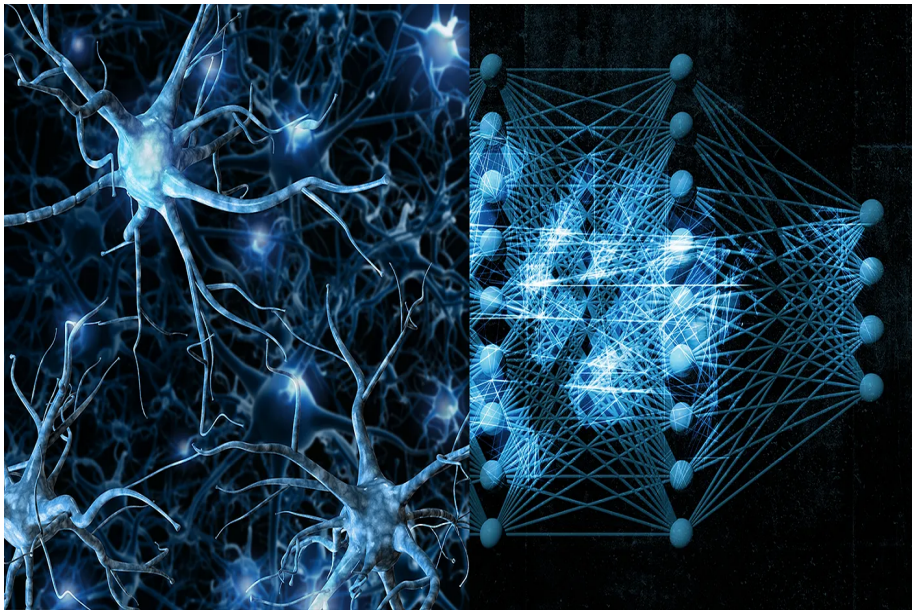
ISEL



Real-time Human Activity Recognition with KANs

SAMUEL SAMPAIO COSTA
(Licenciado)

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Orientadores: Prof. Doutora Matilde Pós-de-Mina Pato
Prof. Doutor Nuno Miguel Soares Datia

Júri:

Presidente:	Prof. Dr. Tiago Miguel Braga da Silva Dias
Vogais:	Prof. Dr. Cátia Luísa Santana Calisto Pesquita
	Prof. Dr. Matilde Pós-de-Mina Pato

Dezembro 2024

Real-time Human Activity Recognition with KANs

SAMUEL SAMPAIO COSTA
(Licenciado)

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Orientadores: Prof. Doutora Matilde Pós-de-Mina Pato, ISEL
Prof. Doutor Nuno Miguel Soares Datia, ISEL

Júri:

Presidente:	Prof. Dr. Tiago Miguel Braga da Silva Dias, ISEL, IPL
Vogais:	Prof. Dr. Cátia Luísa Santana Calisto Pesquita, FCUL, UL
	Prof. Dr. Matilde Pós-de-Mina Pato, ISEL, IPL

Dezembro 2024

Acknowledgements

I express my deepest gratitude to my supervisors, Prof. Dr. Matilde Pato and Prof. Dr. Nuno Datia, for their unwavering support, guidance, and encouragement throughout the research and writing of this thesis. Their expertise, insightful feedback, and patience have been invaluable in completing this work.

I also sincerely thank my employer, Celfocus, for their understanding and support during this process. I am particularly grateful for the generously provided time off that allowed me to dedicate the necessary time and focus to this research.

As I reflect on my journey at ISEL, which began in 2016, I am reminded of the significant personal and professional transformation that these years have brought. At that time, I was employed as an office clerk in the Portuguese Army with limited career prospects. With the help of family members, faculty and the collaboration of my peers, I experienced the transformative power of education. This experience has profoundly shaped my professional path and deepened my sense of purpose and contribution to society.

I want to thank my mother Emília and my stepfather Francisco for their kind and good-hearted influence.

Finally, I wish to express my love to my partner, Sara, for her unwavering support and understanding, particularly during the many evenings spent alone with our son while i was attending classes, and during the weekends. I also extend my thanks to her parents São and António for embracing me and giving me more than a helping hand in this journey.

Statement of integrity

I declare that this **dissertation** is the result of my personal and independent research. Its content is original, and all sources listed in the bibliographic references were consulted and are duly mentioned in the text. I further declare that all scientific and technical references relevant to the development of the work are duly cited and included in the bibliographic references.

Samuel Sampaio Costa

Lisbon, December 8th, 2024

Real-time Human Activity Recognition with KANs

This document was created using the (PDF/A)LaTeX processor, based in the "iselthesis" template [44], developed at the DEETC of ISEL-IPL.

Abstract

Kolmogorov-Arnold Networks (KANs) represent a breakthrough in deep learning by generalizing the Kolmogorov-Arnold Theorem (KAT) to networks of arbitrary depth and width. This theorem facilitates the decomposition of multivariate functions into constituent one-dimensional elements, with learnable activation functions on weights and the sum operator on nodes. KANs have been shown to exhibit robust performance in function approximation, validated across mathematical, physical, and practical domains such as traffic prediction and medical diagnostics. This study focuses on 3 objectives: (1) validating that KANs may be used in classification tasks applied to the real-world by comprehensive evaluations on OpenML, Kaggle and UCI datasets, (2) enhancing Human Activity Recognition (HAR) systems using KANs, and (3) applying these networks to Real-Time HAR in mobile devices. Tests were conducted to evaluate KANs with 7 different kernels. They demonstrate high classification performance compared to conventional machine learning approaches and MLP, and showcase reasonable latency for real-time HAR detection. These findings underscore KANs' potential as scalable, interpretable tools in modern machine learning applications given their favorable neural scaling laws.

Keywords: Classification, Kolmogorov-Arnold Networks, Kernel function, Multivariate functions, Human Activity Recognition, Real-time Classification

Resumo

As redes de Kolmogorov-Arnold representam um avanço na aprendizagem profunda ao generalizar o teorema da representação de Kolmogorov-Arnold para redes de profundidade e largura arbitrárias. Este teorema facilita a decomposição de funções multivariadas nos seus elementos unidimensionais constituintes, com funções de ativação passíveis de serem treinadas nos pesos e o operador de soma nos nós. Foi demonstrado que as KANs exibem um desempenho robusto na aproximação de funções, nos domínios da matemática, física e em problemas do mundo real, como a previsão de tráfego e o diagnóstico médico. Este estudo foca-se em 3 objetivos: (1) validar que as KANs podem ser usadas em tarefas de classificação aplicadas ao mundo real através de avaliações abrangentes em conjuntos de dados abertos do OpenML, Kaggle e UCI, (2) melhorar sistemas de reconhecimento de actividade humana (RAH) utilizando as KANs, e (3) aplicar estas redes ao RAH em tempo real em dispositivos móveis. Foram realizados testes para avaliar as KANs com 7 tipos de funções núcleo. Estes demonstram um alto desempenho de classificação em comparação com abordagens convencionais de aprendizagem automática e perceptrões multi-camada, e apresentam uma latência razoável para RAH em tempo real. Estes resultados destacam o potencial das KANs como ferramentas escaláveis e interpretáveis nas aplicações modernas de aprendizagem automática, dadas as suas leis de escalabilidade neural favoráveis.

Palavras-chave: Classificação, Redes de Kolmogorov-Arnold, Função núcleo, Funções multivariadas, Reconhecimento de Atividade Humana, Classificação em tempo real

Contents

List of Figures

List of Tables

Acronyms

AAL	Ambient Assisted Living 27
ANN	Artificial Neural Network 14
Bi-GRU	Bi-directional Gated Recurrent Unit 15
Bi-GRU-I	Bi-directional Gated Recurrent Unit with Inception Architecture 15
BLSM	Bayesian Latent Space Model 14
CCTV	Closed-Circuit Television 13
CNN	Convolutional Neural Network 13, 14, 15
DL	Deep Learning 4, 7, 8, 10, 15, 17
DMM	Depth Motion Map 12, 13
DNN	Deep Neural Network 29
GPU	Graphics Processing Unit 7
HAR	Human Activity Recognition ix, xv, xvii, 1, 2, 3, 4, 5, 11, 12, 13, 14, 15, 17, 18, 19, 20, 23, 25, 26, 27, 29, 30, 31, 32, 42
HOG	Histogram of Oriented Gradients 13
KANs	Kolmogorov-Arnold Networks ix, xi, xvii, 4, 5, 7, 8, 9, 10, 18, 20, 22, 27, 29, 30, 32, 33, 35, 37, 38, 39, 41, 42
KAT	Kolmogorov-Arnold Theorem ix, 4, 8, 9, 10
kNN	k-Nearest Neighbors 14
LSTM	Long Short-Term Memory 14, 15
ML	Machine Learning 4, 10, 12, 13, 15, 17, 20, 21, 23, 25, 26, 27
MLP	Multi-Layer Perceptron ix, 4, 7, 8, 9, 10, 14, 27, 28, 29, 41
MSR-Action3D	Microsoft Research Action 3D 12
PCA	Principal Component Analysis 13

RBF Radial Basis Functions 4, 27, 28, 29, 30, 33, 35
RF Random Forest 28

SGD Stochastic Gradient Descent 22
SRC Sparse Representation-based Classification 13
SVM Support Vector Machine 13, 14, 28, 29

UAT Universal Approximation Theorem 4, 8
UI User Interface 18, 23
UML Unified Modeling Language xv, 18, 24

1 Introduction

Over the past decade, the field of Human Activity Recognition (HAR) has seen a marked increase in academic interest [1, 3, 10, 15, 23, 27, 30, 42, 56, 57, 63]. This increase can be attributed to advancements in sensor technology, both in terms of affordability and precision. According to DigiKey [9], in 2012 it was expected that prices of sensors fell by 10% by 2013 and by 3% in 2014. The adoption of accelerometer and gyroscope sensors in smartphones has progressively risen. While the first, due to its inexpensiveness, has been incorporated from a very early stage, the latter was first introduced in premium flagship models such as iPhone 4[1] and has been widely adopted, being present today in entry-level phones such as Xiaomi Mi A1[2]. Smartphone sensors have enabled applications in gaming, healthcare, and security, including activities like flight simulation and golf [28].

As the range of sensor applications expands, so does the complexity of methods for HAR. HAR methods are categorized by signal type, label availability, and data processing mode. The 3D classification matrix in Figure 1.1 allows us to visualized these categories using three distinct axes, according to synchronicity between data collection and algorithm execution, type of signal used (video/sensor) and label availability:

- **Video-Based/Sensor-Based (X-axis)**:
  - **Video-Based**: Methods use camera footage to recognize human activities. These methods can capture rich visual information but may be limited by lighting conditions, privacy concerns, and the need for a clear line of sight.
  - **Sensor-Based**: Methods rely on wearable or environmental sensors (e.g., accelerometers, gyroscopes) to detect and classify activities. These methods are less affected by lighting and can provide continuous monitoring, though they may be limited by sensor placement and calibration.
- **Supervised/Non-Supervised (Y-axis)**:

[1] iFixit. "iPhone 4 Gyroscope Teardown" (2011). URL: https://web.archive.org/web/20111124144402/http://www.ifixit.com/Teardown/iPhone-4-Gyroscope-Teardown/3156/1. Accessed 2024-08-17

[2] DeviceSpecifications. "Xiaomi Mi A1 Review" (2017). URL: https://www.devicespecifications.com/en/editor-review/f4bb52/6. Accessed 2024-08-17

- **Supervised**: Methods require labeled training data to learn how to recognize different activities. They often achieve higher accuracy but depend on the availability of extensive and accurately labeled datasets.
- **Non-Supervised**: Methods do not rely on labeled data and can identify patterns or anomalies in the data without predefined categories. These methods are useful when labeled data is scarce but may require more sophisticated algorithms to interpret the results.

- **Online/Offline (Z-axis)**:
  - **Online**: Methods process data in real-time as it is collected, making them suitable for applications that require immediate feedback or decision-making.
  - **Offline**: Methods analyze pre-recorded data, which is useful for tasks that do not need real-time processing but can benefit from extensive batch analysis.

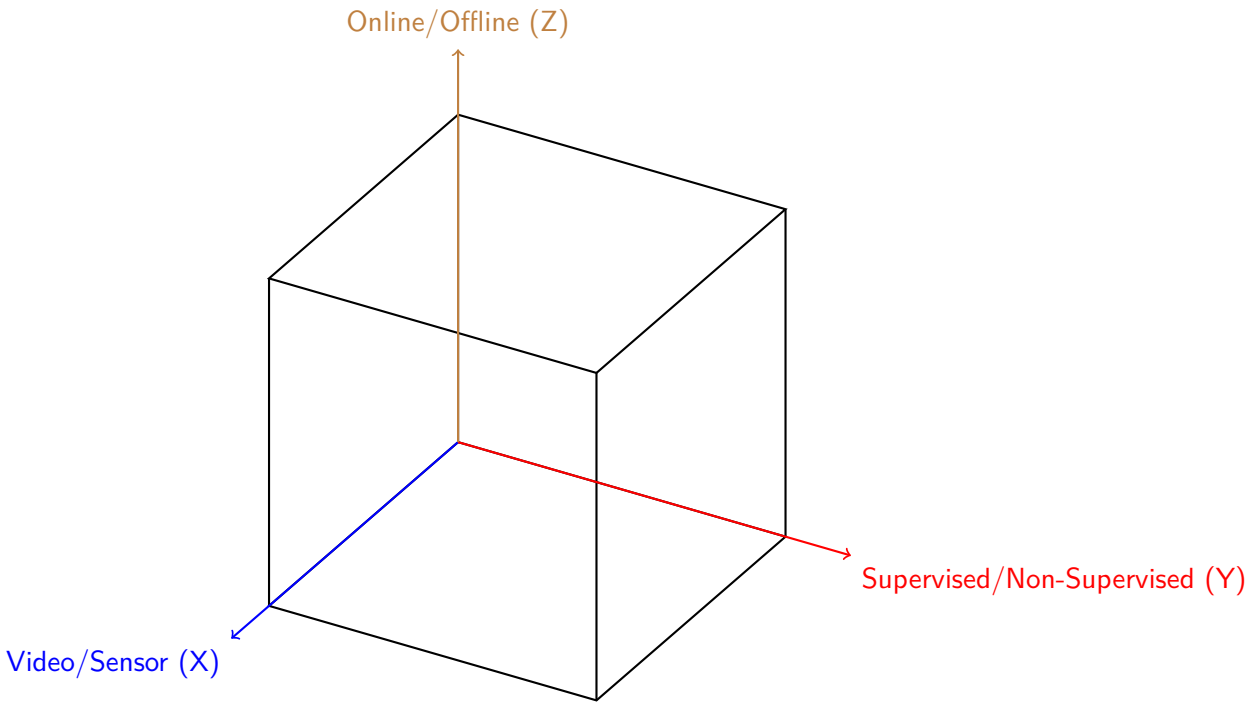


Figure 1.1: *3D Classification Matrix: Video/Sensor (X), Supervised/Non-Supervised (Y), Online/Offline (Z)*

The 3D classification matrix helps to understand how different HAR methods fit within the defined categories. Table 1.1 presents an overview of the methods we will discuss next.

