

People's Democratic Republic of Algeria  
Ministry of Higher Education and Scientific Research  
Mohamed El Bachir El Ibrahim University of Bordj Bou Arréridj  
Faculty of Mathematics and Informatics  
Informatics Department



## DISSERTATION

Presented in fulfillment of the requirements of obtaining the degree  
**Master in Informatics**  
Specialty: Business Intelligence Engineering

## THEME

# Deep Learning: Application to Pervasive Computing System

*Presented by:*

DJITLI Amine

BELBAGRA Ahmed Chaouki

*Publicly defended on: 13/09/2023*

*In front of the jury composed of:*

**President: FARES Nour EL Houda**

**Examiner: MAAZA Sofiane**

**Supervisor: MAACHE Salah**

**2022/2023**

# Dedication

We dedicate this humble memoire to:

**Our dear parents:** That no dedication can express what we owe them, for all their sacrifices, benevolence, love, tenderness, support and prayers throughout our studies. May this work be a testimony of our deep love and our great gratitude “May God keep you”.

**Our dear sisters and our dear brothers:** for their permanent encouragement, and their moral support, we dedicate this modest work to them in testimony of our great love and our infinite gratitude.

**All our Families:** to all the DJITLI and BELBAGRA families, for their support throughout my university career, may this work be the fulfillment of your so-called wishes, and the fruit of your unfailing support.

**All our friends:** For their help and their moral support during the elaboration of the dissertation.

# Acknowledgment

We first thank the good god who gave us the strength and courage to complete this work.

The work presented in this end-of-studies project was carried out as part of the preparation for the master's degree in computer science engineering, specialty "engineering of Business intelligence" at the University of Mohamed El Bachir El Ibrahimi of Bordj Bou Arréridj.

We would like to express our most sincere thanks to the people who have helped us and who have contributed to the elaboration of this dissertation as well as to the success of this wonderful academic year.

The first person we would like to thank is our supervisor Mr. MAACHE SALAH, for the orientation, the trust and the patience which constituted a considerable contribution without which this work could not have been carried out. May he find in this work a living tribute to his lofty personality. Secondly the judges, we would like to thank them for the orientation and recommendations and helpful guidance in our dissertation report.

# Abstract

In today's digital era, the ubiquity of sensors and interconnected devices has ushered in the age of pervasive computing. Pervasive computing envisions a future where computation integrates into our daily lives, creating intelligent and adaptive environments. Central to this vision is the transformative role of deep learning, a subset of machine learning powered by artificial neural networks. This study delves into the profound synergy between deep learning and pervasive computing systems. It explores how deep learning techniques are harnessed to enable interconnected devices and sensors to learn, adapt, and enhance user experiences. The research covers fundamental concepts, applications, and implications, shedding light on the dynamic evolution of pervasive computing. With a focus on efficiency, privacy, and ethical considerations, this work exemplifies the potential of deep learning to revolutionize industries and improve the quality of our technologically enriched lives.

Keywords: Pervasive Computing System, Ubiquitous Computing System, Human Activity Recognition (HAR), Smartphone Sensors, Deep Learning (DL), Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM)

# Résumé

À l'ère numérique d'aujourd'hui, l'omniprésence des capteurs et des appareils interconnectés a inauguré l'ère de l'informatique omniprésente. L'informatique omniprésente envisage un avenir où le calcul s'intègre dans notre vie quotidienne, créant des environnements intelligents et adaptatifs. Au cœur de cette vision se trouve le rôle transformateur de l'apprentissage profond, un sous-ensemble de l'apprentissage automatique alimenté par des réseaux de neurones artificiels. Cette étude explore la synergie profonde entre l'apprentissage profond et les systèmes informatiques omniprésents. Il explore comment les techniques d'apprentissage profond sont exploitées pour permettre aux appareils et capteurs interconnectés d'apprendre, de s'adapter et d'améliorer les expériences utilisateur. La recherche couvre les concepts fondamentaux, les applications et les implications, mettant en lumière l'évolution dynamique de l'informatique omniprésente. En mettant l'accent sur l'efficacité, la confidentialité et les considérations éthiques, ce travail illustre le potentiel de l'apprentissage profond pour révolutionner les industries et améliorer la qualité de nos vies technologiquement enrichies.

Mots-clés : système informatique omniprésent, reconnaissance de l'activité humaine (HAR), capteurs de smartphone, apprentissage en profondeur (DL), réseau de neurones convolutifs (CNN), mémoire à long court terme (LSTM)

## ملخص

في العصر الرقمي اليوم ، بشر انتشار أجهزة الاستشعار والأجهزة المترابطة في كل مكان بعصر الحوسبة المنتشرة. تتصور الحوسبة المنتشرة مستقبلا يتكامل فيه الحساب في حياتنا اليومية ، مما يخلق بيانات ذكية وقابلة للتكيف. محور هذه الرؤية هو الدور التحويلي للتعلم العميق ، وهو مجموعة فرعية من التعلم الآلي المدعوم بالشبكات العصبية الاصطناعية. تتعمق هذه الدراسة في التآزر العميق بين التعلم العميق وأنظمة الحوسبة المنتشرة. يستكشف كيفية تسخير تقنيات التعلم العميق لتمكين الأجهزة وأجهزة الاستشعار المترابطة من التعلم والتكيف وتعزيز تجارب المستخدم. يغطي البحث المفاهيم والتطبيقات والآثار الأساسية ، ويلقي الضوء على التطور الديناميكي للحوسبة المنتشرة. مع التركيز على الكفاءة والخصوصية والاعتبارات الأخلاقية ، يجسد هذا العمل إمكانات التعلم العميق لإحداث ثورة في الصناعات وتحسين نوعية حياتنا الغنية تقنيا.

الكلمات الرئيسية: نظام الحوسبة الشامل ، نظام الحوسبة في كل مكان ، التعرف على النشاط البشري (HAR) ، مستشعرات الهواتف الذكية ، التعلم العميق (DL) ، الشبكة العصبية التلافيفية (CNN) ، الذاكرة طويلة المدى (LSTM)

# Table of contents

<b>Abbreviation list.....</b>	<b>x</b>
<b>List of figures.....</b>	<b>xi</b>
<b>General Introduction .....</b>	<b>xiv</b>
1.1. Context & Problematic.....	xiv
1.2. Objectives.....	xv
1.3. Methodology and results .....	xv
1.4. Report outlines .....	xvii
<b>Chapter 01: Deep Learning: Application to Pervasive computing System .....</b>	<b>1</b>
1.1. Introduction.....	1
1.2. Definition of the concept deep learning.....	1
1.3. Importance of deep learning.....	2
1.4. Pervasive computing systems .....	2
1.5. Deep Learning and Pervasive Computing Systems .....	3
1.6. Deep Learning methods .....	4
1.6.1. Supervised learning .....	4
1.6.2. Unsupervised learning.....	5
1.6.3. Semi-supervised learning.....	5
1.7. Deep Learning Algorithms for Pervasive Computing Systems.....	5
1.7.1. Fully Connected Neural Network (FCNN).....	5
1.7.2. Recurrent Neural Network (RNN).....	6
1.7.3. Convolutional Neural Network (CNN) .....	6
1.7.4. Long Short-Term Memory (LSTM).....	6
1.8. the different hidden layers of a CNN and LSTM network.....	7
1.8.1. Convolutional Layer .....	7
1.8.2. Pooling layer.....	8
1.8.3. SoftMax.....	8
1.8.4. Flatten.....	9

1.8.5. Fully connected layer.....	9
1.8.6. Wearable sensors.....	10
1.8.7. Gyroscope.....	12
1.8.8. Accelerometer .....	12
1.8.9. Magnetometer .....	14
1.9. Conclusion .....	15
<b>Chapter 02: State of the art.....</b>	<b>16</b>
2.1. Introduction.....	16
2.2. Literature review.....	16
2.3. Applications of Deep Learning in Pervasive Computing Systems.....	17
2.3.1. Smart Homes.....	18
2.3.2. Smart Cities .....	19
2.3.3. Smart Healthcare .....	20
2.3.3. Smart Transportation .....	21
2.4. Conclusion .....	21
<b>Chapter 03: Methodology and Architecture .....</b>	<b>22</b>
3.1. Introduction.....	22
3.2. Architecture Selection .....	22
3.6. Data Collection and Preprocessing.....	23
3.7. Model Training .....	24
3.8. Approach Validation.....	25
3.8.1. Confusion Matrix.....	26
3.8.2. Classification rate.....	26
3.8.3. Performance Metrics .....	27
3.9. Ethical Considerations.....	28
3.10. Software and Hardware Frameworks .....	28
3.10.1 Machines .....	28
3.10.2 Spyder IDE.....	29
3.11. Conclusion .....	31
<b>Chapter 04: Implementation and Results.....</b>	<b>32</b>
4.1. Introduction.....	32



4.2. Model Implementation.....	32
4.2.1. Data Preprocessing .....	32
4.2.2. CNN and LSTM Architecture Instantiation.....	33
4.3 Model results and discussion .....	33
4.3.1. CNN Model Performance .....	33
4.3.2. LSTM Model Performance.....	35
4.4. Comparison and Insights.....	36
4.4.1. CNN vs. LSTM.....	36
4.4.2. Accuracy comparison baseline vs LSTM.....	37
4.4.3. Resource-Intensive Models.....	38
4.5. Graphical User Interface implementation (GUI) .....	40
4.5.1. Graphical user interface results.....	41
4.6. Conclusion.....	43
<b>Chapter 05: General Conclusion .....</b>	<b>44</b>
5.1. Contributions .....	44
5.2. Limitations .....	44
5.3. Future work and perspectives.....	45
<b>References .....</b>	<b>46</b>
<b>Appendix A Figure Sources .....</b>	<b>51</b>
A.1. PAGES.....	51

# Abbreviation list

DL	Deep Learning
ML	Machine Learning
HAR	Human Activity Recognition
CNN	Convolutional Neural Network
LSTM	Long-Short Term Memory
IoT	Internet of Things
FCNN	Fully Connected Neural Network
RNN	Recurrent Neural Network
GPS	Global Positioning System
MEMS	Micro-Electromechanical Systems
IDE	Integrated Development Environment
GUI	Graphical User Interface

# List of figures

Figure 1 Functioning of pervasive computing .....	3
Figure 2 Main Methods of DL .....	4
Figure 3 the architecture of FCNN .....	5
Figure 4 RNN Architecture.....	6
Figure 5 CNN Architecture.....	6
Figure 6 LSTM Architecture .....	7
Figure 7 One-dimensional cross-correlation operation.....	8
Figure 8 Pooling layer.....	8
Figure 9 SoftMax Layer.....	9
Figure 10 Flatten Layer.....	9
Figure 11 Fully connected layer .....	10
Figure 12 Wearable sensors position in human body .....	11
Figure 13 (a) Gyroscope (b) Gyroscope in smartphone.....	12
Figure 14 Accelerometer.....	13
Figure 15 magnetometer .....	14
Figure 16 Pervasive computing architecture.....	18
Figure 17 Smart home features .....	19
Figure 18 Data set body positions.....	22
Figure 19 Activity classes Standing, Sitting, Laying, Walking, Upstairs, Downstairs .....	22

Figure 20 CNN layers .....	24
Figure 21 LSTM layers .....	25
Figure 22 Dataset split .....	26
Figure 23 Confusion table of a two-class problem. ....	<b>Error! Bookmark not defined.</b>
Figure 24 Equation of classification rate .....	27
Figure 25 Metric and Formula .....	27
Figure 28 Spyder IDE .....	29
Figure 29 Spyder Console.....	29
Figure 30 Spyder Libraries .....	30
Figure 31 Visualization Terminal .....	30
Figure 32 Flowchart of the sensors pre-processing data.....	32
Figure 33 LSTM model architecture.....	33
Figure 34 CNN model architecture.....	33
Figure 35 CNN confusion matrix .....	34
Figure 36 CNN Model Loss.....	34
Figure 37 CNN Model Accuracy .....	35
Figure 39 LSTM Model Loss .....	36
Figure 41 baseline vs LSTM model.....	37
Figure 42 Resource usage of DL models .....	38
Figure 43 HAR Graphical user interface .....	41
Figure 44 Browse xlsx file .....	41

Figure 45 File training .....	42
Figure 46 File classification.....	42
Figure 47 Displayed results .....	43

# General Introduction

Pervasive computing, also known as ubiquitous computing, is a concept where technology seamlessly integrates into our daily lives. It involves interconnected devices that are embedded in objects, environments, and even our bodies. These devices gather and exchange data to provide services and enhance efficiency. The goal is to create an environment where technology adapts to human needs, blending into the background and supporting our activities. Pervasive computing has applications in various domains, including healthcare, transportation, entertainment, and education. However, privacy, security, and ethical concerns need to be addressed. Ultimately, pervasive computing envisions a world where technology is ubiquitous, making our environments "smart" and responsive to our needs.

## 1.1. Context & Problematic

In the contemporary digital landscape, we find ourselves immersed in a world where technology permeates every aspect of our lives. This phenomenon is at the heart of pervasive computing, a paradigm shift that envisions a future where computation and data processing are seamlessly integrated into our daily routines. Pervasive computing, strives to create intelligent environments where technology becomes a natural and unobtrusive part of our surroundings. It is in this context that we embark on a journey to explore the profound connection between deep learning and pervasive computing systems.

Pervasive computing extends its reach across various domains, including the Internet of Things (IoT), Smart Homes, Smart Cities, and beyond. IoT devices, sensors, and interconnected systems form the fabric of these intelligent ecosystems, collecting data, processing information, and facilitating automation. The potential applications are boundless, from enhancing healthcare and optimizing transportation to improving energy efficiency and making cities more livable. However, the proliferation of pervasive computing systems also raises significant concerns regarding privacy, security, and ethical implications.

Implementing deep learning in pervasive computing systems poses several challenges. One of the main difficulties is the limited computational resources available on these systems. Pervasive computing devices are often small and have low processing power, which can make it difficult to run complex deep learning algorithms. Another challenge is the need for real-time processing, as many pervasive computing applications require immediate responses. This can be difficult to achieve with deep learning algorithms that often require significant computation time.

Another challenge is the lack of labeled data. Deep learning algorithms require large amounts of labeled data to train effectively, but obtaining this data can be difficult for pervasive computing systems. Additionally, the data collected by these systems may be noisy or incomplete, which can further complicate the training process. Finally, there are ethical considerations to take into account when implementing deep learning in pervasive computing systems, such as privacy concerns and potential biases in the algorithms, in our work we concentrated on the sensors part of the pervasive computing system and the issues related to it from privacy and on the algorithmic biases to the ethical considerations.

## **1.2. Objectives**

This dissertation's core objective is to delve into the realm of deep learning and its profound impact on pervasive computing systems. With a specific focus on Human Activity Recognition (HAR), we aim to harness the power of deep learning to enhance the accuracy and ethical considerations of HAR in these pervasive systems.

## **1.3. Methodology and results**

Methodology:

1. Data Collection: we collected data from various sensors, such as accelerometer, gyroscope and magnetometer. The data collection process was done by Shoaib, Muhammad, "Human Activity Recognition Using Heterogeneous Sensors" "Appears in the Adjunct Proceedings of UbiComp 2013.

2. Feature Extraction: Researchers (Shoab, Muhammad) applied signal processing techniques to extract relevant features from the collected sensor data.
3. Algorithm Development: The algorithms that we used include deep learning models which are convolutional neural networks (CNN), Long short-term memory network (LSTM).
4. Training and Evaluation: The developed algorithms are trained using labeled datasets where human activities are manually annotated. We used train-test split dataset for evaluation. Finally, we used performance metrics such as accuracy and confusion matrix, precision to assess the algorithm's effectiveness.

