Activity recognition based on the wrist-worn Axivity AX3

Master's Project

He Liu, Siqi Bao
20-743-506, 20-745-634

Submitted on
May 30, 2023

Thesis Supervisor
Prof. Dr. Manuel Günther

Co-supervisor
Prof. Dr. Christian Holz
Max Möbus

Master's Project

Author:	He Liu, Siqi Bao, he.liu@uzh.ch, siqi.bao@uzh.ch
Project period:	January 15, 2023 - May 30, 2023
Keywords:	human activity recognition; accelerometer

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich
Andreasstrasse 15
8050 Zurich

Sensing, Interaction & Perception Lab
Department of Computer Science, ETH Zurich
Universitätstrasse 6
8092 Zurich

Acknowledgements

First and foremost, we would like to express our deepest gratitude to Prof. Dr. Manuel Günther, whose expertise and guidance have been invaluable throughout this project. His perceptive suggestions and keen insights have been instrumental in addressing previously unrecognized issues and shaping the direction of our research.

We are particularly indebted to Max Möbus, a diligent and dedicated PhD student at SIPLAB, ETH Zurich, who has tirelessly supervised this project. Our weekly meetings have provided a stimulating platform for discussion, learning, and progress. Max's active involvement, constructive feedback, and unwavering support have significantly enriched our experience and contributed to the success of this project. His commitment to our growth and understanding has been truly inspiring, and we are immensely grateful for his time and expertise.

Lastly, we want to thank our families and friends for their constant encouragement and support throughout this journey. Their belief in us has been a source of strength and motivation, and we dedicate this work to them.

Abstract

Wrist-worn accelerometers are increasingly employed for human activity recognition (HAR) in various medical contexts. However, existing algorithms often struggle to generalize to unseen datasets, particularly those recorded in real-life scenarios. This report investigates the performance of hierarchical machine learning (ML) models and multi-branch convolutional neural network (CNN) models for HAR, using multiple publicly available datasets to assess activity recognition and generalizability. We evaluate our models on the Capture24 dataset, which closely resembles real-life activities.

Our research questions and aims focus on evaluating the performance of hierarchical ML and multi-branch CNN models for HAR, understanding their generalizability to unseen datasets, and exploring the effects of transformation techniques on model performance. We employed two approaches for model construction: a hierarchical ML model and a multi-branch CNN model.

Our analysis reveals that, while the hierarchical ML model tends to outperform the CNN model in terms of F1 scores for the majority of activities, the efficacy of the models varies when considering individual activities. The CNN model displays a marked degree of generalizability, and hints at the potential for enhanced performance with the incorporation of more diverse datasets. Our study also underscores the importance of dataset selection, suggesting that datasets closely resembling the target dataset should be preferred. Furthermore, the choice of transformation techniques appears to be activity-specific, further underlining the need for nuanced decision-making when developing models for human activity recognition.

In conclusion, this report provides valuable insights into the performance of ML and CNN models for HAR, highlighting the advantages of hierarchical ML training structures and multi-branch CNN models. Our findings not only emphasize the strengths and limitations of each approach but also underscore the importance of considering the target dataset and the choice of transformation techniques when developing HAR models for real-world applications.

Contents

Chapter 1

Introduction

Wrist-worn accelerometers are increasingly employed for human activity recognition (HAR) in various medical contexts, including sleep studies, chronic disease management, and post-stroke recovery (Acampora et al., 2013). However, one major challenge faced by existing algorithms is their limited ability to generalize to unseen datasets, especially those recorded in realistic, everyday scenarios. While simple algorithms may perform exceptionally well on single, controlled datasets, their performance can significantly degrade when faced with real-life, complex scenarios. Therefore, it is essential to develop robust and generalizable activity classification techniques that can handle diverse, real-world situations.

In this project, we aim to develop activity classification techniques for the Axivity AX3 accelerometer, a widely used device in large-scale studies like the UK Biobank (Doherty et al., 2017). We use multiple publicly available datasets to assess the types of activities that can be recognized and the extent to which these techniques can be generalized to new datasets recorded in varying scenarios. We leverage five meticulously curated datasets, all recorded on various wrist-worn 3-axis accelerometers with sampling frequencies ranging from 20Hz to 256Hz. Our evaluation is conducted on the Capture24 dataset, which closely resembles real-life activities (Chan Chang and Doherty, 2021).

It is important to note that the test data in the Capture24 dataset represents more complex, real-life activities, while the training data employed in this study is relatively "artificial," being recorded in a laboratory or controlled environments. This distinction ensures that the resulting classification model is expected to work well in practice.

In recent years, deep learning techniques have gained significant attention for their ability to achieve remarkable performance in various domains, including HAR. However, conventional machine learning (ML) models still hold relevance because of their simplicity, interpretability, and lower computational requirements. In certain scenarios, these conventional models may provide comparable performance to deep learning models, while being more resource-efficient and easier to implement. Consequently, exploring the potential of hierarchical ML models for HAR remains an important research direction.

To achieve our goal, we construct a pipeline (as shown in Figure 1.1 and Appendix A.2) to compare two different approaches: a hierarchical machine learning (ML) model and a multi-branch convolutional neural network (CNN) model. The hierarchical ML model utilizes a top-down approach, where activities are classified at a high-level and then broken down into sub-activities with increasing specificity. In contrast, the multi-branch CNN model extracts different frequency components from the accelerometer data using parallel CNN branches, where each branch extracts features that correspond to different characteristics of the data, such as different frequency components, frequency or time-domain features.

Our results demonstrate that although the hierarchical ML model outperforms the CNN model in terms of F1 scores for most activities of interest and overall performance, the CNN model ex-

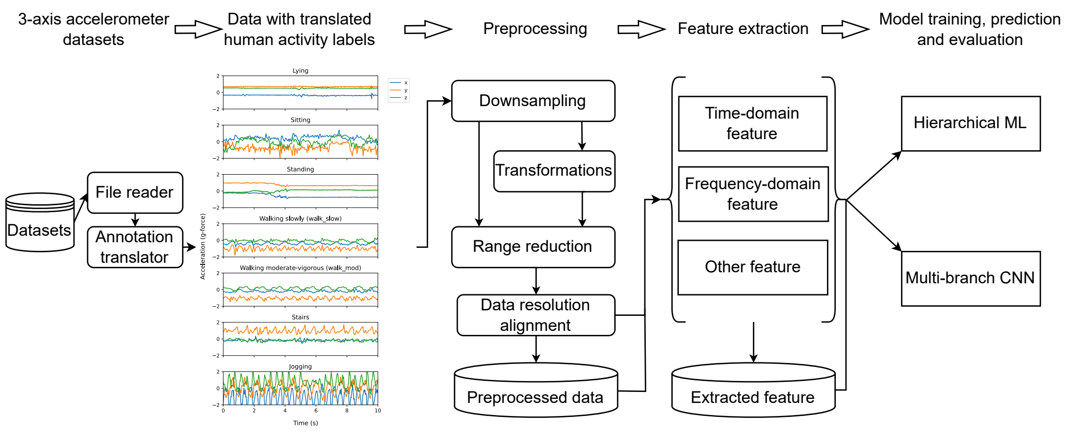


Figure 1.1: DATA PIPELINE.

hibits superior generalizability. Furthermore, the potential of CNN models for enhanced performance increases as more diverse datasets are incorporated. We also recommend that, when possible, the selection of datasets should be closer to the target usage scenario, and the transformation techniques should be tailored to the specific activities being investigated.

The remainder of this report is organized as follows. In the next chapter, we provide an overview of related work in the field of activity recognition, with a focus on wrist-worn accelerometers. Then, we describe our methodology, which encompasses data preprocessing steps, feature extraction techniques, and the ML and CNN algorithms used for activity classification. Next, we present and analyze the results of our experiments. Finally, we discuss the limitations of our study, suggest directions for future research, and conclude with a summary of our findings.

This project's primary contributions can be summarized as follows:

- We designated the Capture24 dataset for testing purposes and examined the performance of both hierarchical machine learning models and CNN models, aiming to optimize classification accuracy on Capture24 without prior training on the Capture24 dataset. This approach ensures that we gain a realistic understanding of the models' performance on unseen datasets.

- We explored four distinct transformation techniques to emulate various wearable sensor positions, enabling us to evaluate the adaptability of our models to diverse scenarios.

- We assessed the hierarchical training structure in ML training and the utilization of multiple branches in CNN models, concluding that these techniques can benefit HAR performance. We also determine that while conventional ML models may achieve better performance with small-scale and similar training data, CNN models offer greater generalizability and hold potential for further improvements.

The work distribution of this group project is detailed in Appendix A.1.

Chapter 2

Related Work

Real-world HAR based on wearable sensors

Wearable sensors, such as accelerometers and gyroscopes, have become a popular choice for capturing human movement data because of their low cost, small size, and ease of use (Yang et al., 2015). Various studies have used wearable sensors to recognize activities, including Banos et al. (2014); Shoaib et al. (2015); Wu et al. (2012). In our work, we utilize publicly available datasets, which were collected using 3-axis accelerometers from wrist-worn devices, allowing us to focus on the most commonly worn body position for activity recognition in everyday life.

It is worth noting that many existing HAR studies have been conducted in laboratory or controlled environments (Lara and Labrador, 2012; Bulling et al., 2014). While these studies provide valuable insights into the field, they may not fully capture the complexity and variability of real-world scenarios.

Data Preprocessing and Feature Extraction

Data preprocessing, such as resampling, windowing, and scaling, plays a crucial role in HAR, as they can significantly impact the performance of the recognition models (Garcia-Gonzalez et al., 2020; Bulling et al., 2014). To improve models' adaptability to various sensor positions and orientations, data transformation techniques are widely used several research fields, such as amplitude perturbation in speech recognition (Qin et al., 2019) and rotation perturbation in image classification (Lowe, 1999). Our research conducts several transformation techniques inspired by these approaches, aiming to enhance the generalizability of our models.

The choice of features also plays an important role. Statistical features, spectral features, geometric features, and temporal features are commonly employed in HAR literature (Bulling et al., 2014; Wang and Chen, 2009). In our study, we adopt some of these feature extraction techniques to represent the underlying patterns in the accelerometer data. We also explore the effect of various data transformations on the classification performance, aiming to understand how these techniques can be leveraged to improve the recognition of activities in real-world settings.

Machine Learning Methods for HAR

A wide range of machine learning algorithms has been applied to the problem of HAR, including Decision Trees, Support Vector Machines (SVMs), Random Forests (RF), and k-Nearest Neighbors (k-NN) (Garcia-Gonzalez et al., 2020; Bulling et al., 2014; Shoaib et al., 2015). Hierarchical approaches for HAR have been previously explored in the literature, with several studies demonstrating the benefits of grouping activities based on their similarities (Leutheuser et al., 2013). In our work, we employ a hierarchical ML training process that categorizes activities into groups based on their intensities, aiming to further improve the classification performance.

Deep Learning Methods for HAR

Deep learning methods have shown great potential in the field of HAR by identifying complex patterns in data, and Convolutional Neural Networks (CNN) have been widely applied in HAR, showing promising results(Wang et al., 2019). Noori et al. (2020a) explored various fusion methods for multi-representations of sensor data using CNNs, while Kalouris et al. (2019) implemented three CNN architectures for activity classification of older people. They also utilize data augmentation and cross-utilize knowledge about physical activity of younger persons to improve generalization. Both of the models are set as benchmarks in our study. Additionally, the multi-branch model of CNN and bidirectional LSTM proposed by Noori et al. (2020b) for wearable sensor data classification inspired us to build our multi-branch CNN model.

Chapter 3

Setup and Dataset

3.1 Environment Setup

In this project, we place great emphasis on creating a consistent and reproducible environment across multiple machines to facilitate seamless collaboration among team members. To achieve this, we utilize well-configured requirement files to establish a uniform Python environment on different machines, ensuring that all required libraries and dependencies are installed and maintained in a consistent manner.

In order to optimize the process of submitting jobs to the Slurm workload management system employed by the ETH Euler cluster, additional steps are necessitated beyond those required for running a Python script on a local machine. These additional procedures are detailed in Appendix A.3. We also curate comprehensive documentation within our repository This documentation elucidates the necessary steps for configuring and submitting the four types of jobs that are integral to our project: data preprocessing, feature extraction, hierarchical ML model training, and multi-branch CNN model training. The documentation accompanying each step includes a detailed description of the folder structure, a template file in YAML demonstrating configurable parameters, and an example bash command for submitting a corresponding Slurm job using `sbatch`. Our comprehensive guide empowers team members to effectively leverage the computing resources provided by the Euler cluster, thereby greatly enhancing the overall efficiency and productivity of our project.

Moreover, we adopt an object-oriented programming (OOP) approach and employ parameterization in our codebase to promote collaboration and simplify the process of integrating individual contributions. By structuring our code using OOP principles, we ensure that the code is modular, reusable, and easy to maintain, making it more accessible for team members to work on different components of the project simultaneously. Parameterization further facilitates the process of fine-tuning the models and experimenting with various configurations, enabling us to identify optimal settings for our activity recognition techniques.

In addition, we harness the power of Dask (Rocklin, 2015), a parallel computing library, in conjunction with parameterization to enable efficient local prototyping. By leveraging Dask, we are able to distribute computations across multiple cores or even clusters, significantly reducing the time required for model training and experimentation. This approach proves invaluable for rapid prototyping and testing of our models, allowing us to iterate and refine our techniques more effectively.

As our activity recognition pipeline is designed to be portable and capable of running on various machines, it is important to consider the recommended hardware specifications to achieve optimal performance and obtain results within an acceptable timeframe. We suggest using a machine equipped with at least 12 CPU cores or threads, 24GB of RAM, and 200GB of SSD storage to ensure efficient processing and data handling. Additionally, for GPU acceleration, we recommend an NVIDIA GPU with support for CUDA 11.8 or higher and at least 12GB of VRAM. Utilizing a machine with these specifications will enable our models to train and process data more effectively, ultimately resulting in more timely activity recognition using the Axivity AX3 accelerometer.

Furthermore, we employ GitHub as a version control and collaboration platform for our project. This allows team members to effectively share, review, and merge their code changes while maintaining a comprehensive history of the project's development. The use of GitHub further enhances the communication and collaboration within the team, ensuring that all contributions are seamlessly integrated into the codebase.

The elements in our activity recognition pipeline allow us to focus on refining our models and addressing the challenges associated with achieving higher accuracy in activity recognition using the 3-axis accelerometers. The specific details, including the various stages and techniques employed, will be presented in the following section. This comprehensive overview will provide a deeper understanding of the methodologies and tools used in the development of our activity recognition models for the Axivity AX3 accelerometer.

While we strive to ensure that our pipeline is scalable, efficient, and reusable, we acknowledge that opportunities for improvement, particularly in terms of processing speed, still exist. With this understanding, we maintain a balanced perspective on our progress and remain dedicated to the continual exploration of enhancements.

3.2 Dataset

In this project, we utilize six different Human Activity Recognition (HAR) datasets, five of which are used as training data, namely WISDM, Selfback, Gotov, Adl_hmp, Act_cp. The Capture24 dataset is used to evaluate the performance of models. It should be noted that the train data used in this project is collected in a controlled laboratory setting, while the test data comprises daily life activity data. To standardize the labels in different datasets, we translate the activities of all datasets to uniform labels, the activities and translations of each dataset are listed in A.5. Activities not included in our project are assigned as "unknown".

3.2.1 Description of Datasets

WISDM The WISDM (Wireless Sensor Data Mining) dataset (Weiss et al., 2019) is a widely used publicly available dataset for HAR. It consists of data collected from 51 subjects who performed 18 activities, each lasting for 3 minutes. During the data collection process, participants wore a smartwatch with a sensor at a rate of 20 Hz on their dominant hand.

SELFBACK The Selfback dataset (Sani et al., 2016) is an HAR dataset that contains data of nine different activities performed by 33 participants. The data was recorded with a tri-axial accelerometer sampling at a frequency of 100Hz, mounted on the participant"s dominant side wrist. Each participant performed an activity for approximately three minutes.

GOTOV The Gotov dataset (Paraschiakos et al., 2021) is a wearable sensor-based HAR dataset of physical activities for 35 healthy elderly individuals over 60 years old. The dataset contains data collected from different body locations and devices with a sampling rate of 88 Hz. We leverage the data from GeneActives accelerometer wearing at wrist. The 35 individuals followed a protocol of 16 activities of daily living for approximately an hour and a half in a semi-lab environment.

ADL_HMP The Adl_hmp (Activities of Daily Living with a Wrist-Worn Accelerometer) dataset (Bruno et al., 2013) consists of recordings of 14 simple activities of daily living, performed by 16 volunteers. The data were collected using a single tri-axial accelerometer attached to the right-wrist of the volunteer with 32 Hz.

ACP_CP The Act_cp dataset (Leotta et al., 2021) contains three-axial accelerometer, magnetometer, and gyroscope data recorded from different parts of the body while performing 17 different daily-life activities. The data were recorded using medical-grade devices at a high sampling frequency (up to 256Hz). The dataset includes data from eight healthy volunteers aged between 23-37. The subjects wore three devices, and we use the data recorded on a Actigraph Centrepoint wearing at the dominant wrist.

Capture24 The Capture24 dataset (Chan Chang and Doherty, 2021) contains Axivity AX3 wrist-worn activity tracker data collected from 151 participants in Oxford, UK, during 2014-2015. Participants were asked to wear the device in their daily lives for approximately 24 hours, resulting in a total of nearly 4,000 hours of data. The ground truth activities performed during this period were obtained using Vicon Autograph wearable cameras and Whitehall II sleep diaries, resulting in over 2,500 hours of labeled data. The labeled data were manually annotated by trained annotators using labels from the Compendium of Physical Activities based on the camera images and time use diaries. The Axivity AX3 device measures acceleration along three axes with a sampling rate of 100Hz.

3.2.2 Utilization of Data and Annotations

The datasets used in this study involve the use of accelerometers placed on various body parts. However, for consistency purposes, only data from the wrist have been included. It should be noted that the datasets comprise varying label sets. To reconcile these variations and maintain consistency, all original activity annotations have been translated into a unified set of eight labels: "lying," "sitting," "standing," "walk_slow" (representing slow-paced walking), "walk_mod" (signifying moderate to vigorous walking), "stairs" (covering both stair ascent and descent), "jogging," and "unknown". The mapping schema delineating the relationship between the original labels and these newly assigned labels for each respective dataset is documented in Appendix A.5. Additionally, to align with the hierarchical ML model delineated in Chapter 1, these labels have been categorized into higher-level groups based on their corresponding activity types. The structure of these groups and their affiliated labels is represented in Figure 3.1.

Notably, we have decided to exclude certain annotations from our training datasets that are ambiguous or potentially encompass multiple labels. Take, for example, "NA" in the Gotov dataset, which represents activities not included in their activity protocol. While these were deemed "unknown" in the context of Gotov"s study, they may include activities that are identifiable in our project, such as "jogging". Another instance of ambiguous labeling is "relaxing" in the Act_cp dataset, which denotes "relaxing on a chair". While this might initially seem akin to "sitting", the specific type of chair and the participant"s exact posture are unclear. Considering that a participant might be reclining or even lying down on the chair, classifying such an activity simply as "sitting" could lead to inaccuracies. Therefore, to maintain clarity and precision in our activity labeling, we"ve chosen to exclude such ambiguous annotations from our model training input.

The sample distributions for training and test datasets are displayed in Table 3.1.