Among these, **Online Video-Based Supervised** methods involve real-time processing of video data with supervised learning techniques. These methods are particularly suited for applications requiring immediate analysis of video feeds, such as real-time security surveillance systems. By using supervised learning, these methods can achieve high accuracy in activity recognition through the use of labeled training data.

In contrast, **Offline Video-Based Supervised** methods analyze pre-recorded video data using supervised techniques. These methods are useful for scenarios where detailed post-processing

is needed, such as in research studies or historical data analysis. The use of supervised learning allows for precise recognition based on comprehensive training datasets, but the analysis is not conducted in real-time.

Online Sensor-Based Supervised methods utilize real-time data from sensors and apply supervised learning algorithms to recognize activities. These methods are ideal for applications like wearable technology, where immediate feedback is necessary, such as in fitness trackers or health monitoring devices. The real-time processing ensures that users receive instant insights based on their activity data.

Offline Sensor-Based Supervised methods analyze sensor data collected over time with supervised learning techniques. This approach is suitable for in-depth analysis of activity patterns, such as in clinical studies or longitudinal research. The data is processed after collection, allowing for thorough evaluation and pattern recognition.

Online Video-Based Non-Supervised methods process video data in real-time without relying on labeled training data. These methods are employed in situations where immediate but unsupervised activity recognition is needed, such as detecting unusual behavior patterns in surveillance footage. The non-supervised approach allows for flexibility in scenarios where labeled data is not available.

Offline Video-Based Non-Supervised methods handle recorded video data without the use of supervised learning. This approach is useful for discovering patterns or anomalies in historical video footage when labeled data is not accessible. It allows for exploratory analysis and pattern identification without predefined categories.

Online Sensor-Based Non-Supervised methods process real-time sensor data with non-supervised techniques. They are utilized in dynamic environments where real-time, adaptive activity recognition is crucial, such as in smart home systems. These methods adapt to new patterns and conditions without the need for pre-labeled data.

Finally, **Offline Sensor-Based Non-Supervised** methods involve analyzing sensor data retrospectively without predefined labels. This approach is beneficial for applications where data is analyzed in batch mode to identify patterns or clusters, such as in exploratory research or anomaly detection.

Table 1.1: Detailed HAR Methods Classification with Supervised and Non-Supervised Divided by Online and Offline

Category	**Supervised**		**Non-Supervised**	
	Online	**Offline**	**Online**	**Offline**
Video-Based	Real-time processing of video data	Pre-recorded video data analysis	Real-time video data without labels	Historical video data without labels
Sensor-Based	Real-time sensor data analysis	Collected sensor data analysis	Real-time sensor data without labels	Collected sensor data without labels

The 3D matrix effectively visualizes how these various HAR methods intersect across the axes

of data type, learning approach, and processing mode, highlighting their different applications and capabilities.

While the accessibility of sensors has coincided with the increased academic interest in the field, the development of novel Machine Learning (ML) techniques and algorithms, such as KANs, and the augmentation of data processing capabilities have driven innovation and research in HAR.

While Multi-Layer Perceptron (MLP) have been the go-to building block in Deep Learning (DL) architectures, they are heavy on resource requirements and often scale inefficiently due to a high number of parameters to update during training. KANs [38] have been proposed as an alternative to MLP in DL [34]. Rather than being based on the Universal Approximation Theorem (UAT), they are based on the generalization of the Kolmogorov-Arnold Theorem (KAT) [7, 31] to networks of arbitrary depth and width. The results have been promising in approximating mathematics functions, as well as in physics tasks, such as Anderson localization [38]. KAT states that if f is a multivariate continuous function on a bounded domain, then f can be written as a finite decomposition of continuous functions of a single variable and the binary operator of addition. Following that, KANs are fully connected networks that have learnable activation functions on edges and the sum operator on nodes (see Figure 2.1). Liu et. al. [38] refer to the favorable neural scaling laws to state that KANs can achieve more accurate results than MLP with fewer parameters.

The interpretability of these networks is compared to other methods. The assumption is tested by extracting symbolic representations of component functions, with activation functions approximated using B-splines and Chebyshev polynomials [53]. Still, a robust feasibility study has not, to the best of our knowledge, been done with other functions, such as the RBF [66], and other kernels that could behave well in local adjustments, such as the Laplacian [50], Cauchy [20], rational quadratic [12] and Matèrn [49] kernels. KANs and their variants have been applied to a number of different fields, including traffic forecasting [64], cryptocurrency trading [17], and medical image segmentation [36].

1.1 Objectives

The objectives of the present work are the following:

1. Evaluate the classification performance of KANs in benchmark open datasets, by comparing it to traditional ML methods and MLPs on published studies, namely using available accuracy, precision, recall and f1-score results.
2. Evaluate the performance of different kernels of KANs. We propose different activation functions to test further if local updates are what guarantee the better performance of KANs compared to MLP and to explore if there are functions that perform better than B-Splines.
3. Develop a framework for real-time human activity recognition, including data collection, annotation, pre-processing, model training and finally real-time classification on a smartphone, in a study with subjects with various ages and sexes.

4. Minimize the latency associated with real-time HAR by searching for the optimal window size threshold in terms of classification gains in relation to associated latency.
5. Estimate the battery drain time for HAR in a smartphone, to evaluate the viability of using KANs for real-time HAR under the present circumstances.

1.2 Contributions

The contributions of the present work are the following:

1. Adjustments to parameter initialization to adapt the KANs implementation to different kernel functions. For instance, to support the RBF kernel, there are function-specific parameters such as center and width that only need to be initialized when that kernel is used.
2. By comparing performance metrics of the different kernels across different datasets we are able to indicate which datasets can yield more reliable results in future studies.
3. We present the results of testing the framework in acquiring and annotating sensor data by having the subjects perform a sequence of activities and by collecting data for an activity on-demand.
4. The results of the application of different noise removal techniques are presented, as well as conclusions on optimal window size in order to maximize accuracy and minimize latency from activity start to successful prediction.
5. An Android application was developed to predict human activity in real-time for subjects of different sex and age using accelerometer and gyroscope data.
6. The datasets and code are freely available under GNU Public License v3.0 from a GitHub public repository [54].
7. An academic paper entitled "An empirical study on the application of KANs for classification" [51] was presented at the ICAAI 2024: The 8th International Conference on Advances in Artificial Intelligence, London, UK, October 17-19, 2024. The paper will be published in the ACM Conference Proceedings with ISBN: 979-8-4007-1801-4.

1.3 Document structure

The document is divided into five chapters. Chapter 1 presents an introduction containing a motivation for the investigation and a description of the context of the project, advancing how the solution is attainable. Chapter 2 of this document consists of the literature review and summary of related work to the problem under study. In Chapter 3, the proposed architecture is described and its components are analyzed. In Chapter 4, testing, tuning and validation with different datasets is performed, and the test results are presented. Chapter 5 includes the conclusions of the work that was performed, with perspectives on the future of HAR in general, as well as the next steps to extend the work done.

2 Research Background

The purpose of this chapter is to define the scope within which the research was conducted, placing it in a research context, and attempting to identify the techniques used and the characteristics that make them suitable for the purpose of this work. Additionally, it aims to explore the techniques origins to make clear what were the problems that motivated them.

2.1 Context and Research Landscape

Kolmogorov-Arnold Networks (KANs) represent a novel approach in Deep Learning (DL), with applications still in the early stages of exploration. Recent studies have demonstrated their versatility and effectiveness in various domains. KANs [38] have been used to approximate hand-crafted mathematical functions, as well as functions from Knot theory and the physics problem of Anderson localization. They have also been tested in time series forecasting for predicting the volume of traded cryptocurrency Bitcoin in a particular hour [17], as well as traffic forecasting using satellite data [64]. These tests have demonstrated strong accuracy in capturing temporal dependencies in sequential data. Moreover, they show that KANs can perform comparably to MLP in time-series forecasting with fewer parameters. This is particularly advantageous since current KANs implementations do not rely on matrix multiplication. Therefore, training cannot leverage the Graphics Processing Unit (GPU).

To mitigate the restricted parallel computing capabilities of GPUs that KANs possess, adaptations have been made performing 20x faster on approximating mathematical functions, with no results on technological problems [47].

Another study demonstrated that, applying approximation functions instead of B-splines – Chebyshev polynomials [53] – can result in good accuracy and low loss on the MNIST dataset and in a fractal 2-D function approximation. Other variants use orthogonal or semi-orthogonal functions such as continuous or discrete wavelet transform to address the performance issues of KANs based on B-splines [6].

2.2 Theoretical Background

Multi-Layer Perceptron (MLP) have traditionally been foundational components in DL architectures [34], leveraging the UAT [32] to approximate continuous functions accurately. The theorem states that a feed-forward neural network with at least one hidden layer (containing a finite number of neurons) can approximate any continuous function on a compact subset of $\mathcal{R}^n$ to any desired degree of accuracy. This is provided that the activation function used by the neurons is non-constant, bounded and continuous. However, their adoption is limited by substantial computational demands and training challenges.

Firstly, neural scaling laws [26] indicate that increasing computational resources exponentially, such as neurons and layers, is necessary for enhanced performance, posing practical constraints for large-scale applications. Secondly, Multi-Layer Perceptron (MLP) suffer from issues related to vanishing or exploding gradients, which occur during backpropagation. When gradients vanish, they become too small to update the weights, hindering the learning process effectively. Conversely, when gradients explode, they become too large, causing numerical instability and inefficient learning [5, 21]. To address these limitations, researchers have explored alternative architectures such as deep residual networks [19] and introduced normalization techniques [24]. These innovations aim to enhance training efficiency and stability in neural networks beyond the traditional MLP framework.

The Kolmogorov-Arnold Representation Theorem As mentioned before, KANs are based on the generalization of Kolmogorov-Arnold Theorem (KAT) [7, 31] to networks of arbitrary depth and width. The theorem can be defined in eq. (2.1).

$$f(x) = f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \Phi_{q,p}(x_p) \right) \tag{2.1}$$

where n indicates the number of input dimensions, and Φ represents the functions that are used to construct the overall function f through a combination of summation and composition.

This form applies to networks of depth-2 and width equal to $2n + 1$. However, this topology restricts the network's learning capacity to simple data representations. A general equation for networks of arbitrary depth and width may be obtained in eq. (2.2).

$$\begin{aligned} f(x) = \sum_{i_{L-1}=1}^{n_{L-1}} \Phi_{L-1,i_L,i_{L-1}} \left(\sum_{i_{L-2}=1}^{n_{L-2}} \Phi_{L-2,i_{L-1},i_{L-2}} \left(\cdots \left(\sum_{i_2=1}^{n_2} \Phi_{2,i_3,i_2} \right. \right. \right. \\ \left. \left. \left. \left(\sum_{i_1=1}^{n_1} \Phi_{1,i_2,i_1} \left(\sum_{i_0=1}^{n_0} \Phi_{0,i_1,i_0}(x_{i_0}) \right) \right) \right) \cdots \right) \right) \end{aligned} \tag{2.2}$$

Where n is the dimensionality of the input space, and L is the depth of the network. This assuming for any two input-output pairs (i_x, o_y), there is a function Φ such that $\Phi(i_x) \approx o_y$. The approximation functions are combined with the sum operator, and its results are used as inputs in the subsequent layer. This characteristic makes KANs well-suited for capturing the non-linear dependencies and compositional patterns in high-dimensional data.

In KANs, the architecture is fundamentally different from traditional neural networks, particularly in how it utilizes nodes and edges. Assuming the KAT, a problem may be approximated by the combination of univariate functions.

Nodes in KANs act as summation operators, aggregating incoming signals from multiple edges. Nodes simply perform addition without applying any non-linear activation functions as shown in Figure 2.1. This allows us to capture the compositional nature of some problems. Edges, on the other hand, are where the innovation of KANs lies. Each edge is associated with a learnable function, typically parametrized as a spline. This means that instead of having fixed weights as in traditional MLP, KANs allow each connection to adaptively learn a univariate function that can capture intricate relationships in the data.

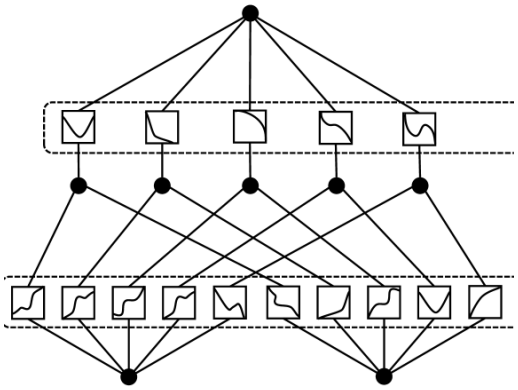


Figure 2.1: *Kolmogorov-Arnold Networks topology composed of learnable activation functions on edges and the sum operator on nodes (from [47])*

Theoretical guarantees for the expressivity of KANs KANs provide significant theoretical guarantees for function approximation. They can approximate functions with a finite grid size and achieving a residue rate independent of dimensionality, thus overcoming the curse of dimensionality [38]. This is achieved by utilizing splines to approximate one-dimensional functions. Neural scaling laws indicating that test loss decreases as the number of model parameters increases. Theoretical and empirical evidence suggests that KANs exhibit faster neural scaling laws than MLP, meaning the error rate decreases more rapidly as the network size grows [38].

Several theories predict the scaling component α. Kaplan et al. [26] suggests that α is derived from data fitting on an input manifold of intrinsic dimensionality d, leading to the relationship $\alpha = (k+1)/d$, where k is the order of piece-wise polynomials used in splines, which suffers from the curse of dimensionality. Michaud et al. [40] propose an α of $(k+1)/2$ by considering computational graphs with unary and binary operators. Poggio et al. [46] introduce the idea of compositional sparsity, giving $\alpha = m/2$, where m represents the order of smoothness of the function class, indicating the highest order of continuous derivatives that the functions possess. Liu et al. [38] assume the existence of smooth Kolmogorov-Arnold representations, decomposing high-dimensional functions into one-dimensional components, resulting in $\alpha = k+1$.

KANs can achieve arbitrary accuracy by refining the spline grid without retraining, unlike

MLP, which require extensive computational resources and retraining for improvements. Grid extension involves fitting a fine-grained spline to a coarse-grained one, optimizing the transition to minimize error. This process leverages both external and internal degrees of freedom within the computational graph, enhancing the model's ability to learn compositional structures and univariate functions simultaneously.

Simplification techniques, such as what is termed as sparsification via L1 regularization and entropy regularization, are introduced to optimize the KANs structure [38]. Pruning, based on input and output scores, helps in achieving a minimal sub-network by removing unimportant neurons. Visualization and symbolic representation of activation functions enhance interactivity and iterative training processes, making KANs more adaptable and efficient.

In contrast, symbolic regression, another method for approximating composite functions, lacks the intermediate outputs necessary for iterative debugging and cannot be universally applied to all functions. KANs, by using splines, offer a robust numerical approximation method that is both versatile and efficient for a wide range of functions.