#### Results:

1. Activity Recognition Accuracy: The primary result is the accuracy achieved by the HAR system in recognizing human activities. This metric represents the system's ability to correctly identify different activities based on the sensor data. Higher accuracy indicates better performance.
2. Confusion Matrix: Illustrates the system's performance in classifying activities. It shows the number of correct and incorrect predictions for each activity class, giving insights into specific recognition errors or misclassifications.
3. Comparative Analysis: We compare the proposed HAR algorithms with existing methods or benchmarks. This analysis demonstrates the strengths and limitations of the proposed approach, highlighting its performance improvements or novel features.
4. Graphical User Interface Performance: The graphical user interface performance application of the HAR system is also an important aspect. It measures the system's ability to detect and recognize activities in an accurate manner.



## **1.4. Report outlines**

This report is structured into several chapters, each serving a specific purpose and contributing to the overall understanding of the project. The outline of the report is as follows:

### **Chapter 01: Deep Learning: Application to Pervasive computing System**

- Introduces the theme.
- Presents the definitions of the fundamental concepts
- Presents the concepts and the models
- Defines the relation between them

### **Chapter 02: State of art**

- Surveys existing literature and research related Pervasive computing applications and deep learning.
- Discusses relevant concepts, techniques, and models.
- Highlights gaps and opportunities in the field.

### **Chapter 03: Methodology and Architecture**

- Describes the data collection process.
- Details data preprocessing steps.
- Explains the architecture and training and validation of the Human Activity Recognition model.

### **Chapter 04: Results and Discussion**

- Provides an analysis of the experimental results.
- Discusses the performance of the model.
- Visualizes the results using plots and graphs.

### **Chapter 05: General Conclusion**

- Summarizes the contributions of the project.

- Critiques the work by acknowledging limitations.
- Identifies future work and perspectives for further research.

# **Chapter 01: Deep Learning: Application to Pervasive computing System**

## **1.1. Introduction**

The evolution of technology has witnessed a transformative shift towards pervasive computing systems (Miorandi, 2012). In this chapter, we embark on a journey into the monarchy of deep learning and its key role in pervasive computing systems (LeCun, Deep learning, 2015). Our exploration revolves around the integration of deep learning techniques within the pervasive computing landscape (Chen M. G., 2014). Here, we lay the foundation for understanding how deep learning empowers pervasive computing to achieve the potential of interconnected devices and sensors (Atzori, 2010).

Furthermore, we will unravel and clarify the fundamental concepts that underpin our study (Deng, 2012). These conceptual definitions are essential to grasp the intricate relationship between deep learning and pervasive computing, setting the stage for a comprehensive exploration of their applications and implications (Song, 2017).

## **1.2. Definition of the concept deep learning**

Deep learning is a subset of machine learning that uses artificial neural networks to learn from data (Goodfellow, 2016). Unlike traditional machine learning techniques, which require humans to manually engineer features, deep learning algorithms can automatically extract relevant features from raw data (Schmidhuber, Deep learning in neural networks: An overview, 2015). This makes deep learning particularly well-suited for tasks such as image and speech recognition (Krizhevsky, 2012).

To illustrate the concept of deep learning, consider the example of a self-driving car. Traditional machine learning techniques might require engineers to manually program the car to recognize different objects on the road, such as stop signs or pedestrians. With deep learning, however, the car can be trained on large datasets of images and videos, allowing it to automatically

learn to recognize these objects without explicit programming (Bojarski M. D., End to end learning for self-driving cars, 2016).

### **1.3. Importance of deep learning**

Deep learning is revolutionizing the way we approach pervasive computing systems (LeCun, Deep learning, 2015). By using complex algorithms to analyze vast amounts of data, deep learning enables us to make more accurate predictions and better decisions than ever before (Hinton, 2012).

In fact, recent studies have shown that deep learning can improve accuracy rates by up to 30% compared to traditional machine learning techniques (Esteva, Dermatologist-level classification of skin cancer with deep neural networks., 2017). This has huge implications for industries such as healthcare, transportation, and agriculture, where even small improvements in accuracy can have a significant impact on outcomes (Bojarski M. D., 2016) (Dhaliwal, 2020). It's better than any substitute mankind has ever known.

### **1.4. Pervasive computing systems**

Pervasive computing systems, as mentioned before, refer to the integration of technology into everyday life (Weiser, 1991). These systems are characterized by their ability to effortlessly connect devices and services, creating a network that is constantly available and accessible (Satyanarayanan, 2001).

Examples of pervasive computing systems include smart homes, wearable devices, and internet-connected vehicles (Cook, 2007) (Bonato, 2010). These systems enable users to control various aspects of their environment, such as temperature, lighting, and security, using their smartphones or other devices. They also allow for the collection and analysis of data, which can be used to improve efficiency and enhance the user experience, and that's the main goal.

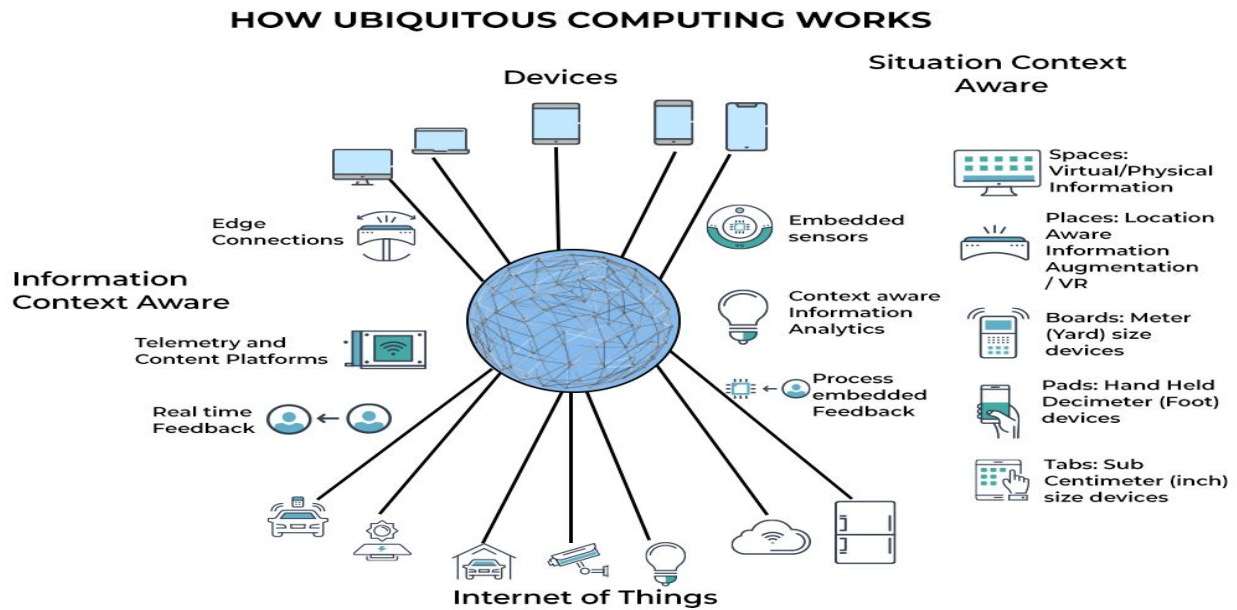


Figure 1 Functioning of pervasive computing

## 1.5. Deep Learning and Pervasive Computing Systems

Deep learning has the potential to develop pervasive computing systems by enabling them to learn from their environment and adapt accordingly (Schmidhuber, Deep learning, 2015). For example, a smart home equipped with deep learning algorithms can learn its occupants' preferences and adjust the temperature, lighting, and music accordingly, which will increase the comfort and satisfaction of the user (Chen M. H., 2018).

In addition, deep learning can improve the accuracy of predictions and decision-making in various industries, such as healthcare and transportation. For instance, deep learning algorithms can analyze medical images and identify potential health risks (Esteva, Dermatologist-level classification of skin cancer with deep neural networks, 2017), or predict traffic patterns and optimize routes for autonomous vehicles (Bojarski M. D., End to end learning for self-driving cars, 2016). This potential for accuracy improvement can result in significant success in these fields.

## 1.6. Deep Learning methods

DL is a subset of machine learning that uses artificial neural networks to learn and make decisions from large volumes of data. Unlike traditional machine learning methods, deep learning models can automatically identify patterns and features in raw data without explicit programming (Géron, 2019), it is one of the leading technologies for automated learning and pervasive computing.

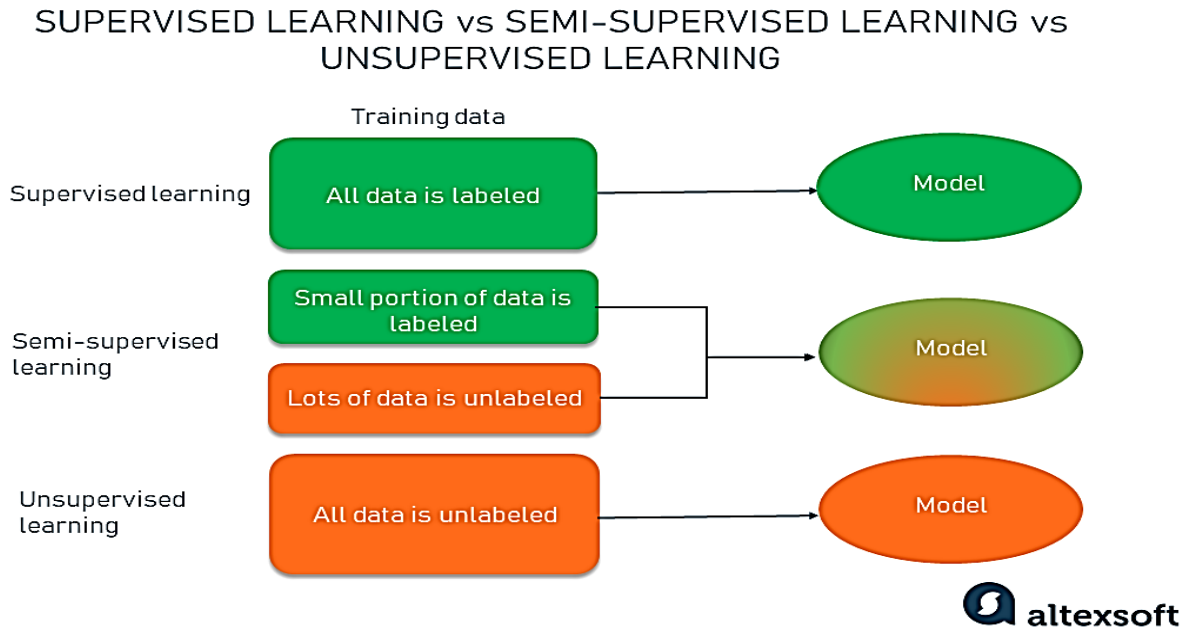


Figure 2 Main Methods of DL

It has established itself and is rapidly developing as one of the most important scientific fields. It is characterized by a neural network, which contains many hidden layers composed of thousands of neurons that each perform small, simple operations. The results of a first layer of neurons are used as input for the calculations of a second layer and so on. There're 3 main types of DL methods that are shown in the figure 4 above.

### 1.6.1. Supervised learning

Supervised learning is a type of machine learning where an algorithm learns from labeled training data to make predictions or decisions without human intervention. It involves training a model on a known dataset, where the input and the corresponding correct output are provided (Alpaydin, 2014).

Our works is pinned under the Supervised learning segment.

### 1.6.2. Unsupervised learning

Unsupervised learning is a type of machine learning where the algorithm is given a dataset without any specific instructions on what to do with it. The goal is to find hidden patterns, structures, or relationships within the data. Unlike supervised learning, there are no labeled outputs or correct answers provided during the training process (Bishop, 2006)

### 1.6.3. Semi-supervised learning

Semi-supervised learning is a machine learning paradigm that combines elements of both supervised and unsupervised learning. In semi-supervised learning, the algorithm is trained on a dataset that contains both labeled and unlabeled data. The goal is to leverage the limited labeled data and the abundance of unlabeled data to improve the model's performance, making it more accurate and robust

## 1.7. Deep Learning Algorithms for Pervasive Computing Systems

There are several deep learning algorithms that can be used in pervasive computing systems, each with its own unique advantages and applications.

### 1.7.1. Fully Connected Neural Network (FCNN)

**Fully connected neural networks** (FCNNs) are a type of artificial neural network where the architecture is such that all the nodes, or neurons, in one layer are connected to the neurons in the next layer.

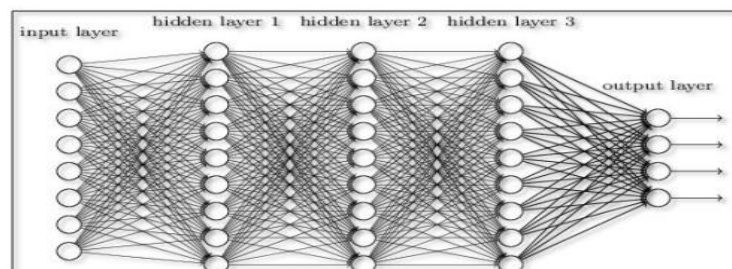


Figure 3 the architecture of FCNN

### 1.7.2. Recurrent Neural Network (RNN)

**Recurrent neural network** (RNNs) is a type of neural network that contains loops, allowing information to be stored within the network. In short, recurrent neural networks use the reasoning of previous experiments to inform future events.

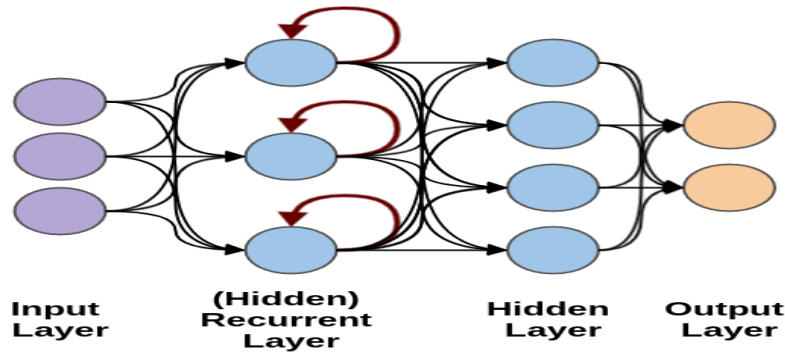


Figure 4 RNN Architecture

### 1.7.3. Convolutional Neural Network (CNN)

CNN has several layers through which data is filtered into categories. CNNs have proven to be very effective in areas such as image recognition, textual language processing, and classification.

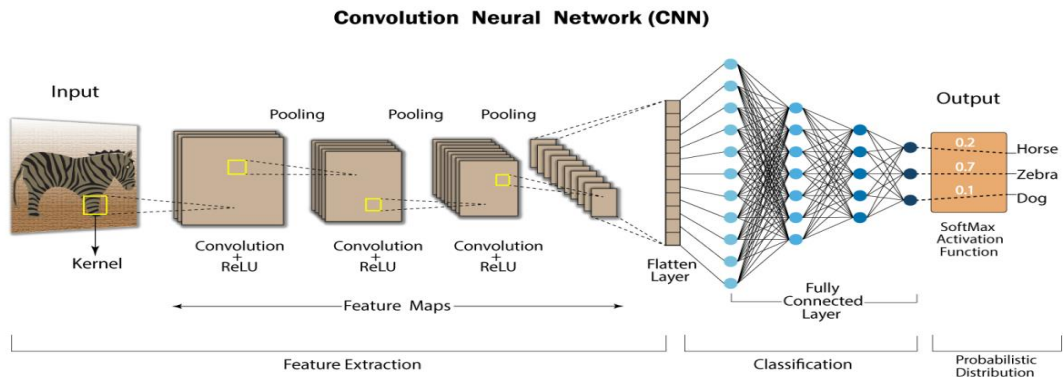


Figure 5 CNN Architecture

### 1.7.4. Long Short-Term Memory (LSTM)

LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This behavior is necessary in complex areas such as machine translation, speech recognition, etc. LSTM networks are a complex area of deep learning.



