

Faculty of Computing & Information



Technology



FINAL YEAR PROJECT - THESIS

FOR CONSIDERATION UNDER

BACHELOR OF INFORMATION TECHNOLOGY

(BS) PROGRAMME

PROJECT TITLE

**EATING ACTIVITY DETECTION USING WEARABLE SENSORS**

SUPERVISOR:

Dr. Muhammad Adeel Nisar

June 19, 2023

(Date of Submission)

This research project is a fulfillment of the requirements for the reward of BS in Information Technology.

Table of Contents

[1.](#_30j0zll) **Introduction**: 1

[1.1. Problem Statement and Motivation: 4](#_1fob9te)

[1.2. Related Work:](#_3znysh7) 6

[1.3. Own Contributions:](#_2et92p0) 11

[1.4. Overview:](#_17dp8vu) 12

[2.](#_3rdcrjn) **Basics:** 14

[2.1. Data Description: 1](#_26in1rg)4

[2.2. Wearable Devices and Sensors: 1](#_lnxbz9)5

[2.3. Data Preprocessing: 1](#_3as4poj)6

[2.3.1. Hand Crafted Feature Engineering Process: 1](#_1pxezwc)7

[2.3.2. Deep Learning Models: 1](#_49x2ik5)9

[2.4. Neural Networks: 1](#_2p2csry)9

[3.](#_147n2zr) **Methods:** 21

[3.1 Hand-Crafted Features](#_3o7alnk) 22

[3.2. Multi-Layer Perceptron 2](#_23ckvvd)5

[3.3. Convolutional Neural Network 2](#_ihv636)6

[3.4 Long-Short Term Memory 2](#_32hioqz)7

[3.4. Bi-Directional Long-Short Term Memory 2](#_1hmsyys)9

[3.5. Autoencoder](#_41mghml) 31

[3.5. Convolutional Autoencoder](#_2grqrue) 32

[3.6. Convolutional Long-Short Term Memory](#_vx1227) 33

[4. **Experimental Results:**](#_3fwokq0) 34

[4.1. Hand Crafted Features (HC)](#_nmf14n) 34

[4.2. Multi-Layer Perceptron (MLP):](#_37m2jsg) 34

[4.3. Convolutional Neural Network 3](#_1mrcu09)6

[4.4. LSTM 3](#_46r0co2)6

[4.5. Bi-LSTM 3](#_2lwamvv)7

[4.6. Convolutional LSTM 3](#_111kx3o)7

[4.7. Results 3](#_3l18frh)7

[5. **Discussion:** 3](#_206ipza)8

[6. **Conclusion:**](#_2zbgiuw) 40

[7. **Future Work:**](#_1egqt2p) 41

[References](#_3ygebqi) 42

EATING ACTIVITIES DETECTION USING

WEARABLE SENSORS

BITF19M004 - Hadia Waseem, BITF19M036 - Wajeeha Fatima,

BITF19M017 – Muhammad Ghufran Ali, BITF19M026 – Muhammad Umair, BITF19M029 – Muhammad Badar

# Abstract

This paper explores the use of wearable sensors and deep neural network models for eating activities detection. While the study primarily focuses on training multiple deep neural network models using the publicly available CogeAge dataset [[8](https://www.mdpi.com/1424-8220/20/12/3463)], acquired from wearable sensors including smart glasses, a smartwatch, and a smartphone, the ultimate goal is to apply these models to eating activities datasets. The research involves feature engineering through handcrafted feature extraction methods and feature learning using various deep learning models. Although specific experiments on eating activities detection are not conducted, the trained models demonstrate the potential for accurately identifying and monitoring eating behaviors. The findings contribute to the advancement of wearable sensor-based activity recognition and pave the way for personalized interventions and improved health outcomes in the domain of eating activities detection.

# Introduction:

The rapid development and widespread use of wearable sensors have opened up new avenues for research in the domain of eating activities detection. With the growing concern about sedentary lifestyles and unhealthy eating habits, there is a pressing need for innovative solutions that can accurately monitor and analyze dietary behaviors. The integration of wearable sensors with advanced technologies, such as deep neural networks, offers a promising approach to tackle this challenge.

Understanding eating activities goes beyond simply recognizing when someone is consuming food. It involves capturing intricate details, such as eating speed, portion sizes, eating patterns, and even emotions related to eating. By leveraging wearable sensors, which can be conveniently worn on the body, researchers can collect real-time data to gain insights into individuals' eating behaviors in various contexts and settings.

Accurate detection and analysis of eating activities can have a profound impact on promoting healthier lifestyles and addressing the complex interplay between nutrition and health. It enables personalized interventions, facilitates dietary management, and provides valuable feedback for individuals striving to make positive changes in their eating habits. Moreover, it offers researchers and healthcare professionals a wealth of data to identify patterns, correlations, and risk factors associated with nutrition-related diseases, ultimately leading to more effective preventive measures and interventions.

However, detecting eating activities poses unique challenges compared to other forms of activity recognition. Unlike activities that involve continuous body movements, such as walking or running, eating activities involve subtle gestures and motions that can be challenging to capture accurately. The diverse nature of eating behaviors further complicates the task, as they can vary greatly between individuals and cultures. Additionally, the social and environmental aspects of eating, such as eating in groups or in noisy environments, add complexity to the detection process.

To address these challenges, our research focuses on training multiple deep neural network models specifically designed for eating activities detection. Deep neural networks have shown remarkable capabilities in learning complex patterns and representations from large-scale datasets. By leveraging existing human activities recognition datasets, we can leverage the knowledge gained from broader activity recognition tasks to enhance the accuracy and robustness of our models.

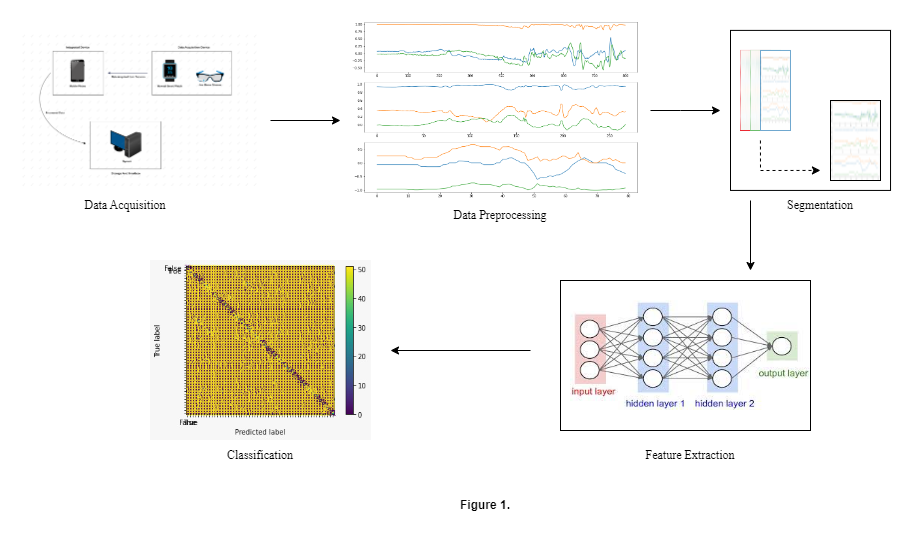
The primary focus of this study lies in training and fine-tuning these DNN models, utilizing a rich human activities recognition dataset as a basis. By harnessing the knowledge acquired from human activities recognition, the research aims to establish robust models capable of effectively detecting various activities, including eating. This approach provides a practical and efficient solution, as the models trained on the broader human activities dataset can be readily applied to eating activities datasets with minimal modifications.

The experimentation and analysis are performed on a publicly available dataset known as CogeAge, which encompasses a wide range of atomic activities. This dataset was meticulously collected using wearable sensors, including smart glasses, a smartwatch, and a smartphone. The integration of multiple sensors enables a comprehensive capture of body, hand, and head movements, ensuring a holistic approach to activity recognition. Using the CogeAge dataset provides a robust foundation for training and evaluating the performance of the deep neural network models in the context of various activities, including eating. [[8](https://www.mdpi.com/1424-8220/20/12/3463)]

Additionally, the wearable sensor setup from which CogeAge dataset is extracted which is used in this research consists of three distinct devices: the [LG G5 smartphone](https://www.lg.com/us/cell-phones/lg-VS987-Titan-g5), [Huawei watch](https://consumer.huawei.com/en/wearables/?ic_medium=hwdc&ic_source=corp_header_consumer), and [JINS MEME](https://jinsmeme.com/en/) glasses. These devices capture crucial movement data, including body, hand, and head motions, through a variety of sensory modalities such as accelerometers, gyroscopes, and magnetometers. This multi-sensor setup provides a comprehensive and dynamic view of the subject's movements during different activities, further enhancing the accuracy and granularity of the detection process.

To achieve accurate and reliable detection, this research employs a comprehensive pipeline encompassing various stages, including feature engineering and feature learning. Figure 1. Illustrates predefined pipeline or steps followed in complete research.

* **Data Acquisition:** The success of eating activities detection relies on the careful acquisition and processing of data obtained from wearable sensors. In this research, the focus is primarily on leveraging the existing CogeAge dataset for human activity recognition. Therefore, data acquisition, as typically understood, is not performed. Instead, the choice and setup of wearable sensors become paramount to ensure the accurate recognition of eating activities while considering potential implementation-related constraints.
* **Data Preprocessing:** During the data pre-processing stage, various operations are applied to make the collected data suitable for further analysis; normalization, de-noising, sampling etc. are essential techniques employed to make the dataset suitable for further proceedings.
* **Segmentation:** It plays a critical role in identifying specific segments within the dataset that contain relevant information about the activities to be recognized. By filtering out irrelevant data and extracting smaller segments, the quantity of data processed in subsequent stages of activity recognition is reduced. This step is particularly important due to potential hardware constraints that limit the amount of data processed at each time step. However, the dataset we obtained was already segmented so we didn’t perform this step.
* **Feature extraction**: It follows segmentation and involves computing values that abstract each data segment into a representation that is highly relevant to the associated activity. This step aims to capture high-level representations to ensure the generalization capacity of the eating activities detection system, especially when applied to large-scale datasets like CogeAge.
* **Classification:** The classification stage focuses on constructing a classifier using the extracted features. The classifier's role is to establish clear boundaries in the feature space, enabling the differentiation between different activities. By establishing rules based on the computed features, the classifier can effectively distinguish between segments associated with one class and segments associated with other activities.



**Figure 1.** illustrates the sequential flow of our study. Starting with data acquisition, we gather the necessary information for our study. The acquired data then undergoes preprocessing, where it is cleaned and prepared for further analysis. Subsequently, the dataset is segmented into meaningful subsets to enable focused examination. Next, feature extraction techniques are applied to identify important patterns or characteristics within each segment. Finally, the study concludes with the classification step.

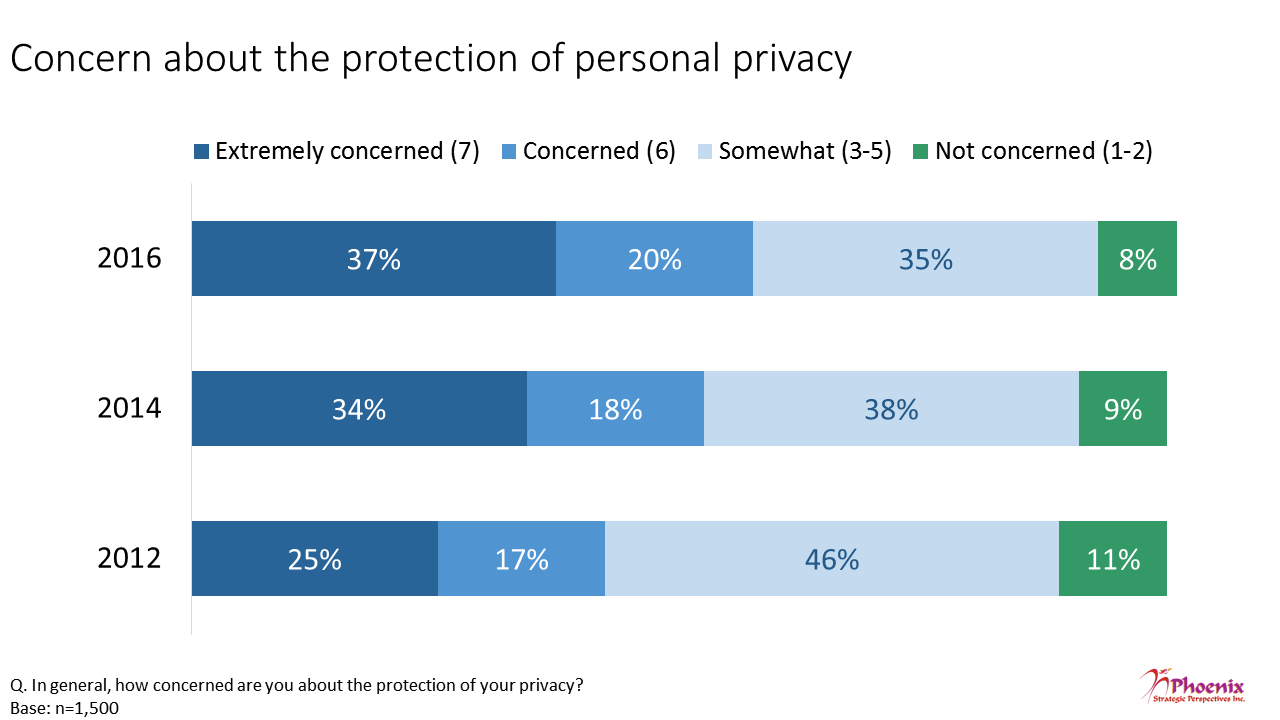
Feature engineering involves the extraction of a rich set of handcrafted features that capture the intricate nuances of different activities. These features serve as essential inputs for the subsequent feature learning and classification process. Feature learning, which encompasses the utilization of deep learning models, focuses on leveraging the extracted features to learn complex patterns and representations in the data. Various architectures, including convolutional neural networks (CNN), multi-layer perceptron (MLP), auto-encoders, LSTM networks, and hybrid models, are trained to analyze the most effective architecture for activity recognitions which will then eventually work for eating activities as well.