Liu et el. [38] highlight the interpretability and accuracy properties of KANs, as well as the capacity of continuous learning, proposing them as an interactive companion to scientists when obtaining approximations of mathematical functions.

This work focuses on testing the potential of KANs in classification problems, comparing them against traditional ML methods present in the literature, as well as MLP.

Alternative Kernels to B-splines Notwithstanding the provision of significant theoretical guarantees for function approximation, the univariate functions making KANs up, may be nonsmooth or fractal, so they may need to be learnable in practice [37]. The studies performed on DL using the theorem attempted to build depth-2 and width $2n+1$ networks. These studies did not utilize contemporary training techniques, such as backpropagation, as evidenced in [13]. Because of topology limitations and the potential non-smooth character of the learnable function, the application of KAT to neural networks has been discarded in practice even though its theoretical foundations may be considered sound.

When the activation function $\Phi(x)$ is non-smooth and thus non-learnable, alternative kernels can be utilized. In such cases, kernels that favor local over global adjustments are optimal candidates. Some of the kernels used are formalized in equations (2.3) to (2.8).

Radial Basis Function Kernel:

$$K(x, x') = \exp\left(-\frac{|x - x'|^2}{2\sigma^2}\right) \tag{2.3}$$

where σ is the bandwidth parameter controlling the spread [66].

Chebyshev Polynomial Kernel:

$$K(x, x') = (x \cdot x' + c)^d \tag{2.4}$$

where c is a constant term, and d is the degree of the polynomial [53].

Cauchy Kernel:

$$K(x, x') = \frac{1}{1 + \frac{||x-x'||^2}{\sigma^2}} \tag{2.5}$$

where σ is the scaling parameter. This kernel [20] is especially robust in the presence of outliers, due to the inverse of the denominator term.

Laplacian Kernel:

$$K(x, x') = \exp\left(-\frac{||x - x'||}{\sigma}\right) \tag{2.6}$$

The kernel [50] employs the L1-norm and performs an exponential function on this value, dividing it by a parameter designated as σ.

Rational Quadratic Kernel:

$$K(x, x') = 1 - \frac{||x - x'||^2}{||x - x'||^2 + c} \tag{2.7}$$

The rational quadratic kernel [12] can be seen as a scale mixture of Gaussian kernels with different characteristic length scales where c is a constant parameter.

Matèrn Kernel:

$$K(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}||x - x'||}{\sigma}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}||x - x'||}{\sigma}\right) \tag{2.8}$$

The Matèrn kernel [49] generalizes the Gaussian kernel by allowing for different levels of smoothness, which is determined by the parameter ν. As ν increases, the Matèrn kernel becomes smoother, approaching the Gaussian kernel in the limit as $\nu \to \infty$. ν controls the smoothness of the function; σ is the *length-scale parameter*, determining how quickly the correlation between points x and x' decays with distance. A larger σ results in a slower decay, allowing distant points to remain correlated, while a smaller σ causes rapid decay, correlating only nearby points. This parameter controls the smoothness of the functions modeled by the Gaussian Process: larger σ values produce smoother functions, and smaller values allow for more rapid changes. $\Gamma(\nu)$ is the Gamma function, a generalization of the factorial function to non-integer values, defined as $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\,dt$ for $\text{Re}(z) > 0$. K_ν is the modified Bessel function of the second kind, defined as a special function that appears in solutions to certain differential equations with cylindrical symmetry.

2.3 Real-time Human Activity Recognition

Over the past decade, Human Activity Recognition (HAR) has experienced a surge in academic interest, as evidenced by the rising number of publications across academic databases, such as Google Scholar, MDPI, IEEE Xplore and ACM Digital Library. For example, showcased a steady rise in the number of HAR-related papers with an average of 24% increase year-over-year from 2012 to 2024, from 950 in 2012 to a peak of 10,500 in 2023 as depicted in Figure This growth trajectory reflects the expanding applications of HAR in diverse areas such as healthcare and caretaking, sports analytics, and security systems, where accurate activity detection is crucial.

Table 2.1: Publications related to HAR from 2014 to 2024 in different academic databases. The values for 2024 are provisional.

	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024
Google Scholar	1550	1990	2440	3010	3790	4770	5930	7360	9080	10500	7340
MDPI	14	30	35	60	80	113	207	242	289	298	133
IEEE Xplore	701	762	878	1122	878	1276	1315	1589	1786	2168	1185
ACM DL	7578	7669	8411	8845	9911	9854	11064	11828	11694	13321	16887

In this section, we go through the: (1) evolution in methods for real-time HAR over time, (2) the main challenges faced by researchers, (3) the methods used and how they compare to each other, (4) the main categories of methods, (5) the canonical stages when building a real-time HAR system, (6) and specific procedures such as windowing, sampling, sensor placement and feature extraction, and (7) we also present some test results available in the reviewed literature in order to reinforce comparison between methods.

In 2012, the first significant study on real-time HAR in mobile devices is published [33]. This study emerges following the incorporation of accelerometers and gyroscopes into smartphones. The authors identify key challenges such as limited processing power, energy constraints, and the unavailability of ML libraries for mobile devices. These hinder the deployment of context-aware applications on mobile devices.

To address these challenges, the authors propose a comprehensive mobile platform designed to facilitate real-time HAR. Central to this platform is a library intended for the evaluation of classification algorithms, accompanied by a mobile application that utilizes this library. At that time, there are no established ML libraries available for mobile platforms. The C4.5 algorithm and a decision tree algorithm are employed as a proof of concept.

The evaluation of the proposed system is thorough, considering metrics such as accuracy, response time, and energy consumption. Specifically, the system achieves an accuracy rate of 92.6%, a response time constituting 8% of the window length used for data processing, and a battery life of approximately 12.5 hours under continuous operation.

The tests focus on classifying three activities: running, walking, and sitting. The response time is defined as the interval comprising data collection, feature extraction, and activity prediction with windows equal to 5 or 12 sec. The approach compares favorably, with a 26.7% reduction in processing time relative to server-based data processing.

An alternative approach to sensor-based recognition is introduced by Chen et al. [10], using Depth Motion Map (DMM) by projecting each depth frame in a video sequence onto three orthogonal Cartesian planes. For a given video sequence, the absolute difference between two consecutive projected maps is accumulated, then the L2-regularized collaborative representation classifier with a distance-weighted Tikhonov matrix is used for activity recognition.

The method is evaluated using the Microsoft Research Action 3D (MSR-Action3D) dataset, which consists of 20 actions performed by 10 subjects, adding complexity to the recognition task. The experimental results indicated the proposed method achieves high recognition rates, particularly excelling in Test One and Test Two, with average recognition rates of 97.4% and 99.1%, respectively. For the more challenging Cross Subject Test, the method achieves a

90.5% recognition rate, which, while slightly lower than some existing methods, is noted for being computationally efficient as it avoids complex feature extraction steps like calculating Histogram of Oriented Gradients (HOG) descriptors.

The study also compares the performance of different classifiers, including the proposed L2-regularized collaborative representation classifier, Sparse Representation-based Classification (SRC), and Support Vector Machine (SVM). The L2 classifier performs comparably to SRC and generally better than SVM in terms of recognition accuracy. The method operates efficiently in real-time, processing each component of the pipeline—projected depth map generation, DMM feature extraction, Principal Component Analysis (PCA) dimensionality reduction, and action recognition—swiftly enough for real-time applications.

As with offline or batch HAR, human activity classification tasks can be categorized as video-based or sensor-based, as highlighted by the last two publications indicated. Chen et al. [10] had to implement their own mobile libraries to deal with ML tasks processing video or sensor data.

Another approach [1] combines video-based activity detection to reduce human monitoring in Closed-Circuit Television (CCTV). The method includes object detection, tracking, and classification, followed by activity recognition. Object detection is accomplished using background subtraction, identifying moving objects by comparing each video frame with a static background to detect changes or movement. Once a moving object is detected, the system proceeds to object tracking and classification, identifying and following individual objects over time and distinguishing between different people by analyzing specific features.

For activity detection, the system utilizes geometrical attributes of the tracked objects, specifically focusing on the centroid (the object's center point) and the aspect ratio (the relationship between the object's width and height). By monitoring changes in these attributes, the system can recognize simple activities, such as walking or standing, based on how the object's shape and position change over time. The recognized activities are walking, running, sitting, standing, and lying, with an average cycle time of 61ms.

Ignatov et al. [23] develop a user-independent system for real-time HAR using accelerometer data. The research employs Convolutional Neural Network (CNN) to classify human activities using accelerometer data from the WISDM dataset. The process includes data preprocessing, such as balancing the dataset, encoding the activity labels, and standardizing the data. The CNN model is trained on the processed data, focusing on using the time series data of a single second for local feature extraction.

The canonical four stages of a HAR application are further examined: data acquisition, feature extraction, classifier training, and classification. The placement and number of sensors are discussed. Researchers have tried placing sensors on the chest, in a hand or pocket, on the wrist, and on multiple body positions: left and right ankle, left and right hand, hip, and two sensors on the ankle and thigh. Gao et al. [15] studied the recognition difference between multi-sensor and single-sensor systems and concluded it is rather small. In [57], the authors demonstrate that a single sensor with an accelerometer and gyroscope is sufficient for good daily activity recognition. The main issue with using a smartphone is the variable orientation,

so the authors choose to place it on the ankle.

Other issues in data processing are sampling, windowing, and feature extraction. Regarding sampling, [56] shows that body movements are less than 10Hz, and the authors of [27] show that there is no significant recognition improvement in sampling rate above 20Hz. That said, smartphone sensors typically exhibit a default sampling rate of 50Hz.

Windowing involves grouping a continuous signal into segments based on a sampling rate, adhering to specific parameters. If the window is too short, it may not provide sufficient information, whereas a long window could encompass multiple activities. Windowing can be static or dynamic and typically spans one to two seconds, depending on the sampling rate. In the referred paper, overlap is not used in tests to expedite processing.

A stable and well-established feature group for HAR does not exist because feature efficiency varies with an individual's movement style [58]. Numerous authors [3, 30, 42, 63] have experimented with features in both the time and frequency domains, as well as using wavelet transformation. According to Suto et al. [58], the efficiency of features is contingent upon the specific movements of each person.

Some approaches use smartphones to collect data but perform offline data analysis and activity prediction. A study on real-time applications of HAR systems is presented in [57], focusing on health monitoring and assisted living. They correctly identify computer vision and wearable sensor networks as the main approaches followed in HAR. They review techniques for handling noisy data, including artificial neural networks, k-nearest-neighbors, and decision trees. Deep architectures are also considered, as they work better than shallow architectures without pre-processing.

According to Suto et al. [57], Artificial Neural Network (ANN) and k-Nearest Neighbors (kNN) are among the most efficient shallow methods for HAR applications. However, the use of CNN has also shown good results. While offline performance is typically better due to controlled conditions, online performance tends to be slightly lower because of greater data variance in real-life scenarios. This discrepancy is particularly notable in activities like sitting and jogging, where performance tends to drop. Another conclusion pointed out by the authors of [57] is that having a larger number of activities can increase the likelihood of recognition errors, making the system more prone to failure.

Taylor et al. [60] highlight comfort in non-invasive methods that use channel state information (CSI) from wireless signals instead of relying on wearable devices.

In the study conducted by Wan et al. [67], their contribution is placed in the field of mobile edge computing. They state that real-time HAR systems can aid in supervising individual health states through monitoring. They state that other methods fail to identify complicated and real-time human activities, and perform tests with different algorithms (CNN, Boltzmann, Long Short-Term Memory (LSTM), Bayesian Latent Space Model (BLSM), MLP, and SVM). The paper also provides indications of preprocessing steps such as normalization and denoising. They apply segmentation length of 25, sliding window mode with 50% coverage and obtained higher results with CNN in terms of precision, recall, F1-score, and accuracy in both the UCI-HAR and Pamap2 datasets.

Wang et al. [68] introduce online HAR via millimeter wave (mmWave) sensing, supported by a commodity mmWave radar chip. They attempt to reduce the impact of noise on recognition and long latency introduced by the radar technology. The authors reduce noise by clustering the point clouds based on the density of their distribution, effectively separating the relevant movement data from noise caused by multi-path effects in a complex environment. The high latency is mitigated with the use of a lightweight neural network called HARnet. They compare online and offline classification, obtaining slightly better results offline than online (93.25% vs. 91.52%). The tests were performed in a real-world situation, specifically a fitness center.

The approach followed by the authors of [62] involves the use of dedicated wearable sensors and the development of Bi-directional Gated Recurrent Unit with Inception Architecture (Bi-GRU-I), a custom ML model. The authors build a wearable hardware system equipped with 6-channel inertial sensing units. They use the Command Actions of Traffic Police (CATP) dataset, which includes various human activities. The model Bi-GRU-I combines a Bi-directional Gated Recurrent Unit (Bi-GRU) to capture temporal features from the inertial sensing signals and an Inception module to analyze spatial features among multi-channel signals. This dual approach allows the model to effectively learn both the time-dependent and spatial characteristics of the activities being recognized. This approach is simpler than the use of LSTM networks, with fewer parameters and a simpler structure, effectively reducing computation time in real-time recognition.

The authors of [16] also use DL in HAR, with a dataset collected from the personal smartphone sensors of 19 individuals, ensuring that the activities performed are reflective of daily life. They process the data to select observations at specific time intervals (e.g., every 200 ms). They explore various configurations for DL models, specifically CNN and LSTM networks, with the latter producing better results. They also apply ablation studies to determine the optimal hyperparameters, use cross-validation, and experiment with different window sizes (30, 60, and 90 sec). The authors find that using a window size of 90 sec results in the highest performance metrics, including accuracy and F1-score. Specifically, the peak performance achieved with the DS-CNN-LSTM model at this window size is an accuracy of 94.80% and an F1-score of 94.27%.

3 Solution Design

In this chapter the proposed architecture is described and its components are analyzed. Software Design is comprised of Architectural design and Detail design. The first assigns responsibility for aspects of behavior to modules of a software, while detail design specifies aspects of each system component, namely its data structures and algorithms [1].

3.1 Solution Architecture

The software architecture emerged directly from the required functionalities, resulting in a modular design that aligns with the four canonical phases of a real-time Human Activity Recognition (HAR) system. Each phase corresponds to a specific functionality, implemented as distinct modules in the proposed solution.

The four canonical phases of a real-time HAR system are:

- **Data Acquisition**: This phase entails collecting raw data from various sensors, including accelerometers, gyroscopes, and other wearable or environmental devices. The data typically consists of time-series readings capturing aspects of human motion or activity.
- **Data Pre-processing**: During this phase, raw sensor data is cleaned and processed for analysis. Steps may include noise reduction, normalization, segmentation, and feature extraction, transforming the raw data into a usable format.
- **Model Training**: During this phase, Machine Learning (ML) or Deep Learning (DL) models are trained on pre-processed data. It includes selecting algorithms, tuning hyper-parameters, and leveraging labeled data to train the model to recognize activities.
- **Classification**: In the final phase, the trained model is used to classify new, unseen data in real-time. This involves feeding the model with new sensor data and having it predict the current activity being performed based on what it has learned during the training phase.