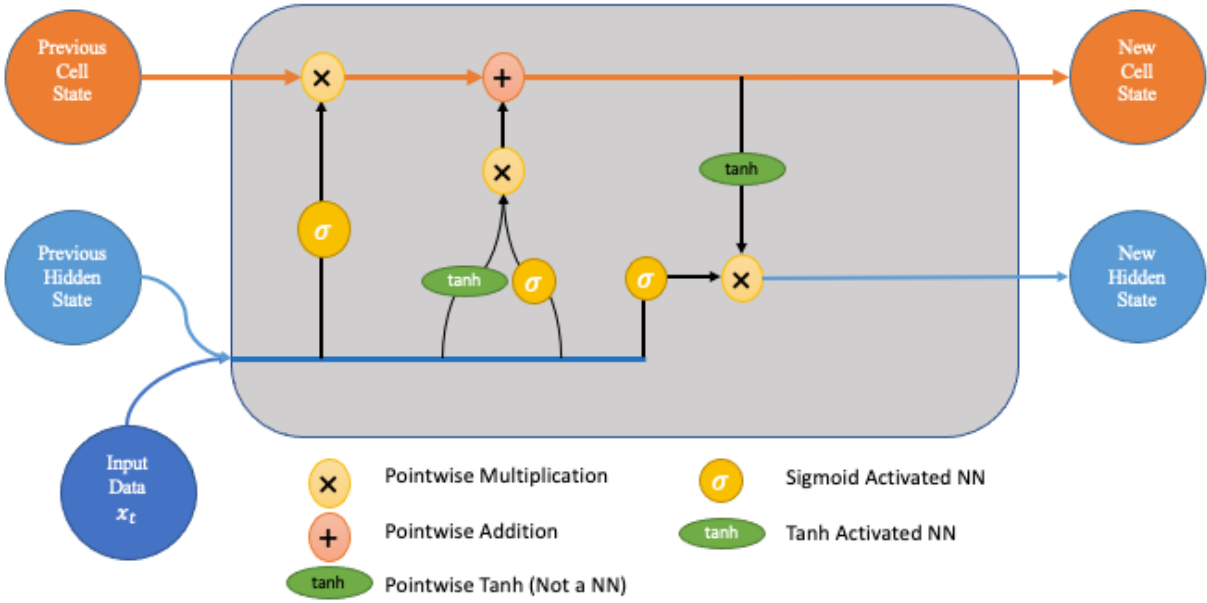


Figure 6 LSTM Architecture

## 1.8. the different hidden layers of a CNN and LSTM network

In the following, the different hidden layers of a CNN and LSTM network are presented.

### 1.8.1. Convolutional Layer

In a CNN, the first layer is always the convolutional layer. This layer applies a convolutional operation to the input and transmits the output to the layer next. A filter (or sometimes called a kernel) is used that traverses all areas of the input and extracts the characteristics. A unidimensional convolutional lock uses a one-dimensional cross-correlation operation in which the convolution window starts on the leftmost side of the input signal and travels through the input signal from left to right successively. When the convolution window slides to a certain position, the input subnets in the window and the kernel network are multiplied and summed element by element to obtain the result at the corresponding location in the output network as shown in Figure 7

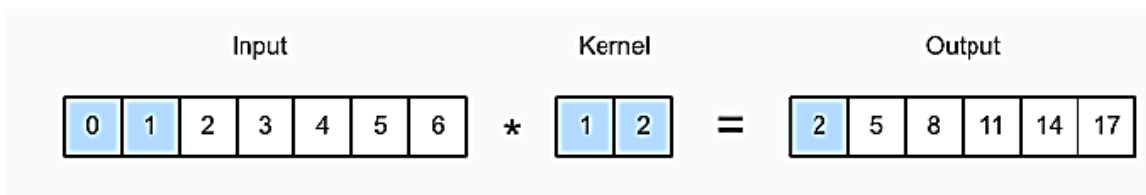


Figure 7 One-dimensional cross-correlation operation.

The blue parts are the first input element as well as the kernel output element used in which:  
 $0 \times 1 + 1 \times 2 = 2$

### 1.8.2. Pooling layer

The pooling layer is used to down sample the output of the convolutional layer. There are several types of pooling such as, mean pooling, max pooling. The max pooling layer essentially takes a window of  $size\ l \times 1$  and then selects the largest of the  $l$  numbers in that window. The same process is applied to the different subregions.

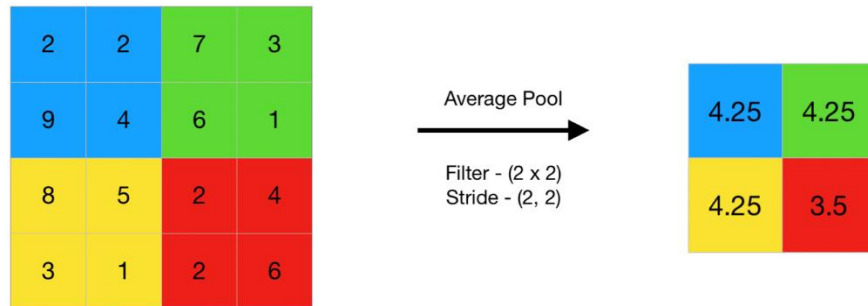


Figure 8 Pooling layer

### 1.8.3. SoftMax

The SoftMax layer in deep learning is the final layer in a neural network used for multi-class classification. It takes raw scores (logits) as input and converts them into a probability distribution over different classes. This layer assigns probabilities to each class, and the class with the highest probability is predicted as the output. It's a critical component for turning neural network predictions into meaningful class probabilities.

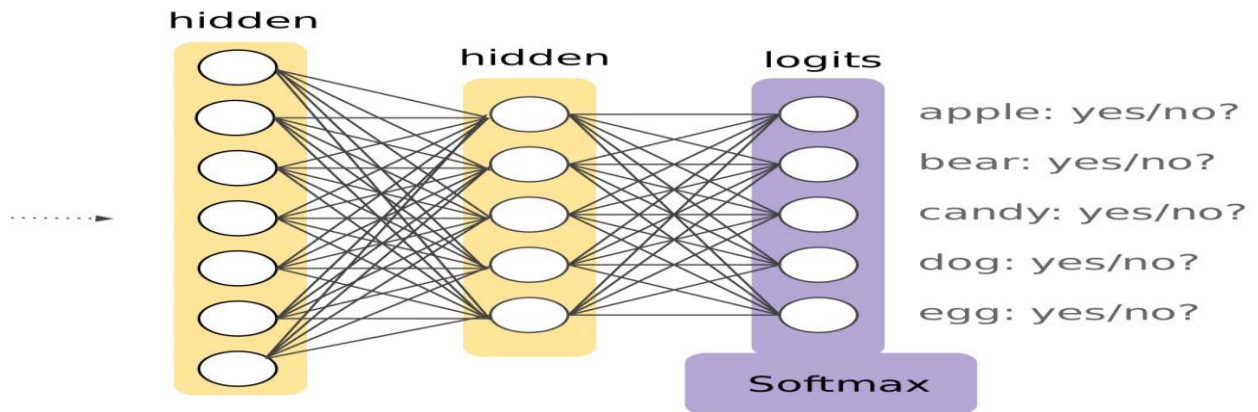


Figure 9 SoftMax Layer

### 1.8.4. Flatten

The role of the Flatten layer is to convert the data into a one-dimensional table and feed it into the next layer. The output of the convolutional layers is flattened to create a single long feature vector, which will be connected to the layer called the fully connected layer.

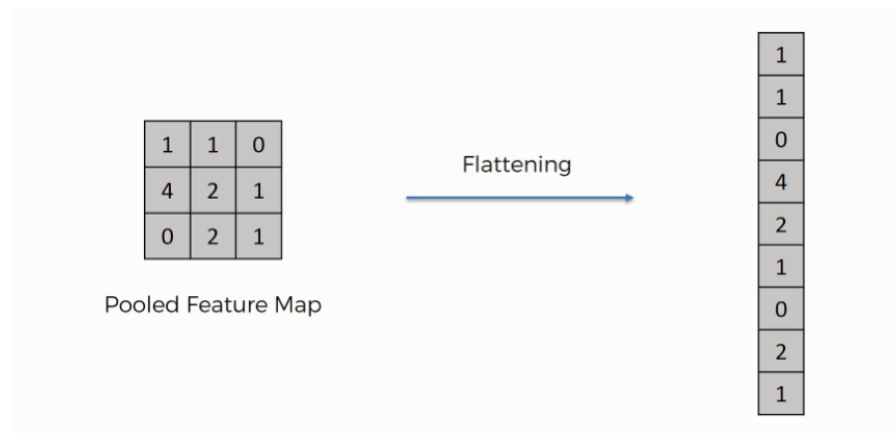


Figure 10 Flatten Layer

### 1.8.5. Fully connected layer

In a fully connected layer, neurons in one layer are connected to all neurons in the second layer. This layer is also present in the classical neural network. The SoftMax function is applied to the output of the second layer. The output of the SoftMax function is used to calculate the probabilities of each label.

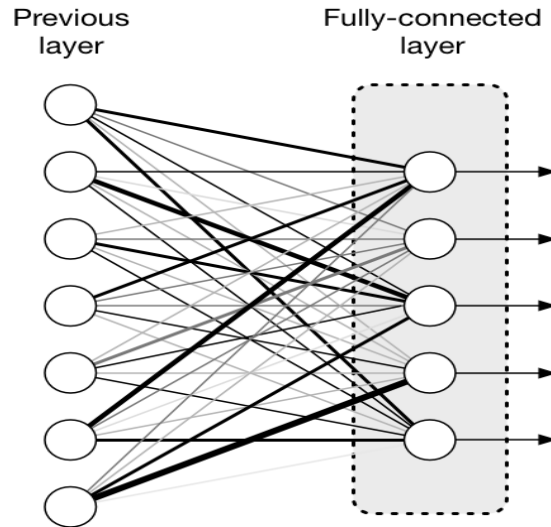


Figure 11 Fully connected layer

### 1.8.6. Wearable sensors

Wearable sensors are used to collect signals directly from users. They are usually attached to different parts of the body such as the waist, wrists, chest, legs and head (Ravi, 2005)[Ravi et al., 2005], but can also be attached to clothing and integrated into other commonly used accessories such as watches, glasses or mobile phones (Brezmes, 2009). They contain a battery unit that

provides the power needed for continuous operation and, for some of them, a wireless unit for transmitting sensor data when needed externally or to interface them with other devices worn on the body. The physiological and motion signals obtained from wearable sensors are very informative for the HAR. Skin temperature, heart rate, heat flow, conductivity, global positioning system (GPS) position, and body movements are some examples of variables that can be measured with current wearable sensor technologies (Yang C. C., 2006). They can be practical, for example, in applications where continuous patient monitoring is required.

Unlike ambient sensors, wearable sensors have advantages in terms of privacy and operating area. In the first case, users are less reluctant to use them anywhere if there is no image or video capture. Second, since these sensors are always worn by the user, they are ubiquitous and their location coverage is virtually unlimited. They also have the advantage of being very portable and do not require stationary equipment. On the other hand, wearable sensors have also brought new

challenges: preserving battery life while being able to collect reliable information from limited detection. These sensors are sometimes uncomfortable for the current user (e.g., if they are too tight or wired or if they need to be constantly repositioned after dressing) and cannot provide a long-term solution for activity monitoring without regular recharging. In addition, hybrid sensing approaches, which combine wearable and ambient sensors from different sources, offer a robust alternative option for HAR. For example, in (Tapia, 2006), a sensor-rich environment was set up for the collection of signals from 72 environmental and body sensors to assess complex activities in an indoor setting.

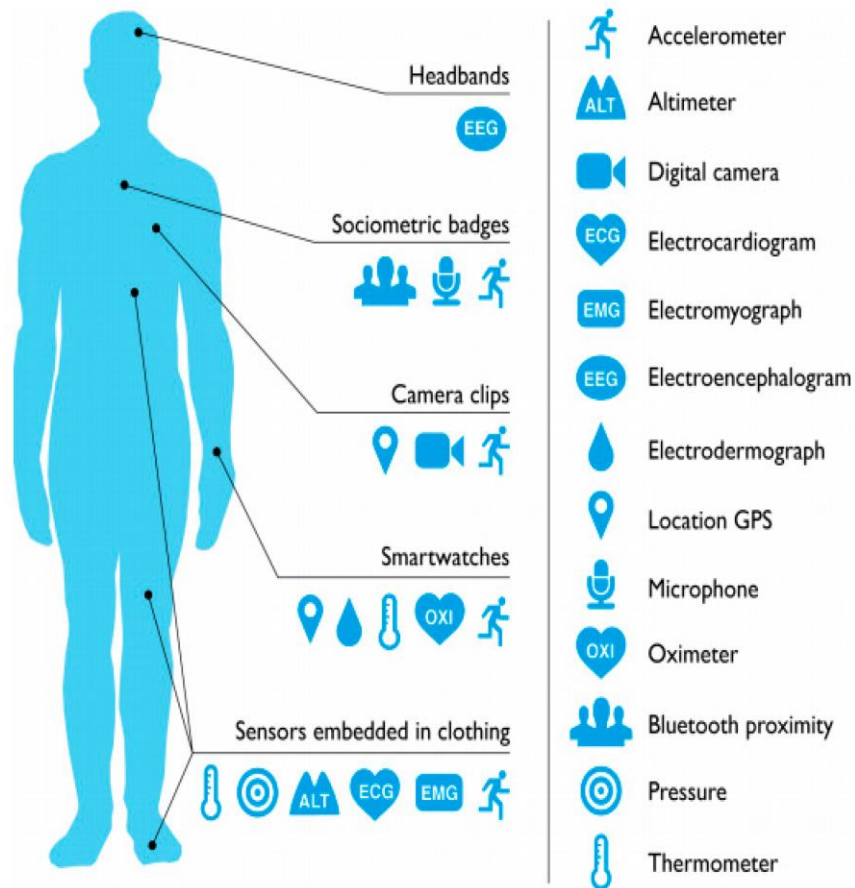


Figure 12 Wearable sensors position in human body

In this work, we use accelerometers and gyroscopes and magnetometer for the classification of human body movements. We describe their main features in the following.

### 1.8.7. Gyroscope

The gyroscope is a sensing device for measuring orientation (Woodman, 2007) It has been used in many applications such as inertial navigation systems, aerial vehicles to increase stability (e.g. in quadricopters) and recently it has been introduced in electronic devices (e.g. smartphones, game consoles) to improve user interfaces and gaming experience. For the HAR, this sensor has been used in various applications such as activity classification (e.g., walking, stair climbing) and transitions between postures (e.g., from standing to sitting) (Coley, 2005) (Altman, 1992).

Gyroscopes have also been produced with MEMS technologies. However, sensors of this type can only measure orientation indirectly. Instead, they estimate the angular velocity that can then be integrated over time to obtain orientation. However, it is first necessary to have an initial angular position of reference to achieve this. These sensors are also very sensitive to noise, which can lead to measurement drift due to integration.

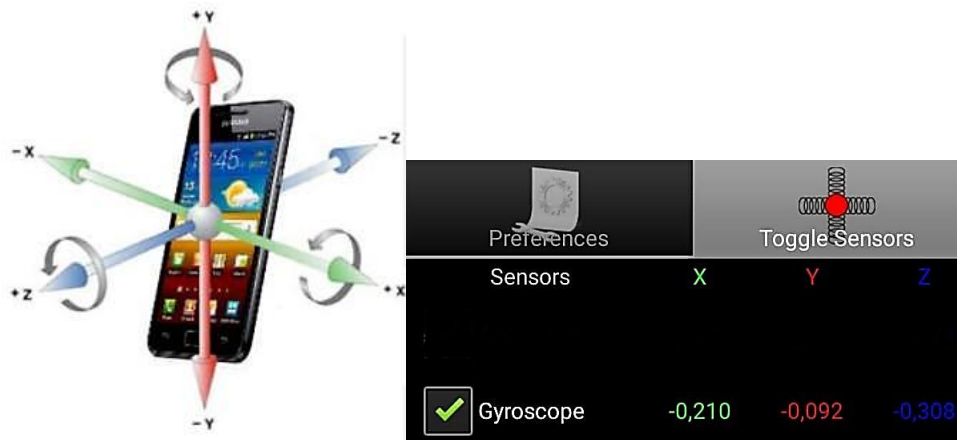


Figure 13 (a) Gyroscope (b)Gyroscope in smartphone

The directions of the axes are shown in **Figure 8**. The raw data of a gyroscope is the rotational speed in (radian per second) rad/s.