The feature engineering process involves the extraction of a total of 17 carefully crafted features, capturing crucial aspects related to different activities. These features are specifically tailored to provide rich representations of the behaviors, enabling the models to discern between various aspects of the activities with improved accuracy. The subsequent feature learning and classification stage involves training the deep learning models on the extracted features and utilizing classifiers such as Support Vector Machines (SVM) to accurately classify the activities.

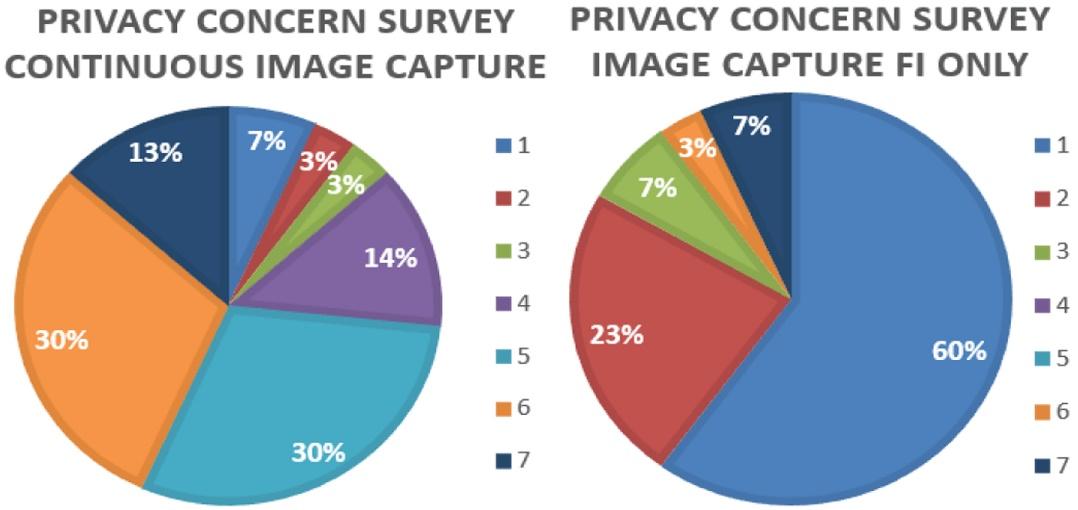
The subsequent sections of the paper delve into each step of the research methodology, providing a comprehensive and detailed exploration of the approaches employed and the results obtained.

## Problem Statement and Motivation:

The detection and analysis of eating activities have gained significant importance in promoting healthier lifestyles and addressing nutrition-related health issues. However, many current approaches often rely on camera-based methods for recognizing eating behaviors and daily activity recognitions, which raise privacy concerns and may not be suitable for real-time monitoring. In a literature review we conducted, we came across many articles where cameras were used for either image capturing or video recordings for accurately monitoring of eating gestures. However, when they conducted a survey many people finds it discomforting and questions over privacy while using cameras. To overcome these limitations, this research aims to develop a robust and privacy-conscious approach (models) for detecting activities using datasets extracted only from wearable sensors without use of cameras. Following are the statistical results containing Figure 2. And Figure 3. From researches [[1](#_2dlolyb)], [[2](#_2dlolyb)] that also conducted surveys about privacy concerns using cameras:



**Figure 2.** Presents a graph showcasing the varying levels of concern regarding the protection of personal privacy over the years. The data is categorized into four distinct categories: "Extremely Concerned," "Concerned," "Somewhat Between," and "Not Concerned."



**Figure 3.** Presents a graph showcasing the varying levels of concern regarding the protection of personal privacy over continuous image capture and image capture FI only.

Another matter of concern arises while using cameras in natural settings, it’s a normal human psychology that when they know that they have been recorded it automatically deviates their behavior and they do not reflect their natural actions. They become more self-conscious as a result it can also affect the gestures and data recorded.

After analyzing many researches, we ponder our need of training models on wearable sensors. Although, many works have been done in this area already but our research stands out in the context that we trained multiple models and their hybrid version to fit in any datasets related to activity recognitions, primarily eating activities. This approach eliminates the need for extensive data acquisition specifically for eating activities, saving time and resources. Our trained models are not limited to human activity recognition alone. They can be effectively applied to various datasets, including the detection and analysis of eating activities.

The motivation behind this research lies in the potential impact of accurately detecting eating activities using wearable sensors. Such a system can provide personalized and improved health monitoring, and effective dietary management. The outcomes of this can have significant implications for promoting healthier lifestyles, combating nutrition-related health issues, and ensuring individuals' privacy.

## 1.2. Related Work:

The first important step in our research was conducting an extensive literature review to gain insights into the existing body of work related to eating activity detection using wearable sensors. This comprehensive review allowed us to understand the progress made in the field, the methodologies employed, and the challenges faced by previous researchers. Through our analysis, we identified various studies that explored the use of different types of wearable sensors and examined the compatibility of these sensors with different modes of data collection and the specific advantages they offered in detecting and monitoring eating behaviors. Furthermore, we investigated the range of models and algorithms utilized in previous research, including traditional machine learning approaches and deep learning architectures. This examination allowed us to identify the strengths and limitations of these models, enabling us to design an approach that applies the best practices and overcomes the existing limitations.

Below is a detailed description of the work we have reviewed so far. For a concise overview of the related work, here is the link [[3](#_2dlolyb)]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr** | **Title of the paper** | **Repository** | **List of activities** | **wearable sensors** | **Feature generation?** | **Machine learning algorithm** | **accuracy/F1?** |
| **1** | Automatic, wearable-based, in-field eating detection approaches for public health research: a scoping review | pubMed | <https://www.nature.com/articles/s41746-020-0246-2/tables/1> | <https://www.nature.com/articles/s41746-020-0246-2/tables/1> | not specified | No | The most frequently reported evaluation metrics were Accuracy (N = 12) and F1-score (N = 10). https://www.nature.com/articles/s41746-020-0246-2/tables/1 |
| 2 | EarBit: Using Wearable Sensors to Detect Eating Episodes in Unconstrained Environments | pubMed | Focused on detection of chewing activity including number of activites related to eating and drinking not mentioned | Earbits(designed and experimental)= VCNL4020,Bose IE2 earbud, 9 Degree-of-Freedom IMU, microphone, HBS-760 Rymemo Bluetooth, teensy 3.2 microcontroller,gyroscope and .. video camera (GoPro Hero) | Statistical Features crafting and for feature selection = sequential forward floating algorithms(SFFS) | Random Forest, leave-one-user-out cross validation technique | Accuracy = 93%, F1-Score = 80.1% |
| 3 | Smartwatch-Based Eating Detection: Data Selection forMachine Learning from Imbalanced Data with Imperfect Labels | pubMed | food intake gestures, cutlery detection, meal taken | Mobvoi TicWatch S, accerlerometer, gyroscope | TFF plus 1.python package tsfresh 2. powerful spectral density | Hidden Markov Model (HMM), leave-one-recording-out (LORO)cross-validation technique | precision = 0.85 and recall = 0.81 |
| 4 | Detection of Activities by Wireless Sensors for Daily Life Surveillance: Eating and Drinking | pubMed | eating and drinking | 3D accelerometer inside Mobile Cardiac Monitor | Euler Angle, kalman filter | Hierarchical Temporal Memory Algorithm, monto carlo runs | success rate with raw data= 84%-86% and success rate with features =86%-88% |
| 5 | Automatic Ingestion Monitor Version 2 - A Novel Wearable Device for Automatic Food Intake Detection and Passive Capture of Food Images | pubMed | eating activity( self reporting by pressing the pedals for diff b/w solid, liquid food plus beverages), chewing, sound in range | Automatic Ingestion Monitor, AIM-2 (experimental designed) | Statistical Features crafting and for feature selection = minimum Redundancy and Maximum Relevance, Forward Feature Selection (FFS) | Support Vector Machine(linear) plus leave-one-subject-out cross-validation, Gaussian smoothing kernel(for image processing), sensor fusion classifiers | F1-score = 81.8 ± 10.1%, accuracy = 82.7%, |
| 6 | NeckSense: A Multi-Sensor Necklace for Detecting Eating Activities in Free-Living Conditions | pubMed | chewing, feeding gestures, lean forward motion | neckSense(experimental designed) plus camera (QSD-722 camera and other things(experimental designed)) | PPA, LPSA, AEPSA, SFE | Friedman’s Gradient Boosting Model, softmax objective, DBSCAN algorithm | F1-score of 81.6% in exploratory study and F1-score of 77.1% for episodes in all-day-long free-living setting |
| 7 | Monitoring eating habits using a piezoelectric sensor-based necklace | pubMed | swallowing, solid and liquid food classification | RFDuinoMicrocontroller, BTLE Transceiver, PiezoelectricVibration Sensor, Coin Battery(CR2032) | swallowing detection algo, peak detection algo | Naive Bayes classifier | solid and liquid foods detection has F-measure of 0.837 and 0.864. The percentage accuracy for chips, water,  and sandwiches were 85.3%, 81.4%, and 84.5%, respectively. |
| 8 | Wearable Sensor-Based Detection of Eating Activities: A Review | Semantic  Scholar | Eating, Walking,Running,Sitting,Standing | Accelerometers, gyroscopes,magnetometers,heart rate monitors,electrodermal activity (EDA) sensors, electroencephalogr-aphy (EEG) | Hand-crafted features | Support vector machines (SVMs): 85% accuracy  decision trees: 80% accuracy  random forests: 82% accuracy | 85% |
| 9 | Towards Continuous Detection of Eating Activities Using Wearable Sensors | Semantic  Scholar | Eating, Walking, Running, Sitting, Standing,  Talking, Laughing, Using a Phone | Accelerometers,gyroscopes,magnetometer  heart rate monitors,EDA sensors | Hand-crafted features | SVMs: 87% accuracy  decision trees: 83% accuracy | 87% |
| 10 | A Deep Learning Approach for Wearable Sensor-Based Eating Activity Recognition | Semantic  Scholar | Eating,Walking, Running,Sitting,Standing  Talking,Laughing Using a Phone,Cooking  Cleaning,Playing Sports | Accelerometers,gyroscopes,magnetometers  heart rate monitors,EDA sensors | Convolutional Neural  Network (CNN) | Convolutional neural networks  (CNNs): 95% accuracy | 95% |
| 11 | A Multimodal Approach for Wearable Sensor-Based Eating Activity Recognition | Semantic  Scholar | Eating,Walking,Running,Sitting,Standing  Talking,Laughing Using a Phone,Cooking  Cleaning,Playing Sports Using a Computer | Accelerometers,gyroscopes,magnetometers  heart rate monitors,EDA sensors, EEG | Recurrent Neural  Network (RNN) | Recurrent neural networks (RNNs): 93% accuracy | 93% |
| 12 | Wearable Sensor-Based Detection of Eating Activities for Weight Loss Management | Semantic  Scholar | Eating,Walking,Running,Sitting,Standing  Talking,Laughing Using a Phone,Cooking  Cleanin ,Playing Sports Using a Computer | Accelerometers,gyroscopes  magnetometers,heart rate monitors  EDA sensors, EEG Sensors | Convolutional Recurrent  Neural Network (CRNN) | Convolutional recurrent neural networks (CRNNs): 97% accuracy | 97% |
| Note: TTF = Time-Frequency Features generated using auto-correlation function and statistical feature generation method plus 1. python package tsfresh(this package allows general-purpose time-series feature extraction) 2. powerful spectral density(frequency domain features), continuous wavelet transfer (CWT for Time Series) Feature Selection = Leave-One-Subject-Out, ENN . | | | | | | | |
| PPA = Prominence-based Peak-finding Algorithm | | | | | | | |
| LPSA = Longest Periodic Subsequence Algorithm | | | | | | | |
| AEPSA = absolute error periodic subsequence algorithm (segmentation | | | | | | | |
| SFEM = statistical-based features extraction method. | | | | | | | |

Bedri et al. [[4](#_2dlolyb)] introduced EarBit, a wearable system that leverages inertial sensing for detecting eating moments. They evaluated the performance of multiple sensing modalities and achieved remarkable results using inertial sensing alone. In a semi-controlled lab study, EarBit accurately recognized all eating episodes and achieved an impressive accuracy of 90.1% and an F1-score of 90.9% in detecting chewing instances. In an unconstrained, outside-the-lab evaluation, EarBit achieved an accuracy of 93% in detecting chewing instances and accurately recognized almost all recorded eating episodes.