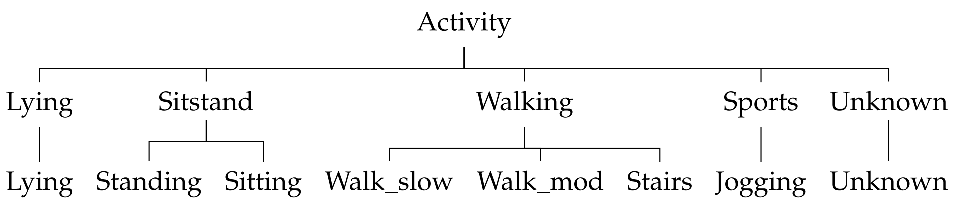


Figure 3.1: Hierarchical structure of activity groups and corresponding labels.

Table 3.1: Number of labels in each dataset with translation applied.

Label	**Dataset**					
	Selfback	Wisdm	Gotov	Adl_hmp	Act_cp	Capture24
Lying	1183	0	2433	0	0	680153
Sitting	1163	2108	3612	0	0	161795
Standing	1198	2142	774	0	0	25056
Walk_slow	1188	0	1219	0	0	97327
Walk_mod	2345	2083	2383	0	368	43483
Stairs	1002	2050	117	224	241	104
Jogging	1569	2036	0	0	0	1500
Unknown	0	20708	3657	687	3975	833140

Chapter 4

Data Processing and Features

In this chapter, we will introduce and discuss the pipeline for data preprocessing, transformation, and feature extraction, which are all vital aspects of activity recognition using the Axivity AX3 accelerometer. The datasets in our study are typically divided into separate files by participants. Our pipeline leverages Dask library to efficiently manage the loading and processing of these files from various training datasets into corresponding partitions or blocks, allowing for concurrent preprocessing while efficiently managing memory usage.

To ensure that concurrent processing does not exhaust the available RAM, we utilize Dask Delayed. This feature allows us to build a task graph that represents the entire computation, enabling Dask to execute the tasks in a well-ordered manner that optimizes the memory usage. As a result, our pipeline can efficiently process large volumes of data without overwhelming the system resources.

The processing of the datasets comprises a series of preprocessing steps and transformations. As illustrated in Figure 1.1, data are initially resampled to a uniform frequency, subsequently undergoing transformations to emulate various sensor orientations and conditions. Following these transformations, additional preprocessing techniques—including data range reduction and resolution alignment—are implemented to further refine the data.

Following the processing stage, partitions of the Dask dataframe are stored as parquet files, serving as the basis for feature extraction. Feature extraction operations are executed concurrently on these parquet files, with results subsequently stored within expandable HDF5 files. In line with the methodology employed by Yuan et al. (2022), we utilize a window length of 10 seconds and a hop length of 5 seconds for feature extraction in this project. The features extracted include raw data, time-domain features and frequency-domain features, along with additional features such as distance matrices. A more detailed discussion of these features can be found in Section 4.3.

Upon completion of the feature extraction process, the output (X, Y) is prepared for training. The input matrix X has a shape of $(N, 3,$ number of features$)$, where 3 is the number of axes and N represents the number of windows, or training samples. The output matrix Y has a shape of $(N, 3)$, which comprises original annotation, group, and label, as introduced in Section 3.2.2.

By applying these preprocessing, transformations, and feature extraction techniques, our pipeline effectively addresses the complexities and challenges associated with processing wrist-worn accelerometer data, thus laying a solid foundation for subsequent HAR tasks. Detailed descriptions and evaluations of these techniques, including their impact on model performance, will be provided in the subsequent chapters and sections.

4.1 Data Preprocessing

The preprocessing stage is an essential part of any data-driven project. In this section, we discuss the preprocessing techniques applied to the raw accelerometer data to prepare it for the subsequent stages of the pipeline. The primary preprocessing techniques used in this project are data loading, annotation translation, resampling, and range reduction. Note that transformation is also an important preprocessing technique employed in this project, but as a kind of data augmentation, it will be discussed in Section 4.2.

4.1.1 Data Loading

The initial step in the preprocessing pipeline is data loading and annotation translation. Loading the raw accelerometer data involves reading one or more CSV files and converting them into a Dask dataframe for efficient processing. Annotation translation ensures that the activity labels from multiple datasets are consistent, enabling their use in training and evaluation of the activity recognition models.

The raw accelerometer data in this project is typically stored in CSV files. We load them into a Dask dataframe using the `read_csv` function. The raw data typically contains several columns, such as timestamp, x, y, z, annotation, and participant. However, some columns may not be necessary for further processing, so we only preserve the relevant columns. After loading the data into a Dask dataframe, we map the annotation to group and label, and convert the unit of the accelerometer data to the gravity acceleration g if it is not already in that unit. Furthermore, to distinguish the records from various sources, we encode the name of the dataset and the path to the CSV file using `UUID version 7` and store this information in the dataframe as well.

4.1.2 Resampling

Resampling is a critical preprocessing step in accelerometer-based activity recognition, as various datasets may have diverse sampling rates. A consistent sampling rate across all datasets is essential for accurate feature extraction, comparison, and model training in the subsequent stages of the pipeline. In this project, we opt to resample the data from all three axes to 20 Hz using the `librosa.resample` function.

The choice of 20 Hz as the target sampling rate is based on the observation that human activities typically have frequencies below 10 Hz (Mannini and Sabatini, 2010). According to the Nyquist-Shannon sampling theorem, the sampling rate should be at least twice the highest frequency component in the signal to avoid aliasing artifacts (Shannon, 1949). Therefore, a sampling rate of 20 Hz is sufficient to capture the majority of the frequency content present in human activity data while minimizing the computational burden.

The resampling process can be performed using various methods, each with its trade-offs between computational efficiency and the preservation of the original signal's characteristics. In this project, we use the `soxr_hq` (high-quality) method from the SoX Resampler library for band-limited sinc interpolation. This method was chosen for its ability to provide high-quality resampling while maintaining a reasonable level of computational efficiency. The `soxr_hq` method utilizes the sinc function, which is an ideal interpolation function in the frequency domain, to reconstruct the signal at the desired sampling rate (Smith, 2011). The band-limited aspect of the method ensures that the resampled signal does not introduce any unwanted frequency components that may distort the original signal's characteristics.

4.1.3 Range Reduction

Human daily activities typically yield accelerations within the range of $-2g$ to $2g$ (Liu et al., 2009). As such, data points falling outside of this range may be interpreted as outliers representing activities conducted with either temporary or sustained extreme intensity. In a bid to mitigate overfitting and enhance the generalizability of our model, we impose a clipping mechanism to confine the data from all three axes within this specified range.

4.1.4 Data Resolution Alignment

Quantization error may influence model accuracy, particularly when the model incorporates multiple accelerometers with varied resolutions (Yoo et al., 2023). One potential strategy to attenuate this quantization error is to construct a linear space between predetermined minimum and maximum values, and subsequently map the data onto this space. However, this method can impose significant computational burdens, particularly with expansive datasets. A similar challenge arises in relation to the resolution refinement method proposed by Yoo et al. (2023).

In the context of this project, we have elected for a simpler and more computationally efficient approach, rounding the data to the nearest thousandth decimal place, to achieve resolution uniformity across data. We anticipate that minor variations within accelerometer data will likely have negligible effects on overall activity recognition. This method, by virtue of its enhanced speed compared to alternative approaches, is well suited to processing large datasets in a computationally efficient manner.

4.2 Transformation

In this section, we detail the distinct transformations applied to the preprocessed accelerometer data, which are designed to emulate different sensor positions and conditions. Such transformations serve a crucial role in evaluating the generalizability of our models to new datasets and diverse scenarios. Our methodology encompasses four principal types of transformations: swapping axes, reverting axes, amplitude perturbation, and rotation perturbation. Each transformation is executed on a window-by-window basis to guarantee consistent application across an entire window. Although our pipeline is designed to accommodate combined transformations, we have elected to perform these transformations separately due to computational resource limitations. Regardless of the specific transformation employed, it should be noted that untransformed data is always incorporated into downstream tasks.

Swapping Axes

Swapping the x and y axes involves interchanging the x and y values of each data point in the accelerometer readings. Mathematically, this transformation is represented as (4.1).

$$(x, y, z) \to (y, x, z) \tag{4.1}$$

This transformation allows our model to adapt to varying accelerometer orientations, a variability arising from differing sensor designs encountered in real-world usage. While there is no standardized positioning of the x and y axes due to the diverse range of wrist-worn accelerometer designs and potential variations in how users wear the device, the z-axis is typically oriented wrist-outwards to ensure a consistent measurement of vertical acceleration (Yuan et al., 2022). Consequently, we excluded the z-axis from this transformation, focusing on the x and y axes to account for potential inconsistencies in their positioning. In this way, the model becomes more robust to variations in sensor position (Qureshi and Golnaraghi, 2017).

Reverting Axes

Reverting the axes consists of multiplying the x, y, and z values by -1, with all possible combinations of the axes being considered. This results in a total of 8 different possibilities, which can be mathematically represented as (4.2).

$$(x, y, z) \rightarrow (\pm x, \pm y, \pm z) \tag{4.2}$$

Reverting the axes simulates changes in the sensor's orientation that may occur because of different wrist movements, mounting positions, or sensor specifications (Jain and Kanhangad, 2015).

Amplitude Perturbation

Amplitude perturbation is inspired by the concept of data augmentation, which is widely employed in various fields, including speech recognition (Qin et al., 2019). The objective of this method is to introduce minor variations into the data, thereby enhancing the model's robustness to real-world signal fluctuations. Such fluctuations might be induced by factors such as sensor calibration discrepancies, changes in battery status, or external interferences. In this project, we apply this idea to accelerometer data by multiplying the data by a random factor k sampled from a Gaussian distribution $\mathcal{N}(1, 0.1)$. Mathematically, this can be represented as (4.3).

$$(x, y, z) \rightarrow (kx, ky, kz) \tag{4.3}$$

Rotation Perturbation

To enhance the model's robustness against variations in sensor rotation, rotation perturbation is often utilized as a data transformation technique in the field of HAR (Heng et al., 2016; Tang et al., 2020). Essentially, rotation perturbation involves rotating the accelerometer vector by a random angle in the range of 1 to 5 degrees, while the magnitude remains constant. Given an original vector $\boldsymbol{v} = (x, y, z)$, the rotated vector $\boldsymbol{v'}$ can be calculated using a rotation matrix $\boldsymbol{R}$, as indicated in (4.4).

$$v' = Rv \tag{4.4}$$

The rotation matrix $\boldsymbol{R}$ can be generated using any axis of rotation and a random angle θ sampled from the uniform distribution between 1 and 5 degrees. This transformation aims to improve the model's robustness to slight variations in sensor orientation that may occur during everyday activities.

4.3 Feature Extraction

4.3.1 Description of Features

In this project, we extract a variety of features from the raw accelerometer data to provide a comprehensive representation of the underlying activities. These features, calculated on each window basis, are extracted from the raw data using various methods. Some features extract combined features from each axis, resulting a size of 1 in the first dimension. To enable concatenation and storage of the features together, the first dimension of all features (except the distance matrix) is padded to a size of 3.

Time-domain Features

Statistical features capture the basic properties of the accelerometer data and provide insights into the overall activity patterns.

- **Magnitude of windows (padded)** [3×200]**:** The magnitude of each window is computed as the Euclidean norm for each sample within the window.
- **Distance matrix of magnitude** [200×200]**:** A distance matrix of magnitude D contains the distance between each pair of points. It's calculated as (4.5), where x_i, x_j are the magnitudes of each data points in a window (Noori et al., 2020a). This feature is only used in one of the CNN benchmark models with shape of 200×200. Therefore, we calculate it separately for training this benchmark model.

$$D_{i,j} = \|x_i - x_j\| \tag{4.5}$$

- **Filtered accelerometer data of windows** [3×200]**:** The filtered accelerometer data of each window is computed after applying a low-pass, a high-pass, and a band-pass filter to the magnitude with Butterworth filters. The butterworth filters are implemented using the `scipy.signal.butter` function from the SciPy library (Virtanen et al., 2020). The filtered data was then obtained using the `scipy.signal.sosfiltfilt` function. The cut-off frequencies for the axis filters were set to 30% and 70% of the Nyquist frequency, corresponding to 3 Hz and 7 Hz, respectively, given the sampling rate of 20 Hz. The selection of these frequencies was determined through experimentation. To better understand the filter's frequency components, examples of the Butterworth filters used in our model is provided in the Appendix A.4.
- **Filtered magnitude of windows (padded)** [3×200]**:** The filtered magnitude of each window is computed after applying a low-pass, a high-pass, and a band-pass filter to the magnitude with Butterworth filters. Unlike the magnitude filter, the cut-off frequencies for filters are determined through experimentation with various combinations of frequencies. The cut-off frequencies of the filters for magnitude are set to 20% and 50% of the Nyquist frequency, resulting in cut-off frequencies of 2 Hz and 5 Hz, respectively, given the sampling rate of 20 Hz.
- **Minimum, maximum, and average amplitude of windows per axis** [3×3]**:** For each window, we calculate the minimum, maximum, and average amplitude for each axis, providing a sense of the range and central tendency of the accelerometer readings.
- **Standard deviation of amplitude of windows per axis** [3×1]**:** The standard deviation of the amplitude for each axis within a window is computed, which captures the variability of the accelerometer readings.
- **Average resultant acceleration (ARA) (padded)** [3×1]**:** The ARA is the average of the square root of the sum of the squares of the x, y, and z axis values. It provides a measure of the overall acceleration magnitude within a window. It's calculated as (4.6), where N is the

number of samples within the window, and x_i, y_i, and z_i are the accelerometer readings for each axis at sample i (Weiss et al., 2019).

$$ARA = \frac{1}{N}\sum_{i=1}^{N}\sqrt{x_i^2 + y_i^2 + z_i^2} \tag{4.6}$$

- **Binned distribution and histogram per axis** $[3 \times 20]$**:** The distribution of the magnitude of the windows and the individual axis values are divided into N equal-sized bins (with $n_{bins} = 20$) for each axis. The number of samples falling into each bin is counted to create a distribution. This captures the overall distribution and frequency of the signal amplitudes (Weiss et al., 2019).

- **Time between peaks** $[3 \times 1]$**:** The time between the first two peaks of the acceleration signal is calculated. This feature provides insight into the periodicity and regularity of the activities being performed (Weiss et al., 2019).

Frequency-domain Features

Spectral features are derived from the frequency domain representation of the accelerometer data, providing insights into the frequency content and energy distribution of the activities.

- **Mel-frequency cepstral coefficients (MFCCs)** $[3 \times 13]$**:** Mel-frequency cepstral coefficients (MFCCs) are widely used in the field of audio processing and speech recognition due to their ability to mimic the human auditory system's nonlinear perception of pitch and frequency. More recently, MFCCs are utilized in Human Activity Recognition (HAR) as they offer a compact representation of the power spectrum of a signal, capturing temporal patterns in sensor data effectively (Cruciani et al., 2020; Ramanujam et al., 2021). Computation involves dividing the signal into small frames, calculating the power spectrum of each frame via a Fast Fourier Transform (FFT), mapping frequencies onto the Mel scale, taking logarithm of powers at each Mel frequency, and applying a Discrete Cosine Transform (DCT). This project uses a set of 13 MFCCs for each time frame as a feature.

- **Spectral centroid** $[3 \times 1]$**:** The spectral centroid represents the center of mass of the power spectrum and can be considered as a measure of the "brightness" of a signal (Klapuri and Davy, 2006). For accelerometer data, it provides an indication of the dominant frequency content of the activities.

- **Spectral bandwidth** $[3 \times 1]$**:** The spectral bandwidth represents the spread of the power spectrum around the spectral centroid. It provides information about the spectral shape and the range of frequency components present in the signal (Klapuri and Davy, 2006). In the context of accelerometer data, it can help distinguish activities with different frequency patterns.

The features described above provide a comprehensive representation of the accelerometer data, capturing various aspects of the underlying patterns and structures present in the signal. The time domain features capture intensity, range information, and statistical measures of signal amplitude, while also considering the filtered frequency components. On the other hand, the frequency domain features capture the distribution of activity frequencies. By combining features from both domains, we achieve an effective characterization of activity patterns across the time and frequency dimensions.

4.3.2 Feature Extraction Pipeline

The feature extraction pipeline is a crucial part of the human activity recognition process, as it transforms the raw accelerometer data into a more informative and compact representation that can be effectively utilized by machine learning models. In this project, we have designed a highly efficient and scalable pipeline for extracting a wide range of statistical and spectral features using the Dask parallel computing library. This section provides a detailed description of the feature extraction pipeline, highlighting its key components and the strategies employed to ensure optimal performance.

Data Partitioning and Storage

The first step in the feature extraction pipeline involves reading the preprocessed accelerometer data from parquet files. To facilitate efficient data processing and avoid out-of-memory issues, the data is partitioned by participants and split into four-hour intervals. This partitioning strategy enables the pipeline to process large volumes of data by loading only a small portion of the data into memory at a time, thereby reducing the overall memory footprint.

The extracted features are then saved to an extendable `HDF5` container, partition by partition. The `HDF5` format is a high-performance, flexible, and portable binary data format that supports efficient I/O operations and can store large, complex, and heterogenous datasets with ease. By adopting the `HDF5` format, we ensure that the feature extraction pipeline can scale effectively to handle large-scale human activity recognition tasks, while also facilitating seamless interoperability with other data processing tools and machine learning frameworks.

Dask Delayed Interface

To further enhance the efficiency and scalability of the feature extraction pipeline, we leverage the `dask.delayed` interface, which allows us to create task graphs directly with a light annotation of normal Python code without triggering the computation immediately. The `dask.delayed` interface provides a simple and flexible way to build and execute complex, multi-stage computation pipelines with minimal overhead and maximal parallelism.

In our pipeline, we use the `dask.delayed` interface to define the feature extraction tasks, including the computation of various statistical and spectral features from the raw accelerometer data. By encapsulating these tasks as delayed objects, we can build a task graph that represents the entire computation pipeline, from data loading to feature extraction and storage, without actually executing any computation. This approach allows us to reason about and optimize the pipeline more effectively, as well as to schedule the tasks for parallel execution using Dask's advanced scheduling capabilities.

Dask Distributed Interface

Once the task graph has been constructed using the `dask.delayed` interface, the next step is to execute the tasks in parallel using the `dask.distributed` interface. The `dask.distributed` interface is a powerful and flexible system for parallel and distributed computing with Dask, providing a range of advanced features such as dynamic task scheduling, data locality, and fault tolerance. By leveraging the `dask.distributed` interface, we can efficiently parallelize the feature extraction pipeline across multiple CPU cores or even distributed clusters, thereby significantly reducing the overall processing time and improving the scalability of the pipeline.

In our implementation, we use the `dask.distributed` interface to schedule and execute the tasks defined in the task graph, ensuring that the tasks are distributed evenly across the available computational resources and executed concurrently whenever possible. This parallelization strategy enables the pipeline to process large volumes of data more quickly and efficiently, while also providing a high degree of fault tolerance and resiliency in the face of hardware or software failures.

Chapter 5

Model: Hierarchical ML

This chapter delves into the hierarchical ML models developed for our study. One motivation for investigating hierarchical ML models is the work of Leutheuser et al. (2013), which demonstrates that a hierarchical ML system for HAR can outperform a non-hierarchical ML system (called "flat" models). By categorizing activities based on their intensities and organizing them into activity groups, the authors were able to create a more effective classification system. Inspired by this approach, we aim to build a similar ML training pipeline for our study.

Drawing inspiration from the related works of Leutheuser et al. (2013), Piczak (2015), and Weiss et al. (2019), we employ the following features introduced in Section 4.3 in this chapter:

- Minimum, maximum, and average amplitude of windows per axis
- Standard deviation of amplitude of windows per axis
- ARA
- Binned distribution per axis
- Time between peaks
- MFCCs
- Spectral centroid
- Spectral bandwidth

To expedite the training and prediction processes and ensure timely generation of results, we employ CUDA-accelerated implementations for the models in this chapter through the `cuML` library.