### 1.8.8. Accelerometer

The accelerometer is an instrument that measures the physical acceleration experienced by an object. It has been used for several applications in science, medicine, engineering and industry,

such as measuring machine vibration, accelerating high-speed vehicles and moving loads on bridges.

Regarding the HAR, the accelerometer is one of the most widely used sensors to read body movement signals (Mannini, 2010). Its principle of operation generally consists of a seismic mass that moves according to the acceleration to which it is subjected. This displacement can then be translated into a measurable electrical signal. This phenomenon has been applied to the development of sensors for micro-electromechanical systems (MEMS). Their technology makes it possible to create nanoscale devices made with semiconductors. They are advantageous over other sensor technologies because they can be produced on a large scale and with low manufacturing costs. The most common MEMS accelerometers function as a capacitive sensor consisting of a cantilever beam with a test mass whose deflection is correlated with the acceleration experienced by the sensor (Yang X. &, 2010) .

The magnitude and direction of acceleration can be measured as a vector quantity by orthogonally arranging the sensors in the three spatial dimensions. These sensors can also be built on a single chip and it is now common to find triaxial accelerometers in several commercial electronic devices.

One of the problems with using accelerometers to detect body movements is the effect of the gravitational field, which is always present in the measurements and has a relatively high magnitude ( $g = 9.81m/s^2$ ). However, it can also be separated from body movement by filtering. Gravity vector detection can also help determine the orientation of an object relative to the axis of gravitational force when triaxial accelerometers are used.

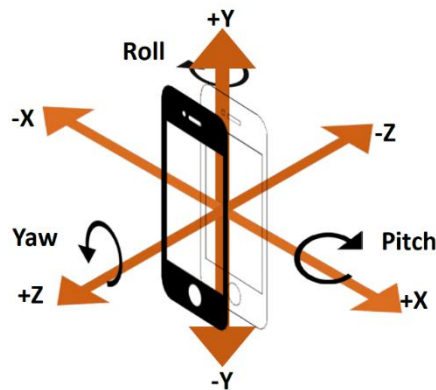


Figure 14 Accelerometer

### 1.8.9. Magnetometer

The magnetometer is a device employed for measuring magnetic fields' strength and direction. It has found applications in various fields, including geophysics, navigation, and consumer electronics. Concerning Human Activity Recognition (HAR), magnetometers are valuable sensors for capturing data related to an individual's orientation and movement in relation to Earth's magnetic field.

In principle, a magnetometer detects changes in magnetic field intensity and direction. This information is then translated into an electrical signal for measurement. In the context of micro-electromechanical systems (MEMS), magnetometers can be designed as nanoscale devices using semiconductor technology. This approach offers scalability and cost-effectiveness in their production, making them widely used in various devices.

MEMS-based magnetometers often employ the Hall effect, where a magnetic field deflects the movement of charge carriers in a semiconductor material, leading to a measurable voltage. These sensors can be arranged in triaxial configurations, allowing them to capture the magnetic field's three-dimensional characteristics.

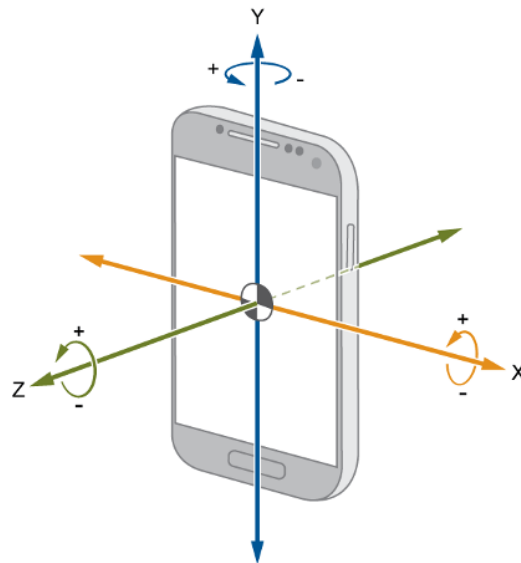


Figure 15 magnetometer

One challenge when using magnetometers for HAR is their susceptibility to external magnetic interference, such as nearby electronic devices or ferrous materials. Proper



calibration and data filtering techniques are necessary to mitigate these interferences and extract meaningful orientation and movement data.

In summary, magnetometers play a crucial role in HAR by providing data on an individual's orientation and movement relative to the Earth's magnetic field. Their MEMS-based design, triaxial configurations, and sensitivity to magnetic changes make them a valuable sensor in many commercial electronic devices.

## **1.9. Conclusion**

In this chapter, we introduced the field of pervasive computing and the integration of deep learning into it and gave the definitions and the importance of the interconnection for both of them.

pervasive computing systems represents a remarkable leap forward in our technological landscape. Deep learning's ability to autonomously analyze data and make precise predictions has far-reaching implications across various sectors, from healthcare to transportation. Pervasive computing systems, with their interconnected devices and user experiences, are poised to benefit immensely from deep learning's capabilities.

# Chapter 02: State of the art

## 2.1. Introduction

Deep Learning, a subset of machine learning, has emerged as a transformative technology with wide-ranging applications. In this chapter, we explore the state of the art in applying Deep Learning to Pervasive Computing. This interdisciplinary field focuses on integrating technology into our daily lives, with applications spanning Internet of Things (IoT), Smart Homes, Smart Cities, and more. Deep Learning techniques have played a pivotal role in enhancing the capabilities of pervasive computing systems.

## 2.2. Literature review

In the realm of applying Deep Learning to Pervasive Computing, numerous research endeavours have significantly advanced the field. This section provides an overview of some of the most influential works, shedding light on the multifaceted approaches adopted to leverage Deep Learning in creating intelligent and adaptive environments. These labours are underpinned by the premise that personal and ubiquitous detection methods, notably those involving smartphones, offer discreet means to continuously collect pertinent data (Garcia-Ceja et al., 2018)

One pivotal study by Wei and Ma (2023) explored the application of deep learning-based speech systems in online music learning systems. This work exemplifies how Deep Learning techniques can enhance user experiences within pervasive computing applications, such as online education (Wei and Ma, July 2023).

Furthermore, Zeng et al. (2020) conducted a comprehensive survey of deep learning-based approaches for Human Activity Recognition (HAR). This survey elucidates the various deep learning techniques employed for HAR, which is essential in domains such as healthcare, smart homes, and more (Zeng et al., 2020).

The fusion of Deep Learning with Pervasive Computing extends to edge computing as well, Shi et al. (2019) reviewed the integration of deep learning into edge computing, a pivotal concept

in pervasive computing. This review underscores the significance of deploying deep learning models at the edge of the network to improve efficiency and responsiveness (Shi et al., 2019).

Yao et al. (2017) introduced "DeepSense," a unified deep learning framework for time-series mobile sensing data processing, making it applicable to pervasive computing (Yao et al., 2017).

Ronao and Cho (2016) demonstrated the use of deep learning models for human activity recognition, a crucial application of pervasive computing, in their work "Human Activity Recognition Using Smartphones with Deep Learning Models" (Ronao and Cho, 2016).

Zhuang et al. (2018) provided a comprehensive overview of deep learning techniques in the context of sensor-based human activity recognition, a core component of pervasive computing, in their survey paper (Zhuang et al., 2018).

The ground-breaking research effort by Hassan et al. (2018) introduced the use of deep convolutional neural networks (CNNs) for Human Activity Recognition (HAR). Their work showcases the power of deep learning in accurately recognizing and classifying human activities, a fundamental capability in pervasive computing systems (Hassan et al., 2018).

These works collectively provide a foundation for understanding the diverse applications of Deep Learning in the pervasive computing domain, offering valuable insights for further research and innovation, also these efforts underscore the dynamic evolution of pervasive computing domain.

### **2.3. Applications of Deep Learning in Pervasive Computing Systems**

Deep learning has numerous applications in pervasive computing systems, from smart homes to smart healthcare and beyond. One example of its application is in the field of image recognition, where deep learning algorithms can be used to identify objects and people in images with high accuracy. This has potential benefits for security systems, as well as for improving accessibility for visually impaired individuals.

Another example is in the area of natural language processing, where deep learning algorithms can be used to improve speech recognition and language translation. This has potential benefits for communication devices, such as smartphones and personal assistants, as well as for language learning and cross-cultural communication.

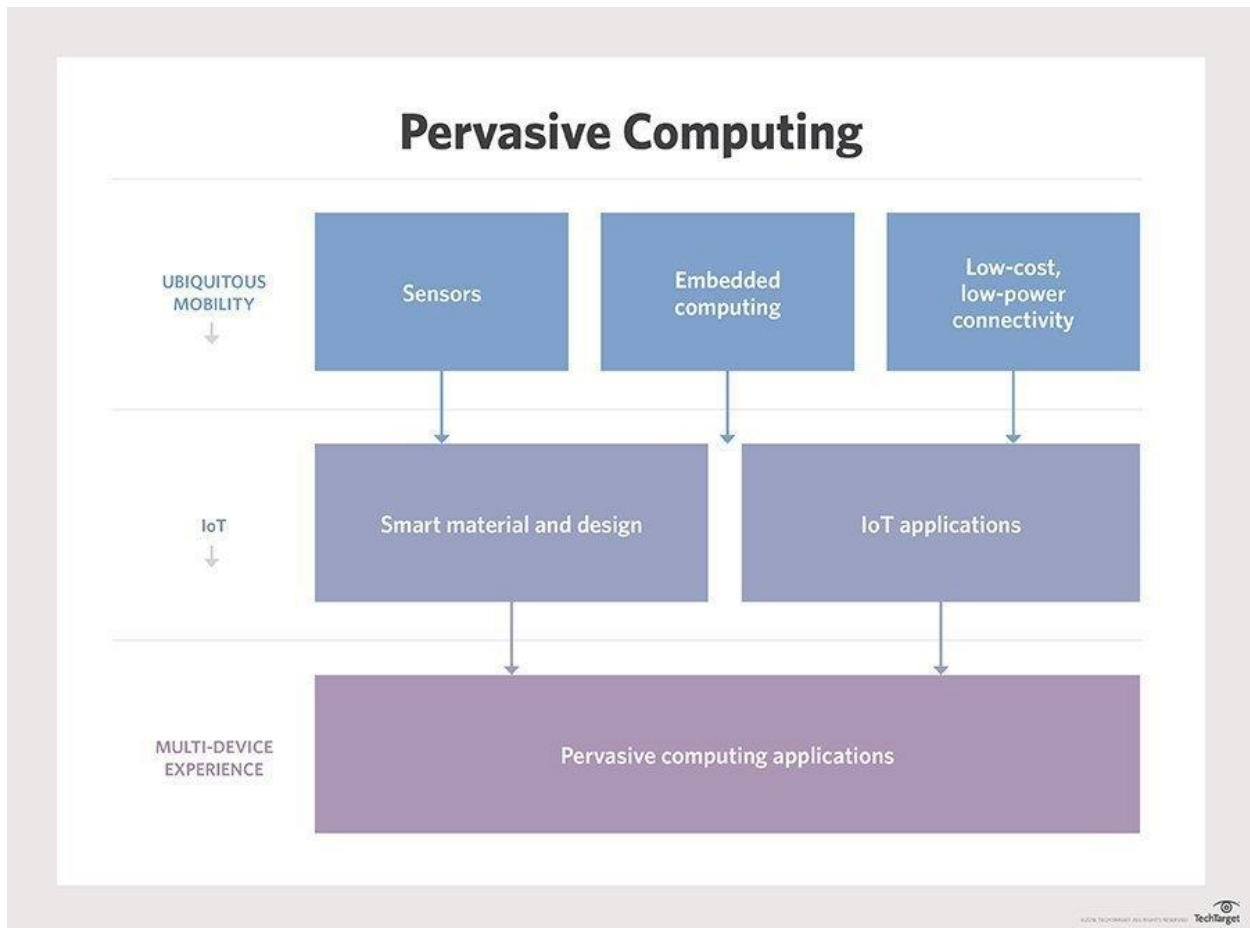


Figure 16 Pervasive computing architecture

### 2.3.1. Smart Homes

Smart homes are becoming increasingly popular as technology advances and more devices become connected to the internet of things (IoT). Deep learning can be applied in smart homes to improve their functionality and make them more efficient. The integration of Deep Learning into Pervasive Computing has also paved the way for more personalized and efficient services. Al-Fuqaha et al. (2018) provided a comprehensive survey of deep learning in the context of the

Internet of Things (IoT). This survey highlights how deep learning techniques are leveraged to enhance IoT, a critical component of pervasive computing systems (Al-Fuqaha et al., 2015).

One example of how deep learning can be used in smart homes is through the implementation of intelligent energy management systems as well, Xu et al. (2019) focused on compressing deep neural networks for resource-constrained IoT devices, a significant aspect of pervasive computing, in their work "DeepIoT" (Xu et al., 2019). These systems use data from various sensors and devices within the home to optimize energy usage and reduce waste. By analyzing patterns in energy consumption, these systems can learn the habits and preferences of the occupants and adjust accordingly. This not only saves money on energy bills but also reduces the carbon footprint of the household.



Figure 17 Smart home features

### 2.3.2. Smart Cities

Smart cities are urban areas that use technology and data to improve the quality of life for their citizens. Deep learning can play a crucial role in making smart cities more efficient and sustainable. By analysing vast amounts of data from sensors and other sources, deep learning algorithms can

provide insights into traffic patterns, energy usage, and other key factors that affect city life. Hodge et al. (2019) explored how deep learning is applied to IoT data, which is a central element of pervasive computing, in their survey titled "Deep Learning for IoT Big Data and Streaming Analytics: A Survey" (Hodge et al., 2019).

For example, deep learning can be used to optimize traffic flow by predicting congestion and suggesting alternative routes. It can also help reduce energy consumption by identifying areas where lighting or heating can be adjusted based on occupancy levels. These are just a few examples of how deep learning can make smart cities more liveable and environmentally friendly.

### **2.3.3. Smart Healthcare**

Deep learning has the potential to revolutionize the healthcare industry by improving patient outcomes and reducing costs. One example of this is the use of deep learning algorithms to analyse medical images such as X-rays and MRIs. By automating the analysis process, doctors can receive more accurate and timely diagnoses, leading to better treatment options for patients.

Another application of deep learning in smart healthcare is the use of wearable devices to monitor patient health. By collecting and analysing data on a continuous basis, doctors can detect potential health issues before they become serious, allowing for earlier intervention and better outcomes. For example, a wearable device could detect changes in heart rate or blood pressure that indicate an impending heart attack, prompting the patient to seek medical attention before it's too late. In the context of healthcare, Nweke et al. (2018) delved into deep learning-based Human Activity Recognition (HAR) for healthcare applications. This research demonstrates the potential of deep learning models in continuously monitoring patients' vital signs and detecting anomalies, enabling early interventions during health emergencies (Nweke et al., 2018).

### **2.3.3. Smart Transportation**

Smart transportation is an area where deep learning can make a significant impact. By using advanced algorithms to analyse data from sensors and cameras, transportation systems can become more efficient and safer. For example, deep learning can be used to predict traffic patterns and optimize routes for public transportation. This can reduce travel time and improve the overall experience for commuters.

Another application of deep learning in smart transportation is autonomous vehicles. By using deep learning algorithms to analyse sensor data from cameras and radar systems, self-driving cars can make decisions in real-time and navigate complex environments with ease. This can reduce the number of accidents on the road and make transportation more accessible for people who are unable to drive. In the domain of transportation, Wei et al. (2019) explored the use of deep reinforcement learning for intelligent transportation systems. Their research illustrates how deep learning can optimize transportation systems within smart and connected cities (Wei et al., 2019).