Zhang et al. [[5](#_2dlolyb)] conducted research on the detection of eating and drinking activities using a two-stage approach for surveillance purposes. Their approach focused on utilizing a wearable accelerometer sensor attached to the wrists of individuals. In the first stage, they employed a limb's three-dimensional kinematics movement model and the Extended Kalman Filter (EKF) to extract real-time arm movement features, represented by Euler angles, from the raw accelerometer data. In the second stage, they utilized the Hierarchical Temporal Memory (HTM) network for classification. The HTM network leveraged the spatial and temporal variations of the extracted features to accurately classify eating and drinking activities. The proposed approach was evaluated through real eating and drinking scenarios using three-dimensional accelerometers. The experimental results demonstrated the successful detection of activities with a very high level of accuracy, showcasing the effectiveness of the EKF and HTM based two-stage approach in activity detection.

Another significant contribution in the field of eating activity detection comes from Zhang et al.[[6](#_2dlolyb)], who developed a multi-sensor necklace called NeckSense. This innovative wearable device was designed to capture detailed information about an individual's eating activity and episodes throughout the entire day in a naturalistic setting. NeckSense integrates multiple sensors, including proximity from the necklace to the chin, ambient light, Lean Forward Angle, and energy signals, to accurately identify chewing sequences, which serve as fundamental indicators of eating activity. By clustering the identified chewing sequences, NeckSense can determine distinct eating episodes. To evaluate the effectiveness of NeckSense, two studies were conducted involving 11 participants with obesity and 9 participants without obesity. Over 470 hours of data were collected in real-life settings to simulate free-living conditions. The results demonstrated the reliability of NeckSense in detecting eating episodes across individuals with diverse body mass index (BMI) profiles throughout an entire waking day. In an exploratory study, the system achieved an impressive F1-score of 81.6% in detecting eating episodes. Even in all-day-long free-living settings, the system attained an F1-score of 77.1% for episode detection.

Kalantarian et al. [[7](#_2dlolyb)] have made a significant contribution with their novel food-intake monitoring system. The core component of this system is a wearable wireless-enabled necklace, which encompasses an embedded piezoelectric sensor, a compact Arduino-compatible microcontroller, a Bluetooth LE transceiver, and a Lithium-Polymer battery. By capturing motion in the throat, this necklace enables real-time monitoring of food intake and provides users with guidance through a mobile application. The study conducted by Kalantarian et al. involved data collection from 30 subjects, and the results showcased the system's ability to detect both solid and liquid foods. Leveraging a naive Bayes classifier, the system achieved an impressive F-measure of 0.837 for solid foods and 0.864 for liquid foods. Additionally, the research emphasized the importance of identifying extraneous motions such as head turns and walking, as they significantly reduced false positive rates in swallow detection.

All these researches mentioned multiple feature extraction, machine learning and deep learning algorithms in addition some have developed their own algorithm as well (swallowing detection algorithm.

By synthesizing the findings from our literature review, we have gained valuable insights into the current state of the art in eating activity detection using wearable sensors. This knowledge has not only informed our research design but also provided a foundation for evaluating the effectiveness and novelty of our proposed methods. In the following sections, we will present our research methodology, including the dataset utilized, the feature extraction techniques employed, the deep learning models trained, and the evaluation measures adopted. By building upon the existing body of knowledge, our research aims to make a significant contribution to the field of eating activity detection using wearable sensors.

## 1.3. Own Contributions:

In this research, we have made several contributions to the field of eating activity detection using wearable sensors and deep neural network models. These contributions include:

### Exploration of Wearable Sensors and Deep Neural Networks:

We conducted an in-depth exploration of the use of wearable sensors and deep neural network models for detecting eating activities. By leveraging wearable sensors, such as smart glasses, a smartwatch, and a smartphone, we collected real-time data to capture the intricate details of eating behaviors. Additionally, we extensively studied the capabilities of deep neural network models in learning complex patterns and representations from large-scale datasets, which is crucial for accurate activity recognition.

### Utilization of the CogeAge Dataset:

Our research focused on training multiple deep neural network models using the publicly available CogeAge dataset. This dataset is widely used for human activity recognition and encompasses a diverse range of atomic activities. By utilizing the CogeAge dataset, which was collected using wearable sensors, we ensured that our models were trained on relevant data specific to eating activities. This allowed us to enhance the accuracy and robustness of our models in detecting and monitoring eating behaviors.

### Feature Engineering and Feature Learning:

We contributed to the field of eating activity detection by implementing a comprehensive pipeline that involved feature engineering and feature learning. We designed and extracted a set of 17 handcrafted features tailored to capture the nuances of different activities, including eating. These features were carefully selected to ensure the generalization capacity of the eating activity detection system. Furthermore, we trained various deep learning models, such as convolutional neural networks (CNN), multi-layer perceptron (MLP), and LSTM networks, to learn complex patterns and representations from the extracted features.

### Evaluation of Model Performance:

To assess the effectiveness of our trained models, we conducted rigorous evaluation and validation processes. We measured various performance metrics, such as accuracy, precision, and recall, to gauge the models' accuracy in detecting eating activities. These evaluations allowed us to determine the strengths and weaknesses of each model and identify the most effective architecture for activity recognition, which can be further applied to eating activity datasets.

### Contribution to Privacy-conscious Approaches:

In response to privacy concerns related to camera-based methods, we focused on developing a robust and privacy-conscious approach for detecting activities using only wearable sensors. By eliminating the need for cameras, we addressed privacy concerns and provided a more comfortable and unobtrusive solution for monitoring and analyzing eating behaviors.

Overall, our research contributes to the advancement of wearable sensor-based activity recognition, specifically in the domain of eating activities detection. We propose and evaluate novel approaches utilizing deep neural networks, feature engineering, and feature learning techniques. The findings from our research have the potential to improve personalized interventions, promote healthier lifestyles, and contribute to better health outcomes in the context of eating activities detection

## 

## 1.4. Overview:

This research investigates the utilization of wearable sensors and deep neural network models for detecting eating activities. The primary objective is to develop a robust and privacy-conscious approach that can accurately monitor and analyze human behaviors without relying on camera-based methods. By leveraging wearable sensors, including smart glasses, a smartwatch, and a smartphone, real-time data is collected to capture important details of eating behaviors in various contexts and settings.

The research methodology encompasses feature engineering and feature learning stages. During feature engineering, a set of 17 carefully crafted features is extracted to capture the complex degrees of different activities, with a particular focus on eating. These features serve as inputs for the subsequent feature learning stage, where various deep learning models, such as convolutional neural networks (CNN), multi-layer perceptron (MLP), and LSTM networks, are trained to learn complex patterns and representations from the extracted features.

The publicly available CogeAge dataset is employed in this study, which comprises a wide range of atomic activities collected using wearable sensors. This dataset enables the training and evaluation of the deep neural network models in the context of various activities, including eating. The integration of multiple sensors, such as accelerometers, gyroscopes, and magnetometers, ensures a comprehensive capture of body, hand, and head movements, enhancing the accuracy and granularity of the detection process.

To evaluate the performance of the trained models, rigorous assessment measures are employed, including accuracy, precision, and recall. These metrics provide insights into the models' ability to accurately detect eating activities. Furthermore, the research emphasizes a privacy-conscious approach by eliminating the need for cameras, addressing privacy concerns, and allowing for more comfortable and unobtrusive monitoring of eating behaviors.

In conclusion, this research presents a practical and efficient approach to detect eating activities using wearable sensors and deep neural network models. By leveraging the power of advanced technologies, it opens new possibilities for accurate monitoring of dietary behaviors and offers a privacy-conscious alternative to camera-based methods. The outcomes contribute to the development of personalized interventions and improved health outcomes in the domain of eating activities detection, ultimately promoting healthier lifestyles and combating nutrition-related health issues.

# Basics:

## 2.1. Data Description:

The continuous recognition of human activity through wearable devices offers significant benefits in terms of detecting health issues and providing healthcare support.

Human activities are typically long-term and complex activities that consist of a sequence of short-term actions. For instance, the activity of preparing a meal involves actions such as cutting food, opening or closing the refrigerator door, and stirring utensils, among others. Recognizing these activities accurately and in real-time requires robust and efficient algorithms that can capture the temporal dynamics and context of the actions.

The data collection process of CogAge dataset for atomic activities encompassed a total of 55 different activities, involving the participation of 8 subjects. These activities were carefully selected to cover a wide range of movements and behaviors. In total, over 9700 instances were collected during the data acquisition phase. To ensure variability and account for different scenarios, the data collection was conducted on separate days for the training and testing phases.

During each phase, the subjects were instructed to wear three devices and perform 10 examples of each activity. Each example lasted for a duration of 5 seconds. However, a few instances had to be excluded from the dataset due to sensor errors, resulting in a final dataset of 9029 instances that were utilized for subsequent experiments and analysis.

There were a total of 61 atomic activities originally which were categorized into two distinct groups: 6 state activities that characterized the posture or position of the subjects, and 55 behavioral activities that represented their specific behaviors. However, we used only behavioral activities in our research.

The detailed breakdown of the atomic activities and their corresponding categories can be found in Table 1. The information of the dataset is taken from article [[8](#_2dlolyb)]. This meticulous organization of activities provided a comprehensive representation of the subjects' movements and behaviors, facilitating the subsequent analysis and evaluation of the collected data.



**Table 01.** Shows the list of all activities present in the dataset along with their labels.

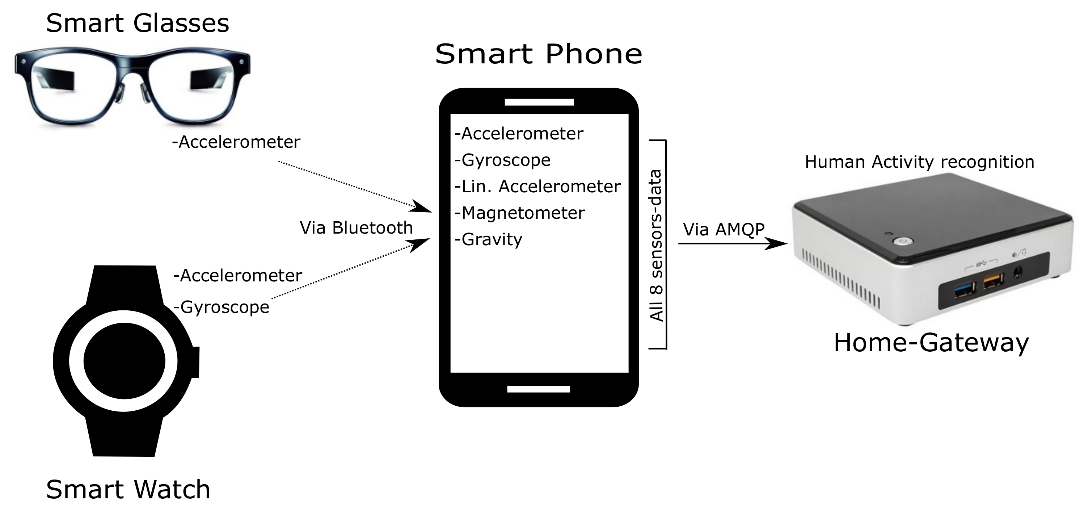
## 2.2. Wearable Devices and Sensors:

The activity recognition system we used utilizes three wearable devices to capture body, hand, and head movements, as depicted in Figure 4. Taken from article [[8](#_2dlolyb)]. The hardware configuration consists of the following devices

1. [**LG G5 smartphone**](https://www.lg.com/us/cell-phones/lg-VS987-Titan-g5)**:** This device, placed in the subject's front left pocket, captures body movements. It offers five sensory modalities:
   1. Accelerometer: Measures three-dimensional acceleration forces (including gravity) along the x, y, and z axes. Sampling rate: 200 Hz.
   2. Gyroscope: Records three-dimensional angular velocities on the x, y, and z axes. Sampling rate: 200 Hz.
   3. Gravity sensor: Provides three-dimensional data representing gravity forces on the x, y, and z axes. Sampling rate: 200 Hz.
   4. Linear accelerometer: Detects three-dimensional acceleration forces (excluding gravity) on the x, y, and z axes. Sampling rate: 200 Hz.
   5. Magnetometer: Measures three-dimensional intensities of the earth's magnetic field along the x, y, and z axes, aiding in orientation determination. Sampling rate: 100 Hz.