To evaluate the performance of the implemented models on our dataset, we utilize precision, recall, and F1 score metrics per class in a one-vs-rest (OvR) manner. This evaluation provides a detailed understanding of the model performance for each activity. Additionally, we calculate the macro-average F1 scores (macro-F1) and Matthews correlation coefficient (MCC) to provide a single-value metric for overall model performance.

This chapter presents the performance evaluations of flat ML model, followed by an in-depth discussion on the hierarchical ML model, which includes its structure, methodology, and results. Finally, we will highlight the key outcomes of our experiments and discuss possible avenues for improvement.

5.1 Baselines

This section seeks to evaluate the performance of two commonly implemented machine learning models, Support Vector Machine (SVM) and Random Forest (RF), in the context of HAR using

Table 5.1: PARAMETERS USED IN ML HYPERPARAMETER TUNING.

Model	Parameters	Candidate(s)	Description
SVM	C	0.1, 1.0, 10, 100	The regularization strength is inversely proportional to C.
	penalty	l2	This refers to the norm used in penalization. In this study, we only consider the L2 regularizer.
	loss	hinge, squared_hinge	Specifies the cost function to be minimized during optimization. We use hinge loss ("hinge") and squared hinge loss ("squared_hinge").
	max_iter	5000	The maximum number of iterations to be run.
RF	n_estimators	100, 300, 500	The number of trees in the forest.
	max_depth	16, 32, 64	The maximum depth of the tree.
	max_features	1.0, sqrt	Determines the number of features to consider when deciding on the best split. A value of 1.0 considers all features; "sqrt" considers the square root of the total number of features.

sensor data. The selection of these models is underpinned by their robust performance in a range of classification problems, including HAR, as shown in previous studies (Bao and Intille, 2004; Gjoreski et al., 2011). Subsequently, we will explore the potential of enhancing model performance by implementing an ensemble model using a soft voting mechanism (Kittler et al., 1998). In this approach, the probabilities for each class produced by the constituent models are averaged to ascertain the final predicted class.

SVMs are a class of supervised learning models originally designed for binary classification problems. However, they can be extended to handle multi-class classification tasks by employing techniques like one-vs-one (OvO) or one-vs-rest (OvR) approaches (Cortes and Vapnik, 1995). SVMs have been extensively used in HAR tasks because of their ability to handle high-dimensional data and their robustness against overfitting (Garcia-Gonzalez et al., 2020). Although SVMs can effectively model complex decision boundaries by employing different kernel functions, due to the computational resources limitation we have, we only employ linear kernel.

On the other hand, RF is an ensemble learning method that constructs multiple decision trees and combines their outputs to improve classification performance (Breiman, 2001). RF is known for its robustness to noise, and resistance to overfitting (Liaw et al., 2002). These characteristics make RF a popular choice for HAR problems, as demonstrated by numerous studies in the literature (Kwapisz et al., 2011; Shoaib et al., 2014). Notably, RF is sensitive to class imbalance, an issue that could influence the model's performance. Considering the variations in class distributions across our training and test data (as illustrated in Table 3.1), we balance the classes by randomly undersampling the training input. This resampling method was chosen due to the computational resources and time constraints we have in our project.

Both SVM and RF models necessitate the optimization of several hyperparameters. To identify the optimal hyperparameter values, we employ `GridSearchCV` from the `scikit-learn` library, which conducts hyperparameter tuning over five stratified folds. The range of candidate values for the hyperparameters under consideration, along with other important parameters, are detailed in Table 5.1. Following this procedure, the parameter set yielding the highest Matthews correlation coefficient (MCC) is chosen for the training of the final classifier. For the specific hyperparameters selected, please refer to Table A.9a for selected hyperparameters.

The performance of the baseline models and their corresponding soft voting ensemble is illustrated in Table 5.2. Each of the models utilizes all five training datasets, along with the transfor-

Table 5.2: Performance of flat ML models. *P=Precision, R=Recall, F1=F1-score. The first 8 rows present the performance metrics for each activity individually, while the Macro Avg row represents the average values of activity-specific scores. The MCC row denotes the overall performance of the model.*

Activity	SVM			RF			SoftVoting(SVM, RF)		
	P	R	F1	P	R	F1	P	R	F1
Lying	0.7774	0.7104	**0.7424**	0.7525	0.5322	0.6235	0.7886	0.6988	0.7410
Sitting	0.1179	0.4266	0.1848	0.1094	0.5626	0.1831	0.1199	0.4734	**0.1913**
Standing	0.0386	0.1727	0.0631	0.0432	0.1839	**0.0700**	0.0419	0.1757	0.0676
Walk_slow	0.2858	0.0276	0.0504	0.3646	0.0319	**0.0587**	0.3258	0.0232	0.0432
Walk_mod	0.3538	0.4472	0.3951	0.4430	0.4994	**0.4695**	0.4194	0.4665	0.4417
Stairs	0.0006	0.1827	0.0011	0.0005	0.1442	0.0010	0.0006	0.1635	**0.0012**
Jogging	0.2554	0.9047	0.3984	0.3287	0.8653	**0.4764**	0.2835	0.9073	0.4320
Unknown	0.7403	0.3735	**0.4965**	0.7406	0.2939	0.4208	0.7393	0.3625	0.4865
Macro Avg	0.3212	0.4057	0.2915	0.3478	0.3892	0.2879	0.3399	0.4089	**0.3006**
MCC		0.3294			0.2639			**0.3306**	

mations detailed in Section 4.2. A review of the scores reveals that SVM generally outperforms RF in terms of both the macro-F1 score and MCC, whereas the soft voting of the two models outperforms each individually. However, performance fluctuates across different activities, and neither model exhibits consistently robust performance for all activity classes. Possible reasons for this outcome could be that the individual models in the ensemble have complementary strengths and weaknesses, which might not necessarily lead to a significant improvement in the overall performance. For comparative purposes, the soft voting ensemble, which boasts the highest macro-F1 score and MCC, is selected as the baseline in this chapter.

5.2 Methodology

In this section, we present the hierarchical training process for HAR. This methodology hinges on training multiple classifiers within a hierarchical structure, thus enabling the classification of activities at varying levels of granularity. As previously discussed, activities are grouped based on their intensities, illustrated in Figure 3.1.

Figure 5.1 provides a broad overview of the training procedure, with each classifier embodying the structure depicted in Figure 5.2. Each classifier performs a similar hyperparameters grid search as detailed in Section 5.1 (the selected hyperparameters can be found in Table A.9). For the hierarchical training, an initial base-layer model is trained to classify the activity groups. If a group consists of more than one label, a sub-layer model is subsequently trained to classify the labels within that particular group. For each group that contains more than one label, a sub-layer model is trained to classify the labels within that group, respectively. Both the group classifier and the label classifiers can be composed of a soft voting ensemble model, similar to the approach used in baseline models.

During the prediction phase, the data is initially processed by the group classifier to predict the group and its associated probability. If the predicted group contains multiple labels, the data is subsequently fed into the corresponding label classifier to determine the final activity label and its probability.

By employing an ensemble of classifiers and adopting a soft voting strategy, our objective is to enhance the performance of the hierarchical training process. This methodology facilitates a more comprehensive interpretation of the inherent patterns in the data. It also enables more accurate

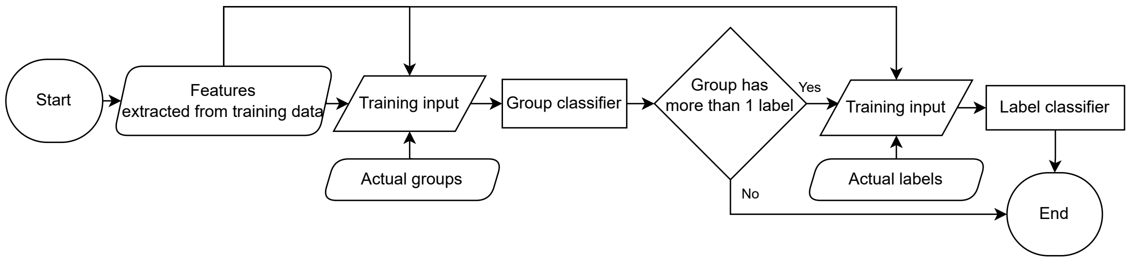


Figure 5.1: OVERVIEW OF THE HIERARCHICAL ML TRAINING STRUCTURE. *The process begins with a base-layer model that classifies the activity groups. For each group containing more than one label, a sub-layer model is trained to further classify the labels within that group.*

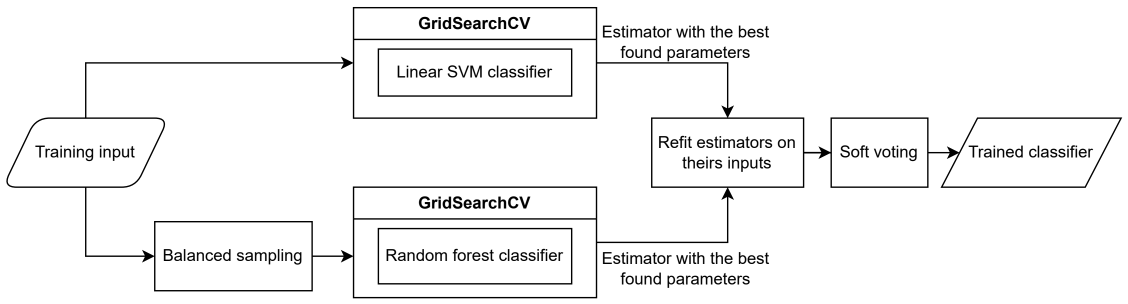


Figure 5.2: CLASSIFIER STRUCTURE USED IN THE HIERARCHICAL TRAINING.

predictions by capitalizing on the unique strengths of different classifiers.

5.3 Results

Table 5.3 demonstrates that the hierarchical training process improves the classification performance for "walk_slow," "walk_mod" and "jogging" activities, and has similar performance as flat ML model for "sitting" activity. However, it has slightly diminished performance on classifying the "lying" and "unknown" labels based on their OvR F1 scores. Neither model has managed to classify "stairs" effectively. Interestingly, labels with improved performance are within the "sit-stand" or "walking" group (refer Figure 3.1), while other activities (except "jogging") exhibit a decrease. This suggests potential avenues for further investigation into activity translation, feature selection, or transformation techniques for the group classifier.

The confusion matrix for the hierarchical ML model, depicted in Figure 5.3, shows that misclassifications primarily occur among "lying," "sitting," and "standing" activities, likely due to feature similarities causing model confusion. "Walk_slow" is mostly misclassified as "unknown," possibly due to its less distinct features. The low score and misclassification of "stairs" as "standing" imply possible feature overlap, ineffective model capture of unique characteristics, or variations in stair activity execution between training and testing datasets.

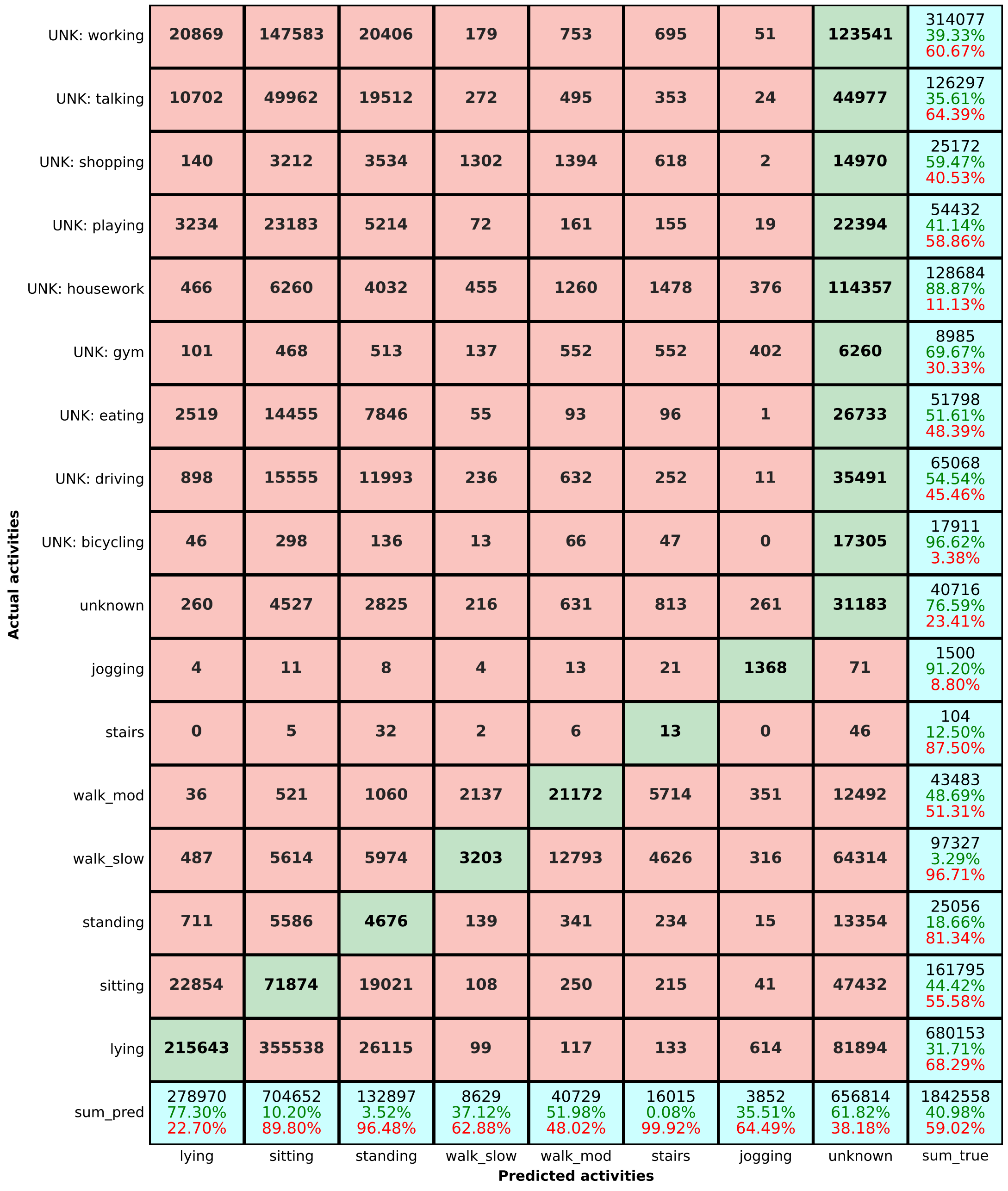
The model exhibits high recall but low precision for "jogging," decreasing confidence in results identified as this activity. As for unseen activities like "bicycling," "driving," "eating," "gym," "housework," and "shopping," the majority are correctly classified as "unknown," aligning with expectations. Most misclassifications for "driving," "eating," "talking," and "working" are as "sitting," likely due to stationary postures or limited movements during these activities. Further

Table 5.3: Performance Comparison of Flat and Hierarchical ML Models. *P=Precision, R=Recall, F1=F1-score. The first eight rows present the performance metrics for each individual activity. The Macro Avg row represents the average values of activity-specific scores. We use macro-F1 and MCC to represent overall performance.*

Activity	Flat ML			Hierarchical ML		
	P	R	F1	P	R	F1
Lying	0.7886	0.6988	**0.7410**	0.7525	0.5322	0.6235
Sitting	0.1199	0.4734	**0.1913**	0.1094	0.5626	0.1831
Standing	0.0419	0.1757	0.0676	0.0432	0.1839	**0.0700**
Walk_slow	0.3258	0.0232	0.0432	0.3646	0.0319	**0.0587**
Walk_mod	0.4194	0.4665	0.4417	0.4430	0.4994	**0.4695**
Stairs	0.0006	0.1635	**0.0012**	0.0005	0.1442	0.0010
Jogging	0.2835	0.9073	0.4320	0.3287	0.8653	**0.4764**
Unknown	0.7393	0.3625	**0.4865**	0.7406	0.2939	0.4208
Macro Avg	0.3399	0.4089	**0.3006**	0.3478	0.3892	0.2879
MCC		**0.3306**			0.2639	

investigation and improvements are, however, required to distinguish these activities more effectively from the "sitting" category.

In conclusion, the results presented in this chapter demonstrate that the hierarchical training process can improve the classification performance for most of the activities, with some exceptions. However, there is still room for improvement, particularly in differentiating between activity groups and detecting unseen activities. Further investigation on the use of training datasetas and transformation techniques will be conducted in Section 7.1.



Actual activities	lying	sitting	standing	walk_slow	walk_mod	stairs	jogging	unknown	sum_true
UNK: working	20869	147583	20406	179	753	695	51	**123541**	314077 39.33% 60.67%
UNK: talking	10702	49962	19512	272	495	353	24	**44977**	126297 35.61% 64.39%
UNK: shopping	140	3212	3534	1302	1394	618	2	**14970**	25172 59.47% 40.53%
UNK: playing	3234	23183	5214	72	161	155	19	**22394**	54432 41.14% 58.86%
UNK: housework	466	6260	4032	455	1260	1478	376	**114357**	128684 88.87% 11.13%
UNK: gym	101	468	513	137	552	552	402	**6260**	8985 69.67% 30.33%
UNK: eating	2519	14455	7846	55	93	96	1	**26733**	51798 51.61% 48.39%
UNK: driving	898	15555	11993	236	632	252	11	**35491**	65068 54.54% 45.46%
UNK: bicycling	46	298	136	13	66	47	0	**17305**	17911 96.62% 3.38%
unknown	260	4527	2825	216	631	813	261	**31183**	40716 76.59% 23.41%
jogging	4	11	8	4	13	21	**1368**	71	1500 91.20% 8.80%
stairs	0	5	32	2	6	**13**	0	46	104 12.50% 87.50%
walk_mod	36	521	1060	2137	**21172**	5714	351	12492	43483 48.69% 51.31%
walk_slow	487	5614	5974	**3203**	12793	4626	316	64314	97327 3.29% 96.71%
standing	711	5586	**4676**	139	341	234	15	13354	25056 18.66% 81.34%
sitting	22854	**71874**	19021	108	250	215	41	47432	161795 44.42% 55.58%
lying	**215643**	355538	26115	99	117	133	614	81894	680153 31.71% 68.29%
sum_pred	278970 77.30% 22.70%	704652 10.20% 89.80%	132897 3.52% 96.48%	8629 37.12% 62.88%	40729 51.98% 48.02%	16015 0.08% 99.92%	3852 35.51% 64.49%	656814 61.82% 38.18%	1842558 40.98% 59.02%

Figure 5.3: Confusion Matrix for Hierarchical ML Model. *Numbers with a green background indicate the count of correctly classified samples. "Sum_pred" and "sum_true" represent column and row summaries, respectively. The green text in "sum_pred" denotes precision, while the green text in "sum_true" signifies recall.*

Chapter 6

Model: Multi-branch CNN

Deep learning has emerged as a promising approach for activity recognition tasks because of its ability to overcome the limitations of traditional methods. Convolutional Neural Networks (CNNs) are particularly powerful for this task since they are capable of identifying complex patterns in the data that are difficult to detect using traditional methods (Wang et al., 2019).

In this chapter, we present a comprehensive analysis of our proposed CNN model for activity recognition using accelerometer data. We begin by benchmarking two existing CNN models and evaluating their performance on our test dataset in Section 6.1. Next, we introduce our base multi-branch CNN model structure in Section 6.2.1, which involves incorporating accelerometer data with Butterworth filters of low-pass, high-pass, and band-pass using parallel CNN branches. Furthermore, we explore the potential for improving our model's performance by combining the predictions of our multi-branch CNN model with features extracted from each window in Section 6.2.1, which we refer to as the combined CNN. Our results suggest that the proposed CNN model outperforms the benchmark models on the test dataset, indicating the effectiveness of our proposed model.

