POLITECNICO DI TORINO

Master Degree course in Communications and Computer Networks Engineering

Master Degree Thesis

Evaluation of Indoor Human Trajectory Regression Techniques

Supervisors
Prof. Mihai Teodor Lazarescu

Candidate
SHI YIYANG

Academic Year 2024

Acknowledgements

Firstly, I wish to express my gratitude to my supervisor, Professor Mihai Teodor Lazarescu, for his generous guidance, encouragement, and patience throughout my thesis journey.

My sincere thanks also go to Guohui, whose practical advice was instrumental in the completion of my work. Without his assistance, the completion of this work would not have been feasible.

I would like to sincerely thank Andre GOMES PADUA for his assistance. With his help, I have greatly improved my understanding of domain knowledge and mastered learning techniques. I am delighted to have met Andre towards the end of my student life.

I also deeply appreciate the economic support provided by my parents during my studies.

Abstract

This thesis investigates machine learning techniques for predicting indoor human movement using data from capacitive sensors. Four models are evaluated: Long Short-Term Memory Networks (LSTM), Qlattice, Multilayer Perceptrons (MLP), and Kolmogorov-Arnold Networks (KANs). These models are assessed based on accuracy, computational efficiency, and interpretability within a controlled 3 m x 3 m area where human-induced electrical capacitance changes are monitored.

The study involves systematic data processing to train and test each model, utilizing metrics like Mean Squared Error (MSE) to evaluate performance, computational demands, and responsiveness. Findings indicate that LSTM and Qlattice offer high interpretability and reasonable accuracy, making them suitable for scenarios requiring understandable model decisions. In contrast, MLP and KANs demonstrate superior performance in handling complex movements but lack interpretability.

The research enhances understanding of indoor human trajectory prediction, suggesting practical applications in smart home automation, security, and elderly care. Future work aims to integrate more diverse sensor inputs and refine models for improved accuracy and real-world applicability, advancing both theoretical insights and practical applications in indoor human movement modeling.

Contents

Acronyms

AI Artificial intelligence. 20

CNN Convolutional Neural Networks. 12

KANs Kolmogorov-Arnold Networks. 1, 8–10, 13, 17, 23–29, 41–46, 48–51, 53

KNN k-Nearest Neighbor. 11

LAN Local Area Network. 27

LSTM Long Short-Term Memory Networks. 1, 8, 10, 12, 13, 17–20, 27, 29, 31, 33–37, 49–51, 53, 54

MAE Mean Absolute Error. 19

MLP Multilayer Perceptrons. 1, 8–10, 13, 17, 21–29, 41–46, 48–51, 53

MSE Mean Squared Error. 1, 8, 19, 29, 31, 33, 42

RMSE Root Mean Squared Error. 19

RNN Recurrent Neural Networks. 12, 18

SVM Support Vector Machines. 11

Chapter 1

Introduction

1.1 The Importance of Indoor Human Trajectory Analysis

The importance of indoor human trajectory analysis primarily lies in the management and optimization of indoor spaces. By enhancing our understanding and predictive capabilities of how people move within enclosed environments, we can better serve people's indoor living and improve the planning and design of indoor spaces, including enhancing safety and convenience. The significance of indoor human trajectory analysis is especially evident in the following areas:

- Medical monitoring in homes: As the aging population grows, more families need to care for elderly members. However, due to most family members having their own work and life commitments, it is challenging to provide comprehensive and real-time care. In this context, the analysis and prediction of elderly movement trajectories within the home become particularly important. Through indoor human trajectory analysis, it is possible to monitor the health conditions of the elderly in real time, thus enabling timely prevention and intervention of potential safety risks. This can significantly enhance the safety of elderly people living alone at home.

- Safety management in public indoor settings: Trajectory analysis and prediction can be applied in densely populated environments such as airports and shopping malls. By understanding typical human movement patterns, potential threatening actions can be detected, preventing accidents and effectively improving evacuation and emergency measures in public settings. This plays a crucial role in accident prevention.

- More rational planning for supermarkets and stores: By predicting and analyzing consumers' movement patterns within stores, supermarkets can adjust store layouts and optimize product placement based on estimated trajectories and hotspots of activity. Predictive analysis can also provide real-time feedback on the popularity of various products within the store, enabling better stock management and merchandise display planning.

- Integration with smart homes and the Internet of Things: As smart homes increasingly become a part of ordinary people's lives, the analysis and prediction of indoor movement trajectories become particularly important. By analyzing the movement patterns and real-time trajectories of residents, intelligent management and control of smart systems within rooms can be achieved, leading to automation in smart homes. This not only enhances the comfort and convenience of people's lives but also improves energy efficiency by adjusting the usage patterns of smart home devices.

In summary, indoor human trajectory analysis has numerous application scenarios that can enhance safety, operational efficiency, and user experience in various environments. With the development of artificial intelligence technologies, the impact and potential applications of such analyses continue to grow, making indoor spaces safer and more responsive, thus better serving humanity and meeting human needs more effectively.

1.2 Objectives of the Thesis

In this thesis, four methods LSTM, Qlattice, MLP, and KANs are used to implement the prediction of indoor human trajectories. The performance of these methods is analyzed, and the accuracy and performance differences between different methods are compared. The objectives are as follows:

1. **Assess the accuracy of each method:** The main goal is to assess the accuracy of predicting indoor human trajectories using LSTM, Qlattice, MLP, and KANs. Firstly, the data has been collected and processed, and then the processed dataset is applied to different models. By measuring the MSE, the accuracy of each model is compared and evaluated. Additionally, each model will plot the predicted human trajectory, which will be compared with the actual trajectory to assess the accuracy and reliability of each method.

2. **Analyze the computational efficiency of each model:** Different models correspond to different computing methods, and the computational efficiency of each model is an important criterion for evaluating the feasibility of the model. In this thesis, one of the main goals is to measure the time required to train and predict trajectories for each method and assess the feasibility and priority of various methods under limited computing resources. By comparing the time taken to complete the trajectory analysis tasks, an analysis comparison of the computational efficiency of different models is achieved.

3. **Compare the differences in interpretability between models:** LSTM, MLP, KANs, and Qlattice are four distinct types of machine learning models, each with unique features, particularly in terms of model interpretability. LSTM is a type of recurrent neural network suitable for sequence data, with an internal gating mechanism that can be understood as regulating the flow of information, but its complex

multilayer structure reduces interpretability. MLP is a basic feedforward neural network; although its structure is relatively simple, increased layer depth and the interaction between weights and activation functions also lower the model intuitiveness. KANs are based on the Kolmogorov-Arnold representation theorem, theoretically capable of mapping any multidimensional function accurately with a small number of nonlinear transformations. While theoretically interpretable, practical interpretability might be limited by understanding these transformations. Qlattice uses symbolic regression to explore models across various hypothesis spaces, typically producing models as concise mathematical expressions, thus providing the highest interpretability among these models. Users can directly see the mathematical form and relationships between variables.

1.3 Scope and Limitations

1.3.1 Scope of the Thesis

- Data Collection Scope: This study is limited to using capacitive sensors to collect data, capturing indoor human position data through capacitive sensors. Capacitive sensors are non-invasive sensors, characteristic of non-invasive sensors.
- Model Scope: This thesis is limited to using four different model methods to predict data, namely genetic algorithms, quantum-inspired algorithms, and two different neural networks.
- Application Environment Scope: The research is limited to indoor environments where capacitive sensors can effectively monitor human trajectories, such as homes, offices, or malls.
- Evaluation Metric Scope: In this thesis, the performance evaluation of each model includes accuracy, computational efficiency, and interpretability.

1.3.2 Limitations of the Thesis

1. **Limitations of Capacitive Sensors:** Although capacitive sensors can be used for non-invasive and privacy-respecting data collection, they also have their limitations. They may not provide the same level of detail as visual sensors (such as cameras). Meanwhile, capacitive sensors are severely affected by environmental disturbances and have limited detection distances. They are also sensitive to the properties of objects, such as size, shape, and dielectric constant. The use of capacitive sensors may lead to certain limitations and unsustainability in sensor selection.
2. **Scalability Limitations:** This thesis may not fully address the scalability of models in larger or more complex environments beyond the scope of data collection. The space is a 3 m x 3 m square space, and the situation and results may differ in more complex environments. Additionally, the results of this thesis may be highly related to the type and layout of indoor environments and the nature of data collected

through capacitive sensors. Generalizing these results to other spatial environments or different types of sensors may require additional modifications and testing.

3. **Limitations of Specific Models**

    - The limitations of LSTM models primarily lie in the potential for vanishing gradients when dealing with very long sequences and the need for extensive tuning and training resources when predicting extremely complex or rapidly changing data.
    - Qlattice is powerful, requires careful tuning to ensure it does not overfit and remains computationally feasible.
    - MLP requires a large amount of data for training to achieve optimal results, and its "black box" nature, compared to more transparent models like Gplearn, makes it difficult to interpret.
    - The main limitation of KANs is their high implementation complexity and significant computational resource demands, making them challenging to scale to large-scale or high-dimensional applications.

4. **Ethical and Privacy Issues**

    Although capacitive sensors are less invasive, any form of human subject data collection must adhere to strict ethical and privacy standards. Privacy concerns are often paramount in data collection, and compliant data collection is an important consideration.

Chapter 2

Literature Review

2.1 Overview of Trajectory Regression Methods

Trajectory regression analysis is a significant area of study, particularly when dealing with the prediction of human motion trajectories in indoor environments. It involves various techniques and methods, ranging from traditional statistical approaches to advanced machine learning and deep learning technologies. The aim of these methods is to enhance understanding of human motion trajectories, optimize environmental layout, enhance safety, and offer practical applications in business and medical fields. Here are some of the main methods currently used for trajectory regression:

1. **Statistical Methods:** Statistical approaches primarily include linear regression and polynomial regression, which are based on mathematical formulas and predict trajectories by minimizing error. These methods are usually suitable for situations where data relationships are simple and predictable, but they may not be accurate enough when dealing with complex nonlinear relationships.

2. **Machine Learning Methods:** Machine learning provides a way to automatically learn and improve from data, enabling automated data handling and analysis, suitable for more complex trajectory prediction challenges. Here are some common machine learning methods:

   - Support Vector Machines (SVM): By finding the optimal boundary between data categories, SVM can handle high-dimensional data and has shown good performance in both classification and regression tasks.
   - Decision Trees and Random Forests: These methods predict trajectories through a tree-like structure of decision rules, especially random forests, which improve prediction accuracy and stability by integrating multiple decision trees.
   - k-Nearest Neighbor (KNN): This method, based on distance or similarity measures, predicts by finding the k closest neighbors to a test point, and is applicable to various types of datasets.

3. **Deep Learning Methods:** Deep learning techniques excel in handling complex

and large-scale datasets and are particularly suitable for cases with large amounts of data, making them ideal for trajectory prediction:

- Convolutional Neural Networks (CNN): Although primarily used for image processing, CNN can also recognize spatial patterns in trajectory data.
- Recurrent Neural Networks (RNN) and LSTM: These networks are particularly suited for processing sequential data, such as time series or continuous trajectory points, capturing temporal dynamics in the data.

4. **Symbolic Regression Methods:** Symbolic regression, through the use of genetic programming, seeks the best mathematical models to describe data. Unlike traditional numeric regression, symbolic regression provides deeper interpretability, and has unique advantages in explainability.

2.2 Framework Overview: Indoor Human Trajectory Prediction

Establishing a comprehensive analytical framework is crucial when addressing indoor human trajectory prediction. This framework spans the entire process from data collection to model deployment, ensuring the systematic and scientific integrity of the research. Here is a detailed overview of the framework, involving key steps:

1. **Data Collection:** Data collection is the foundation of trajectory prediction, determining the quality and depth of subsequent analysis. In an indoor environment, human location data is collected using capacitive sensors, deploying four sensors at the front, back, left, and right of a 3 m x 3 m square. These sensors sample frequencies to provide continuous location data. It is crucial to note that the accuracy of these data and the choice of sampling frequency directly impact model performance.
   - Sensor Placement: Sensors are strategically placed to cover key areas, ensuring data continuity and completeness, completely covering a 3 m x 3 m square in this thesis.
2. **Data Preprocessing :** The purpose of data preprocessing is to ensure data quality, a necessary step to improve analysis efficiency. This includes:
   - Data Cleaning: Removing or correcting erroneous data points and dealing with missing values.
   - Data Transformation: Converting raw data into formats suitable for analysis, such as normalization or standardization, to mitigate the effects of differing scales.
   - Feature Extraction: Extracting features from raw data that aid in trajectory prediction.

3. **Model Development and Training:** Select appropriate machine learning or deep learning models for trajectory prediction. In this thesis, four models were chosen for analysis and prediction, including:

    - Model Selection: Choosing suitable models based on data characteristics and predictive needs, such as LSTM, Qlattice, MLP, and KANs.
    - Model Training: Training models using a dataset of existing coordinates, adjusting model parameters based on output results to improve performance, and performing steps like cross-validation to optimize model performance.
    - Model Validation: Evaluating model predictive performance using a test set to ensure good generalization capabilities and avoid overfitting.

4. **Performance Evaluation:** Evaluating model performance using various metrics, including:

    - Accuracy: The degree of alignment between predicted results and actual data. In this thesis, model prediction accuracy is judged by comparing predicted trajectories with actual trajectories, primarily through indoor pedestrian trajectory plots.
    - Efficiency: The speed at which models process data, which varies significantly among different models and reflects model efficiency. Efficiency is a crucial performance indicator, especially in scenarios requiring real-time predictions.

5. **Result Analysis and Interpretation:** Delving into model prediction results to provide explanations of model behavior

    - Result Visualization: Using charts and dynamic simulations to display prediction results, aiding in a better understanding of model predictive behaviors. In this thesis, charts primarily display prediction outcomes, comparing actual and predicted trajectories to analyze model results.
    - Result Interpretation: Explaining the underlying computational and analytical logic of the results, elucidating the decision-making process of the models.