1. [**Huawei watch**](https://consumer.huawei.com/en/wearables/)**:** Placed on the subject's left arm, this device captures hand movements. It offers two sensory modalities:
   1. Accelerometer:Records three-dimensional acceleration forces on the watch's x, y, and z axes. Sampling rate: 100 Hz.
   2. Gyroscope: Measures three-dimensional angular velocities on the watch's x, y, and z axes. Sampling rate: 100 Hz.
2. [**JINS MEME glasses**](https://jinsmeme.com/en/)**:** Used to capture head movements, these glasses provide one sensor modality:
   1. Accelerometer: Captures three-dimensional acceleration forces along the glasses' x, y, and z axes. Sampling rate: 20 Hz.

CogAge dataset has eight sensor modalities offered by these wearable devices. Figure 4. illustrates the system layout, where the sensory data from the JINS MEME glasses and Huawei watch are initially transmitted via Bluetooth connection to the LG G5 smartphone. Subsequently, all the sensor data is forwarded to a home-gateway through Rabbit-MQ using a Wi-Fi connection. Our home-gateway, powered by an Intel NUC NUC5i5RYK, executes the atomic and composite activity recognition methods. The home-gateway's specifications include a Core i5-5250U 1.6 GHz CPU, 16 GB RAM, 450 GB HDD, and Debian 4.8.4-1 as the operating system.



**Figure 04.** System layout: Our system uses three wearable devices. The sensory data from the smart glasses and the smart watch are firstly sent to smartphone via Bluetooth connection and then all sensory data is sent to home-gateway through Rabbit-MQ using Wi-Fi connection.  **[**[8](#_2dlolyb)**]**

## 2.3. Data Preprocessing:

The CogeAge dataset we obtained for our research was already preprocessed and segmented. The training and testing data were separated into separate files, each containing sensor data for training and testing. The files included in the training dataset are:

* **trainAccelerometer.npy:** This file contains accelerometer data captured from the smartphone and represents three-dimensional sequences of acceleration forces (including gravity) acting on the smartphone's x, y, and z axes.
* **trainGravity.npy:** This file contains gravity sensor data from the smartphone, representing three-dimensional sequences of gravity forces on the x, y, and z axes.
* **trainGyroscope.npy:** This file contains gyroscope sensor data from the smartphone, representing three-dimensional sequences of angular velocities on the x, y, and z axes.
* **trainJinsAccelerometer.npy:** This file contains accelerometer data from the JINS MEME glasses, representing three-dimensional sequences of acceleration forces on the glasses' x, y, and z axes.
* **trainJinsGyroscope.npy:** This file contains gyroscope data from the JINS MEME glasses, representing three-dimensional sequences of angular velocities on the glasses' x, y, and z axes.
* **trainLabels.npy:** This file contains the corresponding labels or ground truth information for the training instances, indicating the activity or behavior being performed.
* **trainLinearAcceleration.npy:** This file contains linear accelerometer data from the smartphone, representing three-dimensional sequences of acceleration forces (excluding gravity) on the x, y, and z axes.
* **trainMSAccelerometer.npy:** This file contains accelerometer data from the Huawei watch, representing three-dimensional sequences of acceleration forces applied to the watch's x, y, and z axes.
* **trainMSGyroscope.npy**: This file contains gyroscope data from the Huawei watch, representing three-dimensional sequences of angular velocities on the watch's x, y, and z axes.
* **trainMagnetometer.npy:** This file contains magnetometer data from the smartphone, representing three-dimensional sequences describing intensities of the Earth's magnetic field along the x, y, and z axes.

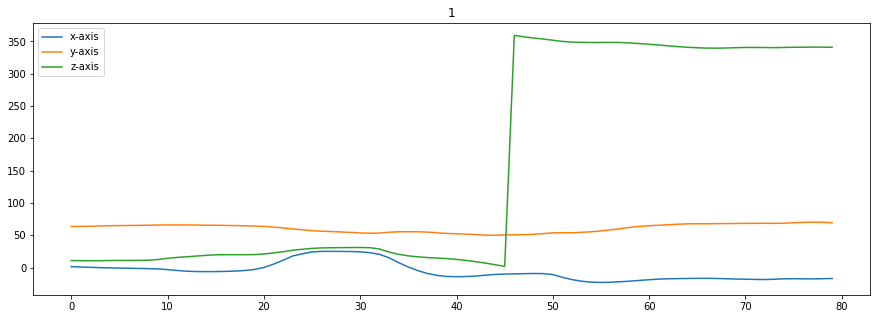
However, further preprocessing was required to adapt the dataset for our specific models. The brief description of which is given below:

### 2.3.1. Hand Crafted Feature Engineering Process:

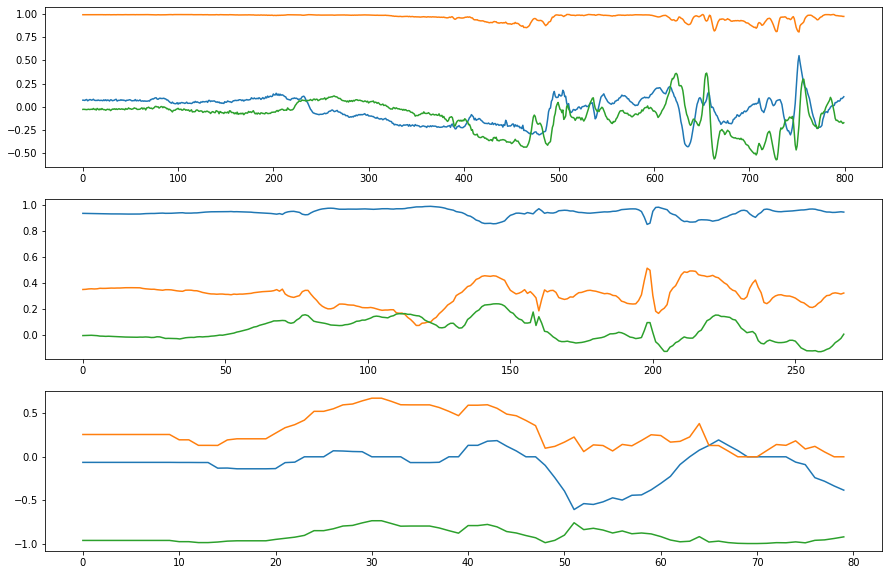
In Hand Crafted feature engineering process, we applied many techniques to the available dataset including denosing, segmentations, normalization and cross validations. However the effective ones were only normalization and cross validation which helped improving accuracy.

* **Normalization:**

The "Normalize" function takes each sensor data array and applies normalization to it. Normalization is a common preprocessing technique that scales the data to a standard range between 0-1. In this case, the "normalize" function is applied to each data instance in the array to normalize its values. The results of data before Figure 5. And Figure 6. after normalization are depicted below:



**Figure 5.** Displays a graph representing the results obtained from a sensor for the X, Y, and Z axes. Graph clearly showing the values ranges different for different axis. Z-axis ranges from 0 to 350 while x-axis only ranges from 0-50.



**Figure 6.** Represents the sensor results for the X, Y, and Z axes with normalization applied. The values on the x,y,z-axis have been transformed through a normalization process. Normalization involves scaling the values to a standardized range, often between 0 and 1, to facilitate fair comparisons and analysis.

* **Cross-validation:**

The code implements K-fold cross-validation, which is a technique used to evaluate the performance of a model. In this case, the data is divided into K subsets. The code iterates K times, each time using one of the subsets as the test set and the remaining subsets as the training set. This helps assess how well the model generalizes to unseen data.

Within each iteration, the code splits the features and labels into training and test sets based on the current fold. It then trains a classifier (Random Forest in this case) on the training features and labels. The trained classifier is used to predict labels for the test features, and the accuracy, F1 scores, and confusion matrix are computed and printed.

The average accuracy across all folds is accumulated in the "avg\_accuracy" variable. After completing all iterations, the average accuracy is calculated by dividing "avg\_accuracy" by the number of folds (K) and printed as the final result.

By applying normalization and utilizing K-fold cross-validation, the code aims to enhance the accuracy of the activity recognition model and evaluate its performance in a robust manner. The reported accuracy of 78.35% indicates the overall performance of the model across different folds of the data.

### 2.3.2. Deep Learning Models:

In the process of feature extraction for deep learning models, several preprocessing techniques were applied. While batch normalization is automatically performed in most models, an additional step of normalization was implemented in the autoencoder model due to its specific requirements.

The preprocessing steps involved in reshaping and resizing the sensor data. First, the training data from various sensors (such as accelerometer, gyroscope, gravity, etc.) is down-sampled using the *block\_reduce* function, which computes the mean value within blocks of data. This reduces the quantity of data while preserving relevant information.

Next, the data from certain sensors, such as Mobile and Jins sensors, are up-sampled using the *resize* function. This brings the data to a consistent shape of **(2284, 400, 3)** to ensure compatibility for further processing. Additionally, some data, like the MsMag sensor data, is repeated to match the desired shape.

Furthermore, the data from all sensors, including both training and testing data, are combined and stacked together. This results in a consolidated dataset of shape (4572, 400, 3), where 4572 represents the total number of samples (training + testing).

To split the data into training and testing sets, the labels associated with the data are also combined accordingly. The training data consists of 90% of the total data, while the remaining 10% is allocated for testing. The training data is further divided into training and validation sets using the *train\_test\_split* function, with a test size of 20%.

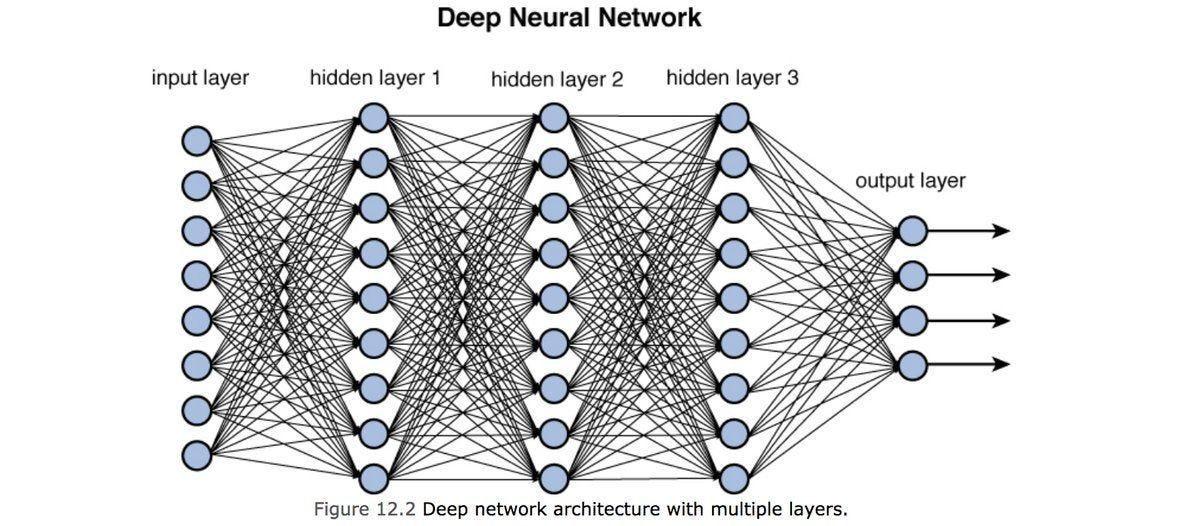
Finally, the labels are converted into categorical format using the *to\_categorical* function, which is necessary for training deep learning models. This converts the labels into a one-hot encoded representation, allowing the model to effectively learn and classify the activities.

Overall, these preprocessing steps ensure that the sensor data is appropriately formatted, normalized, and split for training, validation, and testing purposes.

## 2.4. Neural Networks:

We used neural network deep learning models like LSTM, AE, CNN, MLP, and hybrid models for activity recognition because they are really good at extracting important information from sensor data. These models can find complex patterns and relationships in the data, especially when it comes to activities that happen over time or involve multiple sensors. LSTM is great at understanding sequences of actions and how they change over time. CNN can recognize important spatial patterns in the sensor data. MLP is flexible and can understand complicated connections between different features. Hybrid models combine the strengths of multiple deep learning approaches, giving us even better ways to find important features and recognize activities accurately.

Neural network deep learning models are effective in activity recognition because they can automatically learn important information from the raw sensor data. They don't need humans to tell them what to look for; they can figure it out themselves. These models have many layers and thousands of neurons that work together to find meaningful patterns in the data. They can ignore irrelevant noise or unwanted signals and focus on what matters. These models are also really good at understanding the relationships between different parts of the data. They can handle complex activity patterns and figure out what activities are happening. Overall, these deep learning models are powerful tools for recognizing activities accurately and are used by many researchers to achieve the best results.



**Figure 7.** Illustrates the fundamental architecture of a neural network, comprising an input layer, three hidden layers, and an output layer. This diagram presents a high-level overview of the flow of information through the network.

Neural networks are like super-smart mathematical models that can learn and make predictions. They are inspired by the way our brain works. These networks have many artificial neurons that are connected together. Each neuron takes inputs, performs calculations, and produces an output. The outputs from one neuron become inputs for other neurons, creating a network of connections. Figure 07. Taken from the article[[9](#_2dlolyb)] depicts the basic structure of neural networks.