To realize our model, we provide detailed explanations of the model architecture and training process. The training datasets used in our CNN model include WISDM, Selfback, Gotov, Adl_hmp, and Act_cp. We use 20% of the training datasets as validation to find the best performance, and Capture24 serves as the test dataset to evaluate performance. During the training process, we utilize the cross-entropy loss function to evaluate model performance. To prevent overfitting, we implement early stopping with a tolerance of 5 epochs and set the maximum number of epochs to 100.

6.1 Benchmarks

In order to evaluate the performance of our proposed CNN models, we use two established CNN models as benchmarks. The first benchmark model(Noori et al., 2020a) utilized accelerometer data recorded from a smartphone for human activity recognition. In their work, Noori et al. explored several methods of fusion for multi-representations of data from sensors. They generated multiple representations of sensor data and fused them using Deep Convolutional Neural Networks (CNNs) to achieve a great performance for the Context-Awareness via Wrist-Worn Motion Sensors (HANDY) dataset and the Wireless Sensor Data Mining (WISDM version 1.1) dataset, respectively.

The second benchmark model was trained on datasets that included accelerometer, gyroscope, and optionally magnetometer data(Kalouris et al., 2019). Their work focused on activity classification for older people, who are often difficult to obtain a large number of labeled samples from.

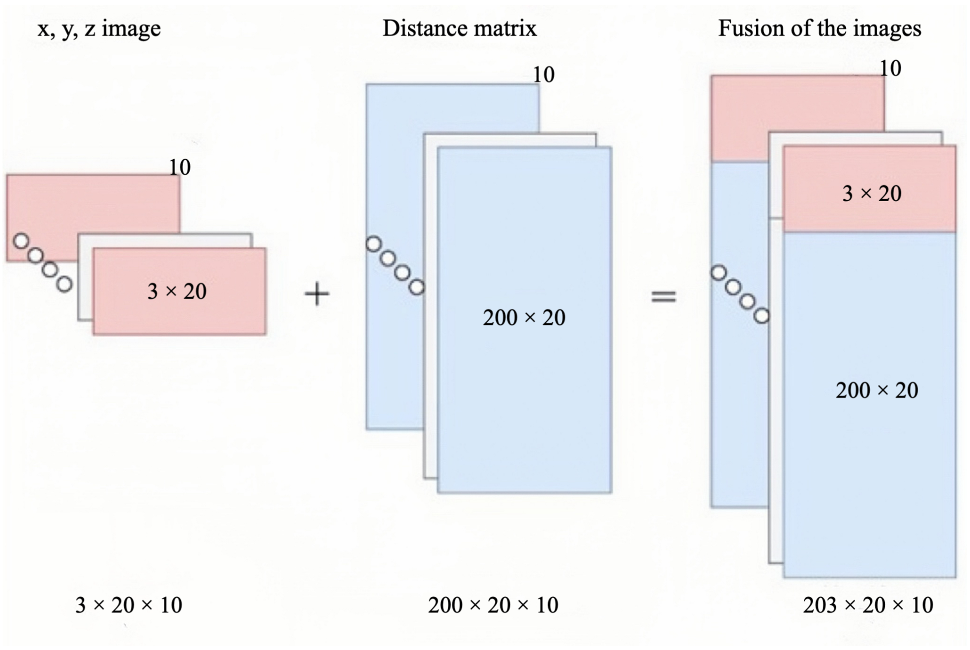


Figure 6.1: THE DATA-LEVEL FUSION STRUCTURE OF THE CNN MODEL IN BENCHMARK 1.

They implemented three different CNN architectures and incorporated Bayesian optimization for efficient hyper-parameter tuning.

To evaluate the performance of our proposed models, we compare them to the two established CNN models in the papers mentioned above and served as benchmarks. However, to ensure comparability with our datasets, we make some adjustments to the benchmark models. Firstly, we use our own datasets, which consist of accelerometer data collected from wrist-worn wearable devices, unlike the benchmark models that used data from smartphones and multiple sensors. Additionally, we adjusted the input shape of models to recognize the activities that were present in our datasets, which may differ from those in the benchmark models.

Benchmark 1

Compared to the smart-watch data used in our project from the WISDM dataset, Noori et al. (2020a) collected data from a smartphone-based accelerometer by instructing participants to carry the phone in their pocket. They sampled the accelerometer with 5-second windows (i.e., 100 data points with a sample rate of 20 Hz) with 25% overlapping.

The authors experimented with two different representations of accelerometer data: a distance matrix and x, y, and z images. These representations were used in fusion at the input level. The distance matrix representation involved calculating the pairwise Euclidean distances between the magnitudes of accelerometer samples, resulting in a 100×100 matrix. The x, y, and z image representation is the raw accelerometer data from each axis with a shape of 3×100. After stacking the distance matrix and x, y, z images into shape of $100 \times 10 \times 10$ and $3 \times 10 \times 10$ respectively to maintain a consistent depth, both representations were stacked, and a new representation was extracted, i.e., $103 \times 10 \times 10$ as input of the model as shown in Figure 6.1.

The CNN architecture utilized in their study included four convolutional layers, two max-pooling layers, and two dropout layers (with $p = 0.25$). The first two convolutional layers had

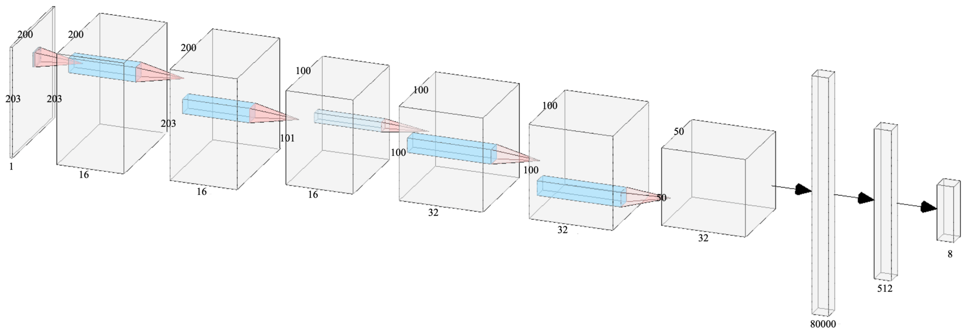


Figure 6.2: THE STRUCTURE OF THE CNN MODEL IN BENCHMARK 1.

16 kernels each, while the last two had 32 kernels each. Each convolutional layer used a kernel of size 3 and a stride size of 1, and ReLU activation was applied to both convolutional and fully connected layers. After passing through the convolutional and max-pooling layers(with max pooling size of 2), the data was flattened and fed into a fully connected layer with 512 units and dropout ($p = 0.50$). Finally, a classification layer with 10 units, corresponding to the ten activities, and a softmax activation function were used for the final classification.

In our project, the data is segmented into 10-second intervals with a sampling rate of 20 Hz, resulting in 200 data points for each input. To adapt the CNN model structure to our datasets, the representations are calculated as a matrix of dimensions $200 \times 20 \times 10$ and $3 \times 20 \times 10$, and concatenated to form an input of shape $203 \times 20 \times 10$. To achieve stable performance with given structure and hyper-parameters, we randomly select five combinations of batch size and learning rate. Among the different combinations, the batch size of 64 and learning rate of 0.0001 yield the lowest validation loss. The model with optimal hyper-parameters is selected to evaluate the performance of the CNN model on test dataset.

Upon reaching the optimal model, the training process of first benchmark model complete 50 epochs. The training loss achieved is 1.595 and the validation loss obtained is 1.624, where there's no evidence of overfitting.

As shown in Table 6.1, despite the high accuracy of the performance in their dataset, the model achieved an MCC score of 0.1707 on the test set of Capture 24. The results of the benchmark model 1 reveal that the performance of the model is quite low. The model's F1-scores for activities such as jogging, stairs, and standing are close to zero, indicating that the model is unable to distinguish these activities accurately. The model's performance is relatively better for activities such as lying, unknown, and walking_mod, with F1-scores of 0.5827, 0.2380, and 0.3081, respectively. The weighted average F1-score of the model is only 0.1962, which indicates that the model is not performing well across all activities.

Benchmark 2

Kalouris et al. (2019) utilized three distinct CNN architectures trained on data from elderly individuals that included accelerometer, gyroscope, and magnetometer data. While CNN2 and CNN3 operate on 2D convolutional layers, utilizing multiple sensing modalities by stacking N sensors (accelerometer, gyroscope, and optionally magnetometer), our project only considers accelerometer data. Therefore, we have chosen to apply the CNN1 architecture with 1D convolutional layers as a benchmark.

In their project, the authors used a time window of 68 time points with 50% overlap, corresponding to a total window duration of 2.72 seconds. The data was collected from three tri-axial

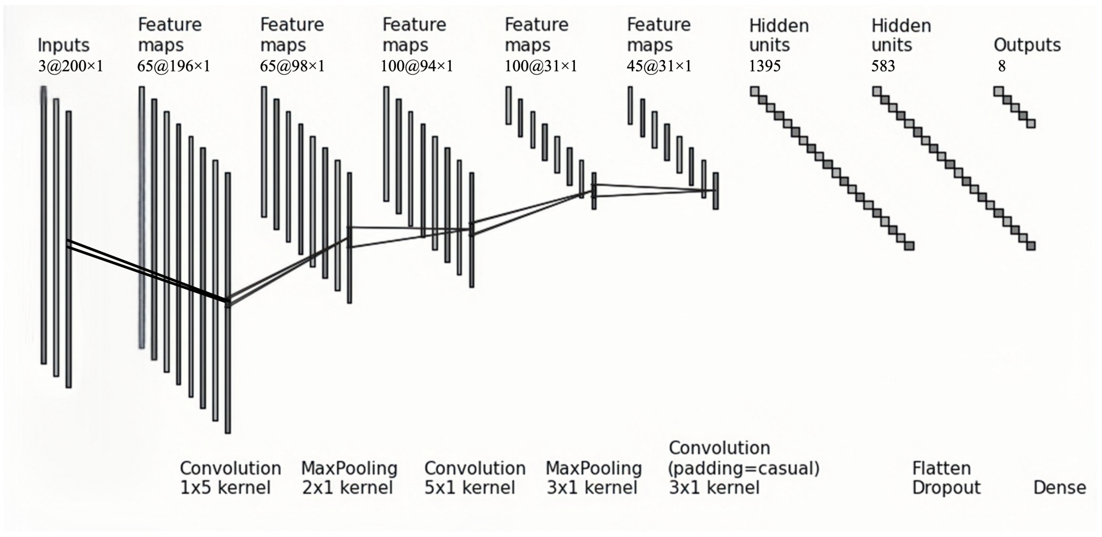


Figure 6.3: THE STRUCTURE OF THE CNN MODEL IN BENCHMARK 2.

sensors, resulting in an input size of 9×68. Figure 6.3 in their work depicts the CNN1 architecture.

The CNN architecture used in their study consists of three convolutional layers, two max-pooling layers, and one dropout layer. The first convolutional layer has 65 kernels, while the second convolutional layer has 100 kernels and the third convolutional layer has 45 kernels. The kernel size for the first and second convolutional layers is 5, while the kernel size for the third convolutional layer is 3. A stride size of 1 is used for all convolutional layers, and batch normalization and ReLU activation are applied after each convolutional layer. The first max-pooling layer reduces the spatial dimensions by a factor of 2 using a max-pooling size of 2, while the second max-pooling layer has a max-pooling size of 4. After passing through the convolutional and max-pooling layers, the data is flattened and fed into a fully connected layer with 583 units and dropout (with a dropout rate of 0.6). Finally, a classification layer with 5 output channels is used for the final classification. We adjust the input size(3×200) and output channels(8 classes) of benchmark 2 model to our dataset, using the same hyper-parameter tuning method employed for benchmark 1 model. This is done to obtain a reliable performance evaluation on our datasets.

The results are shown in Table 6.1, which reports the F1-score for each class, as well as the MCC and macro-averaged F1-score. The results show that the model performs well for some activities, such as lying, walk_mod and unknown, achieving F1-scores of 0.5051, 0.3810 and 0.3489, respectively. However, the model performs poorly for other activities, such as stairs and standing, achieving F1-scores of 0.0007 and 0.0468, respectively. The overall MCC score of the model is 0.1725, indicating that it is also not very effective at correctly classifying the activities. In conclusion, the benchmark model presented in Kalouris et al. (2019) achieved a moderate performance in recognizing activities using accelerometer data and the benchmark 2 model exhibits a slightly better performance than the benchmark 1 model. Therefore, we use the benchmark 2 model to compare with our CNN model.

6.2 Multi-Branch CNN Models

In this section, we introduce our base multi-branch CNN model and the combined CNN model, which aim to accurately recognize activities based on raw accelerometer data, their magnitudes,

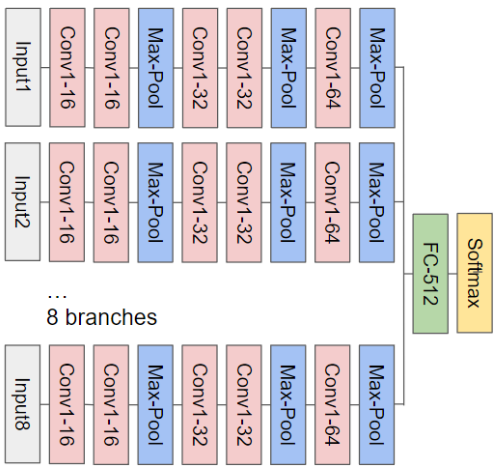
Table 6.1: PERFORMANCE OF CNN BENCHMARK MODELS. *P=Precision, R=Recall, F1=F1-score. The first 8 rows present the performance metrics for each activity individually, while the Macro Avg row represents the average values of activity-specific scores. The MCC row denotes the overall performance of the model.*

Activity	BM1			BM2		
	P	R	F1	P	R	F1
Lying	0.6257	0.5452	0.5827	0.5842	0.4448	0.5051
Sitting	0.1132	0.3007	0.1645	0.1081	0.4674	0.1755
Standing	0.0199	0.2384	0.0367	0.0265	0.2011	0.0468
Walk_slow	0.1628	0.1243	0.1410	0.3126	0.1479	0.2008
Walk_mod	0.2305	0.4646	0.3081	0.4267	0.3442	0.3810
Stairs	0.0002	0.2788	0.0004	0.0004	0.1923	0.0007
Jogging	0.0517	0.9147	0.0979	0.1488	0.8720	0.2542
Unknown	0.6200	0.1473	0.2380	0.6739	0.2354	0.3489
Macro Avg	0.2280	0.3768	0.1962	0.2851	0.3631	0.2391
MCC		0.1707			0.1725	

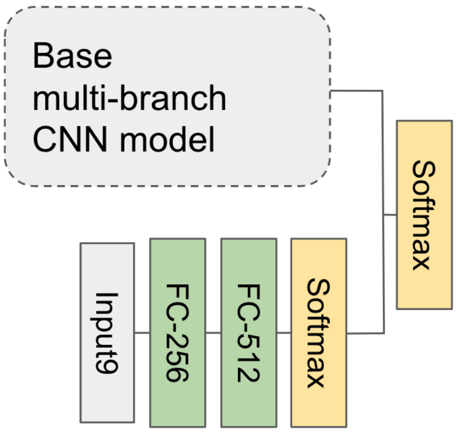
and filtered versions. Incorporating the magnitude of accelerometer data is essential as it provides a measure of the overall intensity of the movement. As shown in Figure 6.4(a), the base CNN model consists of eight parallel branches, each processing input from raw acceleration signals and magnitude. These branches incorporate specific Butterworth filters to extract different frequency components. Additionally, the trained base multi-branch CNN model is enhanced by incorporating an additional branch with fully connected layers that incorporates time domain and frequency domain features as illustrated in Figure 6.4(b). This inclusion aims to further explore improvement of the model's performance.

This approach offers several advantages in enhancing the model's performance. By incorporating parallel branches, the model is able to capture the distinctive characteristics of each filter and their individual contributions to the overall performance. This enables the model to learn and extract features from different frequency ranges independently, leading to a more robust and accurate activity recognition model. Furthermore, the integration of time domain and frequency domain features based on the predictions of the base multi-branch CNN model enhances the precision of the model's predictions. By combining these complementary features, the model gains a more comprehensive understanding of the underlying patterns and dynamics in the data, resulting in improved prediction accuracy.

In the following sections, we describe the details of our model architectures, the training process, and the experimental results, followed by an analysis of the results.



(a) Base multi-branch CNN Structure



(b) Combine CNN model structure

Figure 6.4: CNN STRUCTURE. *This figure including the structure of the base multi-branch CNN model used in this project (a) and how it combines features of each time window in combined CNN model (b)*

.

6.2.1 Model Structure

As illustrated in Section 4.3, we utilize a sliding window technique to extract overlapping time windows of 10 seconds with a 5-second overlap, resulting in each time window having a size of 3×200 for raw accelerometer data. The magnitude data and the filtered magnitude data have a shape of 1×200. Since we use the same convolutional structure for each branch, the magnitude and filtered magnitude data are padded to the same shape as the raw accelerometer data. This ensures that each branch of the model has the same input size of 3×200.

Base Multi-Branch CNN model

As depicted in Figure 6.4(a), our base multi-branch CNN model is designed with a specific architecture that consists of eight parallel branches ($2 + 2 \times 3$), all sharing the same structure. Each branch receives inputs from the raw acceleration signals, magnitude data, and data that has passed through three different Butterworth filters (high pass filter, band pass filter, low pass filter). The purpose of applying these filters is to extract various frequency components from both the raw data and the magnitude.

By incorporating multiple branches, our model can capture different aspects of the input data and leverage the extracted frequency components to enhance its understanding of the underlying patterns and structures. This parallel architecture allows for parallel processing and facilitates the model's ability to learn distinct representations from different branches.

The structure in each branch of our model includes five 1D convolutional layers, two max-pooling layers. The first two convolutional layers have 16 kernels each, while the second two have 32 kernels each, and the final layer has 64 kernels. Each convolutional layer uses a kernel of size 5 and a stride size of 1, and ReLU activation is applied to both convolutional and fully connected layers. After passing through the convolutional and max-pooling layers with size of 2, the data is flattened. After flattened, each branch is then concatenated together and fed into a fully connected layer with 512 units and dropout (p = 0.50). Finally, a classification layer with eight units, corresponding to the eight activities, and a softmax activation function are used for the final classification.

Combined CNN model

To further improve the performance of our activity recognition model, we explore the potential of combining the predictions of our trained base multi-branch CNN model with additional features extracted from each window, named combined CNN model. In particular, we extract a set of time-domain features and frequency-domain features (minimum, maximum, average, and standard deviation amplitude of windows per axis, ARA, binned distribution and histogram per axis, time between peaks, MFCCs, spectral centroid and spectral bandwidth). We incorporate these features by passing them through two fully connected layers with ReLU activation and a classification layer with softmax activation, generating feature representations. These representations are then concatenated with the output of the trained base multi-branch CNN model and fed into a final classification layer with softmax activation. This approach allows us to combine the predictions of the two models, leveraging their different strengths and weights determined during training. Figure 6.4b illustrates the architecture of the combined CNN model.

6.2.2 Training process

During training, 20% of the training data is set aside as a hold-out set for validation. We use a randomized search method for hyper-parameter tuning, conducting 10 trials to randomly sample combinations of hyper-parameters and selecting the best hyper-parameters for final training. The weights of the best performing model on the validation set are utilized for the final evaluation on the test set. To streamline the process of training and testing CNN models, we chose to use PyTorch Lightning, a user-friendly and robust deep learning library.