2.3 Integration of Machine Learning and Symbolic Regression

The integration of machine learning and symbolic regression represents a cutting-edge research field, aiming to combine the data processing capabilities of machine learning with the model interpretability of symbolic regression. This integrated approach leverages the powerful learning capabilities of machine learning models while utilizing symbolic regression to generate interpretable mathematical models, thus offering deeper insights and understanding. Here are some key aspects of integrating machine learning with symbolic regression:

1. **Concept of Symbolic Regression:** Symbolic regression is a regression technique that uses genetic programming to discover the best symbolic expressions describing data. Unlike traditional numerical regression models, symbolic regression not only predicts outputs but also seeks to uncover the underlying mathematical relationships in data. This allows the model to offer predictive capabilities along with intuitive insights into the data generation process. Qlattice, as a representative of this approach, offers a direct expression of the mathematical relationships underlying data generation.

2. **Motivation for Integration:** While symbolic regression provides excellent interpretability, its performance may be less optimal when dealing with large-scale or highly complex data. Moreover, symbolic regression may struggle to capture all the intricate data relationships directly. Thus, integrating machine learning models with symbolic regression combines the strengths of both: the ability of machine learning models to process and learn from complex data, while maintaining interpretability. The complementary nature of the two can effectively compensate for their respective weaknesses.

3. **Integration Methods:** The integration typically involves the following steps:

    - Preprocessing and Feature Extraction: Use machine learning techniques, such as deep learning networks, to process raw data and extract features. These features capture complex patterns in the data, providing a richer input for symbolic regression.
    - Symbolic Model Generation: Input the extracted features into a symbolic regression model like Qlattice. Utilize genetic programming techniques to find the best model from thousands of possible mathematical expressions that describe the relationship between features and prediction targets.
    - Model Optimization and Validation: Combine traditional machine learning evaluation methods, such as cross-validation, to optimize and validate the performance of the integrated model.

4. **Application Example:** For instance, an integrated model using deep learning and Qlattice could first utilize a deep learning network to process complex patterns in image or video data, and then employ Qlattice to generate a mathematical expression describing the relationship between these patterns and the final outcomes. This approach not only enhances prediction accuracy but also provides intuitive explanations of model decisions through symbolic expressions.

By integrating machine learning with symbolic regression, researchers can create powerful and interpretable models, which is especially important for applications requiring high transparency and verifiability, such as in medical diagnostics or financial analysis. The development of this integrated method could help propel the widespread application of machine learning technologies while enhancing user trust and understanding of model predictions.

Chapter 3

Methodology

3.1 Source and Description of Experimental Data

The data used in this experiment originates from another project. [4] This project collected human movement trajectories within a 3 m x 3 m indoor space using capacitive sensors.

1. **Data Collection Environment**

   The data collection environment for this project is a 3 m x 3 m laboratory, free from any obstructions, simulating an open and controlled environment such as a small office or home setting. This setup helps minimize external variables, ensuring more accurate and consistent data collection. Consider the example of Fig. 3.1.

2. **Data Collection Process**

   In this project, the authors used four capacitive sensors placed at the front, back, left, and right positions within the indoor space. Each sensor was fixed at a specific height from the ground to effectively capture the capacitive changes caused by human movement. Consider the example of Fig. 3.2.

3. **Types of Collected Data and Data Processing**

   The data in the project was sampled at a frequency of 3 Hz, meaning data was collected three times per second. This sampling frequency balances the precision of capturing human movement with the real time processing of the data. A moving window is then established on this time series data to predict the next data point outside the window based on the data within the window.

   - Data Window: Data processing is performed within a 5 s window, containing 15 data points per window. This setup allows the capture of sufficient dynamic information without overwhelming the processing speed with excessive data.
   - The raw data undergoes preprocessing steps, including signal denoising and data normalization, to ensure the accuracy of subsequent analysis. The preprocessed data is then used for feature extraction, which is crucial for the training of machine learning models and trajectory prediction.

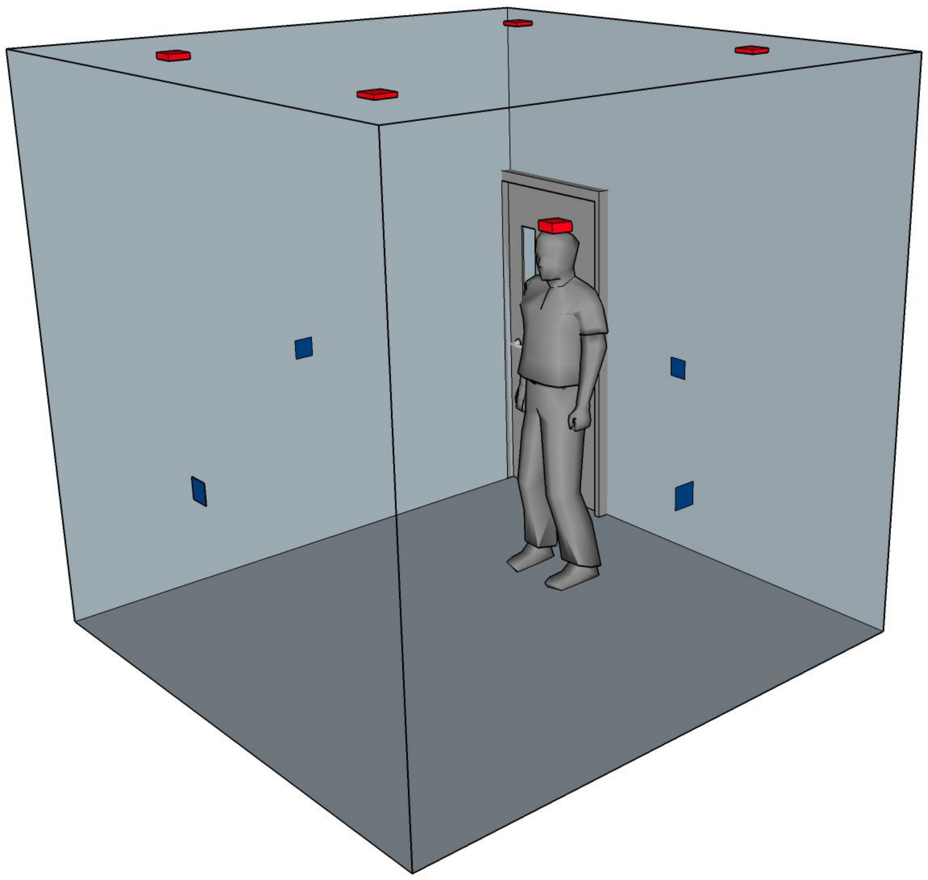


Figure 3.1. The laboratory space measures 3 m x 3 m [4]

3.2 Explanation of the Data Splitting Technique Used

The collected data is divided into three parts: the training set, the test set, and the validation set. This division is designed to evaluate the model performance and prevent overfitting. The training set comprises 60 % of the data, while the test set and the validation set each comprise 20 %.

The training set is used to train the model, meaning it adjusts the model parameters through learning from the training data. The validation set is used for model selection and tuning of hyperparameters, serving as a tool to evaluate the model performance during development. After the model development is complete, the test set is used to assess the model final performance.

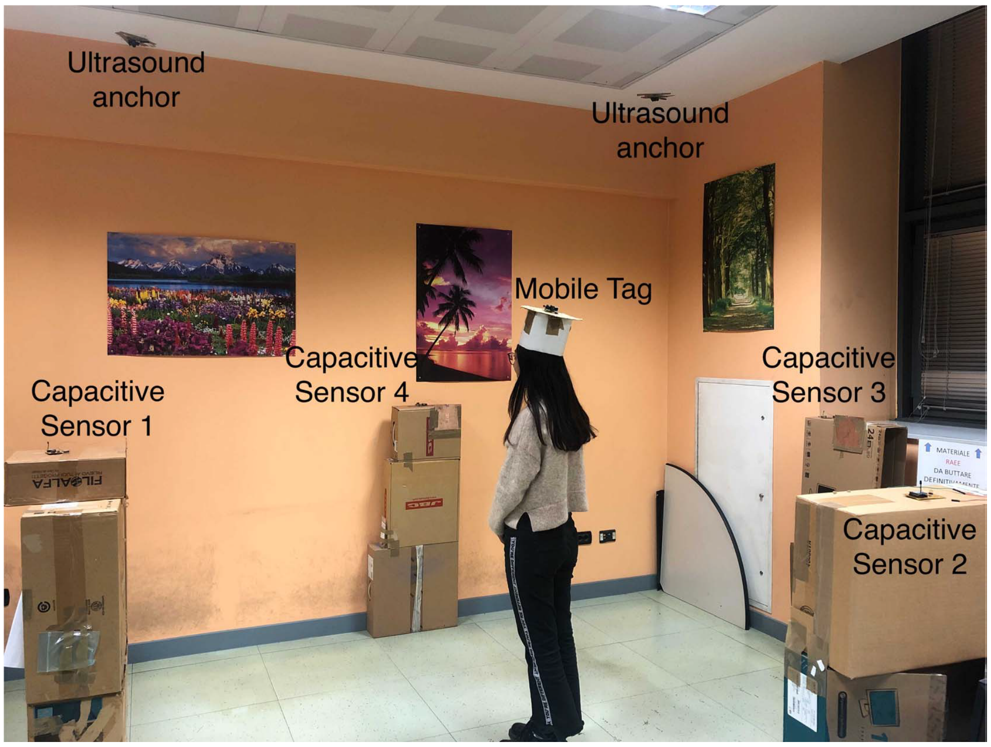


Figure 3.2. Capacitive Sensor Deployment [4]

3.3 Using Long Short-Term Memory Networks (LSTM), Qlattice, Multilayer Perceptrons (MLP) and Kolmogorov-Arnold Networks (KANs) for Data

For the collected and processed human trajectory data in an indoor environment, here are the introductions and detailed steps for analysis and prediction using four methods: LSTM, Qlattice, MLP, and KANs.

- Introduction of input data and a brief description of the training process for the four models: The input data during the training process consists of real trajectory data from the training set, which has been segmented from the larger dataset. A window of 15 data points in length is established, and each training session involves feeding the data information of these 15 positions within the window into the model. The goal is to predict the position of the next point based on the positional information of these 15 points. The complete training process involves progressively moving the window forward point by point across the training set, continuously predicting the position of the next point.

3.3.1 Long Short Term Memory Networks (LSTM)

LSTM is a type of RNN designed to effectively model temporal sequences and capture long-term dependencies in data. It addresses the limitations of traditional RNN, such as the vanishing gradient problem, by introducing a memory cell and gating mechanisms. Here is an overview of LSTM and its application methods

1. Introduction: LSTM networks use a specialized structure that includes memory cells and gates to regulate the flow of information. These components enable LSTM to retain and utilize information over long sequences, making them well-suited for tasks such as time series forecasting, natural language processing, and sequence classification.

   —

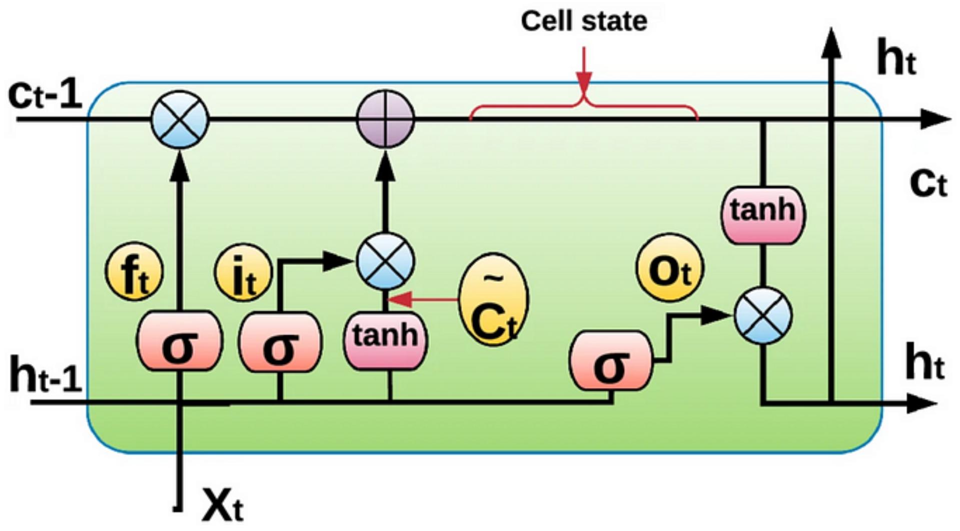


Figure 3.3. LSTM cell [2]

2. LSTM Architecture Fig. 3.3

   - Memory Cell: The core component that maintains the cell state over time, preserving information from previous time steps.
   - Gates: Three gates control the flow of information in and out of the memory cell:
     - Forget Gate: Decides what portion of the cell state should be discarded.
     - Input Gate: Determines which new information should be added to the cell state.
     - Output Gate: Regulates the information to be outputted based on the cell state.