When training a neural network, we give it lots of examples with inputs and desired outputs. The network adjusts its internal weights and biases to find the best way to map inputs to outputs. It does this by comparing its predictions with the correct answers and making small adjustments to improve its performance.

Once the neural network is trained, we can use it to make predictions on new, unseen data. We feed the new data into the network, and it processes the inputs through its layers of neurons to produce an output. The network has learned from the training examples, so it can generalize and make predictions even on data it has never seen before.

Overall, neural networks are powerful tools for solving complex problems by learning from data. They can find patterns, make predictions, and recognize important features. Their ability to adapt and improve through training makes them valuable in various fields, including activity recognition.

# Methods:

This section provides a little detail of feature learning approaches selected used in our study. We test eight different state-of-the-art models in Human Activity Recognition (HAR).

* **Hand-Crafted Features (HCF):**

Hand crafted features are the simple statistical features computed on data. Statistical features include min, max, sum, variance and other insight properties of the data. This approach constitutes a baseline as the only manual feature extraction approach in our study.

* **Multi-Layer Perceptron (MLP):**

It is the most basic type of ANN featuring fully-connected layers. It consists of multiple layers of interconnected neurons, with each neuron applying a non-linear activation function. The features learned by this model are obtained in a supervised way. This approach constitutes a baseline for automatic supervised feature crafting.

* **Convolutional Neural Network (CNN):**

CNN is a deep learning architecture specifically designed to process grid-like structured data. CNN is used for image processing, audio recognition, natural language processing and time-series processing. CNN consist of convolutional layers which perform convolutional product on small patches and extract local patterns.

* **Long Short-Term Memory (LSTM):**

It is one of the most successful and widespread variant of Recurrent Neural Networks which feature layers containing LSTM cells, able to store information over time in an internal memory. Unlike traditional feedforward networks, LSTM can retain information over long time intervals, making it suitable for capturing temporal dependencies in sensor data. LSTM is well-suited for HAR, as it can effectively learn and represent complex activity patterns over time.[[10](#_3ygebqi)]

* **Bi-Directional Long Short-Term Memory (Bi-LSTM):**

Bidirectional Long Short-Term Memory (BiLSTM) is a type of recurrent neural network used primarily in natural language processing. It consists of two LSTMs, one processing the input in the forward direction and the other in the backward direction. BiLSTM is capable of utilizing information from both sides of the input sequence, making it a powerful tool for modeling the sequential dependencies between words and phrases in both directions of the sequence. BiLSTM is an extension of traditional LSTMs that can be used to improve model performance on tasks where all the sequence units shall be available, for example, sequence classification, speech recognition, and forecasting models. [[11](#_3ygebqi)]

* **Autoencoder (AE):**

An autoencoder is an unsupervised learning technique used in artificial neural networks to learn efficient coding of unlabeled data. It learns two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation. Autoencoders are used for dimensionality reduction and can generate new data that is similar to the input data. They consist of an encoder, a decoder, and a bottleneck layer that forms a compressed substitute of the input data. [[12](#_3ygebqi)][[13](#_3ygebqi)] This approach constitutes a baseline for unsupervised feature crafting.

* **Hybrid model featuring CNN and LSTM layers:**

CNN LSTM is a type of neural network architecture that combines Convolutional Neural Network (CNN) layers with Long Short-Term Memory (LSTM) layers. CNN layers are used for feature extraction on input data, while LSTM layers are used to support sequence prediction. [[14](#_3ygebqi)]

* **Hybrid model featuring CNN and Autoencoder:**

A CNN autoencoder is a type of neural network architecture that combines Convolutional Neural Network (CNN) layers with an autoencoder. Autoencoders are unsupervised learning techniques used for efficient data representation by training the network to capture the most important parts of the input image. CNN layers are used for feature extraction on input data, while autoencoders are used for dimensionality reduction. [15]

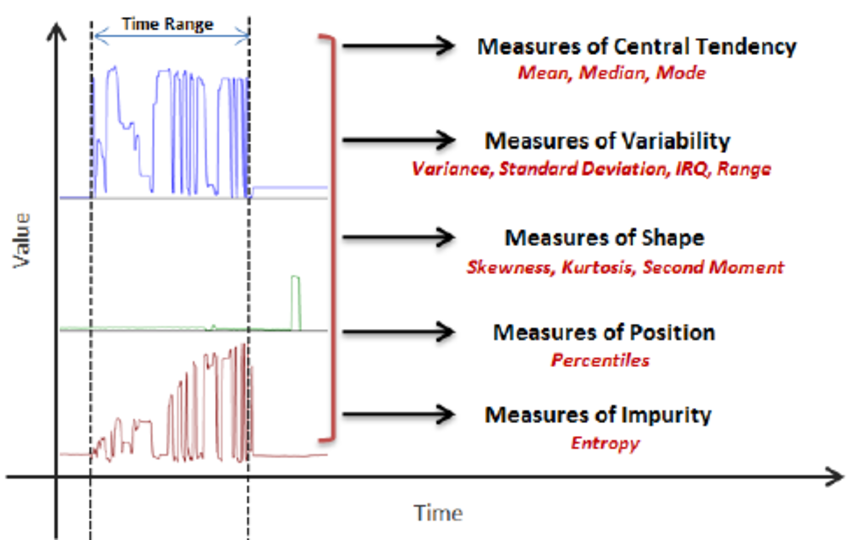
Each of the models aforementioned is used to learn features on training and testing datasets. We use Random Forest in Hand crafted features and loss=*'categorical\_crossentropy'* in the *model.compile()* in remaining models. It is a loss function used for multi-class classification models where there are two or more output labels. The output label is assigned a one-hot category encoding value in the form of 0s and 1s. The Categorical Cross Entropy function calculates the cross-entropy loss between the labels and predictions.

It should be noted that many of the feature extractors in our comparative study require a certain number of hyper-parameters (e.g., learning rate, number of hidden layers, number of units per layer, etc.) to be set. The choice of those parameters has been shown to have a high impact on the final classification performances for HAR using wearable sensors).

A more detailed description of each model and its parameters is provided in the following subsections.

## 3.1 Hand-Crafted Features

In our machine learning project, we explored the significance of handcrafted features for extracting valuable insights from the data. Handcrafted features refer to manually designed numerical representations derived from the raw data, offering a means to capture important characteristics and patterns. These features are calculated using various statistical measures, providing a comprehensive understanding of the data distribution and behavior.



**Figure 8.** This figure showcases the sensor data graph and a list of handcrafted features. Together, this visualization highlights the relationship between raw sensor measurements and the derived features, providing valuable insights for analysis and modeling purposes.

Here are the handcrafted features we calculated:

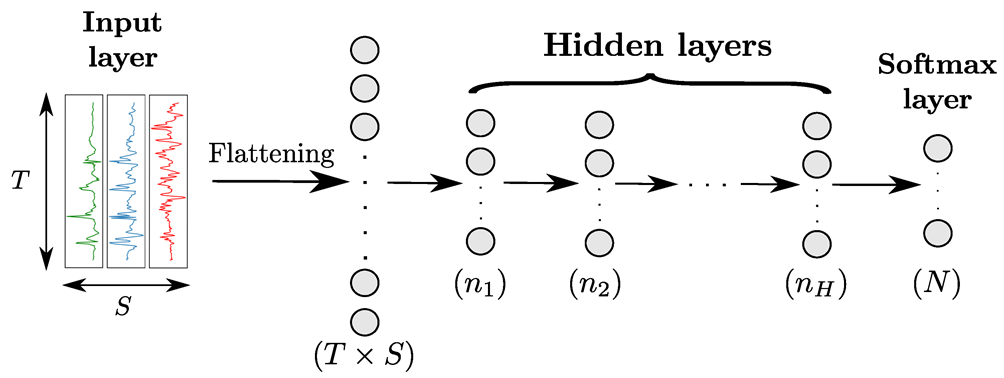
* Minimum (min): This feature represents the lowest value observed in the dataset. By calculating the minimum value for each axis (x, y, z), we gain insights into the lower bounds of the data.
* Maximum (max): The maximum feature captures the highest value observed in the dataset. Similar to the minimum feature, we calculate the maximum value for each axis to understand the upper bounds of the data.
* Mean: The mean feature represents the average value of the dataset. By calculating the mean for each axis (x, y, z), we gain insights into the central tendency of the data.
* Median: The median feature represents the middle value in the dataset, separating the higher and lower halves. Similar to the mean feature, it provides information about the central tendency of the data.
* First-order Mean: The first-order mean calculates the mean of the differences between consecutive data points. This feature captures the rate of change in the data, providing insights into acceleration or deceleration patterns.
* Second-order Mean: The second-order mean calculates the mean of the differences between consecutive first-order mean values. This feature captures the rate of change in the rate of change, offering insights into acceleration or deceleration of acceleration patterns.
* Variance: Variance measures the spread of the data points around the mean. It provides insights into the variability and dispersion of the data.
* Standard Deviation: Standard deviation is another measure of the spread of data points around the mean. It is calculated as the square root of the variance and offers insights into the dispersion of the data.
* Percentiles: We calculated specific percentiles such as the 20th, 50th (median), and 80th percentiles. These percentiles help understand the distribution and position of the data points within the dataset.
* Skewness: Skewness measures the asymmetry of the data distribution. It indicates whether the data leans towards the left or right, providing insights into the shape of the distribution.
* Kurtosis: Kurtosis measures the "peakedness" of the data distribution. It helps identify whether the distribution is more or less peaked compared to a normal distribution.
* Zero Crossing Point: This feature counts the number of times the data crosses the zero line. It provides insights into the frequency content and oscillations within the data.
* Interquartile Range (IQR): The IQR represents the range between the first quartile (25th percentile) and the third quartile (75th percentile). It provides insights into the variability and spread of the data.
* Spectral Energy: Spectral energy measures the distribution of energy across different frequencies in the data. It aids in understanding the frequency content and can be useful for analyzing time-series or sensor data.
* Spectral Entropy: Spectral entropy quantifies the randomness or unpredictability of the frequency content in the data. It provides insights into the complexity and diversity of the frequency components.

**Classification:**

Initially, we utilized the SVM (Support Vector Machine) classifier in our analysis. However, to improve the performance, we explored alternative models and found that the random forest classifier yielded a significantly higher accuracy. The random forest algorithm employs an ensemble of decision trees to make predictions, leveraging the diversity and collective decision-making of multiple trees. This approach proved to be more effective in capturing the underlying patterns and relationships within the sensor-based human activity recognition data, leading to improved classification accuracy.

## 3.2. Multi-Layer Perceptron

In our study, we employed a machine learning model called Multilayer Perceptron (MLP) as part of our sensor-based Human Activity Recognition (HAR) analysis. The MLP is a type of artificial neural network that consists of different layers of interconnected nodes, including an input layer, hidden layers, and an output layer.



**Figure:09** Architecture of a MLP model with H hidden layers for sensor-based HAR. Input data from the different sensor channels are first flattened into a (𝑇×𝑆)-dimensional vector and then fed to the hidden layers. All layers are fully-connected. The numbers in parenthesis indicate the number of neurons per layer. T, S and N designate the time length of the input data, the number of sensor channels, and the number of classes, respectively. [[10](#_3ygebqi)]

The purpose of using the MLP was to teach the model to recognize and classify various human activities based on the sensor data we collected. The MLP has a unique ability to capture complex patterns and relationships in the data, which made it suitable for our task.

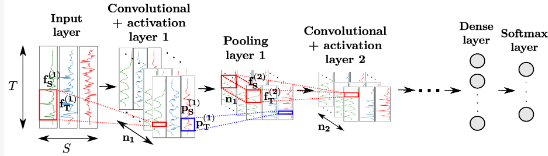
During the training process, the MLP adjusts its internal parameters, known as weights, by repeatedly comparing its predicted outputs with the actual outputs and minimizing the differences. This process, called backpropagation, allows the MLP to learn and improve its accuracy over time.

The advantage of using the MLP model is its capability to handle diverse and high-dimensional sensor data, which is often the case in HAR tasks. By leveraging this model, we were able to achieve higher accuracy in classifying human activities compared to other models we tested.

The MLP's flexibility in capturing intricate patterns and relationships in the data played a crucial role in improving the overall performance of our HAR system. Its ability to adapt and learn from the training data enabled it to make more accurate predictions and enhance our understanding of human activities based on sensor readings.

## 3.3. Convolutional Neural Network

In our research on sensor-based Human Activity Recognition (HAR), we utilized Convolutional Neural Networks as a crucial component of our analysis. CNN is a deep neural network which mimics the visual cortex of the brain that is responsible for the processing of visual information. In CNN, we have input layer, convolutional layers, pooling layers, dense layer and softmax layer. All these layers transform data in some way. The layers are connected sequentially with the output of one layer serving as the input for the next layer.