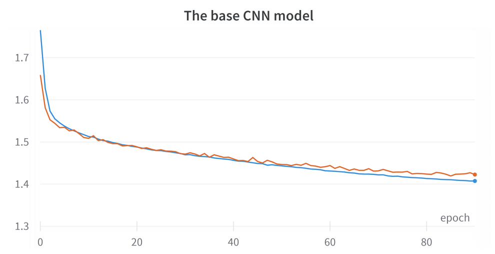
To address the issue of class imbalance in the training data, we experiment two methods: under-sampling and assigning class weights in loss function. Under-sampling involved reducing the number of samples from the majority class, creating a more balanced dataset. On the other hand, class weights were utilized in the loss function to assign higher weights to the minority class samples, penalizing the misclassification of minority class samples during training. The class weights are calculated as (6.1), where n_{samples} represents the total number of samples, n_{classes} represents the total number of classes, and n_{samples_i} represents the number of samples belonging to each class i.

$$\frac{n_{\text{samples}}}{n_{\text{classes}} \times n_{\text{samples}_i}} \tag{6.1}$$

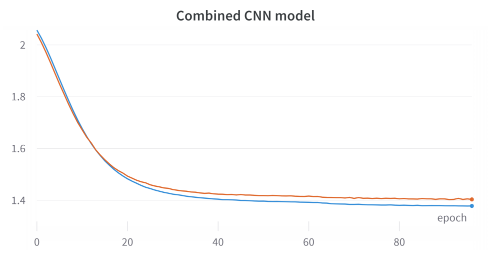
Both the base multi-branch CNN and combined CNN model are trained using the Adam optimizer and cross entropy loss function is used to evaluate the model performance. We train the models for up to 100 epochs, and early stopping of 5 epochs is applied meanwhile.

We first tune the hyper-parameters in the base multi-branch CNN model. Once we obtain the optimal hyper-parameters, we load the optimal base multi-branch CNN model weights and combine them with the extra branch of fully connected layers to create the combined CNN model. The weights of the base multi-branch CNN model are set untrainable when tuning the hyper-parameters for the combined CNN model.

For the base multi-branch CNN model, the hyper-parameters considered are the learning rate, kernel size, and number of neurons in the fully connected layer. The search space for the learning rate includes values of 1e-5, 1e-4, and 1e-3. The kernel size is selected from the options of 3, 4, and 5. The number of neurons in the fully connected layer is chosen from the values of 64, 128, 256, and 512. After evaluating ten combinations, we find that the model with a kernel size of 5, a



(a) Loss of the base multi-branch CNN model



(b) Loss of the combined CNN model

Figure 6.5: TRAINING AND VALIDATION LOSS OF THE CNN MODELS. *This figure shows the training and validation loss of the base multi-branch CNN model (a) and the combined CNN model (b) during the training process. The blue line indicates the training loss while the red line indicates the validation loss.*

learning rate of 0.0001, and 512 units in the fully connected layer achieves the lowest validation loss, indicating its superior performance among the considered combinations.

For the combined CNN model, the hyper-parameters considered are the learning rate, number of neurons in the first and second fully connected layer. The search space for the learning rate includes values of 1e-5, 1e-4, and 1e-3. The numbers of neurons in the fully connected layers are chosen from the values of 64, 128, 256, and 512. After evaluating ten combinations, we find that the model with a learning rate of 0.0001, 256 units in the first fully connected layer and 512 units in the second fully connected layer achieves the lowest validation loss.

Figure 6.5 demonstrates the convergence of the CNN models during the training process. In Figure 6.5(a), we observe the training and validation loss of the base multi-branch CNN model. The figure demonstrates that both the training and validation loss decrease as the model converges. This observation aligns with our implementation of early stopping with a tolerance of 5 epochs, which helps prevent overfitting.

Similarly, in Figure 6.5(b), we present the training and validation loss curves for the combined CNN model. The graph exhibits a similar pattern, with both the training and validation loss decreasing as the model converges. The loss of the combined model, built upon the trained base multi-branch CNN model, quickly reaches a low value and continues to gradually decrease.

6.3 Discussion

In this project, two methods are employed to address the issue of imbalanced training data in CNN models. The prediction results of the base CNN model and the combined CNN model using these methods are presented in Table 6.2 and Table 6.3, respectively.

The first approach involves applying random under-sampling to the training data, aiming to reduce the class imbalance and prevent bias towards the majority class. This technique randomly removes samples from the majority class, resulting in a more balanced distribution of samples across all classes. The second approach utilizes the class weights function in the cross-entropy loss function which assigns higher weights to minority classes, effectively giving them more importance during training. This method aims to explicitly address the class imbalance by adjusting the loss computation.

The results of the models trained with the weighted loss are presented in Table 6.2. The base multi-branch CNN model demonstrates better performance on activities such as "walk_mod," "jogging," and "unknown," achieving F1 scores of 0.5053, 0.5252, and 0.4750, respectively. How-

Table 6.2: PERFORMANCE OF CNN MODELS WITH CLASS WEIGHTS IN LOSS FUNCTION. *P=Precision, R=Recall, F1=F1-score. The first 8 rows present the performance metrics for each activity individually, while the Macro Avg row represents the average values of activity-specific scores. The MCC row denotes the overall performance of the model.*

Activity	Base			Combined		
	P	R	F1	P	R	F1
Lying	0.5507	0.1714	**0.2614**	0.5749	0.1652	0.2567
Sitting	0.0927	0.4814	0.1555	0.0930	0.5651	**0.1597**
Standing	0.0323	0.2623	0.0575	0.0413	0.2041	**0.0688**
Walk_slow	0.3618	0.0888	**0.1427**	0.3760	0.0439	0.0786
Walk_mod	0.4955	0.5156	0.5053	0.4919	0.5755	**0.5304**
Stairs	0.0007	0.2500	0.0015	0.0012	0.2500	**0.0023**
Jogging	0.3676	0.9200	0.5253	0.3788	0.8727	**0.5282**
Unknown	0.6506	0.3740	0.4750	0.7436	0.4036	**0.5232**
Macro Avg	0.3190	0.3829	0.2655	0.3376	0.3850	**0.2685**
MCC		**0.2037**			0.1767	

ever, it shows relatively lower performance on activities like "standing" and "stairs," with F1 scores of 0.0575 and 0.0015, respectively.

After incorporating additional features in the combined CNN model, we observe improvements in the performance of most activities, except for "lying" and "walk_slow." This results in a slight increase in the macro average F1 score from 0.2655 to 0.2685. However, it is noteworthy that the MCC score in the combined CNN model decreases from 0.2037 to 0.1767.

As shown in Table 6.3, the base multi-branch CNN model trained with sampled data shows better performance compared to that trained with weighted loss in most activities, especially "lying," with an F1 score improvement from 0.2614 to 0.6737. However, it performs relatively worse on "walk_slow," "stairs," and "jogging." The base multi-branch CNN model trained with balanced data achieves an overall macro average F1 score of 0.3186 and an MCC score of 0.3018.

However, after incorporating the additional features, only activities such as "sitting," "stairs," and "jogging" show slight improvements. Other activities experience poorer predictions, resulting in a worse overall performance.

Hence, the optimal CNN model for our project is the base multi-branch CNN model trained with balanced data. Our CNN model surpasses both benchmark models in Table 6.1 by achieving higher performance in all activities except "walk_mod," resulting in a significantly superior overall performance. Furthermore, the MCC score of our optimal model is 74.96% higher than that of the optimal benchmark model, which had a MCC performance of 0.1725.

In conclusion, our CNN analysis highlights several key findings. Firstly, balancing the training data in our CNN models proves to be a more effective approach compared to assigning class weights in the loss function. This technique results in improved performance across various activities and addresses the issue of class imbalance.

Secondly, the addition of an extra branch with input features does not yield significant improvements over our base multi-branch CNN model. While there may be marginal enhancements in certain activities, the overall impact on performance is limited.

Finally, our CNN model demonstrates significant improvements in the human activity recognition project compared to the optimal benchmark model. It outperforms the benchmarks in most activities and achieves higher overall performance. Notably, our CNN model achieves a 74.97% improvement in the MCC score, highlighting its superior performance.

These findings emphasize the importance of data balancing techniques and indicate that fea-

Table 6.3: PERFORMANCE OF CNN MODELS WITH RANDOM UNDER SAMPLER APPLIED IN TRAINING DATA. *P=Precision, R=Recall, F1=F1-score. The first 8 rows present the performance metrics for each activity individually, while the Macro Avg row represents the average values of activity-specific scores. The MCC row denotes the overall performance of the model.*

Activity	Base			Combined		
	P	R	F1	P	R	F1
Lying	0.7368	0.6206	**0.6737**	0.7389	0.5995	0.6619
Sitting	0.1174	0.4599	0.1870	0.1187	0.4526	**0.1880**
Standing	0.0398	0.2574	**0.0690**	0.0353	0.3159	0.0635
Walk_slow	0.3960	0.0838	**0.1383**	0.3937	0.0780	0.1302
Walk_mod	0.5684	0.5019	**0.5331**	0.4382	0.5986	0.5060
Stairs	0.0006	0.2212	0.0011	0.0006	0.1923	**0.0012**
Jogging	0.3138	0.9180	0.4677	0.3407	0.9180	**0.4969**
Unknown	0.7794	0.3458	**0.4791**	0.7359	0.2949	0.4210
Macro Avg	0.3690	0.4261	**0.3186**	0.3503	0.4312	0.3086
MCC		**0.3018**			0.2767	

ture augmentation may not always lead to substantial performance gains. The optimal strategy for our project involves balancing the training data and utilizing the base multi-branch CNN model.

Chapter 7

Comparison and Discussion

7.1 Datasets and Transformations

In this section, we aim to assess the informativeness of the training datasets and the effectiveness of transformation techniques by examining their impact on the performance of our hierarchical machine learning model as well as CNN models. To this end, we first conduct experiments covering all transformation techniques and all possible combinations of training datasets on the hierarchical ML models, then proceed to conduct similar experiments by excluding training datasets or transformations individually from the inputs in ML or CNN training for a clearer comparison between the two models.

The results of excluding training datasets are shown in Table 7.1. Observing the results from ML models in Table 7.1a, we notice that including the Selfback dataset for training increases classification performance, whereas including the WISDM dataset for training leads to a considerable decrease in performance except for "walk_mod" activity. The positive contribution from the Selfback dataset may be attributed to the fact that the data in this dataset were recorded using the same product, Axivity AX3, as used in Capture24, the test dataset. In contrast, the device used in the Wisdm dataset is a smartwatch, LG G Watch, which may have different specifications from the Axivity AX3, resulting in relatively low performance even with transformations applied.

In a bid to investigate whether excluding the WISDM dataset would similarly impact the performance of the flat ML model, we executed an additional experiment, the results of which are presented in Table 7.1b. Evidently, the F1 scores across most activities exhibit an improvement, echoing our previous findings. It is, however, noteworthy that this enhancement is not as substantial as that witnessed in the hierarchical ML model. Additionally, with the WISDM dataset excluded, our hierarchical ML model outperforms the flat ML model in terms of the classification performance of the "jogging" activity and the macro-F1 score, despite showing a slight underperformance in classifying the "lying" activity and a lower MCC score.

Interestingly, Table 7.1c shows that including the WISDM dataset in our experiments does not significantly affect the CNN model's performance. Although removing WISDM data leads to minor improvements in activities such as "lying," "jogging," and "walk_slow," it also causes a decrease in terms of F1 scores on other activities, resulting in a slightly lower overall performance. Specifically, the MCC value decreases from 0.3018 to 0.2900. Therefore, it is not necessary to exclude the WISDM dataset from the training data for the CNN model. This finding might indicate that CNN models could generalize our training datasets better than ML models, enabling them to better adapt to variations in the data.

The results in Table 7.2 demonstrate that most transformations can improve the performance of both ML and CNN models across the majority of activities. It is important to note that, due to the computationally intensive training process of the CNN model, we limit the experiment with the CNN model to either applying all transformations or none. Notably, for specific activities like "lying," swapping the x and y axes results in worse performance, possibly because the alteration in movement direction makes the features for "lying" less distinguishable. For the "walk_slow" activity, it is somewhat perplexing to note that the best performance for both ML and CNN models is achieved without applying any transformations. This observation warrants further investigation. Moreover, for activities such as "sitting," "standing," and "walk_mod," the application of amplitude perturbation might cause the intensity of the activities to become too similar to that of other activities within the same group. This could lead to less distinct features and subsequently lower performance.

In summary, ML models tend to perform better when the training input shares similar specifications with the target evaluation data. In contrast, incorporating more diverse datasets in CNN training yields similar or even better results, potentially indicating the higher generalization ability of the CNN model. Transformation techniques can generally benefit the performance of both ML and CNN models. However, it is essential to emphasize that the choice of transformation techniques should be guided by the specific activities being studied, as their effectiveness may vary depending on the context.

7.2 ML and CNN Comparison

In this section, we present a detailed comparison between the performance of ML models and CNN models on the activity recognition task, as shown in Table 7.3. We compare the flat ML, hierarchical ML, CNN benchmark, and our proposed CNN model, taking into account the F1 scores, macro-F1 score, and MCC.

From the results in Table 7.3, it is evident that the best-performing benchmark CNN model from the literature, shows a lower overall performance compared to both the flat ML and hierarchical ML models, with lower macro-F1 and MCC scores. From this project, our proposed CNN model demonstrates a significant improvement over the CNN benchmark model, achieving higher macro-F1 and MCC scores. This suggests that the modifications we made to the CNN model have contributed to the enhancement of its performance.

When comparing the hierarchical ML model and our proposed CNN model on a per-activity basis, it can be observed that the hierarchical ML model achieves higher F1 scores for some of the activities, such as "sitting," "standing," and "jogging." However, our proposed CNN model surpasses the hierarchical ML model in the walking activities such as "walk_slow" and "walk_mod". It is also worth noting that the CNN model has similar macro-F1 score as our hierarchical ML model, while the hierarchical ML model has higher MCC score.

Despite the traditional ML models exhibiting better performance across most activities, this could only exist with Wisdm dataset excluded. The CNN model demonstrates superior generalizability, which can be particularly useful in real-world scenarios where diverse data sources and sensor positions are involved. Furthermore, the potential of CNN models for enhanced performance increases as more diverse datasets are incorporated, as observed in the earlier comparison between ML and CNN models with different training datasets.

Based on the results we currently have and our knowledge, we would suggest that the selection of ML or CNN models for activity recognition should be guided by the specific requirements of the application, such as the level of generalization needed and the diversity of data sources involved.

Table 7.1: Experiment results with removed training datasets. *This table displays the performance of the models when one of the training datasets is removed.*

Metric		Excluded dataset					
		N/A	Selfback	Wisdm	Gotov	Adl_hmp	Act_cp
F1	Lying	0.6235	0.2938	**0.7259**	0.2479	0.4509	0.4477
	Sitting	0.1831	0.1689	**0.1968**	0.1230	0.1659	0.1690
	Standing	**0.0700**	0.0531	0.0695	0.0451	0.0592	0.0598
	Walk_slow	0.0587	0.0457	**0.0935**	0.0300	0.0605	0.0714
	Walk_mod	0.4695	0.5070	0.4466	0.4624	**0.5091**	0.4901
	Stairs	0.0010	0.0017	0.0012	0.0010	**0.0017**	0.0015
	Jogging	0.4764	0.5098	**0.5312**	0.4734	0.4963	0.4319
	Unknown	0.4208	**0.5973**	0.4739	0.6496	0.5869	0.5435
	Macro Avg	0.2879	0.2721	**0.3173**	0.2541	0.2913	0.2768
MCC		0.2639	0.1898	**0.3245**	0.1950	0.2354	0.2152

(a) Experiment results for hierarchical ML model

Metric		Excluded dataset	
		N/A	Wisdm
F1	Lying	0.7410	**0.7423**
	Sitting	**0.1913**	0.1913
	Standing	0.0676	**0.0680**
	Walk_slow	**0.0432**	0.0431
	Walk_mod	0.4417	**0.4417**
	Stairs	**0.0012**	0.0011
	Jogging	0.4320	**0.4449**
	Unknown	0.4865	**0.4879**
	Macro Avg	0.3006	**0.3025**
MCC		0.3306	**0.3317**

(b) Experiment results for flat ML model

Metric		Excluded dataset	
		N/A	Wisdm
F1	Lying	0.6737	**0.7066**
	Sitting	**0.1870**	0.1833
	Standing	**0.0690**	0.0638
	Walk_slow	0.1383	**0.1508**
	Walk_mod	**0.5331**	0.5147
	Stairs	**0.0011**	0.0010
	Jogging	0.4677	**0.6597**
	Unknown	**0.4791**	0.3728
	Macro Avg	0.3186	**0.3316**
MCC		**0.3018**	0.2900

(c) Experiment results for CNN model

Table 7.2: EXPERIMENT RESULTS WITH REMOVED TRANSFORMATIONS. *This table displays the performance of the models when one of the transformation techniques is removed. Wisdm dataset is excluded from ML training*

Metric		Excluded transformation N/A	Swapping x and y	Reverting x, y, and z	Amplitude Perturbation	Rotation Perturbation	All
F1	Lying	0.7259	**0.7354**	0.7161	0.7171	0.7244	0.7100
	Sitting	0.1968	0.1966	0.1843	**0.1973**	0.1972	0.1839
	Standing	0.0695	0.0675	0.0652	**0.0713**	0.0704	0.0474
	Walk_slow	0.0935	0.1011	0.0460	0.1069	0.0998	**0.1631**
	Walk_mod	0.4466	0.4492	0.3796	**0.4515**	0.4487	0.4139
	Stairs	0.0012	0.0011	0.0005	0.0012	**0.0012**	0.0005
	Jogging	0.5312	0.2979	0.4061	0.5290	**0.5327**	0.1824
	Unknown	0.4739	0.4730	0.4677	0.4716	0.4753	**0.5420**
	Macro Avg	0.3173	0.2902	0.2832	0.3182	**0.3187**	0.2721
MCC		0.3245	**0.3283**	0.3074	0.3213	0.3251	0.3080

(a) Experiment results for hierarchical ML model

Metric		Excluded transformation N/A	All
F1	Lying	**0.6737**	0.5008
	Sitting	**0.1870**	0.1668
	Standing	**0.0690**	0.0453
	Walk_slow	0.1383	**0.1731**
	Walk_mod	**0.5331**	0.5272
	Stairs	0.0011	**0.0018**
	Jogging	0.4677	**0.4822**
	Unknown	0.4791	**0.5124**
	Macro Avg	**0.3186**	0.3012
MCC		**0.3018**	0.2216

(b) Experiment results for CNN model

7.3 Discussion and Future Directions

In this report, we have investigated the performance of ML and CNN models for human activity recognition and explored various training datasets and transformation techniques to enhance their performance. Besides our work, several avenues for future research and improvement can be considered.

First, extracting some less-fluctuating features, such as Gabor features or other related frequency-bound features, could be applied to better distinguish between activities with similar features, such as "lying," "sitting," and "standing," and to improve the poor performance of "stairs" classification.

Second, incorporating more diverse datasets, potentially from a wider range of devices, can increase the model's generalization ability and robustness. Although our training datasets were diverse and representative, they were collected in controlled environments, which may not fully capture the complexity and variability of real-world situations. By including more diverse and extensive datasets, the model, especially the CNN model, could better recognize activities across

Table 7.3: FINAL RESULTS FOR THE ML AND CNN MODELS. *This table compares the performance of the flat ML model, hierarchical ML model, CNN benchmark model, and our proposed CNN model. WISDM dataset is excluded from training in ML models. The CNN benchmark model represents the best-performing CNN (benchmark 2) from the literature.*

Metric		Flat ML	Hierarchical ML	CNN Benckmark	CNN
F1	Lying	**0.7423**	0.7259	0.5051	0.6737
	Sitting	0.1913	**0.1968**	0.1755	0.1870
	Standing	0.0680	**0.0695**	0.0468	0.0690
	Walk_slow	0.0431	0.0935	**0.2008**	0.1383
	Walk_mod	0.4417	0.4466	0.3810	**0.5331**
	Stairs	0.0011	**0.0012**	0.0007	0.0011
	Jogging	0.4449	**0.5312**	0.2542	0.4677
	Unknown	**0.4879**	0.4739	0.3489	0.4791
	Macro Avg	0.3025	0.3173	0.2391	**0.3186**
MCC		**0.3306**	0.3245	0.1725	0.3018

different devices and contexts, potentially improving performance for activities like "stairs."