[1] Michael Turner and Dr. Sharon A White - https://sce.uhcl.edu/whiteta/sdp/architectureDesignAndDetailedDesign.html Accessed 2024-08-16

To realize each phase, a functionality needs to be in place, and that functionality is, in this case, materialized by an isolated component. The modules of the architecture are thus the following:

- **Data Collector** - This module is responsible for collecting sensor data from gyroscope and accelerometer, and writing that data to file in the external storage of the mobile phone. The data needs to be annotated so that the performed activity is present in the context of the sensor data. It implements a sequence of activities, comprising of 30 sec of performing each activity and 5 sec pause in-between. The desired behavior consists of the software announcing through the speaker a 5 sec countdown within each activity segment, saying stop when the data collection for that activity stopped, and announcing the name of the activity when data collection for that activity started.
- **Data Pre-processor** - This module prepares data for training. It is executed offline, it consolidates the results of accelerometer and gyroscope, applying windowing and calculating derived measures for each of the axes of the two sensors.
- **Training module** - Use Kolmogorov-Arnold Networks (KANs) and the dataset obtained in the module Data Pre-processor to train the model offline and produce a mobile version of the trained module.
- **HAR Classifier** - Load the trained module, and obtain and transform data from the sensors to provide a prediction of the performed human activity.

The Data Collector and HAR Classifier modules are supported in two distinct Android applications using API Level 31. The Data Pre-processor module is supported in Python source code using `sci-kit learn`, `numpy` and `pandas`. The training module is supported in Python source code using `Pytorch`, `pandas` and `sci-kit learn`.

The architecture can be represented in the Unified Modeling Language (UML) format, as is present in Figure 3.1

3.2 Detail Design

This section elaborates on the algorithms and data structures supporting each of the four modules. A functional diagram of the system's modules is present in Figure 3.2.

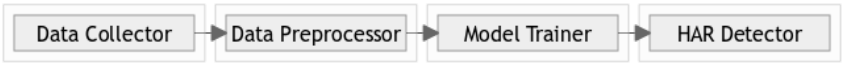


Figure 3.2: *Functional Diagram of the system's modules*

Data Collector As stated, this module is an Android application designed to collect sensor data (accelerometer and gyroscope) for HAR. The main functionality is implemented in the MainActivity class, which manages sensor interactions, data collection, and User Interface (UI) components.

The application begins by configuring permissions and initializing sensors. Within the onCreate method, the SensorManager is accessed from system services to retrieve both the accelerometer and gyroscope sensors. Additionally, a TextToSpeech instance is initialized to provide audible feedback for activity changes. The application checks if the required permissions (`WRITE_EXTERNAL_STORAGE` and `READ_EXTERNAL_STORAGE`) are granted; if not, it requests these permissions from the user.

The initial setup of the user interface is defined using `Jetpack Compose`. The `setContent` block sets up the main layout of the application, which includes a text field for entering the subject's name – which will be present in the file containing the collected data – displays for accelerometer and gyroscope data, and a button to start the human activity routing. This routing, as stated in the architecture section, is comprised of 6 activities – `LAYING, SITTING, STANDING, WALKING, WALKING_UPSTAIRS` and `WALKING_DOWNSTAIRS` – each lasting 30 sec with a 5-second interval. During the refinement process of the application, upon finding class imbalance in the datasets, a new functionality was added. A dropdown menu for selecting the current activity, and a button for starting and stopping data collection. This allowed a novel way of collecting data. It was no longer necessary to execute all activities, but rather we can collect data on-demand for a specific activity, and the data collection interval is not limited to 30 sec, but rather we can collect data for as much or as little time as we see fit.

The onSensorChanged method processes incoming sensor data. When data collection is active, it captures the current timestamp and sensor readings, formats them as CSV entries, and appends them to a list. The accelerometer and gyroscope data are handled separately to ensure correct formatting.

Data collection is managed by the startDataCollection and stopDataCollection methods. When data collection starts, the current activity label is set, the sensor data list is cleared, and the sensors are registered to receive updates at the fastest rate. The TextToSpeech instance announces the activity label. Stopping data collection involves unregistering the sensors and saving the collected data to a `CSV` file in the device's external storage.

The saveDataToCsv method handles the file operations for storing the collected sensor data. It creates a directory for storing the `CSV` files, formats the data, and writes it to a new file named with the subject's name, activity label, and timestamp.

An activity protocol can be started by calling the startActivityProtocol method. This method uses Kotlin coroutines to sequentially collect data for predefined activities, each for 30 sec with a 5-second pause in between.

The ActivitySelection composable displays a dropdown menu for selecting the current activity, and the ActivityButtons composable provides buttons to start/stop data collection and initiate the activity protocol. The SensorDataDisplay composable shows the current sensor readings and activity label.

Finally, the onDestroy method ensures that the TextToSpeech instance is properly shut down when the activity is destroyed.

In summary, this component manages the data acquisition phase of a real-time HAR system by collecting and storing sensor data in a structured format, ready for further processing and

analysis.

Sensor Placement The smartphone was positioned on a belt at the participant's right waist, secured with two folded rubber bands for stability, as detailed in Chapter 4. Further images of the setup can be found in 4.

Sampling frequency As mentioned in the literature review, there have been studies that state that sampling sensor data lower higher than 20Hz does not yield better results [27, 56], showing no recognition improvement in sampling rate above 20Hz. So the default sampling rate of the Android OS was chosen - 50Hz.

Data Pre-processor This module is focused on processing and preparing a HAR dataset for application of the ML algorithm - in this case KANs. That module utilizes sensor data, such as accelerometer and gyroscope readings, to train a model capable of classifying different activities.

Ffil sub-module Before producing segments, forward fill needs to be performed on the data. As the data collector separated gyroscope and accelerometer readings, it is necessary to perform forward fill on the raw files. This is done in the `ffill.py` module. That is designed to process `CSV` files located in the current directory by applying a custom forward fill method to address missing data. It begins by importing the necessary libraries: pandas for data manipulation and glob for finding files. The `glob.glob("*.csv")` function is used to retrieve a list of all `CSV` files in the directory. The module then iterates over each file in this list. For each `CSV` file, it reads the file into a pandas DataFrame using `pd.read_csv()`. After reading the file, it prints a message indicating the current file being processed.

The core of the module is its custom forward fill operation. It uses a loop to go through each row of the DataFrame, starting from the second row. For each row, it uses the combine_first method to fill any missing values (`NaN`) with values from the previous row. This effectively replaces missing data with the most recent non-missing values from earlier rows. Once the forward fill is applied, the DataFrame is saved to a new `CSV` file with a name that prefixes `'ffill_'` to the original filename. Finally, the module prints a confirmation message for each file processed, indicating that the file has been successfully processed and saved under the new filename.

Combine sub-module A second script called `combine.py` is evoked. This module is designed to aggregate multiple `CSV` files with different activities and potentially from different subjects located in the current directory into a single consolidated DataFrame. It begins by importing the necessary libraries: pandas for data manipulation and glob for file searching. The module uses `glob.glob("*.csv")` to gather a list of all `CSV` files in the directory. It then iterates over each file, reading its contents into a pandas DataFrame and appending each DataFrame to a list. During this process, it prints a message for each file indicating that it has been successfully loaded.

After loading all the individual `CSV` files into DataFrames, the module combines them into a single DataFrame using `pd.concat()`, with the `ignore_index=True` parameter to ensure that the index is reset and continuous. It then defines an output filename, `'combined_output.csv'`, and saves the combined DataFrame to this new `CSV` file using `to_csv()`, making sure not

to include row indices in the saved file. Finally, the module prints a confirmation message indicating that the combined file has been successfully saved with the specified filename.

Simple preprocess sub-module The present module performs windowing of particular segments performing simple statistical operations on each particular window. In a nutshell, for each of the 3 axes of accelerometer and gyroscope and for the configured number of samples, this sub-module performs mean, standard deviation, minimum and maximum. So a dataset with Timestamp, `(x,y,z)` axes of accelerometer, `(x,y,z)` axes of gyroscope and label is the input, and the output is a dataset with mean, standard deviation, minimum and maximum for each of the 6 measures from sensors, and the label.

The segmentation is performed process begins by loading the dataset from a `CSV` file into a pandas DataFrame. The dataset includes timestamps, sensor readings, and activity labels. The timestamps are converted to a datetime format to facilitate time-series treatment. Any missing values in the dataset are identified and removed to ensure the quality of the data. This straightforward approach ensures that only complete data is used for further processing.

Next, the sensor readings are standardized using the `StandardScaler` from the `sklearn` library. Standardizing the data transforms the sensor readings to have a mean of 0 and a standard deviation of 1, which helps in normalizing the dataset and improving the performance of ML algorithms.

To prepare the data for ML, the continuous time-series data is segmented into fixed-size windows. A window size of 100 is used, meaning that each segment contains 100 consecutive readings for each sensor. For each segment, the mode (most frequent value) of the activity labels within that window is assigned as the label for that segment. This segmentation process converts the continuous data into discrete chunks that are more manageable for feature extraction and model training.

Features are then extracted from each segment. For each sensor reading within a segment, statistical features are calculated. These include mean, standard deviation, minimum, and maximum. As each of the two sensors has 3 axes, the dataset is comprised of 24 features and 1 label. These features provide a summary of the data within each segment, capturing important patterns and variations.

The features and labels are converted into `numpy` arrays, and the activity labels are encoded into numerical format using `LabelEncoder`. This encoding is necessary for the ML algorithm to process the labels effectively.

The dataset is split into training and testing sets using an 80-20 split. This separation allows for training the model on one portion of the data and testing offline its performance on another, ensuring evaluation of the model.

The training and testing data are saved to separate `CSV` files for future use. These files contain the extracted features and corresponding labels, making them ready for model training and evaluation.

Noise removal Three noise removal approaches were tested: no noise removal, `butterworth` filtering, and the Savitzky-Galoy filter. The Butterworth filter performed worse than the others, while the Savitzky-Galoy filter showed superior results for smaller window sizes. The system

behavior with different noise removal approaches is recorded in chapter 4. The overall better results were obtained with higher window sizes, but given the requirement that the system performs in real-time, lower latency is preferable. In this case, a window size of 100 with sampling frequency of 50Hz, and therefore latency equal to 2 sec was the option that had the best cost-benefit performance.

A comment about the statistical measures employed The authors of [3] produced a dataset with 562 features with windows of 2.56 sec and 50% overlap between them. We tried to reproduce this setup with our collected data, and, apart from a very slow preprocessing step, it resulted in low accuracy values (approx. 20%). For that reason we tried to build up from a very simple scenario, in which we calculate mean, standard deviation, minimum and maximum and were surprised at the robustness of that simple approach. As we will see in Chapter 4, this approach showcases reliable results in a variety of settings, and with the different kernels. Moreover it has the advantage of being simple to implement and faster to apply.

Training module Training was performed over 50 epochs with a batch size of 64 and a learning rate of 0.01. The data was converted to PyTorch tensors and batched using `DataLoaders`.

Seven KANs models are evaluated: (1) KAN (with B-Splines), (2) RBFKAN, (3) LaplacianKAN, (4) ChebyshevKAN, (5) CauchyKAN, (6) RationalQuadraticKAN, and (7) MaternKAN. Each model was trained using either Stochastic Gradient Descent (SGD) (for LaplacianKAN) or Adam optimizer and evaluated on training and test sets. The CrossEntropyLoss criterion was used.

During each epoch, training and validation losses, accuracies, precision, recall, and F1-score are recorded. Confusion matrices are also generated for further analysis.

For offline evaluation of OpenML, Kaggle and UCI datasets, different datasets were evaluated as can be followed in Algorithm 1.

Algoritmo 1 KANs Model Training and Evaluation

```
Load datasets
for each dataset in dataset_names do
    Preprocess data
    for each KAN model do
        Initialize model
        Initialize optimizer
        Initialize criterion
        for each epoch in num_epochs do
            Train model
            Evaluate model
            Record training metrics
            Record validation metrics
        end for
    end for
end for
```

Finally, the `Pytorch` module is obtained by transforming the module to `Torchscript`, a mobile-compatible format. The application of the activation functions in KANs, is done by converting the input to a tensor. As all activation functions are a linear combination of a fixed

set of basis functions, we can activate the input with different basis functions and then apply the sum. This transforms the forward pass into a simple matrix multiplication. This change has been implemented in [8] but not published nor tested in published form.

A change that can be done to bring additional efficiency to the implementation is performing L1 regularization of the weights and discarding the L1 regularization on the input samples, which negatively impacts performance. This is included in the original paper, is a regular procedure in neural networks, and the tests done achieve satisfactory results.

Replacing B-splines with other kernels effectively involves specific coefficient initialization, so when implementing a new kernel, the reset parameters function needs adjustment to the kernel coefficient initialization and shapes. Thus, our implementation, containing these two changes relative to the previous publications, is stored in a GitHub public repository [54] alongside the datasets and test code to reproduce results.

HAR Classifier This module is composed by an Android application that uses the preprocessed sensor data to predict human activities using ML. It is structured around an Activity class, `MainActivity`, that handles both user interaction and sensor management.

The `MainActivity` class extends `ComponentActivity` and implements `SensorEventListener` to receive updates from accelerometer and gyroscope sensors. It sets up the user interface using Jetpack Compose, which displays data from these sensors and the predicted activity.

On creation of the activity, the application requests permission to access body sensors and initializes the sensor manager if permission is granted. It also sets up text-to-speech for vocalizing the predicted activity and loads a PyTorch model for activity recognition.

Sensor data is collected and processed in real-time. The data from accelerometer and gyroscope sensors is combined and accumulated into a window. Once the window is filled, the data is scaled using a `StandardScaler`, features are extracted, and the ML model predicts the activity. The UI is then updated with the predicted activity, which is also spoken out loud.

The `StandardScaler` class is used for normalizing sensor data based on precomputed means and standard deviations. The fit and transform methods are used to prepare the data for the model. Additionally, a preview composable function is provided to visualize the sensor data and predicted activity. Finally, resources such as text-to-speech are properly released when the activity is destroyed. The `MainActivity` manages sensor updates, data processing, and UI updates efficiently, integrating various components to deliver real-time activity recognition.