## **2.4. Conclusion**

In conclusion, the combination of Deep Learning with Pervasive Computing is transforming the way we interact with technology. It empowers our environments, from IoT devices to entire cities, to become smarter, more responsive, and better aligned with human needs. The diverse applications and innovative techniques discussed in this chapter exemplify the dynamic evolution of this interdisciplinary field. As researchers continue to innovate, the potential for Deep Learning in Pervasive Computing becomes increasingly promising

# Chapter 03: Methodology and Architecture

## 3.1. Introduction

This chapter will explain the comprehensive methodology, architectural choices, and approach applied to harness our deep learning for Human Activity Recognition (HAR) within pervasive computing systems. The chosen approach encompasses data collection, preprocessing, model architecture, training, validation, ethical considerations, and technological frameworks.

## 3.2. Architecture Selection

In pursuit of optimal results, we embarked on selecting two primary architectures: Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks. These architectures were chosen due to their aptness for sequential and spatial data. A CNN excels in capturing spatial features, while an LSTM adeptly handles sequential dependencies. This choice catered to the diverse data dimensions inherent in HAR.



Figure 18 Data set body positions

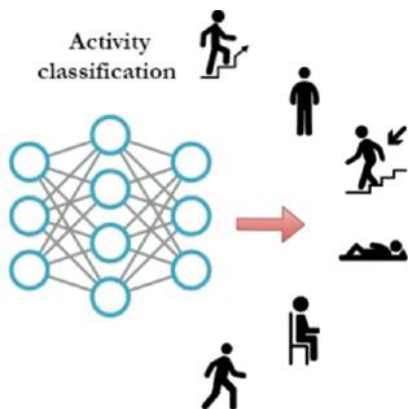


Figure 19 Activity classes Standing, Sitting, Laying, Walking, Upstairs, Downstairs



	A	B	C	D	E	F	G	H	I	J	K
1	Time_Stamp	Ax	Ay	Az	Gx	Gy	Gz	Mx	My	Mz	Activity_Label
2	1,3644E+12	-17,365944	19,517958	0,88532263	-0,12186762	2,1774292	1,5357152	18,3	-44,16	8,639999	Downstairs
3	1,3644E+12	-9,684067	13,933616	1,1577295	-0,053145275	-1,751656	1,2541064	17,279999	-44,16	9,179999	Downstairs
4	1,3644E+12	-4,0452433	7,709117	-1,2666923	-0,59650993	-3,4718525	1,1765264	16,5	-44,399998	9,36	Downstairs
5	1,3644E+12	-1,7706453	5,7886477	-0,7354988	-0,8677341	-2,9837713	0,89369583	15,9	-44,52	9,36	Downstairs
6	1,3644E+12	2,819412	3,9635212	0,5992953	-0,5412266	-2,6627617	0,3286455	15	-44,7	9,24	Downstairs
7	1,3644E+12	5,611583	4,1814466	0,8308412	-0,48624873	-1,9947804	-0,14752395	13,98	-44,76	9,12	Downstairs
8	1,3644E+12	6,2789803	3,8273177	0,8036005	-0,51007247	-0,8530733	-0,41508293	13,559999	-45	8,88	Downstairs
9	1,3644E+12	5,47538	2,0158114	0,626536	-0,4025602	0,28802297	-0,49296826	12,78	-45,3	8,5199995	Downstairs
10	1,3644E+12	2,2882185	3,336985	2,0566726	-0,36865717	0,95569867	-0,6178902	12,3	-45,66	8,04	Downstairs
11	1,3644E+12	-0,040861044	7,43671	2,6832085	-0,3387248	1,1157454	-0,6640105	12,3	-45,84	7,56	Downstairs
12	1,3644E+12	-3,2280223	18,632635	0,5720546	-0,44409904	0,9224065	-0,44959682	12,48	-46,2	6,8999996	Downstairs
13	1,3644E+12	-0,14982383	19,504337	7,3141265	-0,301462	0,5021312	1,2776246	12,78	-46,02	6,24	Downstairs
14	1,3644E+12	1,96133	19,517958	10,051817	0,78801614	1,1615603	0,7852673	12,84	-46,14	6	Downstairs
15	1,3644E+12	0,87170225	18,864182	9,575105	1,344209	2,4242187	0,3475823	13,0199995	-46,2	6	Downstairs
16	1,3644E+12	-5,271075	11,672638	1,7706453	0,43799037	2,7363708	-0,31978795	13,08	-46,44	6,06	Downstairs
17	1,3644E+12	-3,2280223	7,8317	-0,13620348	-0,3549127	1,9657644	-0,6206391	13,139999	-46,559998	6,06	Downstairs
18	1,3644E+12	0,9670447	5,3527966	0,35412905	-0,77549344	1,0763446	-0,61269784	12,9	-46,559998	6,2999997	Downstairs
19	1,3644E+12	2,2882185	4,3176503	0,7627395	-0,83658	0,5852089	-0,41996986	12,9	-46,62	6,24	Downstairs
20	1,3644E+12	3,92266	1,0896279	1,5390993	-0,70463306	0,4990769	-0,25442538	12,78	-46,62	6,24	Downstairs
21	1,3644E+12	2,669588	-2,3426998	2,0975335	-0,63529986	0,8772025	-0,28802297	12,48	-46,379997	6,18	Downstairs
22	1,3644E+12	0,6537767	-3,1871614	0,84446156	-0,7480045	1,5070046	-0,29718593	12,36	-46,26	6,18	Downstairs
23	1,3644E+12	-0,54481393	-3,3233647	-1,7297841	-0,98746365	1,6331482	0,12614368	12,24	-46,32	6,12	Downstairs

Figure 10 Schematic diagram showcasing the structure of the collected data

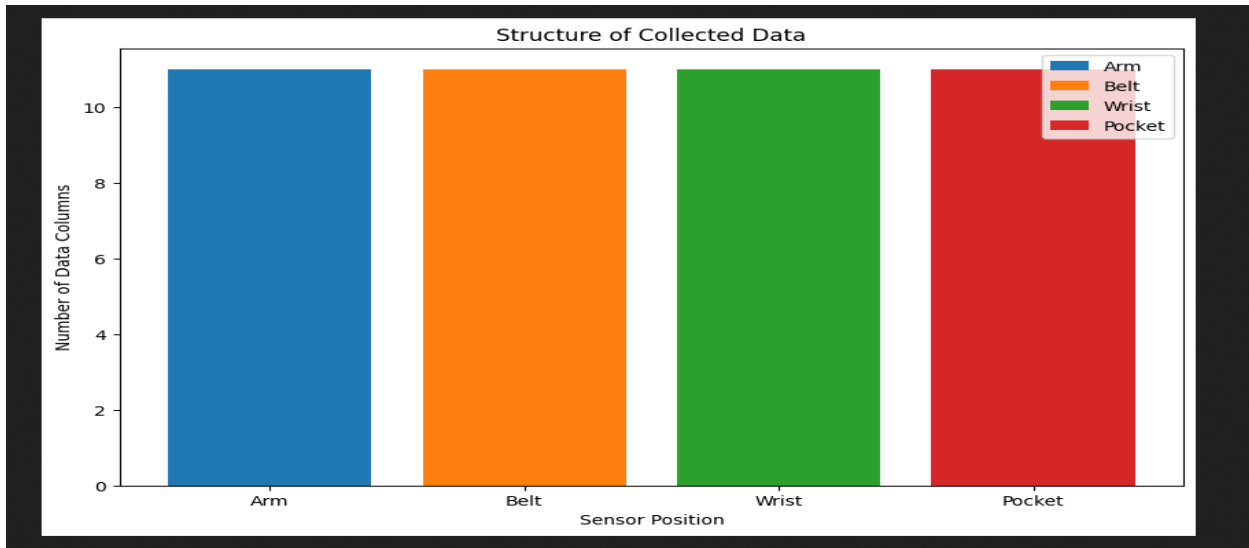


Figure 11 Arm.xlsx dataset file sensors and their positions & readings with activity labels

### 3.6. Data Collection and Preprocessing

The foundation of accurate HAR lies in robust data collection and preprocessing. We utilized the "Shoib, Muhammad" dataset, comprising Arm, Belt, Wrist, and Pocket sensor positions. These datasets encompassed timestamped readings of accelerometer, gyroscope, magnetometer axes, and corresponding activity labels. Leveraging the meticulous data labeling and pre-

processing culminating in the fusion of datasets sourced from various sensor positions provided by "Shoaib, Muhammad," we ensured a reliable basis for our research

### 3.7. Model Training

The model training process was instrumental in realizing the potential of deep learning for Human Activity Recognition (HAR) within pervasive computing systems. For our investigation, we considered two prominent architectures: Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks.

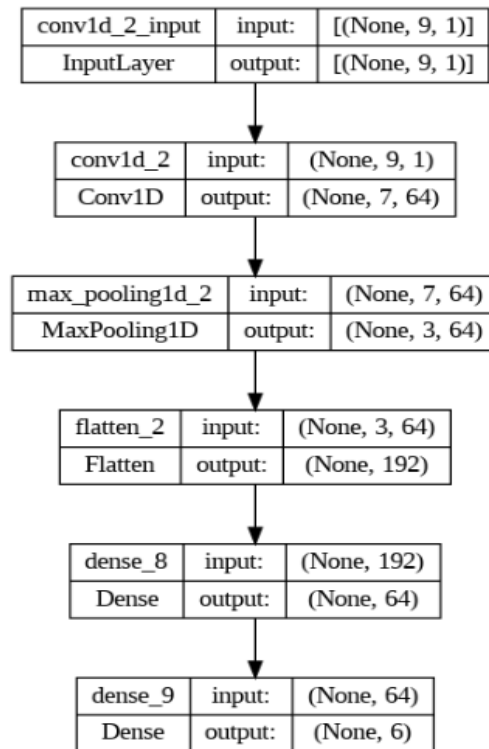


Figure 20 CNN layers

For the CNN architecture, we harnessed its capacity for spatial feature extraction. Our CNN model was constructed using sequential layers comprising a Conv1D layer, a MaxPooling1D layer, and a Flatten layer. This design enabled the CNN to identify spatial patterns within the sensor data. Subsequently, fully connected Dense layers were added to enhance the model's representation capabilities. The final Dense layer employed a softmax activation function, enabling multiclass classification as shown above in figure 13.

In contrast, our LSTM architecture leveraged its strength in capturing temporal dependencies. We fashioned an LSTM model, characterized by a single LSTM layer preceded by an input layer. This LSTM layer was chosen for its ability to process sequential data effectively. The architecture was augmented with Dense layers to enhance feature extraction and classification. Similar to the CNN, a softmax-activated Dense layer was employed for multiclass prediction.

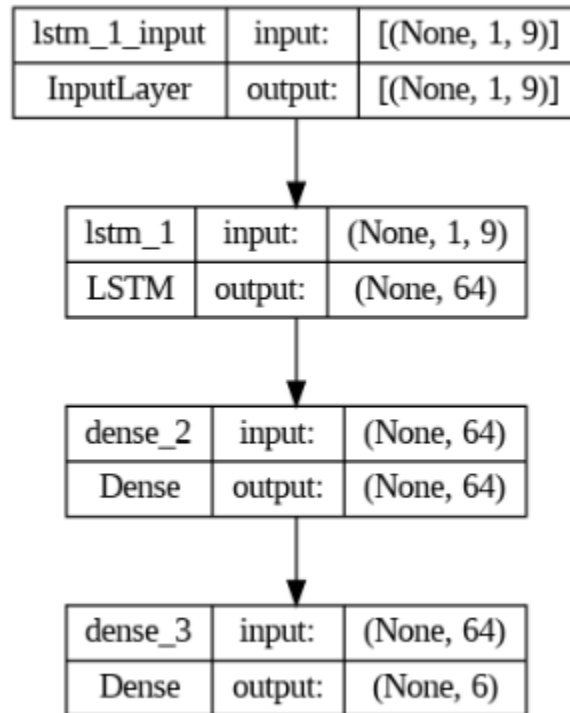


Figure 21 LSTM layers

The two distinct architectures allowed us to explore both spatial and sequential characteristics of the data, enriching our understanding of their effectiveness in HAR scenarios.

### 3.8. Approach Validation

Approach validation entailed evaluating the CNN and LSTM models against a comprehensive dataset. The dataset partitioning scheme, as suggested by "Shoaib, Muhammad," was employed, facilitating consistent and trusted validation. Our dataset was split into 80% for training and 20% for testing. Our assessment involved utilizing established evaluation metrics to gauge the performance of the models, including confusion matrices and accuracy graphs.

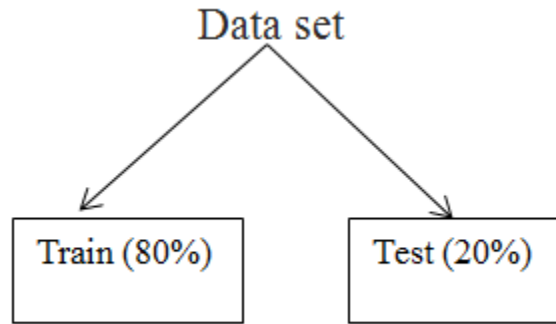


Figure 22 Dataset split

### 3.8.1. Confusion Matrix

In a two-class problem, the classification model predicts one of two classes that are generally referred to as positive and negative classes. Given a classifier and an example to be classified, four situations are conceivable. If the example is positive and is classified as positive, it counts as a true positive TP; if it is classified negative, it counts as a false negative FN.

Table 1 Confusion table of a two-class problem

Confusion matrix	
TP	TN
FP	FN

If the example is negative and is classified as negative, it counts as a true negative TN; if it is classified positive, it counts as a false positive FP. The outputs produced by the classifier can be represented by a  $2 \times 2$  construction matrix. Table 4.1 shows the confusion matrix for a two-class problem where the rows indicate the actual class and the columns indicate the predicted class.

### 3.8.2. Classification rate

The rate of good classification is the simplest measure of performance and is the basis of any other criterion (Pierre Baldi et al., 2000). The classification rate calculates the percentage of

correctly classified samples in relation to the total number of samples. From the values in the confusion table, the classification rate is given by the following equation:

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

Figure 23 Equation of classification rate

### 3.8.3. Performance Metrics

To assess the efficacy of our models, we employed a set of performance metrics, including accuracy, precision, recall, and F1-score. These metrics provided a comprehensive understanding of the models' abilities to accurately classify different human activities.

Metric	Formula
True positive rate, recall	$\frac{TP}{TP+FN}$
False positive rate	$\frac{FP}{FP+TN}$
Precision	$\frac{TP}{TP+FP}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
F-measure	$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Figure 24 Metric and Formula

In the case of the CNN model, we observed that it exhibited commendable accuracy, effectively capturing spatial nuances within the sensor data. On the other hand, the LSTM model excelled at capturing temporal dependencies, showcasing its prowess in recognizing sequential patterns in human activities.

Architecture	Accuracy	Loss	Epoch
LSTM	92%	0.22%	100

Table 2 LSTM accuracy and loss and time per learning step

Architecture	Accuracy	Loss	Epoch
CNN	88%	0.3%	100

Table 3 CNN accuracy and loss and time per learning step

We visualized the validation outcomes through training and validation accuracy per time step over epochs. These images illuminated the models' learning trajectories, illustrating their convergence and potential overfitting points.