Third, the data augmentation techniques utilized in this study, such as swapping the x and y axes, could potentially result in ambiguity between analogous activities like "sitting" and "standing." Furthermore, our application of rotation perturbation was limited to a singular direction, when an exploration of both directions may yield performance improvements. Future research should consider exploring alternative data augmentation methodologies that more effectively maintain the integrity of activity labels. Additionally, gathering more diverse and comprehensive datasets encompassing a wider variety of sensor modalities could enhance the model's generalizability and performance within natural environments.

For the hierarchical ML model, further exploration of other machine learning models or ensemble techniques could improve classification performance. For instance, employing techniques like AutoEncoder (Hinton and Salakhutdinov, 2006) to automatically learn a hierarchical representation of the data could lead to more meaningful groupings of activities and address the issue of poor performance for certain activities.

For the CNN model, while our multi-branch CNN architecture was designed to be adaptable to different sensor modalities and sub-tasks, we only use a single fixed structure for our specific human activity recognition task. Future work could investigate the benefits of adjusting the architecture and hyperparameters of each branch for different sub-tasks or sensor modalities and explore how to effectively fine-tune the model for optimal performance.

Additionally, although we attempt to address the issue of class imbalance using focal loss, unfortunately, the results were meaningless. This highlights the need for further exploration into alternative methods for handling class imbalance. Meanwhile, in an effort to mitigate the effects of network agnostophobia and enhance the model's performance, the adoption of Entropic Open-Set and Objectosphere losses proposed by Dhamija et al. (2018) might prove effective. These techniques could potentially offer more robust handling of unknown activities and improve the model's discrimination capacity.

Finally, the activity translation in our study is largely informed by our knowledge and intuition. However, it is conceivable that there may be room for refinement or alternative perspectives in this area. For instance, the subtleties of certain activities or the interrelationships between them may be better captured with input from domain experts, such as physiologists or exercise scientists. Collaborating with such experts could yield more nuanced and precise definitions of activity classes, leading to more accurate translations and potentially improved classification performance.

In summary, by addressing these potential areas of improvement and exploring new avenues for research, we can further enhance the performance of hierarchical ML and CNN models for human activity recognition and better understand the nuances of these techniques in real-world applications.

Chapter 8

Conclusion

In this report, we have investigated and compared the impact of datasets and data transformation techniques, and the performance of ML and CNN models for HAR. Through the analysis of various training datasets, transformation techniques, and model structures, we have identified several key insights that can guide future research in HAR.

First, we found that employing a hierarchical training structure enhances the classification performance in HAR tasks. This approach allows the model to focus on differentiating between broader categories of activities first and then classify the activities within each category.

Second, utilizing a multiple branches CNN model to capture different frequency components of data and features proves advantageous. Incorporating an additional branch with fully connected layers for other features did not significantly improve the model's performance. This suggests that the existing branches already capture sufficient information from the data, rendering the additional features redundant.

Third, while traditional ML models, such as SVM and RF, demonstrate higher performance in our experiments, CNNs offer greater generalizability. This finding suggests that CNNs may have the potential to perform better on new datasets or under more diverse conditions, given their ability to generalize across different contexts.

Fourth, our results indicate that CNNs hold potential for improved performance as more diverse datasets are incorporated. As the model's training data becomes more representative of real-world situations, the CNN's ability to recognize activities across different devices and contexts is expected to increase.

Fifth, the choice of transformation techniques should be guided by the specific activities under study. While some transformations improve the overall performance of the models, certain techniques may negatively affect the classification of specific activities. Therefore, it is essential to carefully consider the impact of each transformation technique on the activities of interest.

Lastly, the choice of datasets for training should, if possible, be as close as possible to the target dataset. Our results show that ML models perform better when the training data has similar specifications to the target evaluation data. However, incorporating more diverse datasets in CNN training results in similar or even better performance, further highlighting the generalization ability of CNN models.

In conclusion, this report has provided valuable insights into the performance of ML and CNN models for HAR and the impact of various training datasets and transformation techniques on their performance. By building on these findings and addressing the potential areas of improvement, future research in HAR can continue to advance our understanding of these techniques and improve their applicability in real-world settings.

Appendix A

Attachments

A.1 Work distribution

In this project, we purposefully allocated tasks to effectively leverage the unique strengths and areas of expertise of each group member, aiming to maintain a roughly equal workload distribution. We postulate that such an approach not only cultivates a productive and cooperative team environment, but also potentially enhances the quality of our collective output. For greater transparency, Table A.1 delineates the specifics of task allocation, listing the names of members assigned to each distinct task.

Table A.1: DISTRIBUTION OF PROJECT TASKS AMONG GROUP MEMBERS.

Tasks		Study	Implementation	Writing report
Abstract and introduction		N/A	N/A	He Liu
Dataset and data loader		Siqi Bao, He Liu	He Liu	Siqi Bao
Data	Preprocessing	He Liu	He Liu	He Liu
	Transformation	He Liu	He Liu	He Liu
Feature	Selection	Siqi Bao	Siqi Bao & He Liu	Siqi Bao
	Extraction pipeline	He Liu	He Liu	He Liu
Hierarchical ML		He Liu	He Liu	He Liu
CNN	Models	Siqi Bao	Siqi Bao	Siqi Bao
	Training pipeline	He Liu	He Liu	Siqi Bao
Results, discussion, and conclusion		He Liu, Siqi Bao	He Liu, Siqi Bao	He Liu, Siqi Bao
Others	Code documentation	N/A	N/A	He Liu

A.2 Data pipeline

Figure A.1 showcases our data processing pipeline for model training, highlighting the specific libraries and functions used at each step.

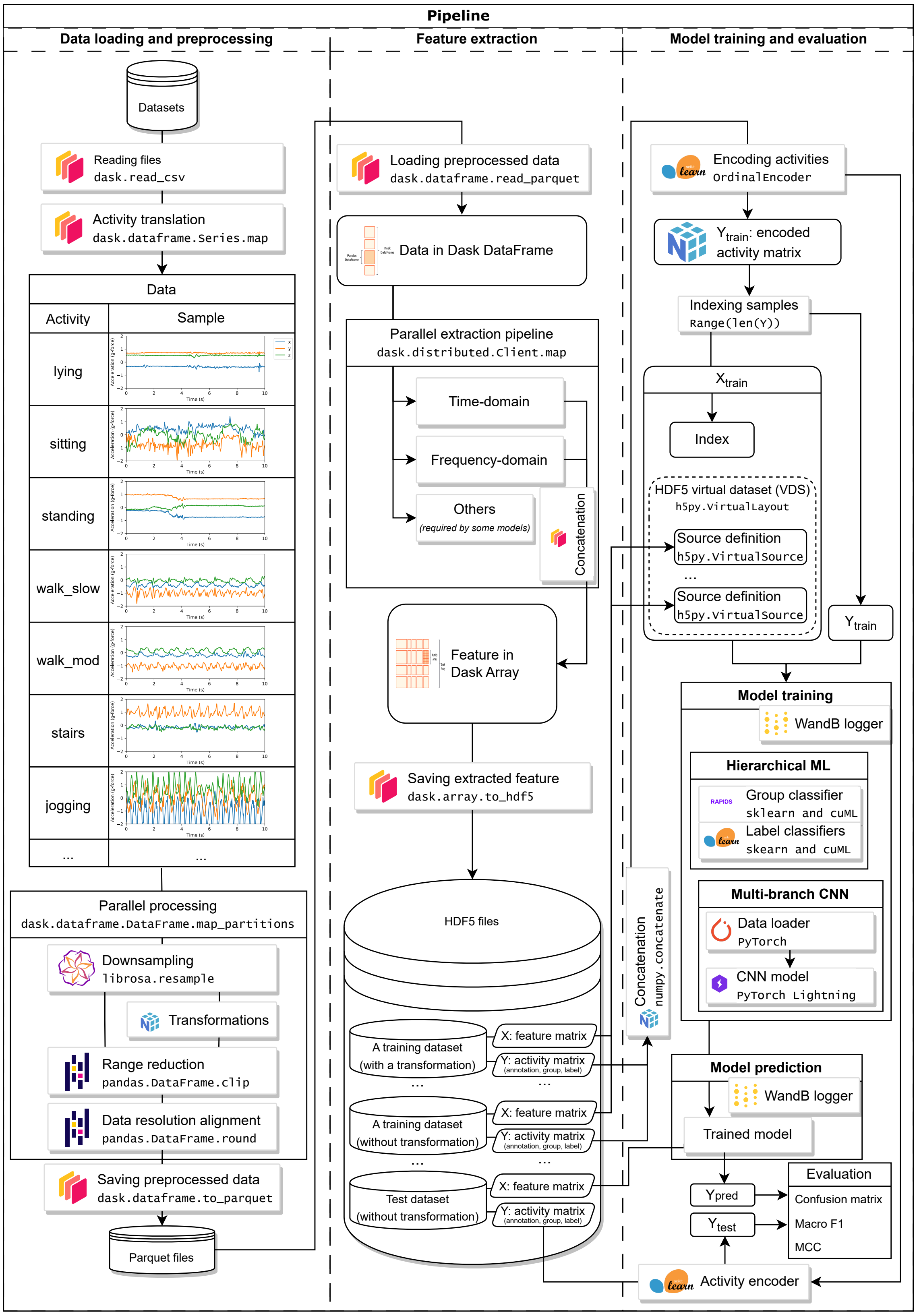

Pipeline
Data loading and preprocessing
Feature extraction
Model training and evaluation
Datasets
Reading files dask.read_csv
Activity translation dask.dataframe.Series.map
Data
Activity
Sample
lying
sitting
standing
walk_slow
walk_mod
stairs
jogging
Parallel processing dask.dataframe.DataFrame.map_partitions
Downsampling librosa.resample
Transformations
Range reduction pandas.DataFrame.clip
Data resolution alignment pandas.DataFrame.round
Saving preprocessed data dask.dataframe.to_parquet
Parquet files
Loading preprocessed data dask.dataframe.read_parquet
Data in Dask DataFrame
Parallel extraction pipeline dask.distributed.Client.map
Time-domain
Frequency-domain
Others (required by some models)
Concatenation
Feature in Dask Array
Saving extracted feature dask.array.to_hdf5
HDF5 files
A training dataset (with a transformation)
A training dataset (without transformation)
Test dataset (without transformation)
X: feature matrix
Y: activity matrix (annotation, group, label)
Encoding activities OrdinalEncoder
Y_train: encoded activity matrix
Indexing samples Range(len(Y))
X_train
Index
HDF5 virtual dataset (VDS) h5py.VirtualLayout
Source definition h5py.VirtualSource
Y_train
Concatenation numpy.concatenate
Model training
WandB logger
Hierarchical ML
Group classifier sklearn and cuML
Label classifiers skearn and cuML
Multi-branch CNN
Data loader PyTorch
CNN model PyTorch Lightning
Model prediction
Trained model
Ypred
Ytest
Evaluation
Confusion matrix
Macro F1
MCC
Activity encoder

Figure A.1: DATA PIPELINE WITH DETAILS IN EACH STAGE.

A.3 Preparation of the Computational Environment on the ETH Euler Cluster

The present section aims to illustrate the steps involved in the preparation of an environment suitable for computational task execution on the ETH Euler cluster. Essential environmental variables employed in the context of this project are enumerated in Table A.2. Beyond the establishment of these variables, the preloading of certain Lmod modules, detailed in Table A.3, is a requisite[1].

In instances where there is a need for establishing a connection between a compute node and an external service beyond the ETH network (for instance, tracking model training on WandB), it becomes necessary to configure a proxy [2].

Upon appropriate establishment of environment variables and preloading of required Lmod modules, the next progression is the preparation of the Python environment. In the ETH Euler environment, it is advocated to utilize Python's native virtual environment (venv) rather than conda[3]. The Python package dependencies crucial for the project are listed in Table A.4.

Table A.2: Environmental Variables Used in the Project Execution.

Variable	Description
ON_CLUSTER	Set to 1 for running on ETH Euler; otherwise set for local development.
OUTPUT_DIR	Specifies the directory where the prediction results and trained models are saved. Defaults to the "_output" directory in the repository root.
DATA_DIR	Indicates the directory where the datasets are located. Defaults to the "data" directory in the repository root.
CACHE_DIR	Defines the directory where preprocessed data and extracted features are saved and loaded. Defaults to the "cache" directory in the repository root.
VENV_DIR	Points to the location of the Python virtual environment. Not required for local development.
OMP_NUM_THREADS	Determines the number of threads requested. Must be set before job submission.
WANDB_API_KEY	Represents the authentication key for logging metrics on WandB. Leave unset to disable WandB logging.

[1]Consult the Euler wiki for additional details about Lmod modules: https://scicomp.ethz.ch/wiki/Euler_applications_and_libraries

[2]The following guide provides the necessary steps for proxy setup: https://scicomp.ethz.ch/wiki/Accessing_the_clusters#Security

[3]For more details about not using conda on Euler, refer to this page: https://scicomp.ethz.ch/wiki/Conda

Table A.3: LMOD MODULES REQUIRED IN THE PROJECT EXECUTION. *For proper functioning,* `gcc/8.2.0` *should always be the first module to be loaded.*

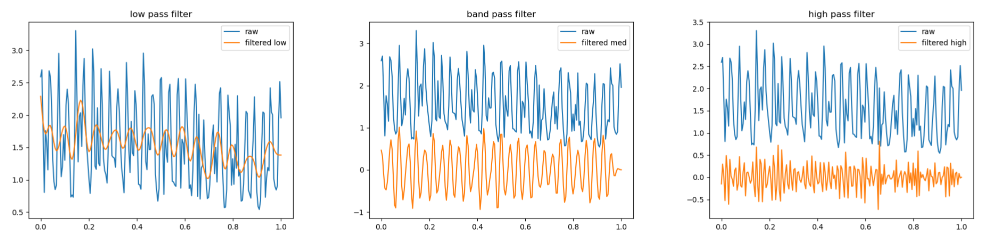
Module	Version
gcc	8.2.0
python_gpu	3.8.5
cuda	11.8.0
cudnn	8.8.1.3
hdf5	1.10.9
openmpi	4.1.4
libsndfile	1.0.28
curl	7.73.0

Table A.4: PYTHON PACKAGE DEPENDENCIES.

Packages	Version	Packages	Version	Packages	Version
pyyaml	6.0	seaborn	0.12.2	pytorch-lightning	2.0.1
coloredlogs	15.0.1	librosa	0.10.0	scikit-learn	1.2.2
kaggle	1.5.13	dask	2023.1.1	imbalanced-learn	0.10.1
uuid6	2022.10.15	dask-ml	2023.3.24	fastparquet	2023.2.0
shortuuid	1.0.11	numpy	1.23.5	pyarrow	10.0.1
wandb	0.15.2	pandas	1.5.3	h5py	3.8.0
torch	2.0.0	scipy	1.10.1	bokeh	2.4.3
kornia	0.6.11	bottleneck	1.3.7	python-snappy	0.6.1

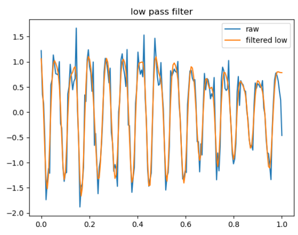
A.4 Butterworth filters

This section illustrates "filtered accelerometer date of windows" and "filtered magnitude of windows" features for a "jogging" activity using Butterworth filters, serving as an appendix to Section 4.3. Figure A.3 and Figure A.2 depict the raw data components along each axis, as well as the magnitude.

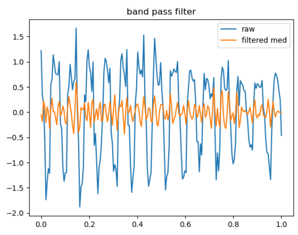


(a) Low frequency component of magnitude (b) Middle frequency component of magnitude (c) High frequency component of magnitude

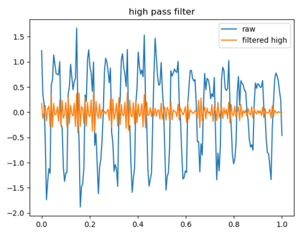
Figure A.2: FILTERED MAGNITUDE WITH AN EXAMPLE OF JOGGING. *This figure includes the three frequency parts of magnitude of accelerometer data in each axis filtered by Bandwidth filters. The cut points for low pass filter is 2Hz (a), (2Hz, 5Hz) for band-pass filter (b), 5Hz for high pass filter (c).*



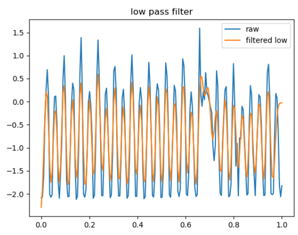
(a) Low frequency component of raw data in axis 0



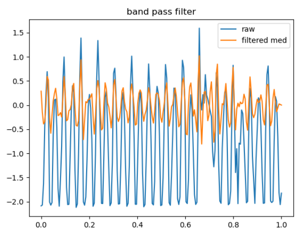
(b) Middle frequency component of raw data in axis 0



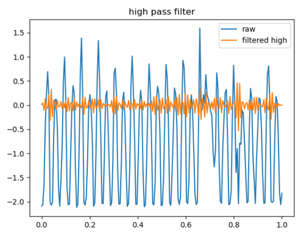
(c) High frequency component of raw data in axis 0



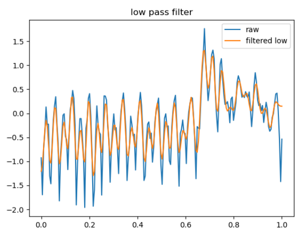
(d) Low frequency component of raw data in axis 1



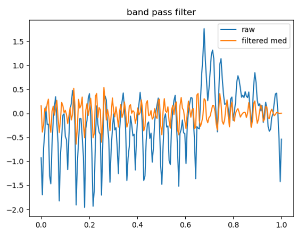
(e) Middle frequency component of raw data in axis 1



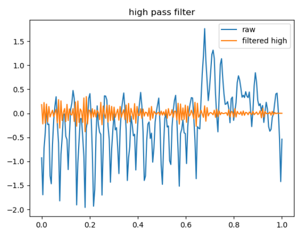
(f) High frequency component of raw data in axis 1



(g) Low frequency component of raw data in axis 2



(h) Middle frequency component of raw data in axis 2



(i) High frequency component of raw data in axis 2

Figure A.3: FILTERED DATA WITH AN EXAMPLE OF JOGGING. *This figure includes the three frequency parts of accelerometer data in each axis filtered by Bandwidth filters. The cut points for low pass filter is 3Hz (a), (3Hz, 7Hz) for band-pass filter (b), 7Hz for high pass filter (c).*

A.5 Label Translation

We standardized the activity labels across different datasets by assigning them to uniform labels. The label translations for the five training datasets are presented in two separate tables, namely Table A.5 and Table A.6, due to page limitations. The label transformation details for the test dataset, Capture 24, are provided in Table A.7 and Table A.8.

A.6 Hierarchical ML parameters

This section outlines the optimal hyperparameters for the flat and hierarchical ML models utilized within this project (as shown in Table A.9). The candidate hyperparameters considered for tuning are detailed in Table 5.1, with the optimal values listed in this section being the results derived from the `best_params_` attribute of the `GridSearchCV` function from the `scikit-learn` library. The output represents the hyperparameter combination that yielded the highest MCC score during cross-validation.