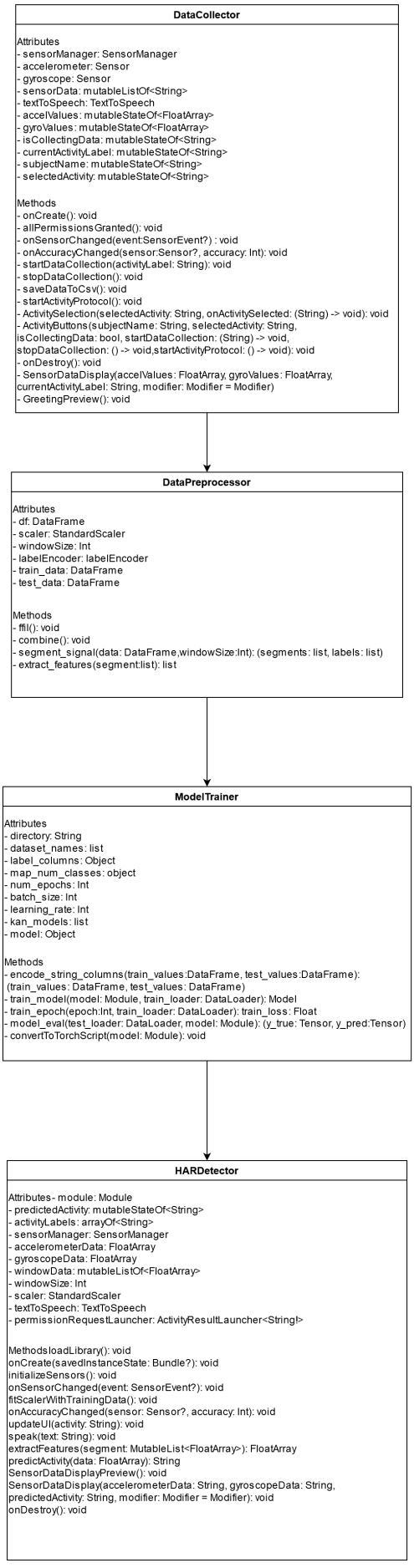


Figure 3.1: *UML Diagram of the architectural design of the system*

4 Test Results

4.1 Benchmarks datasets

For benchmarking, we utilized four open-source datasets from OpenML [65] and Kaggle [25], along with three datasets from the UCI Repository [29] focused on HAR. This diverse selection of datasets from various sources allows for a thorough evaluation across different data types and complexities, with a particular emphasis on human activity data to assess the model's performance in real-world scenarios. All chosen datasets are intended for classification tasks, as summarized in Table 4.1, which details the number of features and instances. Although some datasets are also used for regression tasks, we have not included these details as they are not relevant in this work.

The datasets MONKS2 are sourced from OpenML, while MNIST is obtained from Kaggle. CREDIT-G and KC2 are present in both OpenML and Kaggle. The datasets AIDS-175, UCI-HAR, and HAR-in-AAL are from the UCI Repository.

Table 4.1: Benchmark datasets

Dataset	# Features	# Instances	Associated Tasks
MONKS2 [2]	6	601	classification
CREDIT-G [4, 22]	21	1K	classification
KC2 [39]	22	522	classification and clustering
MNIST [35]	784	70K	classification, clustering and regression
AIDS-175 [18]	23	2,139	classification and regression
UCI-HAR [3]	561	10,299	classification and clustering
HAR-in-AAL [11]	561	5,744	classification

MONKS2 The MONK's problems, first introduced by Andrews et al. 1995 [2], represent a significant milestone in ML. These problems formed the foundation for the first international comparison of learning algorithms, setting a benchmark for evaluating various ML techniques. This dataset contains six features and two classes with 601 instances.

CREDIT-G The German Credit dataset, analyzed in depth by Hoffman, 1994 and Barbiero et al. 2020 [4, 22], is a widely used benchmark in credit risk assessment, with the purpose of classifying individuals into good or bad credit risks. The dataset contains 21 features and two classes (good or bad credit risk) with 1000 instances. It includes features relating to financial

status, credit characteristics, employment and income, demographics, residential information, credit obligations, support and guarantees, and other attributes such as the purpose of the credit and whether the client is a foreign worker or not. This dataset provides a multifaceted view of credit applicants, making it valuable for developing and testing credit scoring models. Its diverse set of features allows for the exploration of various factors influencing credit risk, making it a robust tool for both academic research and practical applications in the financial sector.

KC2 The KC2 dataset, analyzed by Menzies et al., 2004 [39], is a valuable resource in software engineering for defect prediction provided by NASA's Metrics Data Program. This dataset exemplifies real-world software engineering challenges, offering insights into the relationship between code metrics and defect occurrence. Its use of established complexity metrics (McCabe and Halstead) makes it particularly relevant for studying how code structure and complexity correlate with defect proneness. The KC2 dataset's moderate size and focused scope make it suitable for both educational purposes in software engineering courses and for benchmarking defect prediction algorithms in research settings. It includes 22 features and two classes, indicating defective or non-defective, with 522 instances.

MNIST The MNIST dataset, as referenced in Lecun et al., 1998 [35], is a large collection of handwritten digits widely used for training and testing in ML. It consists of 70K gray scale images of handwritten digits from 0 to 9, split into a training set of 60K images and a test set of 10K images. Each image is 28×28 pixels in size, resulting in 784 pixels per image, with pixel values ranging from 0 to 255, where 0 represents white, 255 represents black and intermediate values represent varying shades of grey. Each image is labeled with a corresponding digit from 0 to 9. One of the primary applications of MNIST is in the development of systems capable of recognizing handwritten digits. Beyond digit recognition, the principles learned from working with MNIST can be applied to broader image classification problems.

AIDS Clinical Trials Group Study 175 (AIDS-175) The dataset, as discussed by Hammer et al., 1996 [18], was created to examine the performance of two different types of AIDS treatments, by predicting whether each patient died within a certain time-frame. It includes demographic details, medical history, treatment history, and lab results. The dataset is structured with a total of 2,139 instances. Each instance in the dataset has 23 features and is annotated with a class label indicating the observed activity.

HAR using Smartphones The HAR dataset, detailed by Anguita et al. 2013 [3], was created using data from 30 volunteers whose ages ranged from 19 to 48 years. Each participant was asked to perform a series of six distinct activities: standing, sitting, lying down, walking, descending stairs, and ascending stairs. During these activities, the participants carried a Samsung Galaxy S II smartphone securely mounted around their waist. The data collection process involved manually annotating the recorded movements through video footage. To ensure clear activity transitions and consistent experimental conditions, a 5-sec pause was implemented between each activity. This interval helped to establish distinct boundaries between activities and allowed each new activity to start from a neutral state, which was essential for accurate data labeling and reproducibility. The protocol was executed twice by each participant. In the first trial, the smartphone was fixed on the left side of the participant's belt, which provided a

controlled and consistent data collection environment. This setup was intended to minimize variability caused by different sensor positions and to control for the effects of sensor placement on activity recognition. In the second trial, participants were allowed to place the smartphone as they preferred, reflecting more realistic usage scenarios. This variation aimed to explore how changes in sensor placement might influence the accuracy of activity recognition algorithms. By analyzing data from both standardized and variable placements, the study aimed to ensure that the developed models are robust and adaptable to different real-world conditions.

Smartphone Dataset for HAR in Ambient Assisted Living (AAL) (HAR-in-AAL) This dataset, created by [11], an upgrade of the dataset present in [3], seeks to enhance the precision of HAR algorithms within the realm of Ambient Assisted Living (AAL). It aggregates time-series data gathered from the embedded accelerometer and gyroscope sensors of smartphones carried by 30 individuals, spanning ages 22 to 79. Each participant executed six activities (standing, sitting, lying down, walking, ascending stairs, and descending stairs) for a duration of 60 sec, with data captured at a consistent frequency of 50Hz. The dataset comprises 5,744 instances, all free from missing values, and incorporates tri-axial acceleration and angular velocity data, alongside a 561-dimensional feature vector derived from these measurements.

4.2 Offline Human-Activity Recognition

The same tests were performed with all seven datasets: first, a classification task with KANs is performed with seven different kernels across 50 epochs. The kernels used were the B-spline implementation, the Radial Basis Functions (RBF) basis function, Laplacian, Chebyshev polynomials, Cauchy, Rational Quadratic and Matèrn kernels. Table 4.4 summarizes the results with the best outcomes highlighted, compared against an Multi-Layer Perceptron (MLP) and a traditional Machine Learning (ML) algorithm, recording precision, recall, and F1-score metrics for comprehensive analysis. Additionally, Figures 4.1 and 4.2 depicts the training accuracy and loss per epoch in case of HAR with smartphones and HAR-in-AAL datasets, with further result details to be discussed later. The results obtained by other authors can be found in Table 4.2.

Dataset	Algorithm	Metric	Value
MONKS2 [43]	XGBoost	Precision	0.98
CREDIT-G [41]	RF	Recall	0.8534
CREDIT-G [41]	SVM	Recall	0.4793
CREDIT-G [41]	XGBoost	Recall	0.6182
CREDIT-G [41]	MLP	Recall	0.7528
KC2 [61]	SVM	Accuracy	0.8272
KC2 [52]	SVM	Accuracy	0.8625
MNIST [53]	ChebyshevKAN	Accuracy	0.9646
MNIST [59]	SNN	Accuracy	0.9791
HAR [3]	MC-SVM	Accuracy	0.96
HAR [14]	Markov Chains	Accuracy	0.9189
HAR-in-AAL [11]	ANN	Accuracy	0.914
HAR-in-AAL [11]	SVM	Accuracy	0.976

Table 4.2: Performance of other algorithms on selected benchmark datasets.

Dataset	Algorithm	Metric	Value
MONKS2 [43]	XGBoost	Precision	**0.98**
MONKS2	KAN-Matern	Precision	0.8074
CREDIT-G [41]	RF	Recall	**0.8534**
CREDIT-G [41]	SVM	Recall	0.4793
CREDIT-G [41]	XGBoost	Recall	0.6182
CREDIT-G [41]	MLP	Recall	0.7528
CREDIT-G	KAN-Matern	Recall	0.6548
KC2 [61]	SVM	Accuracy	0.8272
KC2 [52]	SVM	Accuracy	0.8625
KC2	KAN-Chebyshev	Accuracy	**0.9740**
MNIST [53]	KAN-Chebyshev	Accuracy	0.9646
MNIST [59]	SNN	Accuracy	**0.9791**
HAR [3]	MC-SVM	Accuracy	**0.96**
HAR [14]	Markov Chains	Accuracy	0.9189
HAR	KAN-RBF	Accuracy	0.9292
HAR-in-AAL [11]	ANN	Accuracy	0.914
HAR-in-AAL [11]	SVM	Accuracy	**0.976**
HAR-in-AAL	KAN-Chebyshev	Accuracy	0.8711

Table 4.3: Comparison of various algorithms on selected benchmark datasets. The highest value for each dataset is highlighted in bold.

MONKS2 The B-Spline implementation, RBF, and Chebyshev polynomials show lower train loss results and faster convergence, while Chebyshev polynomials, rational quadratic, and Matèrn kernels exhibit lower test loss; furthermore, RBF and Matèrn kernels outperform Laplacian and Cauchy kernels in accuracy, with Matèrn, rational quadratic, and RBF kernels achieving better precision, recall, and f1-score results. However, these precision results are slightly less favorable compared to those obtained with XGBoost, which range from 0.95 to 0.98 [43].

CREDIT-G Tests with the Credit-g dataset demonstrated a lot of variability, indicating that Chebyshev and Cauchy's kernels are more stable in this scenario. This may be attributed to the resistance to outliers that these kernels exhibit.

The RBF and the Chebyshev polynomials yield better precision metrics, but the quadratic function returns better recall, with B-Spline following. The Matèrn algorithm performs reasonably well on precision, recall and F1-score.

Munkhdalai et al. [41] records various methods, including Logistic Regression, Support Vector Machine (SVM), and XGBoost, highlighting Random Forest (RF) with recall equal to 0.8534, which is higher than our implementation. However, our method compares favorably in recall to SVM (0.4793 vs 0.6548) and XGBoost (0.6192 vs 0.6548). An MLP with the sigmoid activation function acchieves 0.7528 for recall.

KC2 The Rational Quadratic Function and RBF show less stability compared to Chebyshev, Cauchy, and B-Spline, which offer lower loss, while the Matèrn kernel exhibits high accuracy; furthermore, excluding Laplacian, rational quadratic, and RBF, the other four kernels perform well on the accuracy, comparing favorably with SVM on accuracy (0.8272 vs. 0.9740) [61], precision, and matching in recall and F1-score [48], ultimately resulting in better accuracy (0.8625 vs. 0.9740) [52] compared to some previously reported results.

MNIST Chebyshev polynomials of degree 5 achieved an accuracy of 0.9740 with significant train loss decrease and half the number of epochs, outperforming the KANs implementation in previous studies which yielded 0.9646 accuracy [53]; moreover, our test accuracy (0.9737) is comparable to other recent results (0.9791) [59], while using fewer training parameters than Deep Neural Network (DNN5) implementations (51,6640 vs 57,5051) [45]. RBF achieves better results than other kernels in this case. Most studies focus on accuracy and do not provide precision, recall and F1-score metrics.

AIDS-175 B-Spline, Chebyshev, Cauchy, and Matèrn kernels demonstrated outstanding performance in train and test loss, with Chebyshev achieving losses close to 0.2 for both; all kernels except Laplacian performed above 0.8 in accuracy, though RBF showed less stability and more variability on this dataset; ultimately, every kernel excluding Laplacian achieved good results, with Rational Quadratic and Chebyshev emerging as the top two performers.

Concerning train and test loss, B-Spline, Chebyshev, Cauchy, and Matèrn kernels performed better, with Chebyshev achieving train and test loss close to 0.2. Apart from the Laplacian kernel, all other kernels performed above 0.8 in terms of accuracy. RBF yields fewer stable results on this dataset with more variability than other kernels apart from Laplacian. Every kernel apart from Laplacian achieved good results, with Rational Quadratic and Chebyshev being the two best.

Human Activity Recognition (HAR) using Smartphones Dataset All kernels except the Laplacian operator exhibited loss values below 0.1, with RBF and Chebyshev performing better than others; Matèrn and Cauchy's kernels showed pronounced spikes during testing, while Chebyshev, KANs, and RBF demonstrated the most stability; as illustrated in Figure 4.1, all kernels achieved accuracies well exceeding 0.9, with test accuracies consistently ranging between 0.80 and 0.86 across all kernels.

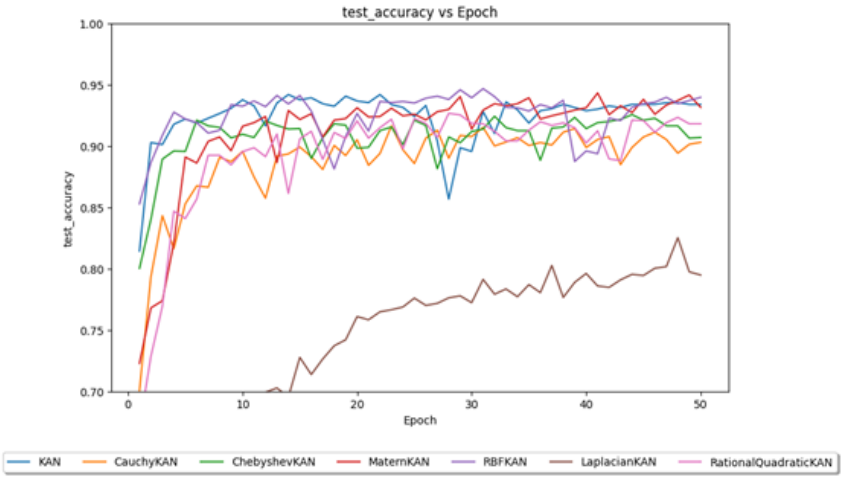


Figure 4.1: *Train accuracy per epoch with the Smartphone HAR*

Compared with an MLP method used in [55], train accuracy in most kernels is higher with a third of the epochs, with faster convergence. We also achieve comparable results to MC-SVM [3] with Chebyshev KANs (0.96 vs. 0.9601) and performs better than an approach with Markov chains used in [14] (0.9189).

