optimization of the firmware and the use of energy-efficient components contributed to a significant reduction in power consumption. These advances strengthen the practicality and sustainability of the solution, laying the groundwork for its broader adoption in future smart healthcare ecosystems\cite{67,68,69,70}.

Having established the hardware and sensing methodology, we now evaluate the system's performance using statistical tools such as the ROC curve.

\subsection{Smart Health Monitoring: Affordable and Adaptive Arduino UNO Platform}

The implementation of a smart health monitoring system is based on a cost effective modular development platform. Central to this platform is the widely accessible Arduino UNO Rev3 microcontroller, which is selected because of its affordability, versatility, and open source nature\cite{55}.

Arduino UNO Rev3 features 14 digital I/O pins, USB connectivity, and compatibility with a wide variety of shields, rendering it suitable for both novice users and advanced developers. In our design, the microcontroller was integrated with a photoplethysmography (PPG) sensor, cloud enabled communication shield, and local LCD display interface. These components collectively allow real time data acquisition, cloud transmission, and user friendly visualization.

The LCD includes a brightness adjustment feature via a potentiometer, whereas a photo diode is employed as a visual alert mechanism for pulse detection, which is beneficial for users with visual or auditory limitations. Furthermore, the platform supports a detachable and replaceable power source, ensuring continuous operation during outages and prolonged monitoring sessions.

This configuration not only achieves low power consumption and reduced operational costs, but also offers scalability and portability for personal and clinical use.

\subsection{Challenges of Building Neural Networks with Arduino uno 3 development board}

Efforts to utilize the Arduino Uno 3 Development Board faced significant challenges due to its limited computational power. While the Arduino UNO 3 and ESP 8266 are useful for applications with moderate computational needs, they struggle with processing large data sets, such as ECG signals. This limitation hinders the construction of neural networks, underscoring the critical role of computational power in tackling modern computing challenges.

\end{itemize}

\printbibliography
\end{document}