3. Key Steps in Using Long Short-Term Memory Networks (LSTM)
    - Data Preparation: Read the training data from files and select appropriate feature columns and target variables. Sequence data often requires careful preprocessing, such as normalization and sequence padding.
    - Model Configuration: Instantiate an LSTM model using a deep learning framework (e.g., TensorFlow or PyTorch). Configure its parameters based on the problem complexity, such as the number of layers, units per layer, activation functions, and dropout rates.
    - Training the Model: Train the LSTM model using the prepared training data. Typically, optimization algorithms like Adam are used to update the model weights.
    - Model Validation and Testing: Evaluate the model performance using validation and test datasets. Common metrics include accuracy, MSE, and Mean Absolute Error (MAE).
    - Results Analysis: Analyze the model predictions and understand its behavior by visualizing the learned patterns and comparing them with actual data.
4. Evaluation Metrics
    - Similar to other machine learning algorithms, LSTM evaluation metrics include loss functions that measure the difference between predicted and actual values. Common loss functions for LSTM include: MAE,MSE,Root Mean Squared Error (RMSE).
    - For consistency, the same error function (MSE) can be used to compare LSTM with other methods like gplearn.
5. Typical Process
    - Initialization: Start by initializing the LSTM network with appropriate hyperparameters, such as the number of hidden layers, units, and dropout rates.
    - Training: During training, the model learns by updating its weights based on the loss calculated at each time step. This process involves backpropagation through time.
    - Optimization: Use techniques like learning rate scheduling and early stopping to optimize the model performance and prevent overfitting.
    - Hyperparameter Tuning: Experiment with different configurations to find the optimal set of hyperparameters for the given data.
6. Parallel Computing
    - Scalability: LSTM models can be computationally intensive, especially with large datasets and long sequences. Leveraging parallel computing and hardware accelerators like GPUs can significantly speed up the training process.

- Frameworks: Use deep learning frameworks that support parallel computing and distributed training to handle large-scale data efficiently.

By following these steps, LSTM networks can effectively model and predict sequential data, providing powerful insights into time-dependent patterns and relationships. Despite their complexity, LSTM models offer significant advantages in tasks requiring temporal understanding and long-term dependency modeling.

3.3.2 Qlattice

Qlattice is a quantum-inspired algorithm developed by Feynman Artificial intelligence (AI), designed for automatically discovering patterns and relationships in data. [1] Here is a basic overview of how Qlattice is typically used

- Data Preparation: Similar to gplearn, prepare your data by defining features and targets.
- Connect to Qlattice: Set up a connection to Qlattice and define your model query.
- Model Search: Explore potential solutions through Qlattice to find the models that best describe the data.
- Model Optimization: Select several of the best models for further optimization and validation.
- Evaluation and Testing: Assess the performance of the selected models on an independent test dataset.
- Result Application: Interpret the model outputs and apply them to practical problems or further analysis.

Qlattice operates within the Feyn framework, inspired by quantum mechanics, which evaluates the most probable models for a given problem and then selects and tests the best models. The data used in this thesis are positional time series data, which are utilized to train the Qlattice model to solve regression problems and predict the next position points of human trajectories indoors. Qlattice has a promising future in the field of deep learning research.
To some extent, to understand Qlattice, one could view it as a probability distribution from which models continuously sample. After each training batch, the distribution is updated with the best structures, removing poorly performing models. This selection process aims to make room for new, more competitive models.
In terms of setting parameters for Feyn, you can restrict the search to specific functions and impose limits on model complexity and other similar constraints by setting Feyn's parameters.
In Feyn, models are visualized as diagrams, which are easy to read. These diagrams transform mathematical formulas into input-to-output conversion flows, depicted as left-to-right, unidirectional flowcharts that illustrate the relationships between the features

used in each model. They also show how these features interact to produce results. Feyn's visualization and interpretability are particularly strong, offering unique features such as evaluating the relationship between one feature and another and identifying biases within the model. Consider the example of Fig. 3.4

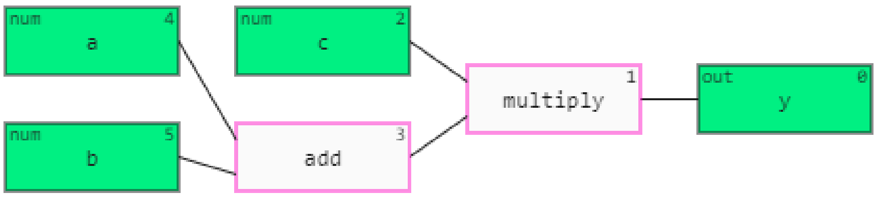


Figure 3.4. Qlattice models are visualized as diagrams [1]

In Qlattice, there are a variety of functions available for use in model calculations, known as interactions. Interactions are the basic computational units of each model, receiving data, transforming it, and then outputting it to the next model. Consider the example of Fig. 3.5.

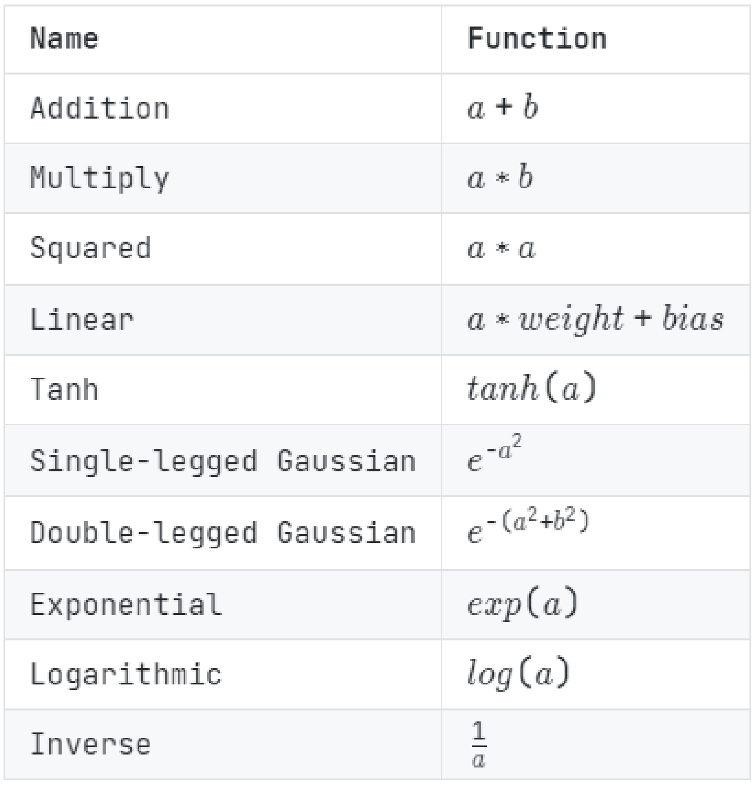
It's important to highlight the advantages of Qlattice as a supervised machine learning algorithm for symbolic regression. Qlattice allows users to decide which models are useful, which should be enhanced, and how to limit the decision space. To some extent, Qlattice involves users more directly and makes the model training process more comprehensible. Overall, here are the advantages of using the Qlattice method:

- Fewer nodes and connections are needed.
- Some functionalities are not typically seen in neural networks.
- The models are easier to inspect, simpler, and less prone to overfitting.
- The models are mathematical formulas, allowing you to infer the consequences of hypotheses.
- A diverse range of models have been tried, ensuring that nothing is missed during the training process.

3.3.3 Multilayer Perceptrons (MLP)

MLP is a deep learning method well-suited for addressing non-linear problems involving large amounts of complex data. [3] Here is the basic process for utilizing MLP.

- Data Preprocessing: Normalize or standardize the data to ensure effective input to the network.



Name	Function
Addition	$a + b$
Multiply	$a * b$
Squared	$a * a$
Linear	$a * weight + bias$
Tanh	$tanh(a)$
Single-legged Gaussian	e^{-a^2}
Double-legged Gaussian	$e^{-(a^2+b^2)}$
Exponential	$exp(a)$
Logarithmic	$log(a)$
Inverse	$\frac{1}{a}$

Figure 3.5. Unctions available for use in model calculations [1]

- Building the Network: Construct an MLP model using frameworks like TensorFlow or PyTorch, configuring multiple hidden layers and nodes.
- Model Training: Train the model on the training data, optimizing performance by adjusting parameters such as learning rate and batch size.
- Validation and Testing: Adjust model parameters on the validation set to avoid overfitting, then evaluate the model performance on the test set.
- Result Analysis: Analyze and interpret the model predictive results, assessing the model performance on unknown data.

The full name of MLP is multilayer perceptron, a deep learning model based on feed-forward neural networks, consisting of multiple layers of neurons, each fully connected to the preceding layer. multilayer perceptrons are used to solve machine learning problems such as classification, regression, and clustering.
Each layer of the MLP is composed of numerous neurons: the input layer receives features, the output layer provides final predictions, and the intermediate hidden layers are used for extracting features and performing non-linear transformations. Each neuron receives output from the previous layer, processes it through complex weighted sums and

activation functions, and produces the output for the current layer. Through iterative training, MLP can autonomously learn complex relationships between input features and make predictions on new data.
In Python, various deep learning frameworks are available to implement MLP modeling. In this thesis, the deep learning framework used is PyTorch. Developed by Facebook, PyTorch differs from other frameworks by utilizing a dynamic graph, which facilitates easier debugging and development.

Here are some advantages of MLP.

- Expressive Power: MLP have strong expressive capabilities, able to handle nonlinear issues and high-dimensional data.
- Backpropagation Training: MLP can be trained through the backpropagation algorithm to autonomously learn features and patterns.
- Multiclass and Regression Problems: MLP are capable of handling both multiclass classification and regression problems, demonstrating good generalization capabilities.
- Overfitting Prevention: Techniques like regularization and dropout can be employed to prevent overfitting in MLP.

3.3.4 Kolmogorov-Arnold Networks(KANs)

KANs is a deep learning method appropriate for handling large volumes of complex data involving nonlinear problems. [5] Below is the basic workflow for KANs

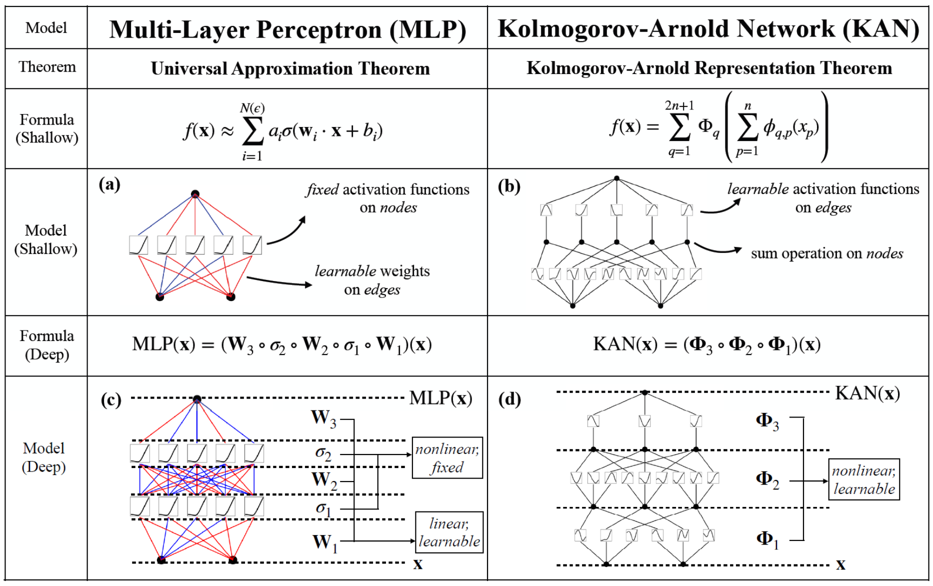
- Parameter Initialization: Initialize all parameters.
- Dynamic Update of Spline Grids: As spline functions are defined within bounded regions and activation values during training may exceed predefined intervals, KANs dynamically updates grid points upon receiving new input activations to ensure all possible activation values are covered.
- Model Training and Symbolization: KANs training is similar to conventional neural networks, with activation functions based on spline expressions. After training, nodes are automatically pruned to simplify the network structure. Once all activation functions are symbolized, training continues until the loss is minimized, indicating that the correct symbolic expressions have been found.
- Validation and Testing: Adjust model parameters on the validation set to prevent overfitting, then assess the model performance on the test set.
- Result Analysis: Analyze and interpret the model prediction results, evaluating its performance on unknown data.

Kolmogorov-Arnold Networks challenge traditional multilayer perceptron designs by moving activation functions from nodes to edges, enhancing model performance and interpretability. The core of KANs is that all weight parameters are replaced by univariate

spline functions, which adaptively adjust based on training data, offering greater flexibility and adaptiveness than fixed activation functions.
The Kolmogorov-Arnold theorem states that for any continuous function $f(x_1, \dots, x_n)$ defined on the closed interval $[0,1]^n$, there exists an integer $2n+1$ and a series of one-dimensional continuous functions $\phi_{q,p}$ and Φ_q, such that the multivariate function can be expressed as

$$f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

- The function $\phi_{q,p}$ maps individual variables x_p to real numbers.
- The function Φ_q processes the sum of the mappings by $\phi_{q,p}$.
- The $2n+1$ represents the minimum number of functions required to represent any continuous function in this form.



Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) *fixed* activation functions on *nodes*; *learnable* weights on *edges*	(b) *learnable* activation functions on *edges*; sum operation on *nodes*
Formula (Deep)	$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$
Model (Deep)	(c) MLP(**x**), $\mathbf{W}_3$, σ_2, $\mathbf{W}_2$, σ_1, $\mathbf{W}_1$, **x**; *nonlinear, fixed*; *linear, learnable*	(d) KAN(**x**), $\mathbf{\Phi}_3$, $\mathbf{\Phi}_2$, $\mathbf{\Phi}_1$, **x**; *nonlinear, learnable*

Figure 3.6. Comparison chart between MLP and KANs [5]

Unlike MLP, where fixed activation functions are placed at the nodes (neurons), KANs places learnable activation functions at the edges (weights). Thus, Kolmogorov-Arnold Networks entirely lack a linear weight matrix: each weight parameter is replaced by a learnable one dimensional spline function. Nodes in KANs simply sum input signals without performing any nonlinear processing and generally allow a smaller computational graph than MLP. Consider the example of Fig. 3.6.
To better understand KANs, let's consider the diagram on the left showing the network

layered architecture and data flow, and on the right, a key component the B-spline activation function is detailed, showing how to switch between coarse and fine grids using "grid expansion techniques."Consider the example of Fig. 3.7

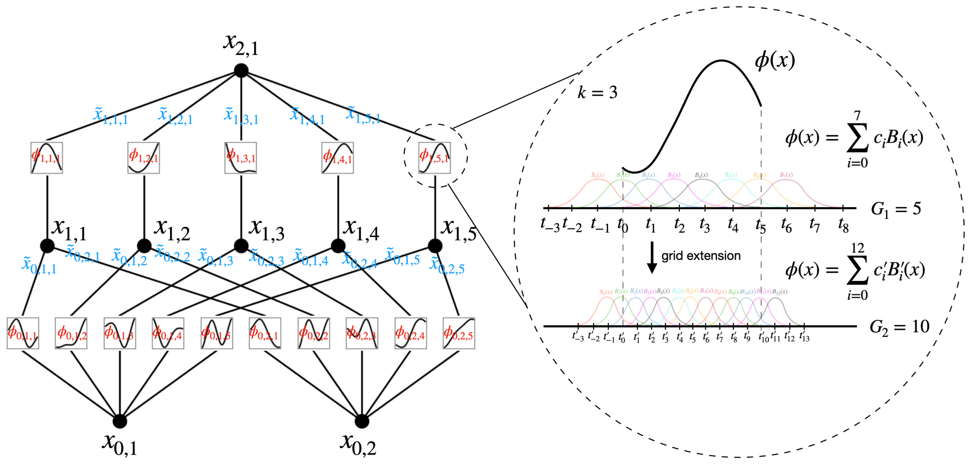


Figure 3.7. KANs network structure diagram [5]

The left diagram displays the layered architecture of a KANs. Each layer includes a set of nodes, each processing input data through a specific set of functions, outputting to the next layer. Small icons on each node indicate the type of activation function, here represented by B-spline functions.
The right diagram demonstrates an activation function $\phi(x)$, parameterized as a B-spline function. It also shows how the granularity of the function can be adjusted by changing the number of knots (also called control points) in the B-spline.
This illustration corely showcases how KANs uses B-splines as activation functions, combined with the network's multi-layer structure and dynamic adjustment of activation functions (grid expansion techniques), to handle complex, high-dimensional data. This design allows the network not only to adapt to different data resolutions but also to optimize performance by adjusting the precision of the activation functions.

The image below illustrates the training process of the KANs model. Consider the example of Fig. 3.8

Performance Advantages

- Accuracy Improvement: Compared to MLP of the same or even larger scale, KANs demonstrates higher accuracy in data fitting and partial differential equation solving tasks. Research also shows that even small-scale KANs can match or exceed the performance of large-scale MLP.

- Neural Scaling Rate: KANs exhibits a better neural scaling rate than MLP, meaning that as the number of models increases, KANs performance significantly improves.

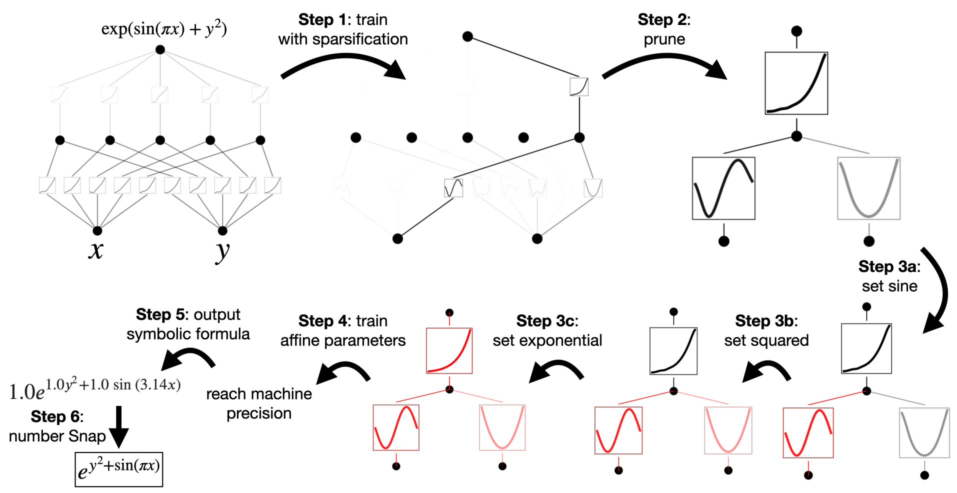

$\exp(\sin(\pi x) + y^2)$
x
y
Step 1: train with sparsification
Step 2: prune
Step 3a: set sine
Step 3b: set squared
Step 3c: set exponential
Step 4: train affine parameters
reach machine precision
Step 5: output symbolic formula
$1.0e^{1.0y^2+1.0\sin(3.14x)}$
Step 6: number Snap
$e^{y^2+\sin(\pi x)}$

Figure 3.8. KANs model training flow chart [5]

- Enhanced Interpretability: The structure of KANs allows for intuitive visualization, enabling users to actively participate in optimization and debugging, which is particularly beneficial in tasks like symbolic regression. Users can even manually specify or have the system suggest appropriate symbolic functions to represent activation functions, resulting in more readable expressions.

Due to similarities between KANs and MLP, although KANs might seem to have a higher parameter magnitude than MLP, it often achieves better generalization capability with a smaller N-value, thereby reducing parameter quantity to enhance the model interpretability and versatility. As a formidable competitor to MLP, KANs not only demonstrates enormous potential in terms of accuracy and interpretability in deep learning but also offers stronger adaptability and robustness in non-symbolic function learning processes.

Chapter 4

Thesis Setup

4.1 Hardware and Software Requirements for the Thesis

The design involves processing human trajectory data collected through capacitive sensors using four methods: LSTM, Qlattice, MLP, and KANs. To effectively execute these tasks, specific hardware and software configurations are required to support data collection, processing, and analysis.

4.1.1 Hardware Requirements

1. **Data Collection and Processing Unit**

    - Computer System: A high-performance computer is necessary, with sufficient processor speed and memory (recommended 16 GB or more) to handle and analyze data. Consider the example of Fig. 4.1
    - Storage Device: A high-speed Solid State Drive (SSD) for storing data and models.

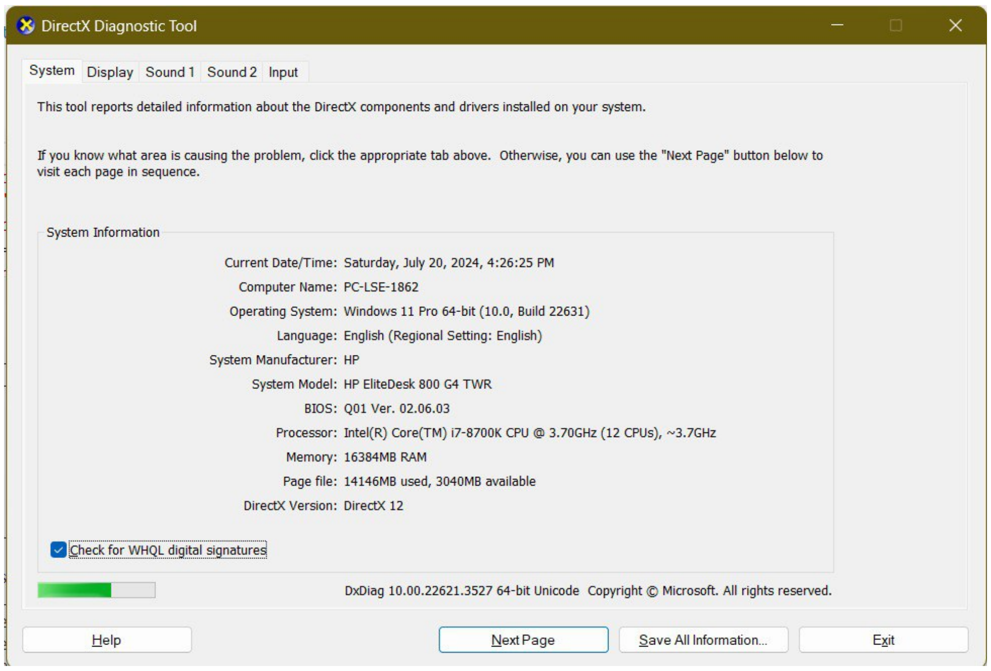
2. **Network Facilities**

    - High-Speed Internet Connection: For software updates, remote access, and potential use of cloud computing resources.
    - Internal Network: A stable Local Area Network (LAN) or Wi-Fi for the wireless transmission of sensor data.

4.1.2 Software Requirements

1. **Operating System**

    - Compatibility: Windows 10, depending on the chosen analysis software and personal preferences.



This tool reports detailed information about the DirectX components and drivers installed on your system.

If you know what area is causing the problem, click the appropriate tab above. Otherwise, you can use the "Next Page" button below to visit each page in sequence.

System Information

Current Date/Time:	Saturday, July 20, 2024, 4:26:25 PM
Computer Name:	PC-LSE-1862
Operating System:	Windows 11 Pro 64-bit (10.0, Build 22631)
Language:	English (Regional Setting: English)
System Manufacturer:	HP
System Model:	HP EliteDesk 800 G4 TWR
BIOS:	Q01 Ver. 02.06.03
Processor:	Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz (12 CPUs), ~3.7GHz
Memory:	16384MB RAM
Page file:	14146MB used, 3040MB available
DirectX Version:	DirectX 12

✓ Check for WHQL digital signatures

DxDiag 10.00.22621.3527 64-bit Unicode Copyright © Microsoft. All rights reserved.

Figure 4.1. Computer equipment information

2. **Data Analysis and Machine Learning Tools**
   - Python: A widely used programming language, supporting various data processing and machine learning libraries.
   - Scientific Computing Libraries: NumPy and Pandas for data manipulation; Matplotlib and Seaborn for data visualization.
   - Machine Learning Libraries
     - Scikit-learn: Provides extensive support for traditional machine learning algorithms.
     - TensorFlow or PyTorch: For building and training complex neural networks (such as MLP and KANs).
     - gplearn: A library for implementing genetic programming, used for symbolic regression.
   - Qlattice Software: Specialized software provided by Feynman AI, which may require integration with the Python environment through an API.
3. **Software Development Environment**
   - IDE: Visual Studio Code, offering code editing, debugging, and version control. The choice of IDE depends on personal preferences.
   - Version Control System: Such as Git, used for code management and collaboration.

4.2 Comparative Evaluation Criteria

When comparing the accuracy of LSTM, Qlattice, and MLP and KANs, it is crucial to choose appropriate evaluation criteria to fairly assess their performance in human trajectory prediction tasks. In this thesis, Mean Squared Error (MSE) is selected as the comparative evaluation standard.

Definition: Mean Squared Error is the average of the squares of the differences between the predicted values and the actual values. It is a commonly used metric for assessing the accuracy of models. The formula for MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

This metric provides a clear quantitative measure to evaluate how well each model predicts human trajectories by penalizing larger errors more significantly than smaller ones, thereby ensuring that models with lower MSE values are more precise in their predictions.

Chapter 5

Results and Discussion

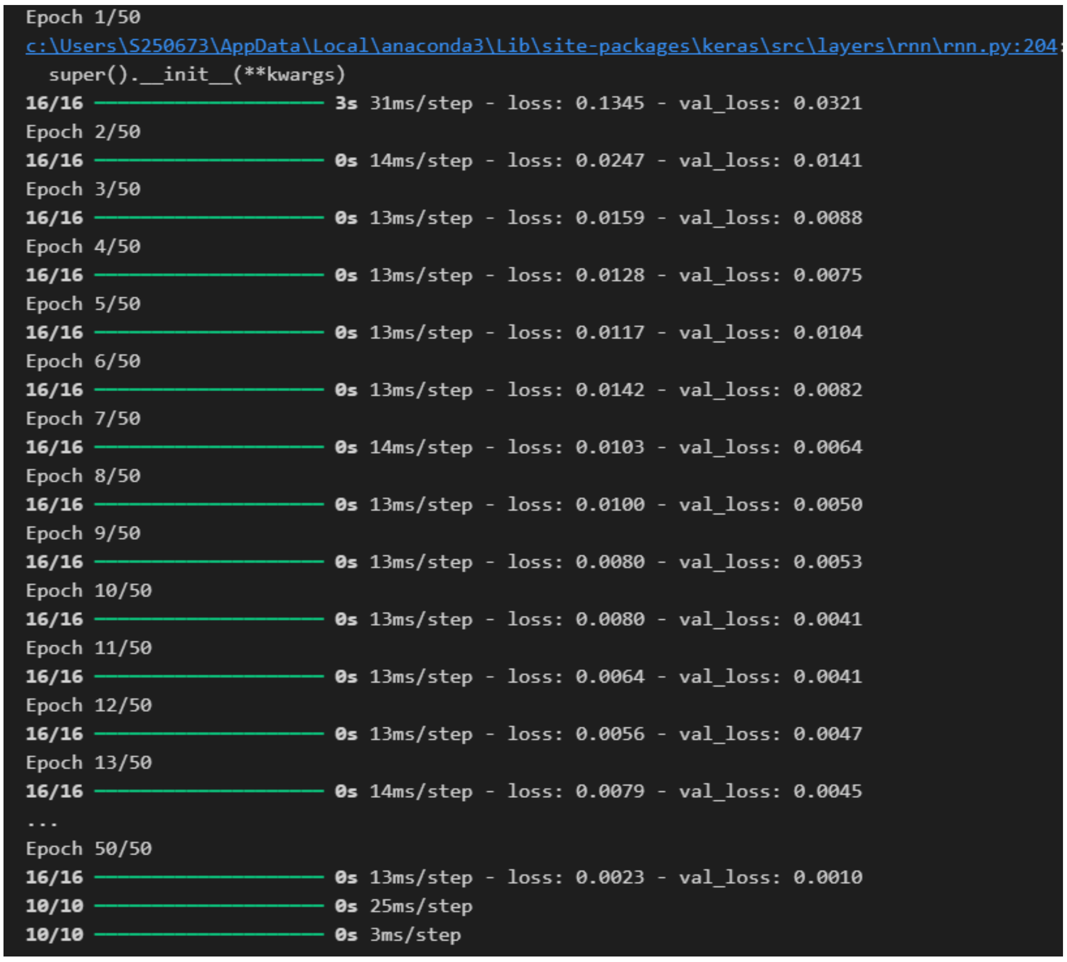
5.1 Presentation of Results

5.1.1 Long Short-Term Memory Networks (LSTM)

LSTM is a type of recurrent neural network used in deep learning, excelling at learning from data sequences. With its unique architecture that combines memory cells and gates, LSTM can effectively capture long-term dependencies within data. This capability allows it to discern complex relationships and patterns, making it crucial for tasks such as time series forecasting, natural language processing, and sequence classification.