Regarding precision, recall and F1-score, all kernels performed well, with the Matèrn kernel and Spline being the best. This can be attributed to the convergence of internal and external degrees of freedom when performing classification with KANs. While the learnable activation functions on edges allow for capturing each component of data distribution separately, the sum operator on nodes allows for these components to be merged, capturing the compositional patterns in the dataset.

Smartphone Dataset for HAR in Ambient Assisted Living The RBF and Chebyshev kernels perform exceptionally well regarding train loss, but RBF performs more poorly in test loss, with every kernel except Laplacian performing increasingly worse with the advancement in epochs as depicted in Figure 4.2. As far as accuracy is concerned, all kernels performed in the interval 0.80 to 0.86 in test accuracy. The Chebyshev kernel performed best in terms of precision, recall, and F1-score, with B-Spline, Rational Quadratic, and Matèrn kernels following closely.

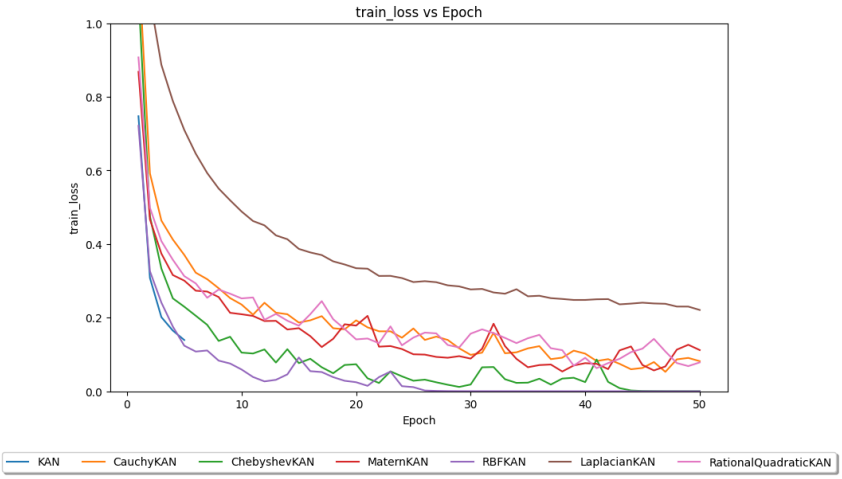


Figure 4.2: *Train loss per epoch with the HAR ambient sensor for assisted living*

Overall, the Laplacian kernel performs poorer than other kernels, with Chebyshev and Matèrn obtaining the best results overall. The results in terms of train and test losses of this implementation are better than the literature [53].

4.3 Real-time Human Activity Recognition

To test real-time HAR, 6 subjects participated in the research, 3 of them masculine and 3 feminine. The subjects' ages range from 35 to 58 years. After acquiring data for each subject, an offline test was conducted to analyses potential differences in performance related to age and sex. The datasets from all subjects were then combined and a model was trained with the 7 different kernels in order to evaluate the best kernel to use in real-time HAR using KANs. The subjects' and dataset's characteristics are listed in Table 4.5.

Sensor placement The smartphone was attached to a belt using two rubber bands. Each rubber band was folded before being inserted in the belt for stability. The rubber bands were placed at the extremes of the smartphone. The belt was placed at the waist of the subjects in

Table 4.4: Kernel Performance Metrics for Different Datasets

Dataset	Metric	Cauchy	Chebyshev	Spline	Laplacian	Matérn	Rational Quadratic	RBF
MONKS2	Precision	0.7177	0.7633	0.7624	0.4420	**0.8074**	0.7935	0.7731
	Recall	0.7158	0.7637	0.7601	0.4433	**0.8048**	0.7886	0.7737
	F1-score	0.7124	0.7624	0.7567	0.4413	**0.8055**	0.7841	0.7732
CREDIT-G	Precision	0.6429	0.7491	0.6187	0.3990	0.5311	0.6313	**0.7530**
	Recall	0.6026	0.5900	0.6404	0.4876	**0.6548**	0.6499	0.5533
	F1-score	0.6081	0.5832	0.6041	0.2456	**0.6217**	0.5856	0.5209
KC2	Precision	0.9578	**0.9736**	0.9565	0.3136	0.9655	0.8662	0.9000
	Recall	0.9689	0.9636	**0.9722**	0.4907	0.9365	0.8935	0.7857
	F1-score	0.9627	**0.9682**	0.9629	0.3826	0.9482	0.8632	0.8080
MNIST	Precision	0.9694	0.9725	0.9734	0.9023	0.9715	0.9725	**0.9737**
	Recall	0.9693	0.9723	0.9733	0.9025	0.9714	0.9724	**0.9736**
	F1-score	0.9693	0.9724	0.9734	0.9022	0.9714	0.9724	**0.9737**
AIDS-175	Precision	0.8500	**0.8835**	0.8586	0.5068	0.8533	**0.9011**	0.8532
	Recall	0.7633	**0.8169**	0.7195	0.5078	0.7667	0.6903	0.7442
	F1-score	0.7908	**0.8431**	0.7558	0.5055	0.7668	0.7293	0.7778
HAR	Precision	0.9393	0.9613	0.9630	0.9450	**0.9634**	0.9604	0.9627
	Recall	0.9322	0.9579	0.9605	0.9394	**0.9616**	0.9577	0.9602
	F1-score	0.9339	0.9587	0.9612	0.9410	**0.9621**	0.9582	0.9609
HAR-in-AAL	Precision	0.8404	**0.8676**	0.8660	0.8543	0.8601	0.8617	0.8622
	Recall	0.8426	**0.8670**	0.8628	0.8524	0.8574	0.8577	0.8605
	F1-score	0.8385	**0.8671**	0.8290	0.8528	0.8560	0.8587	0.8604

Table 4.5: Subject Datasets Characteristics

Subject	Sex	Age	Raw Set	Training Set	Test Set
1	M	58	143,836	1,151	289
2	F	57	122,039	977	245
3	F	35	166,869	1,335	335
4	M	35	152,333	1,219	306
5	F	36	228,069	1,825	457
6	M	42	186,310	1,490	374

such a way that the smartphone is located in the right side of the waist, next to the right leg, as can be seen in Figures 4.3 and **??**.

Age and sex Breakdown A dataset for each subject was trained separately to evaluate differences in performance based on age and sex. Figure 4.6 presents the best results for each subject based on test accuracy.

Table 4.6: Model Performance Metrics for each subject dataset

Subject	Kernel	Train Loss	Test Loss	Test Accuracy	Precision	Recall	F1-Score
1	Matèrn	0.0385	0.4404	0.8906	0.8913	0.8899	0.8903
2	RBF	0.0002	0.3428	0.9378	0.9076	0.8723	0.8857
3	Matèrn	0.0252	0.2356	0.9427	0.9152	0.9180	0.9163
4	RBF	0.0001	0.2309	0.9627	0.9681	0.9676	0.9678
5	Chebyshev	0.00006	0.0649	0.9902	0.9786	0.9775	0.9776
6	Chebyshev	0.0007	0.1130	0.9786	0.9610	0.9493	0.9544

The model was found to perform slightly worse in subjects with more than 50 years, however in order to generalize this conclusion, more subjects in that age range should be considered.

Analysis of confusion matrices In all subjects, the model is very stable for actions `LAYING`, `SITTING`, and `STANDING`. For subject 1, there is a slight confusion between `WALKING` and

Figure 4.3: *Sensor placement in Subject #4 - Detail*

WALKING_DOWNSTAIRS, as well as between WALKING_UPSTAIRS and WALKING, and between WALKING_DOWNSTAIRS and WALKING. Subject 2 shows slight confusion between WALKING and WALKING_DOWNSTAIRS, and between WALKING_UPSTAIRS and WALKING. Subject 3 experiences slight confusion between WALKING_UPSTAIRS and WALKING_DOWNSTAIRS, and between WALKING_DOWNSTAIRS and WALKING_UPSTAIRS. Subject 4 demonstrates confusion between WALKING_UPSTAIRS and WALKING_DOWNSTAIRS, and vice versa. For subject 5, confusion seems to occur only between WALKING_UPSTAIRS and WALKING_DOWNSTAIRS. Finally, subject 6 displays confusion between WALKING_UPSTAIRS and WALKING.

Overall the model mislabels data when the subject is walking up or downstairs, mislabeling it as walking. This is not likely due to subject's particular gait, but probably because generally stair flights are not continuous and have intervals – segments between flights of 8 to fifteen stairs where there are no stairs but flat ground as seen in Figures 4.4 – 4.6, In those intervals, the subjects are effectively walking.

No subjects below 35 years and above 58 years were studied.

Comparison with XGBoost Tests were also performed with the different datasets using XGBoost. Table 4.8 document the results obtained. For the full dataset, XGBoost provides slightly better accuracy than KANs (0.9364 vs. 0.9193). For subjects 1, 2 and 3 the results are similar. For subjects 4, 5 and 6, KANs outperform XGBoost. In subject 4, this is mainly due to the performance in the WALKING activity. In subject 5, it is due to the performance in WALKING_DOWNSTAIRS. In subject 6, the difference is due to better performance in the activity WALKING_UPSTAIRS. We observe that for activities SITTING and STANDING, XGBoost may be causing overfitting.

Evaluating models for real-time HAR The subjects datasets were combined, and a dataset with all subjects' data was obtained. The resulting dataset has 999,449 instances. Windowing was performed, and a training-test split was performed. The train dataset has 7,995 instances

Table 4.7: Confusion Matrix

(a) Subject #1

	1	2	3	4	5	6
1	44	0	0	0	0	0
2	0	51	0	0	1	0
3	0	1	46	0	0	0
4	0	0	0	41	2	7
5	0	0	0	4	41	3
6	0	0	1	12	4	30

(b) Subject #2

	1	2	3	4	5	6
1	48	0	0	0	0	0
2	0	54	0	0	0	0
3	0	0	45	1	0	0
4	0	0	1	38	0	6
5	0	0	0	1	30	2
6	0	0	0	8	1	36

(c) Subject #3

	1	2	3	4	5	6
1	41	0	0	0	0	0
2	0	42	0	1	0	0
3	0	0	49	0	0	0
4	0	0	0	117	0	3
5	0	0	0	0	33	7
6	0	0	0	0	11	30

(d) Subject #4

	1	2	3	4	5	6
1	42	0	0	0	0	0
2	0	48	0	1	0	0
3	0	0	39	0	0	0
4	0	0	0	51	1	2
5	0	0	0	5	58	0
6	0	0	0	7	1	50

(e) Subject #5

	1	2	3	4	5	6
1	228	0	0	2	0	0
2	0	18	0	1	0	0
3	0	0	116	0	1	0
4	0	0	0	26	1	1
5	0	0	0	0	31	3
6	0	0	0	0	1	27

(f) Subject #6

	1	2	3	4	5	6
1	58	0	0	0	0	0
2	0	170	0	0	0	0
3	0	0	55	0	0	0
4	0	0	0	87	4	0
5	0	0	0	5	19	1
6	0	0	0	2	0	30

and the test dataset has 1,999 instances. Table 4.9 summarizes the number of instances in train and test datasets per activity.

The models were evaluated based on quantitative criteria, namely accuracy, train and test loss, F1-score, precision, recall, battery usage and response time, as well as qualitative criteria namely real-time predictive capacity.

Performance metrics based on prediction content The 7 kernels were used on the full dataset and the performance metrics were recorded. They are are summarized in Table 4.10. B-Spline KANs and RBF perform better overall.

Performance metrics based on OS performance The model is evaluated based on prediction ability, as well as device performance, namely battery usage. A linear battery usage model is assumed to evaluate battery usage. In this case, the device battery was loaded to full capacity, and subsequently the HAR Detector app was launched. The initial battery level was 100%. The application was executed in first plan for 20 minutes, and its recorded battery level was 94%. It was extrapolated that, assuming linear energy consumption and battery drain, if the battery was discharged 6% after 20 minutes, it will be totally discharged in approximately

Figure 4.4: *Side elevation of First Flight of stairs*

Figure 4.5: *Front Elevation of First Flight of Stairs*

333 min, or 5,5 hr.

Qualitative Evaluation In real-time we can collect characteristics that are not measurable about the model, such as the perception we have about the accuracy of the model. In this case, 3 subjects that were not part of data collection tested the application and the impressions recorded. The impression of the subjects was that the application was "responsive" and "fast", "accurate when laying, sitting, standing", "confuses walking with walking upstairs".

Correcting class imbalance - Collecting more data Given that the classes that perform worse on classification have the lowest number of instances in the dataset, additional data is included to extend the original full dataset for subjects 4 and 6. The resulting dataset has

Figure 4.6: *Staircase Interval Detail*

Figure 4.7: *Top View of Second Flight of Stairs*

753,289 records and the label counts in Table 4.11. The training set has instances 8,668 and the test set has 2,168 instances.

This represented a 2 to 3% increase in a given metric, with B-Spline KANs and RBF KANs performing better than the rest.

Correcting class imbalance - Collecting finer data In order to avoid collecting data from stairs intervals, a feature was developed in the data acquisition module as has been described in Chapter 3, to allow us to collect data for a given activity starting and stopping data collection on-demand.

As was stated before, we need to evaluate if the predictive capacity of our model increases

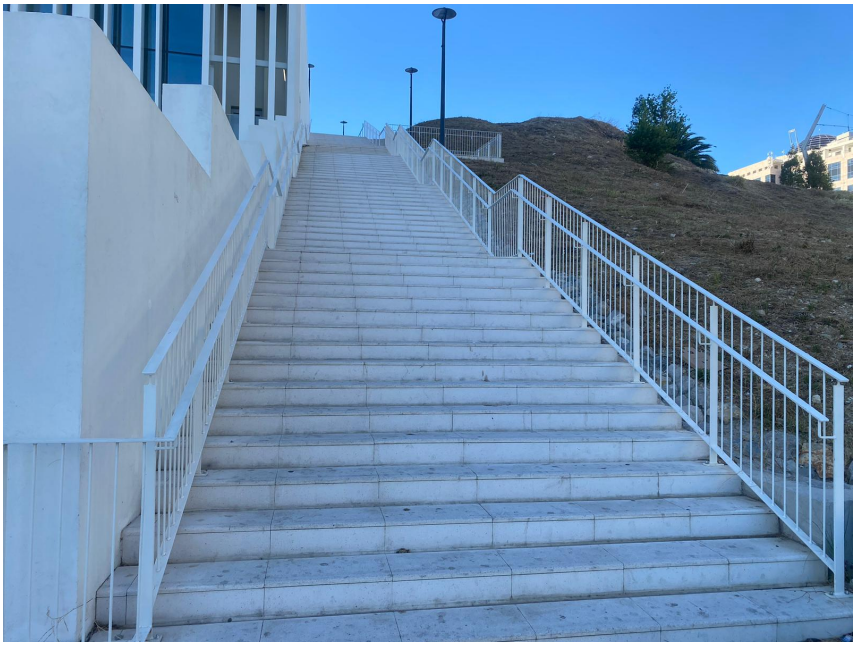


Figure 4.8: *Bottom View of Second Flight of Stairs*

Figure 4.9: *Side Elevation of Second Flight of Stairs*

when we eliminate the intervals between steps. For that, that we developed a feature to permit turning off and on collection of a given activity on demand. We use that feature to collect more data to correct class imbalance according to Table 4.13.