**Figure: 10** Architecture of a CNN model for sensor-based HAR. T, S and 𝑛𝑘 designate the time length of the input data, the number of sensor channels, and the number of convolutional kernels of the kth layer, respectively. The convolutional and pooling kernels of the kth layer process patches of the input data of sizes (𝑓(𝑘)𝑇,𝑓(𝑘)𝑆) and (𝑝(𝑘)𝑇,𝑝(𝑘)𝑆) respectively. Neurons of intermediate convolutional layers perform convolution products across all convolutional maps of the previous layer. [[16](#_3ygebqi)]

The beginning of a CNN is the input layer that takes the raw sensor data. In our analysis, this input consists of time-series data of T dimensions from multiple sensors S. This layer will then feed the data into subsequent convolutional layer.

After the input layer, we have convolutional layers. Each convolutional layer has multiple filters or kernels. In the kth convolutional layer, there are 𝑛𝑘 filters. Each filter is of size ((𝑘)𝑇, 𝑓(𝑘)𝑆), where 𝑓(𝑘)𝑇 and 𝑓(𝑘)𝑆 are the filter dimensions along the time and sensor channel axes respectively. These filters help in extracting local features from the input data. These features slides over input to create a feature map which is passed through an activation function. This step is crucial for introducing non-linearities into the network that allows it to learn even complex patterns. In our research, we have used the Rectified Linear Unit (ReLU), which introduces non-linearity by setting all negative values to zero.

After the convolutional layer, comes pooling layer. This is used to reduce the dimensions of the feature map to reduce computation and make the representation more robust. In our research, we have used max pooling. In max pooling we have to select a window of specific size n then slide it over the feature map to select the maximum value. In this way, we get a downsampled feature map from the pooled values.

After the processing of all the convolutional layers, we have dense layer in the end which is essentially Multilayer Perceptron (MLPs) where the network begins to combine simpler features to form more complex and high-level representations. Before passing our data to the dense layer, we have to flatten it which is simply means by converting our 3D output into 1D vector. The dense layer is also called Fully Connected Layer as all the neurons in it are connected to every neuron in the previous layer.

In the dense layer, we have used softmax activation function which takes a vector of raw scores and transforms it into a probability distribution. Each element of the output vector represents the probability that the input belongs to a particular class.

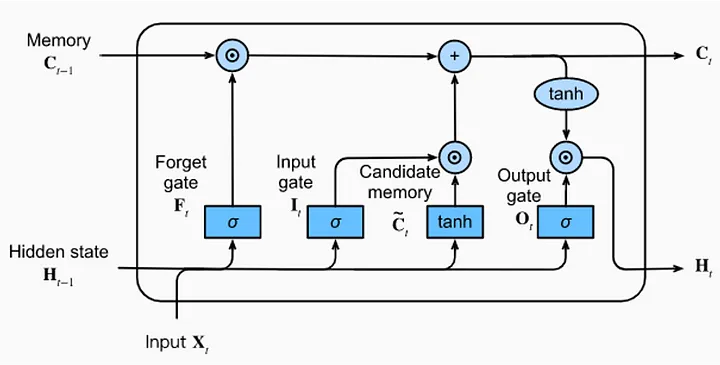
## 3.4 Long-Short Term Memory

In our research on sensor-based Human Activity Recognition (HAR), we utilized Long Short-Term Memory (LSTM) networks as a crucial component of our analysis. LSTM is a type of recurrent neural network (RNN) that excels at capturing long-term dependencies and patterns in sequential data.

LSTM networks are particularly well-suited for analyzing time series data, such as sensor readings collected over a period of time. They can effectively model the temporal dynamics and dependencies present in the data, making them highly applicable for HAR tasks where activities unfold over time.

Unlike traditional feedforward neural networks, LSTM networks incorporate memory cells that can store and retrieve information over extended sequences. This memory mechanism allows LSTMs to overcome the limitations of standard RNNs, which often struggle to capture long-range dependencies.

The LSTM architecture consists of recurrent units that are interconnected through gated mechanisms, including the input gate, forget gate, and output gate. These gates regulate the flow of information and control the updating and forgetting of memory within the network. By selectively retaining relevant information and discarding unnecessary details, LSTMs can effectively process and remember patterns that are important for activity recognition.



**Figure 11**: Illustrates the architecture of a Long Short-Term Memory (LSTM) neural network. LSTMs are designed to process sequential data by incorporating specialized gates and memory cells. The input gate controls new input information, the forget gate manages discarding unnecessary information, and the output gate determines the relevant output. This architecture enables LSTMs to effectively capture long-term dependencies and is commonly applied in tasks such as time series analysis. [[17](#_3ygebqi)]

LSTM unit can be described as follows:

First, let's define the input and output variables:

* Input: x\_t (input at time t)
* Previous Hidden State: h\_{t-1}
* Previous Cell State: c\_{t-1}
* Output: h\_t (output at time t)
* Cell State: c\_t (cell state at time t)

The equations for an LSTM cell are as follows:

1. Input Gate (i\_t): i\_t = sigmoid(W\_i \* [h\_{t-1}, x\_t] + b\_i)
2. Forget Gate (f\_t): f\_t = sigmoid(W\_f \* [h\_{t-1}, x\_t] + b\_f)
3. Cell State Update (g\_t): g\_t = tanh(W\_c \* [h\_{t-1}, x\_t] + b\_c)
4. New Cell State (c\_t): c\_t = f\_t \* c\_{t-1} + i\_t \* g\_t
5. Output Gate (o\_t): o\_t = sigmoid(W\_o \* [h\_{t-1}, x\_t] + b\_o)
6. New Hidden State (h\_t): h\_t = o\_t \* tanh(c\_t)

In the equations above:

* W\_i, W\_f, W\_c, W\_o are weight matrices.
* b\_i, b\_f, b\_c, b\_o are bias vectors.
  + represents matrix multiplication.
* [h\_{t-1}, x\_t] denotes the concatenation of h\_{t-1} and x\_t.

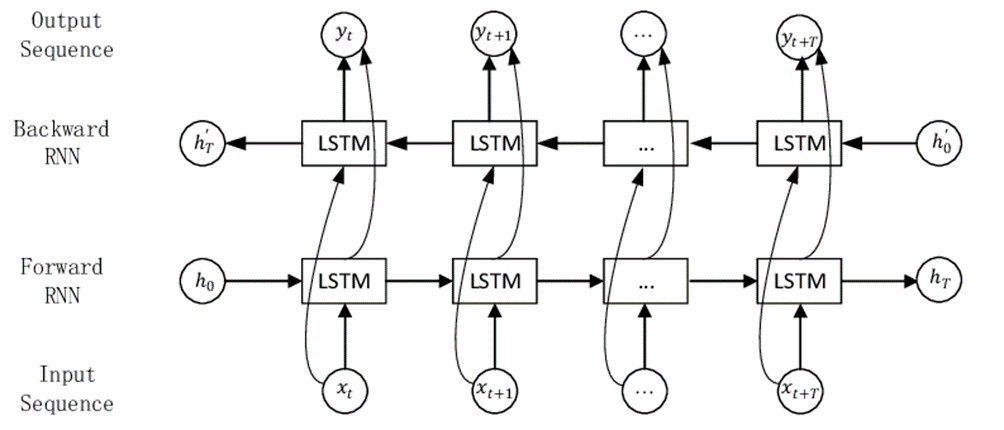
These equations govern the flow of information within a single LSTM cell, allowing it to retain and update information over time and produce relevant outputs. In practice, these equations are applied sequentially across multiple time steps to process and analyze sequential data.

One of the key advantages of LSTMs is their ability to handle sequential data of variable length. This flexibility makes them well-suited for HAR, where the duration and timing of activities may vary across different individuals or scenarios. LSTMs can dynamically adapt to different sequence lengths and capture meaningful patterns without requiring fixed-size inputs.

Furthermore, LSTMs can learn and model both short-term dependencies and long-term dependencies in the data. This capability is essential in HAR, as activities often exhibit complex temporal dynamics that span over multiple time steps. By retaining memory of past observations, LSTMs can capture the context and temporal evolution of activities.

## 3.4. Bi-Directional Long-Short Term Memory

Bidirectional Long Short-Term Memory (Bi-LSTM): Bi-LSTM is an extension of the LSTM architecture that incorporates information from both past and future contexts. It consists of two LSTM layers: one processes the input sequence in the forward direction, while the other processes it in the backward direction. By capturing information from both directions, Bi-LSTM can potentially capture dependencies that span across the entire sequence.



**Figure. 12**: depicts the architecture of a Bidirectional LSTM (Bi-LSTM) model. The Bi-LSTM combines forward and backward information flow through two parallel LSTM layers. The outputs from both directions are then combined to generate a comprehensive representation of the input sequence. [[18](#_3ygebqi)]

Mathematical Equations: Let's denote the input sequence as X = [x₁, x₂, ..., xₙ], where xᵢ represents the i-th input element. We'll denote the hidden states of the forward LSTM layer as hᶠ = [hᶠ₁, hᶠ₂, ..., hᶠₙ], and the hidden states of the backward LSTM layer as hᵇ = [hᵇ₁, hᵇ₂, ..., hᵇₙ].

The forward LSTM layer computes the forward hidden states as follows:

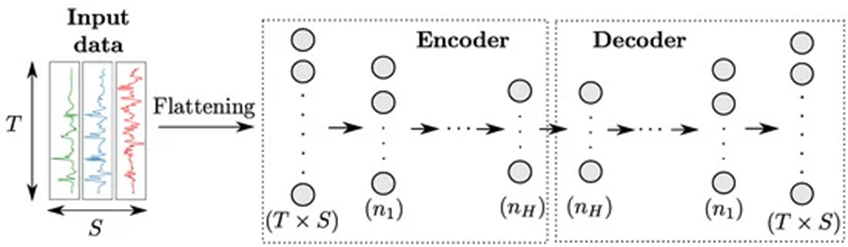
* Forget gate: fᶠᵢ = σ(Wᶠ[f] · [hᶠᵢ₋₁, xᵢ] + bᶠ[f])
* Input gate: iᶠᵢ = σ(Wᶠ[i] · [hᶠᵢ₋₁, xᵢ] + bᶠ[i])
* Candidate state: ˜Cᶠᵢ = tanh(Wᶠ[C] · [hᶠᵢ₋₁, xᵢ] + bᶠ[C])
* Cell state: Cᶠᵢ = fᶠᵢ \* Cᶠᵢ₋₁ + iᶠᵢ \* ˜Cᶠᵢ
* Output gate: oᶠᵢ = σ(Wᶠ[o] · [hᶠᵢ₋₁, xᵢ] + bᶠ[o])
* Hidden state: hᶠᵢ = oᶠᵢ \* tanh(Cᶠᵢ)

Similarly, the backward LSTM layer computes the backward hidden states as follows:

* Forget gate: fᵇᵢ = σ(Wᵇ[f] · [hᵇᵢ₊₁, xᵢ] + bᵇ[f])
* Input gate: iᵇᵢ = σ(Wᵇ[i] · [hᵇᵢ₊₁, xᵢ] + bᵇ[i])
* Candidate state: ˜Cᵇᵢ = tanh(Wᵇ[C] · [hᵇᵢ₊₁, xᵢ] + bᵇ[C])
* Cell state: Cᵇᵢ = fᵇᵢ \* Cᵇᵢ₊₁ + iᵇᵢ \* ˜Cᵇᵢ
* Output gate: oᵇᵢ = σ(Wᵇ[o] · [hᵇᵢ₊₁, xᵢ] + bᵇ[o])
* Hidden state: hᵇᵢ = oᵇᵢ \* tanh(Cᵇᵢ)

The final hidden states of the Bi-LSTM layer can be obtained by concatenating the corresponding forward and backward hidden states: h = [hᶠ, hᵇ].

## 3.5. Autoencoder

Autoencoder is a type of neural network that compresses the input into a lower-dimensional code and then reconstructs the output from this representation. It is a form of unsupervised learning that learns efficient data representations by training the encoding and decoding methods, and a loss function to compare the output with the target [[19](#_3ygebqi)].

**Figure. 13:** Shows the architecture of Autoencoder. Encoder converts the raw data of high dimensional to low dimensional. While the decoder converts the low dimensional data to high dimension.

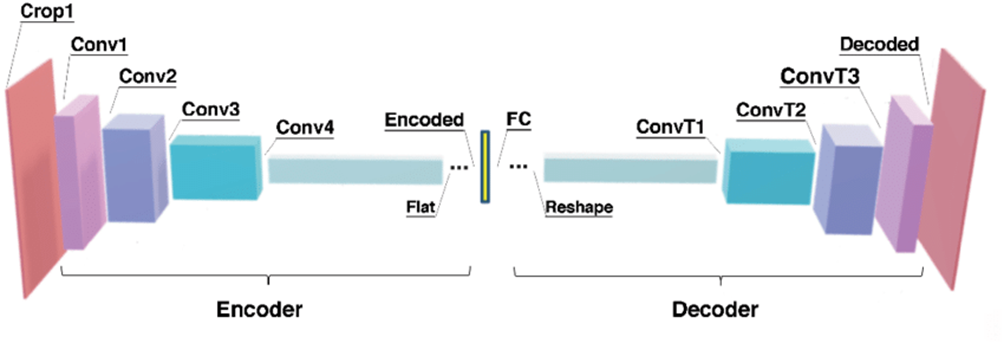
The code is a compact “summary” or “compression” of the input, also called the latent-space representation [[19](#_3ygebqi)]. Autoencoders are surprisingly simple neural architectures that are basically a form of compression, similar to the way an audio file is compressed using MP3, or an image file is compressed using JPEG [11]. The simplest architecture for constructing an autoencoder is to constrain the number of nodes present in the hidden layer(s) of the network, limiting the amount of information that can flow through the network. The loss function used in autoencoders is typically the mean squared error between the input and the output. Autoencoders are very useful dimensionality reduction techniques and can be used for feature extraction on raw data that can be used to train a different model [[20](#_3ygebqi)].