From the process information of the training output for an LSTM model, several important insights and trends can be observed regarding the model learning performance across multiple epochs. Here's a detailed analysis. Consider the example of Fig. 5.1

1. Epoch Count and Duration
    - Each line in the log represents one epoch of training, where the model processes the entire training data.
    - The time per step and overall duration for each epoch are consistently short, indicating efficient computation, likely due to a manageable dataset size or efficient hardware.
2. Loss and Validation Loss
    - Training Loss
      This metric represents the MSE of the model on the training dataset. It's calculated as the average of the squared differences between the predicted values and actual values. The training loss decreases significantly from the first epoch (0.1345 m^2) to the last reported epoch (0.0023 m^2), which suggests that the model is learning effectively and adapting well to the training data.
    - Validation Loss
      This metric shows the MSE on the validation dataset, which is used to evaluate the model generalization ability, i.e., how well it performs on unseen data.



```
Epoch 1/50
c:\Users\S250673\AppData\Local\anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:204
  super().__init__(**kwargs)
16/16 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - loss: 0.1345 - val_loss: 0.0321
Epoch 2/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0247 - val_loss: 0.0141
Epoch 3/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0159 - val_loss: 0.0088
Epoch 4/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0128 - val_loss: 0.0075
Epoch 5/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0117 - val_loss: 0.0104
Epoch 6/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0142 - val_loss: 0.0082
Epoch 7/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0103 - val_loss: 0.0064
Epoch 8/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0100 - val_loss: 0.0050
Epoch 9/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0080 - val_loss: 0.0053
Epoch 10/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0080 - val_loss: 0.0041
Epoch 11/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0064 - val_loss: 0.0041
Epoch 12/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0056 - val_loss: 0.0047
Epoch 13/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0079 - val_loss: 0.0045
...
Epoch 50/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0023 - val_loss: 0.0010
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

Figure 5.1. Training process information of LSTM model

Similar to the training loss, the validation loss also decreases over time, starting from 0.0321 m^2 in the first epoch and reducing to 0.0010 m^2 in the last reported epoch.

3. Convergence of Model

   - Both the training and validation losses show a steady decline, which is a positive indicator of model convergence. The model does not show signs of overfitting, as the validation loss decreases in tandem with the training loss, and actually reaches lower values in some epochs.
   - The convergence rate, indicated by the difference in loss between successive epochs, stabilizes after initial larger drops, suggesting that early epochs make the most significant adjustments to the model weights.

4. Performance Considerations

- The similar and low values of training and validation losses in later epochs suggest that the model has achieved a good balance between learning the training data characteristics and generalizing to new, unseen data.
- The final epochs indicate very low MSE, demonstrating high accuracy and precision in the model predictions relative to the scale of data values (assuming typical scaled data for LSTM usage).

5. Potential Next Steps

   - If this trend continues beyond the 50 epochs, the model might be close to optimal performance for this particular architecture and dataset. However, monitoring for more epochs could be useful to ensure stability in loss reduction.
   - Experimenting with further hyperparameter tuning (e.g., learning rate adjustments, more LSTM units or layers) could potentially squeeze further improvements or efficiencies out of the model.
   - Evaluating the model on a separate test set would be crucial to truly assess its performance and ensure that it has not memorized the validation dataset.

Overall, the results are promising, showing effective learning and good generalization. The detailed decrease in loss values indicates a successful training process with potential for practical application deployment.

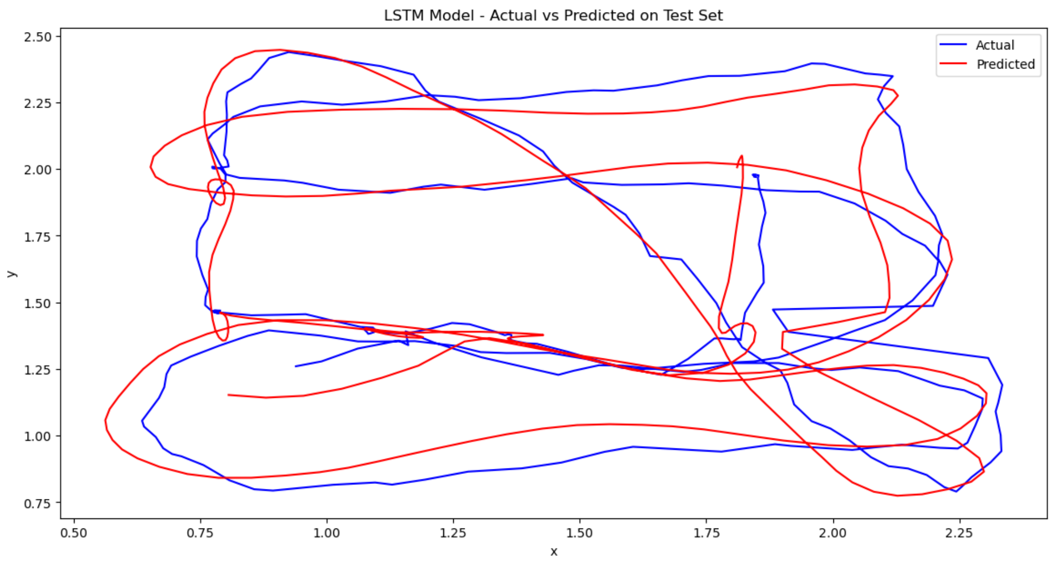


Figure 5.2. Testing trajectory of the LSTM model

Following the predictions made on the test set with the trained model, let's delve deeper into the comparison of the model-predicted trajectories against the actual trajectories. Consider the example of Fig. 5.2.

- Detailed Analysis of Long Short-Term Memory Networks (LSTM) Model Performance

  1. Trajectory Tracking Accuracy
     - Path Alignment
       The LSTM model closely follows the overall path of the actual trajectory in most parts of the dataset. This demonstrates a strong capability to model time dynamics, indicating that LSTM effectively captures the main patterns and behaviors in the trajectory data.
     - Local Deviations
       From the predictions on the test set, it is evident that there are deviations between the predicted trajectory and the actual data, particularly noticeable at sharp turns or where the trajectory direction changes abruptly. These deviations may reflect the model limited responsiveness to rapid changes or sensitivity to initial conditions.
  2. Key Performance Characteristics
     - Consistency in Open Areas
       The model performs well in parts of the trajectory that are smoother and simpler. This consistency in simpler sections suggests that LSTM can reliably handle linear or slightly curved sequences.
     - Challenges in Complex Areas
       Areas with tight loops and overlapping paths pose challenges for the LSTM. Here, the model struggles to replicate precise paths, often simplifying or smoothing out the features of the trajectory. This may indicate the model generalization behavior, where it opts for a path that minimizes overall error across the dataset rather than capturing every detail.
  3. Precision and Accuracy
     - Start and End Points
       The model accurately predicts the start and end points of the trajectory. This precision in capturing the endpoints of sequences highlights LSTM capability to effectively initialize and terminate sequences.
     - Variations in the Middle of Sequences
       Some differences in the middle of sequences indicate the model inability to dynamically adjust to sudden changes in direction or speed. These areas (especially where trajectories intersect or deviate sharply) highlight potential limitations in the model ability to learn complex dependencies or temporal resolutions.
  4. Dataset Generalization
     - Overall Fit
       The LSTM model generally fits well with the test data, indicating effective learning and generalization from the training phase. The close tracking of actual trajectories for most of the test set indicates that the model is well-trained and capable of handling various patterns and trajectory behaviors.

  - Error Distribution
    Areas with significant errors are sporadic and localized, which can be attributed to inherently challenging aspects of the data that are difficult to model, such as sudden changes in direction and overlapping trajectory segments.

- Analysis Summary

  A detailed examination of the LSTM model performance reveals its strong capability to predict complex trajectories with high precision, particularly evident in its tracking of major path trends and correct identification of sequence endpoints. While there are local deviations, especially in areas of the dataset involving complex movements and rapid changes, the overall performance of the model is commendable. These insights showcase the strengths of LSTM in modeling time series and highlight its potential utility in applications requiring precise trajectory predictions.

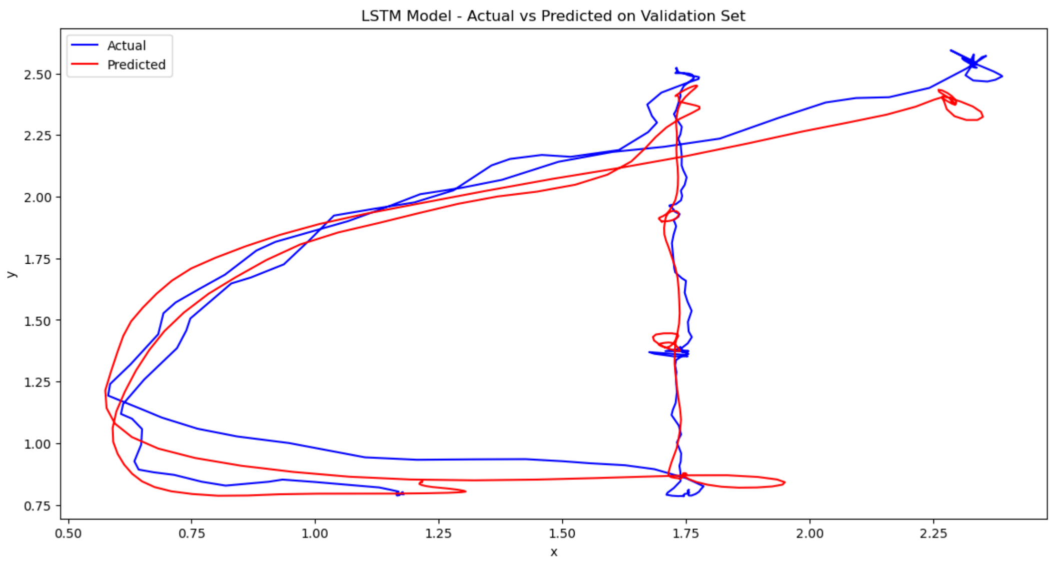


Figure 5.3. Validation trajectory of LSTM model

Based on the LSTM model performance on the validation set, we will conduct a detailed analysis of the trajectories predicted by the model, drawn in red, compared to the actual trajectories, drawn in blue. This will be contrasted with the model performance on the test set to comprehensively understand the model generalization capabilities and potential areas for improvement. Consider the example of Fig. 5.3.

- Detailed Analysis of LSTM Model Predictions on the Validation Set

  1. Overall Trajectory Alignment
     - General Consistency

Similar to the test set, the LSTM predictions generally align with the actual trajectory overall path. The model has successfully captured broad trends and movements, indicating robust learning of the sequence patterns present in the validation set.

   - Curve Following Precision
     The model closely follows the actual path on smoother and more gentle sections, demonstrating its effectiveness in learning from medium dynamic sequences.

2. Significant Deviations
   - Sharp Turns and Sudden Changes
     The model struggles with sharp turns and sudden changes in trajectory direction, with the red line separating from the blue at approximately x = 1.75 m, y = 2.25 m, and again near the top of the graph at x = 2.25 m. This reaffirms issues observed in the test set, indicating ongoing challenges in the model ability to adapt to rapid directional changes.
   - Error Concentration in Complex Areas
     Errors are more pronounced in areas where the trajectory undergoes complex maneuvers. For example, near the bottom of the graph at x = 0.75 m, y = 1.0 m and x = 1.25 m, y = 1.25 m, the model predicted path is simpler compared to the actual complex movements.
3. Comparison with Test Set Performance
   - Similar Error Patterns
     Both the validation and test set results indicate that the LSTM model struggles to accurately predict trajectories when there are overlapping paths or paths nearing a return to previous ones. This suggests potential limitations in the model memory or state management, struggling to distinguish closely occurring sequence features.
   - Model Consistency
     The consistent error patterns across both datasets suggest systemic issues in the model or data representation, possibly failing to capture all nuances required for precise predictions in complex scenarios.
4. Impact on Model Generalization
   - Good Adaptation to Overall Trends
     The LSTM model shows good fit to overall trajectory patterns, indicating that it has effectively generalized the main dynamic behaviors from the training phase to unseen data in both test and validation sets.
   - Generalization Across Datasets
     The similar performance in general trajectory tracking and specific misalignment areas on both test and validation sets supports the model robustness and reliability for general trajectory prediction tasks.

- Analysis Conclusion

The detailed comparison of the LSTM model performance on both test and validation datasets highlights its ability to model complex trajectories with high accuracy in terms of overall trends. The recurrent issues observed in predicting sharp turns and complex paths highlight inherent challenges in the LSTM architecture or training methods, which may be focal points for further research or model improvements.

5.1.2 Qlattice

The training results obtained using the Qlattice model, with the following parameters explained.

- Epoch Information
  - Epoch: Each epoch in this context represents a complete cycle through the training dataset. The model training involves trying different models or variations to optimize performance.
  - Tried Models: This number indicates how many different models or model configurations were evaluated during each epoch. It shows the complexity and the exploration capability of the Feyn library, which attempts numerous models to find the best fitting one.
  - Elapsed Time: This is a running tally of time spent as the training progresses. It provides an estimate of how long the training is taking and updates dynamically as more data about the training duration becomes available.
- Incremental Updates
  - The updates you see from epochs 1 to 10 show an increase in the number of models tried. This increment suggests that the library is either expanding its search space based on prior findings or simply processing through a predefined list of model configurations.
  - The time "est." (estimated completion time) adjusts as the system gains more insight into how long the computations are taking, providing a more accurate forecast of the total duration as the process unfolds.
- Interpreting the Output
  - Efficiency and Exploration: The increasing number of models tried indicates a thorough exploration of the model space. Feyn is likely using a method akin to evolutionary algorithms or other heuristic searches that progressively try to improve the models based on prior iterations' performance.
  - Performance Considerations: The elapsed and estimated times provide insight into the computational demands of the model training. If the times are longer than desirable, it might be necessary to adjust the model complexity, the number of epochs, or the hardware used for training.

- The Following Is the Trained Model

$$X = 0.674\cdot(0.457-0.892\cdot x_{13})\cdot\tanh(1.04\cdot x_1+0.827)+0.674\cdot(2.39-1.67\cdot x_{14})\cdot(0.0040044\cdot y_5-1.41)+1.97$$

$$Y = -0.664\cdot(0.725-0.00396\cdot x_2)\cdot(0.464\cdot y_{10}-2.52\cdot y_{14}+4.21)+2.033$$

- Understanding the Variables

$$x_{13}, x_1, x_{14}, x_2$$

  These variables represent specific features in data that have been numerically indexed. These variables depend on the context of data.

$$y_5, y_{10}, y_{14}$$

  Similarly, these are other features in the dataset.

Next, use the trained model to make predictions on the test and validation datasets, and compare the actual data with the predicted results.

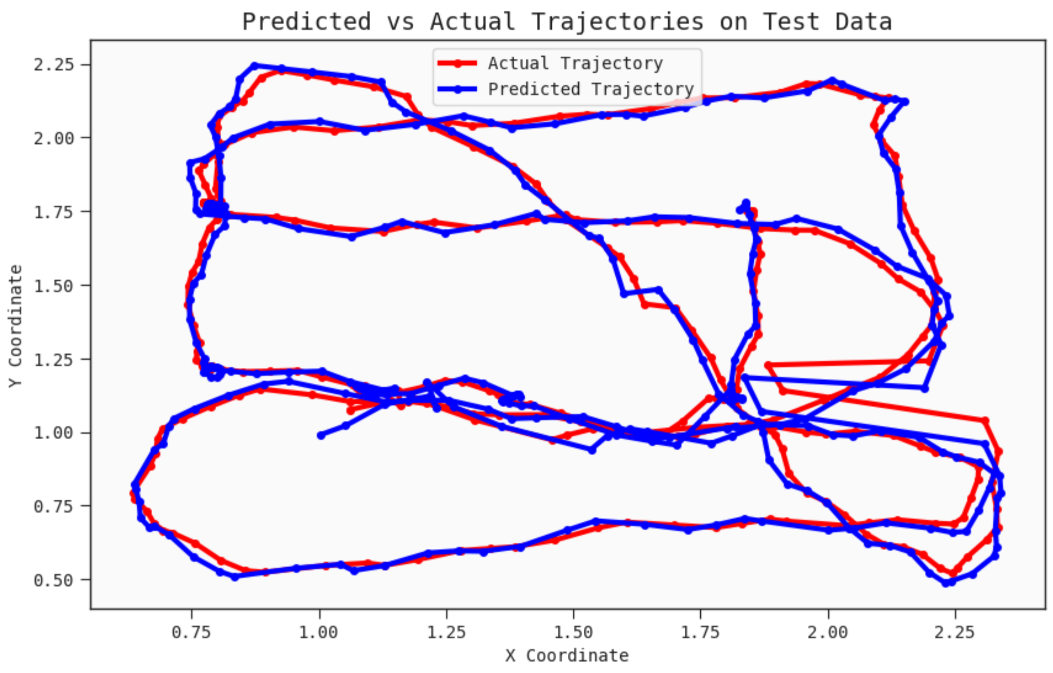


Figure 5.4. Testing trajectory of Qlattice model

We make a comparison between the predicted trajectory and the actual trajectory based on test data evaluated by the Qlattice model. The diagram illustrates two trajectories plotted on a coordinate system, where the x-axis represents the X coordinates, and the y-axis represents the Y coordinates. Consider the example of Fig. 5.4

- Plot Description
  - X-axis and Y-axis: The chart plots trajectories in a two-dimensional space, where "X coordinate" represents the horizontal position, and "Y coordinate" represents the vertical position.
  - Red Line (Actual Trajectory): This line represents the actual path or motion recorded in the test data. It shows the movement trajectory of a person in indoor space during the observation period.
  - Blue Line (Predicted Trajectory): This line displays the trajectory predicted by the QLattice model based on the input data from the given test set. It represents the model best guess of the motion based on its training.
- Plot Analysis
  - Overall Fit: The predicted trajectory (blue line) generally follows the shape and path of the actual trajectory (red line), indicating that the model has a reasonable understanding of the dynamics and patterns in the data.
  - Specific Observations
    * Tight Matching: In several segments of the trajectory, the predicted path closely matches the actual path, indicating strong performance by the model in these areas.
    * Deviations: There are significant deviations between the predicted trajectory and the actual path in some areas. These deviations are key focus areas, as they represent inaccuracies in the model predictions.
- Insights and Further Analysis
  - Model Strengths

    The model captures the overall trend and major turns or transitions in the trajectory well.
  - Model Weaknesses
    * Precision and Accuracy: The deviations suggest that the model may struggle with certain conditions or specific types of motion. This could be due to a variety of factors, such as insufficient training data covering these conditions, inherent limitations of the model complexity, or noise and outliers in the training data.
    * Overfitting or Underfitting: If the model is too simple (underfitting), it may not capture complex patterns adequately. Conversely, if the model is too complex (overfitting), it may react too strongly to noise in the training data and not generalize well.

We make a comparison between the predicted and actual trajectories based on validation data evaluated by the QLattice model. The diagram shows two trajectories plotted

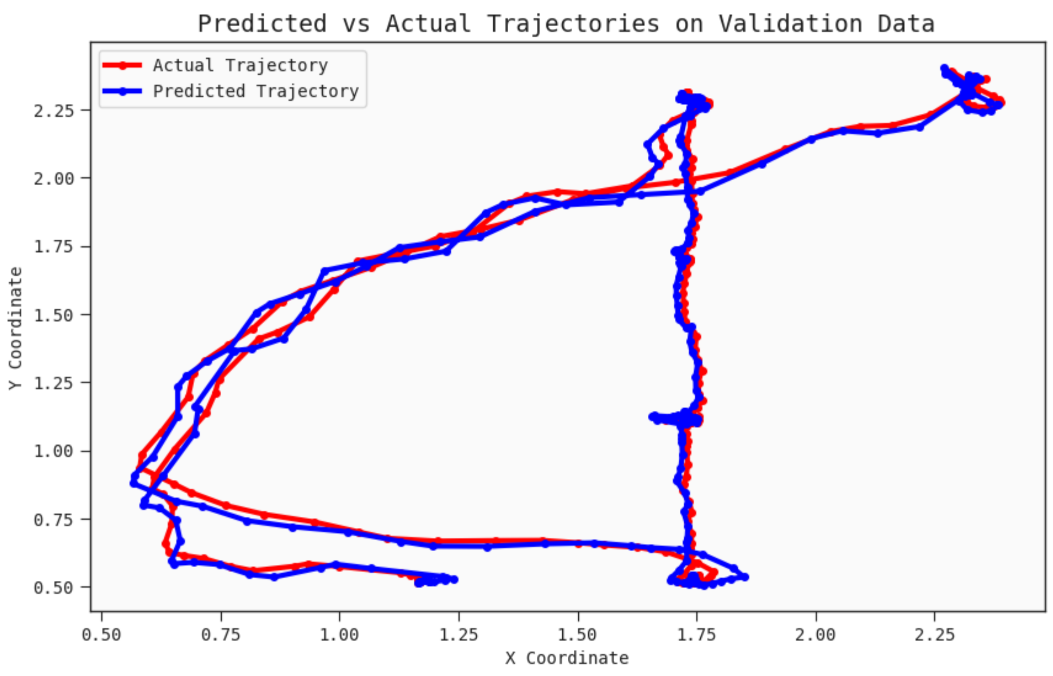


Figure 5.5. Validation trajectory of Qlattice model

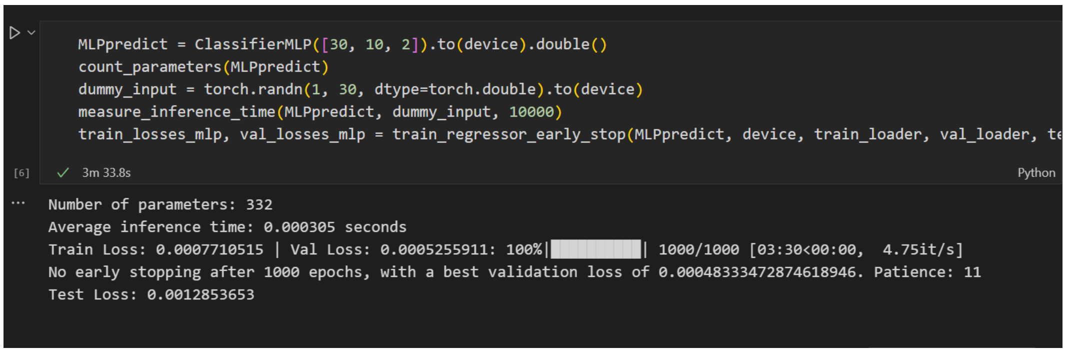
on a coordinate system, where the x-axis represents the X coordinates and the y-axis represents the Y coordinates. Consider the example of Fig. 5.5

- Plot Description
  - Red Line (Actual Trajectory): This line represents the actual movement or behavior recorded in the validation dataset. It shows the true path walked by a person indoors over a period of time.
  - Blue Line (Predicted Trajectory): This line represents the trajectory predicted by the trained QLattice model based on the learning from the training dataset. It represents the model best guess of the walking path of indoor personnel.
- Plot Analysis
  - General Observations
    * Overall Fit: The predicted trajectory is very close to the actual trajectory in several segments, indicating that the model effectively captures the underlying patterns and dynamics of the dataset.
    * Deviations: There are noticeable differences between the predicted and actual trajectories at certain points, especially in areas where the trajectory makes sharp turns or changes direction.

- Detailed Observations and Insights
  - Alignment and path following: The model appears to follow the general path well but lacks accuracy on a finer scale, particularly during rapid changes.
  - Model strengths: The model performs well in smooth or linear segments, accurately capturing the general direction and length of movement.
  - Model weaknesses: Predictions of sharp turns and complex movement patterns are not very accurate. This may be due to the model limited ability to infer complex dynamics from the features provided or possibly due to overfitting simpler patterns seen in the training data.

5.1.3 Multilayer Perceptrons (MLP) and Kolmogorov-Arnold Networks(KANs)

The following is the model training and data prediction process of MLP and KANs. In order to better compare the two models, the final predicted trajectory graphs of the two models are placed in the same figure.

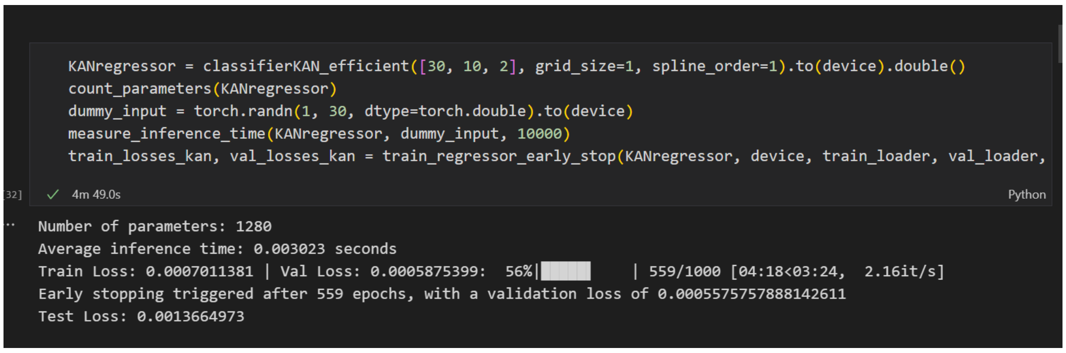