### 3.9. Ethical Considerations

Our pursuit of accurate HAR in pervasive computing systems was accompanied by a deep commitment to ethical considerations. We were mindful of the implications of dealing with paramount importance of addressing algorithmic bias. We acknowledged that, despite the power of deep learning, there exists a potential for misinterpretation and bias in activity recognition. This insight underscored the need for thorough validation, fair representation, and continuous monitoring to mitigate biases that could result in skewed results, unfair treatment, and erroneous insights

### 3.10. Software and Hardware Frameworks

#### 3.10.1 Machines

Specifications:

• <b>PC: Mars Gaming / Lenovo-thinkpad L420</b>
• <b>Operating system :</b> Windows 10 Professionnel
• <b>Processor :</b> 11th Gen Intel® Core™ i5-11400F @ 2.60GHz / intel(r) core(tm) i5-2430m cpu @ 2.40ghz 2.40 ghz
• <b>RAM :</b> 16,00 GB / 4,00 Go
• <b>GPU :</b> RTX 3070TI / HD Intel® 3000

Table 4 Machine Specifications

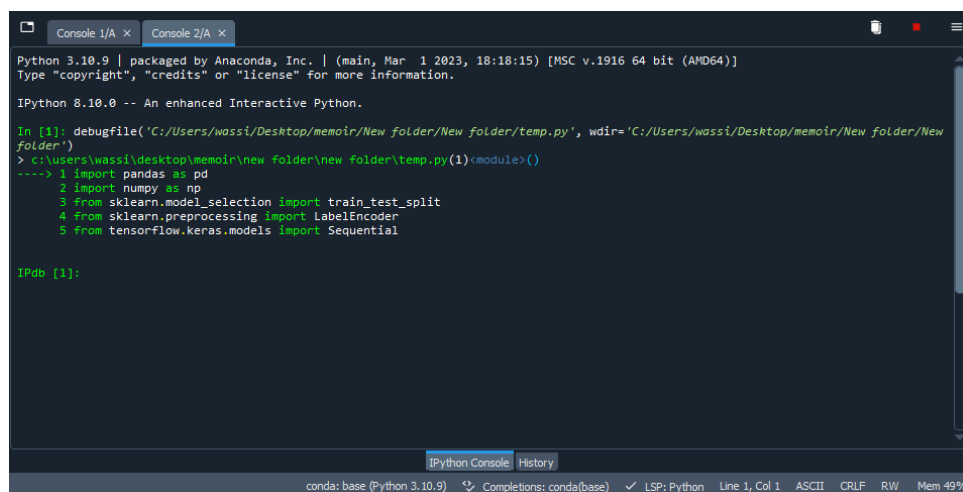
## 3.10.2 Spyder IDE

Spyder's integrated development environment provided us with several key advantages, the Spyder IDE served as an invaluable tool in our deep learning research journey, providing us with a controlled and feature-rich environment for developing, optimizing, and fine-tuning our models

```
19
20 # Step 3: Load and preprocess the data
21 arm_data = pd.read_excel(r'C:\Users\wassil\Desktop\memoir\New folder\New folder\Activity_Recognition_DataSet\Arm.xlsx')
22 belt_data = pd.read_excel(r'C:\Users\wassil\Desktop\memoir\New folder\New folder\Activity_Recognition_DataSet\Belt.xlsx')
23 pocket_data = pd.read_excel(r'C:\Users\wassil\Desktop\memoir\New folder\New folder\Activity_Recognition_DataSet\Pocket.xlsx')
24 wrist_data = pd.read_excel(r'C:\Users\wassil\Desktop\memoir\New folder\New folder\Activity_Recognition_DataSet\Wrist.xlsx')
25
26 # Step 4: Merge the datasets
27 all_data = pd.concat([arm_data, belt_data, pocket_data, wrist_data], ignore_index=True)
28
29 # Step 5: Encode the activity labels
30 label_encoder = LabelEncoder()
31 all_data['Encoded_Label'] = label_encoder.fit_transform(all_data['Activity_Label'])
32
33 # Step 6: Prepare the data for CNN
34 X = all_data[['Ax', 'Ay', 'Az', 'Gx', 'Gy', 'Gz', 'Mx', 'My', 'Mz']]
35 y = all_data['Encoded_Label']
36
37 # Convert X and y to numpy arrays
38 X = np.array(X)
39 y = np.array(y)
40
41 # Reshape X to a 3D array [samples, timesteps, features]
42 X = np.reshape(X, (X.shape[0], X.shape[1], 1))
43
44 # Split the data into training and testing sets
45 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
46
47 # Step 7: Build and train the CNN model
48 CNN_model = Sequential()
49 CNN_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
50 CNN_model.add(MaxPooling1D(pool_size=2))
51 CNN_model.add(Flatten())
52 CNN_model.add(Dense(64, activation='relu'))
53 CNN_model.add(Dense(len(label_encoder.classes_), activation='softmax'))
54
55
56 CNN_model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 25 Spyder IDE

**Code Proficiency:** With Spyder, we could write, test, and debug our code seamlessly. The IDE's robust debugging tools and interactive consoles enabled us to identify and resolve issues efficiently.



```
Python 3.10.9 | packaged by Anaconda, Inc. | (main, Mar 1 2023, 18:18:15) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 8.10.0 -- An enhanced Interactive Python.

In [1]: debugfile('C:/Users/wassil/Desktop/memoir/New folder/New folder/temp.py', wdir='C:/Users/wassil/Desktop/memoir/New folder/New folder')
> C:/Users/wassil/Desktop/memoir/new folder/new folder/temp.py(1)<module>()
----> 1 import pandas as pd
      2 import numpy as np
      3 from sklearn.model_selection import train_test_split
      4 from sklearn.preprocessing import LabelEncoder
      5 from tensorflow.keras.models import Sequential

IPdb [1]:

Python Console History
conda: base (Python 3.10.9)  Completions: conda(base)  LSP: Python  Line 1, Col 1  ASCII  CRLF  RW  Mem 49%
```

Figure 26 Spyder Console

**Integration with Libraries:** Spyder effortlessly integrated with popular Python libraries and deep learning frameworks. This compatibility streamlined the implementation of complex neural network architectures.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tqdm import tqdm
import tkinter as tk
from tkinter import filedialog, messagebox
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score
from tensorflow.keras.models import load_model
import seaborn as sns
```

Figure 27 Spyder Libraries

**Data Visualization:** Spyder seamlessly integrated with data visualization libraries like Matplotlib and Seaborn, enabling us to generate informative plots and graphs to visualize our research results.

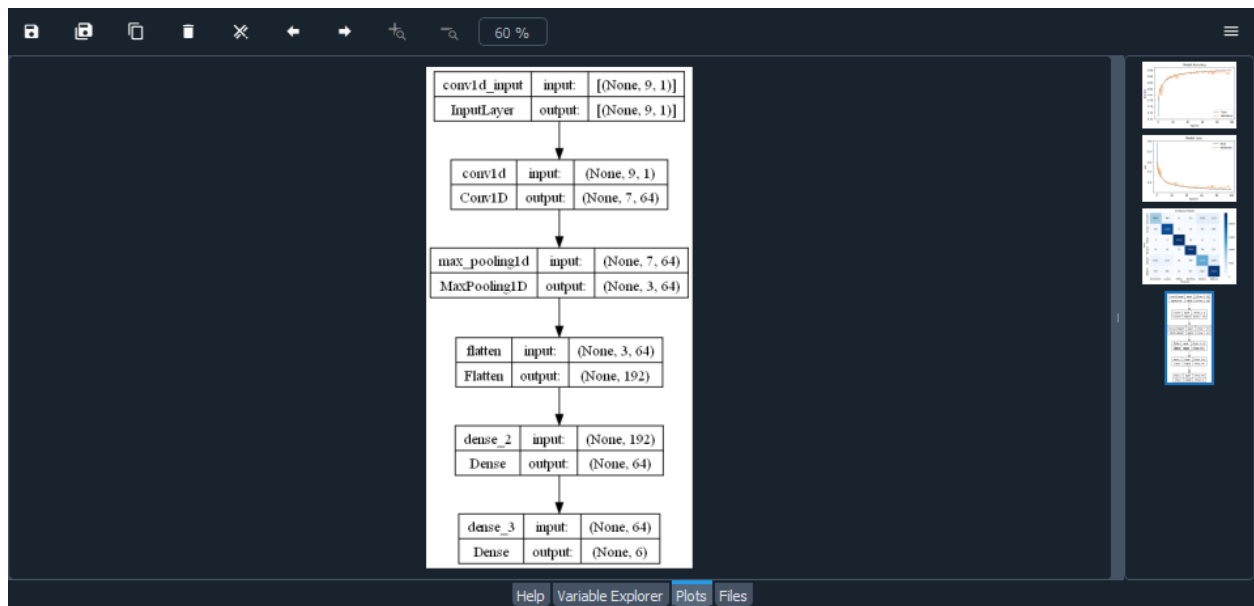


Figure 28 Visualization Terminal



### **3.11. Conclusion**

This chapter outlined a comprehensive methodology for Human Activity Recognition (HAR) within pervasive computing systems. We began by meticulously collecting and preprocessing data, utilizing Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to build robust HAR models. Rigorous validation, ethical considerations, and model evaluation played pivotal roles in our approach. By synergizing software and hardware frameworks, we harnessed deep learning's potential. Subsequent chapters will delve into our findings, contributing to the field's progress.

# Chapter 04: Implementation and Results

## 4.1. Introduction

In this chapter, we delve into the practical implementation of our approach and present the outcomes of our experiments. We begin by discussing the setup and execution of the deep learning models developed for Human Activity Recognition (HAR) within pervasive computing systems. Subsequently, we present a comprehensive analysis of the results obtained from our models' performance and evaluate their effectiveness in recognizing human activities.

## 4.2. Model Implementation

### 4.2.1. Data Preprocessing

Before training our models, the collected sensor data underwent thorough pre-processing as mentioned in chapter 3. This involved handling missing values, normalizing the data, and encoding the activity labels. The processed dataset was then divided into training and testing sets, with an 80-20 split, respectively. This partitioning strategy ensured a robust evaluation of model generalization.

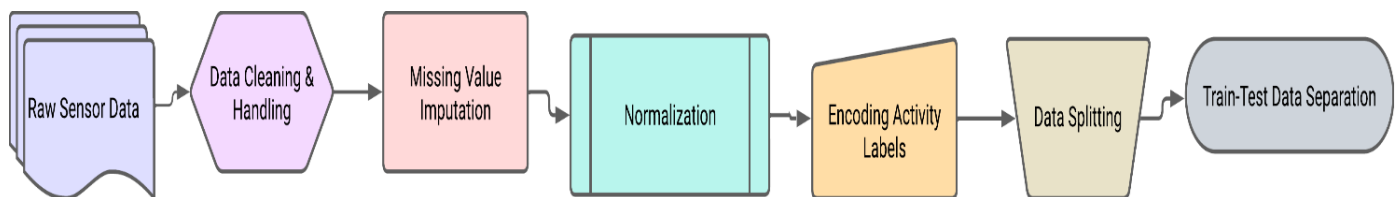


Figure 29 Flowchart of the sensors pre-processing data

## 4.2.2. CNN and LSTM Architecture Instantiation

We instantiated two distinct deep learning architectures: The Convolutional Neural Network (CNN) and the Long Short-Term Memory (LSTM) network. The CNN was designed to capture spatial features within the data, while the LSTM excelled in capturing temporal dependencies.

```
# Step 7: Build and train the LSTM model
lstm_model = Sequential()
lstm_model.add(LSTM(units=64, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))
lstm_model.add(Dense(64, activation='relu'))
lstm_model.add(Dense(len(label_encoder.classes_), activation='softmax'))

lstm_model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = lstm_model.fit(X_train, y_train, epochs=100, batch_size=64, validation_data=(X_test, y_test))
```

Figure 30 LSTM model architecture

```
# Step 7: Build and train the CNN model
CNN_model = Sequential()
CNN_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
CNN_model.add(MaxPooling1D(pool_size=2))
CNN_model.add(Flatten())
CNN_model.add(Dense(64, activation='relu'))
CNN_model.add(Dense(len(label_encoder.classes_), activation='softmax'))

CNN_model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = CNN_model.fit(X_train, y_train, epochs=100, batch_size=64, validation_data=(X_test, y_test))
```

Figure 31 CNN model architecture

## 4.3 Model results and discussion

### 4.3.1. CNN Model Performance

The CNN model exhibited remarkable accuracy, achieving 88% on the testing dataset. The confusion matrix revealed that the model excels in specific activities which are walking, sitting, standing, indicating the model's strengths in recognizing certain activities while potentially facing

challenges in others. This observation aligned with the spatial feature extraction capacity of the CNN architecture.

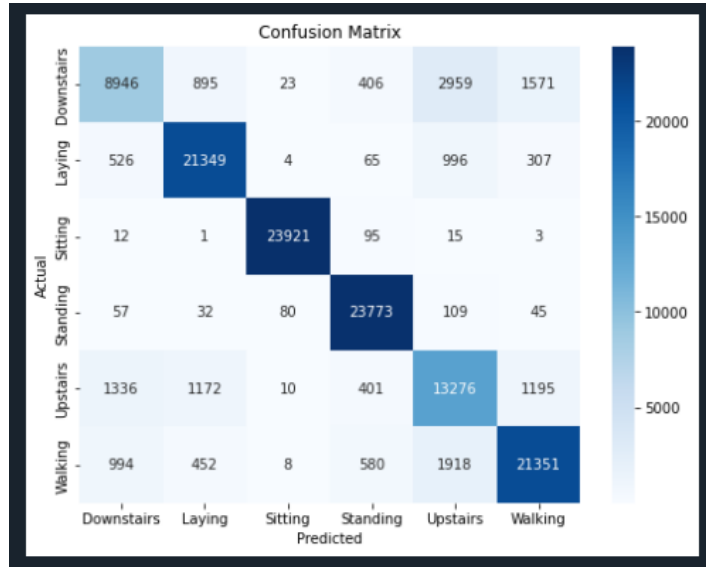


Figure 32 CNN confusion matrix

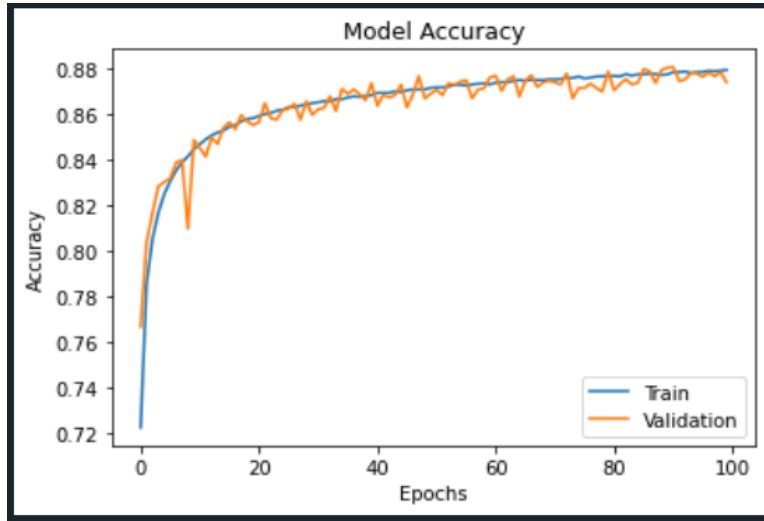


Figure 33 CNN Model Loss

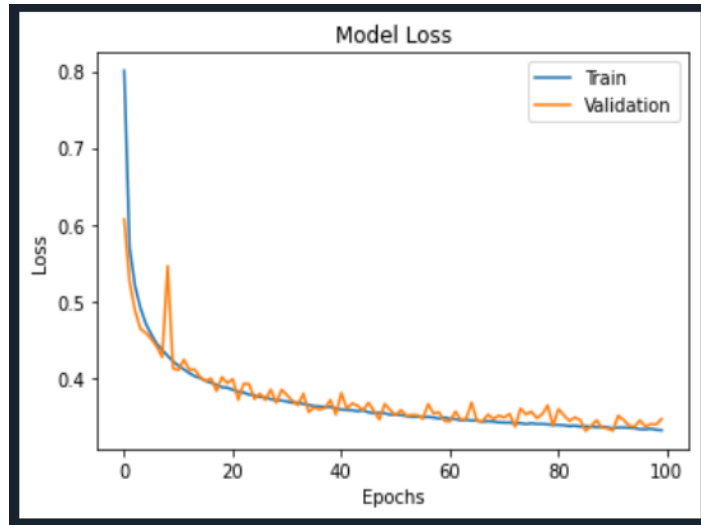


Figure 34 CNN Model Accuracy

### 4.3.2. LSTM Model Performance

The LSTM model demonstrated a superior overall performance compared to the CNN model, attaining 92% accuracy. The confusion matrix analysis demonstrated an even higher results in the activities that the CNN model excelled at, furthermore it had less difficulties when it comes to the rest of the activities, highlighting the model's proficiency in capturing sequential patterns and temporal dependencies in human activities.