## 

## 3.5. Convolutional Autoencoder

A convolutional autoencoder (CAE) is a type of neural network that uses convolutional layers to extract features from images and compress them into a lower-dimensional code, and then reconstructs the output from this representation. Here are some details about the convolutional autoencoder model:

* CAEs are unsupervised neural network models that summarize the general properties of data in fewer parameters while learning how to reconstruct it after compression [[21](#_3ygebqi)].
* CAEs are mainly utilized for reducing and compressing the input dimension size, removing noise while simultaneously keeping all the important features of the input data [[22](#_3ygebqi)].
* CAEs are a type of autoencoder that can be used for a variety of tasks, such as data compression, feature extraction, image denoising, anomaly detection, and classification in high-dimensional noisy image datasets [[21](#_3ygebqi)][[22](#_3ygebqi)][[23](#_3ygebqi)].
* The primary use of CAEs is the generation of the latent space or the bottleneck, which forms a compressed substitute of the input data and can be easily used for further analysis [[21](#_3ygebqi)].
* CAEs are a powerful tool for data compression and analysis. They can be used to discover hidden patterns within your data and then use those patterns to generate new data that is similar to the input data [[23](#_3ygebqi)].
* The core architecture of an autoencoder comprises of three parts: encoder, code, and decoder. The encoder maps the input into the code, the code represents the compressed input that is fed to the decoder, and the decoder takes encoder output and generates an image using it [[23](#_3ygebqi)].
* The encoding part of autoencoders helps to learn important hidden features present in the input data, in the process to reduce the dimensionality of the input data.
* CAEs are commonly used for image generation tasks, image denoising, compression, and image reconstruction [[24](#_3ygebqi)].
* Variational autoencoders (VAE) are a type of generative model that learns a probabilistic representation of the input data. VAE models are trained to learn a mapping from the input data to a probability distribution in a lower-dimensional latent space, and then to generate new samples from this distribution [[24](#_3ygebqi)].

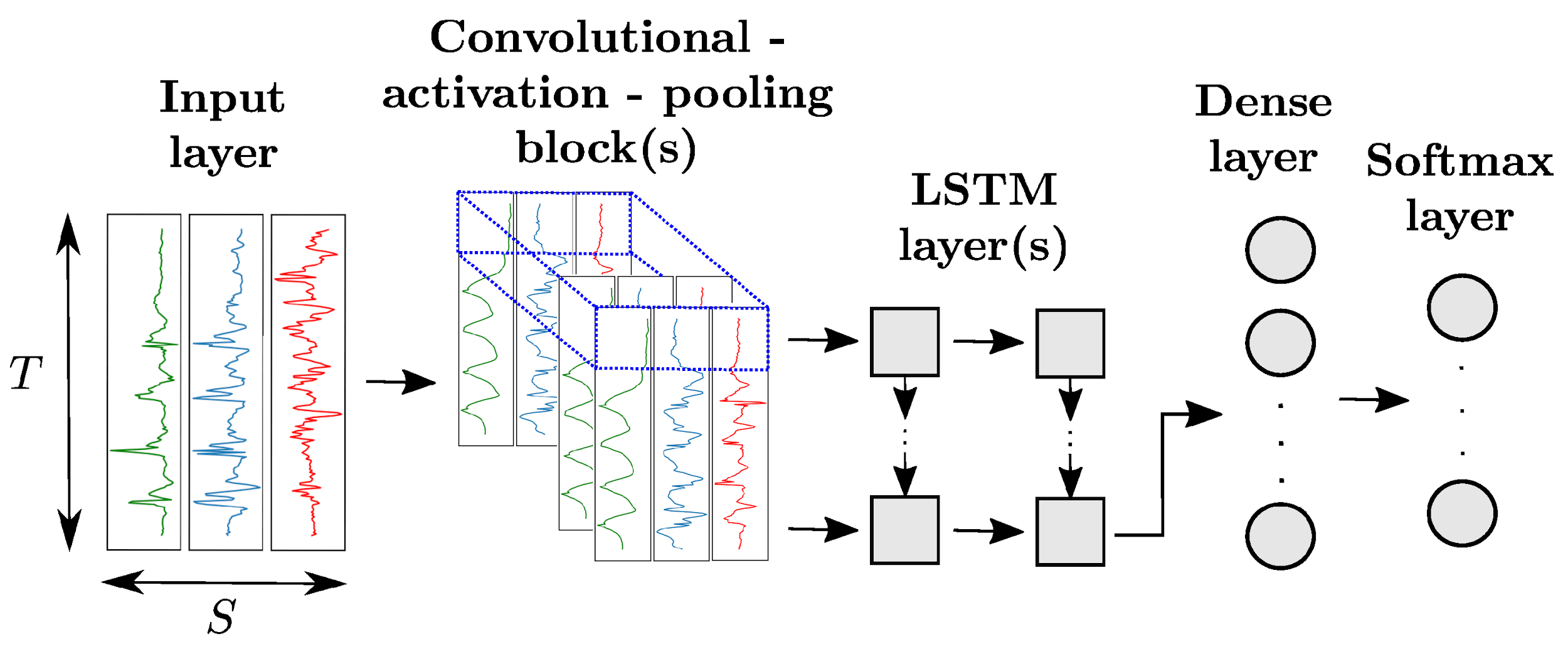


**Figure 14**: Showcases the architecture of a Convolutional Autoencoder, a deep learning model used for unsupervised representation learning and data compression. The autoencoder consists of an encoder, which compresses the input data using convolutional and pooling layers, and a decoder, which reconstructs the original input from the compressed representation using upsampling and deconvolutional layers. [[25](#_3ygebqi)]

In summary, convolutional autoencoder models are a type of neural network that can be used for data compression, feature extraction, image denoising, anomaly detection, and classification in high-dimensional noisy image datasets. They are a powerful tool for data compression and analysis, and can be used to discover hidden patterns within your data.

## 3.6. Convolutional Long-Short Term Memory

In convolutional LSTM, the LSTM layers are appended to convolutional layers due to the high modularity of the ANN architecture. This hybrid architecture is designed to leverage the strengths of both CNNs and LSTMs in processing time-series data from the sensors.



**Figure 15:** Architecture of a hybrid CNN+LSTM model for sensor-based HAR. Each slice along the time dimension of the output of the convolutional block(s) (in blue) is fed to one LSTM cell. All LSTM layers are organized in a many-to-many pattern, except the last which follows a many-to-one scheme. [[26](#_3ygebqi)]

In this architecture, the CNN component is composed of multiple convolutional blocks. These convolutional blocks process the input data to produce an n-dimensional time series, where 'n' corresponds to the number of neurons in the final convolutional layer. This output captures high-level features across time and is then partitioned into slices along the time dimension.

Each slice represents high-level features at a specific time step that is fed into an LSTM cell after being flattened to a one-dimensional vector. The LSTM excels in handling time-series data as it can maintain memory of past information. The LSTM comprises several LSTM cells, one for each time slice, and learns the temporal dependencies across time steps. Each LSTM cell outputs a value that feeds into the next layer except the last one.

With the fusion of spatial feature extraction capabilities of CNNs and the temporal analysis proficiency of LSTMs, this hybrid architecture is especially powerful for our research on Human Activity Recognition (HAR) in which both spatial and temporal patterns are crucial. This synergy between CNN and LSTM enables the model to make highly informed predictions or classifications based on the sensor data

# 

# 4. Experimental Results:

This section presents the implementation details of the different feature learning approaches and their comparative results on the Cognitive dataset.

In our study, all algorithms and models were implemented using Python in Colab. For the algorithms SVM, and RF, and the deep learning models MLP, CNN, LSTM, Bi-LSTM and AE the libraries sklearn and Keras with Tensorflow 2.2.0 backend were used. ADAM was chosen as the optimizer for our deep learning model with an initial learning rate of 0.001, and trained with 50 epochs at a batch size of 32. The categorical cross entropy was used as the loss function for the deep learning models.

The accuracy result of each model is given in the result [section 4.7](#_3l18frh).

## 4.1. Hand Crafted Features (HC)

First of all to remove the variation between the values, we perform normalization using sklearn.preprocessing library to project its values in the interval [0,1]. Then we computed 17 hand crafted features on sensor channel of the data frames independently, following the suggestion of [20]. The data is then concatenated to achieve a window of 17xS (where S is the number of sensors x number of channels). In our case we have 9 sensors of 3 channels.

List of these hand-crafted features is given below.

|  |  |  |
| --- | --- | --- |
| **Table 02.** List of the hand-crafted features used in our study. Each feature is computed on each sensor channel independently. | | |
| Hand-Crafted Features | | |
| Maximum | Minimum | Mean |
| Median | First Order Mean | Second Order Mean |
| Skewness | Percentile 20 | Percentile 50 |
| Percentile 80 | Variance | Standard Deviation |
| Kurtosis | Zero Crossing Point | Spectral Entropy |
| Spectral Energy | Inter quartile Range |  |

## 4.2. Multi-Layer Perceptron (MLP):

We used a MLP with two hidden layers with Rectified Linear Units (RELU) activations, taking vectors obtained by flattening frames of data as inputs. We noticed that the addition of a batch normalization layer significantly improved the classification results. Batch normalization layers standardize the inputs of the subsequent layers by computing parameters on batches of training data to normalize each sensor channel independently. We also use kernel regularizer L2 with a value of 0.01 to avoid overfitting.

|  |  |  |
| --- | --- | --- |
| **Layer** | **Neurons** | **Activation** |
| Batch Normalization | --- | --- |
| Flatten | 10800 | --- |
| Dense | 2500 | Relu |
| Dense | 2500 | Relu |
| Dense | 55 | Softmax |

**Table 03.** MLP architecture with kernel\_regularizer= l2(0.01) in Dense Layers.

## 4.3. Convolutional Neural Network

|  |  |  |
| --- | --- | --- |
| **Layer** | **Neurons** | **Activation** |
| Batch Normalization | --- | --- |
| Conv2D | 390, 3, 50 | relu |
| Max Pooling2D | 195, 3, 50 | --- |
| Flatten | 29250 | --- |
| Dense | 500 | relu |
| Dense | 55 | softmax |

## 4.4. LSTM

|  |  |  |
| --- | --- | --- |
| **Layer** | **Neurons/Dropout** | **Activation** |
| Batch Normalization | 400,3,9 | --- |
| Reshape | 400,27 | --- |
| LSTM | 400,150 | --- |
| Dropout | 0.3 | --- |
| LSTM | 150 | --- |
| Dropout | 0.2 | --- |
| Dense | 4500 | relu |
| Dropout | 0.2 | --- |
| Dense | 55 | softmax |

## 4.5. Bi-LSTM

|  |  |  |
| --- | --- | --- |
| **Layer** | **Neurons** | **Activation** |
| Batch Normalization | 400, 3, 9 | --- |
| Reshape | 400, 27 | --- |
| Bidirectional | 400,200 | --- |
| Flatten | 80000 | --- |
| Dense | 500 | relu |
| Dense | 55 | softmax |

## 4.6. Convolutional LSTM

|  |  |  |
| --- | --- | --- |
| **Layer** | **Neurons** | **Activation** |
| Batch Normalization | 400,3,9 | --- |
| CONV2D | 390, 3, 60 | relu |
| Max Pooling | 195, 3, 60 | --- |
| Reshape | 195,180 | --- |
| LSTM | 150 | --- |
| Dense | 2500 | relu |
| Dense | 55 | softmax |

## 4.7. Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Model Name** | **Accuracy** | **Weighted Score** | **F1 Score** |
| Hand Crafted | 78.35% | 81% | 81% |
| MLP | 67% | 57% | 57% |
| CNN | 68% | 68% | 67% |
| LSTM | 59% | 51% | 50% |
| BI LSTM | 63% | 54% | 54% |
| Autoencoder | 63.72% | ----- | ------ |
| Conv LSTM | 68% | 62% | 61% |
| Conv AE | 1.64% | ------ | ------ |

# 5. Discussion:

The presented research explores the use of wearable sensors and deep neural network models for the detection of eating activities. The primary focus is on training multiple deep neural network models using the publicly available CogeAge dataset, acquired from wearable sensors such as smart glasses, a smartwatch, and a smartphone. While specific experiments on eating activities detection were not conducted, the trained models demonstrate the potential for accurately identifying and monitoring eating behaviors.