```
MLPpredict = ClassifierMLP([30, 10, 2]).to(device).double()
count_parameters(MLPpredict)
dummy_input = torch.randn(1, 30, dtype=torch.double).to(device)
measure_inference_time(MLPpredict, dummy_input, 10000)
train_losses_mlp, val_losses_mlp = train_regressor_early_stop(MLPpredict, device, train_loader, val_loader, te
```

```
[6]  ✓ 3m 33.8s                                                                                    Python
```

```
Number of parameters: 332
Average inference time: 0.000305 seconds
Train Loss: 0.0007710515 | Val Loss: 0.0005255911: 100%|██████████| 1000/1000 [03:30<00:00,  4.75it/s]
No early stopping after 1000 epochs, with a best validation loss of 0.00048333472874618946. Patience: 11
Test Loss: 0.0012853653
```

Figure 5.6. The training process of MLP



```
KANregressor = classifierKAN_efficient([30, 10, 2], grid_size=1, spline_order=1).to(device).double()
count_parameters(KANregressor)
dummy_input = torch.randn(1, 30, dtype=torch.double).to(device)
measure_inference_time(KANregressor, dummy_input, 10000)
train_losses_kan, val_losses_kan = train_regressor_early_stop(KANregressor, device, train_loader, val_loader,
```

```
32]  ✓ 4m 49.0s                                                                                    Python
```