Table A.5: LABEL TRANSLATION FOR TRAINING AND TEST DATASETS. *The tables present the translation from the original annotations to the new labels and groups adopted in our project. Annotation represents the activities performed in the data, while annotation_id stands for the corresponding unique ID of the annotation. Label denotes the raw labels used in the datasets. Label-group and label-use indicate the groups and labels utilized in our project. "-" indicates the activities excluded from training.*

annotation	annotation_id	label	label-group	label-use
brush_teeth	adl_hmp-1	brush_teeth	unknown	unknown
climb_stairs	adl_hmp-2	upstairs	walking	stairs
descend_stairs	adl_hmp-3	downstairs	walking	stairs
drink_glass	adl_hmp-4	drinking	unknown	unknown
eat_meat	adl_hmp-5	eating	unknown	unknown
eat_soup	adl_hmp-6	eating	unknown	unknown

(a) Label translation in Adl_hmp

annotation	annotation_id	label	label-group	label-use
NA	gotov-1	unknown	unknown	unknown
standing	gotov-2	standing	sitstand	standing
lyingDownLeft	gotov-3	lying	lying	lying
lyingDownRight	gotov-4	lying	lying	lying
sittingSofa	gotov-5	sitting	sitstand	sitting
sittingCouch	gotov-6	sitting	sitstand	sitting
sittingChair	gotov-7	sitting	sitstand	sitting
walkingStairsUp	gotov-8	upstairs	walking	stairs
washingDishes	gotov-9	housework	unknown	unknown
vacuumCleaning	gotov-10	housework	unknown	unknown
walkingSlow	gotov-11	walk_slow	walking	walk_slow
walkingNormal	gotov-12	walk_mod	walking	walk_mod
walkingFast	gotov-13	walk_fast	walking	walk_mod
cycling	gotov-14	bicycling	unknown	unknown

(b) Label translation in Gotov

annotation	annotation_id	label	label-group	label-use
other	act_cp-1	other	-	-
relax	act_cp-2	relaxing	-	-
keyboard_writing	act_cp-3	keyboard_writing	unknown	unknown
laptop	act_cp-4	laptop	unknown	unknown
handwriting	act_cp-5	handwriting	unknown	unknown
handwashing	act_cp-6	handwashing	unknown	unknown
facewashing	act_cp-7	facewashing	unknown	unknown
teethbrush	act_cp-8	teethbrush	unknown	unknown
sweeping	act_cp-9	sweeping	unknown	unknown
vacuuming	act_cp-10	vacuuming	unknown	unknown
eating	act_cp-11	eating	unknown	unknown
dusting	act_cp-12	dusting	unknown	unknown
rubbing	act_cp-13	rubbing	unknown	unknown
downstairs	act_cp-14	downstairs	walking	stairs
walking	act_cp-15	walking	walking	walk_mod
walking_fast	act_cp-16	walking_fast	walking	walk_mod
upstairs	act_cp-17	upstairs	walking	stairs
upstairs_fast	act_cp-18	upstairs_fast	walking	stairs

(c) Label translation in Act_cp

Table A.6: LABEL TRANSLATION FOR TRAINING AND TEST DATASETS.. *The second part of the translation tables.*

annotation	annotation_id	label	label-group	label-use
upstairs	selfback-1	walking	stairs	stairs
downstairs	selfback-2	walking	stairs	stairs
walk_slow	selfback-3	walking	walk_slow	walk_slow
walk_mod	selfback-4	walking	walk_mod	walk_mod
walk_fast	selfback-5	walking	walk_mod	walk_mod
sitting	selfback-6	sitstand	sitting	sitting
lying	selfback-7	lying	lying	lying
standing	selfback-8	sitstand	standing	standing
jogging	selfback-9	sports	jogging	jogging

(a) Label translation in Selfback

annotation	annotation_id	label	label-group	label-use
A	wisdm-1	walking	walking	walk_mod
B	wisdm-2	jogging	sports	jogging
C	wisdm-3	stairs	walking	stairs
D	wisdm-4	sitting	sitstand	sitting
E	wisdm-5	standing	sitstand	standing
F	wisdm-6	typing	unknown	unknown
G	wisdm-7	brushing teeth	unknown	unknown
H	wisdm-8	eating soup	unknown	unknown
I	wisdm-9	eating chips	unknown	unknown
J	wisdm-10	eating pasta	unknown	unknown
K	wisdm-11	drinking from cup	unknown	unknown
L	wisdm-12	eating sandwich	unknown	unknown
M	wisdm-13	kicking	unknown	unknown
O	wisdm-14	playing catch	-	-
P	wisdm-15	dribbling	-	-
Q	wisdm-16	writing	unknown	unknown
R	wisdm-17	clapping	-	-
S	wisdm-18	folding clothes	unknown	unknown

(b) Label translation in Wisdm

Table A.7: Label translation for test dataset Capture 24-1. *The first part of the label translation of Capture 24.*

annotation	annotation_id	label-group	label-use
transportation;private transportation;1010 bicycling;MET 4.0	capture24-1	unknown	unknown
home activity;household chores;house cleaning;floors;5010 cleaning sweeping carpet or floors;MET 3.3	capture24-2	unknown	unknown
home activity;household chores;house cleaning;furniture;5020 cleaning heavy such as car/windows/garage;MET 3.5	capture24-3	unknown	unknown
home activity;miscellaneous;5025 (generic) multiple household tasks all at once including standing/lifting/sitting;MET 2.8	capture24-4	unknown	unknown
home activity;household chores;house cleaning;furniture;5032 dusting or polishing furniture;MET 2.3	capture24-5	unknown	unknown
home activity;household chores;preparing meals/cooking/washing dishes;5035 kitchen activity general cooking/washing/dishes/cleaning up;MET 3.3	capture24-6	unknown	unknown
home activity;household chores;preparing meals/cooking/washing dishes;5035 cleaning up table after meal implied walking (e.g. leaving from eating table to the kitchen);MET 3.3	capture24-7	unknown	unknown
occupation;interruption;5041 kitchen activity in the working place;MET 1.8	capture24-8	unknown	unknown
occupation;interruption;miscellaneous;5041 kitchen activity in the working place;MET 1.8	capture24-9	unknown	unknown
home activity;household chores;preparing meals/cooking/washing dishes;5051 serving food/setting table implied walking and standing;MET 2.5	capture24-10	unknown	unknown
leisure;miscellaneous;walking;5060 shopping miscellaneous;MET 2.3	capture24-11	unknown	unknown
leisure;miscellaneous;5060 shopping miscellaneous;MET 2.3	capture24-12	unknown	unknown
home activity;household chores;grocery shopping;5060 shopping;MET 2.3	capture24-13	unknown	unknown
leisure;eating;not-social;5060 buying foods or drinks as a takeaway;MET 2.3	capture24-14	unknown	unknown
leisure;eating;social;5060 buying foods or drinks as a takeaway;MET 2.3	capture24-15	unknown	unknown
leisure;eating;5060 buying foods or drinks as a takeaway;MET 2.3	capture24-16	unknown	unknown
home activity;household chores;washing/ironing/mending clothes;5070 ironing;MET 1.8	capture24-17	unknown	unknown
home activity;miscellaneous;sitting;5080 sitting non-desk work (with or without eating at the same time);MET 1.3	capture24-18	unknown	unknown
leisure;miscellaneous;sitting;5080 sitting non-desk work (with or without eating at the same time);MET 1.3	capture24-19	unknown	unknown
home activity;household chores;washing/ironing/mending clothes;5090 folding or hanging clothes/put clothes in or out of washer or dryer/packing suitcase limited walking;MET 2.0	capture24-20	unknown	unknown
home activity;household chores;washing/ironing/mending clothes;5095 putting away /gathering clothes involving walking;MET 2.3	capture24-21	unknown	unknown
home activity;household chores;house cleaning;miscellaneous;5100 making bed/changing linens;MET 3.3	capture24-22	unknown	unknown
home activity;miscellaneous;walking;5121 walking with moving and lifting loads such as bikes and furniture;MET 4.0	capture24-23	unknown	unknown
home activity;household chores;house cleaning;floors;5131 scrubbing floors on hands and knees scrubbing bathroom bathtub;MET 2.0	capture24-24	unknown	unknown
home activity;household chores;house cleaning;floors;5140 sweeping garage sidewalk or outside of house;MET 4.0	capture24-25	unknown	unknown
home activity;miscellaneous;standing;5146 standing packing/unpacking household items occasional lifting;MET 3.5	capture24-26	unknown	unknown
home activity;miscellaneous;standing;5146 standing packing/unpaking household items occational lifting;MET 3.5	capture24-27	unknown	unknown
home activity;miscellaneous;walking;5147 walking moving away light items (pens/papers/keys not included);MET 3.0	capture24-28	unknown	unknown
home activity;miscellaneous;walking;5165 (generic) walking non-cleaning task such as closing windows lock door putting away items;MET 3.5	capture24-29	unknown	unknown
home activity;leisure;activties for maintenance of a household;with children;5170 sitting playing with child(ren);MET 2.2	capture24-30	unknown	unknown
home activity;leisure;activities for maintenance of a household;with children;5170 sitting playing with child(ren);MET 2.2	capture24-31	unknown	unknown
home activity;leisure;activties for maintenance of a household;5170 sitting playing with child(ren);MET 2.2	capture24-32	unknown	unknown
leisure;recreation;outdoor;5171 standing playing with child(ren);MET 2.8	capture24-33	unknown	unknown
leisure;recreation;outdoor;5175 walking/running playing with child(ren);MET 3.5	capture24-34	unknown	unknown
home activity;child/elderly/pet care;child care;5181 walking and carrying child;MET 3.0	capture24-35	unknown	unknown
home activity;child/elderly/pet care;child care;5183 standing holding child;MET 2.0	capture24-36	unknown	unknown
home activity;child/elderly/pet care;child care;5185 child care sitting/kneeling occasional lifting;MET 2.0	capture24-37	unknown	unknown
home activity;child/elderly/pet care;child care;5185 child care sitting/kneeling;MET 2.0	capture24-38	unknown	unknown
home activity;child/elderly/pet care;child care;5185 child care sitting/kneeling occasional lifting;MET 2.0	capture24-39	unknown	unknown
home activity;child/elderly/pet care;child care;5186 child care standing occasional lifting;MET 3.0	capture24-40	unknown	unknown
home activity;miscellaneous;sitting;7010 sitting/lying and watching television with TV on as the primary activity;MET 1.0	capture24-41	sitstand	sitting
home activity;miscellaneous;sitting;7010 lying and watching television with TV on as the primary activity;MET 1.0	capture24-42	sitstand	sitting
transportation;waiting;7021 sitting;MET 1.3	capture24-43	sitstand	sitting
home activity;miscellaneous;sitting;7021 sitting without observable actiivties;MET 1.3	capture24-44	sitstand	sitting
home activity;miscellaneous;sitting;7021 sitting without observable activities;MET 1.3	capture24-45	sitstand	sitting
7030 sleeping;MET 0.95	capture24-46	lying	lying
transportation;waiting;7040 standing in a line;MET 1.3	capture24-47	sitstand	standing
occupation;interruption;standing;9015 standing scanning documents;MET 1.5	capture24-48	unknown	unknown
occupation;interruption;9015 standing scanning documents;MET 1.5	capture24-49	unknown	unknown
home activity;miscellaneous;standing;9020 standing writing/drawing/painting;MET 1.8	capture24-50	unknown	unknown
occupation;interruption;standing;9020 standing writing/drawing/painting;MET 1.8	capture24-51	unknown	unknown
leisure;miscellaneous;standing;9020 standing writing/drawing/painting;MET 1.8	capture24-52	unknown	unknown
occupation;interruption;9020 standing writing/drawing/painting;MET 1.8	capture24-53	unknown	unknown
home activity;miscellaneous;sitting;9030 sitting desk entertainment/hobby (with or without eating at the same time);MET 1.3	capture24-54	unknown	unknown
home activity;miscellaneous;sitting;9030 sitting desk work (with or without eating at the same time);MET 1.3	capture24-55	unknown	unknown
home activity;miscellaneous;sitting;9030 sitting desk work (with or without eating at the same time);MET 1.3	capture24-56	unknown	unknown
home activity;miscellaneous;standing;9050 standing talking in person/on the phone/computer (skype chatting) or using a mobile phone/smartphone/tablet;MET 1.8	capture24-57	unknown	unknown
leisure;miscellaneous;standing;9050 standing talking in person/using a phone/smartphone/tablet;MET 1.8	capture24-58	unknown	unknown
occupation;interruption;standing;9050 standing talking in person/using a phone/smartphone/tablet;MET 1.8	capture24-59	unknown	unknown
occupation;interruption;9050 standing talking in persone/using a phone/smartphone/tablet;MET 1.8	capture24-60	unknown	unknown
transportation;walking;9050 standing miscellaneous (talking to others etc.);MET 1.8	capture24-61	unknown	unknown
home activity;miscellaneous;standing;9050 standing talking in person on the phone/computer (skype chatting) or using a mobileo phone/smartphone/tablet;MET 1.8	capture24-62	unknown	unknown
occupation;interruption;standing;9050 standing talking in persone/using a phone/smartphone/tablet;MET 1.8	capture24-63	unknown	unknown
home activity;miscellaneous;sitting;9055 sitting/lying talking in person/using a mobile phone/smartphone/tablet or talking on the phone/computer (skype chatting);MET 1.5	capture24-64	unknown	unknown
leisure;miscellaneous;sitting;9055 sitting talking to person/using the phone;MET 1.5	capture24-65	unknown	unknown
occupation;interruption;sitting;9055 sitting using a mobile phone/smartphone/tablet or talking on the phone/computer (skype meeting etc.);MET 1.5	capture24-66	unknown	unknown
occupation;interruption;9055 sitting using a mobile phone/smartphone/tablet or talking on the phone/computer (skype meeting etc.);MET 1.5	capture24-67	unknown	unknown
home activity;miscellaneous;sitting;9060 sitting/lying reading or without observable/identifiable activities;MET 1.3	capture24-68	sitstand	sitting
leisure;miscellaneous;sitting;9060 (generic) sitting/lying reading or without observable/identifiable activities;MET 1.3	capture24-69	sitstand	sitting
home activity;miscellaneous;sitting;9060 sitting/lying reading or without observable activities;MET 1.3	capture24-70	sitstand	sitting
occupation;interruption;sitting;9060 sitting without observable/identifiable activities;MET 1.3	capture24-71	sitstand	sitting
leisure;miscellaneous;sitting;9060 (generic) sitting/lying reading or without observable activities;MET 1.3	capture24-72	sitstand	sitting
occupation;interruption;9060 (generic) sitting without observable activities;MET 1.3	capture24-73	sitstand	sitting
occupation;interruption;9060 (generic) sitting without observable/identifiable activities;MET 1.3	capture24-74	sitstand	sitting
home activity;miscellaneous;sitting;9060 sitting reading or using a mobile phone/smartphone/tablet or talking on the phone/computer (skype chatting);MET 1.3	capture24-75	sitstand	sitting
occupation;interruption;9060 sitting using a mobile phone/smartphone/tablet or talking on the phone/computer (skype meeting etc.);MET 1.3	capture24-76	sitstand	sitting
home activity;miscellaneous;standing;9070 standing reading or without observable/identifiable activities;MET 1.8	capture24-77	sitstand	standing
leisure;miscellaneous;standing;9070 standing reading or without observable/identifiable activities;MET 1.8	capture24-78	sitstand	standing
occupation;interruption;standing;9070 standing reading or without observable/identifiable activities;MET 1.8	capture24-79	sitstand	standing
home activity;miscellaneous;standing;9070 standing reading or without obvious activities;MET 1.8	capture24-80	sitstand	standing
occupation;interruption;9070 standing reading or without observable/identifiable activities;MET 1.8	capture24-81	sitstand	standing
leisure;miscellaneous;standing;9070 standing reading or without obvious activities;MET 1.8	capture24-82	sitstand	standing
occupation;interruption;9070 standing reading or without obvious activities;MET 1.8	capture24-83	sitstand	standing
home activity;miscellaneous;standing;9071 (generic) standing miscellaneous;MET 2.5	capture24-84	sitstand	standing
leisure;miscellaneous;standing;9071 (generic) standing miscellaneous;MET 2.5	capture24-85	sitstand	standing
occupation;interruption;standing;9071 (generic) standing miscellaneous;MET 2.5	capture24-86	sitstand	standing
leisure;miscellaneous;9071 (generic) standing miscellaneous indoor or outdoor;MET 2.5	capture24-87	sitstand	standing
transportation;walking;9071 standing miscellaneous (talking to others etc.);MET 2.5	capture24-88	sitstand	standing
home activity;leisure;activities for maintenance of a household;miscellaneous;9100 retreat/family reunion activities involving sitting eating relaxing talking with more than one person;MET 1.8	capture24-89	unknown	unknown
home activity;leisure;activties for maintenance of a household;miscellaneous;9100 retreat/family reunion activities involving sitting eating relaxing talking with more than one person;MET 1.8	capture24-90	unknown	unknown
home activity;leisure;activties for maintenance of a household;9100 retreat/family reunion activities involving sitting eating relaxing talking with more than one person;MET 1.8	capture24-91	unknown	unknown
home activity;leisure;activities for maintenance of a household;miscellaneous;9101 retreat/family reunion activities playing games with more than one person;MET 3.0	capture24-92	unknown	unknown
occupation;miscellaneous;11475 (generic) manual labour;MET 2.8	capture24-93	unknown	unknown
occupation;miscellaneous;11475 (generic) manual or unskilled labour;MET 2.8	capture24-94	unknown	unknown
occupation;office and administrative support;11580 office/computer work general;MET 1.5	capture24-95	unknown	unknown
occupation;office and administrative support;11580 office wok/computer work general;MET 1.5	capture24-96	unknown	unknown
home activity;miscellaneous;sitting;11580 office/computer work general;MET 1.5	capture24-97	unknown	unknown
home activity;miscellaneous;sitting;11580 office work such as writing and typing (with or without eating at the same time);MET 1.5	capture24-98	unknown	unknown
occupation;office and administrative support;11580 office work/computer work general;MET 1.5	capture24-99	unknown	unknown
occupation;interruption;sitting;11585 sitting meeting/talking to colleagues with or without eating;MET 1.5	capture24-100	unknown	unknown
occupation;interruption;11585 sitting meeting/talking to colleages with or without eating;MET 1.5	capture24-101	unknown	unknown
occupation;interruption;11585 sitting meeting/talking to colleagues with or without eating;MET 1.5	capture24-102	unknown	unknown
occupation;office and administrative support;11600 (generic) standing tasks such as store clerk/libarian/packing boxes/repair heavy parts;MET 3.0	capture24-103	unknown	unknown
occupation;office and administrative support;11600 (generic) standing tasks such as store clerk/librarian/packing boxes/repair heavy parts;MET 3.0	capture24-104	unknown	unknown
occupation;miscellaneous;11615 (generic) standing lifting items continuously with limited walking;MET 4.5	capture24-105	unknown	unknown
occupation;interruption;11791 walking on job in office or lab area;MET 2.0	capture24-106	walking	walk_slow
occupation;interruption;walking;11791 walking on job in office or lab area;MET 2.0	capture24-107	walking	walk_slow
occupation;interruption;walking;11795 walking on job and carrying light objects such as boxes or pushing trolleys;MET 3.5	capture24-108	unknown	unknown
occupation;interruption;11795 walking on job and carrying light objects such as boxes or pushing trolleys;MET 3.5	capture24-109	unknown	unknown
leisure;sports;gymnasium and athletics;athletics;12150 running;MET 8.0	capture24-110	sports	jogging
transportation;walking;12150 running;MET 8.0	capture24-111	sports	jogging
home activity;self care;13000 getting ready for bed standing;MET 2.3	capture24-112	unknown	unknown
occupation;interruption;miscellaneous;13009 toilet break;MET 1.8	capture24-113	unknown	unknown
home activity;self care;13009 toilet eliminating or squatting;MET 1.8	capture24-114	unknown	unknown
occupation;interruption;13009 toilet break;MET 1.8	capture24-115	unknown	unknown
home activity;self care;13020 dressing/undressing;MET 2.5	capture24-116	unknown	unknown
home activity;eating;13030 eating sitting alone or with someone;MET 1.5	capture24-117	unknown	unknown
leisure;eating;social;13030 eating sitting indoor/outdoor;MET 1.5	capture24-118	unknown	unknown
leisure;eating;13030 eating sitting indoor/outdoor;MET 1.5	capture24-119	unknown	unknown
occupation;interruption;13030 eating sitting;MET 1.5	capture24-120	unknown	unknown
occupation;interruption;sitting;13030 eating sitting;MET 1.5	capture24-121	unknown	unknown
leisure;eating;not-social;13030 eating sitting indoor/outdoor;MET 1.5	capture24-122	unknown	unknown
home activity;eating;13035 eating standing alone or with others;MET 2.0	capture24-123	unknown	unknown
occupation;interruption;13035 eating standing;MET 2.0	capture24-124	unknown	unknown
leisure;eating;social;13035 eating standing indoor/outdoor;MET 2.0	capture24-125	unknown	unknown