One of the key contributions of this research lies in the advancement of wearable sensor-based activity recognition. By leveraging wearable sensors, researchers can collect real-time data on individuals' eating behaviors in various contexts and settings. This allows for a comprehensive understanding of eating activities beyond simply recognizing when someone is consuming food. It enables the capture of intricate details such as eating speed, portion sizes, eating patterns, and emotions related to eating.

Accurate detection and analysis of eating activities have significant implications for promoting healthier lifestyles and addressing the complex interplay between nutrition and health. It enables personalized interventions, facilitates dietary management, and provides valuable feedback for individuals striving to make positive changes in their eating habits. Moreover, it offers researchers and healthcare professionals a wealth of data to identify patterns, correlations, and risk factors associated with nutrition-related diseases, leading to more effective preventive measures and interventions.

The challenges associated with detecting eating activities are discussed in the introduction. Unlike activities that involve continuous body movements, eating activities involve subtle gestures and motions that can be challenging to capture accurately. The diverse nature of eating behaviors and the social and environmental aspects of eating add complexity to the detection process. However, by training multiple deep neural network models, researchers can overcome these challenges and enhance the accuracy and robustness of the detection system.

The research methodology involves feature engineering through handcrafted feature extraction methods and feature learning using various deep learning models. Feature engineering plays a crucial role in capturing the intricate nuances of different activities, including eating. The extraction of carefully crafted features enables the models to discern between various aspects of the activities with improved accuracy. Feature learning, on the other hand, focuses on leveraging the extracted features to learn complex patterns and representations in the data. Deep learning models, such as convolutional neural networks, multi-layer perceptron, auto-encoders, LSTM networks, and hybrid models, are trained to analyze the most effective architecture for activity recognition, which can be applied to eating activities as well.

The utilization of the CogeAge dataset, which encompasses a wide range of atomic activities and was collected using multiple wearable sensors, provides a robust foundation for training and evaluating the performance of the deep neural network models. The multi-sensor setup captures crucial movement data, including body, hand, and head motions, through a variety of sensory modalities, enhancing the accuracy and granularity of the detection process.

The research pipeline includes various stages, such as data preprocessing, segmentation, feature extraction, and classification. Data preprocessing involves operations such as normalization, de-noising, and sampling to make the collected data suitable for further analysis. Segmentation plays a critical role in identifying specific segments within the dataset that contain relevant information about the activities to be recognized. Feature extraction computes values that abstract each data segment into a representation highly relevant to the associated activity. Classification focuses on constructing a classifier using the extracted features to establish clear boundaries in the feature space and differentiate between different activities.

The discussion also addresses the problem statement and motivation of the research. Current approaches for detecting eating activities often rely on camera-based methods, which raise privacy concerns and may not be suitable for real-time monitoring. By developing a robust and privacy-conscious approach using only wearable sensors, this research overcomes these limitations and provides an alternative solution for detecting activities. The trained models can be applied to various datasets, including the detection and analysis of eating activities, without the need for extensive data acquisition.

The motivation behind this research lies in the potential impact of accurately detecting eating activities using wearable sensors. By providing personalized and improved health monitoring and effective dietary management, such a system can promote healthier lifestyles and combat nutrition-related health issues. Moreover, the research contributes to individuals' privacy by eliminating the need for intrusive camera-based methods.

In conclusion, the research presented in this paper demonstrates the potential of wearable sensors and deep neural network models for accurately detecting and monitoring eating activities. The combination of feature engineering and feature learning enhances the accuracy and robustness of the models, enabling them to discern intricate details of eating behaviors. The findings contribute to the advancement of wearable sensor-based activity recognition and have significant implications for promoting healthier lifestyles, addressing nutrition-related health issues, and ensuring individuals' privacy. Further research can focus on conducting specific experiments on eating activities detection to validate and refine the proposed approach.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 6. Conclusion:

In conclusion, this research has explored the use of wearable sensors and deep neural network models for the detection of eating activities. By training multiple deep neural network models using the CogeAge dataset, acquired from wearable sensors such as smart glasses, a smartwatch, and a smartphone, the potential for accurately identifying and monitoring eating behaviors has been demonstrated. The research has highlighted the importance of accurate detection and analysis of eating activities in promoting healthier lifestyles, addressing nutrition-related health issues, and providing personalized interventions for individuals.

The research methodology involved feature engineering through handcrafted feature extraction methods and feature learning using various deep learning models. By extracting carefully crafted features and leveraging deep learning architectures, the models have shown the ability to capture complex patterns and representations in the data, leading to improved accuracy in recognizing eating activities. The integration of multiple wearable sensors in the data collection process has provided a comprehensive view of body, hand, and head motions, enhancing the granularity and reliability of the detection system.

The findings of this research contribute to the advancement of wearable sensor-based activity recognition, particularly in the domain of eating activities detection. The ability to accurately detect eating behaviors opens up opportunities for personalized interventions, effective dietary management, and valuable insights for researchers and healthcare professionals. Moreover, the use of wearable sensors offers a privacy-conscious alternative to camera-based methods, addressing concerns and facilitating real-time monitoring of individuals' eating activities.

# 

# 7. Future Work:

While this research provides a solid foundation for detecting eating activities using wearable sensors and deep neural network models, there are several avenues for future work to expand and improve upon these findings. Some potential areas for future research include:

* Conducting Specific Experiments: The current research focused on training and fine-tuning deep neural network models using the CogeAge dataset without specific experiments on eating activities detection. Conducting dedicated experiments on eating activities would provide empirical evidence of the models' performance in accurately recognizing different eating behaviors.
* Dataset Expansion and Diversity: The CogeAge dataset used in this research provided a valuable starting point. However, future work could involve the collection and integration of larger and more diverse datasets specifically focused on eating activities. This would further enhance the models' robustness and generalizability across different populations and cultural contexts.
* Real-time Monitoring and Feedback: Integrating the developed models into real-time monitoring systems would enable immediate feedback and intervention for individuals engaging in eating activities. This could involve the development of wearable devices or mobile applications that provide real-time alerts, suggestions, or personalized recommendations to improve eating behaviors.
* Long-Term Behavior Analysis: Extending the research to analyze long-term eating behavior patterns could provide insights into individuals' habits, trends, and potential correlations with health outcomes. Long-term monitoring using wearable sensors would enable the identification of risk factors or triggers related to unhealthy eating behaviors.
* User Interaction and Engagement: Exploring methods to enhance user interaction and engagement with the wearable sensor-based system could improve its usability and effectiveness. This could involve incorporating gamification elements, social support features, or personalized coaching strategies to encourage individuals to adopt healthier eating habits.
* Privacy and Ethical Considerations: As wearable sensor-based systems involve the collection of personal data, addressing privacy and ethical considerations becomes crucial. Future work should focus on developing robust privacy frameworks and ensuring transparent data handling practices to protect individuals' privacy while utilizing their data for research and intervention purposes.

By pursuing these directions for future research, the field of eating activities detection using wearable sensors and deep neural network models can further advance, leading to more accurate, personalized, and impactful interventions for promoting healthier eating behaviors and improving overall well-being.

# References

1. O. of the P. C. of Canada, “2016. *Survey of Canadians on Privacy*,” 26-Jan-2017.[Online].Available: https://www.priv.gc.ca/en/opc-actions-and-decisions/research/explore-privacy-research/2016/por\_2016\_12/.[Accessed:16-Jun-2023]
2. Doulah, A., Ghosh, T., Hossain, D., Imtiaz, M. H., & Sazonov, E. (2021). "Automatic Ingestion Monitor Version 2" - A Novel Wearable Device for Automatic Food Intake Detection and Passive Capture of Food Images. *IEEE journal of biomedical and health informatics*, *25*(2), 568–576. https://doi.org/10.1109/JBHI.2020.2995473
3. Link to literature review excel sheet.

[https://github.com/Wajeeha- Fatima/FYP/blob/b761d192b1feda1c59ad39de36e479fa2ee2e7e1/Generalized%20literature%20Review.xlsx](https://github.com/Wajeeha- Fatima/FYP/blob/b761d192b1feda1c59ad39de36e479fa2ee2e7e1/Generalized literature Review.xlsx)

1. Bedri, A., Li, R., Haynes, M., Kosaraju, R. P., Grover, I., Prioleau, T., Beh, M. Y., Goel, M., Starner, T., & Abowd, G. (2017). EarBit: Using Wearable Sensors to Detect Eating Episodes in Unconstrained Environments. Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies, 1(3), 37. https://doi.org/10.1145/3130902GHVH
2. Zhang, S., Ang, M. H., Jr, Xiao, W., & Tham, C. K. (2009). Detection of activities by wireless sensors for daily life surveillance: eating and drinking. Sensors (Basel, Switzerland), 9(3), 1499–1517. https://doi.org/10.3390/s90301499
3. Zhang, S., Zhao, Y., Nguyen, D. T., Xu, R., Sen, S., Hester, J., & Alshurafa, N. (2020). NeckSense: A Multi-Sensor Necklace for Detecting Eating Activities in Free-Living Conditions. Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies, 4(2), 72. https://doi.org/10.1145/3397313
4. Kalantarian, H., Alshurafa, N., Le, T., & Sarrafzadeh, M. (2015). Monitoring eating habits using a piezoelectric sensor-based necklace. Computers in biology and medicine, 58, 46–55. https://doi.org/10.1016/j.compbiomed.2015.01.005
5. Nisar MA, Shirahama K, Li F, Huang X, Grzegorzek M. Rank Pooling Approach for Wearable Sensor-Based ADLs Recognition. *Sensors*. 2020; 20(12):3463. https://doi.org/10.3390/s20123463
6. Parmar, R. (2018, September 11). Training Deep Neural Networks. Medium. https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964
7. Li, F.; Shirahama, K.; Nisar, M.A.; Köping, L.; Grzegorzek, M. Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors. *Sensors* **2018**, *18*, 679. https://doi.org/10.3390/s18020679
8. Gureja, S. (2023, April 25). LSTMs and Bi-LSTM in Pytorch. Scaler Topics. https://www.scaler.com/topics/lstm-pytorch/
9. https://www.tensorflow.org/tutorials/generative/autoencoder
10. Badr, W. (2019, July 1). Auto-encoder: What is it? and what is it used for? (part 1). Medium. https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726
11. Brownlee, J. (2019, August 14). CNN long short-term memory networks. MachineLearningMastery.com. https://machinelearningmastery.com/cnn-long-short-term-memory-networks/
12. Dataman, C. K. (2023, January 30). Convolutional autoencoders for image noise reduction. Medium. https://towardsdatascience.com/convolutional-autoencoders-for-image-noise-reduction-32fce9fc1763
13. [Li, F.; Shirahama, K.; Nisar, M.A.; Köping, L.; Grzegorzek, M. Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors. Sensors 2018, 18, 679. https://doi.org/10.3390/s18020679](https://www.mdpi.com/1424-8220/18/2/679)
14. [https://d2l.ai/chapter\_recurrent-modern/lstm.html](https://www.mdpi.com/1424-8220/18/2/679)
15. Xiang, Jinyong & Qiu, Zhifeng & Hao, Qihan & Cao, Huhui. (2020). Multi-time scale wind speed prediction based on WT-bi-LSTM. MATEC Web of Conferences. 309. 05011. 10.1051/matecconf/202030905011
16. Dertat, A. (2017, October 8). Applied deep learning - part 3: Autoencoders. Medium. https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798
17. Matthew Stewart, P. (2023, February 10). Comprehensive introduction to autoencoders. Medium. https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368
18. Mrgrhn. (2021, January 25). Convolutional autoencoders (CAE) with tensorflow. plainenglish.io/blog/convolutional-autoencoders-cae-with-tensorflow-97e8d8859cbe. https://plainenglish.io/blog/convolutional-autoencoders-cae-with-tensorflow-97e8d8859cbe
19. Pintelas, E., Livieris, I. E., & Pintelas, P. E. (2021). A Convolutional Autoencoder Topology for Classification in High-Dimensional Noisy Image Datasets. Sensors (Basel, Switzerland), 21(22), 7731. https://doi.org/10.3390/s21227731
20. Jesuthasan, T. (2021, August 9). Autoencoders: A comprehensive guidew. Medium. https://tonyjesuthasan.medium.com/autoencoders-a-comprehensive-guide-d0723c2f988b
21. Boesch, G. (2023, March 16). Autoencoder in Computer Vision - complete 2023 guide. viso.ai. https://viso.ai/deep-learning/autoencoder/
22. Snover, Dylan & Johnson, Christopher & Bianco, Michael & Gerstoft, Peter. (2020). Deep Clustering to Identify Sources of Urban Seismic Noise in Long Beach, California. Seismological Research Letters. 92. 10.1785/0220200164
23. Li, F.; Shirahama, K.; Nisar, M.A.; Köping, L.; Grzegorzek, M. Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors. Sensors 2018, 18, 679. https://doi.org/10.3390/s18020679