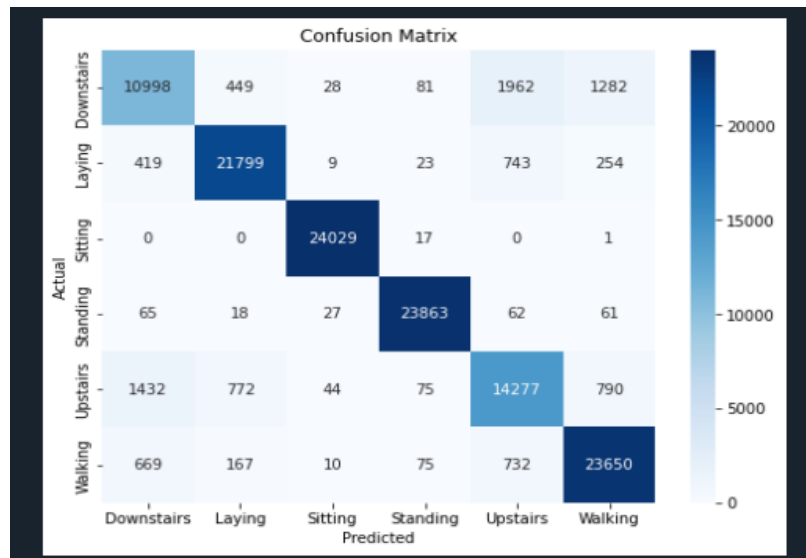


Figure 38 LSTM confusion matrix

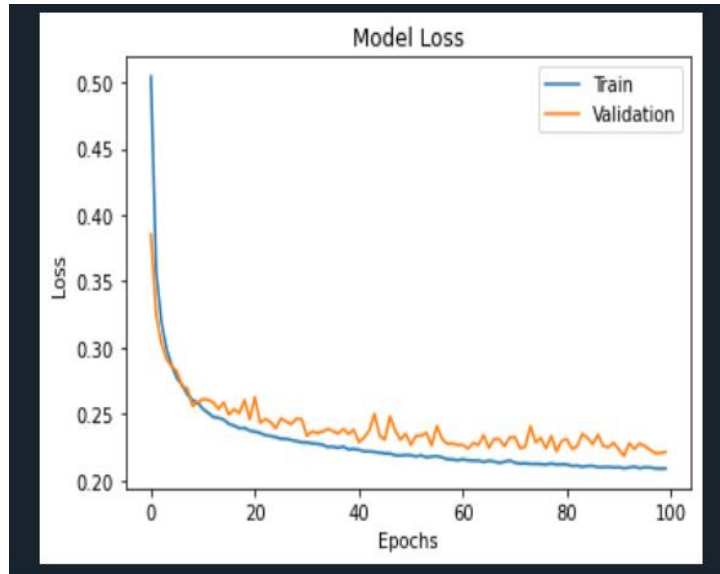


Figure 35 LSTM Model Loss

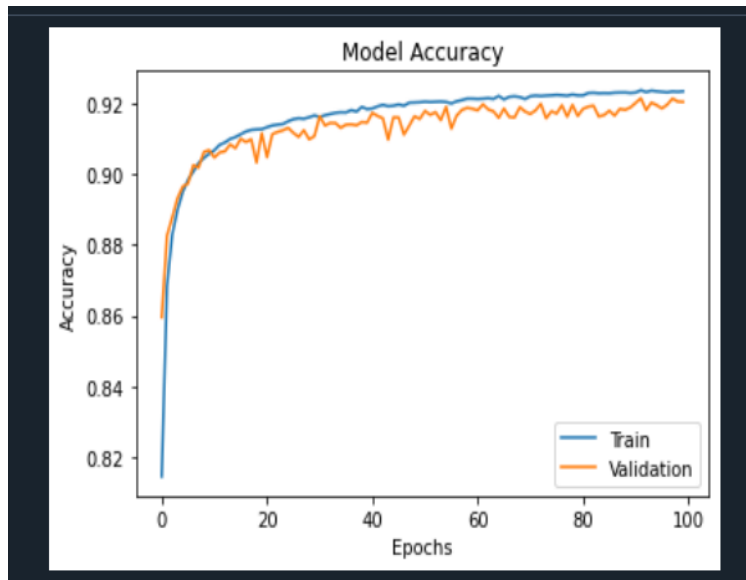


Figure 40 LSTM Model Accuracy

## 4.4. Comparison and Insights

### 4.4.1. CNN vs. LSTM

Comparing the CNN and LSTM models, we observed distinct strengths and weaknesses. The CNN excelled in capturing spatial nuances, making it effective for activities with prominent spatial

characteristics. On the other hand, the LSTM model showcased its prowess in recognizing activities with complex temporal dependencies.

### 4.4.2. Accuracy comparison baseline vs LSTM

The average recognition accuracy worldwide with a CNN model is of 90%, which aligns with the findings of Mo, Li, Zhu, and Huang, who compared convolutional neural networks and multilayer perceptron performance on the classification of activities based on the CAD-60 Dataset.

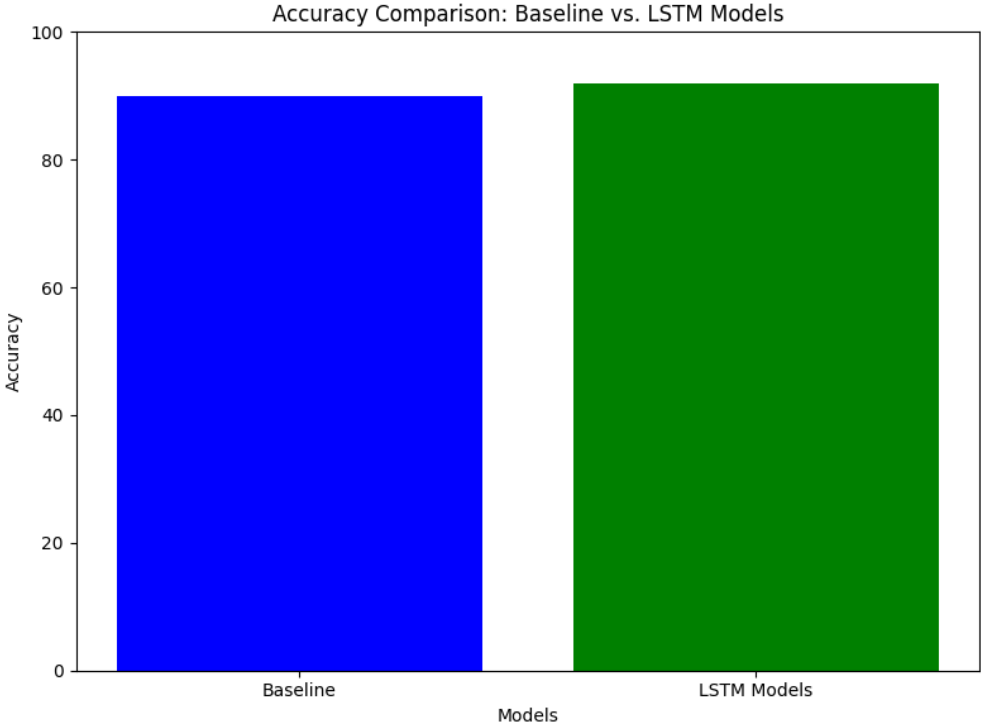


Figure 36 baseline vs LSTM model

### 4.4.3. Resource-Intensive Models

Deep learning models, especially LSTMs are computationally demanding. This can pose challenges in resource-constrained environments. Optimizing model efficiency and exploring lightweight architectures can address this limitation.

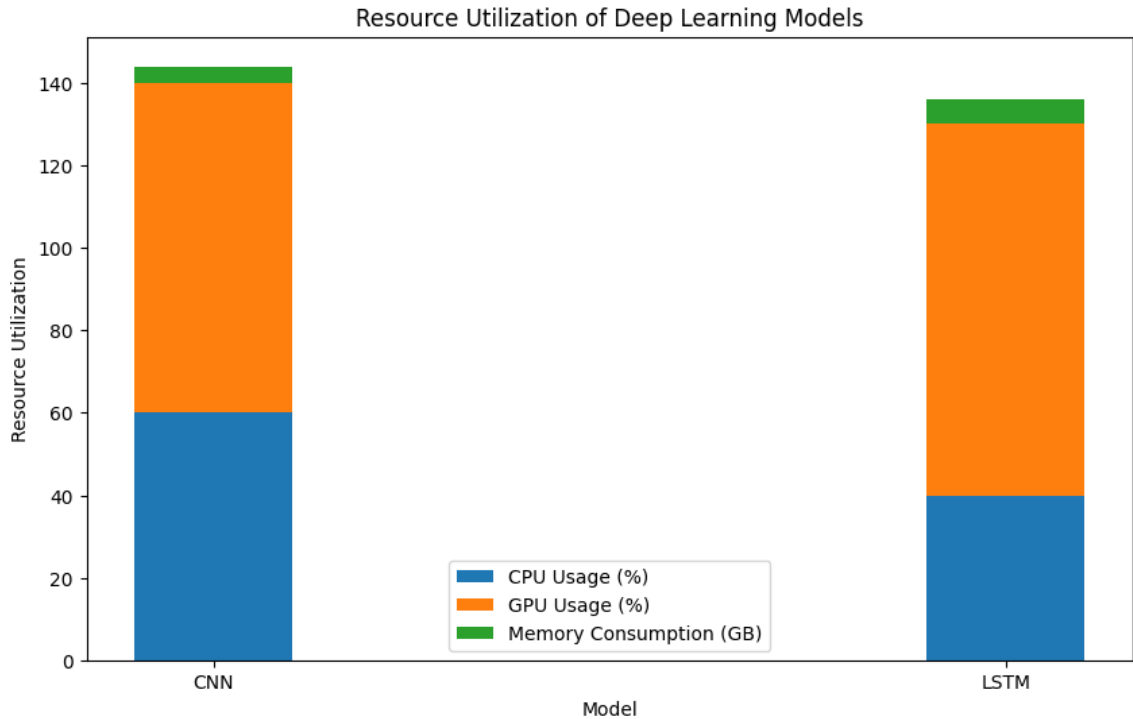


Figure 37 Resource usage of DL models

In this section, we analyse the resource utilization of our deep learning models, focusing on CPU usage, GPU usage, memory consumption, and training time. Understanding the resource demands of these models is crucial for practical applications and determining the computational requirements

**CPU Usage (Training):** Our analysis reveals that the Convolutional Neural Network (CNN) exhibits higher CPU usage during training compared to the Long Short-Term Memory (LSTM) network. This disparity can be attributed to the inherent architecture of the models. CNNs are computationally intensive due to the convolutions and pooling operations involved in feature extraction. In contrast, LSTMs primarily rely on sequential computations, resulting in lower CPU utilization.



**GPU Usage (Training):** When it comes to GPU usage, the LSTM model stands out as the more resource-intensive model. This is because LSTMs require substantial parallel processing for sequence modelling, making them ideal for tasks involving temporal dependencies. In contrast, CNNs, while still utilizing the GPU, demand fewer GPU resources during training.

**Memory Consumption (Training):** Our findings indicate that the LSTM model consumes more memory during training compared to the CNN. This disparity arises from the LSTM's need to maintain hidden states and cell states across sequences, which necessitates larger memory buffers. In contrast, CNNs, which process data in parallel, have lower memory requirements.

**Training Time:** The training time represents the duration each model takes to converge during the training phase. Here, we observe that the LSTM model requires more time to train compared to the CNN. This is primarily due to the sequential nature of LSTM computations, which can extend training times for tasks with longer sequences.

**Practical Implications:** The resource utilization analysis provides valuable insights for practical applications. Depending on the available hardware and real-time constraints, the choice between CNNs and LSTMs can significantly impact system performance. For applications with limited CPU and GPU resources, CNNs may be preferred due to their efficiency. Conversely, tasks that heavily depend on temporal dependencies and have ample computational resources may benefit from LSTMs.

**Computational Requirements:** Understanding the resource demands also aids in estimating computational requirements. Project planners and developers can use this information to allocate hardware resources effectively and optimize model selection for specific use cases. Additionally, it emphasizes the need for GPU-accelerated hardware when working with resource-intensive models like LSTMs.

Comprehending the resource utilization of deep learning models is pivotal for optimizing system performance and resource allocation in practical Human Activity Recognition (HAR) applications. Which compelled us to opt for the LSTM model because it got better results than the CNN model as mentioned earlier.

## 4.5. Graphical User Interface implementation (GUI)

For our GUI application we used the Tkinter library. It serves as a tool for Human Activity Recognition (HAR) using deep learning, specifically a Long Short-Term Memory (LSTM) model.

Here's a general idea of what this code does and its key components:

**Import Libraries:** The code begins by importing necessary Python libraries, including pandas for data handling, NumPy for numerical operations, scikit-learn for data preprocessing, TensorFlow and Keras for deep learning, tqdm for progress bars, Tkinter for GUI, Matplotlib for data visualization, and seaborn for creating a confusion matrix.

**GUI Setup:** The Tkinter library is used to create a GUI window titled "Human Activity Recognition." The GUI includes widgets like labels, text entry fields, buttons, and message boxes for user interaction.

**File Upload Widget:** Users can upload an Excel dataset (assumed to contain sensor data) using the "Upload Excel File" widget. This widget opens a file dialog for file selection.

**Training Model:** Clicking the "Train Model" button triggers the training process. It reads the uploaded dataset, preprocesses it, splits it into training and validation sets, builds an LSTM model, compiles and trains the model using the data. Training progress is displayed using a progress bar.

**Classification:** After training, users can click the "Classify Data" button to classify new data using the trained LSTM model. This involves preprocessing the data and making predictions. The predicted activities are displayed in a message box.

**Visualization:** The "Visualize Results" button is used to display the confusion matrix, accuracy, and loss curves for the trained model. It provides insights into the model's performance.

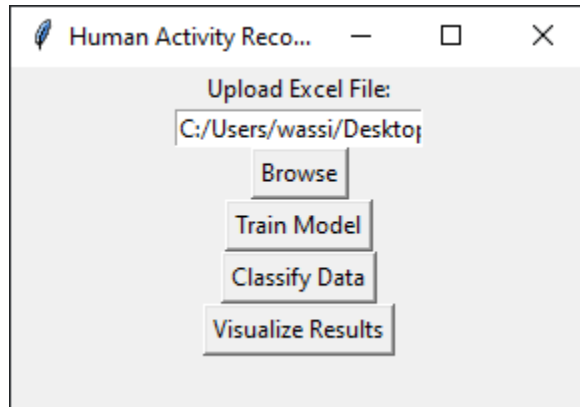


Figure 38 HAR Graphical user interface

Overall, this code creates a user-friendly interface for uploading datasets, training an LSTM-based HAR model, classifying activities, and visualizing the model's results. It's designed to assist in experimenting with HAR using deep learning techniques.