We activate data collection upon start of a stair flight, we turn it off in the end of the stair flight.

All metrics were not improved when doing this change, so we take that having class balance improves classification capabilities of the model, but having on-demand collection for stairs-related activities does not improve classification. According to the results that were presented, step interval of the present size does not affect the model predictive capacity.

Table 4.8: XGBoost Results for Activity Recognition by Subject and Activity

Subject	Metric	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
All Subjects	Precision	1.00	1.00	1.00	0.87	0.89	0.89
	Recall	1.00	0.99	0.98	0.92	0.88	0.85
	F1-Score	1.00	1.00	0.99	0.89	0.88	0.87
Subject 1	Precision	1.00	1.00	1.00	0.88	0.80	0.77
	Recall	0.98	1.00	0.98	0.92	0.77	0.72
	F1-Score	0.99	1.00	0.99	0.90	0.78	0.75
Subject 2	Precision	1.00	1.00	1.00	0.88	0.75	0.88
	Recall	1.00	1.00	1.00	0.95	0.73	0.84
	F1-Score	1.00	1.00	1.00	0.91	0.74	0.86
Subject 3	Precision	1.00	1.00	1.00	0.88	0.87	0.89
	Recall	1.00	1.00	0.99	0.92	0.85	0.87
	F1-Score	1.00	1.00	1.00	0.90	0.86	0.88
Subject 4	Precision	1.00	1.00	1.00	0.88	0.87	0.91
	Recall	0.98	1.00	1.00	0.94	0.83	0.91
	F1-Score	0.99	1.00	1.00	0.91	0.85	0.91
Subject 5	Precision	1.00	1.00	1.00	1.00	0.71	0.86
	Recall	1.00	1.00	1.00	0.91	0.69	0.82
	F1-Score	1.00	1.00	1.00	0.95	0.70	0.84
Subject 6	Precision	1.00	1.00	1.00	0.96	0.99	1.00
	Recall	1.00	1.00	1.00	0.98	1.00	0.99
	F1-Score	1.00	1.00	1.00	0.97	0.99	0.99

Label	Train Instances	Test Instances
LAYING	1,690	419
SITTING	1,471	380
STANDING	1346	322
WALKING	1,586	397
WALKING_UPSTAIRS	881	209
WALKING_DOWNSTAIRS	1,024	272

Table 4.9: Instances of each label for training and testing datasets.

Table 4.10: Performance metrics for different kernels with full dataset.

Algorithm	Train Loss	Test Loss	Test Accuracy	Precision	Recall	F1 Score
CauchyKAN	0.1481	**0.2484**	0.9149	0.9033	0.9005	0.9017
ChebyshevKAN	0.0596	0.3941	0.9029	0.8934	0.8928	0.8903
KANs	0.0479	0.5090	**0.9193**	0.9053	0.9094	0.9059
LaplacianKAN	0.3096	0.3194	0.8757	0.8605	0.8519	0.8553
MaternKAN	0.1098	0.2739	0.9158	0.9031	**0.9080**	0.9036
RationalQuadraticKAN	0.1215	0.2743	0.9188	0.9062	0.9061	0.9059
RBFKAN	**0.0296**	0.4973	0.9192	**0.9122**	0.9053	**0.9079**

Table 4.11: Activity counts in expanded dataset 1.

Activity	Raw Count	Train Count	Test Count
LAYING	92,446	1,702	419
SITTING	91,885	1,447	380
WALKING	135,544	1340	397
STANDING	85,732	1,092	322
WALKING_UPSTAIRS	190,798	1,266	209
WALKING_DOWNSTAIRS	156,884	1,500	272

Window Size Our experiments involved testing a range of window sizes from 25 to 200 with step 25 to determine the optimal balance between predictive capacity and latency. The results of this iterative search are present in Table 4.15 For comparison we included the accuracy

Table 4.12: Performance metrics of different kernels with expanded dataset

Kernel	Train Loss	Test Loss	Test Accuracy	Precision	Recall	F1 Score
CauchyKAN	0.1548	**0.2643**	0.9046	0.9216	0.9161	0.9185
ChebyshevKAN	0.0412	0.4047	0.9135	0.9275	0.9274	0.9271
KANs	**0.0006**	0.4126	**0.9232**	**0.9369**	0.9325	**0.9345**
LaplacianKAN	0.3088	0.3373	0.8875	0.9052	0.9019	0.9033
MaternKAN	0.1232	0.3601	0.9119	0.9307	0.9242	0.9260
RationalQuadraticKAN	0.1190	0.3002	0.9207	0.9351	**0.9338**	0.9342
RBFKAN	0.0010	0.4805	0.9206	0.9332	0.9310	0.9320

Activity	Raw Count	Train Count	Test Count
LAYING	92,446	737	187
SITTING	91,885	753	166
WALKING	135,544	684	173
STANDING	85,732	1,086	269
WALKING_UPSTAIRS	190,798	1,219	293
WALKING_DOWNSTAIRS	156,884	1,511	410

Table 4.13: Activity counts in expanded dataset 2.

Algorithm	Train Loss	Test Loss	Test Accuracy	Precision	Recall	F1 Score
CauchyKAN	0.1472	**0.3349**	0.8854	0.9094	0.9067	0.9053
ChebyshevKAN	0.0623	0.4675	0.8945	0.9129	0.9097	0.9108
KANs	**0.0026**	0.4963	**0.9024**	**0.9202**	**0.9198**	0.9199
LaplacianKAN	0.3355	0.3744	0.8630	0.8828	0.8841	0.8827
MaternKAN	0.0994	0.3739	0.9043	0.9196	0.9222	**0.9208**
RationalQuadraticKAN	0.1383	0.3410	0.8942	0.9139	0.9160	0.9136
RBFKAN	0.2359	0.6175	0.8845	0.9062	0.9079	0.9058

Table 4.14: Performance metrics with on-demand data collection

with Random Forest Classifier. For each window size, the table includes test loss, accuracy, precision, recall and f1-score. In each step, the gain in accuracy, precision, recall and f1-score is calculated in percentage. The window size that yielded better gains was 100. After 100 there are gains of performance for the model, but they are not considerable enough to justify the increase in prediction latency.

Sometimes when window size increases model performance decreases. We find this behavior to be counter-intuitive, given that less source samples limit the capacity to distinguish one activity from the other when activity patterns are more similar. But we also find that the performance decrease may be due to the following: larger window size might average out short-term variations and so sometimes diminish predictive capacity. Also, longer window sizes might also include more noise in the data and require adjusting the noise removal process.

Table 4.15: Optimal window size iterative search

Window Size	Random Forest Accuracy	Test Loss	Test Accuracy	Precision	Recall	F1-Score	Gain in Accuracy	Gain in Precision	Gain in Recall	Gain in F1-Score
25	0.9102	0.4192	0.8870	0.9087	0.9086	0.9084				
50	0.9097	0.6200	0.8975	0.9187	0.9204	0.9192	1.18%	1.10%	1.30%	1.19%
75	0.9138	0.6107	0.8911	0.9128	0.9071	0.9098	-0.71%	-0.64%	-1.45%	-1.02%
100	0.9203	0.4126	0.9232	0.9369	0.9325	0.9345	3.60%	2.64%	2.80%	2.71%
125	0.9203	0.5465	0.9246	0.9274	0.9256	0.9264	0.15%	-1.01%	-0.74%	-0.87%
150	0.9223	0.4099	0.9181	0.9369	0.9305	0.9334	-0.70%	1.02%	0.53%	0.76%
175	0.9128	0.3844	0.9178	0.9371	0.9377	0.9374	-0.03%	0.02%	0.77%	0.43%
200	0.9128	0.3132	0.9308	0.9451	0.9429	0.9439	1.42%	0.85%	0.55%	0.69%

Through this process, we found that a window size of 100 consistently provided the best results, delivering higher perceived accuracy while keeping the latency at 2 seconds. This particular window size effectively balances the trade-off between model performance and responsiveness, making it well-suited for real-time applications dependent on low latency. The 2-second latency ensures that the system remains responsive enough for practical use, while still benefiting from the enhanced predictive capacity achieved with the 100-sized window.

The other alternative that was presented to us by this search was defining window size of 200. This we considered that was not reasonable, given that with 50Hz sampling frequency, we would have a system latency of 4 seconds, which we found too high for the present use-case.

Noise Removal Several approaches to noise removal were tested, namely not using any noise removal technique, using a Butterworth filter and Savitzky-Golay filter.

In the implementation of the Butterworth filter, a cutoff frequency of 0.1 Hz was used, which determines the threshold above which high-frequency components are attenuated. The data was sampled at a frequency of 1.0 Hz, and the filter was set to an order of 4, which controls the steepness of the transition between the passband and the stopband. These parameters work together to provide smooth filtering of the signal, effectively reducing high-frequency noise.

The Savitzky-Golay filter, fit for removal of Gaussian or random noise, was applied with a window length of 51 and a polynomial order of 3. The window length, an odd integer, defines the number of data points used for smoothing, and the polynomial order specifies the degree of the polynomial fitted to the data within each window. These choices enable the filter to reduce noise while maintaining significant features of the signal, ensuring that important characteristics like peaks and trends are preserved despite the smoothing process.

Table 4.16: Window size search with expanded model, B-Spline KANs and Savitzky-Golay filter

Window Size	Random Forest Accuracy	Test Loss	Test Accuracy	Precision	Recall	F1-Score
25	0.9118	0.3844	0.8966	0.9180	0.9169	0.9170
50	0.9110	0.6230	0.8865	0.9105	0.9119	0.9110
75	0.9153	0.6207	0.8989	0.9157	0.9138	0.9146
100	0.9170	0.5068	0.9140	0.9291	0.9229	0.9257
125	0.9136	0.6144	0.9009	0.9202	0.9171	0.9185
150	0.9223	0.3763	0.9192	0.9368	0.9313	0.9339
175	0.9082	0.3668	0.9272	0.9381	0.9420	0.9399
200	0.9323	0.3197	0.9353	0.9474	0.9473	0.9470

Results with butterworth filter are omitted as they were worse than without any noise removal technique. Table 4.16 showcases the evolution of metrics across different window sizes using Savitzky-Golay filter. While for lower window sizes the results are better than with no noise removal, the better results occur in window sizes too high to be practical. Window size 100 has better results with no noise removal technique, so that continues to be the most balanced approach considering accuracy and latency.

5 Conclusions

In general, the Laplacian kernel performs worse than other kernels, with the Chebyshev and Mat'ern kernels achieving the best overall results. The training and testing losses in this implementation are better than those reported in the literature [53].

- The Chebyshev polynomial kernel consistently performs well across multiple datasets in terms of accuracy, precision, recall, and F1-score. It shows stable results in both training and testing phases, indicating robustness against outliers.
- The Matèrn kernel generally performs well, particularly notable for its balanced precision and recall across different datasets.
- The radial basis function kernel demonstrates strong performance on some datasets but shows less stability compared to the Chebyshev and Matèrn kernels.
- The Laplacian kernel consistently underperforms across datasets, especially in terms of accuracy. This may be due to its high sensitivity to variations. However, it could still have utility for anomaly detection.

Current kernel methods often outperform SVMs, XGBoost, and MLP in accuracy and other metrics. Specific datasets highlight the strengths of certain kernels; for example, the rational quadratic kernel excels on MNIST. Future research will explore hybrid kernels that combine the strengths of multiple kernels and evaluate their performance concerning catastrophic forgetting. Enhancing interpretability through symbolic representation of model components and pruning during implementation will also be investigated.

In contrast, on-demand data collection did not produce similar improvements and reintroduced some of the class imbalance issues that the expanded dataset had resolved. This was evident in the generally lower test accuracy and higher test loss for most kernels. For example, KANs's test accuracy declined, as did its precision and F1 scores, indicating a reduced ability to generalize to newly collected data. These results support our hypothesis that stair intervals negatively impact the classification results. We conclude that the steps performed between stair flights are insufficient to label the activity as WALKING.

The model's evaluation considered both prediction accuracy and battery usage, using a linear battery consumption model. Running the HARDetector app for 20 minutes reduced the battery level from 100

Another limitation to generalizing this method for predicting Human Activity Recognition (HAR) in real time is sensor placement. In everyday use, subjects place mobile phones in various locations, such as purses, back pockets, front pockets, or carry them in hand. This variability in placement requires a method proven to be effective under different sensor orientations. Current methods assume that smartphones are carried on the waist, as has been common practice in prior academic studies, including [3]. However, this assumption does not generalize well to real-world usage patterns.

Our experiments with varying window sizes from 20 to 200 showed that a window size of 100 yielded the best results in terms of predictive accuracy, with a corresponding latency of 2 seconds. This configuration strikes an effective balance, optimizing model accuracy while maintaining low latency suitable for real-time applications.

Several areas for future research can build upon the findings of this study, namely:

- **Comparison of the performance of KANs on our custom dataset with other methods:** A comprehensive comparison of real-time KANs with not only XGBoost but other traditional methods, such as SVMs, and modern methods, including MLPs, can be conducted. This would involve evaluating their performance for real-time classification HAR to determine which method performs better. Include more metrics in the comparison.
- **Comprehensive Study of Battery Usage:** A more detailed investigation into battery usage can be conducted, incorporating profiling tools to provide precise results into the energy demands of real-time HAR. This study can aim at evaluating KANs regarding battery usage by comparing them with other methods.
- **Privacy-Preserving Subject Identification:** Future work can explore methods to identify subjects while obfuscating their identities to preserve privacy. This could involve using techniques like differential privacy or cryptographic methods to anonymize the data without sacrificing classification accuracy.
- **Integrating KANs with Transformer Architectures for LLMs:** The integration of KANs into Transformer architectures offers a promising direction, particularly because they don't use matrix multiplication and require fewer parameters. This alternative could be less demanding in terms of energy consumptions while maintaining acceptable performance.

Bibliography

[1] N Albukhary and Y. Mustafah. "Real-time human activity recognition". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 260. 1. IOP Publishing. 2017, p. 012017 (cit. on pp. 1, 13).

[2] R. Andrews, J. Diederich, and A. B. Tickle. "Survey and critique of techniques for extracting rules from trained artificial neural networks". In: *Knowledge-based systems* 8.6 (1995), pp. 373–389 (cit. on p. 25).

[3] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, et al. "A public domain dataset for human activity recognition using smartphones." In: *Esann*. Vol. 3. 2013, p. 3 (cit. on pp. 1, 14, 22, 25–29, 42).

[4] P. Barbiero, G. Squillero, and A. Tonda. "Uncovering coresets for classification with multi-objective evolutionary algorithms". In: *arXiv preprint arXiv:2002.08645* (2020) (cit. on p. 25).