Table A.8: Label translation for test dataset Capture 24-2. *The second part of the label translation of Capture 24.*

annotation	annotation_id	label-group	label-use
leisure;eating;13035 eating standing indoor/outdoor;MET 2.0	capture24-126	unknown	unknown
occupation;interruption;standing;13035 eating standing;MET 2.0	capture24-127	unknown	unknown
leisure;eating;not-social;13035 eating standing indoor/outdoor;MET 2.0	capture24-128	unknown	unknown
home activity;self care;13040 (generic) self care such as grooming/washing hands/shaving/brushing teeth/putting on make-up not eliminating and bathing (not necessary in the toilet);MET 2.0	capture24-129	unknown	unknown
home activity;self care;13045 hairstyling standing;MET 2.5	capture24-130	unknown	unknown
transportation;private transportation;16010 driving automobile or light truck (not a semi);MET 2.5	capture24-131	unknown	unknown
transportation;private transportation;16015 riding in a car or truck;MET 1.3	capture24-132	unknown	unknown
transportation;public transportation;16016 riding in a bus or train;MET 1.3	capture24-133	unknown	unknown
leisure;miscellaneous;walking;17031 loading /unloading a car implied walking;MET 3.5	capture24-134	unknown	unknown
leisure;miscellaneous;17031 loading /unloading a car implied walking;MET 3.5	capture24-135	unknown	unknown
occupation;interruption;walking;17070 walking downstairs;MET 3.5	capture24-136	walking	stairs
leisure;miscellaneous;walking;17070 descending stairs;MET 3.5	capture24-137	walking	stairs
leisure;sports;miscellaneous;17082 hiking or walking at a normal pace through fields and hillsides;MET 5.0	capture24-138	walking	walk_mod
occupation;interruption;17133 walking upstairs;MET 4.0	capture24-139	walking	stairs
leisure;miscellaneous;walking;17133 walking upstairs;MET 4.0	capture24-140	walking	stairs
occupation;interruption;walking;17133 walking upstairs;MET 4.0	capture24-141	walking	stairs
home activity;miscellaneous;walking;17150 walking household without observable loads;MET 2.0	capture24-142	walking	walk_slow
transportation;walking;17161 walking not as the single means of transports e.g.from house to transports or vice versa/from car to places or vice versa/between transports;MET 2.5	capture24-143	walking	walk_slow
transportation;walking;17250 walking as the single means to a destination not to work or class;MET 3.0	capture24-144	walking	walk_mod
transportation;walking;17270 walking as the single means to work or class (not from);MET 3.5	capture24-145	walking	walk_mod
leisure;miscellaneous;sitting;21000 sitting meeting;MET 1.5	capture24-146	unknown	unknown
leisure;miscellaneous;21000 sitting meeting or talking with others;MET 1.5	capture24-147	unknown	unknown
leisure;miscellaneous;sitting;21005 (generic) sitting light office writing typing work;MET 1.5	capture24-148	unknown	unknown
leisure;miscellaneous;21005 (generic) sitting light office writing typing work;MET 1.5	capture24-149	unknown	unknown
home activity;miscellaneous;sitting;21010 sitting non-desk work (with or without eating at the same time);MET 2.5	capture24-150	unknown	unknown
leisure;miscellaneous;21010 sitting non-desk work (with or without eating at the same time);MET 2.5	capture24-151	unknown	unknown
leisure;miscellaneous;sitting;21016 sitting child care only active periods;MET 2.0	capture24-152	unknown	unknown
leisure;miscellaneous;21016 sitting child care only active periods;MET 2.0	capture24-153	unknown	unknown
leisure;miscellaneous;21017 standing child care only active periods;MET 3.0	capture24-154	unknown	unknown
leisure;miscellaneous;standing;21017 standing child care only active periods;MET 3.0	capture24-155	unknown	unknown
leisure;miscellaneous;walking;21070 (generic) walking and occasional standing (no more than two consecutive images);MET 2.5	capture24-156	walking	walk_slow
leisure;miscellaneous;21070 (generic) walking/standing combination indoor;MET 3.0	capture24-157	walking	walk_slow
sitting;sitstand+activity;social;MET 1.8	capture24-158	unknown	unknown
walking;MET 3.0	capture24-159	walking	walk_mod
household-chores;sitstand+activity;MET 2.5	capture24-160	unknown	unknown
sitting;sitstand+lowactivity;screen;MET 1.0	capture24-161	sitstand	sitting
household-chores;walking+activity;MET 3.0	capture24-162	unknown	unknown
manual-work;sitstand+activity;MET 5.0	capture24-163	unknown	unknown
sitting;sitstand+activity;MET 1.8	capture24-164	sitstand	sitting
household-chores;sitstand+lowactivity;MET 1.3	capture24-165	unknown	unknown
mixed-activity;sitstand+activity;MET 2.0	capture24-166	unknown	unknown
manual-work;walking+activity;MET 3.5	capture24-167	unknown	unknown
mixed-activity;sitstand+activity;MET 2.5	capture24-168	unknown	unknown
sports/gym;MET 3.5	capture24-169	unknown	unknown
manual-work;sitstand+activity;MET 2.0	capture24-170	unknown	unknown
sports/gym;MET 2.5	capture24-171	unknown	unknown
manual-work;walking+activity;MET 3.8	capture24-172	unknown	unknown
sports/gym;MET 6.0	capture24-173	unknown	unknown
manual-work;sitstand+activity;MET 5.5	capture24-174	unknown	unknown
sitting;sitstand+lowactivity;MET 1.3	capture24-175	unknown	unknown
manual-work;walking+activity;MET 3.0	capture24-176	unknown	unknown
sports/gym;MET 5.0	capture24-177	unknown	unknown
sports/gym;MET 5.5	capture24-178	unknown	unknown
sports/gym;MET 8.0	capture24-179	unknown	unknown
manual-work;walking+activity;MET 5.5	capture24-180	unknown	unknown
sports/gym;MET 3.0	capture24-181	unknown	unknown
household-chores;sitstand+activity;social;MET 4.5	capture24-182	unknown	unknown
manual-work;sitstand+activity;MET 3.0	capture24-183	unknown	unknown
home activity;child/elderly/pet care;MET 2.3	capture24-184	unknown	unknown
sports/gym;MET 8.5	capture24-185	unknown	unknown
home activity;leisure;mixed-activity;walking+activity;MET 3.0	capture24-186	unknown	unknown
home activity;household chores;house cleaning;MET 2.5	capture24-187	unknown	unknown
standing;sitstand+activity;social;MET 2.5	capture24-188	unknown	unknown
manual-work;sitstand+activity;MET 1.5	capture24-189	unknown	unknown
sports/gym;MET 7.0	capture24-190	unknown	unknown
sports/gym;MET 5.3	capture24-191	unknown	unknown
vehicle;MET 2.8	capture24-192	unknown	unknown
mixed-activity;walking+activity;MET 2.0	capture24-193	unknown	unknown
sitting;sitstand+lowactivity;MET 1.5	capture24-194	unknown	unknown
home activity;child/elderly/pet care;pet care;MET 2.5	capture24-195	unknown	unknown
mixed-activity;MET 4.5	capture24-196	unknown	unknown
sports/gym;MET 2.8	capture24-197	unknown	unknown
sports/gym;MET 9.0	capture24-198	unknown	unknown
household-chores;sitstand+activity;MET 4.0	capture24-199	unknown	unknown
standing;sitstand+activity;MET 2.0	capture24-200	unknown	unknown
standing;sitstand+activity;social;MET 1.8	capture24-201	unknown	unknown
home activity;leisure;sitting;sitstand+activity;MET 2.5	capture24-202	unknown	unknown
carrying heavy loads;MET 8.0	capture24-203	unknown	unknown
manual-work;MET 8.0	capture24-204	unknown	unknown
household-chores;sitstand+lowactivity;MET 2.8	capture24-205	unknown	unknown
vehicle;MET 1.3	capture24-206	unknown	unknown

Table A.9: Optimal Hyperparameters Selected During Cross-Validation for Conventional ML Models. *The columns "Excl. dataset" and "Excl. transformation" respectively indicate the training dataset and transformation techniques that were excluded from the experiment. Due to space constraints, we abbreviate the transformation names to their first word, or "all" for all transformations.*

Excl.	Excl.	RF			SVM	
dataset	transformation	max_features	max_depth	n_estimators	C	loss
N/A	N/A	1.0	64	500	1	squared_hinge
Wisdm	N/A	1.0	64	500	1	squared_hinge

(a) Flat ML

Excl.	Excl.	RF			SVM	
dataset	transformation	max_features	max_depth	n_estimators	C	loss
N/A	N/A	sqrt	64	500	100	hinge
Adl_hmp	N/A	sqrt	64	500	100	hinge
Act_cp	N/A	sqrt	64	500	100	hinge
Selfback	N/A	sqrt	64	500	100	hinge
Gotov	N/A	sqrt	64	500	0.1	hinge
Wisdm	N/A	sqrt	64	300	100	hinge
Wisdm	all	sqrt	32	300	100	hinge
Wisdm	amplitude	sqrt	64	500	100	hinge
Wisdm	reverting	sqrt	64	500	100	hinge
Wisdm	rotation	sqrt	64	500	10	hinge
Wisdm	swapping	sqrt	64	500	10	hinge

(b) Group classifier in hierarchical ML

Excl.	Excl.	RF			SVM	
dataset	transformation	max_features	max_depth	n_estimators	C	loss
N/A	N/A	1.0	64	500	1	hinge
Adl_hmp	N/A	1.0	64	500	10	hinge
Act_cp	N/A	1.0	64	500	100	hinge
Selfback	N/A	1.0	64	500	1	hinge
Gotov	N/A	1.0	64	300	10	hinge
Wisdm	N/A	1.0	64	100	10	hinge
Wisdm	All	sqrt	32	300	100	hinge
Wisdm	amplitude	1.0	64	500	0.1	hinge
Wisdm	reverting	sqrt	64	500	100	hinge
Wisdm	rotation	1.0	64	500	0.1	hinge
Wisdm	swapping	1.0	64	500	0.1	squared_hinge

(c) Label classifier for "walking" group in hierarchical ML

Excl.	Excl.	RF			SVM	
dataset	transformation	max_features	max_depth	n_estimators	C	loss
N/A	N/A	1.0	64	500	1	hinge
Adl_hmp	N/A	1.0	32	500	0.1	hinge
Act_cp	N/A	1.0	32	500	100	hinge
Selfback	N/A	1.0	32	500	0.1	hinge
Gotov	N/A	1.0	32	500	10	hinge
Wisdm	N/A	1.0	64	100	1	hinge
Wisdm	All	sqrt	16	100	1	hinge
Wisdm	amplitude	sqrt	32	500	1	hinge
Wisdm	reverting	sqrt	32	500	100	hinge
Wisdm	rotation	sqrt	32	500	0.1	hinge
Wisdm	swapping	sqrt	32	500	100	hinge

(d) Label classifier for "sitstand" group in hierarchical ML

List of Figures

List of Tables

Bibliography

Acampora, G., Cook, D. J., Rashidi, P., and Vasilakos, A. V. (2013). A survey on ambient intelligence in healthcare. *Proceedings of the IEEE*, 101(12):2470–2494.

Banos, O., Galvez, J.-M., Damas, M., Pomares, H., and Rojas, I. (2014). Window size impact in human activity recognition. *Sensors*, 14(4):6474–6499.

Bao, L. and Intille, S. S. (2004). Activity recognition from user-annotated acceleration data. In *International Conference on Pervasive Computing*, pages 1–17. Springer.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T., and Zaccaria, R. (2013). Analysis of human behavior recognition algorithms based on acceleration data. In *2013 IEEE International Conference on Robotics and Automation*, pages 1602–1607. IEEE.

Bulling, A., Blanke, U., and Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):1–33.

Chan Chang, S. and Doherty, A. (2021). *Capture-24: Activity tracker dataset for human activity recognition*. University of Oxford.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20:273–297.

Cruciani, F., Vafeiadis, A., Nugent, C., Cleland, I., McCullagh, P., Votis, K., Giakoumis, D., Tzovaras, D., Chen, L., and Hamzaoui, R. (2020). Feature learning for human activity recognition using convolutional neural networks: A case study for inertial measurement unit and audio data. *CCF Transactions on Pervasive Computing and Interaction*, 2(1):18–32.

Dhamija, A. R., Günther, M., and Boult, T. (2018). Reducing network agnostophobia. *Advances in Neural Information Processing Systems*, 31.

Doherty, A., Jackson, D., Hammerla, N., Plötz, T., Olivier, P., Granat, M. H., White, T., Van Hees, V. T., Trenell, M. I., Owen, C. G., et al. (2017). Large scale population assessment of physical activity using wrist worn accelerometers: the uk biobank study. *PloS one*, 12(2):e0169649.

Garcia-Gonzalez, D., Rivero, D., Fernandez-Blanco, E., and Luaces, M. R. (2020). A public domain dataset for real-life human activity recognition using smartphone sensors. *Sensors*, 20(8).

Gjoreski, H., Gams, M., and Lustrek, M. (2011). Accelerometer placement for posture recognition and fall detection. In *2011 Seventh International Conference on Intelligent Environments*, pages 47–54. IEEE.

Heng, X., Wang, Z., and Wang, J. (2016). Human activity recognition based on transformed accelerometer data from a mobile phone. *International Journal of Communication Systems*, 29(13):1981–1991.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Jain, A. and Kanhangad, V. (2015). Exploring orientation and accelerometer sensor data for personal authentication in smartphones using touchscreen gestures. *Pattern Recognition Letters*, 68:351–360. Special Issue on "Soft Biometrics".

Kalouris, G., Zacharaki, E. I., and Megalooikonomou, V. (2019). Improving CNN-based activity recognition by data augmentation and transfer learning. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 1387–1394.

Kittler, J., Hatef, M., Duin, R., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239.

Klapuri, A. and Davy, M. (2006). *Signal Processing Methods for Music Transcription*. Springer Science & Business Media.

Kwapisz, J. R., Weiss, G. M., and Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82.

Lara, O. D. and Labrador, M. A. (2012). A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209.

Leotta, M., Fasciglione, A., and Verri, A. (2021). Daily living activity recognition using wearable devices: A features-rich dataset and a novel approach. In Del Bimbo, A. et al., editors, *Proceedings of 25th International Conference on Pattern Recognition Workshops (ICPR 2021 Workshops)*, volume 12662 of *LNCS*, pages 171–187. Springer.

Leutheuser, H., Schuldhaus, D., and Eskofier, B. M. (2013). Hierarchical, multi-sensor based classification of daily life activities: Comparison with state-of-the-art algorithms using a benchmark dataset. *PLOS ONE*, 8(10):1–11.

Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Liu, J., Zhong, L., Wickramasuriya, J., and Vasudevan, V. (2009). uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675. PerCom 2009.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2.

Mannini, A. and Sabatini, A. M. (2010). Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2):1154–1175.

Noori, F. M., Riegler, M., Uddin, M. Z., and Torresen, J. (2020a). Human activity recognition from multiple sensors data using multi-fusion representations and CNNs. *ACM Transactions on Multimedia Computing Communications and Applications*, 16(2):1–19.

Noori, F. M., Riegler, M., Uddin, M. Z., and Torresen, J. (2020b). Human activity recognition from multiple sensors data using multi-fusion representations and cnns. *ACM Trans. Multimedia Comput. Commun. Appl.*, 16(2).

Paraschiakos, S., (Marian), B. M., (Arno), K. A., (Ricardo), C. R., and (Eline), S. P. (2021). Gotov human physical activity and energy expenditure dataset on older individuals. 4TU.ResearchData.

Piczak, K. J. (2015). Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6.

Qin, Y., Carlini, N., Cottrell, G., Goodfellow, I., and Raffel, C. (2019). Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5231–5240. PMLR.

Qureshi, U. and Golnaraghi, F. (2017). An algorithm for the in-field calibration of a MEMS IMU. *IEEE Sensors Journal*, 17(22):7479–7486.

Ramanujam, E., Perumal, T., and Padmavathi, S. (2021). Human activity recognition with smartphone and wearable sensors using deep learning techniques: A review. *IEEE Sensors Journal*, 21(12):13029–13040.

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. In Huff, K. and Bergstra, J., editors, *Proceedings of the 14th Python in Science Conference*, pages 130 – 136.

Sani, S., Wiratunga, N., Massie, S., and Cooper, K. (2016). Selfback–activity recognition for self-management of low back pain. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 281–294. Springer.

Shannon, C. (1949). Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21.

Shoaib, M., Bosch, S., Incel, O. D., Scholten, H., and Havinga, P. J. (2014). Fusion of smartphone motion sensors for physical activity recognition. *Sensors*, 14(6):10146–10176.

Shoaib, M., Bosch, S., Incel, O. D., Scholten, H., and Havinga, P. J. (2015). A survey of online activity recognition using mobile phones. *Sensors*, 15(1):2059–2085.

Smith, J. O. (2011). Spectral audio signal processing. *Center for Computer Research in Music and Acoustics (CCRMA)*, pages 1–674.

Tang, C. I., Perez-Pozuelo, I., Spathis, D., and Mascolo, C. (2020). Exploring contrastive learning in human activity recognition for healthcare. *arXiv preprint arXiv:2011.11542*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Wang, H. and Chen, P. (2009). A feature extraction method based on information theory for fault diagnosis of reciprocating machinery. *Sensors*, 9(4):2415–2436.

Wang, J., Chen, Y., Hao, S., Peng, X., and Hu, L. (2019). Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11. Deep Learning for Pattern Recognition.

Weiss, G. M., Yoneda, K., and Hayajneh, T. (2019). Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access*, 7:133190–133202.

Wu, W., Dasgupta, S., Ramirez, E. E., Peterson, C., Norman, G. J., et al. (2012). Classification accuracies of physical activities using smartphone motion sensors. *Journal of medical Internet research*, 14(5):e2208.

Yang, J., Nguyen, M. N., San, P. P., Li, X., and Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, pages 3995–4001. Buenos Aires, Argentina.

Yoo, S. S., Kabir, H., and Lee, H.-S. (2023). Multiple imu sensor fusion using resolution refinement method to reduce quantization error. *Journal of Mechanical Science and Technology*, pages 1–6.

Yuan, H., Chan, S., Creagh, A. P., Tong, C., Clifton, D. A., and Doherty, A. (2022). Self-supervised learning for human activity recognition using 700,000 person-days of wearable data.