```
Number of parameters: 1280
Average inference time: 0.003023 seconds
Train Loss: 0.0007011381 | Val Loss: 0.0005875399:  56%|█████     | 559/1000 [04:18<03:24,  2.16it/s]
Early stopping triggered after 559 epochs, with a validation loss of 0.0005575757888142611
Test Loss: 0.0013664973
```

Figure 5.7. The training process of KANs

Consider the training processes for both models, which are implemented using custom neural networks in PyTorch. For activation functions, MLP employs the ReLU activation function, while the essence of KANs is that the activation function is learned during training. For loss functions, MSE is chosen as the criterion for both models. Here are the detailed training outcomes for KANs and MLP. Consider the example of Fig. 5.6 and Consider the example of Fig. 5.7

The formula for the total number of parameters of MLP is:

$$P = (n_{\text{input}} \times n_{\text{hidden}}) + (n_{\text{hidden}} \times n_{\text{output}}) + n_{\text{hidden}} + n_{\text{output}}$$

$\mathrm{n}_{\text{input}}$: Number of neurons in the input layer
$\mathrm{n}_{\text{hidden}}$: Number of neurons in the hidden layer
$\mathrm{n}_{\text{output}}$: Number of neurons in the output layer

In this study, multiple experiments were conducted to set different parameters for both MLP and KANs models. When the number of parameters in the KANs model was equal to that of the MLP model, the KANs model demonstrated significantly poorer fitting capability and trajectory prediction accuracy compared to the MLP. Therefore, in the following experiments, the number of parameters in the KANs model was increased beyond that of the MLP. Under these conditions, the difference in validation loss between the two models was negligible. Consequently, the focus shifted to comparing the accuracy of trajectory predictions.

1. **Kolmogorov-Arnold Networks(KANs)**

    - Number of Parameters (1,280): The model has 1,280 trainable parameters, indicating a relatively small model. This is generally beneficial for quick training and inference times but can sometimes restrict the capability to capture complex data patterns.
    - Average Inference Time (0.00302 ms): The inference time is quite fast, approximately 0.0026 ms per sample, making it efficient for applications requiring rapid responses.
    - Training and Validation Losses
        - Train Loss: 0.000701 m^2
        - Val Loss at Early Stopping: 0.000557 m^2
        - These losses are quite low, suggesting good model performance on both training and validation datasets. The slightly higher validation loss, while still improving, indicates potential for further training benefits, provided overfitting is controlled.
    - Early Stopping: Early stopping was triggered after 559 epochs due to a lack of improvement in validation loss. This strategy helps prevent overfitting and unnecessary computations. Adjusting the patience parameter might allow for further improvements without leading to overfitting.
    - Test Loss (0.00137 m^2): The test loss is higher than both training and validation losses, which is expected as the test set should ideally contain unseen

examples, presenting a tougher challenge. However, the increase in loss suggests some degree of overfitting or a lack of generalization to new data compared to training and validation sets.

2. **Multilayer Perceptrons (MLP)**
    - Number of parameters (332): With 332 parameters, the model is compact, aiding in maintaining low training and inference times but may sometimes limit learning complex data patterns.
    - Average inference time (0.000305 ms): The inference time is extremely fast, about 0.000305 ms per sample, excellent for applications requiring real-time predictions.
    - Training Process
        - Train Loss: 0.000771 m^2 . This is relatively low, indicating effective learning from the training dataset.
        - Validation Loss: 0.000483 m^2 . The best validation loss being lower than the training loss after 1000 epochs suggests excellent model performance without overfitting.
        - The training reached 1000 epochs without triggering early stopping, set with a patience of 11 epochs, implying that the model continued to improve or at least did not degrade significantly over many epochs.
    - Test Loss (0.00128 m^2): The test loss is higher than both the training and validation losses. While an increase is typical since it assesses the model on unseen data, the difference indicates potential overfitting or a need for better tuning to generalize beyond the training and validation sets.

Even with the number of parameters in the KANs model being significantly greater than that of the MLP, the accuracy of trajectory prediction remained higher for the MLP.

Consider the results of predictions made by two different machine learning models,MLP and KANs, compared against actual data (ground truth). Here is a detailed explanation and analysis. Consider the example of Fig. 5.8:

- Axis Explanation
- Curve Explanation: Blue Curve (with circular markers) represents the actual test set data (ground truth), which is the true indoor human walking trajectory captured in the test set. Orange Curve (with X shaped markers) represents the predictions made by the KANs model, which are the results predicted from the trained KANs model using the test set data. Green Curve (with + shaped markers) represents the predictions made by the MLP model, which are the results predicted from the trained MLP model using the test set data.
- Analysis
    - Prediction Accuracy: Overall, the MLP model (green) aligns well with the actual data, especially in the left half of the graph. This suggests that for some

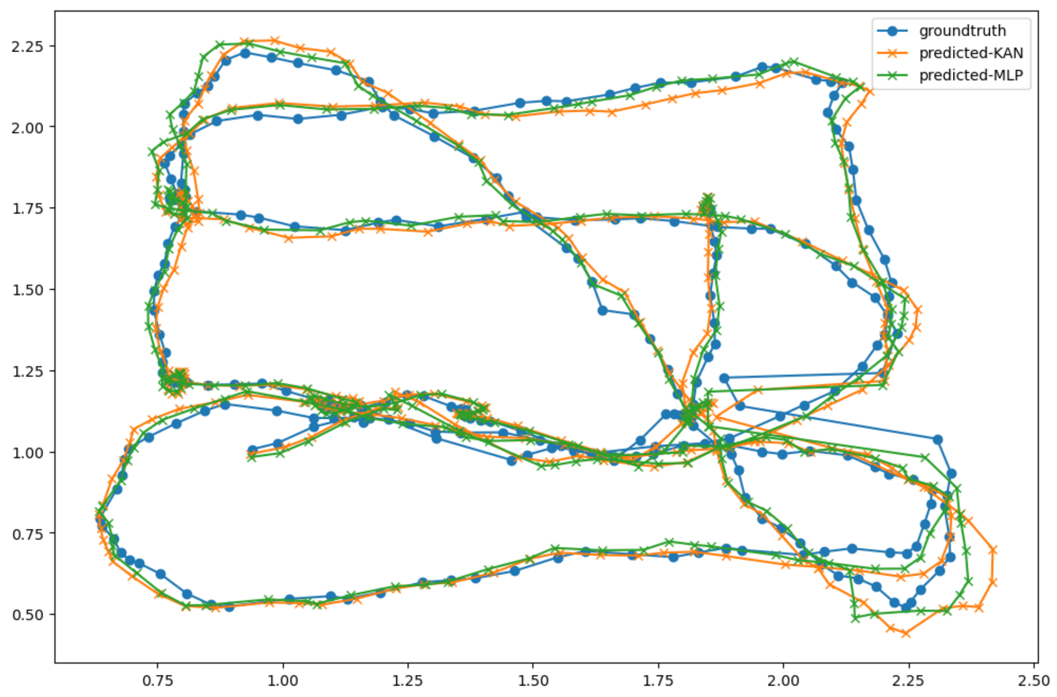


Figure 5.8. Test set trajectory prediction for MLP and KANs

of the test set data, the MLP model may provide more accurate predictions. The KANs model (orange) shows significant deviations from the actual data in certain areas (such as the right half of the graph), which may indicate poor predictive performance in these regions.

- Model Stability: The MLP model appears to be more stable across most areas, especially where there are significant data variations. This can be observed from the smoothness of the curve and its ability to follow the actual data. The KANs model shows greater fluctuations in areas with sharp data changes, which might be due to the model sensitivity to noise or variability in the data.
- Model Suitability: Based on this graph, the MLP model may be more suitable for handling this type of data, especially in applications where high accuracy and stability are required. The KANs model may need further tuning or optimization to improve its predictive accuracy and stability.

Consider the comparative results of predictions made by two machine learning models MLP and KANs against the actual data from a validation set. Below is a detailed analysis of the content and performance of these models. Consider the example of Fig. 5.9

- Axis Explanation: Horizontal Axis (X-axis) and Vertical Axis (Y-axis) represent spatial coordinates within a 3 m x 3 m square.

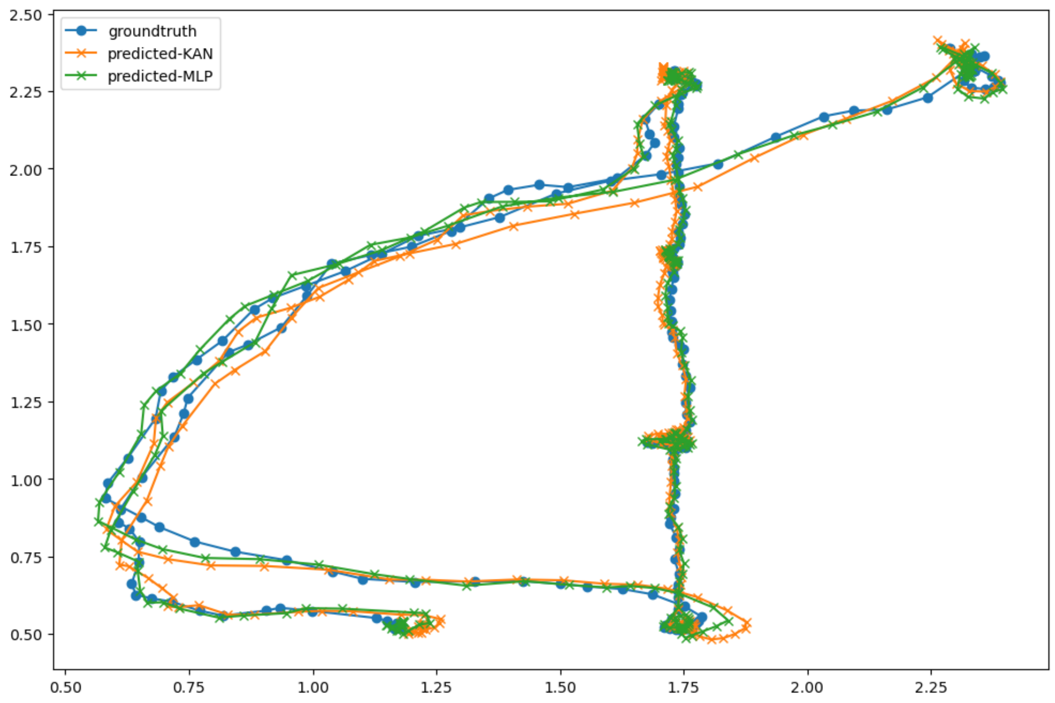


Figure 5.9. Validation set trajectory prediction for MLP and KANs

- Curve Explanation: Blue Curve (with circular markers) represents the ground truth from the validation set, which is the actual trajectory of indoor human movement. Orange Curve (with "X" shaped markers) represents the predictions made by the KANs model, which are derived from the trained model applied to the validation set data. Green Curve (with "+" shaped markers) represents the predictions made by the MLP model, which are derived from the trained model applied to the validation set data.
- Analysis
  - Prediction Accuracy: blue curve shows the true path or distribution of the validation data, while the green and orange curves demonstrate the predictive capabilities of the two models. In most areas, both the MLP and KANs models approximate the true data path well, but there are noticeable deviations at some critical nodes, particularly at the inflection points and extremes of the curves.
  - Comparative Model Performance: The MLP model (green) appears to closely follow the actual data, exhibiting higher accuracy and lower deviations overall. The KANs model (orange) shows greater deviations in certain areas, especially in the middle and top complex structures of the graph.

  - Generalization and Stability: The overall trend indicates that the MLP model exhibits better stability and generalization capabilities, likely due to its consistent performance across a variety of data points. The KANs model may require further adjustments, especially in areas with sharp changes or strong nonlinearities in the data.

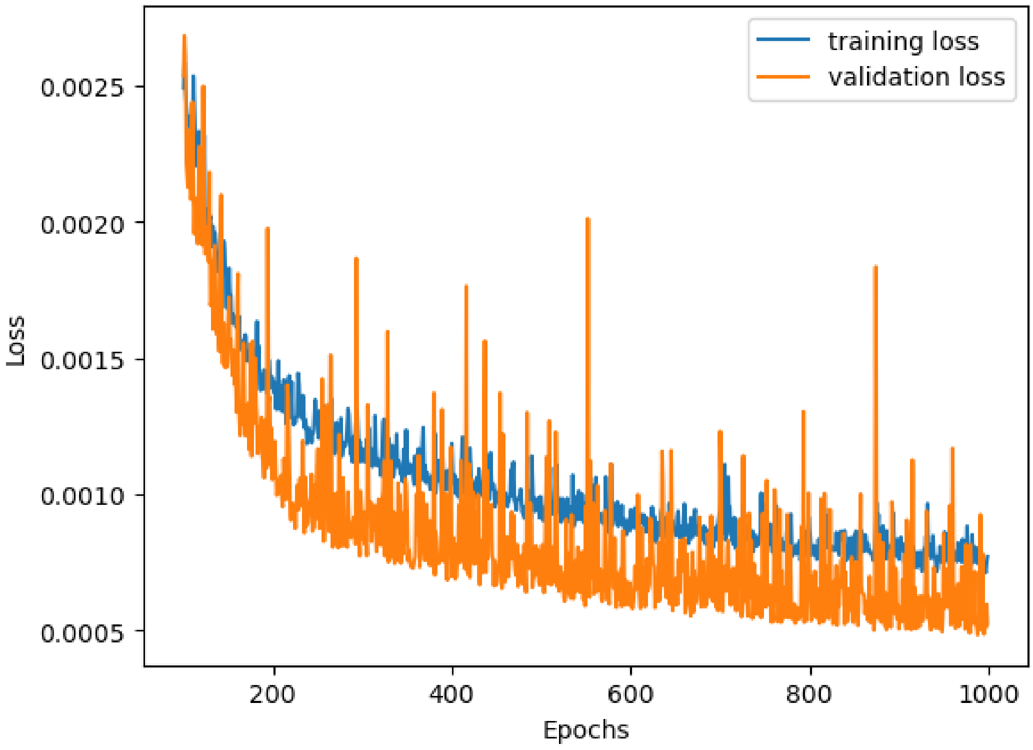


Figure 5.10. Training losses and validation losses of MLP

- Graph analysis: The graph shows two curves, representing the training loss (orange) and the validation loss (blue) across epochs. Consider the example of Fig. 5.10
  - Trends and Patterns
    - Training Loss: This curve displays a general downward trend, indicative of the model ability to learn and improve from the training data. There are minor fluctuations, but the overall decrease is consistent, which is expected behavior as the model optimizes its weights to minimize the error on the training set.
    - Validation Loss: While it tracks closely with the training loss and also shows a downward trend, it exhibits several sharp spikes. These spikes

could suggest episodes of overfitting where the model learns specific patterns or noise in the training data that do not generalize well to new, unseen data represented by the validation set.

- Loss Values
    - The initial sharp decline in both training and validation loss suggests that the most significant learning occurs in the early epochs, which is typical as the model corrects major inaccuracies in its initial random weights.
    - The continuation of training shows diminishing returns on loss reduction, a common phenomenon as the model begins to converge to a minimum in the loss landscape.
- Spikes in Validation Loss: The sudden increases in validation loss at certain points (notably around epochs 200, 400, 600, 800) could be a result of several factors, such as an inadequate learning rate (perhaps too high during these periods), mini-batch selection during training, or other stochastic elements in the training process.

• Model Metrics

- Number of Parameters: The model is relatively small with 332 parameters, which helps in faster computation but could limit the complexity of functions it can learn.
- Average Inference Time: Extremely quick at 0.000305 ms, making the model very efficient during the deployment phase, especially in environments where decision speed is crucial.
- Loss Metrics
    - Training Loss at 1000 Epochs: 0.000771 m^2 , indicating the model effectiveness on training data.
    - Validation Loss at 1000 Epochs: 0.000526 m^2 , slightly higher than the best recorded validation loss of 0.000483 m^2 but still indicative of good generalization.
    - Test Loss: 0.001285 m^2 , which is higher than both training and validation losses, suggesting some degree of overfitting or the presence of previously unencountered patterns or noise in the test set.

• Graph Analysis

The graph shows training loss (orange) and validation loss (blue) trends over 559 epochs. Consider the example of Fig. 5.11

- Trends and Patterns:
    - Training Loss: This curve starts high and exhibits a rapid decrease initially, reflecting quick learning from the training data. The curve then shows a more gradual decline, indicating a slower rate of improvement as the model starts to converge.

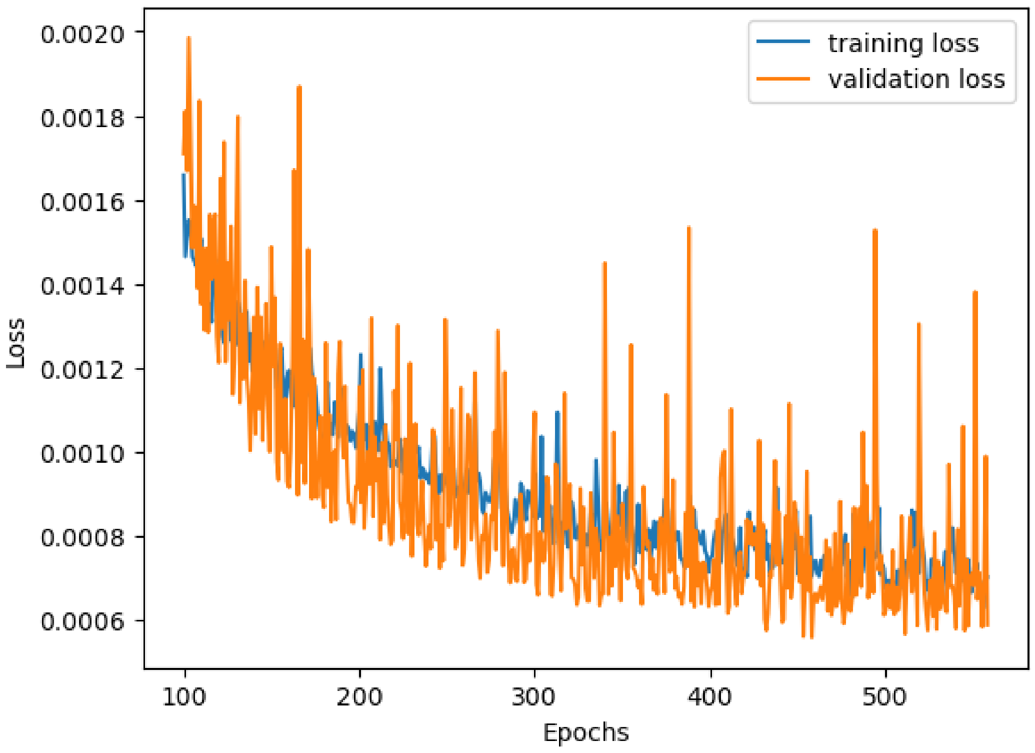


Figure 5.11. Training losses and validation losse of KANs

  * Validation Loss: Mirroring the training loss trend, the validation loss decreases significantly at the beginning but displays sharper fluctuations throughout the training process. Notably, the validation loss has more pronounced spikes than the training loss, suggesting that the model may occasionally overfit to the training data or react sensitively to certain batches of data.

- Spikes and Dips: The graph displays significant spikes in validation loss at several points, which are higher than corresponding spikes in training loss. These deviations suggest the model occasional struggle to generalize from the training data to the validation data, potentially due to overfitting.

• Model Metrics and Performance

  - Number of Parameters: With 1280 parameters, the model is larger than the previously discussed MLP model, likely having more complexity and capacity to learn detailed features. Although the parameters of KANs are larger than those of MLP in this thesis, but KANs have different ways of calculating the parameters.

    - Average Inference Time: At 3.02 ms, the inference time is longer than the previous model but still within a range suitable for many real-time applications.
    - Loss Metrics
        * Training Loss at Last Epoch: 0.000701 m^2 , reflecting the model capability to fit the training data relatively well.
        * Best Validation Loss: 0.000558 m^2 , the trigger for early stopping, indicating the lowest loss achieved on validation data before overfitting became more pronounced.
        * Test Loss: 0.00136 m^2 , higher than both training and validation losses, echoing the potential overfitting problem where the model does not perform as well on unseen test data.

- Early Stopping

  Implementation and Impact: Early stopping was employed after 559 epochs when the validation loss did not improve beyond 0.000558 m^2 . This mechanism helps prevent further overfitting by halting the training when the validation performance ceases to improve, preserving the model state that generalizes best to unseen data.

5.2 Applicability of Indoor Trajectory Analysis Results

After discussing in detail their individual characteristics, advantages, limitations, and optimal application scenarios, here we delve deeper into analyzing and comparing the suitability of MLP, KANs, Qlattice, and LSTM for trajectory prediction tasks in the field of machine learning.

- MLP
    - High Precision Requirements: MLP is well-suited for scenarios that require high precision and complex data handling, such as robotic navigation and advanced indoor positioning systems, where it can accurately predict complex paths and dynamic changes.
    - Resource-Rich Environments: Given that MLP requires substantial computational resources and data, it is more applicable in environments that can provide these resources, such as research facilities with robust computing capabilities or commercial applications.

- KANs
    - Applications Requiring High Real-Time Performance: KANs has advantages in speed and simplicity, making it suitable for applications that need quick responses but can tolerate relatively higher errors, such as real-time crowd monitoring systems and preliminary event response systems.
    - Large-Scale Monitoring: KANs simplicity and speed make it ideal for extensive area monitoring and trajectory prediction, where overly detailed trajectory specifics are not necessary.

- Qlattice
  - Moderately Complex Environments: Qlattice performs well in environments with moderate complexity, suitable for personnel tracking in settings like malls, schools, or hospitals.
  - Highly Adaptable Scenarios: Due to Qlattice good adaptability to environments, it can be applied in frequently changing settings, such as convention centers or large scale event venues.
- LSTM
  - Temporal Data Analysis: LSTM excels in handling sequential and temporal data, making it highly suitable for applications involving time-series trajectory predictions, such as tracking the movement of people over time in indoor spaces.
  - Complex Sequence Prediction: LSTM ability to remember long-term dependencies makes it ideal for precise monitoring in dynamic environments where understanding the sequence of events is crucial, such as in advanced security systems and detailed activity recognition.
  - Trajectory Smoothing: Although LSTM may not provide the most accurate real-time predictions, it can produce smoother trajectories within an acceptable error range, making it valuable for applications where trajectory smoothness is important.
- Overall Comparison
  - MLP: Offers high-precision trajectory predictions but requires extensive data and computational resources, making it suitable for scenarios where computing resources are abundant, and high accuracy is crucial.
  - KANs: More suited for fast and extensive applications, sacrificing some accuracy for speed and simplicity. Compared to MLP, KANs perform poorly both in terms of parameter complexity and trajectory prediction accuracy.
  - Qlattice: Strikes a balance between accuracy and adaptability, making it suitable for environments with medium complexity and frequent changes. It performs well in trajectory prediction accuracy.
  - LSTM: Excels in sequential and temporal data analysis, making it highly effective for dynamic environments that require understanding long-term dependencies. Although it may not provide the most accurate real-time predictions, it can produce smoother trajectories within an acceptable error range.

In the context of indoor human trajectory prediction, LSTM, while not the most accurate in real-time predictions, provides smoother trajectories within acceptable errors. Both MLP and Qlattice show good accuracy in trajectory prediction, with MLP performing particularly well in high-precision tasks. Conversely, KANs, despite their simplicity and speed, do not perform well in terms of parameter complexity and trajectory prediction accuracy compared to MLP.

In this thesis, we conducted a simple comparison of four models LSTM, Qlattice, MLP, and KANs based on their performance in solving the trajectory regression problem. The metrics considered were accuracy, inference time, model size, and explainability, each evaluated using three levels.5.1

	LSTM	Qlattice	MLP	KAN
Accuracy	Average	Good	Good	Average
Inference time	Moderate	Short	Long	Moderate
Size	Medium	Medium	Large	Small
Explainability	Average	Good	Poor	Good

Table 5.1. Comparison of different models

Chapter 6

Conclusion and Future Work

- Conclusion This thesis has systematically explored the application of four different machine learning techniques LSTM, Qlattice, MLP and KANs in predicting indoor human trajectories using capacitive sensor data. Each method was evaluated on its accuracy, computational efficiency, and interpretability.

  - LSTM demonstrated excellent capability in handling sequential and temporal data, making it suitable for dynamic environments requiring long-term dependencies. However, its real time prediction accuracy was lower, though it produced smoother trajectories.
  - Qlattice balanced accuracy and adaptability, performing well in medium complexity environments with frequent changes. It was found to be effective in moderately complex trajectory predictions but less so in highly dynamic settings.
  - MLP provided high precision trajectory predictions, excelling in environments where computational resources were abundant and high accuracy was crucial. Its requirement for extensive data and computational resources, however, was a limiting factor.
  - KANs offered simplicity and speed, making them ideal for applications requiring quick responses and large scale monitoring, although their accuracy was lower compared to other models.

  Overall, the thesis has shown that while no single model is superior in all aspects, each has distinct advantages and limitations depending on the specific application scenario. This work contributes valuable insights into the strengths and weaknesses of different machine learning approaches in indoor human trajectory prediction.

- Future Work

  - Data Collection Expansion: Future studies could incorporate additional sensor types, such as visual or thermal sensors, to enrich the dataset. Combining these with capacitive sensors could improve model robustness and accuracy by providing more comprehensive data.

– Algorithmic Enhancements: There is potential for enhancing the algorithms used in this study. Techniques such as automatic feature engineering and the integration of more advanced machine learning methods could better handle high dimensional data and improve prediction accuracy.
– Real-World Testing: While this study was conducted in a controlled environment, future research should validate the models in various real-world indoor settings, such as hospitals, shopping malls, and office spaces. This would test the models' effectiveness under dynamic conditions and provide more practical insights.
– Ethical and Privacy Considerations: As the technology for tracking human movement indoors advances, it is crucial to address ethical implications and privacy concerns. Future research should focus on developing methods that ensure data collection and analysis are conducted in a manner that safeguards individual privacy and complies with ethical standards.
– Hybrid Models: Exploring the development of hybrid models that leverage the strengths of multiple machine learning techniques could yield better performance. For instance, combining the temporal analysis capability of LSTM with the interpretability of Qlattice might offer a more balanced approach.
– Scalability and Generalization: Further research should investigate the scalability of these models in larger and more complex environments. Additionally, efforts should be made to generalize the results to different types of sensors and spatial environments, ensuring the models' applicability across diverse settings.

In conclusion, the research has laid a solid foundation for the advanced analysis of indoor human trajectories using capacitive sensors. Future advancements in this field could lead to significant improvements in automated environmental control, security monitoring, and resource management in various indoor settings.

Bibliography

[1] abzu ai. Qlattice-clinical-omics: Machine learning framework for clinical omics data analysis. https://github.com/abzu-ai/QLattice-clinical-omics, 2024. Accessed: date-of-access.

[2] Rebeen Jaff. What is lstm? introduction to long short-term memory. https://medium.com/@rebeen.jaff/what-is-lstm-introduction-to-long-short-term-memory-66bd3855b9ce, 2019. Accessed: yyyy-mm-dd.

[3] rcassani. mlp-example: Example implementation of a multilayer perceptron. https://github.com/rcassani/mlp-example, 2024. Accessed: date-of-access.

[4] Osama Bin Tariq, Mihai Teodor Lazarescu, and Luciano Lavagno. Neural networks for indoor human activity reconstructions. *IEEE Sensors Journal*, 20(22):13571–13584, 2020.

[5] Kind Xiaoming. Pykan: Simple kanban board tool. https://github.com/KindXiaoming/pykan, 2024. Accessed: date-of-access.