[5] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994). PubMed-not-MEDLINE, pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181 (cit. on p. 8).

[6] Z. Bozorgasl and H. Chen. *Wav-KAN: Wavelet Kolmogorov-Arnold Networks*. 2024. arXiv: 2405.12832 [cs.LG]. URL: https://arxiv.org/abs/2405.12832 (cit. on p. 7).

[7] J. Braun and M. Griebel. "On a constructive proof of Kolmogorov's superposition theorem". In: *Constructive approximation* 30 (2009), pp. 653–675 (cit. on pp. 4, 8).

[8] B. Cao. *A More Efficient Version of KAN*. https://github.com/Blealtan/efficient-kan. Accessed: 2024-07-17. 2024 (cit. on p. 23).

[9] J. Carbone. *Expect Sensor Prices to Fall*. Accessed: 2024-07-23. 2013. URL: https://www.digikey.com/en/articles/expect-sensor-prices-to-fall (cit. on p. 1).

[10] C. Chen, K. Liu, and N. Kehtarnavaz. "Real-time human action recognition based on depth motion maps". In: *Journal of real-time image processing* 12 (2016), pp. 155–163 (cit. on pp. 1, 12, 13).

[11] K. Davis, E. Owusu, V. Bastani, L. Marcenaro, J. Hu, C. Regazzoni, and L. Feijs. "Activity recognition based on inertial sensors for ambient assisted living". In: *2016 19th international conference on information fusion (fusion)*. Ieee. 2016, pp. 371–378 (cit. on pp. 25, 27, 28).

[12] D. Duvenaud. "Automatic model construction with Gaussian processes". PhD thesis. Apollo - University of Cambridge Repository, 2014. DOI: 10.17863/CAM.14087. URL: https://www.repository.cam.ac.uk/handle/1810/247281 (cit. on pp. 4, 11).

[13] D. Fakhoury, E. Fakhoury, and H. Speleers. *ExSpliNet: An interpretable and expressive spline-based neural network*. 2022. arXiv: 2205.01510 [cs.LG]. URL: https://arxiv.org/abs/2205.01510 (cit. on p. 10).

[14] S. Fallmann and J. Kropf. "Human activity recognition of continuous data using Hidden Markov Models and the aspect of including discrete data". In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE. 2016, pp. 121–126 (cit. on pp. 27–29).

[15] L. Gao, A. Bourke, and J. Nelson. "Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems". In: *Medical engineering & physics* 36.6 (2014), pp. 779–785 (cit. on pp. 1, 13).

[16] D. Garcia-Gonzalez, D. Rivero, E. Fernandez-Blanco, and M. R. Luaces. "Deep learning models for real-life human activity recognition from smartphone sensor data". In: *Internet of Things* 24 (2023), p. 100925 (cit. on p. 15).

[17] R. Genet and H. Inzirillo. *TKAN: Temporal Kolmogorov-Arnold Networks*. 2024. arXiv: 2405.07344 [cs.LG]. URL: https://arxiv.org/abs/2405.07344 (cit. on pp. 4, 7).

[18] S. M. Hammer, D. A. Katzenstein, M. D. Hughes, H. Gundacker, R. T. Schooley, R. H. Haubrich, W. K. Henry, M. M. Lederman, J. P. Phair, M. Niu, et al. "A trial comparing nucleoside monotherapy with combination therapy in HIV-infected adults with CD4 cell counts from 200 to 500 per cubic millimeter". In: *New England Journal of Medicine* 335.15 (1996), pp. 1081–1090 (cit. on pp. 25, 26).

[19] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385 (cit. on p. 8).

[20] K. J. Heuvers. "A characterization of Cauchy kernels". In: *aequationes mathematicae* 40.1 (Dec. 1990), pp. 281–306. ISSN: 1420-8903. DOI: 10.1007/BF02112301. URL: https://doi.org/10.1007/BF02112301 (cit. on pp. 4, 11).

[21] S. Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen". In: (Apr. 1991) (cit. on p. 8).

[22] H. Hofmann. *Statlog (German Credit Data)*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77. 1994 (cit. on p. 25).

[23] A. Ignatov. "Real-time human activity recognition from accelerometer data using convolutional neural networks". In: *Applied Soft Computing* 62 (2018), pp. 915–922 (cit. on pp. 1, 13).

[24] S. Ioffe and C. Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167 (cit. on p. 8).

[25] *Kaggle*. https://www.kaggle.com (cit. on p. 25).

[26] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361 (cit. on pp. 8, 9).

[27] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler. "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring". In: *IEEE transactions on information technology in biomedicine* 10.1 (2006), pp. 156–167 (cit. on pp. 1, 14, 20).

[28] A. Kearney and GSma Wireless Intelligence. *The Mobile Economy 2013*. Accessed: 2024-08-17. 2013. URL: https://www.gsma.com/newsroom/wp-content/uploads/2013/12/GSMA-Mobile-Economy-2013.pdf (cit. on p. 1).

[29] M. Kelly, R. Longjohn, and K. Nottingham. *The UCI Machine Learning Repository*. https://archive.ics.uci.edu (cit. on p. 25).

[30] A. Khtun and S. G. S. Hossain. "A Fourier Domain Feature Approach for Human Activity Recognition & Fall Detection". In: *2023 10th International Conference on Signal Processing and Integrated Networks (SPIN)*. 2023, pp. 40–45. DOI: 10.1109/SPIN57001.2023.10116360 (cit. on pp. 1, 14).

[31] A. Kolmogorov. "On the representation of functions of several variables as a superposition of functions of a smaller number of variables". In: Springer Berlin Heidelberg, 2009. Chap. 5 (cit. on pp. 4, 8).

[32] A. Kratsios and L. Papon. *Universal Approximation Theorems for Differentiable Geometric Deep Learning*. 2022. arXiv: 2101.05390 [cs.LG]. URL: https://arxiv.org/abs/2101.05390 (cit. on p. 8).

[33] O. D. Lara and M. A. Labrador. "A mobile platform for real-time human activity recognition". In: *2012 IEEE consumer communications and networking conference (CCNC)*. IEEE. 2012, pp. 667–671 (cit. on p. 12).

[34] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (cit. on pp. 4, 8).

[35] Y. LeCun, C. Cortes, and C. J. Burges. *The MNIST Database of Handwritten Digits*. http://yann.lecun.com/exdb/mnist/. New York, USA. 1998 (cit. on pp. 25, 26).

[36] C. Li, X. Liu, W. Li, C. Wang, H. Liu, and Y. Yuan. *U-KAN Makes Strong Backbone for Medical Image Segmentation and Generation*. 2024. arXiv: 2406.02918 [eess.IV]. URL: https://arxiv.org/abs/2406.02918 (cit. on p. 4).

[37] H. W. Lin, M. Tegmark, and D. Rolnick. "Why Does Deep and Cheap Learning Work So Well?" In: *Journal of Statistical Physics* 168.6 (2017), pp. 1223–1247. ISSN: 1572-9613. DOI: 10.1007/s10955-017-1836-5. URL: https://doi.org/10.1007/s10955-017-1836-5 (cit. on p. 10).

[38] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark. "KAN: Kolmogorov-Arnold Networks". In: *arXiv preprint arXiv:2404.19756* (2024). DOI: https://doi.org/10.48550/arXiv.2405.07200 (cit. on pp. 4, 7, 9, 10).

[39] T. Menzies and J. Di Stefano. "How Good is your Blind Spot Sampling Policy". In: vol. 8. Jan. 2004, pp. 129–138. DOI: 10.1109/HASE.2004.1281737 (cit. on pp. 25, 26).

[40] E. J. Michaud, Z. Liu, and M. Tegmark. "Precision Machine Learning". In: *Entropy* 25.1 (Jan. 2023), p. 175. ISSN: 1099-4300. DOI: 10.3390/e25010175. URL: http://dx.doi.org/10.3390/e25010175 (cit. on p. 9).

[41] L. Munkhdalai, T. Munkhdalai, O.-E. Namsrai, J. Y. Lee, and K. H. Ryu. "An empirical comparison of machine-learning methods on bank client credit assessments". In: *Sustainability* 11.3 (2019), p. 699 (cit. on pp. 27, 28).

[42] A. Nedorubova, A. Kadyrova, and A. Khlyupin. *Human Activity Recognition using Continuous Wavelet Transform and Convolutional Neural Networks*. 2021. arXiv: 2106.12666 [cs.CV]. URL: https://arxiv.org/abs/2106.12666 (cit. on pp. 1, 14).

[43] OpenML. *OpenML Task 146065*. https://api.openml.org/t/146065. Last accessed 2024-06-19 (cit. on pp. 27, 28).

[44] M. Pato. *The ISELthesis LaTeX Template's Manual*. Instituto Superior de Engenharia de Lisboa (ISEL-IPL). 2024. URL: https://github.com/matpato/iselthesis (cit. on p. viii).

[45] E. Pishchik. *Trainable Activations for Image Classification*. 2023. DOI: 10.20944/preprints202301.0463.v1. URL: https://github.com/Pe4enIks/TrainableActivation (cit. on p. 29).

[46] T. Poggio, A. Banburski, and Q. Liao. *Theoretical Issues in Deep Networks: Approximation, Optimization and Generalization*. 2019. arXiv: 1908.09375 [cs.LG]. URL: https://arxiv.org/abs/1908.09375 (cit. on p. 9).

[47] Q. Qiu, T. Zhu, H. Gong, L. Chen, and H. Ning. "ReLU-KAN: New Kolmogorov-Arnold Networks that Only Need Matrix Addition, Dot Multiplication, and ReLU". In: (2024). arXiv: 2406.02075 [cs.LG]. URL: https://arxiv.org/abs/2406.02075 (cit. on pp. 7, 9).

[48] K. S. Raju, M. R. Murty, M. V. Rao, and S. C. Satapathy. "Support vector machine with k-fold cross validation model for software fault prediction". In: *International Journal of Pure and Applied Mathematics* 118.20 (2018), pp. 321–334 (cit. on p. 28).

[49] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press, 2006 (cit. on pp. 4, 11).

[50] M. Rupp. "Machine learning for quantum mechanics in a nutshell". In: *International Journal of Quantum Chemistry* 115 (2015), pp. 1058–1073. URL: https://api.semanticscholar.org/CorpusID:94314605 (cit. on pp. 4, 11).

[51] N. D. Samuel Costa Matilde Pato. "An empirical study on the application of KANs for classification". In: *Proceedings of the 2024 8th International Conference on Advances in Artificial Intelligence*. ICAAI '24. Istanbul, Turkiye: Association for Computing Machinery, 2024. ISBN: 979-8-4007-1801-4 (cit. on p. 5).

[52] R. Singh and R. R. Ade. "Various Approaches for Multiclass Imbalance Learning Issues with MLP". In: *Communications on Applied Electronics* 5.4 (2016), pp. 13–16 (cit. on pp. 27, 28).

[53] S. SS, K. AR, G. R, and A. KP. "Chebyshev Polynomial-Based Kolmogorov-Arnold Networks: An Efficient Architecture for Nonlinear Function Approximation". In: *arXiv preprint arXiv:2405.07200* (2024). DOI: https://doi.org/10.48550/arXiv.2405.07200 (cit. on pp. 4, 7, 10, 27–30, 41).

[54] sscosta. *GitHub: Bench-KAN*. https://github.com/sscosta/bench-kan. Accessed: 2024-07-17 (cit. on pp. 5, 23).

[55] T. Su, H. Sun, C. Ma, L. Jiang, and T. Xu. "HDL: Hierarchical deep learning model based human activity recognition using smartphone sensors". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8 (cit. on p. 29).

[56] J. Sütő, S. Oniga, and A. Buchman. "Real time human activity monitoring". In: *Annales Mathematicae et Informaticae*. Vol. 44. Eszterházy Károly University Eger, Hungary. 2015, pp. 187–196 (cit. on pp. 1, 14, 20).

[57] J. Suto, S. Oniga, C. Lung, and I. Orha. "Comparison of offline and real-time human activity recognition results using machine learning techniques". In: *Neural computing and applications* 32.20 (2020), pp. 15673–15686 (cit. on pp. 1, 13, 14).

[58] J. Suto, S. Oniga, and P. P. Sitar. "Feature analysis to human activity recognition". In: *International Journal of Computers Communications & Control* 12.1 (2017), pp. 116–130 (cit. on p. 14).

[59] L. Taylor, A. King, and N. Harper. "Robust and accelerated single-spike spiking neural network training with applicability to challenging temporal tasks". In: *arXiv preprint arXiv:2205.15286* (2022) (cit. on pp. 27–29).

[60] W. Taylor, S. A. Shah, K. Dashtipour, A. Zahid, Q. H. Abbasi, and M. A. Imran. "An intelligent non-invasive real-time human activity recognition system for next-generation healthcare". In: *Sensors* 20.9 (2020), p. 2653 (cit. on p. 14).

[61] D. Tomar and S. Agarwal. "Feature selection based least square twin support vector machine for diagnosis of heart disease". In: *International Journal of Bio-Science and Bio-Technology* 6.2 (2014), pp. 69–82 (cit. on pp. 27, 28).

[62] L. Tong, H. Ma, Q. Lin, J. He, and L. Peng. "A novel deep learning Bi-GRU-I model for real-time human activity recognition using inertial sensors". In: *IEEE Sensors Journal* 22.6 (2022), pp. 6164–6174 (cit. on p. 15).

[63] A. Tran, J. Guan, T. Pilantanakitti, and P. Cohen. *Action Recognition in the Frequency Domain*. 2014. arXiv: 1409.0908 [cs.CV]. URL: https://arxiv.org/abs/1409.0908 (cit. on pp. 1, 14).

[64] C. J. Vaca-Rubio, L. Blanco, R. Pereira, and M. Caus. *Kolmogorov-Arnold Networks (KANs) for Time Series Analysis*. 2024. arXiv: 2405.08790 [eess.SP]. URL: https://arxiv.org/abs/2405.08790 (cit. on pp. 4, 7).

[65] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. "OpenML: Networked Science in Machine Learning". In: *SIGKDD Explorations* 15.2 (2013), pp. 49–60. DOI: 10.1145/2641190.2641198. URL: http://doi.acm.org/10.1145/2641190.2641198 (cit. on p. 25).

[66] J.-P. Vert, K. Tsuda, and B. Schölkopf. "A Primer on Kernel Methods". In: MIT Press, July 2004, pp. 35–70. ISBN: 9780262256926. DOI: 10.7551/mitpress/4057.003.0004 (cit. on pp. 4, 10).

[67] S. Wan, L. Qi, X. Xu, C. Tong, and Z. Gu. "Deep learning models for real-time human activity recognition with smartphones". In: *mobile networks and applications* 25.2 (2020), pp. 743–755 (cit. on p. 14).

[68] Y. Wang, H. Liu, K. Cui, A. Zhou, W. Li, and H. Ma. "m-activity: Accurate and real-time human activity recognition via millimeter wave radar". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 8298–8302 (cit. on p. 15).

@is@a@figure