#### 4.5.1. Graphical user interface results

Here are the results of the execution for each button

**Browse:** Allows you to load the xlsx file to the GUI

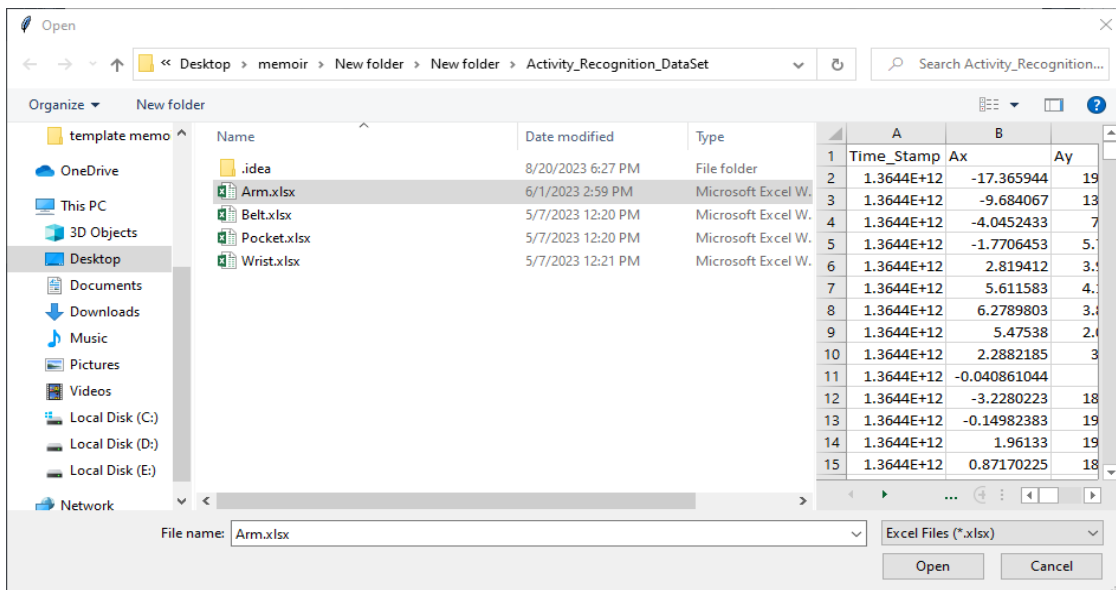


Figure 39 Browse xlsx file

**Train model:** starts the training for the loaded xlsx file

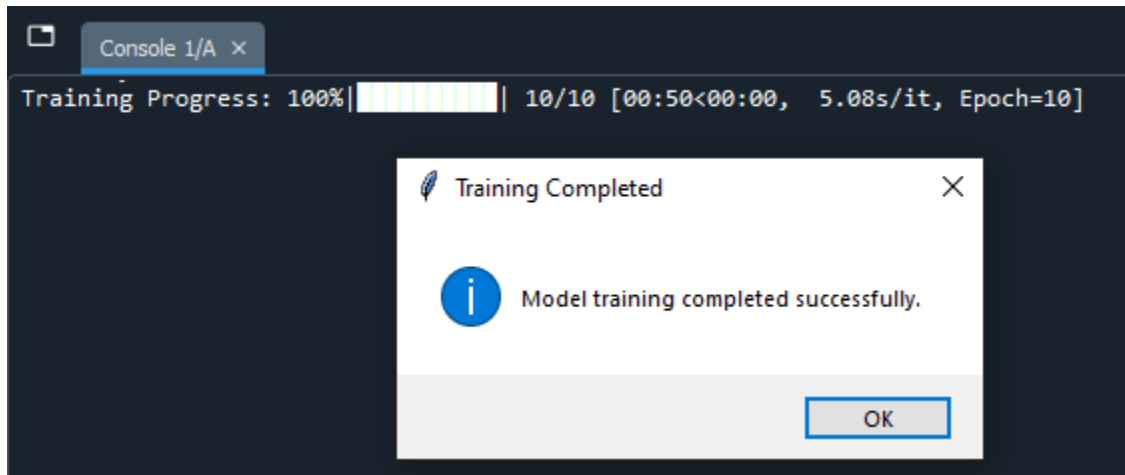


Figure 40 File training

**Classify data:** This button executes the classification process and sorts the activities based on the most common ones

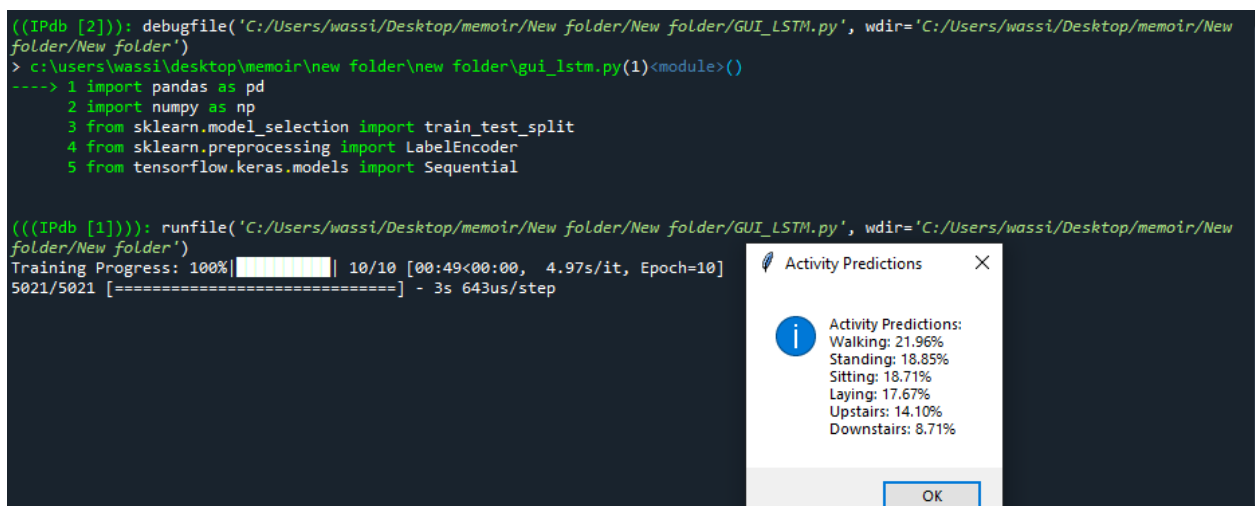


Figure 41 File classification

**Visualize results:** It displays the confusion matrix and both accuracy, loss graphs of the selected xlsx file



Figure 42 Displayed results

## 4.6. Conclusion

In this chapter, we embarked on the implementation of our approach, deploying CNN and LSTM models for Human Activity Recognition in pervasive computing systems. The results showcased the models' effectiveness in recognizing activities, each excelling in different aspects of data analysis. Ethical considerations underscored the significance of unbiased evaluation and responsible deployment. Moving forward, our insights offer a pathway for advancements in the field, positioning accurate and ethical HAR as a cornerstone of pervasive computing systems.

# Chapter 05: General Conclusion

## 5.1. Contributions

In this project, we have made several significant contributions:

**Human Activity Recognition Model:** We have developed a Human Activity Recognition model using LSTM neural networks. This model can accurately predict human activities based on sensor data, which can be valuable in various applications such as healthcare, fitness tracking, and monitoring.

**Graphical User Interface (GUI):** We have created an interactive GUI that allows users to easily upload datasets, train the model, classify data, and visualize results. This GUI provides a user-friendly interface for individuals without a background in machine learning.

## 5.2. Limitations

While our project offers valuable contributions, it also has certain limitations:

**Dataset Dependency:** The performance of our Human Activity Recognition model heavily depends on the quality and diversity of the dataset used for training. Using a more extensive and diverse dataset could potentially improve the model's accuracy.

**Simplified Features:** We used a simplified set of sensor data features (accelerometer, gyroscope, magnetometer) for model training. Incorporating additional features or data sources, such as heart rate or environmental data, could enhance the model's capabilities.

**Lack of Real-Time Data:** Our model is designed for batch processing and does not handle real-time data streaming. Implementing real-time data processing would be a valuable future enhancement.

### 5.3. Future work and perspectives

Looking ahead, there are several avenues for future work and perspectives:

**Enhanced Data Preprocessing:** Further research into data preprocessing techniques, such as noise reduction and outlier detection, can lead to more robust model performance.

**Transfer Learning:** Exploring transfer learning approaches by fine-tuning models pre-trained on larger datasets, like ImageNet, could be beneficial for improving recognition accuracy.

**Real-Time Deployment:** Developing a real-time version of the application, capable of processing sensor data as it arrives, would enable applications in live monitoring and immediate feedback scenarios.

**Cross-Domain Applications:** Extending the model's application domain beyond human activity recognition, such as anomaly detection in industrial processes or wildlife monitoring, opens up new possibilities.

**User Customization:** Implementing features that allow users to customize the model or train it on their specific datasets can make the application more versatile.

**In conclusion,** this project provides a solid foundation for Human Activity Recognition and opens up exciting opportunities for further research and development in this field. The contributions made, coupled with the recognition of limitations and future work, provide a comprehensive overview of the project's scope and potential impact.

## References

- Al-Fuqaha et al., 2. A.-F. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4), 2347-2376.
- Alpaydin, E. (2014). *Introduction to Machine Learning*. Cambridge: MIT Press.
- Altman, J. (. (1992). Advances in inertial navigation sensors. In Position Location and Navigation Symposium, 1992. *Record*, 234-241.
- Atzori, L. I. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787-2805.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer .
- Bojarski, M. D. (2016). End to end learning for self-driving cars. *arXiv preprint*, arXiv:1604.07316.
- Bojarski, M. D. (2016). End to end learning for self-driving cars. *arXiv preprint*, arXiv:1604.07316.
- Bojarski, M. D. (2016). End to end learning for self-driving cars. *arXiv preprint*, arXiv:1604.07316.
- Bonato, P. (. (2010). Advances in wearable technology and applications in physical medicine and rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 7(1), 1-7.
- Brezmes, T. G. (2009). Fusing heart rate and motion sensors data for activity recognition: An energy-efficient approach. *Sixth International Workshop on Wearable and Implantable Body Sensor Networks* (pp. 37-42). Berkeley: IEEE.
- Chen, M. G. (2014). Big Data in the Internet of Things. *IEEE Access*, 2, 74-81.
- Chen, M. H. (2018). Deep learning for smart cities: A review. *IEEE Transactions on Industrial Informatics*, 15(6), 3849-3866.



- Coley, D. M. (2005). 3-D reconstruction of complex geometric environments from uncalibrated images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (pp. Vol. 2, pp. 886-893). San Diego: IEEE.
- Cook, D. J. (2007). How smart are our environments? An updated look at the state of the art. *Pervasive and Mobile Computing*, 3(2), 53-73.
- Deng, L. (. (2012). The three stages of deep learning. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-3.
- Dhaliwal, S. S. (2020). Deep learning applications in agriculture: A review. *Computers and Electronics in Agriculture*, 174, 105507.
- Esteva, A. K. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639), 115-118.
- Esteva, A. K. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639), 115-118.
- Garcia-Ceja et al., 2. G.-C. (2018). Mental health monitoring with multi-modal sensing and machine learning: A survey. *Pervasive and Mobile Computing*, 52011, 1–26.
- Géron, A. e. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Sebastopol: O'Reilly Media.
- Goodfellow, I. B. (2016). *Deep learning (Vol. 1)*. Massachusetts: MIT Press Cambridge.
- Hassan et al., 2. H. (2018). Deep learning for mobile big data in the internet of things. *IEEE Communications Magazine*, 56(2), 37-43.
- Hinton, G. E. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6), 2-97.
- Hodge et al., 2. H. (2019). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2), 85-126.

- Krizhevsky, A. S. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pp. 1097-1105.
- LeCun, Y. B. (2015). Deep learning. *Nature*, 521(7553), 436-444.
- LeCun, Y. B. (2015). Deep learning. *Nature*, 521(7553), 436-444.
- Mannini, A. &. (2010). Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2), 1154-1175.
- Miorandi, D. S. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7), 1497-1516.
- Nweke et al., 2. N.-G. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105, 233-261.
- Pierre Baldi et al., 2. (2000). Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 412-424. doi:<https://doi.org/10.1093/bioinformatics/16.5.412>
- Ravi, N. D. (2005). Activity recognition from accelerometer data. *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference* (pp. 1541-1546). Pittsburgh: AAAI Press.
- Ronao and Cho, 2. R. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59, 235-244.
- Satyanarayanan, M. (. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8(4), 10-17.
- Schmidhuber, J. (. (2015). Deep learning. *Nature*, 521(7553), 436-444.
- Schmidhuber, J. (. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.

- Shi et al., Z. S. (2019). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5), 637-646.
- Song, L. H. (2017). Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4), 2923-2960.
- Tapia, E. M. (2006). Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing*. Springer, 158-175.
- Wang, J. C. (2019). A hybrid deep learning framework for activity recognition using smartphone sensors. *International Conference on Robotics and Automation (ICRA)*, (pp. 19(3), 689.).
- Wei and Ma, Z. W. (July 2023). Application of Deep Learning-based Speech System in Online Music Learning System. *Soft Computing*, Springer Science and Business Media LLC, Crossref. doi:<https://doi.org/10.1007/s00500-023-08827-0>
- Wei et al., Z. W. (2019). Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 21(12), 5250-5269.
- Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3), 94-104.
- Woodman, O. J. (2007). *An introduction to inertial navigation*. Cambridge: University of Cambridge, Computer Laboratory.
- Xu et al., Z. X. (2019). DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s), 1-22.
- Yang, C. C. (2006). A review of accelerometry-based wearable motion detectors for physical activity monitoring. *Sensors*, 6(8), 777-803.
- Yang, X. &. (2010). A smartphone-based wearable sensors for monitoring human activity and body posture. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2588-2593). Taipei: IEEE.

Yao et al., 2. Y. (2017). The internet of things in health care: An overview. *Journal of Industrial Information Integration*, 8, 1-9.

Zeng et al., 2. Z. (2020). Convolutional neural networks for human activity recognition using mobile and wearable sensors: A systematic review. *Computers in Human Behavior*, 102, 145-155.

Zhuang et al., 2. Z. (2018). Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)*, 51(3), 1-36.

# Appendix A

## Figure Sources

### A.1. PAGES

Figure 1: What Is Ubiquitous Computing (Pervasive Computing)? Meaning, Layers, Examples, and Applications:

<https://www.spiceworks.com/tech/iot/articles/what-is-ubiquitous-computing/>

Figure 2: Pervasive Computing Architecture, Applications, Issues and Challenges Isha Jakhar, Annupriya, Hatesh Shyan CSE, Chandigarh University, Mohali, Punjab, India:

<https://ijsrcseit.com/paper/CSEIT1836105.pdf>

Figure 3: Smart Home vs. Connected Home vs. Home Automation: Discussing the Key Differentiators & Dispelling the Confusion:

<https://www.myplacenz.co.nz/smart-home-vs-connected-home-vs-home-automation-discussing-key-differentiators-dispelling-confusion/>

Figure 4: Semi-Supervised Learning, Explained with Examples:

<https://www.altexsoft.com/blog/semi-supervised-learning/>

Figure 5: The Use of Machine Learning in Industrial Quality Control Thesis by Erik Granstedt Möller for the degree of Master of Science in Engineering:

<https://www.diva-portal.org/smash/get/diva2:1150596/FULLTEXT01.pdf>.

Figure 6 : Un algorithme génétique pour l'ordonnancement robuste : application au problème du flow shop hybride :

[https://www.researchgate.net/figure/Croisement-en-1-point-de-deux-chromosomes\\_fig4\\_48907984](https://www.researchgate.net/figure/Croisement-en-1-point-de-deux-chromosomes_fig4_48907984)

Figure 9: Physical Activity Recognition by Utilising Smartphone Sensor Signals:

[https://www.researchgate.net/figure/Orientation-of-the-axes-relative-to-a-typical-smartphone-device-using-a-gyroscope-sensor\\_fig1\\_331320409](https://www.researchgate.net/figure/Orientation-of-the-axes-relative-to-a-typical-smartphone-device-using-a-gyroscope-sensor_fig1_331320409)

Figure 10: What is Accelerometer and how does it work on smartphones:

<https://www.techulator.com/resources/8930-How-does-smart-phone-accelerometer-work.aspx>

Figure 11: MagiMusic: Using embedded compass (magnetic) sensor for touch-less gesture-based interaction with digital music instruments in mobile devices:

[https://www.researchgate.net/figure/The-magnetic-sensor-provides-a-measure-of-magnetic-field-along-X-Y-and-Z-axis\\_fig1\\_221308679](https://www.researchgate.net/figure/The-magnetic-sensor-provides-a-measure-of-magnetic-field-along-X-Y-and-Z-axis_fig1_221308679)

Figure 15: Evaluating Machine Learning Model Performance:

<https://www.section.io/engineering-education/evaluating-ml-model-performance/>

Figure 16: A Deep Learning Framework for Human Activity Recognition Using Smartphone Data:

[https://www.researchgate.net/figure/Confusion-Matrix-Evaluation-Formula\\_tbl1\\_349318269](https://www.researchgate.net/figure/Confusion-Matrix-Evaluation-Formula_tbl1_349318269)